Advanced Machine Learning for 6G Networks

by **Hoai Nam Chu**

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of

A/Prof. Diep N. Nguyen

A/Prof. Hoang Dinh

Prof. Eryk Dutkiewicz

School of Electrical and Data Engineering

Faculty of Engineering and IT

University of Technology Sydney

February 22, 2024

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, Hoai Nam Chu, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the 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:

Date: February 22, 2024

ABSTRACT

Advanced Machine Learning for

6G Networks

by

Hoai Nam Chu

Beyond 5G and 6G communications are foreseen to transform the world, connecting not only people but also vehicles, wearables, devices, sensors, and even physical and digital worlds. To achieve that, 6G systems are expected to employ various disruptive technologies (e.g., non-terrestrial networks (NTNs), mmWave communications, pervasive artificial intelligence, and ambient backscatter communications) to enable/support new use cases, e.g., autonomous cyber-physical systems and Metaverse/holographic teleportation. Thus, this thesis aims to leverage the latest advances in machine learning (ML) to address different problems facing 6G systems. We first envision that UAVs will play a critical role in 6G and NTNs, e.g., flying data collectors. To tackle the uncertainty in the data collection process and the UAV's energy capacity limitation, we propose an innovative deep reinforcement transfer learning approach to control the UAV's speed and energy replenishment process and allow UAVs to "share" and "transfer" learning knowledge, thus reducing learning time and improving learning quality significantly.

6G is also envisioned as ubiquitous sensors thanks to the Integrated Communications and Sensing (ICAS) technology, e.g., for flood sensing/warning or in autonomous vehicles (Avs). Optimizing the waveform structure for ICAS applications to AVs is one of the most challenging tasks due to the strong influences between sensing and data communication functions under dynamic environments. Therefore, we develop a novel framework that intelligently and adaptively optimize its waveform structure to maximize sensing and data communication performance.

Another key application/service of 6G is to enable the seamless deployment and operation of Metaverse. Building and maintaining the Metaverse not only demand enormous resources but also need to address the dynamic, uncertain, and real-time resource demands. Thus, we develop a novel ML-based framework that offers a highly effective and comprehensive solution for managing various resource types for Metaverse by leveraging the similarities among applications.

Security is always one of the top concerns in wireless communications, especially for 6G connected by a massive number of heterogeneous devices. We design a lightweight framework leveraging ambient backscatter communications and deep meta-learning to counter eavesdropping attacks, effectively decode weak backscattered signals without requiring perfect information, and quickly adapt to new environments with very limited knowledge.

The above results demonstrate the great potential of advanced machine learning in addressing the emerging issues of 6G and enabling new applications/services. As future works, one may look into the applications of Generative AI to 6G and how to design 6G systems to enable Generative AI as a service.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisors, A/Prof. Diep N. Nguyen, A/Prof. Hoang Dinh, and Prof. Eryk Dutkiewicz, for their unwavering support, guidance, and encouragement. Without them, completing this dissertation would have been impossible. Throughout my studies, they not only provided valuable guidance for pursuing impactful research but also offered invaluable advice for my career. I consider myself truly privileged and fortunate to have had the opportunity to be supervised by them.

I would like to thank all my colleagues and friends at the University of Technology Sydney for their support, discussion, and friendship. Additionally, I extend my thanks to the SEDE admin team for efficiently handling all the paperwork and forms during my PhD study.

I would like to thank the Vingroup Scholarship Program, University of Technology Sydney, and the University of Transport and Communications for providing me with numerous opportunities and invaluable financial support throughout these past years.

I would like to express my heartfelt gratitude to my family and friends for their endless love and unwavering support, which gives me the strength to overcome life's difficulties. I am especially grateful to my beloved wife, Thao Pham, and our wonderful son, Viet Chu (aka. Beau). My parents and my sister also deserve heartfelt thanks for their belief in my abilities and constant motivation. Their support, understanding, and encouragement throughout this journey have been the cornerstone of my success. Thank you all from the bottom of my heart.

Contents

List of Publications

Journal Papers

J-1 **N. H. Chu,** D. T. Hoang, D. N. Nguyen, N. V. Huynh and E. Dutkiewicz, "Joint speed control and energy replenishment optimization for UAV-assisted IoT data collection with deep reinforcement transfer learning," *IEEE Internet of Things Journal,* vol. 10, no. 7, pp. 5778-5793, Feb. 2022, doi: 10.1109/JIOT.2022.3151201 *(Corresponding to Chapter 3).*

J-2. **N. H. Chu,** D. N. Nguyen, D. T. Hoang, Q. V. Pham, Khoa T. Phan, W.J. Hwang, and E. Dutkiewicz, "AI-enabled mm-Waveform configuration for autonomous vehicles with integrated communication and sensing," *IEEE Internet of Things Journal,* vol. 10, no. 19, pp. 16727-16743, Apr. 2023, doi: 10.1109/JIOT.2023.3270420 *(Corresponding to Chapter 4).*

J-3. **N. H. Chu,** D. T. Hoang, D. N. Nguyen, K. T. Phan, E. Dutkiewicz, Dusit Niyato, and Tao Shu, "MetaSlicing: A novel resource allocation framework for Metaverse," *IEEE Transactions on Mobile Computing*, Jun. 2023, doi: 10.1109/TMC.2023.3288085 *(Corresponding to Chapter 5).*

J-4. C. T. Nguyen, N. V. Huynh, **N. H. Chu**, Y. M. Saputra, D. T. Hoang, D. N. Nguyen, Q. V. Pham, D. Niyato, E. Dutkiewicz and W. J. Hwang, "Transfer learning for wireless networks: A comprehensive survey," *Proceedings of the IEEE*, vol. 110, no. 8, pp. 1073-1115, Aug. 2022.

J-5. **N. H. Chu,** N. V. Huynh, D. N. Nguyen, D. T. Hoang, S. Gong, T. Shu, E. Dutkiewicz, and K. T. Phan, "Countering Eavesdroppers with Meta-learning-based Cooperative Ambient Backscatter Communications," submitted to *the IEEE Transactions on Wireless Communications*, **major revision** *(Corresponding to Chapter 6).*

J-6. **N. H. Chu**, Nguyen Q. Hieu, D. N. Nguyen, D. T. Hoang, K. T. Phan, E. Dutkiewicz, Dusit Niyato, and Tao Shu, "Dynamic Multi-tier Resource Allocation Framework for Metaverse," submitted to *IEEE Network*, **minor revision** *(Partially corresponding to Chapter 5).*

J-7. Hai M. Nguyen, **Nam H. Chu**, Diep N. Nguyen, Dinh Thai Hoang, Minh Hoàng Hà, Eryk Dutkiewicz, "Optimal Privacy Preserving in Wireless Federated Learning over Mobile Edge Computing," submitted to *IEEE/ACM Transactions on Networking*, **under review**.

J-8. Thai T. Vu, **Nam H. Chu**, Khoa T. Phan, Dinh Thai Hoang, Diep N. Nguyen, and Eryk Dutkiewicz "Energy-based Proportional Fairness in Cooperative Edge Computing," submitted to *IEEE Transactions on Mobile Computing*, **under review**.

Conference Papers

C-1. **N. H. Chu,** D. T. Hoang, D. N. Nguyen, N. V. Huynh and E. Dutkiewicz, "Fast or slow: An autonomous speed control approach for UAV-assisted IoT data collection networks," *in Proceedings of IEEE Wireless Communications and Networking Conference,* 2021, pp. 1-6. *(Partly Corresponding to Chapter 3).*

C-2. **N. H. Chu,** D. N. Nguyen, D. T. Hoang, K. T. Phan, E. Dutkiewicz, Dusit Niyato, and Tao Shu, "Dynamic resource allocation for Metaverse applications with deep reinforcement learning," *in Proceedings of IEEE Wireless Communications and Networking Conference*, 2023, pp. 1-6, (*Partly corresponding to Chapter 5).*

C-3. N. V. Huynh, Nguyen Quang Hieu, **N. H. Chu,** D. N. Nguyen, D. T. Hoang, and E. Dutkiewicz, "Defeating eavesdroppers with ambient backscatter communications," *in Proceedings of IEEE Wireless Communications and Networking Conference*, 2023, pp. 1-6. *(Partly corresponding to Chapter 6).*

C-4. N. Q. Hieu, **N. H. Chu,** D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, "A unified resource allocation framework for virtual reality streaming over wireless networks," *in Proceedings of IEEE International Conference on Communications,* 2023, pp. 1-6.

C-5. H. M. Nguyen, **N. H. Chu,** Diep N. Nguyen, D. T. Hoang, M. H. Ha, and E. Dutkiewicz, "Optimal privacy preserving in wireless federated learning over mobile edge computing", *in Proceedings of IEEE International Conference on Communications,* 2023, pp. 1-6.

Book Chapters

B-1 M. Aljumaie, H. C. Nguyen, **N. H. Chu,** C. T. Nguyen, D. N. Nguyen, D. T. Hoang, E. Dutkiewicz, "Potential applications and benefits of Metaverse," in *Metaverse Communication and Computing Networks: Applications, Technologies, and Approaches*, John Wiley & Sons, 2023. doi:10.1002/9781394160013.ch2

B-2 Diep N. Nguyen, **Nam H. Chu**, Dinh Thai Hoang, Octavia A. Dobre, Dusit Niyato, and Petar Popovski, "Generative AI for Communications Systems: Fundamentals, Applications, and Prospects," John Wiley & Sons, *in production.*

List of Figures

Abbreviation

5G/6G	The Fifth/Sixth Generation of Wireless Networks
ACK	Acknowledgment
AF	Ambiguity Function
AI	Artificial Intelligence
AmB	Ambient Backscatter
API	Application Programming Interface
AP	Access Point
AV	Autonomous Vehicle
BER	Bit Error Ratio
C-PHY	Control Physical Layer
CPI	Coherent Processing Interval
CSCG	Circularly Symmetric Complex Gaussian
CTMC	Continuous-Time Markov Chain
D3QL	Deep Dueling Double Q-learning
D3QL-TL	Deep Dueling Double Q-learning with Transfer Learning
DDQN	Double deep Q-network algorithm
DL	Deep Learning
DNN	Deep Neural Network
DQN	Deep Q-network algorithm
DRL	Deep Reinforcement Learning
eMBB	Enhanced Mobile Broadband
FFT	Fast Fourier Transform
ICAS	Integrated Communications and Sensing
IoT	Internet of Things
LRR	Long-Range Radar

LSTM	Long Short-Term Memory
MAML	Model-Agnostic Meta-Learning
MCS	Modulation and Coding Scheme
MDP	Markov Decision Process
MIMO	Multiple Input Multiple Output
MISP	Metaverse Infrastructure Service Provider
MLK	Maximum Likelihood
mmWave	Millimeter Wave
mMTC	Massive Machine-Type Communication
NF	Network Functions
NLOS	Non-Line-of-Sight
NSI	Network Slice Instance
NTNs	Non-Terrestrial Networks
OFDM	Orthogonal Frequency-Division Multiplexing
OFDMA	Orthogonal Frequency-Division Multiple Access
PER	Packet Error Ratio
QES	Quality of Experience
QoS	Quality of Service
RF	Radio Frequency
RL	Reinforcement Learning
SGD	Stochastic Gradient Descent
sMDP	semi-Markov Decision Process
sMP	semi-Markov Process
SNR	Signal-to-Noise Ratio
TL	Transfer Learning
UAV	Unmanned Aerial Vehicle
URLLC	Ultra-Reliable Low-Latency Communication
VNF	Virtual Network Function
VR	Virtual Reality
XR	Extended Reality

Chapter 1

Introduction and Literature Review

This chapter first overviews the specifications of the sixth-generation (6G) networks with its emerging services and challenges. Then, the existing solutions for handling these problems are comprehensively reviewed. After that, this chapter highlights the main contributions of this thesis. Finally, the thesis structure is provided.

1.1 Motivations

Over the past decade, wireless communication networks have experienced remarkable growth. While the fourth generation (4G) can only offer typical download speeds of around 100 Mbps, the latest generation of cellular network, i.e., 5G, is expected to deliver multi-Gbps peak data rates (such as 20 Gbps downlink and 10 Gbps uplink) and ultra-low latency communications (with delays as low as one millisecond) [1–3]. Notably, the fifth-generation (5G) shifts the focus from data rate-centric services (in previous generations) to supporting various service categories, such as enhanced mobile broadband (eMBB), Ultra-reliable low-latency communication (URLLC), and massive machine-type communication (mMTC), making it the enabler for emerging applications, e.g., Internet of Things (IoT) and smart cities.

However, existing 5G systems thus far do not reach the promised revolution. In particular, most 5G mobile connections are operating at sub-6 GHz, making it difficult to meet the 5G expectation rate [1]. Moreover, new services, ranging from extended reality (XR)-based services to connected autonomous systems, may not fit well the 5G focus of facilitating small packet and sensing based services [1]. For

Table 1.1 : Comparison of technology standards for 5G, Beyond 5G, and 6G [1].

	5G	**6G**
Application types	New applications: - eMBB - URLLC - mMTC	New applications: - mBRLIC - mURLIC - HCS - MPS
Device types	- Smartphones - Sensors - Drones	- Sensors and DII devices - CRAS - XR and BCI equipment - Smart implants.
Rate requirements	1 Gb/s	1 Tb/s
End-to-end delay	5 ms	<1 ms
Radio-only delay	100 ns	10 ns
Processing delay	100 ns	10 ns
End-to-end reliability	99.999%	99.99999%
Frequency bands	- Sub-6 GHz - MmWave for fixed access.	- Sub-6 GHz - MmWave for mobile access - THz bands (above 300 GHz) - Non-RF (e.g., optical and VLC)

example, extended reality (XR)-based services (especially with the rise of Metaverse), haptics, holographic teleportation, and connected autonomous systems may intentionally use large packet [1, 2]. Additionally, they also demand wireless communications that can sustain high data rates (Terabit per second) for both up and down links, high reliability (99.99999%), and ultra-low latency (microsecond-level latency) [2, 3]. To address these issues and facilitate emerging services, a disruptive 6G is required.

To shape 6G networks, many cutting-edge technologies (e.g., pervasive Artificial Intelligence (AI), mmWave and even higher frequency bands, unmanned aerial vehicles-assisted communications, and passive communications such as ambient backscattering for energy saving) are expected to be deployed [2], as illustrated in Figure. 1.1. By doing so, 6G can offer various new use cases, such as Metaverse/holographic teleportation, non-terrestrial communications, and autonomous cyber-physical sys-

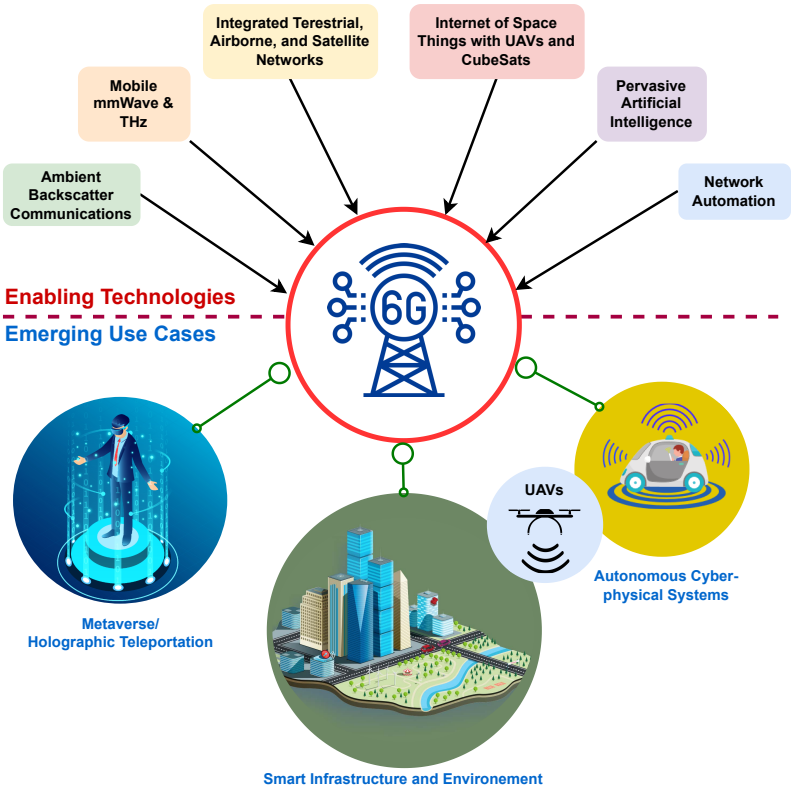


Figure 1.1 : 6G's enabling technologies and emerging use cases [1–3].

tems [1–3]. The integration of these technologies, on the one hand, provides an opportunity to overcome the limitations of current wireless communication networks. However, on the other hand, it also introduces new challenges due to the heightened complexity and strong correlations among network entities and users.

Firstly, non-terrestrial networks (NTNs) are important components in 6G communications systems since they can provide seamless coverage as well as extend the coverage to remote areas, where traditional terrestrial networks are expensive or even unavailable. Thus, NTNs offer promising solutions for collecting data from IoT devices in such areas, especially UAV-based solutions due to its flexibility and low cost. In particular, when UAVs act as on-demand flying access points (APs), thanks to their aerial superiority, they can establish good line-of-sight (LoS) links for the IoT nodes. In remote areas without access to terrestrial infrastructures, UAVs can

provide a much more economical solution to collect IoT data than other approaches, e.g., long-range ground broadcasting stations, or high-cost satellite communications in NTNs. Due to the flexibility, mobility, and low operational cost, UAVs have been being deployed as flying APs for some real-world projects, e.g., Google's Loon and Facebook's Aquila [4,5]. However, there are still some challenges that hinder the applications of UAVs in IoT data collection networks. In particular, unlike traditional solutions for collecting IoT data (e.g., deploying fixed APs), UAVs have limited energy resources supplied by batteries. When the UAVs' batteries are depleted, they must replenish their energy by flying back to the charging stations to charge or replace their batteries. It is worth noting that given a fixed working duration, the more time the energy replenishment process takes, the less time the UAVs can spend for collecting IoT data. Alternatively, the energy replenishment process is highly dynamic since it depends on the distance between the UAV and the charging stations. Therefore, optimizing energy usage and the energy replenishment process is critical to achieving high system performance, but very challenging in practice. Moreover, the UAVs often fly around to collect IoT data, while IoT nodes are statically allocated over different zones, and their sensing data are random depending on surrounding environments. To that end, optimizing operations of UAVs in different zones to maximize data collection efficiency is another major challenge that needs to be considered.

Secondly, Integrated Communications and Sensing (ICAS) technology plays a critical role in enabling 6G systems to become ubiquitous sensors [1]. Additionally, ICAS also emerges as a promising solution for Autonomous Vehicles (AVs), a use case of 6G [2], where sensing and data communications are two important functions that often operate simultaneously. The sensing function enables AVs to detect objects around them and estimate their distance and velocity for safety management, (e.g., collision avoidance) or for other use cases of 6G, such as Metaverse/holographic

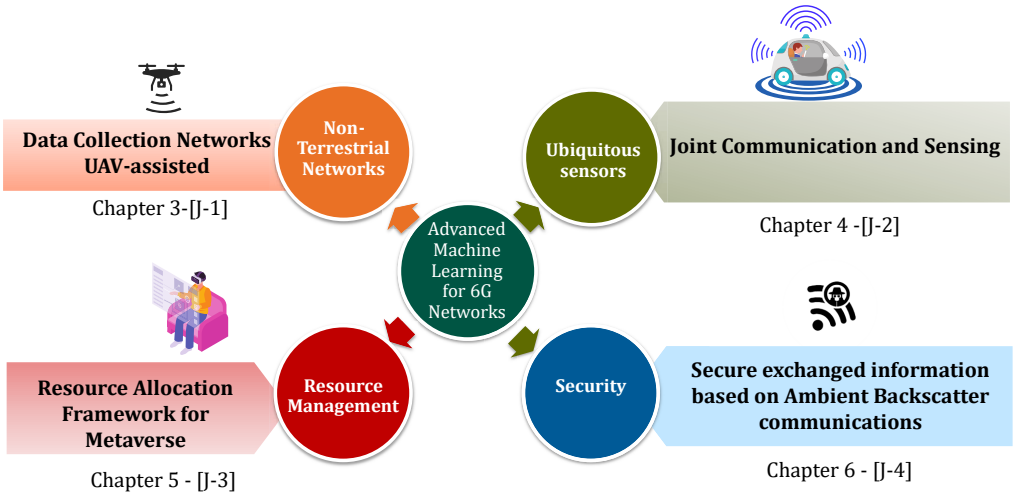


Figure 1.2 : The thesis overview.

teleportation. The data communication function allows AVs to exchange information with other AVs or infrastructure via Internet of Vehicles (IoV). For example, they can send/receive safety messages and even their own raw sensing data (e.g., traffic data around the AV) for applications such as transportation safety, transportation monitoring, and user services distributed to the AVs [6]. Although automotive sensing and vehicular communication can share many commonalities (e.g., signal processing algorithms and the system architecture [7]), they are typically designed and implemented separately. As such, communication and sensing functions require separate hardware components operating at different frequency bands that become increasingly expensive and inefficiency because of an ever-growing number of connected devices and services. Consequently, this makes the implementation of communication and sensing functions in AVs more costly in hardware, complexity, and radio spectrum resources. These challenges can be effectively addressed by combining both communication and sensing functions into a unified system, i.e., ICAS that can offer an efficient spectrum sharing mechanisms to avoid interference and coexist within a transmitter or other users. However, optimizing the waveform structure is one of the most challenging tasks due to strong influences between

sensing and data communication functions. Moreover, as 6G systems aim to utilize mmWave, it is necessary to investigate mmWave ICAS for AVs.

Thirdly, 6G will support a variety of emerging services and use cases with different performance metrics, such as throughput, latency, and reliability. It also needs to provide a consistent and satisfactory quality of service and user experience for all the users and applications. Recently, Metaverse has just been attracting more attention from academia and industry in the last few years, thanks to the recent advances in technologies (e.g., extended reality and edge intelligence) along with great efforts of many big corporations such as Facebook [8] and Microsoft [9]. The Metaverse is expected to bring a new revolution to the digital world [10]. Unlike existing virtual worlds (e.g., Second Life and Roblox), where the users' presentations (e.g., avatars/characters) and assets are limited in specific worlds, the Metaverse can be realized as a seamless integration of multiple virtual worlds [11]. Each virtual world in the Metaverse can be created for a certain application, such as entertainment, education, and healthcare. Similar to our real lives, Metaverse users can bring their assets from one to another virtual world while preserving their values, and vice versa. Moreover, the Metaverse is expected to further integrate digital and physical worlds, e.g., digitizing the physical environment by the digital twin [12]. For example, in the Metaverse, we can create our virtual objects, such as outfits and paintings, and then bring them to any virtual world to share or trade with others. We can also share virtual copies of a real object in different virtual worlds. Thus, the Metaverse will bring total new experiences that can change many aspects of our daily lives, such as entertainment, education, e-commerce, healthcare, and smart industries [13–15]. However, extremely-high resource demand of Metaverse makes resource management for the underlying infrastructure one of the biggest challenges that is impeding the deployment of the Metaverse [10].

Fourthly, 6G is expected to accommodate a massive number of devices and users

with diverse requirements and sensitivities, exposing new surfaces for cyber attacks. Among security threats in wireless communications, the eavesdropping attack is one of the most common types of wireless attacks. To perform the attack, the eavesdropper usually stays close to the victim system to "wiretap" the legitimate wireless channel and acquire exchanged information. Since the eavesdropper operates passively without introducing noise or altering transmit signals, detecting and preventing eavesdropping attacks are usually challenging. To deal with eavesdroppers, conventional approaches primarily rely on implementing encryption at the application and transportation layers [16]. Nevertheless, these approaches face several issues that significantly limit their practical applications, especially in resource-constrained devices. Moreover, an eavesdropper with sufficient computational capacity can decrypt the encrypted data, especially with the recent advances in quantum computing [17]. Furthermore, by using side-channel analysis, a strong eavesdropper can defeat many cryptographic schemes, even those with very robust schemes [16, 18]. Unlike cryptography-based approaches, physical layer security leverages physical characteristics of wireless channels (e.g., the signal strength) to protect information from eavesdroppers without requiring additional distributing and managing of cryptographic keys [19]. However, this approach often requires prior information of the legitimate channel state information (CSI) in order to achieve effective protection performance. In practice, acquiring such information is often challenging or even impractical in real-world scenarios.

To overcome the shortcomings of existing methods, machine learning emerges as a promising solution. Thanks to its ability of self-learning, machine learning has capability of intelligently handling intricate and extensive challenges in diverse domains, such as search engines, speech recognition, medical diagnosis, and computer vision. In 6G networks, machine learning can bring transformative capabilities (e.g., trainable radios), optimize network resources for enhanced perfor-

mance, enable intelligent connectivity to support emerging real-time applications like Metaverse/holographic teleportation. Moreover, personalized user experiences and energy-efficient operations further highlight machine learning's role, promising smarter, more adaptive, and sustainable 6G networks. However, conventional machine learning approaches (e.g., deep learning and reinforcement learning) usually require a large amount of high-quality data for the training process [20]. This requirement makes them less efficient in practice when data is expensive and/or contains noise due to the wireless environment's dynamics and uncertainty. Thus, this thesis seeks to pioneer advanced machine learning-based solutions that effectively and intelligently address the aforementioned challenges. Specifically, we leverage the latest advances in ML/AI to address the following questions:

- How can UAVs be dynamically and optimally controlled for speed and energy replenishment to enhance data collection efficiency in 6G non-terrestrial networks, considering the limitations of UAV energy capacity and the uncertainty in data collection tasks?

- How can the waveform configuration of the Integrated Communications and Sensing (ICAS) system be intelligently and adaptively optimized to maximize sensing and data communication performance within dynamic environments, and how does this address the intricate relationship between these functions?

- What machine learning-based framework can be developed to offer an effective and comprehensive solution for optimally managing enormous resource demands in various resource types for the Metaverse, leveraging similarities among applications to enhance resource management?

- How can wireless communication be secured in the presence of eavesdropper with minimal additional resource requirements but highly effective?

1.2 Literature Review and Contributions

This section begins by overviewing current studies in tackling the above problems. Then, it highlight gaps in the literature. Finally, this section emphasizes the main contributions of this thesis.

1.2.1 UAV-based data collection systems

1.2.1.1 Literature Review

In the literature, several works study the UAV's energy replenishment process for a UAV-assisted IoT data collection system where a charging station is deployed to prolong UAV serving time [21–25]. To minimize the age of information (AoI) under the constraint of the UAV's charging rate and battery capacity, the authors in [21] propose a least-charging-timed Metropolis-Hasting trajectory and a least-visit-time-based trajectory. They also point out that the UAV's charging rate has much more influence on the low bound of AoI than the UAV's battery capacity. The study in [22] aims to minimize the data collection time by optimizing the UAV trajectory and the order of IoT devices that the UAV is going to visit. Specifically, they employ a deep deterministic policy gradient-based algorithm to find a route between two positions, and a Q-learning based scheduler to determine the order of visiting positions where the IoT devices or charging stations are located. Furthermore, a transfer learning model is introduced to speed up the training process. However, the effectiveness of the proposed transfer learning technique is not well investigated. Similarly, the study in [23] aims to minimize the total time that the UAVs need to collect data from backscatter sensor nodes. If the UAVs' remaining energy is insufficient to complete the task, they can return to a charging station for charging. The authors first use the Gaussian mixture model to group IoT nodes into different clusters and formulate the trajectory optimization problem as a semi-Markov decision process. Then, deep reinforcement learning (DRL) approaches are proposed to find the optimal policy

for the UAV.

In [24], the authors consider that a UAV is wirelessly being charged during the data collection task. They first formulate the problem as a Markov decision process (MDP), then propose a Q-learning algorithm to maximize the energy efficiency and system throughput. In [25], the authors propose a blockchain-enabled UAV-assisted framework to provide security for IoT data collection networks. A charging coin is introduced to reward UAVs when they successfully collect IoT data. Then, the UAVs can use collected coins to recharge their batteries at a charging station. In addition, they develop an adaptive linear prediction model to reduce the number of transactions in the system, resulting in a decrease in energy consumption.

All of the above works [21–25] assume that the UAVs always fly at a constant speed during the data collection process. However, in practice, a UAV can choose different speeds during its data collection process depending on its surrounding environment. Alternatively, the UAV's speed can strongly influence the system's efficiency because it has a substantial impact on energy consumption during the data collection process [26]. Thus, optimally controlling UAVs' speed can significantly improve the energy usage and data collection efficiency of the system, especially in UAV-assisted IoT data collection networks where UAVs have limited battery capacities. Unfortunately, this important factor is not investigated in all the above studies.

Notably, only a few works investigate the speed control problem for UAV-assisted IoT data collection networks [27–30]. In [27], the authors aim to minimize the flight time for a data collection task by jointly optimizing the UAV's speed, data collection duration, and the IoT devices' transmit power. Their numerical results show that the UAV's optimal speed depends on the distance between sensors, sensors' energy, and the data upload requirements. In [28] and [29], the authors aim to maximize

the data collection efficiency by controlling UAV's speed according to the IoT device density. In particular, the authors in [28] first introduce an analytical model for the transmission between the UAV and IoT nodes, then the UAV's speed is optimized based on this model. In [29], the authors reveal a tradeoff between system throughput and IoT devices' energy efficiency. By optimizing the UAV's speed, altitude, as well as the MAC layer frame length, we can achieve the balance between the two conflict factors.

All the above works (i.e., [27–29]) apply the conventional optimization theories, which statically optimize the UAV's speed during the IoT data collection process. Therefore, their algorithms need to rerun whenever the environments are changed, leading to a high computational complexity. As a result, optimization-based solutions are inefficient in addressing the high dimensional state space as that in the considered problem. More importantly, they cannot be used in scenarios in which the complete information about the surrounding environment is unknown, like what we consider in this work (e.g., packet arrival probabilities for the whole network and flight time for replacing the battery). In this context, reinforcement learning emerges as the best approach to address the highly dynamic and uncertainty of the environment since it can help the UAV adapt its behavior according to the environment's changes. In [30], the authors employ deep Q-learning to control the UAV's speed during its data collection task, where the UAV can also wirelessly charge the IoT devices while collecting their data. This work aims to minimize the data packet loss by selecting the best devices to be charged and interrogated, together with the optimal UAV's speed. Their simulation results show that the UAV's speed is proportional to the number of IoT devices and inversely proportional to the data queue lengths of IoT devices. Similar to [27–29], the study in [30] does not consider the impacts of UAV's energy consumption and energy replenishment processes during the data collection task. It is worth highlighting that the energy replenishment process

is a critical factor that cannot be ignored since the UAVs' energy is limited.

It can be observed that all of the aforementioned works do not jointly optimize energy replenishment and speed control activities simultaneously. However, they are among the most important factors to achieve high efficiency in terms of energy and data collection for UAV-assisted IoT data collection networks. In addition, these proposed approaches require a high computational complexity that may be inefficient to deploy on UAVs. Moreover, most RL-based optimal operation policies for UAV-based collectors (i.e., [22–24, 30]) rely on conventional Q-learning or deep Q-learning based algorithms that are prone to overestimating state-action values. This problem can make the learning process unstable [31]. In addition, the work in [22] applies transfer learning to speed up the learning process of the proposed DRL algorithm. However, the impacts of transfer learning are not well investigated.

Note that transfer learning does not always improve or even can cause negative impacts on the learning process [32]. Furthermore, the study in [22] does not consider the UAV's speed control, one of the most important factors influencing to decisions of energy replenishment. To fill these gaps, this study develops a highly efficient solution based on deep reinforcement transfer learning for UAV-assisted IoT data collection networks. Specifically, our proposed approach can effectively address the overestimation problem and stabilize the learning process by adopting recent advanced techniques in RL, including deep Q-learning [33], deep double Q-learning [31], and dueling neural network architecture [34]. In addition, the proposed solution can simultaneously optimize the UAV's speed and energy replenishment processes and allow the learned knowledge to be effectively "shared" and "transferred" between UAVs. Thus, leveraging the transfer learning technique can improve the learning quality and reduce the learning time, thereby leading to a decrease of computational complexity. Therefore, our proposed solution can be effectively implemented on UAVs.

1.2.1.2 Contributions

Given the above, to jointly optimize the speed control and battery replacement activities for a considered UAV under the dynamic and uncertainty of IoT data collection process, we propose a dynamic decision solution leveraging the Markov decision process (MDP) framework. This framework allows the UAV to make optimal decisions (regarding the flying speed and battery replacement activities) based on its current observations about the surrounding environment. Although Q-learning can be used to find the optimal policy for the UAV, its convergence rate is slow, especially in a highly complex problem as the one considered in this problem where we need to jointly optimize the speed and energy replenishment activities for the UAV. In addition, Q-learning-based algorithms usually suffer from overestimation problems when estimating action values, especially for complicated problems with hybrid actions like what we consider in this problem (i.e., speed selection and energy replenishment actions) [31]. Thus, we develop a highly-effective Deep Dueling Double Q-learning (D3QL) to address these challenges. The key ideas of D3QL are to (1) separately and simultaneously estimate the state values and action advantages, making the learning process more stable [34], and (2) address the overestimation by using two estimators (e.g., deep neural networks), resulting in the stability of estimating action values.

To further reduce the learning time and enhance the learning quality, we develop transfer learning techniques to allow the UAV to learn more knowledge from other UAVs learning in similar environments. In addition, these techniques also help the UAV leverage knowledge obtained from different environments to improve its policy, making our solution more applicable and scalable. Therefore, our proposed solution can be deployed on resource-constrained devices, e.g., UAVs. Extensive simulation results demonstrate that our proposed solution, i.e., D3QL with transfer learning (D3QL-TL), can simultaneously optimize the energy usage and data collection, and

thereby leading to the best performance compared to other methods. To the best of our knowledge, this is the first study investigating a UAV operation control approach taking the dynamic of the IoT data collection, energy limitation, and impact of the energy replenishment process into considerations.

Our major contributions are summarized as follows.

- We propose a novel framework that allows the UAV to jointly optimize its flying speed and battery replacement activities under the dynamic and uncertainty of data collection and energy replenishment processes. In addition, this framework can not only allow the UAV to dynamically and automatically make optimal decisions through real-time interactions with the surrounding environment but also enable the "share/transfer" learning knowledge among UAVs working in the same and/or similar environments.

- We develop a highly-effective DRL algorithm leveraging recent advances of deep Q-learning, deep double Q-learning, and dueling neural network architecture to stabilize the learning phase, thereby quickly obtaining an optimal operation policy for the UAV.

- To reduce the learning time and improving learning quality for the UAV, we develop advanced transfer learning techniques that allow UAVs to "share" and "transfer" their learning knowledge. In addition, these techniques help UAVs to utilize the knowledge and information learned from different environments, making our approach more scalable and applicable in practice, e.g., scenarios with multiple UAVs.

- Then, we perform extensive simulations to demonstrate the efficiency of our proposed approaches and reveal critical elements that can significantly impact on the performance of UAV-assisted IoT data collection networks.

1.2.2 Intergrated Communication and Sensing Systems based on mmWave

1.2.2.1 Literature Review

Currently, two standards operating at 5.9 GHz for vehicular communication networks are C-ITS based on IEEE 802.11bd in Europe [35] and DSRC based on IEEE 802.11p in the U.S. [36]. Unfortunately, their data rates (i.e., up to 27 Mbps) do not meet the requirements of AVs' applications. For example, precise navigation that needs to download a high definition three-dimension map and raw sensor data exchange between AVs to support fully automated driving may require connections up to a few Gbps [37]. In addition, the performance of communication and sensing in ICAS systems operating at sub-6 GHz is limited due to the bandwidth availability [38]. In this context, millimeter wave (mmWave), whose frequency is from 30 GHz to 300 GHz, has been emerging as a promising solution to address the above challenges in ICAS systems [39]. First, owing to the high-resolution sensing and small antenna size, mmWave is predominantly utilized for automotive Long-Range Radar (LRR) [40]. Second, an mmWave system, e.g., a wireless local area network (WLAN) operating at the 60 GHz band, can provide a very high data rate to meet AVs' intensive communication requirements.

However, several challenges are hindering the applications of mmWave ICAS systems in AVs. In particular, unlike the conventional approaches where sensing and communication are separated, the ICAS-AV leverages a single waveform for both sensing and communication functions. Thus, it needs to jointly optimize these two functions simultaneously to achieve high performance of data communication and sensing for ICAS systems. In addition, since the ICAS operates while ICAS-AVs are moving, the surrounding environments of AVs are highly dynamic and uncertain. This makes ICAS-AVs' performance unstable as mmWave is more severely impacted by wireless environments than those of the sub-6 GHz bands [41]. Therefore, the

highly dynamic and uncertainty of mmWave ICAS's environment is another critical challenge that needs to be addressed. To that end, mmWave ICAS systems are demanding an effective and flexible solution that can not only jointly optimize communication and sensing functions but also adaptively handle the highly dynamic and uncertainty of the surrounding environment. As a result, it can best sustain high data rate communication links (given the highly directional mmWave communications) and sensing accuracy, e.g., low target miss detection probability and low estimation error of the target's range and velocity.

A few works in the literature have recently studied communication mmWave waveforms for ICAS systems [38, 42–46]. In [42] and [43], the authors exploit a single IEEE 802.11ad data communication frame to provide the sensing function. Specifically, the authors in [42] propose to use the preamble of the Single Carrier Physical Layer (SC-PHY) frame in IEEE 802.11ad to extract sensing information. The simulation results show that this approach can achieve a data rate of up to 1 Gbps with high accuracy in target detection and range estimation. However, the velocity estimation is poor because the preamble is short. Particularly, the proposed approach achieves the desired velocity accuracy (i.e., 0.1 m/s) only when the Signal-to-Noise Ratio (SNR) is high, i.e., greater than 28 dB. In [43], the authors aim to overcome this issue by using the IEEE 802.11ad Control Physical Layer (C-PHY) frame that has a longer preamble than that of IEEE 802.11ad SC-PHY. However, it is still not large enough to improve the velocity estimation, whereas the data rate is only 27.5 Mbps, significantly lower than the desired data rate for AV's communication [37]. These results from the above studies (i.e., [42] and [43]) suggest that a single frame processing is unable to satisfy the desired velocity estimation accuracy for AVs.

Multi-frame processing has been recently considered to be a potential solution for ICAS systems to improve sensing information extracted from targets' echoes,

e.g., [38, 44–46]. The authors in [44] propose velocity estimation algorithms that leverage multiple fixed-size frames based on IEEE 802.11ad SC-PHY in a CPI. Their results demonstrate that the proposed solution can achieve the desired velocity accuracy of AVs (i.e., 0.1 m/s [40]) when the number of frames is greater than 20. In [45], the authors develop a similar multi-frame processing method to embed sensing functionality in IEEE 802.11ad physical layer frame for a Vehicle-to-Infrastructure (V2I) scenario. By doing so, they can reduce the beam training time of 802.11ad up to 83%. Instead of using the 802.11ad standard, the authors in [46] propose a ICAS waveform based on Orthogonal Frequency-Division Multiple Access (OFDMA) for a bi-static automotive ICAS system in which the sensing area is extended to non-light-of-sight positions by exploiting reflected signals from other obstacles. However, in this work, the maximum communication data rate is only up to 0.1 Mbps which is dwarfed compared to the desired data rate in AVs.

Recently, reinforcement learning has been leveraged to address the dynamic and uncertainty of environments in various aspects of wireless communication, such as ICAS [47], spectrum sensing [48], and anti-jamming in wireless sensor networks [49]. In particular, the authors in [47] consider the ICAS system that is the combination of two different subsystems (i.e., communication and sensing) working separately at different beams and channels (i.e., frequencies). They develop a DRL-based algorithm to optimally allocate resource blocks (i.e., a tuple of beam, channel, and power) to sensing and communication tasks of UAVs. The major shortcoming of [47] is that it does not consider the dynamic and uncertainty in the data arrival process and wireless channel, which are addressed in our framework. In practice, these dynamics and uncertainties are very important and cannot be ignored since autonomous vehicles are of high mobility, and the transmission demand of users and the AV system varies over time. Moreover, the system in [47] requiring separate beams and channels (i.e., frequencies) for communications and sensing is not as

spectrum-efficient as the ICAS system considered in our work that leverages the same signal for both communication and sensing functions.

A common drawback in the above studies (i.e., [44–46]) is that the waveform structures (e.g., number of frames in CPI) are not optimized. Instead, these parameters are manually set. In practice, the dynamic and uncertainty of the ICAS's environment (e.g., SNR and data arrival rate) can significantly influence the ICAS's data transmission rate as well as sensing accuracy in velocity/range estimation. Thus, finding the optimal waveform structure according to the surrounding environment and timely adapting the selected structure with the dynamics of the surrounding environment play vital roles. To address this problem, in [38], the authors propose an adaptive virtual waveform design for mmWave ICAS based on the 802.11ad standard to achieve the optimal waveform structure (i.e., number of frames in CPI) that can balance between communication and sensing performance. The results show that given a fixed length of CPI, increasing the number of frames in CPI can increase the sensing performance, but it will degrade the communication performance (i.e., data rate). However, this approach requires complete information about the surrounding environment in advance, which may be impossible to obtain in practice. As such, their proposed solution needs to be rerun from scratch if there is any change in the environment.

In addition, none of the above studies (i.e., [38,42–46]) considers the dynamic and uncertainty problem of the information and environment, e.g., the changes of the wireless channel quality and the arrival rate of data that need to be transmitted via ICAS. This problem is critical to the performance of the ICAS system because the surrounding environment consistently changes as the ICAS-AV is moving. In particular, the rapid change of the wireless channel quality (e.g., SNR) highly impacts the ICAS's communication efficiency (i.e., packet loss due to transmission failure) and sensing performance (i.e., target detection and targets' range and speed estimation

accuracy). The problem is even more critical for mmWave systems that are highly directional and prone to blockages/fading. Moreover, the data arrival process at the ICAS-AV is often unknown in advance since it varies in different applications (e.g., navigation and automated driving). When the data arrival rate at the AV's ICAS system is higher than its maximum transmission rate, data starts to pile up in a data queue/buffer. Since a data queue/buffer size is always limited, packet loss will occur when the queue is full. This problem can cause serious issues for AVs as they cannot communicate with other AVs and infrastructure. Given the above, adaptively optimizing the waveform of ICAS is an effective approach to not only jointly optimize both sensing and communication performance but also effectively deal with the dynamic and uncertainty of the surrounding environment. However, to the best of our knowledge, this approach has not been investigated in the literature.

1.2.2.2 Contributions

To fill this gap, this thesis aims to propose a novel framework to maximize the performance of an ICAS system by adaptively optimizing the waveform structure under the dynamic and uncertainty of the surrounding environment when the AV is moving. It is worth noting that since the sensing processing for the 802.11ad-based ICAS is well-investigated in [38, 42–45], this study only focuses on addressing the waveform structure optimization problem for mmWave ICAS AVs under the dynamic and uncertainty of surrounding environments. To that end, we first model the problem as a Markov Decision Process (MDP) because it can allow the AV to determine the optimal waveform (e.g., the number of frames in CPI) based on its current observation (e.g., channel state and number of data packets in the data queue). Then, we adopt the Q-learning algorithm, which is widely used in Reinforcement Learning (RL) due to its simplicity and convergence guarantee, to help the ICAS-AV gradually learn the optimal policy via interactions with the surrounding

environment. However, Q-learning may face the curse of dimensionality and overestimation problems that lead to a low converge rate and an unstable learning process when the state space is large [31]. In our case, the state space that consists of all possible observations of the surrounding environment is very large, while ICAS-AV requires fast learning to promptly respond to the highly dynamic and uncertainty of the ICAS-AV's environment. Therefore, we develop a highly-effective learning algorithm based on the most recent advances in RL, namely i-ICS, to deal with these problems. First, i-ICS addresses the high dimensional state space problem by utilizing a deep neural network (DNN) to estimate the values of states [33]. Second, the overestimation is handled by using the deep double Q-learning [31]. Finally, the learning process is further stabilized and accelerated by leveraging the dueling neural network architecture that separately and simultaneously estimates the advantage values and state values [34]. Our major contributions are as follows.

- Design a novel framework by which the ICAS-AV can dynamically and automatically optimize the waveform structure under the highly dynamic and uncertainty of its surrounding environment to jointly optimize the communication efficiency and sensing accuracy, thereby maximizing the ICAS's performance.

- Develop a highly-effective deep RL (DRL) algorithm taking advantages of recent advances in RL, including deep Q-learning, deep double Q-learning, and dueling neural network architecture, that can help the ICAS-AV quickly obtain the optimal policy.

- Perform extensive simulations to investigate the effectiveness of our proposed solution under different scenarios and reveal key factors that can significantly influence the performance of the ICAS system.

1.2.3 Resource Managent for Metaverse

1.2.3.1 Literature Review

Mountainous resource demand in the Metaverse is one of the biggest challenges that is impeding the deployment of the Metaverse [10]. To fulfil the Quality-of-Service (QoS) and meet the user experience requirements in the Metaverse, it demands enormous resources that may have never been seen before. First, Metaverse is expected to support millions of users simultaneously since each Metaverse application can host a hundred thousand users simultaneously. For example, the peak number of concurrent players of Counter Strike - Global Offensive is more than one million in 2021 [50]. It is forecasted that data usage on networks can be expanded more than 20 times by the operation of Metaverse [51]. Second, unlike the current online platforms (e.g., massive multiplayer online role-playing games where the uplink throughput can be much lower than that of downlink throughput [52]), the Metaverse requires extremely-high throughput for both uplink and downlink transmission links. The reason is that Metaverse users can create their digital objects and then share/trade them via this innovation platform. Therefore, to maintain the Quality-of-Experience (QoE) for users, the Metaverse's demand for resources (e.g., computing, networking, and storage) likely exceeds that of any existing massive multiplayer online application [10].

More importantly, the required resource types are highly correlated. In particular, the Extended Reality (XR) technology is believed to be fully integrated into Metaverse's applications such that users can interact with virtual and physical objects via their digital avatars, e.g., digital twin [10]. Therefore, it requires not only extensive computing to render three-dimensional (3D) objects, a large amount of data collected from perceived networks, e.g., the Internet of Things (IoT), but also an ultra-low delay communication to maintain a seamless user experience. Thus,

intensive resources are required not only to contain and operate Metaverse applications but also to support massive data forwarding over networks. Given the above, resource management for Metaverse is a challenging task due to the mountainous size of resources of different types and the correlations between these types.

In this context, although deploying the Metaverse on the cloud is a possible solution, it leads to several challenges. First, the cloud is often located in a physical area (e.g., a data center), making it potentially a point-of-congestion when millions of users connect at once. Second, since users come from around the world, a huge amount of exchanged data puts stress on the communication infrastructure. This results in high delay, which severely impacts the Metaverse since the delay is one of the crucial drivers of user experience [53]. In this context, multi-tier resource allocation architecture, where the computing, storage, networking, and communication capabilities are distributed along the path from end-users to the cloud, is a promising solution for the Metaverse implementation.

In the literature, there are only a few attempts to investigate the Metaverse resource management [54–57]. Specifically, in [54], the authors consider computing resource allocation for a single-edge computing architecture that has limited computing resources to allocate for some nearby Metaverse users. Similarly, in [55] and [56], resource allocation at the edge is considered, but more resource types, i.e., computation and communication, are considered. In particular, the work in [55] proposes a pricing model-based resource management to accelerate the trading of Virtual Reality (VR) services between end-users and VR service providers. In [56], the authors address the stochastic demand problem for an application of education in the Metaverse. Specifically, they propose a stochastic optimal resource allocation method to minimize the cost for the virtual service provider. Unlike the above works, in [57], the authors propose an evolutionary game-based resource management for perception networks (e.g., IoT) that are used to collect data for the Metaverse.

It can be observed that none of the above studies considers the multi-tier computing architecture for resource allocation problem. Instead, their approaches are only appropriate for a single-tier edge computing architecture [54–57]. However, as analyzed above, due to the extremely high resource demands of Metaverse applications, the single-tier computing resource model may not be appropriate and effective. First, relying on a single-tier computing architecture may not guarantee optimal performance for Metaverse applications. The available resources at the edge (i.e., near end-users) are often much lower than those of the cloud and may not satisfy the intensive resource requirements of Metaverse applications [58]. This can lead to high computational latency or even disrupting services due to the lack of resources. Second, all resources in the single-tier model concentrate in one place, possibly resulting in a single point of failure problem, and thus this single architecture has a low ability of scalability and flexibility. Third, the single-tier edge computing tends to initialize Metaverse applications near the user's location. As such, if the user moves far away from that location, the system cannot guarantee a good Quality of Experience (QoE) [58]. Given the above, the single-tier architecture may not be an optimal solution for deploying Metaverse applications.

Furthermore, it can be observed that in the Metaverse, many Metaverse applications may share some common functions. For example, a digital map is indeed a common function between tourism and navigation applications. Currently, sharing functions among applications has already been made. For instance, Google Map's Application Programming Interface (API) provides various functions (e.g., digital map, check-in, display live data synching with location [59]) that can be shared among many applications, e.g., Pokemon Go [60], Wooorld [61], and CNN iReport Map [62]. This special feature of the Metaverse indeed can be leveraged to maximize resource utilization for Metaverse applications. However, none of the above works can exploit the similarity among applications to improve resource utilization

for the Metaverse. Moreover, in practice, users join and leave the Metaverse at any time, leading to the high uncertainty and dynamic of resource demands. Among the aforementioned works, only the study in [56] addresses the stochastic demands of users. Nevertheless, it only considers resource allocation for a Metaverse education application. In addition, this approach is only appropriate for a single-tier resource allocation and could not leverage the similarities among Metaverse applications' functions in order to maximize system performance. Thus, there is an urgent need for an effective and comprehensive solution for the Metaverse to handle not only the massive resource usage but also the dynamic and uncertain resource demand.

1.2.3.2 Contributions

To address all the aforementioned challenges, we propose a novel framework, namely MetaSlicing, to intelligently allocate diverse types of resources for Metaverse applications by analyzing the incoming requests and allocating appropriate resources, and thereby maximizing the whole system performance. Firstly, we introduce the idea of decomposing an application into multiple functions to facilitate the deployment and management of Metaverse, which are highly complex. In particular, each function of an application can be initialized separately and placed at a different tier in the system according to functions' requirements and tiers' available resources. For example, functions with low latency requirements can be placed at a low tier (e.g., tier-1), while those with low update frequency can be placed at a higher tier. By doing so, the application decomposition can not only provide a flexible solution for deploying Metaverse applications but also utilize all networks' resources from different tiers. Secondly, we propose a novel technique, called MetaInstance, to address extreme-high resource demands of the Metaverse. The MetaInstance aims to improve resource utilization by exploiting the similarities among Metaverse applications. To be more specific, applications with common functions will be grouped

into a MetaInstance, and the common functions will be shared among these applications instead of creating one for each application. Therefore, this technique can save more resources. Finally, to address the uncertainty and dynamic of resource demands as well as the real-time response of applications in Metaverse, we develop a highly-effective framework based on the semi-Markov decision process together with a reinforcement learning algorithm that can automatically find out an optimal policy under the dynamic and uncertain resource demand.

In summary, our contributions are as follows:

- We propose a novel framework in which different types of resources at different tiers of the computing architecture can be allocated smartly to maximize the system performance for the Metaverse.
- We introduce two innovative techniques, including Metaverse application decomposition and MetaInstance, to maximize resource utilization for the proposed multi-tier computing-based Metaverse.
- We propose a highly effective admission control model based on the semi-Markov decision process that can capture the high dynamic and uncertainty of resource demand as well as the real-time characteristic of the Metaverse.
- We develop an intelligent algorithm that can automatically find the optimal admission control policy for the Metaverse without requiring the complete information about the dynamic and uncertainty of resource demand.
- We perform extensive simulations not only to explore the resilience of our proposed framework but also to gain insights into the key factors that can affect the system performance.

1.2.4 Counter Eaveasdopping Attacks based on Ambient Backscatter Communications

1.2.4.1 Literature Review

To deal with eavesdropping attacks, conventional approaches primarily rely on implementing encryption at the application and transportation layers [16]. Nevertheless, these approaches possess several issues that significantly limit their practical applications, especially in resource-constrained devices. Firstly, these approaches require additional computing resources to facilitate the encryption and decryption, making them less practical or even infeasible for resource-limited devices such as IoT devices [63]. Secondly, distributing and managing cryptographic keys also require additional communication resources (e.g., frequencies and transmission power) and are very challenging tasks, especially in decentralized systems comprising a large number of mobile devices [64]. Finally, an eavesdropper with sufficient computational capacity can decrypt the encrypted data, especially with the recent advances in quantum computing [17]. In addition, by using side-channel analysis, a strong eavesdropper can defeat many cryptographic schemes, even those with very robust schemes [16], [18].

Unlike cryptography-based approaches, physical layer security leverages physical characteristics of wireless channels (e.g., the signal strength) to protect information from eavesdroppers without requiring additional distributing and managing of cryptographic keys [19]. In physical layer security, friendly jamming is a popular approach in which artificial noise is intentionally injected into wireless channels to disrupt eavesdroppers' signal reception [65–67]. However, this approach may not always ensure a positive secrecy rate, which is the difference of capacity between the legitimate channel (from the transmitter to the intended receiver) and the "tapped" channel (from the transmitter to the eavesdropper). Moreover, the generated inter-

ferences may also severely degrade signal receptions at neighboring legitimate devices, especially in densely populated wireless networks. Recently, friendly jamming and beamforming techniques have been leveraged in cooperative transmissions to counter eavesdropping attacks [68]. In particular, the cooperative transmission uses relays to forward data from a transmitter to a receiver. In the cooperative jamming mode, relays can also inject noises (i.e., jamming signals) to confuse eavesdroppers in the relays' coverages. Whereas in the cooperative beamforming mode, these relays perform distributed beamforming toward the legitimated receiver to minimize the signal strength at eavesdroppers. The main drawback of friendly jamming- and beamforming-based solutions is that they generally require prior information of the legitimate channel state information (CSI) in order to achieve effective protection performance. However, acquiring such information is often impractical and challenging in real-world scenarios. Moreover, these techniques require additional resources (e.g., energy and computing) for generating jamming signals and performing beamforming.

Given the above, this article leverages the cooperative Ambient Backscatter (AmB) communications in which the transmitter is equipped with an AmB tag that can backscatter ambient radio signals to convey information to the receiver [69]. In the literature, the AmB communications have been well investigated in various aspects, such as hardware design [70], performance improvement [71, 72], power reduction [73, 74], ambient backscatter-based applications [75, 76], and security [77–79]. However, utilizing AmB technology to counter eavesdropping attacks has not yet been well investigated. For example, the works in [77, 78] investigate the security and reliability of AmB-based network with imperfect hardware elements, while the authors in [79] analyze the application of physical layer security for AmB communications. Unlike these works in the literature, our proposed solution exploits the advantages of AmB communications together with a simple encoding technique to

secure the transmitted information against eavesdroppers. In particular, the original message is split into two parts: (i) the active message transmitted by the transmitter using conventional active transmissions and (ii) the AmB message transmitted by the backscatter tag using AmB transmissions. Then, the receiver reconstructs the original message based on both active and AmB messages. Note that the AmB tag operates in a passive manner, meaning that it does not actively transmit signals. Instead, it utilizes the active signals from the transmitter to backscatter the AmB message without requiring additional power. In this way, the AmB message can be transmitted on the same frequency and at the same time as the active message. However, the AmB signal strength at the receiver is significantly lower than that of the active signal. Hence, the AmB signal can be considered as pseudo background noise for the active signal [63, 80]. As such, without knowledge about the system in advance (i.e., the settings of AmB transmission), the eavesdropper is even not aware of the existence of the AmB message. Without accessing the AmB message, the eavesdropper may not be able to recover the original message.

It is worth noting that if eavesdropper is aware of the AmB message, it is still a challenging task to capture and reconstruct the original message. Specifically, the values of resistors and capacitors in the AmB circuit are different for different backscatter rates. As such, even if the eavesdropper deploys the AmB circuit but do not know the exact backscatter rate, it still cannot decode the backscatter signals [70]. In addition, in the worse case in which the eavesdropper knows the exact backscatter rate in advance, it still does not know how to construct the original message based on the active and AmB messages since the message encoding technique is unknown to them. To quantify the security of our proposed anti-eavesdropping solution in the worst case, this thesis considers the guessing entropy metric [81]. It is worth noting that due to the lower rate of AmB transmission compared to that of the active transmission, the AmB message's size is smaller than that of the ac-

tive message. Thus, in a system with sufficient computing and energy capacities for encryption/decryption, the AmB message can be used to carry the encryption key, while the active message carries the encrypted message.

To detect backscattered signals at the receiver, traditional methods often employ MLK, which may require complex mathematical models and perfect CSI to achieve high detection performance [82, 83]. Thus, this approach introduces significant complexity and dependency on accurate CSI. To overcome these limitations, we develop a low-complexity Deep Learning (DL)-based detector that can effectively detect the backscattered signals. The rationale behind is that DL has the ability to learn directly from data (e.g., received signals), eliminating the need of complex mathematical models and perfect CSI. Note that in the literature, several works consider using DL for signal detection and channel estimation [84–88]. However, most of them consider conventional signals (e.g., OFDM, BPSK, and QAM64 signals), and thus it may not perform well for very weak signals like AmB signals. In particular, the authors in [84, 85] consider DL-based detectors for OFDM signals. Whereas, the studies in [86] and [87] consider the signal classification task, i.e., predicting the modulation type of signals. Unlike the above works, in [88], the authors propose an AmB signal detector based on the Long Short-Term Memory (LSTM) architecture. Since LSTM requires more computing capability than that of the conventional architecture, e.g., fully connected architecture [20], it does not fit well in our considered lightweight anti-eavesdropping framework. Therefore, this work proposes a new detector with low complexity elements, such as *tanh* activation, and a few small-size fully connected hidden layers. By doing so, our DL-based detector offers a more practical and efficient solution for backscatter signal detection at the receiver.

Although DL-based detectors can achieve good detection performance, they usually require a large amount of high-quality data (i.e., a collection of received signals)

for the training process [89]. This makes DL less efficient in practice when data is expensive and/or contains noise due to the wireless environment's dynamics and uncertainty. For example, new objects (e.g. the passing of a bus) can significantly impact wireless channel conditions, even changing links from line-of-sight (LOS) to non-line-of-sight (NLOS). As a result, in conventional DL, models may need to be retrained from scratch with newly collected data, and thus it is a time-consuming task [90]. In this context, meta-learning (i.e., learning how to learn) emerges as a promising approach to quickly learn a new task with limited training data [91].

1.2.4.2 Contributions

This work develops a meta-learning algorithm to train the DL model to quickly achieve a good detection performance in new environments. Extensive simulation results substantiate the effectiveness of our proposed solution in effectively mitigating eavesdropping attacks. They also show that the proposed DL-based signal detector, without requiring perfect CSI, can attain a comparable Bit Error Ratio (BER) to the MLK-based detector, which is an optimal detector requiring a complex mathematical model and perfect CSI. The main contributions of this work are:

- Propose a novel anti-eavesdropping framework leveraging the AmB communications. In particular, we propose to use a low-cost and low-complexity AmB tag to assist in transmitting a part of information to the receiver by backscattering right on the transmit signals. This solution is expected to open a new direction for future anti-eavesdropping communications.

- Develop the DL-based detector to detect the AmB signals at the receiver to overcome limitations of the conventional MLK-based approaches. In particular, we propose a low complexity data preprocessing and design a lightweight Deep Neural Network (DNN) architecture to effectively detect AmB signals.

- Develop a meta-learning algorithm to quickly achieve high detection performance in new environments with little knowledge. The main idea of meta-learning is to utilize knowledge obtained from similar environments in order to reduce the size of the necessary training dataset, while still preserving the quality of learning.

- Perform extensive simulations in several scenarios to get insights into various aspects of our proposed framework, such as maximum achievable rate, security, and robustness. We also analyze the security level of our proposed framework based on the guessing entropy for the worse case when the eavesdropper has some prior information about the AmB communication settings.

1.3 Thesis Organization

The remainder of this thesis is structured as follows.

- Chapter 2: This chapter presents the brief overview of machine learning. In particular, Sections 2.1 and 2.2 provide a background of deep learning (DL) and reinforcement learning (RL), respectively. After that, deep reinforcement learning (DRL) and advanced machine learning techniques, i.e., transfer learning and meta learning, are discussed in Sections 2.2.3 and 2.3, respectively.

- Chapter 3: This chapter discusses our proposed framework that allows a UAV to jointly optimize its flying speed and battery replacement activities under the dynamic and uncertainty of data collection and energy replenishment processes. Specifically, the system mode is described in Section 3.1. Section 3.2 presents the problem formulation in details. Our proposed Deep Reinforcement Transfer Learning-based solution is proposed in Section 3.3. Evaluation results are then discussed in Section 3.4. Finally, conclusions are given in Section 3.5.

- Chapter 4: This chapter presents our proposed solution that can automatically optimize the waveform configuration for the integrated communications sensing of autonomous vehicles to strike a balance between these functions. In particular, Sections 4.1 and 4.2 introduce the ICAS system model and the problem formulation, respectively. Then, the Q-learning-based and the proposed i-ICS algorithms are proposed in Section 4.3. In Section 4.4, simulation results are analyzed. Finally, we conclude our study in Section 4.5.

- Chapter 5: This chapter introduces MetaSlicing, our proposed resource allocation framework for Metaverse. Particularly, Sections 5.1 introduces the system model based on multi-tier resource allocation architecture to facilitate the deployment and operation of Metaverse applications. Then, the Metaverse application admission control formulation is presented in Section 5.2. After that, the Metaverse application analysis and the proposed deep reinforcement learning-based algorithm are discussed in Section 5.3. In Section 5.4, simulation results are analyzed. Finally, Section 5.5 provides the conclusion of this work.

- Chapter 6: This chapter introduces a novel lightweight framework using ambient backscattering communications to counter eavesdroppers. Specifically, Our proposed anti-eavesdropping system and the channel model are discussed in Sections 6.1 and 6.2. Then, Sections 6.3 and 6.4 present the MLK-based detector and our proposed DL-based detector for the AmB signal, respectively. Our proposed deep meta-learning-based approach is presented in Section 6.5. Section 6.6 discusses our simulation results. Finally, Section 6.7 wraps up our study with a conclusion.

- Chapter 7: This chapter draws the conclusions and highlights future research directions.

Chapter 2

Background

Machine learning (ML) is poised to play a pivotal role in the evolution and operational efficacy of 6G networks, primarily due to the unprecedented complexity, heterogeneity, and dynamism that these networks are expected to exhibit. The envisioned 6G ecosystem will likely incorporate a vast array of technologies, including but not limited to, ultra-massive MIMO (Multiple Input Multiple Output) systems, terahertz (THz) communications, and dense networks of small cells. All of these emerging technologies are integrated with sophisticated sensing and communication capabilities. This complexity introduces challenges in network management, optimization, and the provisioning of services that conventional optimization approaches may struggle to address effectively. Conventional solutions, which often rely on predefined parameters, often fall short in dynamically adapting to the rapidly changing network states and user demands inherent in 6G environments. In contrast, thanks to its ability to learn from data, predict outcomes, and adapt in real-time, ML offers a promising solution. It can optimize network operations, enhance resource allocation, and improve service delivery through predictive analytics and intelligent decision-making processes. This adaptability is crucial for managing the intricate interplay between network elements in 6G, ensuring optimal performance, and meeting the high expectations for speed, reliability, and latency that define next-generation wireless networks.

Given the above, this thesis aims to develop advanced machine learning-based solutions for addressing various challenges in 6G networks. In this chapter, the fun-

damentals of deep learning is first presented. Then, the advanced machine learning techniques, i.e., transfer learning and meta-learning, are discussed in details.

2.1 Deep Learning

Deep learning (DL) is an area in machine learning in which neural networks are trained from a vast amount of data to perform some tasks automatically, e.g., classification and prediction. Although the concept of neural networks has been around for decades, recent advances in computing power, availability of large datasets, and breakthroughs in network architecture and algorithm design have brought a resurgence of interest and remarkable progress in DL. Nowadays, DL has been successfully applied in many applications supporting our daily lives, ranging from face and voice recognition to intelligent assistance systems. In comparison with conventional machine learning, DL has many advantages on various aspects [92]:

- **Feature Learning and Representation:** Conventional machine learning often relies on a manual feature selection (i.e., expert-engineered features) that is not only a time-consuming task but also may not able to capture complex patterns in high-dimensional data effectively. Whereas, DL can automatically learn hierarchical representations from raw data, thus eliminating the need for extensive feature engineering and allowing relevant features to be extracted directly.

- **Handling unstructured data:** Given the ability of automatic feature learning, DL is able to effectively handle unstructured data (e.g., images and audio) while conventional machine learning (e.g., linear regression or support machine vector) may struggle or even be unable to process such data.

- **Handling Large-Scale Data:** Traditional machine learning methods may struggle with big data due to computational limitations and memory con-

straints. In contrast, thanks to the ability to utilize parallel computing on specialized hardware, e.g., GPU, DL can efficiently process massive amounts of data, making them well-suited for big data tasks, such as optimizing large-scale and heterogeneous 6G networks.

- **Model Reusable:** In DL, trained models (i.e., DNNs) can be partially or fully reused to perform similar tasks. For example, convolution layers of AlexNet (a model trained on 1.3 million high-resolution images to perform object recognition) can be fine-tuned with datasets collected in communication systems to classify signal modulations, e.g., (BPSK, QPSK, and 64QAM) [93]. Such a situation is challenging for conventional machine learning methods since they often require a dataset trained from the previous learning as well as the new dataset [94].

In DL, a deep neural network (DNN) and a learning algorithm are two requisite components. Inspiring by the structure and function of the biological neuron network in the human brain, a DNN organizes neurons in multiple layers (hence the word "deep"), and each neuron can connect to one or more neurons, as shown in Figure 2.1. A connection between neurons is assigned a weight value. A neuron performs a mathematical operation on its input and then passes the result through an activation function $a_f(\cdot)$ to produce an output, as illustrated in Figure 2.1.

During the learning, a DL algorithm optimizes DNN's parameters θ (i.e., weights and bias) to minimize a loss function $\mathcal{L}(f(\mathbf{d};\theta),\psi)$. Generally, a loss function captures the error between the DNN's output $f(\mathbf{d};\theta)$ and the ground truth ψ, as follows:

$$\min_{\theta} \quad \mathbb{E}\big[\mathcal{L}(f(\mathbf{d};\theta),\psi)\big], \tag{2.1}$$

where $\mathbf{d}$ is the input vector. There are many types of activation functions (e.g.,

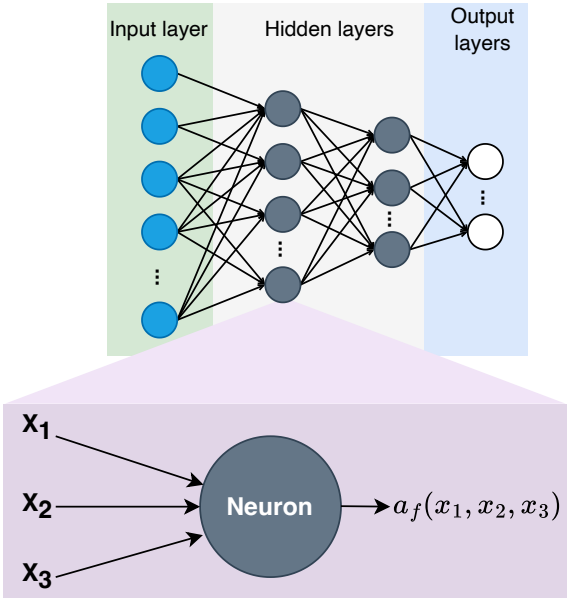


Figure 2.1 : An example of Deep Neural Network, a feed forward neural network(FNN).

sigmoid, ReLU, and Tanh) and loss functions (e.g., the cross-entropy loss, mean squared error loss, and Hinge loss) that can be used depending on specific problems [95].

The loss function $\mathcal{L}(f(\mathbf{d};\theta)$ can be minimized by the Gradient Descent (GD), which is the fundamental algorithm for minimizing deep learning loss functions due to its simplicity in implementation [96]. However, GD requires calculating the gradient and cost function for all data points at each time step, yielding a very high processing time for large data. To that end, this thesis proposes to use the Stochastic Gradient Descent (SGD) that only needs to compute gradients and cost function for a mini-batch data, i.e., $\mathbf{b}$, sampled uniformly from the dataset. By doing so, SGD can accelerate the convergence rate while still guaranteeing the convergence of learning [97]. Thus, SGD (presented in Algorithm 2.1) is widely used to optimize the DNN parameters due to the simplicity in implementation while still achieving good performance [95]. Specifically, at learning iteration t, a minibatch $\mathbf{b}_t$ is sampled

Algorithm 2.1 Stochastic Gradient Descent [95]

1: Input: Dataset consist of multiple data points, i.e.,($\mathbf{d}$, ψ).
2: Initialize DNN's parameters θ_0
3: **for** $t = 1$ *to* T **do**
4: Sample a minibatch $\mathbf{b}_t$ from the dataset.
5: Calculate the cost function and update DNN parameters according to (2.2) and (2.3), respectively.
6: **end for**

from the training dataset. Then, the cost function J_t is computed as follows:

$$J_t = \frac{1}{|\mathbf{b}_t|} \sum_{(\mathbf{d},\psi)\in\mathbf{b}_t} \mathcal{L}\big(f(\mathbf{d};\theta_t),\psi\big). \tag{2.2}$$

After that, the DNN's parameters are updated by

$$\theta_{t+1} \leftarrow \theta_t - \gamma_t \nabla_{\theta_t} J_t(\theta_t), \tag{2.3}$$

where γ_t is the step size controlling how much the parameters are updated, and $\nabla_{\theta_t}(\cdot)$ is the gradient operator with respect to the DNN parameters θ_t.

To effectively calculate the gradient of a cost function according to input data, DL often uses the backpropagation algorithm. To do that, it calculates the output error (i.e., a difference between predicted and actual values) when passing data from the input to the output layers, i.e., forward pass. Then, this output error is propagated backward through the network layer (i.e., back pass) to compute gradients. Through the training process, DL can learn complex patterns in the dataset, making them powerful tools for solving various data-driven tasks.

Note that the above discussion regards to the Feed Forward Networks (FNNs), one of the most fundamental and widely used DNN architectures. Besides FNNs, there are various types of DNNs (such as, Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and Autoencoders), and each of them is

tailored to different types of data and tasks. Among existing DNN architectures, the FNNs are the simplest and have the fastest inference latency [95]. Thus, this thesis leverages the typical FNNs that are simple and suitable to be implemented in wireless systems, which can well operate in rapidly changing environments with heterogeneous resource-constrained devices.

2.2 Reinforcement Learning

In machine learning, Reinforcement Learning (RL) focuses on training agents in environments so that they can automatically make decisions. Unlike supervised learning, where the model is provided with labelled data, and unsupervised learning, which deals with unlabelled data, RL operates in situations when we do not have data to train. As a result, in RL, an agent needs to interact with its surrounding environment to collect data to learn. Specifically, an agent in RL is employed to take actions and interact with the environment. Once an action is taken, the agent receives the immediate reward and observes the subsequent state of the environment, as shown in Figure 2.2. Then, the agent use these observations to learn and derive the optimal policy. In this way, RL is a powerful approach for training agents to make intelligent decisions in dynamic and complex environments, and its potential for solving real-world problems. In RL, underlying environments are typically formulated as a Markov Decision Process. Then, the agent uses an RL algorithm to gradually learn an optimal policy.

2.2.1 Markov Decision Process

The Markov Decision Process (MDP) functions as a mathematical framework to address decision-making challenges in dynamic and uncertain systems. It is widely utilized in dynamic programming and RL to formulate optimization problems. Generally, an MDP is described by four elements: (i) $\mathcal{S} \triangleq \{s\}$ denoting the state space,

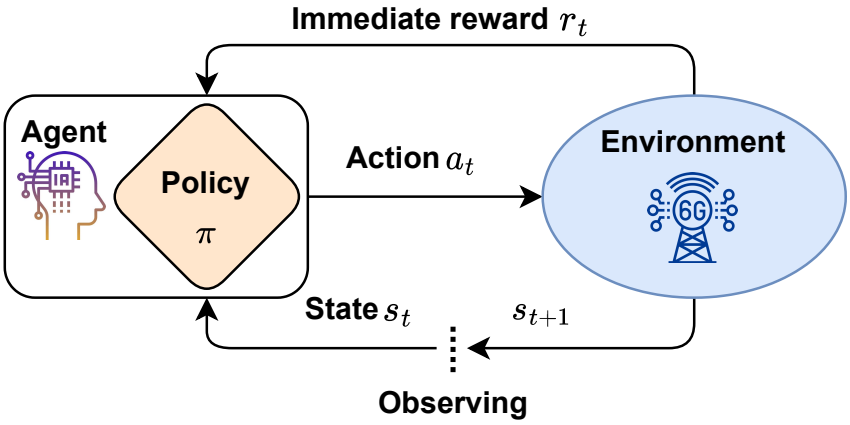


Figure 2.2 : Reinforcement learning.

(ii) $\mathcal{A} \triangleq \{a\}$ referring to the action space, (iii) $\mathcal{P}_a$ denoting the transition probability from the current state s to the next state s' by taking action a, and (iv) r_a representing the immediate reward obtained by performing action a. A policy is a mapping from the state space to the action space, i.e., $\pi : \mathcal{S} \rightarrow \mathcal{A}$, indicating the decisions made by the agent. The MDP aims to get the optimal policy π^* that maximizes the expected total reward calculated by $\sum_{t=0}^{\infty} \eta^t r_t \left(s_t, a_t = \pi^*(s_t)\right)$. Here the discount factor $\eta \in [0, 1]$ represents the importance of future rewards.

Note that the MDP divides time in equal time slots, and decisions are made at every time slot. As such, MDP may work inefficiently in system with stringent latency requirements, e.g., real-time processes. To address this shortcoming, the semi-Markov Decision Process (SMDP) is widely used in the literature. Specifically, in addition to the state space $\mathcal{S}$ and action space $\mathcal{A}$, a SMDP is also characterized by the a decision epoch t_i (i.e., points of time at which decisions are made) and $\mathcal{T}$, which refers to the state transition probabilities and also captures the duration of time spent in each state [98]. While a conventional MDP makes decisions at every time slot, an SMDP makes decisions whenever an event occurs, making it more suitable for effectively modelling real-time systems in practical applications.

2.2.2 Q-learning

In RL, the value of state s at any time step t under policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ is calculated by the state value function [99], i.e.,

$$V_\pi(s) = \mathbb{E}_\pi\Big[\sum_{m=0}^{\infty} \eta^m r_{t+m}\Big| s_t = s\Big], \forall s \in \mathcal{S}, \tag{2.4}$$

where $\mathbb{E}_\pi[\cdot]$ is the expectation under policy π, and η is the discount factor that indicates the importance of future rewards. Similarly, the state-action value function under policy π evaluates how good to performing action a at state s then following policy π, which is given by [99]:

$$Q_\pi(s,a) = \mathbb{E}_\pi\Big[\sum_{m=0}^{\infty} \eta^m r_{t+m}\Big| s_t = s, a_t = a\Big], \tag{2.5}$$

Under the optimal policy π^*, the optimal state-action value function, i.e., $\mathcal{Q}^*(s,a)$, is given by [99]:

$$Q^*(s,a) = \mathbb{E}\Big[r_t + \eta \max_{a' \in \mathcal{A}} Q^*(s_{t+1}, a')\big| s_t = s, a_t = a\Big]. \tag{2.6}$$

Thus, once $Q^*(s,a)$ is obtained, the optimal policy is achieved by taking actions that maximize $Q^*(s,a)$ for all state $s \in \mathcal{S}$.

In [100], the Q-learning algorithm is proposed to learn the optimal state-action value function $Q^*(s,a)$, also known as the optimal Q-function. Since then, Q-learning has been one of the most widely used algorithms because it can guarantee to converge to the optimal policy after the learning process [100]. The detail of the Q-learning algorithm is presented in Algorithm 2.2. Specifically, this algorithm uses a table, namely Q-table, making its implementation simple. Each cell of the Q-table keeps the estimated value of Q-function (named Q-value) for taking action a at state s, denoted by $Q(s,a)$. The Q-table is iteratively updated based on interactions with

Algorithm 2.2 Q-learning

1: The agent establishes the parameters (i.e., η, α, and ϵ) and create a Q-table arbitrarily (e.g., all cells are set to zero).
2: **for** $t = 1, 2, 3, \ldots$ **do**
3: The agent performs an action following the ϵ-greedy policy as follows:

$$a_t = \begin{cases} \underset{a \in \mathcal{A}}{\text{argmax}}\, Q(s_t, a), & \text{with probability } 1-\epsilon, \\ \text{random action } a \in \mathcal{A}, & \text{otherwise.} \end{cases} \tag{2.9}$$

4: The agent observes next state s_{t+1} and reward r_t, then updates $Q(s_t, a_t)$ by (2.7) and reduces ϵ.
5: **end for**

the surrounding environment. Given action a_t is selected under the ϵ-policy (2.9) at state s_t at time t, the agent obtains the immediate reward r_t and observes a next state s_{t+1}. Based on this observation, $Q(s_t, a_t)$ is updated by 2.7.

$$\begin{aligned} Q(s_t, a_t) \leftarrow & Q(s_t, a_t) + \\ & \underbrace{\alpha_t \big[\underbrace{r_t(s_t, a_t) + \eta \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})}_{\text{Target Q-value } \mathcal{Y}_t} - Q(s_t, a_t) \big]}_{\text{Temporal difference (TD)}}, \end{aligned} \tag{2.7}$$

where α_t is the learning rate that controls how important of new knowledge (i.e., TD) to the update of Q-function. By using (2.7) and α_t that satisfies (2.8) to iteratively updating the Q-function, it is proven that $q(s, a)$ will converge to $q^*(s, a)$ [100].

$$\alpha_t \in [0, 1), \ \sum_{t=1}^{\infty} \alpha_t = \infty, \text{ and } \sum_{t=1}^{\infty} (\alpha_t)^2 < \infty. \tag{2.8}$$

However, the usage of a table in Q-learning leads to the curse of dimensionality problem that results in a long time for learning, especially for challenging problems in 6G networks that often pose high dimensional state spaces, e.g., hundred thousand

states [101]. In addition, the uncertainty and dynamic of 6G networks' environment (e.g., the probability of frame loss, the packet arrival rate, and unprecedented resource demand) make it more challenging for the agent to achieve an optimal policy. To that end, we will introduce the deep reinforcement learning approach that can effectively address this problem to quickly achieve the optimal operation policy for the agent, thus maximizing the system performance for 6G networks.

2.2.3 Deep Reinforcement Learning

To address the aforementioned problems of Q-learning, the study [33] proposes DQN algorithm that uses DNNs as function approximators to handle large and continuous state spaces effectively, opening a sub-group of RL algorithms named deep reinforcement learning (DRL). Recall that traditional RL methods (e.g., Q-learning) often rely on tabular representations of state-action value functions (Q-values) or policy functions, which become computationally infeasible and inefficient in high-dimensional environments. In addition, the usage of the Q-table is only feasible when values of states are discrete, but the state in many problem can consists of real numbers. DRL addresses this limitation by leveraging DNNs to approximate the value functions or policy functions in complex and high-dimensional environments, making it highly applicable for addressing problems in 6G networks.

However, both DQN and Q-learning have the problem of overestimation when estimating Q-values [31]. This issue makes the learning process unstable or even results in a sub-optimal policy if overestimations are not evenly distributed across states [102]. To that end, this thesis introduces a DRL algorithm, namely Deep Dueling Double Q-learning, that can address these above issues effectively by adopting three innovation techniques in RL, including (i) Deep Q-Network (DQ) [33], (ii) Double Deep Q-learning (DDQ) [31], and (iii) dueling architecture [34]. First, the DNN is employed to estimate the state-action function so that the curse-of-dimensionality

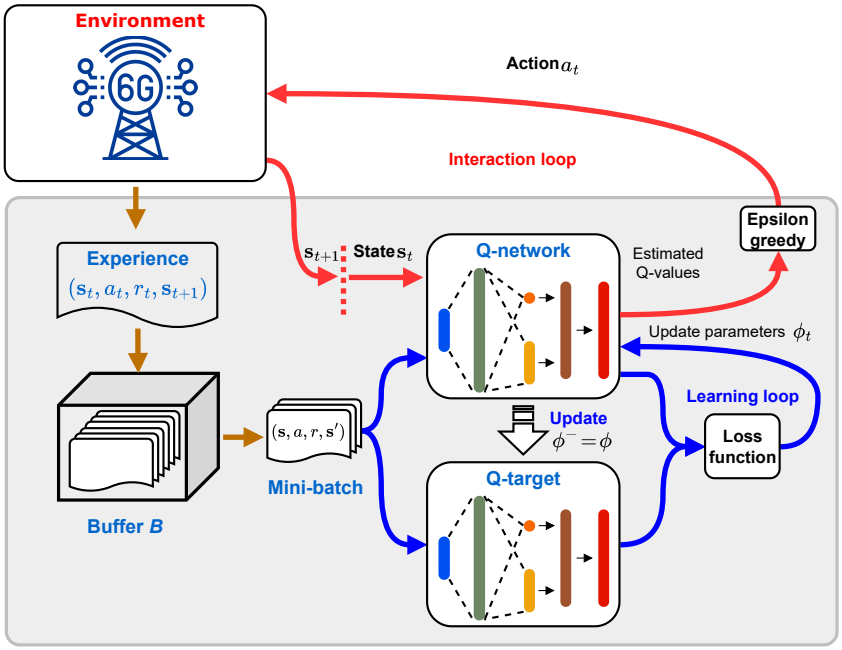


Figure 2.3 : The Deep Dueling Double Q-learning model.

problem of Q-learning can be effectively handled [33]. Second, the overestimation in Q-learning can be overcome by using DDQ that separates the action selection and action evaluation processes instead of combining them in the Q-learning [31]. Finally, the learning process is stabilized by adopting the dueling neural network architecture where the state value function and advantage value function are estimated separately and simultaneously [34]. In this way, our proposed approach can inherit all advantages of these techniques, thereby stabilizing the learning process, improving the learning speed, and reducing the overestimation.

2.2.4 Deep Dueling Double Q-learning

The details of Deep Dueling Double Q-learning (D3QL) are provided in Algorithm 2.3. Suppose that the learning phase consists of T time steps. At time step t, the agent observes the current state s_t and takes action a_t according to the ϵ-policy. After that, it observes a next state s_{t+1} and gets a reward r_t. This experience data, represented by a tuple (s_t, a_t, s_{t+1}, r_t), should not be used directly to train the DNN

Algorithm 2.3 Deep Dueling Double Q-learning

1: Initialize ϵ and buffer $\mathbf{B}$.
2: Create Q-network $\mathcal{Q}$ with random parameters ϕ.
3: Create target Q-network $\hat{\mathcal{Q}}$ by cloning from $\mathcal{Q}$.
4: **for** *step = 1, 2, 3, ..., T* **do**
5: Get action a_t following the ϵ-greedy policy as follows:

$$a_t = \begin{cases} \underset{a \in \mathcal{A}}{\text{argmax}}\ \mathcal{Q}(s_t, a; \phi_t), & \text{with probability } 1-\epsilon, \\ \text{random action } a \in \mathcal{A}, & \text{otherwise.} \end{cases} \tag{2.10}$$

6: Perform a_t, then observe reward r_t and next state s_{t+1}.
7: Save experience (s_t, a_t, r_t, s_{t+1}) in $\mathbf{B}$.
8: Create a mini-batch of experiences by sampling randomly from buffer $\mathbf{B}$, i.e., $(s, a, r, s') \sim U(\mathbf{B})$.
9: Obtain the Q-value and the target Q-value by using (2.13) and (2.15), respectively.
10: Update ϕ based on SGD algorithm.
11: Decrease ϵ.
12: Set $\phi^- = \phi$ at every C steps.
13: **end for**

since the consecutive experiences are highly correlated, which may lead to a slow convergence rate, unstable, or even divergence [33,103]. As such, the memory replay mechanism is adopted, where experiences are stored in a buffer $\mathbf{B}$, as illustrated in Figure 2.3. Then, at each time step, experiences are sampled uniformly at random to train the DNN. By doing so, correlations among experiences can be removed, thereby accelerating the learning process. Moreover, as one data point can be used multiple times to train the DNN, this mechanism can indeed improve the data usage efficiency.

In the proposed D3QL, since a DNN is used to approximate the Q-values, we define the input and output layers of the DNN according to the state- and action-space dimensions, respectively. Specifically, feeding a state s to the DNN will return Q-values for all actions at this state, each given by a neuron at the DNN's output layer. To improve the stability and increase the convergence rate, we propose to

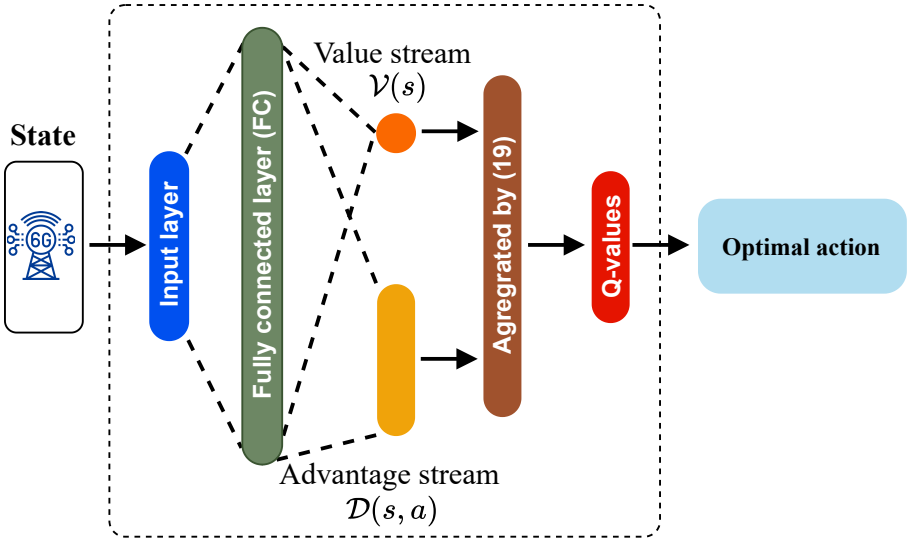


Figure 2.4 : The Deep Dueling Network Architecture.

use the state-of-the-art dueling neural network architecture [34] for D3QL's DNN, as shown in Figure 2.4. In particular, the dueling architecture divides the DNN into two streams. The first one estimates the state-value function $\mathcal{V}(s)$, which indicates the value of being at a state s. The second stream estimates the advantage function $\mathcal{D}(s,a)$ that demonstrates the importance of action a compared to other actions at state s.

Recall that the state-action value function $\mathcal{Q}(s,a)$ (namely Q-function) expresses the value of taking an action a at state $\mathbf{s}$, i.e., Q-value. Thus, the advantage function under policy π can be given as $\mathcal{D}^{\pi}(s,a) = \mathcal{Q}^{\pi}(s,a) - \mathcal{V}^{\pi}(s)$ [34]. Then, we can obtain the estimated Q-function for feeding state s to the Deep Dueling Neural Network (DDNN) by

$$\mathcal{Q}(s,a;\zeta,\beta) = \mathcal{V}(s;\zeta) + \mathcal{D}(s,a;\beta), \tag{2.11}$$

where ζ and β are the parameters of the state-value and advantage streams, respectively. It can be observed that given $\mathcal{Q}$, $\mathcal{V}$ and $\mathcal{D}$ could not be determined uniquely. For instance, $\mathcal{Q}$ is unchanged if $\mathcal{D}$ decreases the same amount that $\mathcal{V}$ increases. As such, using (2.11) directly may result in a poor performance of the algorithm.

Therefore, similar to [34], we propose to use the following output of the advantage stream:

$$\mathcal{Q}(\mathbf{s}, a; \zeta, \beta) = \mathcal{V}(\mathbf{s}; \zeta) + \Big(\mathcal{D}(s, a; \beta) - \max_{a' \in \mathcal{A}} \mathcal{D}(s, a'; \beta)\Big). \tag{2.12}$$

In this way, for an optimal action a^* at state s, i.e.,

$$a_s^* = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\ \mathcal{Q}(s, a; \zeta, \beta) = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\ \mathcal{D}(s, a; \beta),$$

the $\mathcal{Q}(s, a_s^*; \zeta, \beta)$ is forced to equal $\mathcal{V}(s, \zeta)$. However, (2.12) still faces an issue, i.e., the advantage function changes at the same speed at which the advantage of the predicted optimal action changes, making the estimation of Q-values unstable. To address this issue, the max operator is replaced by the mean as follows [34]:

$$\mathcal{Q}(s, a; \zeta, \beta) = \mathcal{V}(s; \zeta) + \Big(\mathcal{D}(s, a; \beta) - \frac{1}{|\mathcal{A}|} \sum_{a' \in \mathcal{A}} \mathcal{D}(s, a'; \beta)\Big). \tag{2.13}$$

The root of the overestimation problem in Q-learning and DQN comes from the max operation when estimating the target Q-value at time t as follows [31]:

$$\mathcal{Y}_t = r_t + \eta \max_{a_{t+1}} \mathcal{Q}(s_{t+1}, a_{t+1}), \tag{2.14}$$

where η is the discount factor indicating the importance of future rewards. To handle this issue, we adopt the deep double Q-learning algorithm [31] that leverages two identical deep dueling neural networks. One deep dueling neural network $\mathcal{Q}$ is for action selection, namely Q-network, and the other $\hat{\mathcal{Q}}$ is for action evaluation, namely target Q-network. Then, the target Q-value is computed by

$$\mathcal{Y}_t = r_t + \gamma \hat{\mathcal{Q}}\big(s_{t+1}, \underset{a}{\operatorname{argmax}}\ \mathcal{Q}(s_{t+1}, a; \phi_t,); \phi_t^-\big), \tag{2.15}$$

where ϕ and ϕ^- are the parameters of the Q-network and target Q-network, respec-

tively.

As the aim of training the Q-network is to minimize the gap between the target Q-value and the current estimated Q-value, the loss function at time t is given by [33]

$$\mathcal{L}_t(\phi_t) = \mathbb{E}_{(\mathbf{s},a,r,\mathbf{s}')}\left[\left(\mathcal{Y}_t - \mathcal{Q}(\mathbf{s}, a; \phi_t)\right)^2\right], \tag{2.16}$$

where $\mathbb{E}[\cdot]$ is the expectation according to a data point (s, a, r, s') in the buffer $\mathbf{B}$. To minimize the loss function $\mathcal{L}_t(\phi_t)$, this thesis proposes to use the Stochastic Gradient Descent (SGD), discussed in Section 2.1. Recall that SGD is one of the most popular algorithms for minimizing deep learning loss functions due to its simplicity in implementation [96]. Thus, it is well applicable for 6G networks, which often comprise of resource-constrained devices.

It is worth mentioning that even though the target Q-value $\mathcal{Y}_t$ in (2.16) looks like labels that are used in the supervised learning, $\mathcal{Y}_t$ is not fixed before starting the learning process. Moreover, it changes at the same rate as that of the target Q-network's parameters, possibly leading to instability in the learning process. To that end, instead of updating the parameters of $\hat{\mathcal{Q}}$ at every time step, ϕ^- is only updated by copying from ϕ at every C steps.

It is also important to note that the value of ϵ in the D3QL controls the environment exploration. Specifically, the higher the value of ϵ is, the more frequently the agent takes a random action. The reason for decreasing ϵ stems from the fact that an RL agent does not have complete information about its environment in advance. As such, in the beginning, the agent should explore its environment by taking actions randomly. By doing so, it can obtain information about the environment via the feedback of its actions, e.g., corresponding rewards and next states. Then, the agent adjusts its policy according to these experiences, i.e., the state, action, reward, and the next state. Therefore, to converge to an optimal policy, the agent should take

actions based on its policy more frequently than the random action [99], leading to the decrease of ϵ at every step.

2.2.5 Optimality and Computational Complexity Analysis

In RL, if a linear function approximates the value function, the learning process can be guaranteed to converge to the optimal policy [104]. Whereas, if a nonlinear function (e.g., a neural network) is used instead, it may not converge to the optimal one [104]. In our proposed learning algorithms, we adopt two innovative techniques (i.e., the dueling architecture and double Q-learning) and three transfer learning approaches to stabilize the learning process and improve the learning quality, thereby improving the convergence rate. Thus, even though the optimality of our deep reinforcement learning algorithm, i.e., D3QL, could not be proven theoretically, the intensive simulation results show that D3QL obtains a stable and superior performance compared to those of other approaches.

We now discuss the complexities of D3QL, which mainly depend on the training process of the Q-network. In the Q-network, there is one input layer L_i, one hidden layer L_h, and two output layers L_v and L_a corresponding to the state value and advantage streams, respectively. It is worth noting that training a DNN is typically carried out with matrix multiplication. Thus, the complexity of feeding a mini batch with size S_b to the Q-network is $\mathcal{O}\Big(S_b\big(|L_i||L_h| + |L_h||L_v| + |L_h||L_v|\big)\Big)$, where $|.|$ is the layer's size, i.e., a number of neurons in a layer. Given the training process takes T iterations, the complexity of the proposed algorithms are $\mathcal{O}\Big(TS_b\big(|L_i||L_h| + |L_h||L_v| + |L_h||L_v|\big)\Big)$.

In general, more computing resource is required to train a DNN, especially the one with a complex architecture. However, the DNN in our proposed algorithms is only contains a few (e.g., four) layers in which only one is fully connected (i.e., the hidden layer). Therefore, the decision time (i.e., the inference time of the Q-

network) is very marginal. In practice, while computational complexity presents a significant consideration in integrating ML into wireless devices, research indicates that with strategic optimizations, reduction techniques, and advancements in learning algorithms, ML can be effectively and efficiently implemented to enhance the capabilities of wireless devices across various applications. For instance, techniques like model pruning, model compression, and quantization become essential to navigate computational limits, making ML models suitable for resource-constrained IoT devices [105]. A few deep learning applications have been deployed in AVs, such as Tesla Autopilot [106] and ALVINN [107]. Thus, they are clear evidence of the effectiveness and efficiency of deploying machine learning algorithms on wireless devices in practice. Notably, transfer learning and meta-learning methods can significantly reduce the computational complexity of machine learning, such as deep learning and reinforcement learning, as seen in the next section.

2.3 Transfer Learning and Meta Learning

Although DL-based approaches can perform at compatible human level, it faces several shortcomings. First, the training process of DNN often requires a huge amount of collected data for the training process to achieve good performance [89]. Thus, this makes DL less efficient in practice, especially when data is expensive and contains noise due to the environment's dynamic and uncertainty, as in the 6G networks with massive connected heterogeneous devices. Second, conventional DL may face the over-fitting problem, i.e., performing excellently on training datasets but very poorly on test datasets, if the training dataset does not contain enough samples to represent all possible scenarios. Due to the dynamic nature of the wireless environment, the channel conditions may vary significantly over time. For example, a moving bus may change channel conditions from LoS to NLoS and vice versa. Consequently, real-time data may greatly differ from training data, making these above

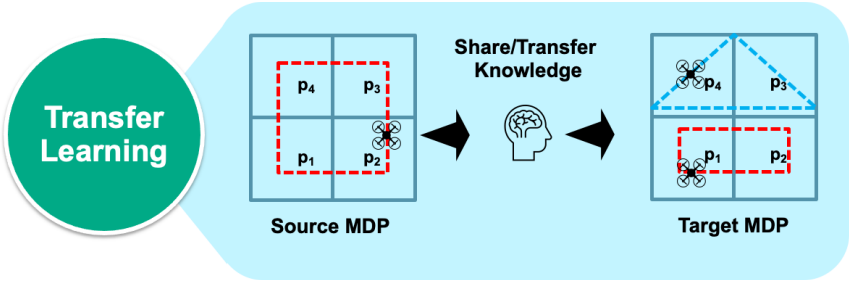


Figure 2.5 : An example of transfer learning in reinforcement learning.

problems more severe. In wireless communication systems, the necessity to retrain DNNs hinges on several key factors that reflect the dynamic nature of wireless environments. These factors can include changes in environmental conditions, network deployment and configuration changes, variations in traffic patterns and user mobility, and performance degradation. Third, wireless channel conditions can also be very different at different areas due to their landscapes, so the DL model trained at one area may not perform well at other areas. As such, different sites may need training different models from scratch, which is time-consuming and costly [90]. In this context, transfer learning and meta learning emerge as promising solutions to address the above problems.

2.3.1 Transfer Learning

Transfer learning is a method of leveraging knowledge obtained when performing a source task in a source domain to enhance the learning process of target tasks in target domains [32, 108–110]. In transfer learning, a model trained on a source task, called the source domain, is adapted or fine-tuned to work on a target task, known as the target domain. The idea behind transfer learning is to transfer the knowledge gained from the source domain to the target domain, thereby reducing the need for large amounts of data in the target domain and potentially improving the generalization and performance of the target model.

Typically, a domain contains labelled or unlabelled data given before the considered training process starts. However, data in RL is obtained via interactions between the agent and its surrounding environment. As a result, both the domain and task can be represented by an MDP, as shown in Figure 2.5. Note that transfer learning for DRL may look like supervised learning since they both use existing data, but they are very different. In particular, all DRL data used to train a DNN are unlabelled and on-the-fly data generated by interactions between an agent and its surrounding environment. Although the source data are collected in advance for the agent, they are just observations of the agent about the source environment, which do not have any label to indicate which action the agent should take. Thus, the agent in the target domain still needs learning algorithms (e.g., DQN) to learn the optimal policy gradually.

To measure the effectiveness of transfer learning, we can use three metrics, including the jump-start, asymptotic performance, and time-to-threshold [32]. In particular, jump-start measures how much the agent's performance at the beginning of the learning process can be improved by applying TL, while the asymptotic performance measures this improvement at the end of the learning process. The third metric, i.e., time-to-threshold, measures how fast TL can help the agent achieve a predefined performance level compared with the scenario without TL. It is worth highlighting that transfer learning cannot guarantee improvement in the learning curve. Notably, TL may even negatively impact the learning in the target domain if the transfer knowledge is not carefully chosen. Thus, Chapter 3 will explore the effectiveness of transfer learning that allows UAVs to "share" and "transfer" learning knowledge, thereby reducing the computational complexity and resource consumption for the UAV as well as significantly improving learning quality.

2.3.2 Meta-Learning

Meta-learning has an ability of learning to learn, i.e., self-improving the learning algorithm [90]. The main idea of meta-learning is to train the model on a collection of similar tasks, e.g., image classifications. By doing so, it enables the model to acquire generalization capabilities. Therefore, the trained model can quickly perform well in a new task only after a few update iterations, even when provided with a small dataset specific to the new task [91].

Generally, meta-learning comprises two nested loops:

- In the inner loop, a base-learning algorithm (e.g., SGD or Adam) solves task τ, e.g., image classification. The objective of the inner loop's learning is to minimize the loss $\mathcal{L}_\tau$. Note that a base-learning algorithm can be a typical DL algorithm, such as stochastic gradient descent (SGD). Thus, the inner loop is similar to the training process of the conventional DL algorithms. Let θ denote the DNN's parameters. After p steps of learning at the inner loop, the resulting parameters are denoted by $\bar{\theta} = U_\tau^p(\theta)$, where $U_\tau^p(\cdot)$ denotes the update operator.

- Then, at the outer loop, a meta-learning algorithm (e.g., Reptile [89] and MAMAL [91]) updates the model parameters as follows:

$$\theta \leftarrow \theta + \eta_o(\bar{\theta} - \theta), \tag{2.17}$$

  where η_o is the outer step size controlling how much the model parameters are updated. By doing so, the generalization in learning is improved.

Note that if $p = 1$, i.e., performing a single step of gradient descent in the inner loop, meta-learning becomes a joint training on the mixture of all tasks, which may not learn a good initialization for meta-learning [89]. In this thesis, Chapter 6 will

further examine the effectiveness of meta-learning when being used to help a DL-based signal detector quickly achieve good performance in new environments with minimal knowledge.

Given the above, transfer learning and meta-learning are both AI techniques that aim to improve the performance and generalization of models on new tasks. However, they differ in their objectives and approaches. Transfer learning focuses on transferring knowledge from a source task to a target task, while meta-learning aims to improve the model's learning process itself so that it can quickly adapt to new tasks with limited data. Both techniques have their unique strengths and applications and are essential tools in advancing machine learning capabilities in the 6G networks. However, their efficiency may vary depending on specific problems, and thus, this thesis will explore their applicability in addressing emerging challenges for 6G networks. A detail comparison between popular machine learning methods is presented in Table 2.1.

Table 2.1 : Summary of Machine Learning Types and Their Suitability for Wireless Networks

ML Type	**Advantages**	**Disadvantages**	**Suitability for Wireless Networks**
Deep Learning	• Excellent at handling large datasets. • Can model complex non-linear relationships.	• Require significant computational power. • Prone to over-fitting without sufficient data.	Effective for signal processing, image and speech recognition, and complex decision-making tasks in advanced wireless networks.
Reinforcement Learning	• Learn through interaction with the environment. • Can adapt to changing conditions.	• Require a lot of computational resources. • Learning can be slow.	Well-suited for dynamic resource allocation and network optimization in real-time.
Transfer Learning	• Reduce the need for large labelled datasets in new tasks. • Can leverage pre-trained models to accelerate learning.	• Performance heavily depends on the source and target domain similarity.	Useful for adapting models trained in one part of the network to similar tasks elsewhere, enhancing scalability and efficiency.
Meta-Learning	• Enable models to learn how to learn. • Improve the model's ability to adapt to new tasks with minimal data.	• Complex to implement and requires carefully designed meta-tasks for effective learning.	Highly suitable for dynamic environments where the network needs to quickly adapt to new conditions or tasks, such as predictive maintenance, anomaly detection, and optimizing network configurations based on evolving user demands or spectrum availability.

Chapter 3

Joint Speed Control and Energy Replenishment Optimization for UAV-assisted IoT Data Collection with Deep Reinforcement Transfer Learning

This chapter focuses on addressing a problem that hindering the application of UAVs, an important component of NTNs in 6G networks. In particular, we introduce a novel framework based on deep reinforcement transfer learning that can jointly optimize the speed and energy replenishment process of the Unmanned aerial vehicle (UAV). Recently, UAV-assisted data collection has been emerging as a prominent application due to its flexibility, mobility, and low operational cost. However, under the dynamic and uncertainty of IoT data collection and energy replenishment processes, optimizing the performance of UAV collectors is a very challenging task. Thus, this work introduces a novel framework that jointly optimizes the flying speed and energy replenishment for each UAV to significantly improve the overall system performance (e.g., data collection and energy usage efficiency). Specifically, we first develop a Markov decision process to help the UAV automatically and dynamically make optimal decisions under the dynamics and uncertainties of the environment. Although traditional reinforcement learning algorithms such as Q-learning and deep Q-learning can help the UAV to obtain the optimal policy, they often take a long time to converge and require high computational complexity. Therefore, it is impractical to deploy these conventional methods on UAVs with limited computing capacity and energy resource. To that end, we develop advanced transfer learning techniques that allow UAVs to "share" and "transfer" learning knowledge, thereby

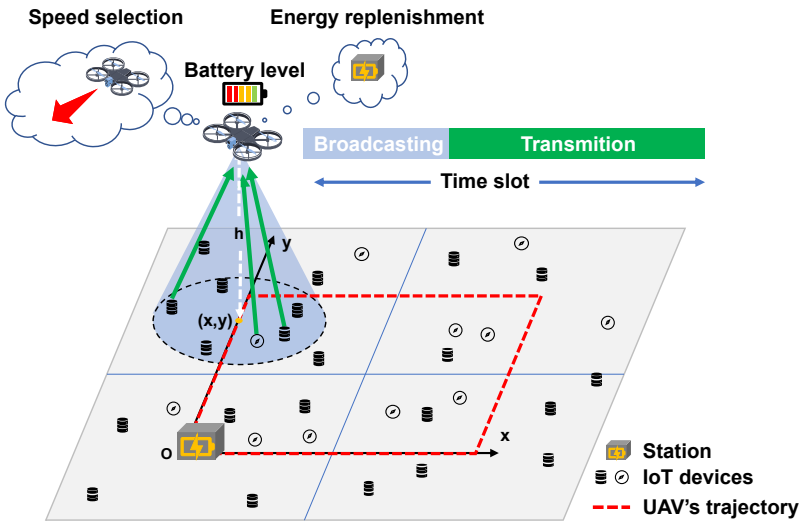


Figure 3.1 : System model for UAV-assisted IoT data collection network.

reducing the computational complexity and resource consumption for the UAV as well as significantly improving learning quality. As a result, our proposed solution is more applicable to aerial computing than conventional DRL approaches. Extensive simulations demonstrate that our proposed solution can improve the average data collection performance of the system up to 200% and reduce the convergence time up to 50% compared with those of conventional methods.

The rest of this chapter is structured as follows. The system model and operation control formulation are described in Sections 3.1 and 3.2, respectively. Section 3.3 presents the proposed learning algorithms. Then, the simulation results are analyzed in Section 3.4. Finally, Section 3.5 concludes this chapter.

3.1 System Model

In this work, we consider a UAV-assisted IoT data collection system where a UAV is deployed to collect IoT data over a considered area, as illustrated in Figure 3.1. We assume that the considered area is divided into N zones. The IoT devices are distributed randomly over these zones to execute various tasks, e.g., sensing temperature and humidity. In practice, the numbers of IoT nodes in these zones are

different due to different sensing demands in these zones. We consider time to be slotted (as in [28, 29]) with an equal duration, and each time slot is split into two consecutive intervals, broadcast and transmission, respectively. In the broadcasting interval, the UAV uses a dedicated channel to broadcast a wake-up signal [111] to all IoT nodes in its communication range. After acquiring this signal, these nodes will send their data to the UAV during this transmission interval. The use of a dedicated channel for wake-up signals is critical for reducing energy consumption among IoT devices, as it allows these devices to remain in a low-power state until they are activated by the wake-up signal. This strategy is particularly important for IoT deployments in remote or inaccessible areas where power sources may be limited or non-renewable. We assume that the communication link from IoT devices to the UAV adopts the OFDMA technique, while the communication link from the UAV to IoT devices uses the OFDM technique, as in [112]. In this way, the IoT devices can simultaneously transmit data to the UAV. This dual approach leverages the strengths of both OFDMA and OFDM to handle multiple access channels efficiently, thereby enhancing the throughput and reliability of the wireless communication system in dynamic and densely populated IoT environments. Let p_n denote the probability of a data packet successfully collected by the UAV in a time slot in zone n. Because the IoT nodes are distributed unevenly over N zones, p_n may vary over these zones. In the considered a UAV-assisted IoT data collection network, the UAV flies at a fixed altitude h (similar as that of [27, 28, 112–114]). Similar to the studies in [27–29], we assume that the UAV follows a predefined trajectory to sweep through all IoT devices of the system in each round.

However, unlike [27–29], we consider a more realistic scenario where the UAV is equipped with a battery that has limited energy storage. It is worth mentioning that the energy consumption for the wireless data collection process (i.e., broadcasting the wake-up signal and receiving data packets) is much lower than that of the

flying operation [26]. In addition, the decision of the UAV at each time slot (i.e., speed selecting or returning for energy replenishment) does not influence the power consumption for the wireless data collection process. Thus, our model can straightforwardly capture the communication power consumption by adding a constant to the UAV's energy consumption at each time slot, similar to that in [24]. To that end, in this work, we only focus on optimizing energy consumption for the UAV's flying operation, similar as that in [21, 26].

In a time slot, we assume that the UAV's velocity is constant (as in [27, 112]), but in different time slots the UAV can choose to fly at different speeds, e.g., $v_w = \{v_1, \ldots, v_A\}$. Each speed may cost a different amount of energy. For example, if the UAV flies faster, it may use more energy per time slot [26]. Note that UAVs in UAV-assisted IoT data collection systems often fly at a low speed to maintain the reliability of the data collection process.

When the UAV's energy is depleted, it will fly back to the charging station placed at a fixed location to replace the battery, as illustrated in Figure 3.1. Furthermore, during the flight, the UAV can decide to go back to the charging station to change the battery, for example, when it is near the charging station and its energy level is low. Once the energy replenishment process is accomplished, the UAV will fly back to its trajectory and continue its task. Here, we consider that the UAV has a maximum of E energy units for its operation (i.e., flying to collect IoT data) and a backup energy storage for flying back to the charging station for battery replacement. During the energy replenishment process, including back-and-forth flights and battery replacement, the UAV cannot collect data. Suppose that it takes the UAV t_f and t_b time slots to fly from its current location to the station and to replace the battery, respectively. Indeed, the battery replacement time, i.e., t_b, may be known in advance, while the return flight time, i.e., t_f, is highly dynamic depending on the distance between UAV's current location and the charging

station. In addition, t_f also depends on the return speed of the UAV, denoted by v_r. Assuming that UAV flies with a constant speed when returning to the station, the duration of energy replenishment is calculated by the equation $t_e = 2t_f + t_b$. Therefore, the energy replenishment process is also dynamic due to the dynamic of flying time t_f.

In practice, the surrounding environment is highly dynamic and uncertain. Specifically, the UAV does not know the probabilities of receiving a packet in different areas in advance, as they are very uncertain depending on sensing tasks. It is important to note that the UAV may collect more data when moving in a zone with a high probability of receiving packets, i.e., a high value of p_n. As a result, to maximize the data collection efficiency, the UAV must gradually learn this knowledge in order to adapt its operations accordingly, e.g., flying speed and energy level status. Moreover, the returning flight time depends on the distance between the UAV's current position and the station, which is highly dynamic. Therefore, if the UAV appropriately decides when to return for battery replacement (e.g., when it is near the station and its energy level is low), the energy replenishment time will be reduced significantly, resulting in high system performance. In contrast, if the UAV goes back to replace its battery when its energy level is high and it is far from the station, it will waste both time and energy, leading to low system performance. Thus, optimizing the UAV's operations to maximize the long-term system performance is a very challenging task. In the following sections, we will present our proposed learning algorithms that can effectively and quickly obtain the UAV's optimal operation policy under the limited energy of the UAV and the uncertainty of the data collection process.

3.2 Optimal Operation Control Formulation

To overcome the uncertainty and highly dynamic of the data collection and energy replenishment processes under the limited energy storage of the UAV, we for-

Table 3.1 : The list of notations.

Symbol	Description
$\mathcal{S}$	State Space
$\mathcal{A}$	Action Space
r	Immediate Reward Function
r_t^a	Speed Selection Reward Function at Time t
r_t^b	Battery Replacement Reward Function at Time t
E	Maximum Energy Capacity of UAV
p_n	Probability of Data Packet Collection in Zone n
v_w	Set of Possible Flying Speeds
v_r	Return Speed of UAV to Charging Station
t_f	Time to Fly from Current Location to Charging Station
t_b	Battery Replacement Time
t_e	Total Energy Replenishment Time
Ω	Working Reward for Data Collection
w_1, w_2	Weights in Reward Function
m_t^a	Energy Consumption for Action a at Time t
d_t^s	Number of Collected Data Packets at State s at Time t

mulate the UAV's operation control problem as the Markov decision process (MDP) framework. This MDP is determined by three components, state space $\mathcal{S}$, action space $\mathcal{A}$, and immediate reward function r. Based on the MDP framework, at each time slot the UAV can dynamically make the best actions (e.g., flying at appropriate speeds or returning for battery replacement) based on its current observations (i.e., its location and energy level) to maximize its long-term average reward without requiring complete information about data collection and energy replenishment processes in advance.

3.2.1 State Space

In this work, we aim to maximize the efficiency of collecting data and energy usage efficiency, and thus there are some important factors which we need to take into considerations. The first important factor is the current location of the UAV. The main reason is that the UAV's location can reveal important information about the expected amount of data that can be collected by the UAV and the time it takes if the UAV chooses to fly back to the station for battery replacement. Specifically,

the UAV will likely collect more data when moving in a zone with a high probability of receiving data than in a zone with a low probability of receiving data. In addition, the farther the distance between the UAV and the station is, the more time it takes to travel between these two positions. As mentioned above, the UAV always flies at a fixed altitude so that the UAV's position can be given by its 2D projection on the ground, i.e., (x, y) coordinates. The second crucial factor is the current UAV's energy level, denoted by e, which affects the decision of the UAV at every time slot. For example, the UAV should not select the battery replacement action (i.e., return the station to replace the battery) unless its energy level is low. Otherwise, it may waste time and energy for the flying back trip. To that end, this information is embedded into the state space of the UAV, which can be defined as follow:

$$\begin{aligned}\mathcal{S} = \Big\{(x, y, e) : x \in \{0, \ldots, X\}; y \in \{0, \ldots, Y\}; \\ \text{and } e \in \{0, \ldots, E\}\Big\} \cup \{(-1, -1, -1)\},\end{aligned} \tag{3.1}$$

where X and Y are the maximum x and y coordinates of the UAV's trajectory, and E is the maximum energy capacity of the UAV. As a result, the system state can be indicated by a tuple $s = (x, y, e) \in \mathcal{S}$. Moreover, because of the energy replenishment process, it is necessary to introduce a special state, i.e., $s = (-1, -1, -1)$. This special state is only visited when UAV's energy is depleted (i.e., UAV's state is $(x, y, 0)$) or if the UAV selects the battery replacement action. Then, after the energy replenishment process completes, the UAV will return to the previous position (where it decided to go back for battery replacement or where its energy is depleted) with a full battery, i.e., $s = (x, y, E)$. This design ensures that the system process is continuous, i.e., no terminal state.

3.2.2 Action Space

During the operation, to maximize the system performance in terms of energy usage and data collection efficiency, the UAV needs to not only choose the most suitable flying speed but also decide when to go back to the station to replace the battery. It is worth mentioning that given different states, the action spaces for these states may be different. For example, at a non-working state, i.e., $s = (-1, -1, -1)$, the UAV cannot select a flying speed. Instead, the possible action at this state is to stay "idle" until the UAV returns to its trajectory with a full battery. In other words, the UAV will stay at the non-working state after performing an "idle" action until the energy replenishment process completes. As a result, we can define the action space for the UAV as follows:

$$\mathcal{A} \triangleq \{a : a \in \{-1, 0, 1, \ldots, A\}\}, \tag{3.2}$$

where action $a = -1$ is to indicate the "idle" action and action $a = 0$ is to express that the UAV will choose to return to the station for replacing the battery, namely battery replacement action. Actions $a = \{1, \ldots, A\}$ are to represent the speed level that the UAV selects to fly at the current time slot. In addition, given the state $s \in \mathcal{S}$ the action space based on state s, i.e., $\mathcal{A}_S$, consists of all possible actions that are feasible at this state. Thus, we can express $\mathcal{A}_S$ as follows:

$$\mathcal{A}_S = \begin{cases} \{-1\}, & \text{if } s = (-1, -1, -1), \\ \{0, \ldots, A\}, & \text{otherwise.} \end{cases} \tag{3.3}$$

3.2.3 Reward Function

As discussed above, two main actions (i.e., flying speed and battery replacement actions) have significant effects on the system performance. Specifically, choosing

an appropriate flying speed at each time slot can maximize the efficiency of the collecting data process as well as energy usage. Alternatively, selecting the right time to return for battery replacement can reduce the energy replenishment time, thereby improving the overall system performance. For example, when the UAV is flying near the charging station and its energy is low, it should return to the charging station for battery replacement. Therefore, our proposed immediate reward function consists of (i) speed selection reward function, i.e., r_t^a, and (ii) battery replacement reward function, i.e., r_t^b, as follows:

$$r_t(s_t, a_t) = \begin{cases} r_t^a(s_t, a_t), & \text{if } a_t \in \{\mathcal{A} \setminus \{0\}\}, \\ r_t^b(s_t, a_t), & \text{if } a_t = 0, \\ 0, & \text{otherwise}, \end{cases} \tag{3.4}$$

where a_t is the selected action at time t.

3.2.3.1 Speed Selection Reward Function

Since our objective is to maximize the system performance by striking a balance between the data collection efficiency and energy usage efficiency, the speed selection reward function needs to capture this information. We define data collection efficiency as the number of collected data packets over a time slot and operation status of the UAV. For example, given a time slot, if the UAV is moving to collect data, it will receive a working reward, denoted by $\Omega > 0$. Otherwise, it will not receive the working reward, i.e., $\Omega = 0$. In this way, the working reward will encourage the UAV to collect data rather than return and then wait at the station for the battery replacement. The energy usage efficiency can be represented by the cost of choosing a flying speed, i.e., energy consumed by the UAV to fly at speed a during a time slot t. Clearly, the selected speed determines the energy consumption per time slot of the UAV, e.g., a low speed will cost the UAV less energy per time slot than that

of a high speed [26]. At time slot t, the cost of performing action a is denoted by m_t^a, and the number of collected data packets at the current state s is denoted by d_t^s. Thus, the speed selection reward function can be expressed by:

$$r_t^a(s_t, a_t) = \Omega + w_1 d_t^s - w_2 m_t^a, \quad \text{if } a_t \in \mathcal{A} \setminus \{0\}, \tag{3.5}$$

where w_1 and w_2 are the weights to balance between collected data and energy consumption of the UAV. It is worth noting that these weights can be defined in advance based on the service provider's requirements. For example, in case if the data is more important and valuable than energy, we can set the value of w_1 to be higher than that of w_2. In contrast, if the energy is scarce, we can set the value of w_1 to be lower than that of w_2. Therefore, the speed selection reward function can capture the UAV's data collection efficiency and energy consumption efficiency.

3.2.3.2 Battery Replacement Reward Function

Although the speed selection reward function gives the UAV sufficient information to learn the optimal speed control, it cannot help the UAV learn when it is good to return to the station for the battery replacement. First, if the UAV performs the battery replacement action, the values of Ω and d_t^s in (3.5) will be zero, leading to a negative value of r_t^a. Consequently, the UAV may consider this action as a bad choice and will not choose it in the future. Second, the speed selection reward function is unable to guide the UAV to learn where is good to return for replacing its battery, e.g., the further from the station the UAV is, the smaller reward for battery replacement action it may receive. Therefore, battery replacement action needs a different reward function, which needs to take into account of both the UAV's current energy level, i.e., e, and the distance between its current position and the station, i.e., l. However, in practice, it is challenging to design such a reward function for battery replacement action because the complex relationship between e and

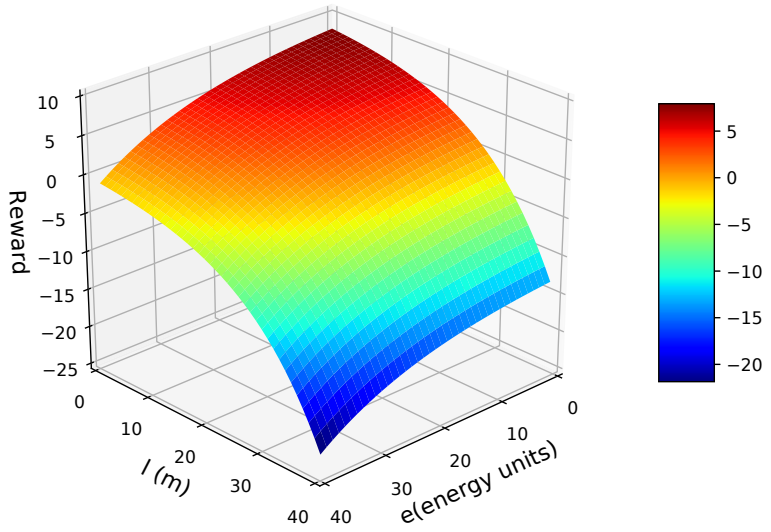


Figure 3.2 : An example of the proposed battery replacement reward function.

l makes more difficulties for the UAV to decide whether it should go back or keep flying to collect data. In particular, the UAV may choose the battery replacement action if both e and l are small, while the UAV should continue its collection task if any of these factors is large. To that end, in the following, we propose a reward function for battery replacement action that can address this problem, and to the best of our knowledge, this is the first work in the literature addressing the battery replacement problem for UAV-assisted IoT data collection networks.

Suppose the UAV decides to return for battery replacement at time t, at which its energy and the distance to the station are e_t and l_t, respectively. Then, its immediate reward is derived from the battery replacement reward function as follows:

$$r_t^b(s_t, a_t) = c - w_3 \exp(w_4 e_t) - w_5 \exp(w_6 l_t), \quad \text{if } a_t = 0, \tag{3.6}$$

where c is a constant controlling the maximum value of r_t^b, which may affect the learning policy of the UAV. For example, if c is smaller than the smallest value of r_t^a, the value of returning reward function is always lower than that of the speed

selection reward function, making the return action always "worse" than those of the speed selection actions. The second and third terms in (3.6) express the influence of the energy level e_t and the distance l_t to the UAV's decision for battery replacement. The tradeoff between the energy level e_t and the current distance l_t is controlled by four weights, i.e., w_3, w_4, w_5, w_6. Figure 3.2 demonstrates an example of the proposed battery replacement reward function in which $c = 10$, $w_3 = w_5 = 1$, $w_4 = 0.06$, and $w_6 = 0.08$. It can be observed that as e and l are large (e.g., greater than 36 and 17, respectively), the UAV will receive negative rewards, meaning that the UAV is not encouraged to return to replace its battery if its energy is high or it is far from the station. In this way, this function will encourage the UAV to return to the station for replacing the battery when its current energy and distance to the station are small.

3.2.4 Optimization Formulation

In this work, our aim is to maximize the average long-term reward function by finding a UAV's optimal policy π^*, i.e., $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$. In particular, given the UAV's current energy and location, π^* determines an action that maximizes the long-term average reward function as follows:

$$\max_{\pi} \quad \mathcal{R}(\pi) = \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\big(r_t(s_t, \pi(s_t))\big), \tag{3.7}$$

where $\mathcal{R}(\pi)$ is the long-term average reward obtained by the UAV according to the policy π, $\pi(s_t)$ is the selected action at state s at time slot t based on policy π, and $r_t(s_t, \pi(s_t))$ is the immediate reward by following policy π at time slot t. Thus, the optimal policy π^* will assist the UAV in dynamically making the optimal action according to its current observation, i.e., its position and energy level. Moreover, Proposition 1 shows the optimality of the proposed immediate reward function $r_t(s_t, a_t)$.

Proposition 1. *"There always exists a maximum value of the immediate reward function."*

Proof: We first prove the optimality for each component of the immediate reward function, i.e., r_t^a and r_t^b. From (3.5), the speed selection reward function is a linear function of energy consumption m_t^a and an amount of collected data packet d_t^s. Since the values of m_t^a and d_t^s are always positive and finite, there always exists a maximum value of the speed selection reward function $r_t^a(s_t, a_t)$. For the second component, i.e., $r_t^b(s_t, a_t)$, we can rewrite it as follows:

$$r_t^b(s_t, a_t) = c - \big(w_3 \exp(w_4 e_t) + w_5 \exp(w_6 l_t)\big), \quad \text{if } a_t = 0. \tag{3.8}$$

We now prove that $f(e_t, l_t) = (w_3 \exp(w_4 e_t) + w_5 \exp(w_6 l_t)$ is convex. Specifically, the Hessian matrix of $f(e_t, l_t)$ is calculated as follows:

$$H = \begin{pmatrix} w_3 w_4^2 exp(w_4 e_t) & 0 \\ 0 & w_5 w_6^2 exp(w_6 l_t) \end{pmatrix}. \tag{3.9}$$

Then, the determinant of H is computed by: $det_H = w_3 w_4^2 exp(w_4 e_t) w_5 w_6^2 exp(w_6 l_t)$. Since the weights in the immediate function are always positive, we have $det_H > 0$. Therefore, H is positive definite and $f(e_t, l_t)$ is convex, meaning that there always exists a minimum value of $f(e_t, l_t)$. As a result, $r_t^b(s_t, a_t)$ always has a global optimal value. Given the above, there always exists a maximum value of the immediate reward function as its components are optimality and independent.

3.3 Optimal Operation Policy for UAVs with Deep Reinforcement Transfer Learning

In the considered problem, the UAV does not have complete information about the surrounding environment in advance (e.g., the data arrival probabilities and the

charging process) to obtain the optimal policy. In addition, the limited resource of UAV requires a low computational complexity solution. Thus, Q-learning and D3QL algorithms, discussed in Section 2.2.2, can be used to learn the optimal policy for the UAV.

Although D3QL can effectively address the shortcomings of Q-learning, it still poses some drawbacks inherited from conventional DRL when addressing scenarios with high sample complexity, as the considered problem in this work where the surrounding environment of the UAV is unknown in advance. Firstly, it often takes a lot of time to train DNN, e.g., DQN's training time is up to 38 days for each Atari game [33]. If the environment dynamics (such as the data arrival process) or the trajectory of the UAV changes, the DNN may need to be fine-tuned or even retrained from scratch, yielding a high computational complexity. Consequently, it is unable to deploy on a UAV that has very limited energy and computing resources. Secondly, since the UAV should only return for replacing its battery when it is close to the station or its energy level is low, it needs sufficient experiences in this region, especially when flying over the station. However, as the UAV flies over its fixed trajectory, experiences obtained from this region are often very small compared with all obtained over the entire considered area. Therefore, the UAV may not have adequate information to learn an optimal policy. To address these challenges, we develop a novel framework leveraging transfer learning techniques, namely Deep Dueling Double Q-learning with Transfer Learning (D3QL-TL).

3.3.1 Transfer Learning in Reinforcement Learning

Transfer learning is a method of leveraging knowledge obtained when performing a source task in a source domain to enhance the learning process of target tasks in target domains [32, 108–110]. Typically, a domain contains labelled or unlabelled data given before the considered training process starts. However, data in RL is

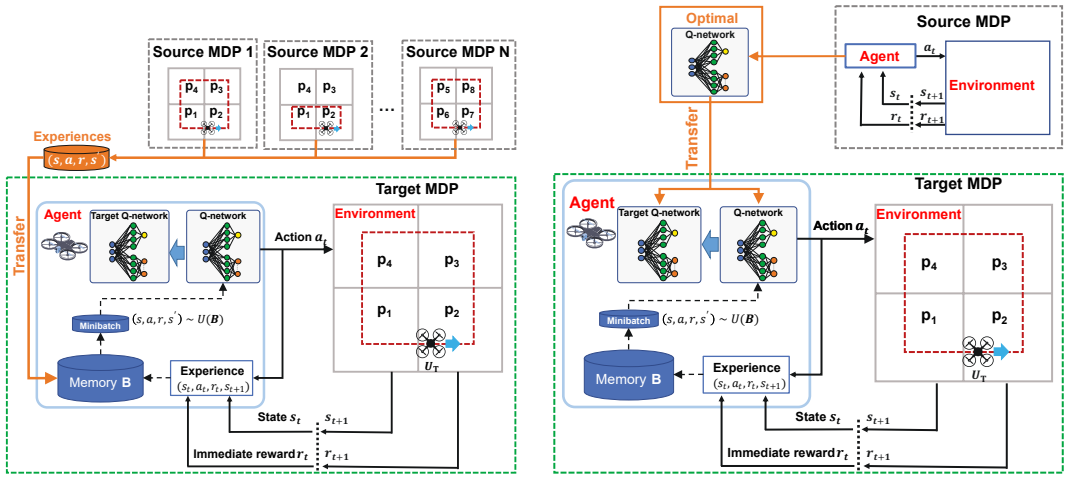
obtained via interactions between the agent (i.e., the UAV) and its surrounding environment. As a result, both the domain and task can be represented by an MDP. Thus, we can define transfer learning in RL as in Definition 3.1 [109].

Definition 3.1. *Transfer learning in RL: Suppose the source and target MDPs are defined. Transfer learning (TL) in RL intends to leverage the knowledge $\mathcal{K}_{\mathrm{S}}$ obtained from the source MDP, i.e., the policy, the environment dynamics, and the data, as a supplement to the target MDP's information $\mathcal{K}_{\mathrm{T}}$ to efficiently learn the target optimal policy π_{T}^{*} as follows:*

$$\pi_{\mathrm{T}}^{*} = \underset{\pi_{\mathrm{T}}}{\operatorname{argmax}} \mathbb{E}_{s\sim\mathcal{S}_{\mathrm{T}},a\sim\pi_{\mathrm{T}}} \left[\mathcal{Q}^{\pi_{\mathrm{T}}}(s,a)\right], \tag{3.10}$$

where π_{T} is a target MDP's policy approximated by an estimator, e.g., a table or a DNN, that is trained on both $\mathcal{K}_{\mathrm{S}}$ and $\mathcal{K}_{\mathrm{T}}$.

Note that transfer learning for DRL may look like supervised learning since they both use existing data, but they are very different. In particular, all DRL data used to train a DNN are unlabelled and on-the-fly data generated by interactions between an agent (e.g., the UAV) and its surrounding environment. Although the source data are collected in advance for the target UAV, they are just observations of the source UAV about the source environment, which do not have any label to indicate which action the UAV should take. Thus, the target UAV still needs learning algorithms (e.g., D3QL) to learn the optimal policy gradually. Our proposed TL can help the UAV utilize the knowledge from the source domain to avoid bad decisions at the beginning of the learning process when the UAV is exploring the environment by taking a random action, thereby improving the learning rate and learning quality. Recall that to measure the effectiveness of TL, we can use three metrics, including jump-start, asymptotic performance, and time-to-threshold, as discussed in Section 2.3



(a) Experience transfer (b) Policy transfer

Figure 3.3 : The proposed D3QL-TL based models.

It is worth highlighting that TL cannot guarantee improvement in the learning process. It may even negatively impact the learning in the target MDP if the transfer knowledge is not carefully chosen. Thus, in the following sections, we propose a transfer learning framework that can reduce the learning time and learning quality for D3QL.

3.3.2 Deep Dueling Double Q-learning with Transfer Learning

The details of D3QL-TL are presented in Algorithm 3.1. In particular, as illustrated in Figure 3.3, we consider a UAV U_{S} working in an IoT data collection environment formulated by the MDP framework $\mathcal{M}_{\mathrm{S}}$. Then, the knowledge of U_{S} can be leveraged to help a new UAV U_{T} effectively learn the optimal policy for working in another environment formulated by the MDP framework $\mathcal{M}_{\mathrm{T}}$. In practice, $\mathcal{M}_{\mathrm{T}}$ can be the same or different with $\mathcal{M}_{\mathrm{S}}$. Moreover, the transferring knowledge can be in the form of the policy and/or experiences of the source UAV, i.e., U_{S}.

To that end, D3QL-TL defines three types of knowledge transferring as follows:

- *Experience Transfer (ET)*: This approach aims to leverage a set of experiences

Algorithm 3.1 The D3QL-TL

Establish memory $\mathbf{B}$ and ϵ.
Establish Q-network $\mathcal{Q}$ with random parameters ϕ, and target Q-network $\hat{\mathcal{Q}}$ with parameters $\phi^{-} = \phi$.
if Experience transfer **then**
    Copy the experience set of source MDP $\mathcal{M}_{\mathrm{S}}$ to $\mathbf{B}$.
else if Policy transfer **then**
    Re-initialize $\mathcal{Q}$ and $\hat{\mathcal{Q}}$ with the parameter of source Q-network's parameters ϕ_{S}.
else if Hybrid transfer **then**
    Re-initialize $\mathcal{Q}$ and $\hat{\mathcal{Q}}$ with the parameter of source Q-network's parameters ϕ_{S}.
    Copy the experience set of source MDP $\mathcal{M}_{\mathrm{S}}$ to $\mathbf{B}$.
end if
for *step = 1 to T* **do**
    Choose action a_t according to the ϵ-greedy policy (2.10).
    Execute a_t, observe reward r_t and next state s_{t+1}.
    Save experience (s_t, a_t, r_t, s_{t+1}) in $\mathbf{B}$.
    Sample mini-batch of experiences randomly from $\mathbf{B}$, i.e., $(s, a, r, s') \sim U(\mathbf{B})$.
    Calculate Q-value $\mathcal{Q}(s_k, a_k; \phi)$ and target Q-value $\mathcal{Y}_t$ by (2.13) and (2.15), respectively.
    Take a gradient descent step with respect to the parameters of Q-network.
    Decrease the value of ϵ.
    Set $\hat{\mathcal{Q}} = \mathcal{Q}$ at every C steps.
end for

E_{S}, in which each element is an experience tuple $< s, a, r, s' >$, obtained in the source MDP, i.e., $\mathcal{M}_{\mathrm{S}}$, to improve the learning process of the target UAV, i.e., U_{T}, working in the target MDP, i.e., $\mathcal{M}_{\mathrm{T}}$. Specifically, E_{S} is first copied to the memory buffer of the target UAV. Then, these transferred experiences and target UAV's new experiences are used to train the Q-network. In this manner, the target UAV can quickly get adequate information, and thereby significantly improving the learning speed. In addition, the quality of the experiences also affects the learning process. For example, an experience does not have much value if it is easy to be obtained by the target UAV. In contrast, an experience is considered to be valuable if it is hard to obtain and highly impacts the system performance. For example, experiences obtained when the

UAV is near the station may have high values because they not only contain information about environment dynamics (i.e., packet arrival probabilities) but also may reveal value information about the right time to take the battery replacement action.

- *Policy Transfer (PT)*: This approach directly transfers the policy of a source UAV to a target UAV. In particular, U_{T} starts the learning process with the policy of U_{S}, which is represented by the Q-network of U_{S}, called the source Q-network. Hence, the U_{T}'s Q-network is initialized by the source Q-network parameters ϕ_{S}. Then, the U_{T}'s Q-network is trained with the new experiences of U_{T} obtained in the target MDP $\mathcal{M}_{\mathrm{T}}$. Thus, starting with the source policy can help U_{T} to avoid random decisions caused by the randomness of action selection at the beginning of the learning process, e.g., inappropriately choosing the battery replacement action.

- *Hybrid*: This approach aims to leverage the benefits of both experience and policy transfer types. Particularly, the hybrid scheme can improve not only the jump-start but also the asymptotic performance.

Note that the efficiency of each transferring technique depends on the relationship between the source and the target MDPs. For example, if the source and target MDPs are very similar, policy transfer may yield a better result in terms of convergence rate than that of the experience transfer technique. In contrast, when the source and target MDP are not similar, e.g., differences in environment dynamics, the experience transfer should be a better choice.

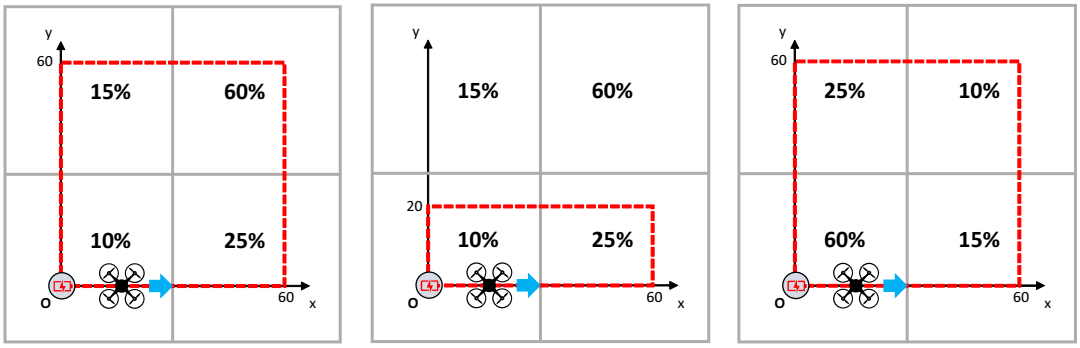
Parameters	Ω	w_1	w_2	c_1	c_2	c_3
Value	1	1	0.3226	5	0.5	0.5
Parameters	c_4	c_5	E	$\mathbf{p}$	t_b	v_r
Value	0.022	0.2	300	$[0.1, 0.25, 0.6, 0.15]$	10	1

Table 3.2 : Simulation parameters.

3.4 Performance Evaluation

3.4.1 Parameter Setting

We first evaluate our proposed approaches in an IoT system, in which a UAV flies over a predefined trajectory in a considered area to collect IoT data, as in [27–29]. This area is divided into four zones such that the UAV's travel distance in each zone equals 60 m, as illustrated in Figure 3.4(a). The station is located at the origin, i.e., $(0, 0)$. The probabilities of packet arrival in these zones are given by a vector $\mathbf{p} = [p_1, p_2, p_3, p_4]$, e.g., p_1 is the packet arrival probability when the UAV is flying over zone 1 and so on. Since the UAV collects data while flying, it often flies at a low speed (e.g., 5 m/s) to maintain the reliability of the data collection process [28]. Therefore, we consider that the UAV has three speeds: $1, 3,$ and 5 (m/s). The UAV consumes 2, 3, and 4 (energy units/time slot) when flying at $1m/s$, $3m/s$, and $5m/s$, respectively. Note that our proposed MDP framework and learning algorithms can help the UAV learn the optimal policy according to its observation (i.e., its current position and energy level) regardless of the UAV's specifications, i.e., its available speeds and corresponding energy consumption. Here, the energy unit is used to quantify an amount of energy (as in [115, 116]), and energy consumption values are only to demonstrate how much energy the UAV uses at each time slot for its operation. In practice, UAVs manufactured by different brands may have different specifications, but the proposed algorithm still can obtain the optimal policy for the UAV. The parameters of the reward function are provided in Table. 3.2.



(a) $\mathcal{M}_{\mathrm{S}}$ (b) $\mathcal{M}_{\mathrm{T}}^{1}$ (c) $\mathcal{M}_{\mathrm{T}}^{2}$

Figure 3.4 : (a) Source MDP, (b) the first target MDP, and (c) the second target MDP.

The settings for our proposed algorithms are set as follows. For the ϵ-greedy policy, ϵ is first set at 1, then gradually decreased to 0.01. For all the proposed algorithms, the discount factor is set at 0.9. In Q-learning, the learning rate β is 0.1. The architectures of Q-network and target Q-network are illustrated in Figure 2.4. We use typical hyperparameters for training DNN, e.g., the learning rate and the frequency update of $\hat{\mathcal{Q}}$ are set to 10^{-4} and 10^{4}, respectively, as those in [33, 95].

For the experience transfer approach in D3QL-TL, experiences are selected to be transferred based on their valuable information. Recall that if the UAV performs the battery replacement action when it is far from the station, it always receives a very small reward compared with those of other actions, as in (3.4). Therefore, the UAV's experiences in this area can imply that it should not take the battery replacement action. In contrast, experiences obtained when the UAV is near the station can contain both information, i.e., when it not worth to return for charging (e.g., current energy level is high) and when it is worth to take the battery replacement action (e.g., current energy level is low). Thus, we choose experiences obtained when the UAV flies near the station to be transferred.

We study a scheme where the UAV does not have complete information about the surrounding environment in advance, e.g., the data arrival probabilities and the

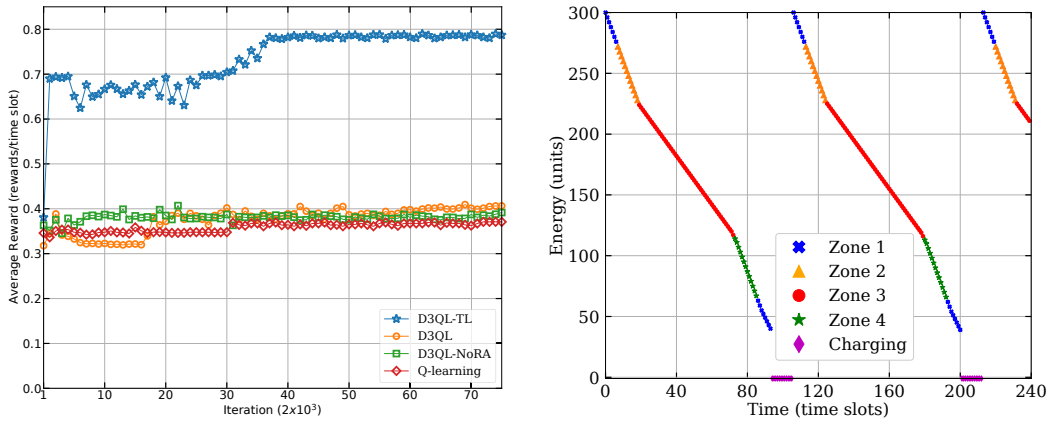
energy replenishment process. Hence, we compare our approach with three other deterministic policies, i.e., the UAV always flies at (1) lowest speed ($1m/s$), (2) middle speed ($3m/s$), and (3) highest speed ($5m/s$). In addition, the Q-learning and D3QL are selected as the baseline methods to demonstrate the effectiveness of our proposed TL techniques since they are widely used algorithms and the most recent advanced techniques in RL, respectively. Moreover, to investigate impacts of the battery replacement action on the system performance, we consider an approach, namely D3QL-NoRA, where the D3QL will still be implemented on the UAV, but the UAV will not select the battery replacement action.

3.4.2 Simulation Results

In the simulation, we first evaluate the performance of our proposed learning algorithm, i.e., D3QL-TL, by examining the convergence rate and the obtained policy. Then, we evaluate the system performance when varying some important parameters (e.g., battery replacement time, UAV's energy capacity, return speed, and packet arrival probabilities) to assess their influences on the system performance. For the D3QL-TL, the experience transfer type is chosen because it can leverage the experiences obtained during the learning phases of other algorithms, i.e., D3QL and Q-learning. Finally, to gain more insights into the effectiveness of three transferring types in D3QL-TL, we compare their performance in different scenarios, i.e., changing the UAV trajectory and the probabilities of receiving a packet.

3.4.2.1 Convergence and Policy

In Figure 3.5 (a), we compare the convergence of proposed algorithms in terms of average rewards. At the beginning of the learning processes, the average rewards of proposed approaches are close to each other, approximately 0.35. However, only after $4,000$ iterations, D3QL-TL's average reward is nearly 170% greater than those of other approaches. Then, D3QL-TL almost converges to the optimal policy after



(a) Convergence rate of proposed algorithms (b) Policy obtained by D3QL-TL

Figure 3.5 : Convergence rate and policy.

7.5×10^4 iterations, and its average reward becomes stable at around 0.78, which is more than 190% greater than those of other learning algorithms. Interestingly, D3QL and D3QL-NoRA converge to policies that achieve similar average rewards. This suggests that D3QL is unable to take advantage of the battery replacement action. In other words, D3QL cannot effectively handle this complicated decision-making situation. This result implies the outperformance of our proposed algorithm, i.e., D3QL-TL, compared with other methods when addressing the extremely complex problem as the one considered in this study.

Next, we show the policy obtained by D3QL-TL after 1.5×10^5 iterations in Figure 3.5 (b). In particular, a point indicates the energy level of the UAV at the beginning of a time slot. The slope of a straight line between two points indicates the selected action, e.g., the steeper this line is, the higher speed is selected. Generally, the lowest speed is selected in a zone that has a high probability of successfully collecting a packet. In contrast, the highest speed is selected in a zone that has a low probability of receiving a packet, as shown in Figure 3.5 (b). Given $\mathbf{p} = [0.1, 0.25, 0.6, 0.15]$, as in Table. 3.2, the D3QL-TL selects the lowest speed in zone 3, and the highest speed in zones 1, 2, and 4. More interestingly, the UAV experiences

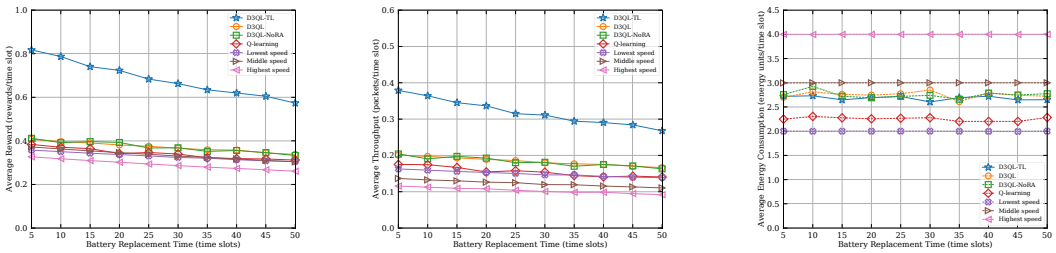
the middle and high speeds in zone 1. In particular, it travels at the highest speed until its energy decreases to 55 energy units at time slot 88, then the middle speed is selected. When its energy drops to 40 energy units at time slot 93, equivalent to 13.3% energy level, the UAV takes the battery replacement action. Note that Figure 3.5 (b) also reveals information of the location where the UAV should take the battery replacement action by energy replenishment time, i.e., t_e. Particularly, given $t_e = 12$ time slots, the location that the UAV decides to return only $1m$ away from the station. This result demonstrates the impacts of location and energy level on the optimal policy of the UAV.

3.4.2.2 Performance Analysis

In this section, we perform simulations to evaluate our proposed algorithms in terms of average reward, throughput, and system energy consumption. The parameters are set to be the same as those in 3.4.2.1. The policies of proposed learning algorithms, including Q-learning, D3QL, and D3QL-TL, are obtained after 1.5×10^5 iterations.

In Figure 3.6, we vary the battery replacement time, i.e., t_b. Clearly, the average reward and throughput of all policies decrease as the battery replacement time increases from 5 to 50 time slots. This is stemmed from the fact that given a fixed duration, the less time the UAV needs to replace the battery, the more time it can spend collecting data. As a result, the data collection efficiency of the system reduces. It can be observed that D3QL-TL can significantly outperform other approaches in terms of average reward and throughput, while it still obtains a reasonable energy consumption per time slot. In particular, the average reward and throughput achieved by D3QL-TL are up to 200% and 185% greater than those of the second-best policy, i.e., D3QL, respectively.

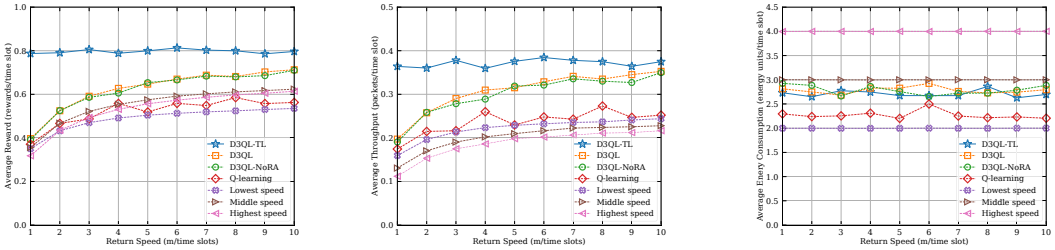
Next, we vary the return speed v_r of the UAV to observe the system's perfor-



(a) Average reward (b) Average throughput (c) Average energy consumption

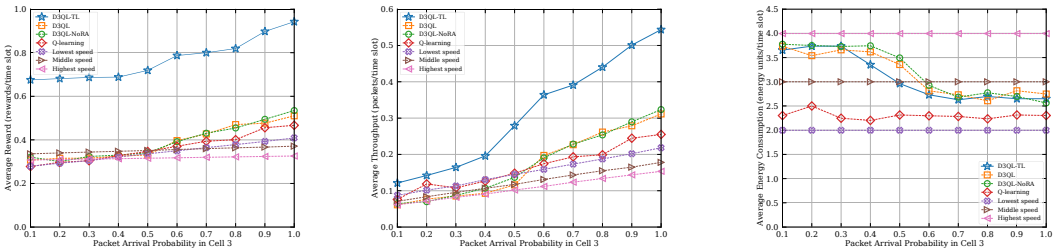
Figure 3.6 : Vary battery replacement time.

mance in terms of average reward, throughput, and energy consumption. Figures 3.7 (a) and (b) show that as the return speed v_r increases from $1(m/s)$ to $10(m/s)$, all policies have upward trends in terms of the average reward and throughput, except that of the D3QL-TL. The reason is that the increase of v_r leads to the decrease of time for returning to the station, i.e., t_f, and thus the UAV has more time to collect data in a fixed duration. More interestingly, when the return speed is low (e.g., lower than $3(m/s)$), the lowest speed policy obtains a higher average reward than that of the highest speed policy, as observed in Figure 3.7 (a). Nevertheless, the lowest speed policy obtains the lowest performance (i.e., the average reward is lowest) when the return speed is large. This emanates from the fact that given a fixed serving time, the energy consumption of the highest speed is higher than that of the lowest speed. Therefore, the UAV has to replace its battery more frequently if it flies at the highest speed rather than if it flies at the lowest speed. Consequently, as the return speed increases, the highest speed policy eventually performs better than that of the lowest speed policy. Unlike other policies, D3QL-TL achieves a stable average reward, approximately 0.8, that is always much higher than those of other policies, as shown in Figure 3.7 (a). This is because the UAV equipped with D3QL-TL can learn an excellent policy, e.g., taking battery replacement action when the UAV is close to the station, making it more adaptable to the changes of the return speed.



(a) Average reward (b) Average throughput (c) Average energy consumption

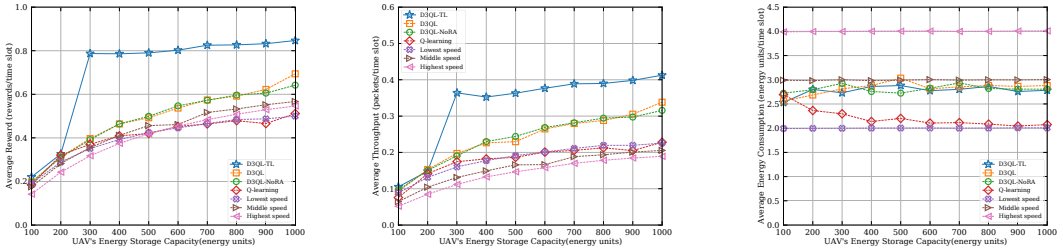
Figure 3.7 : Vary the UAV's return speed.



(a) Average reward (b) Average throughput (c) Average energy consumption

Figure 3.8 : Vary the packet arrival probability of zone 3.

We then vary the packet arrival probability p_3 of zone 3 while those of other zones are unchanged, as provided in Table 3.2, and observe the performance of our proposed approaches. Figures 3.8 (a) and (b) clearly show the increase of average rewards and throughputs for all policies when p_3 increases from 0.1 to 1.0. Interestingly, as shown in Figure 3.8 (a), when p_3 is small, e.g., lower than 0.4, the lowest speed policy obtains lower rewards than those of the highest speed. However, when p_3 becomes larger, the lowest speed achieves a higher average reward than that of the highest speed. This implies that the UAV should fly at the lowest speed if the packet arrival probability is high, and vice versa. Figure 3.8 (c) demonstrates that our proposed algorithm, i.e., D3QL-TL, can learn the environment's dynamic, e.g., the packet arrival probability. In particular, when the probability of receiving a packet is low, e.g., less than 0.4, the UAV's average energy consumption is high, approximately 3.65 energy units/time slot, indicating that the highest speed is se-



(a) Average reward (b) Average throughput (c) Average energy consumption

Figure 3.9 : Vary the UAV's energy capacity.

lected more frequently than the lowest speed. In contrast, when this probability becomes higher, e.g., larger than 0.5, the UAV's energy consumption decreases to around 2.6, implying that the lowest speed is the most frequently selected speed. To that end, D3QL-TL can leverage this knowledge to consistently obtain the best performance compared with other policies.

Finally, in Figure 3.9, we vary the UAV's energy storage capacity E to study its impact on the system performance. In particular, when E is varied from 100 to 1000 energy units, the average rewards and throughputs of all policies increase, as shown in Figure 3.9 (a) and (b), respectively. It is worth highlighting that if E is small, e.g., less than 500, the performance of the lowest speed is better than that of the highest speed, as illustrated in Figure 3.9 (a). However, the highest speed outperforms the lowest speed when E is larger than 500. This is due to the fact that when E is small, the UAV's battery has to be replaced more frequently, leading to a downgrade of the UAV's data collection efficiency. Thus, the UAV must conserve more energy by flying at the lowest speed. By balancing between energy usage efficiency and data collection efficiency, our proposed D3QL-TL approach can always achieve the highest performance compared to other policies.

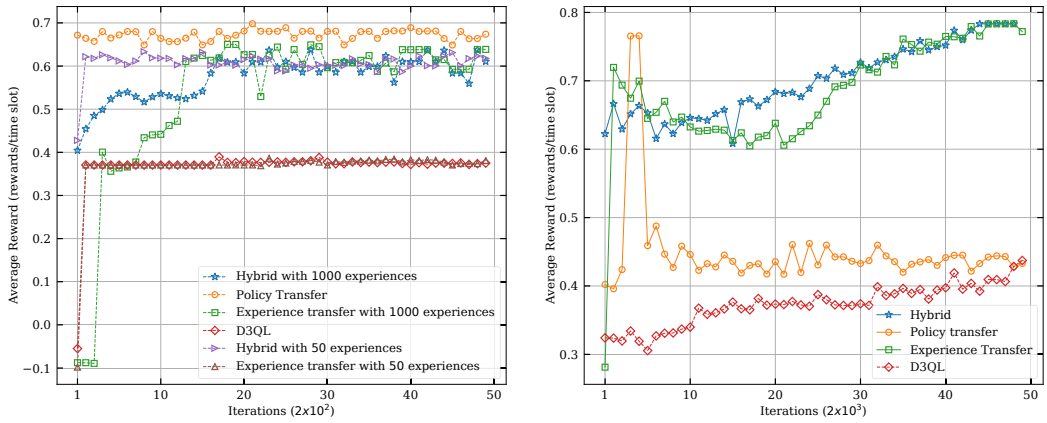
3.4.2.3 Transfer Learning Strategies

In this section, we evaluate and compare the effectiveness of three TL types in D3QL-TL (i.e., experience transfer (ET), policy transfer (PT), and hybrid transfer) in different scenarios, as shown in Figure 3.4 (b) and (c). In particular, the source MDP, i.e., $\mathcal{M}_{\mathrm{S}}$ is defined as the MDP described in Section 3.4.1, and the simulation parameters are also provided in Table. 3.2. Then, the optimal policy obtained by D3QL-TL after $1.5{\times}10^5$ iterations and the UAV's experiences gathered in the source MDP are leveraged to reduce the learning time and learning quality. To gain an insight of when and how much these transfer learning techniques can improve the learning process of D3QL, we define two scenarios as follows:

- In the first scenario (illustrated in Figure 3.4 (b)), the target MDP (i.e., $\mathcal{M}_{\mathrm{T}}^1$) is the same as $\mathcal{M}_{\mathrm{S}}$ except the trajectory. The UAV only flies over two zones, i.e., zone 1 and 2. In each zone, it travels $80m$.

- In the second scenario (illustrated in Figure 3.4 (c)), the difference between target MDP (i.e., $\mathcal{M}_{\mathrm{T}}^2$) and $\mathcal{M}_{\mathrm{S}}$ is the probabilities of receiving a packet, i.e., $\mathbf{p} = [0.6, 0.15, 0.1, 0.25]$.

We first compare the convergence rate of several transfer learning schemes of D3QL-TL and D3QL in the first scenario in Figure 3.10 (a). To investigate how experiences impact on the learning process of D3QL-TL, we select two sizes of experience sets, which are 50 and 100. As shown in Figure 3.10 (a), after $5{\times}10^4$ learning iterations, the average rewards obtained by all types of D3QL-TL can achieve up to 179% greater than that of D3QL, except the ET with the size of 50. For the ET group, when the experience size is small, e.g., 50, transfer learning does not improve the learning process of D3QL since the average reward of ET is similar to that of D3QL. However, when this size is large enough, e.g., 10^3, the asymptotic



(a) The first scenario (b) The second scenario

Figure 3.10 : Convergence of the proposed TL schemes.

performance of D3QL-TL is 171% greater than that of D3QL in terms of average reward. Interestingly, for the hybrid approach, when the experience size decreases, the D3QL-TL's performance increases. Especially, the PT, which is equivalent to the hybrid with zero experience size, consistently outperforms other transfer approaches in all the metrics, i.e., jump-start, time-to-threshold, and asymptotic performance. Specifically, only after 10^3 iterations, the PT obtains the optimal policy, and its average reward is stable at around 0.68. Thus, the PT can help the UAV to reduce the learning time up to 50% compared with those of other approaches. These results demonstrate that if the environment's dynamics in the target MDP, e.g., probabilities of receiving a packet, are similar to those in the source MDP, PT is the best choice. The reason is that the change of trajectory makes source experiences less efficient than that of the source policy. It is worth noting that only schemes with PT can improve the system performance at the beginning of the learning process because this policy can help the UAV choose valuable actions in this period, e.g., selecting battery replacement action when it is near the station and its energy level is low.

In Figure 3.10 (b), we show the results of the second scenario where the packet

arrival probabilities are different from that of the source MDP. In this scenario, we set the experience size to 10^3 for the ET and hybrid schemes. Again, it can be observed that only schemes with PT can improve the initial performance, e.g., during the first 2×10^3 iterations. Unlike the first scenario, the PT yields the worst performance among the transfer learning schemes, and its asymptotic performance is almost zero, meaning that there is no improvement in terms of average reward at the end of the learning process. In contrast, the ET and hybrid schemes achieve similar asymptotic performance, approximately 172%. However, hybrid's jump-start metric, i.e., 192%, is significantly higher than that of ET, i.e., 86%. These results suggest that when the environment dynamics change, the hybrid scheme should be selected.

Lessons learned: The above results demonstrate that applying TL to DRL is not straightforward. If it is not carefully implemented, TL may not be able to improve the performance of DRL, e.g., the ET with 50 experiences in the first scenario and the PT in the second scenario. In the first case, the change in UAV's trajectory means that the UAV still works in the same environment, but it follows another path. Therefore, the PT can quickly achieve the best policy because it can directly improve the performance of the UAV instead of gradually transferring the source knowledge in the ET. On the other hand, the probability of successfully collecting a packet changes in the second scenario, meaning that the UAV is placed in a different environment. In this context, the hybrid transfer technique can obtain the highest performance since it can leverage both the policy and the important experiences in the source MDP.

3.5 Conclusions

In this chapter, we develop a novel Deep Dueling Double Q-learning with Transfer Learning algorithm (D3QL-TL) that jointly optimizes the flying speed and energy

replenishment activities for the UAV to maximize the data collection performance of a UAV-assisted IoT system. The proposed algorithm effectively addresses not only the dynamic and uncertainty of the system but also the high dimensional state and action spaces of the underlying MDP problem with hundreds of thousands of states. In addition, the proposed TL techniques (i.e., experience transfer, policy transfer, and hybrid transfer) allow UAVs to "share" and "transfer" their learned knowledge, resulting in a decrease of learning time and an improvement of learning quality. The simulation results show that our proposed solution can significantly improve the system performance (i.e., data collection and energy usage efficiency) and has a remarkably lower computational complexity compared with other conventional approaches. In the next chapter, we will explore an emerging technology, i.e., integrated communication and sensing (ICAS), that enables the ubiquitous sensing capability of 6G networks.

Chapter 4

AI-enabled mm-Waveform Configuration for Autonomous Vehicles with Integrated Communication and Sensing

This chapter focuses on the Integrated Communications and Sensing (ICAS) technology that plays a critical role in enabling 6G systems to become ubiquitous sensors [1]. Additionally, ICAS also emerges as a promising solution for Autonomous Vehicles (AVs), a use case of 6G [2], where sensing and data communications are two important functions that often operate simultaneously. For ICAS application to AVs, optimizing the waveform structure is one of the most challenging tasks due to strong influences between sensing and data communication functions. Specifically, the preamble of a data communication frame is typically leveraged for the sensing function. As such, the higher number of preambles in a Coherent Processing Interval (CPI) is, the greater the sensing task's performance is. In contrast, communication efficiency is inversely proportional to the number of preambles. Moreover, surrounding radio environments are usually dynamic with high uncertainties due to their high mobility, making the ICAS's waveform optimization problem even more challenging. To that end, this chapter presents our proposed solution that can automatically optimize the waveform configuration for the ICAS of autonomous vehicles to strike a balance between these functions. In particular, we develop a novel ICAS framework established on the Markov decision process and recent advanced techniques in deep reinforcement learning. By doing so, without requiring complete knowledge of the surrounding environment in advance, the ICAS-AV can adaptively optimize its waveform structure (i.e., number of frames in the CPI) to maximize

sensing and data communication performance under the surrounding environment's dynamic and uncertainty. Note that the optimal waveform is not fixed, instead, the ICAS-AV needs to optimally adapt its waveform configuration (i.e., the number of preambles and their locations in a CPI) according to its surrounding environment and system requirements. Since the IEEE 802.11ad is leveraged for the ICAS system in this work, the optimized waveform has the same characteristics as the IEEE 802.11ad. According to [117], the waveform used in the IEEE 802.11ad system is represented as a generalized form of the Gaussian family. Extensive simulations show that our proposed approach can improve the joint communication and sensing performance up to 46.26% compared with other baseline methods.

The content of this chapter is organized as follows. Sections 4.1 and 4.2 introduce the ICAS system model and the problem formulation, respectively. Then, the Q-learning-based and the proposed i-ICS algorithms are proposed in Section 4.3. In Section 4.4, simulation results are analyzed. Finally, we conclude this chapter in Section 4.5.

4.1 System Model

In this work, we consider an autonomous vehicle, namely ICAS-AV, that is equipped with an intelligent millimeter wave integrated communication and sensing (mm-Wave ICAS) system based on the IEEE 802.11ad SC-PHY specification. Note that this work leverages the preamble in the SC-PHY for sensing task since it is similar to those in Control Physical Layer (C-PHY) and OFDM Physical Layer (OFDM-PHY), making the proposed solution easily to be extended to other physical layer types in IEEE 802.11ad. Moreover, according to [118], the OFDM-PHY is obsolete and may be removed in a later revision of the IEEE 802.11ad standard. The ICAS-AV maintains a communication link at a wavelength ζ with a vehicle AV_X, called the recipient vehicle. Let d_X and v_X denote the distance and the relative

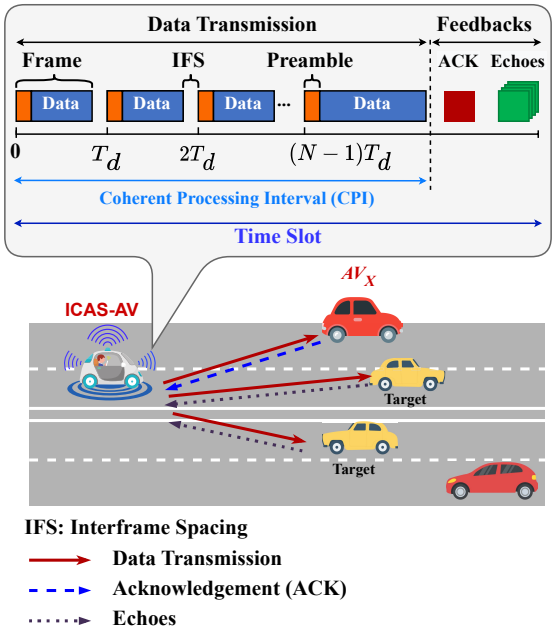


Figure 4.1 : The ICAS system model in which the ICAS-AV maintains a data communication with AV_X based on IEEE 802.11ad. At the same time, the ICAS-AV senses its surrounding environment by utilizing echoes of its transmitted waveforms.

speed between ICAS-AV and AV_X, respectively. At the same time, the ICAS-AV gathers echoes of transmitted signals from surrounding targets (e.g., moving vehicles $AV_1, \ldots, AV_X$) to perform the sensing function, as depicted in Figure 4.1. This work assumes that time is divided into equal slots. The time slot is small enough so that the velocities of targets can be considered constant in a time slot [38, 44].

In this work, similar to [38] and [119], the waveform structure is defined by the number of frames and their locations in the Coherent Processing Interval (CPI). Note that the maximum target's relative velocity v_{max} can only be explicitly estimated when frames are located at specific locations in the CPI time [38]. Specifically, the n-th frame is located at nT_d (as illustrated in Figure 4.1). Here, $n \in [0, 1, \ldots, N-1]$ and $T_d \leq 1/(2\Delta f_{max})$ is a sub-Doppler Nyquist sampling interval with a maximum Doppler shift $\Delta f_{max} = 2v_{max}/\zeta$ [120]. The rationale of these setting is that preambles of IEEE 802.11ad frames can act analogously as pulses in the pulsed radar in which pulses are repeated after a Pulse Repetition Interval (PRI) [120]. At the

beginning of a time slot, the ICAS-AV decides the mm-Wave ICAS waveform structure (i.e., the number of frames in the CPI) that will be used to transmit data in this time slot. Then, it observes feedback from the receiving vehicle AV_X (i.e., the acknowledgment of frame) and echoes signals from its surrounding targets at the end of this time slot.

The ICAS-AV has a data queue with a maximum of Q packets, each with B Bytes. When a new packet arrives, it will be stored in the data queue if this queue is not full. Otherwise, this packet will be dropped. The packet arrival is assumed to follow the Poisson distribution with the mean λ packets per time slot. Note that the 802.11ad frame has a varying data field; therefore, each frame can contain one or multiple packets. In the following, we first elaborate the proposed ICAS transmit and receive signal models, then discuss the sensing processing and ICAS performance metrics.

4.1.1 Signal Models

4.1.1.1 Transmitted Signal Model

In IEEE 802.11ad, the SC-PHY frame consists of a fixed-size preamble and a varying length data field. The preamble contains multiple Golay sequences whose Ambiguity Function (AF) exhibits an ideal auto-correlation without a side-lobe along the zero Doppler axis, making it perfect to be utilized for sensing function, e.g., range estimation and multi-target detection [44]. However, its AF is very susceptible to Doppler shifts, leading to a poor velocity estimation. To that end, multi-frame processing is proposed to address this problem [38, 44]. By doing so, the preambles across frames act as radar pulses in a Coherence Processing Interval (CPI). This work considers an ICAS waveform structure that consists of N IEEE 802.11ad frames in a CPI. The IEEE 802.11ad system can recognize this aggregated frame as the block/no acknowledgment policy [121].

Table 4.1 : Table of Notations

Notation	Description
Q	Maximum number of packets in the data queue (Packets)
B	Size of each packet (Bytes)
N	Number of frames in a CPI
n	Frame index in the Coherent Processing Interval (CPI)
ζ	Wavelength of the communication link (m)
d_X	Distance to the recipient vehicle AV_X (m)
v_X	Relative speed between ICAS-AV and AV_X (m/s)
v_{max}	Maximum target's relative velocity (m/s)
T_d	Sub-Doppler Nyquist sampling interval (s)
Δf_{max}	Maximum Doppler shift (Hz)
λ	Mean packet arrival rate per time slot (Packets/time slot)
G_c	Large-scale path loss in data communication
G_{TX}, G_{RX}	Antenna gains of transmitter and receiver
σ_c^2	Variance of the complex white Gaussian noise
σ_o^b	Radar cross-section of a scattering center (m^2)
G_b^o	Large-scale channel gain for a scattering center
Δv	Velocity resolution (m/s)
δ	Velocity measurement accuracy (RMS error) (m/s)
$\mathcal{S}$	State Space
$\mathcal{A}$	Action Space
r	Immediate Reward Function
w_1, w_2, w_3	Weights in the reward function r

The maximum target's relative velocity, denoted by v_{max}, can only be explicitly estimated when frames are located at specific locations in the CPI time [38]. Specifically, the n-th frame is located at nT_d (as illustrated in Figure 4.1), where $n \in [0, 1, \ldots, N-1]$ and $T_d \leq 1/(2\Delta f_{max})$ is a sub Doppler Nyquist sampling interval with a maximum Doppler shift $\Delta f_{max} = 2v_{max}/\zeta$ [120]. Note that the desired ICAS system performance can be achieved by optimizing the ICAS waveform parameters (e.g., the number of frames in the CPI), which will be described in more details in Section 4.1.3. The transmit signal model is then defined as follows. Let $s_n[k]$ denote the symbol sequence corresponding to n-th transmitted frame with K_n symbols. Then, the complex-baseband continuous time of transmitted signal in the

CPI can be given by [38, 122]:

$$x(t) = \sum_{n=0}^{N-1} \sum_{k=0}^{K_n-1} s_n[k] g_{TX}(t - kT_s - nT_d), \tag{4.1}$$

where $g_{TX}(t)$ is the unit energy pulse shaping filter at the transmitter of ICAS-AV and T_s is the symbol duration. In this study, similar to [38, 123], we consider a single data stream model where the adaptive analog beamforming can be applied to achieve higher directionality beamforming. The use of multiple antennas in mmWave communication systems facilitates beamforming techniques that can dynamically focus energy towards specific directions. This not only improves communication link quality but also enhances sensing capabilities by increasing the signal-to-noise ratio (SNR) of received signals from objects of interest. By leveraging the spatial diversity and increased aperture provided by multiple antenna elements, a system can more accurately determine the angle of arrival (AoA) and angle of departure (AoD) of signals. This is particularly beneficial in environments where high precision is required for tracking or locating objects. Thus, the above received signals corresponding to communication and sensing functions can be modelled in the following subsections.

4.1.1.2 Received Signal Models

Data communication received signal Suppose that the mmWave communication link between AV_X and ICAS-AV is established, the large-scale path loss is defined as follows [41]:

$$G_c = \frac{G_{TX} G_{RX} \lambda^2}{8\pi^2 d_X^2}, \tag{4.2}$$

where G_{TX} and G_{RX} are the antenna gains of the transmitter (TX) and the receiver (RX), respectively. After beamforming, symbol and frequency synchronization phases, the received communication signal is the composition of P_c attenuated and delayed versions of $x(t)$. Thus, the received communication signal corresponding

to symbol k in the n-th frame can be represented as follows [38]:

$$y_n^c[k] = \sqrt{G_c} \sum_{p=0}^{P_c-1} \beta_c[p] s_n[k-p] + z_n^c[k], \tag{4.3}$$

where $z_n^c[k]$ is the complex white Gaussian noise with zero mean and variance σ_c^2, i.e., $\mathcal{N}_C(0, \sigma_c^2)$, and $\beta_c[p]$ is the small-scale complex gain of the p-th path. Note that we assume that $\beta_c[p]$ is independent and identically distributed (i.i.d) $\mathcal{N}(0, \sigma_p^2)$ where $\sum_{p=0}^{P_c-1} \sigma_p^2 = 1$, as in [38, 44]. The SNR of the communication channel is defined as $SNR_c \triangleq E_s G_c / \sigma_c^2$, where E_s is the energy per symbol of the transmitted signal.

Sensing received signal Similar to [38, 44, 124], this work uses the scattering center representation to describe the sensing channel. We consider that there are O range bins, and at the o-th range bin there are B_o scattering centers (i.e., targets). A scattering center (d_o, b) can be defined by its distance d_o, velocity v_o^b, radar cross-section σ_o^b, round-trip delay $\tau_o = d_o/c$ with c being the speed of light, and Doppler shift $\Delta f_o^b = 2v_o^b/\lambda$. The large-scale channel gain corresponding to a scattering center (d_o, b) can be given as follows [38, 44, 124]:

$$G_o^b = \frac{G_{TX} G_{RX} \lambda^2 \sigma_o^b}{64\pi^3 d_o^4}. \tag{4.4}$$

As in [38,125], only a target whose d_o is large in comparison with the distance change during the CPI (i.e., $d_o \geqq v_o^b T_{CPI}$) is considered, so the small-scale channel gain β_o^b can be assumed to be constant during the CPI. Thus, the received sensing signal model corresponding to symbol k in the n-th frame can be given as follows [38]:

$$y_n[k] = \sum_{o=0}^{O-1} \mathcal{E}_n^o[k] \sum_{b=0}^{B_o-1} \sqrt{G_o^b} e^{-j2\pi\Delta f_o^b (kT_s + nT_d)} + z_n^s[k], \tag{4.5}$$

where $z_n^s[k] \sim \mathcal{N}_C(0, \sigma_n^2)$ is the complex white Gaussian noise of the sensing channel, and $\mathcal{E}_n^o[k]$ is the delayed and sampled Matched Filtering (MF) echo from the o-th range bin, i.e., $\mathcal{E}_n^o[k] = \sum_{i=0}^{K_n-1} s_n[i]g((k-i)T_s - nT_d - \tau_o)$, where $g(t) = g_{TX}(t) * g_{RX}(t)$ is the net TX-RX pulse shaping filter.

In this study, similar to [38, 119, 125], we assume that the channel is stationary during the preamble period of a frame due to the small preamble duration. As such, the received signal model corresponding to the preamble $\mathcal{E}_t^o[k]$ of a frame can be given as follows [38]:

$$y_n^t[k] = \sum_{o=0}^{O-1} \mathcal{E}_t^o[k] \sum_{b=0}^{B_o-1} \sqrt{G_o^b} e^{-j2\pi\Delta f_o^b n T_d} + z_n^s[k]. \tag{4.6}$$

Note that kT_s can be omitted from the phase shift term in the signal model corresponding to the preamble part since the channel is assumed to be time-invariant within the preamble period.

4.1.2 Sensing Signal Processing

We now discuss sensing signal processing in the ICAS system. Based on the cross-correlation output between the transmitted preamble of a single 802.11ad frame and the received signal, the ICAS system can detect a target with high probability (more than 99.99%) and achieve the desired range resolution (i.e., 0.1 m [40]) for automotive LRR [44]. However, the AF of IEEE 802.11ad preamble is sensitive to Doppler shift, making it less accurate in velocity estimation. Therefore, this work only considers the velocity estimation of the ICAS system, which is more challenging to obtain a high accuracy than those of target detection and range estimation processes.

After detecting targets and obtaining the corresponding range bins, the velocity estimation can be executed as follows. Given the n-th frame received in (4.6),

the sensing channel corresponding to detected targets at the o-th range bin can be expressed as follows [38]:

$$h_o^n = \sum_{b=0}^{B_o-1} u_o^b e^{-j2\pi\Delta f_o^b n T_d} + z_o^n, \tag{4.7}$$

where $u_o^b = \gamma\sqrt{E_s G_o^b}$ is the signal amplitude, γ is the correlation integration gain, and z_o^n is the complex white Gaussian noise $\mathcal{N}_C(0, \sigma_n^2)$. Then, the channel vector corresponding to the o-th range bin for N frames in the CPI, i.e., $\mathbf{h}_o = [h_o^1, h_o^2, \ldots, h_o^{N-1}]$, can be given as follows:

$$\mathbf{h}_o = \mathbf{D}_o \mathbf{u}_o + \mathbf{z}_o, \tag{4.8}$$

where $\mathbf{z}_o = [z_o^0, z_o^1, \ldots, z_o^{N-1}]$ is the channel noise vector, $\mathbf{u}_o \triangleq [u_o^0, u_o^1, \ldots, u_o^{B_o-1}]^T$ denotes the channel signal amplitude vector, $\mathbf{d}(v_o^b) \triangleq [1, e^{-j2\pi\Delta f_o^b T_d}, \ldots, , e^{-j2\pi\Delta f_o^b (N-1) T_d}]^T$ is the vector of channel Doppler corresponding to b-th velocity at o-th range bin, and $\mathbf{D}_o \triangleq [\mathbf{d}(v_o^0), \mathbf{d}(v_o^1), \ldots, \mathbf{d}(v_o^{B_o-1})]$ is the matrix of channel Doppler. The target velocity can be estimated based on (4.8) by using Fast Fourier transform (FFT)-based algorithms that are widely used in the classical radar processing [44, 120].

4.1.3 ICAS Performance Metrics

As discussed in the previous subsection, this work focuses on the target velocity estimation. Thus, the sensing performance for the ICAS can be determined by the velocity estimation accuracy (i.e., velocity resolution). For the FFT-based velocity estimation approach, the velocity resolution is defined by [44]:

$$\Delta_v = \frac{\zeta}{2N_f T_d}, \tag{4.9}$$

where N_f is the number of frames used for the velocity estimation process. Then, the velocity measurement accuracy can be characterized by the root mean square

error that depends on the SNR of the received sensing signal as follows [40, 126]:

$$\delta = \frac{\zeta}{2N_f T_d \sqrt{2SNR_r}}. \tag{4.10}$$

Equation (4.10) implies that given a fixed CPI time, a fixed T_d, and a constant data rate, the velocity estimation accuracy increases (i.e., δ decreases) as the value of N_f increases. Recall that frames are placed at consecutive multiples of T_d in the CPI. Therefore, the last frame's size is larger than those of others if the total number of frames is less than the maximum number of frames in the CPI. As such, as the number of frames in the CPI increases, the size of the last frame decreases since the CPI duration is constant.

The communication metrics for the ICAS must represent the data transmission performance. Two typical communication metrics are the transmission rate and reliability (e.g., packet loss). Thus, this work considers two metrics, 1) the length of the data queue representing the efficiency of data transmission, and 2) the number of dropped packets demonstrating the system reliability.

Note that unlike the system models in [38, 42, 46] that only consider static environments, in this work, we consider a more practical dynamic environment with many uncertainties, such as the wireless channel quality and the number of arrival packets. Compared with [38, 42, 46], such dynamics make the waveform optimization problem more challenging. First, the wireless channel is dynamic and uncertain due to the high mobility of the vehicle, resulting in time-varying successful data transmission probability. Second, since the transmission demand of the user as well as the vehicle's autonomous system may change over times, the number of packet requests arriving at the system may be different at different times. Therefore, the data arrival process at the ICAS-AV is also highly dynamic. As a result, due to these uncertainties and dynamics, optimizing the waveform structure without completed

knowledge of the surrounding environment is a practical yet very challenging task.

It is also worth noting that a large frame has a higher drop probability than that of a smaller frame in the same wireless environment. Thus, using multiple small-size frames can increase not only the reliability of transmission but also the sensing performance (i.e., the velocity estimation accuracy). However, it leads to an overhead because it increases the number of preambles that do not contain user data. Consequently, packets pile up at the ICAS-AV, leading to packet drop in the queue. In addition, the characteristics of the ICAS-AV's surrounding environment (e.g., the wireless channel quality, which can be represented through the packet drop probability and SNR, and the packet arrival rate) highly influence the ICAS performance in regard to data transmission reliability and sensing accuracy. Therefore, the ICAS system needs to obtain an optimal policy that optimizes the ICAS waveform (i.e., the number of frames in the CPI) to achieve the desired performance in terms of data transmission and sensing accuracy. Furthermore, the ICAS-AV's surrounding environment may change significantly from time to time, especially in the ICAS environment where AVs usually travel. Thus, optimizing the ICAS waveform in each time slot is an intractable problem. The following sections will describe our proposed MDP framework for the ICAS operation problem that enables the ICAS-AV to quickly and effectively learn the optimal policy without requiring complete information from the surrounding environment, thereby achieving the best performance compared with traditional solutions.

4.2 Problem Formulation

The ICAS-AV's operation problem is formulated using the MDP framework to deal with the highly dynamic and uncertain of the surrounding environment. The MDP framework can help the ICAS-AV adaptively decide the best action (i.e., the ICAS waveform structure) based on its current observations (i.e., the current data

queue length and channel quality) at each time slot to maximize the ICAS system performance without requiring complete knowledge on the packet arrival rate, data transmission process, and channel quality in advance. An MDP is generally defined by the state space $\mathcal{S}$, the action space $\mathcal{A}$, and the immediate reward function r. The following sections will discuss more details about the components of our proposed framework.

4.2.1 State Space

As we aim to maximize the performance of the ICAS system with regard to data transmission efficiency and sensing accuracy, we need to consider the following key factors. The first one is the current data queue length (i.e., the number of packets in the data queue) because it reflects the efficiency of the data transmission process. For example, given a packet arrival rate, the lower the number of packets in the queue is, the higher the data transmission efficiency is. The second one is the link quality that can be estimated by using the SNR metric at the ICAS-AV. At the beginning of a time slot, the link quality is estimated based on the feedback (i.e., the recipient vehicle's ACK frame and targets' echoes) of the transmitted frame in the previous time slot. Although it does not represent the instantaneous channel state, it help to provide valuable information about the surrounding environment.

In this work, we consider that the channel quality level can be grouped into C different classes, which are analogous to the Modulation and Coding Scheme (MCS) levels in IEEE 802.11ad [121]. These classes have different probabilities of bit errors due to the different wireless channel qualities, denoted by a probability vector $\mathbf{p_e} = [p_1, p_2, \ldots, p_C]$. Note that given the transmission link with bit error probability p_b, the error probability of an F-bit frame can be calculated by $p_f = 1-(1-p_b)^F$ [127]. In addition, if a frame drops, all packets in this frame will be lost. To that end, the

ICAS's state space can be given as follows:

$$\mathcal{S} = \Big\{(q, c) : q \in \{0, \ldots, Q\}; c \in \{0, \ldots, C\}\Big\}, \tag{4.11}$$

where q is the current number of packets in the data queue and c is the channel quality. Here, Q is the maximum number of packets that the data queue can store. In this way, the system state can be represented by a tuple $s = (q, c)$. By this design, the ICAS system continuously operates without falling into the terminal state.

4.2.2 Action Space

As discussed in Section 4.1.3, the ICAS waveform plays a critical role in the system performance. In particular, given a fixed CPI time T, using a large number of frames results in a high reliability of data transmission and a high sensing accuracy. However, it reduces the efficiency of data transfer as there are more overhead data. At each time slot, the ICAS-AV needs to select the most suitable ICAS waveform structure (i.e., the number of frames in the CPI) to maximize the system performance. Thus, the action space can be defined as follows:

$$\mathcal{A} = \big\{1, \ldots, N\big\}, \tag{4.12}$$

where N is the maximum number of frames in the CPI. Recall that the beginning of each frame needs to be placed at the multiple of T_d consecutively, and thus $N = \lfloor \frac{T_{CPI}}{T_d} \rfloor$, where T_{CPI} is the CPI time and $\lfloor \cdot \rfloor$ is the floor function. As such, if the number of frames in the CPI selected by the ICAS-AV is less than N, the last frame will be longer than others. Note that when the data queue is empty, the ICAS-AV can still send dummy frames (e.g., frames whose data fields contain random bits) to maintain the ICAS's sensing function continuously.

4.2.3 Reward Function

Since the ICAS system performs two functions simultaneously, i.e., data transmission and sensing, we aim to maximize the ICAS system performance by balancing the data transmission efficiency and the sensing accuracy. Thus, the reward function needs to capture both of them. The data transmission efficiency can be defined according to the number of packets waiting in the queue and the number of dropped packets. Specifically, the lower the number of packets in the data queue and the number of dropped packets are, the higher the efficiency of the ICAS system is. Suppose that at time slot t, the ICAS-AV observes state s_t and takes action a_t. Let q_t, δ_t, and l_t denote the current size of the data queue, the sensing accuracy, and the number of dropped packets that the ICAS-AV observes at the end of t, respectively. Then, the immediate reward function can be defined as follows:

$$r_t(s_t, a_t) = -(w_1 q_t + w_2 \delta_t + w_3 l_t), \tag{4.13}$$

where w_1, w_2, and w_3 are the weights to tradeoff between the number of packets waiting in the queue, the sensing accuracy, and the number of dropped packets due to the data queue full. In practice, these weights should be determined carefully based on the system requirements as well as the environment characteristics, e.g., wireless channel quality and/or data arrival rate at the transmitter of an ICAS-AV. For example, in a system requiring high sensing accuracy, these weights can be set such that the sensing accuracy contributes the largest share in the reward function. Note that the value range and unit of each component in the reward function are different. For example, the number of loss packets is the positive integer number, e.g., 5 packets, while the sensing accuracy can be a fraction, e.g, 0.001 m/s. Therefore, a higher value of its weight does not guarantee a higher contribution to the immediate reward. In Section V.B.2, experiments are conducted to gain insights

into the impact of weights under various environmental conditions. The negative function in (4.13) implies that the ICAS-AV should take an action that can quickly free the data queue, lower the number of dropped packets, and achieve a high sensing accuracy. Note that the lower the value of δ_t is, the higher the velocity estimation accuracy of the system is. Given the above, the immediate reward function (4.13) effectively captures joint performance of communication and sensing function of the ICAS.

4.2.4 Optimization Formulation

The objective of this study is to find an optimal policy for the ICAS-AV that maximizes the long-term reward function. Let $R(\pi)$ denote the long-term average reward function under policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$, then the problem can be formulated as:

$$\max_{\pi} \; R(\pi) = \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\big(r_t(s_t, \pi(s_t))\big), \tag{4.14}$$

where $\pi(s_t)$ is the action at time t according to policy π. Thus, given the ICAS-AV's current data queue length and wireless channel quality, the optimal policy π^* gives an optimal action that maximizes $R(\pi)$. In addition, Theorem 4.1 shows that the average reward function is well defined regardless of the initial state.

Theorem 4.1. *With the proposed MDP framework, the average reward function $R(\pi)$ is well defined under any policy π and regardless of a starting state.*

Proof. We first prove that the Markov chain of the considered problem is irreducible as follows. Recall that the state of the ICAS consists of two factors, i.e., the current queue length q and the wireless channel quality c. For each time slot, the data arrival rate is assumed to follow the Poison distribution and the channel quality is derived from C class accordingly a probability vector $\mathbf{p_e} = [p_1, p_2, \ldots, p_C]$. Therefore, given the ICAS is at state s at time t, it can move to any other states $s' \in \mathcal{S}\{s\}$ after

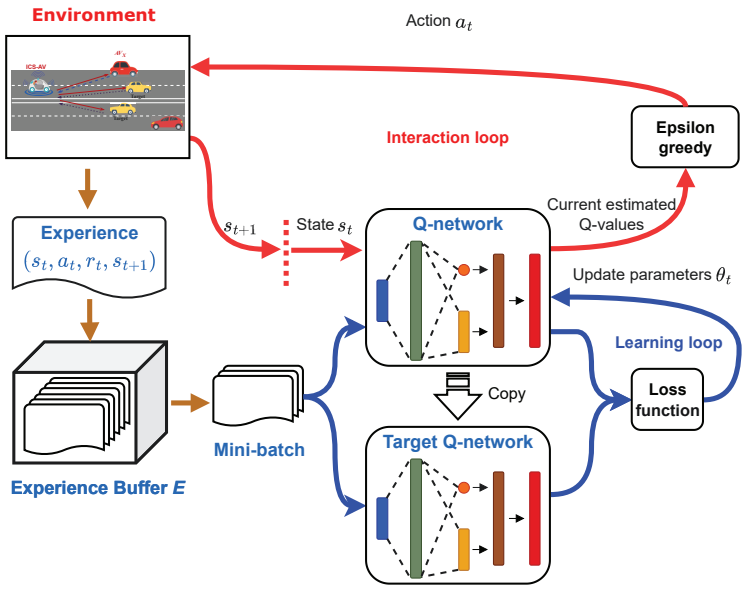


Figure 4.2 : The proposed i-ICS model, in which the ICAS-AV obtains an optimal policy by gradually updating its policy based on its observations of the surrounding environment.

finite time steps. As such, the proposed MDP is irreducible with the state space $\mathcal{S}$, thereby making the average reward function $R(\pi)$ is well defined under any policy π and regardless of a starting state. □

4.3 Reinforcement Learning-based Solutions for ICAS-AV Operation Policy

Due to the highly dynamic and uncertainty of the environment (e.g., packet drop probability due to the channel quality and the data arrival rate), the ICAS-AV is unable to obtain this information in advance. In this context, RL can help the ICAS-AV obtain the optimal policy without requiring completed knowledge about surrounding environment in advance. As such, this thesis proposes an approach i-ICS, as illustrated in Figure 4.2. Specifically, i-ICS is based on D3QL, discussed in Section 2.2.4, that can help the ICAS-AV gradually learn an optimal policy through interacting with its surrounding environment.

Generally, training a DNN requires a high computing resource, especially for complex neural architectures. Nevertheless, the Q-network in i-ICS only contains three layers in which only the hidden layer is fully connected. In addition, the Q-network is built only from traditional simple component, e.g., neurons with Tanh activation function. As such, our proposed approach can be effectively implemented on AVs that are equipped with sufficient computing resources. It is also important to mention that the decision of action for the ICAS-AV is an output of the Q-network after feeding the state to the Q-network. Therefore, the decision time (i.e., the inference time of the Q-network) is very marginal. It is worth noting that since i-ICS leverages DNNs, which are non-linear approximators, its optimality cannot be guaranteed theoretically. However, our simulation results in the next section demonstrate that i-ICS consistently outperforms baseline approaches. As i-ICS is based on D3QL, its discussion of the optimality and computational complexity was discussed in Section 2.2.5. In practice, a few deep learning applications have been deployed in AVs such as Tesla Autopilot [106] and ALVINN [107]. Thus, they are clear evidences of the effectiveness and efficiency when deploying machine learning algorithms on AVs in practice.

4.4 Performance Evaluation

4.4.1 Simulation Parameters

In this work, we investigate our proposed solutions in a scenario that includes three types of objects: (i) ICAS-AV, which is an autonomous vehicle equipped with 802.11ad based ICAS, (ii) a receiving vehicle AV_X that maintains communication with ICAS-AV, and (iii) a target AV_1 moving at a distance of around the ICAS-AV, as illustrated in Figure 4.1. Specifically, in the simulation, the related velocity of the target is selected randomly between -45 m/s to 50 m/s, similar to those in [38]. As such, the maximum related velocity between ICAS-AV and a target is

50 m/s (i.e., 111.847 miles/h). Therefore, this setting is appropriate for practical scenarios. Note that the proposed ICAS system uses the mmWave transmission, a highly directional communication. Therefore, we only consider the line-of-sight (LOS) scenario, similar to what considered in related works [38, 42, 119]. The CPI time T_{CPI} is set to $10T_d$, meaning that the maximum number of frames in the CPI is 10, similar to that in [38]. Note that one frame in the CPI can contain multiple packets. The ICAS-AV has a data queue containing a maximum of 50 packets. Recall that the data queue is used to store data packets when the amount of arrival data at the ICAS transmitter excesses the ICAS maximum transmission rate. All packets' sizes are assumed to be equal to 1500 Bytes, which is the typical value of the maximum transfer unit (MTU) in WiFi networks and the Internet [128]. Other parameters of the ICAS system are set based on IEEE 802.11ad standard, e.g., a carrier frequency of 60 GHz and a sampling rate of 1.76 GHz [121].

By considering the above settings, the environment is highly uncertain, dynamic, and unknown to the ICAS-AV in advance. The detailed settings of environment are as follows. First, as the wireless channel is dynamic and uncertain, the successful data transmission probability varies over time according to the channel condition. Thus, the packet error ratio PER can represent the Channel Quality Indicator. We consider that the ICAS system operates at Modulation and Coding Scheme 3 (MCS-3) of IEEE 802.11ad SC-PHY mode [121]. Note that the IEEE 802.11ad standard requires the PER of MCS-3 to be less than or equal to 1% and the $SNR \approx 1.5$ dB so that the system can normally perform [121]. In practice, the PER depends on many factors, such as wireless channel quality, modulation technique, and transmit power [127]. Therefore, for demonstration purposes, we consider three channel levels with different values of PER and SNR: (i) level-1 with $PER = 10\%$ and $SNR = -1.5$ dB, (ii) level-2 with $PER = 1\%$ and $SNR = 1.5$ dB, and (iii) level-3 with $PER = 0.3\%$ and $SNR = 4.5$ dB. Based on these quality levels, we assume that

the wireless channel can fall into one of the three types. The first one is the poor channel condition, in which the probability of channel quality at level-1, level-2, or level-3 at a time slot corresponds to a probability vector $\mathbf{p}_c^p = [0.6, 0.2, 0.2]$. Specifically, for the poor channel condition, the probability of the channel quality at level 1 at a time slot is 60% and so on. Besides poor channel, other types correspond to a normal and strong channel whose probability vectors are $\mathbf{p}_c^n = [0.2, 0.6, 0.2]$, and $\mathbf{p}_c^g = [0.2, 0.2, 0.6]$, respectively. Second, the packet arrival is also dynamic so that it is assumed to follow the Poisson distribution with the mean λ packets per time slot. Our proposed framework will be evaluated under these channel conditions (i.e., poor, normal, and strong channel) with different values of the mean data arrival rate λ. Moreover, we further evaluate our approach with two sets of weights for the immediate reward function, i.e., $\{w_1 = 0.05, w_2 = 0.4, w_3 = 0.5\}$ and $\{w_1 = 0.025, w_2 = 0.8, w_3 = 0.5\}$. Note that these above settings are just for simulation purposes. Our proposed learning algorithm (i.e., i-ICS) does not require these parameters in advance and can adapt to them through real-time interactions with the surrounding environment.

The parameters for the proposed learning algorithms are set as follows. Specifically, ϵ-greedy is leveraged to control the environment exploration in our proposed algorithms. Specifically, the agent takes a random action with probability ϵ and an action based on its current policy with probability $1 - \epsilon$. Unlike rule-based scenarios where agents have predefined policies, the RL agent does not have prior knowledge about its surrounding environment. As such, it needs to explore the environment by taking action randomly at the beginning. Then it adjusts its policy based on the selected action's feedback, i.e., the reward for performing this action. Therefore, the agent can gradually adjust the probability of selecting random action to obtain an optimal policy that maximizes the system performance [129]. However, due to the dynamic and uncertainty of the surrounding environment, the probability of taking

random action should not be zero since the RL needs to update its policy according to changes in the environment. Given the above, in this work, ϵ is set to 1 at the beginning and then linearly decreases to 0.1 after 8×10^5 iterations, which is similar to those in [33, 34].

Other parameters for the Q-learning-based and the i-ICS algorithms are set as typical values, as in [31, 33, 34]. Specifically, the discount factor η is 0.9 for both the Q-learning-based algorithm and the i-ICS. For the Q-learning based approach, the learning rate is 0.1. For the proposed i-ICS, the Adam optimizer is used to train the Q-network with a learning rate of 10^{-4} and the target Q-network's parameters are updated at every 10^4 time steps.

In this work, the simulation framework is programmed in Python and consists of two main entities, i.e., the ICAS-AV agent and the environment. At the beginning of time slot t, the agent observes the current state, i.e., $s_t = (q_t, c_t)$, where q_t is the current number of packets in the data queue and c_t is the channel quality. As described in Algorithm 2, the ICAS-AV agent maintains two DNNS (i.e., the Q-network and the target Q-network) built based on the Pytorch framework. Upon receiving the state, the agent feeds the current state to the Q-network to obtain an action a_t (i.e., the number of frames in the CPI) at this time slot. After the agent performs the selected action, the environment calculates an immediate reward r_t given by Eq. (4.13). Based on these information, the environment obtains (i) the number of arrival packets by drawing from the Poison distribution with the mean λ and (ii) the channel quality that is randomly generated based on the probability vector, e.g., $\mathbf{p}_c^p = [0.6, 0.2, 0.2]$ for the poor channel condition. Then, the environment updates the number of packets in the data queue and constructs a next state $s_{t+1} = (q_{t+1}, c_{t+1})$. After that, the environment sends the next state and the reward to the agent. Once receiving them, the agent stores the experience, i.e.,(s_t, a_t, r_t, s_{t+1}) to buffer $\mathbf{E}$. Then, it trains the Q-network as presented in Algo-

rithm 2. The target Q-network is updated at every 10^4 time steps by cloning from the Q-network's parameters. The process is repeated until the agent converges to an optimal policy. Recall that the ICAS-AV does not have any prior information about its surrounding environment's uncertainties and dynamics, e.g., the packet drop probabilities and packet arrival rate. Therefore, the proposed solutions are compared with two baseline policies: 1) a greedy policy where the ICAS-AV selects an action to maximize the reward function without caring about the uncertainties and dynamics of the environment and 2) a deterministic policy in which the ICAS-AV always sends n_{dp} frames in the CPI time. Here, we set n_{dp} to be a half of N (i.e., 5) to demonstrate the ICAS's average performance when the number of frames in the CPI is fixed. Note that we do not consider conventional optimization-based methods (e.g., [38, 119]) as baselines since they require complete information about the surrounding environment in advance to optimize the system parameters.

4.4.2 Simulation Results

To evaluate the ICAS system performance, we first examine the convergence rates of our proposed approaches, i.e., Q-learning based algorithm and i-ICS. We then study the influences of several key factors (e.g., the packet arrival rate, wireless channel quality, and weights in the immediate reward function) on the performance of the ICAS system.

4.4.2.1 Convergence Rate

Figure 4.3 illustrates the convergence rates of our proposed algorithms for the ICAS system, i.e., the Q-learning and the i-ICS. Here, we compare their performance in the normal channel, and the mean number of arrived packets λ is set to 14. It can be observed that the i-ICS achieves a superior result in terms of average reward compared with that of the Q-learning. Specifically, at the beginning of the learning process, the Q-learning and i-ICS obtain similar results. However, after 2×10^4

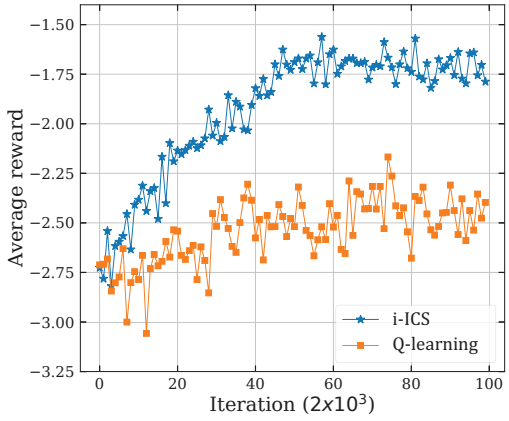


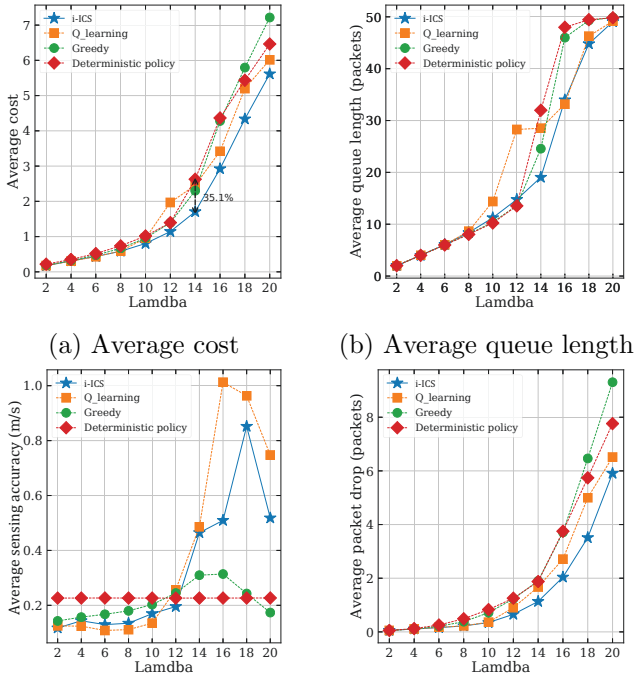
Figure 4.3 : Convergence rate of proposed algorithms.

iterations, the i-ICS's average reward is 20% greater than that of the Q-learning. Then, the i-ICS eventually converges to the optimal policy after 4.5×10^4 while Q-learning still struggles with a mediocre policy. Under the optimal policy obtained by i-ICS, the ICAS-AV's average reward is stable at around -1.75, approximately 40% higher than that of the policy learned by the Q-learning.

4.4.2.2 Performance Analysis

We then evaluate the robustness of our proposed approach, i.e., i-ICS, by varying the mean number of packets arrived at a time slot λ from 2 to 20. The learned policies of Q-learning and i-ICS are obtained after 2×10^5 training iterations. To evaluate the performance of the considered ICAS system, we consider four metrics, including (i) the average cost, which is the negation of the average reward, (ii) the average queue length, (iii) the average velocity estimation accuracy, i.e., sensing accuracy, and (iv) the average packet drop. The reason for introducing the average cost is to make the demonstration consistent across system performance metrics (i.e., the smaller the cost is, the better the system performance is). Note that the average reward indicates the joint performance of communication and sensing functions.

We first set the wireless channel to normal quality. The weights of the immediate reward function are presented by a weights vectors $\mathbf{W_1} = [0.05, 0.4, 0.5]$, i.e., $w_1 =$



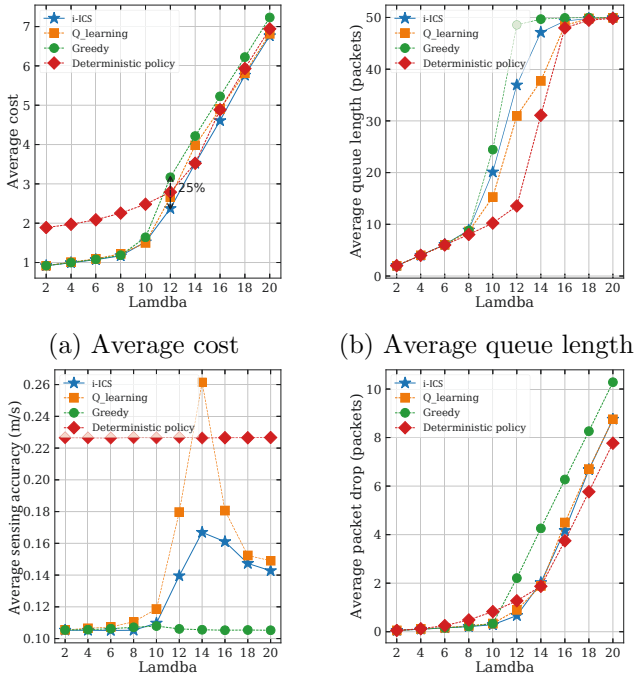
(a) Average cost (b) Average queue length

(c) Average sensing accuracy (d) Average packet drop

Figure 4.4 : Varying the data arrival rate λ under normal channel condition, i.e., $\mathbf{p}_c^n = [0.2, 0.6, 0.2]$, with the weight vector $\mathbf{W_1} = [0.05, 0.4, 0.5]$.

0.05, $w_2 = 0.4$, $w_3 = 0.5$, as shown in Figure 4.4. Clearly, the average costs of all policies increase as the packet arrival rate increases, i.e., λ increases from 2 to 20, as shown in Figure 4.4 (a). It stems from the fact that given a data queue with fixed capacity and the ICAS system operates under the same environment's characteristics, the higher the value of λ, the higher the number of dropped packets due to the full packet queue. Indeed, Figures 4.4 (b) and (d) clearly show that when the packet arrival rate increases, the average queue length and average packet drop increase for all policies. It can be observed that our proposed algorithm (i.e., i-ICS) achieves the lowest average cost, up to 64.9% (equivalent to an decrease of 35.1%) compared with those of other policies. Similarly, i-ICS has the lowest average packet drop regardless of the packet arrival rate, and it consistently maintains the average number of packets in the queue as one of the lowest values.

Regarding the sensing metric, i-ICS and Q-learning achieve the highest sensing



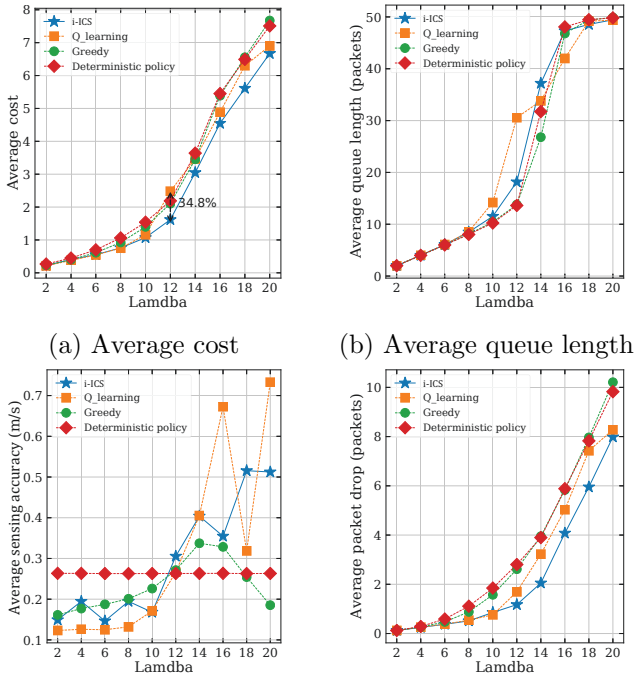
(a) Average cost (b) Average queue length

(c) Average sensing accuracy (d) Average packet drop

Figure 4.5 : Varying the data arrival rate λ under normal channel condition, i.e., $\mathbf{p}_c^n = [0.2, 0.6, 0.2]$, with the weight vector $\mathbf{W_2} = [0.025, 0.8, 0.5]$.

accuracy (i.e., the lowest value of average velocity estimation accuracy) when λ is less than 12. Whereas their average sensing accuracy results are not good compared to other policies if λ is larger than 12. The reasons are as follows. When the packet arrival rate is low (i.e., $\lambda < 12$), the average numbers of packets in the queue of all policies never pass 30% of the data queue capacity, as shown in Figure 4.4 (b). Thus, the ICAS-AV can increase the number of frames in the CPI, meaning a decrease in the packet sent in the CPI, to achieve a higher sensing performance without worrying about packet loss due to a full queue. Figure 4.4 (c) clearly shows that the Q-learning and i-ICS can learn this strategy to obtain the best performance in terms of average sensing accuracy when $\lambda < 12$.

As λ increases from 12 to 20, the average queue lengths of the greedy and deterministic policies quickly reach the maximum number of packets that can be stored in the data queue, i.e., 50, as shown in Figure 4.4 (b). This can lead to a high

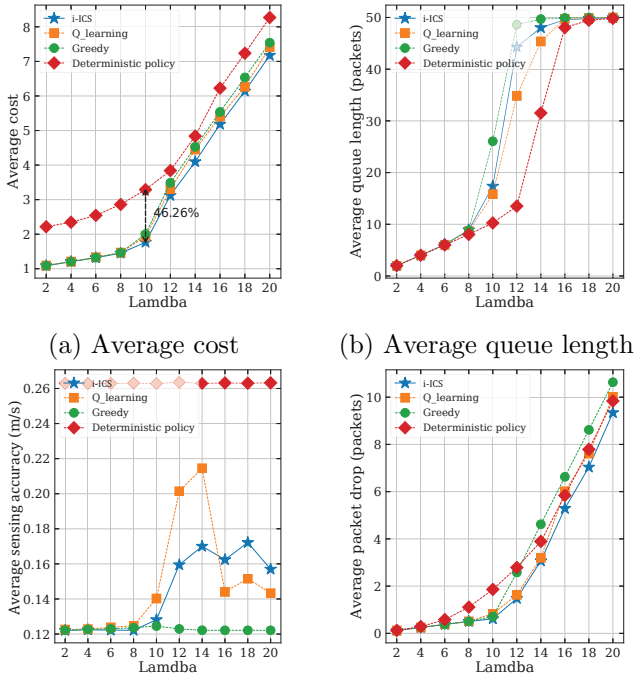


(a) Average cost (b) Average queue length

(c) Average sensing accuracy (d) Average packet drop

Figure 4.6 : Varying the data arrival rate λ under poor channel condition, i.e., $\mathbf{p}_c^p = [0.6, 0.2, 0.2]$, with the weight vector $\mathbf{W_1} = [0.05, 0.4, 0.5]$.

possibility of packet drop due to the full data queue. Interestingly, sensing performance of Q-learning and i-ICS policies decreases to the worst at $\lambda = 16$ and $\lambda = 18$, respectively. Then, they manage to increase the sensing accuracy when the packet queue is mostly always full at $\lambda = 20$. The reason is that the data transmission efficiency is unable to be improved because of a very high packet arrival rate that the system cannot handle. Thus, it might be better to improve the sensing accuracy instead of communication efficiency. On the other hand, since the greedy and deterministic policies do not consider the uncertainty of the environment (e.g., the packet drop possibility), they can maintain better sensing accuracy when the packet arrival rate is high. However, their transmission efficiency is very low. As can be seen in Figure 4.4 (d), the average numbers of packet drops of the greedy and deterministic policies are up to 50% higher than that of our proposed learning algorithm, i.e., i-ICS. Thus, the proposed algorithm can help the ICAS-AV to ob-



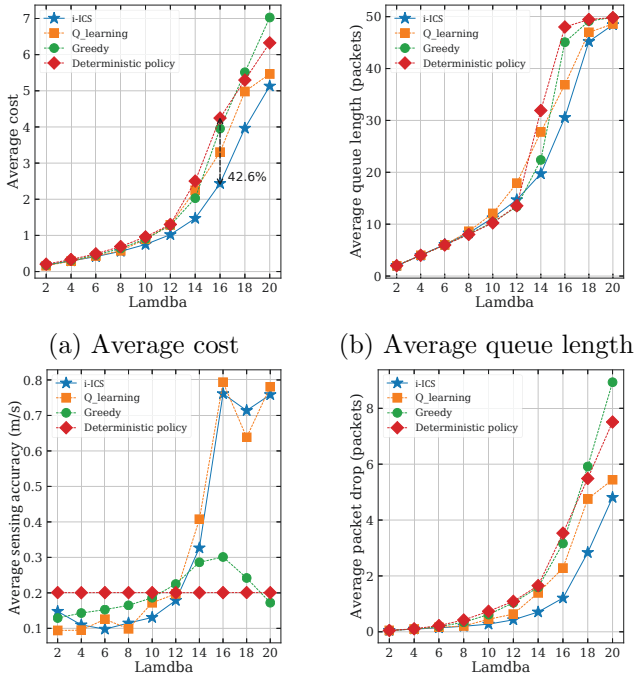
(a) Average cost (b) Average queue length

(c) Average sensing accuracy (d) Average packet drop

Figure 4.7 : Varying the data arrival rate λ under poor channel condition, i.e., $\mathbf{p}_c^p = [0.6, 0.2, 0.2]$, with the weight vector $\mathbf{W_2} = [0.025, 0.8, 0.5]$.

tain an optimal policy that strikes a balance between sensing and data transmission metrics, thereby achieving the best overall system's performance compared with those of other policies. Although i-ICS and Q-learning experience a similar trend when λ increases from 2 to 20, i-ICS consistently outperforms Q-learning. This stems from the fact that i-ICS can effectively address the high dimensional state in a complicated problem.

Interestingly, although i-ICS always achieves the lowest cost when λ increases from 2 to 20, as shown in Figure 4.4 (a), its sensing performance is not good as those of greedy and deterministic policies when $\lambda > 12$, as shown in Figure 4.4 (c). The reason is that the proposed reward function defined in Eq. (4.13) is a weighted sum of communication and sensing metrics. Specifically, the higher the reward obtained by the ICAS system is, the better performance the ICAS system achieves. Since the cost is the negation of the reward, the ICAS system performs better if its



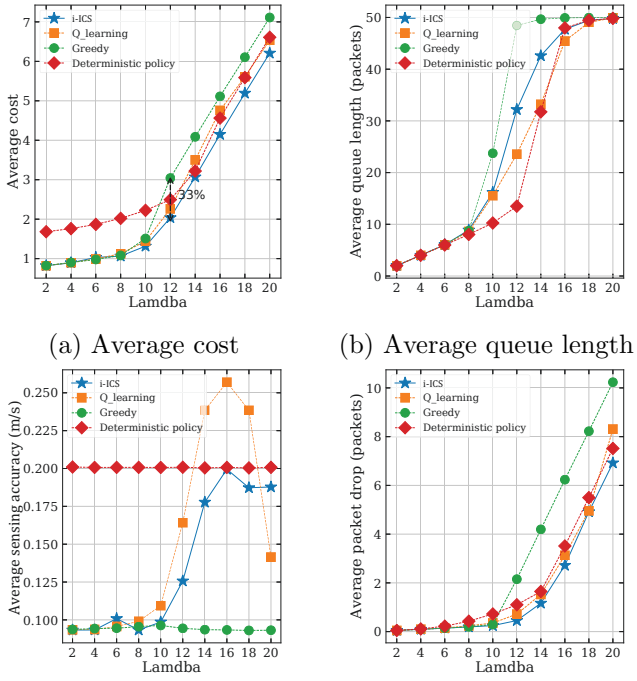
(a) Average cost (b) Average queue length

(c) Average sensing accuracy (d) Average packet drop

Figure 4.8 : Varying the data arrival rate λ under strong channel condition, i.e., $\mathbf{p}_c^g = [0.2, 0.2, 0.6]$, with the weight vector $\mathbf{W_1} = [0.05, 0.4, 0.5]$.

cost is lower. However, the cost only captures the joint/overall performance of the communication and sensing function. Therefore, one of these functions may perform worse than those of others whose costs are higher.

Next, we investigate how the immediate reward function's weights can influence the system performance by changing the weight vector to $\mathbf{W_2} = [0.025, 0.8, 0.5]$ and varying the packet arrival rate. In Figure 4.5, it can be observed that the results of deterministic policy are mostly unchanged, except the average cost result, when changing these weights because the ICAS-AV's environment is still the same as the previous experiment, and this policy does not rely on the immediate function. As the weight of sensing metric (i.e., w_2) is doubled, i-ICS, Q-learning, and greedy policies achieve sensing accuracy results that are much better than those in the previous experiment. In addition, except for the Q-learning, they also consistently outperform the deterministic policy in terms of sensing metric. In contrast, these



(a) Average cost (b) Average queue length

(c) Average sensing accuracy (d) Average packet drop

Figure 4.9 : Varying the data arrival rate λ under strong channel condition, i.e., $\mathbf{p}_c^g = [0.2, 0.2, 0.6]$, with the weight vector $\mathbf{W_2} = [0.025, 0.8, 0.5]$.

policies' data transmission metrics (i.e., the average packet drop and average queue length) become worse than those in the first experiment, as shown in Figures 4.5 (b) and (d). The reason is that when the ratios w_1/w_2 and w_3/w_2 become smaller, the ICAS system pays more attention to the sensing accuracy. Thus, Figure 4.5 clearly shows that in practice, these weights can be adjusted so that our proposed learning algorithm can obtain a policy that fulfils different requirements of a ICAS system at different times. Thanks to the ability to learn without requiring complete information of the surrounding environment, i-ICS still achieves the best overall performance when increasing the sensing metric's weight.

We now examine the robustness of our proposed solution i-ICS by considering different channel qualities, i.e., (i) poor quality with the PER probability vector $\mathbf{p}_c^p = [0.6, 0.2, 0.2]$ and (ii) good quality with the PER probability vector $\mathbf{p}_c^g = [0.2, 0.2, 0.6]$. To do so, we vary the packet arrival rate λ. For each of these channel qualities, two

sets of results are collected according to $\mathbf{W_1}$ and $\mathbf{W_2}$, as shown in Figures 4.6 to 4.9. Overall, all policies' results experience similar trends as those in normal channel quality. It can be observed that the channel quality significantly affects the joint communication and sensing performance, indicated by the average cost. Specifically, when the channel quality changes from poor to normal and then to good, the overall system performance increases regardless of the weight vector. The reason is that as the channel quality becomes better, the packet drop probability decreases, leading to a better communication performance of the ICAS system. Hence, the overall performance improves.

In terms of sensing performance, as observed in the sub-figures (c) of Figures 4.4 to 4.9, when the mean packet arrival rate is small, i.e., $\lambda \leq 12$, the performance of i-ICS increases as the channel changes from poor to good quality for both $\mathbf{W_1}$ and $\mathbf{W_2}$. Interestingly, when λ is larger than 12, the sensing performance of i-ICS experiences differently with the weight vectors $\mathbf{W_1}$ and $\mathbf{W_2}$. Specifically, with $\mathbf{W_1}$, the i-ICS's sensing performance generally becomes deteriorating as the channel changes from poor to good quality. Whereas, with $\mathbf{W_2}$, the i-ICS achieves the highest and lowest sensing performance under the normal channel and strong channel. Thus, sensing performance depends on not only the channel quality but also the immediate reward function's weights and the mean packet arrival rate λ. As such, better channel quality does not guarantee better sensing performance. Note that the proposed approach in this study aims to maximize the joint communication and sensing performance (i.e., overall performance). Therefore, as discussed above, even though the performance of one function (e.g., sensing) is a bit lower, our proposed i-ICS still achieves the best overall performance.

In summary, i-ICS achieves the highest overall performance boost among the considered policies when channel quality changes from poor to good. For instance, with $\mathbf{W_1}$, i-ICS's average cost decreases up to 51.7% while those of the greedy and

deterministic policies reduce up to 41.17%. This is because our proposed approach can effectively adapt its behaviour according to the changes in its surrounding environment to improve the system performance significantly.

4.5 Conclusion

In this work, we have developed a novel MDP-based framework that allows an ICAS-AV to automatically and adaptively decide its optimal waveform structure based on the observations to maximize the overall performance of the ICAS system. Then, we have proposed an advanced learning algorithm, i.e., i-ICS, that can help the ICAS-AV gradually learn an optimal policy through interactions with the surrounding environment without requiring complete knowledge about the environment in advance. As such, our proposed approach can effectively handle the environment's dynamic and uncertainty as well as the high dimensional state space problem of the underlying MDP framework. The extensive simulation results have clearly shown that the proposed solution can strike a balance between communication efficiency and sensing accuracy, thereby consistently outperforming the benchmark methods in different scenarios.

Chapter 5

MetaSlicing: A Novel Resource Allocation Framework for Metaverse

This chapter presents MetaSlicing, our proposed resource allocation framework for Metaverse, an emerging service supported by 6G. Creating and maintaining the Metaverse require enormous resources that have never been seen before, especially computing resources for intensive data processing to support the Extended Reality, enormous storage resources, and massive networking resources for maintaining ultra high-speed and low-latency connections. Therefore, this work aims to propose a novel framework, namely MetaSlicing, that can provide a highly effective and comprehensive solution for managing and allocating different types of resources for Metaverse applications. In particular, by observing that Metaverse applications may have common functions, we first propose grouping applications into clusters, called MetaInstances. In a MetaInstance, common functions can be shared among applications. As such, the same resources can be used by multiple applications simultaneously, thereby enhancing resource utilization dramatically. To address the real-time characteristic and resource demand's dynamic and uncertainty in the Metaverse, we develop an effective framework based on the semi-Markov decision process and propose an intelligent admission control algorithm that can maximize resource utilization and enhance the Quality-of-Service for end-users. Extensive simulation results show that our proposed solution outperforms the Greedy-based policies by up to 80% and 47% in terms of long-term revenue for Metaverse providers and request acceptance probability, respectively.

This chapter is structured as follows. Sections 5.1 introduces the system model

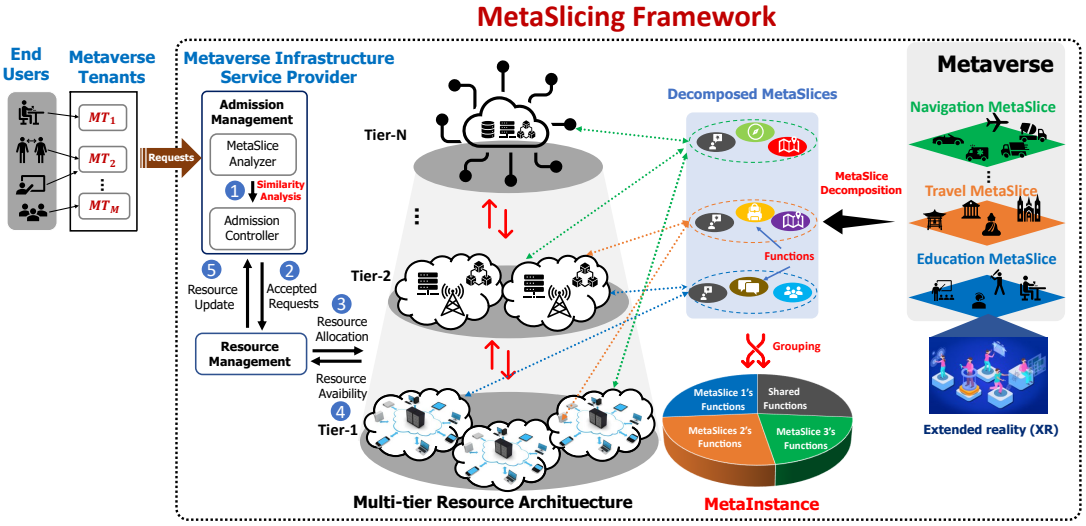


Figure 5.1 : The system model of the proposed MetaSlicing framework. In this framework, different resource types in different tiers can be used and shared to create Metaverse applications (i.e., MetaSlices).

based on multi-tier resource allocation architecture to facilitate the deployment and operation of Metaverse applications. Then, the Metaverse application admission control formulation is presented in Section 5.2 After that, the Metaverse application analysis and the proposed deep reinforcement learning-based algorithm are discussed in Section 5.3. In Section 5.4, simulation results are analyzed. Finally, Section 5.5 provides the conclusion of this chapter.

5.1 System Model

In this work, we consider a system model including three main parties, i.e., (i) End-users, (ii) Metaverse tenants, and (iii) the Metaverse Infrastructure Service Provider (MISP), as illustrated in Figure 5.1. First, an end-user subscribes to a Metaverse tenant to request a Metaverse application, namely MetaSlice. Then, the Metaverse tenant will request the MetaSlice from the MISP according to its subscribed users' demands. If a request is accepted, the MISP will allocate its resources to initiate this MetaSlice. In the following, we explain the main components together with their interactions in our proposed framework in more details.

5.1.1 MetaSlicing: Dynamic Resource Allocation Framework for Metaverse

5.1.1.1 Multi-tier Resource Allocation Architecture-based Metaverse

As discussed in the previous section, although the centralized cloud can be used for deploying Metaverse's applications, it poses critical challenges, e.g., a point-of-congestion in the network and a point of failure. In the literature, most existing works related to the Metaverse resource management consider single-tier edge computing architecture [54, 56, 57]. However, the single-tier edge architecture may not be appropriate and effective since the edge capacity is often limited while Metaverse applications often demand intensive resources and strict requirements. Moreover, in single-tier edge computing, Metaverse applications are likely created near the user's location so that the user's QoE requirements may not be satisfied if the user moves far away.

To address these challenges, a multi-tier resource allocation architecture can be leveraged to effectively and dynamically allocate resources for Metaverse applications. First, this architecture alleviates the extensive resource demands of Metaverse applications for both the Metaverse tenants and end-users. Second, this architecture enables the distribution of different types of resources, e.g., computing, storage, networking, and communication capabilities, along the path from end-users to the cloud. By doing so, Metaverse applications can leverage the resources placed near end-users, resulting in a low delay and high QoS for users. Third, when a user moves to a new location, these applications can be migrated to a site near the user's new location to maintain the QoE. Thus, distributing resources increases the resilience of the system compared to the traditional centralized cloud-based resource allocation architecture.

Although the multi-tier resource architecture offers a promising approach to de-

ploy Metaverse applications, it still faces several challenges. First, recall that Metaverse applications are heavily integrated with XR technologies requiring stringent latency. As a result, they are supposed to be created at the edge servers that are placed near users, i.e., tier-1, thereby possibly causing overload at low tiers. Note that the resource capacity at low tiers is often much lower than those at high tiers, e.g., cloud. As such, it may result in high latency in both computing and transmission or even interrupting services, thus reducing users' QoE since latency is one of the most important factors to guarantee QoE in Metaverse. Second, a user can physically move while using MetaSlices, and the latency requirement may not be guaranteed if they move too far from the place where ongoing applications are created. This issue can be mitigated by migrating these MetaSlices to a new place near the user's new location. However, the migration may introduce long delay due to the transmission of user's data and the application re-initialization. In the following subsection, we will discuss our proposed technique (i.e., MetaSlice decomposition) to alleviate these problems.

5.1.1.2 MetaSlice Decomposition

Recall that the Metaverse can be realized as a seamless integration of multiple virtual worlds, and each virtual world can be regarded as an application for a specific purpose, e.g., entertainment, healthcare, or education [13–15]. We observe that an application may have several functions that can operate independently. For example, a tourism application may have a recommendation, digital map, and driving assistant functions. Based on the user's location, the recommendation function can suggest nearby attractive places, and then the user can use the digital map function to get more information about these places. After that, the user can use the driving assistant function to get to the chosen place. Since these functions can be used or unused depending on different users, they can run separately without interfer-

ing with other functions. As such, Metaverse applications should be developed in a modular way by which their functions can be initialized and operated independently. By doing so, these independent functions can be connected to each other via application programming interfaces (API), similar to how existing applications connect to current online services, e.g., Google Maps API. Given the above, this work considers that a MetaSlice, i.e., application, can be decomposed into multiple dependent functions.

Note that this work does not focus on optimal application decomposition, and we assume that applications can be decomposed by using existing methods, e.g., [130, 131]. We consider that each function is allocated dedicated resources due to the strict QoS requirements of Metaverse applications. In particular, a function with the dedicated resource allocation scheme likely executes faster than this function under the dynamic scheme. This is because, under the dynamic resource allocation scheme, a resource allocation procedure is performed whenever a request for processing arrives at a function, leading to an unavoidable additional delay that may degrade the QoS of Metaverse users. In addition, because resources are not reserved for each function in a dynamic allocation scheme, a function may not have sufficient resources to support an application's request that arrives when a server is overloading with other functions, leading to a high delay or even service disruption. This problem can be alleviated by migrating the function to another server with adequate resources; however, it introduces another additional delay to the function execution. In this context, dedicated resource allocation can avoid this issue by reserving resources for each function. Moreover, the study in [132] points out that under a typical load of business applications, the dedicated resource allocation can achieve a better energy efficiency than that of dynamic resource allocation.

In practice, the MetaSlice decomposition can offer numerous benefits for deploying and managing MetaSlices in Metaverse. First, as functions are independent

entities and are connected via API, they can be developed and upgraded independently and simultaneously, thereby speeding up MetaSlice's deployment and evolution. Second, this technique also helps Metaverse developers to concentrate on building their distinctive functions (e.g., a recommendation function in tourism) rather than putting their resources on other functions (e.g., driving assistant or digital map) that can be effectively developed by specialized third parties. Third, the MetaSlice decomposition can help to manage MetaSlices more convenient. For instance, when a function in a MetaSlice fails, users can be quickly redirected to a compatible function to alleviate service disruption. Fourth, the MetaSlice decomposition with a multi-tier resource architecture provides flexibility for Metaverse implementation. In particular, functions can be initiated at different tiers since they are independent entities. For example, a MetaSlice for travel may consist of several major functions, such as a digital map, real-time traffic, real-time weather, and a driving assistant. In this case, these functions can be placed dynamically in the multi-tier resource allocation architecture depending on the functions' requirements. Specifically, real-time traffic and driving assistant functions can be placed at tier-1 near end-users since they require low delay, while a digital map (that does not need frequent updates) can be located at a high tier, e.g., at the cloud. Finally, this technique can also alleviate the application migration problem in a multi-tier resource architecture. Particularly, by leveraging application decomposition, instead of migrating a MetaSlice, only functions with stringent delay requirements (e.g., less than 20 ms [133]) will be migrated, thereby significantly decreasing migration delay. Thus, the application decomposition technique offers a flexible and effective implementation of multi-tier-based MetaSlices.

To support the decomposition technique, we consider that a MetaSlice is created based on the MetaBlueprint, i.e., a template describing the workflow, configuration, and structure for initializing and managing this MetaSlice during its life cycle. From

Table 5.1 : Comparison between MetaSlicing, network slicing, and virtualized network function allocation.

Approach	**Aim**	**Novel Features**
Virtualized network function (VNF) allocation	Distributing VNFs under limited resources.	Utilizing the virtualization technologies to virtualize network devices' functions (such as routing, switching, and load balancing) that are used to create and deliver communication services [134].
Network slicing	Creating of multiple virtual networks (i.e., network slices) that can coexist on a single physical network infrastructure [135].	Leveraging VNFs to create various specialized and dedicated network services for different applications, such as autonomous vehicles, manufacturing, or virtual reality applications [135, 136].
Our proposed MetaSlicing	Creating of multiple virtual worlds or "Metaverses" that can be accessed and interacted with by users through a variety of devices and platforms.	- The application decomposition (with the underlying multi-tier resource allocation architecture) help enhance manageability and accelerate the deployment and evolution of Metaverse applications (i.e., MetaSlice). - The MetaInstance can exploit the functional similarities between MetaSlices to maximize resource utilization.

a technical standpoint, a MetaSlice (i.e., an application in the Metaverse) can be analogous to the network slice paradigm in the fifth generation of the cellular network (5G) that consists of multiple network functions [135]. It is worth mentioning that slicing is the general term in describing virtualization methods that can partition and organize resources (e.g., computing and networking) of the physical infrastructure to flexibly support diverse requirements [137]. As such, the proposed MetaSlicing mechanism seems similar to the network slicing mechanism in 5G networks [135] due to their names, but they are essentially different, as shown in Table 5.1.

Specifically, MetaSlicing refers to the creation of multiple virtual worlds or "Metaverses" that can be accessed and interacted with by users through a variety

of devices and platforms. Each Metaverse application (i.e., MetaSlice) is a self-contained virtual environment that can be customized to suit the needs of its users. As such, MetaSlicing is intended to enable the creation of personalized and immersive virtual experiences for users. On the other hand, network slicing refers to the creation of multiple virtual networks (i.e., network slices) that can coexist on a single physical network infrastructure. Each network slice is a self-contained network that can be customized to meet the needs of its users, with its own set of network functions and performance characteristics. Thus, network slicing is intended to enable the creation of specialized and dedicated network services for different applications, such as autonomous vehicles, manufacturing, or virtual reality applications.

It is worth noting that MetaSlicing also is a framework to manage resources for Metaverse applications effectively. To do so, it first decomposes an application into independent functions and then optimally distributes them at different tiers. Then, MetaSlicing arranges MetaSlice into multiple groups, i.e., MetaInstances, where a number of functions can be shared among MetaSlices, thus improving resource utilization.

5.1.1.3 MetaInstance

To further take advantage of MetaSlice decomposition, we introduce the MetaInstance that can improve system resource utilization. Suppose that Metaverse may consist of different MetaSlice types, e.g., tourism, education, industry, and navigation. In addition, MetaSlices can also be grouped into I classes based on their characteristics, such as occupied resources, technical configurations, and QoS. For example, navigation MetaSlice may have driving assistant functions requiring ultra-low latency and highly-reliable connections, while security and resilience are among the top concerns of e-commerce and industry MetaSlices. Moreover, different types of MetaSlice may share the same functions. For instance, tourism and naviga-

tion MetaSlices can use the same underlying digital map and the real-time traffic and weather functions whose data are collected by the same perception network, e.g., IoT. Furthermore, we can observe that different Metaverse tenants can create/manage multiple variants from the same type of MetaSlice (e.g., education, industry, or navigation). Therefore, ongoing MetaSlices may share the same functions. In this case, a lot of resources can be shared, leading to greater resource utilization and higher revenue for the Metaverse Infrastructure Service Provider (MISP).

Based on this fact, Metaverse applications can be classified into groups, namely MetaInstances. A MetaInstance can be defined by two function types, i.e., (i) shared function and (ii) dedicated function belonging to specific MetaSlices, as illustrated in Figure 5.1. In this case, a MetaInstance can maintain a function configuration consisting of a list of functions and a description of interactions among them. Because the capability of a function is limited, sharing a function for too many MetaSlices may lead to a decrease in user experience (e.g., processing delay) or even service disruption. Therefore, in practice, a function can be only shared by a maximum number of N_L MetaSlices. From the technical perspective, the implementation of a MetaInstance can be similar to that of the Network Slice Instance (NSI), where network slices can share some Network Functions (NFs) [135].

It is worth mentioning that even though the MetaSlicing may be similar to the network slicing [135], they are actually not the same. In particular, network slicing aims to address the diversity (or even conflict) in communication requirements among various businesses by running multiple logical networks (i.e., Network Slices) over a physical network. For example, one (e.g., automotive customers) may require ultra-low latency connections while others (e.g., manufacturing customers) require ultra-reliable connections. Thus, the network slicing focuses on providing diverse types of communications. In contrast, MetaSlicing is a framework to effectively manage resources in a multi-tier computing architecture by decomposing an

application into functions and then optimally distributing them at different tiers. In addition, as explained in the previous paragraphs, due to different features of Metaverse applications, resource allocation scheme for MetaSlicing is also designed different from that of the network slicing, where dynamic resource allocation approaches are preferred [138].

Based on the aforementioned analysis, we can observe that, on the one hand, our proposed MetaSlicing framework can offer a great solution to the MISP by maximizing the resource utilization and at the same time minimizing the deployment cost and initialization time for Metaverse's applications. On the other hand, this framework can also benefit end-users by achieving greater user experience, e.g., lower delay and more reliable services. To achieve these results, the Admission Controller in MetaSlicing plays a critical role. For example, accepting requests of MetaSlices that share some functions with the ongoing MetaSlices may help the system to save more resources than accepting those with less or without sharing functions with the MetaSlices running in the system. In addition, Resource Management is another important factor determining the Metaverse system performance. In the following subsection, we explain these components in our proposed MetaSlicing framework.

5.1.2 Admission Control and Resource Management

Recall that according to the demands of subscribed end-users, a Metaverse tenant sends a MetaSlice request associated with MetaBlueprint to the MISP. Then, the Admission Management and Resource Management are executed as follows. As shown in Figure 5.1, the Admission Management block of the MISP consists of a MetaSlice Analyzer and an Admission Controller. Once receiving a MetaSlice request, the MetaSlice Analyzer will analyze the request's MetaBlueprint to determine similarities between the functions and configuration of the requested MetaSlice and those of the ongoing MetaInstances. In this study, we consider that a MetaBlueprint

consists of at least (i) the function configuration record, including the list of required functions and the description of interactions among them, and (ii) the MetaSlice configuration (e.g., class ID and required resources). Based on the similarity analysis (to be presented in Section IV) obtained from the MetaSlice Analyzer and the currently available resources of the system, the Admission Controller decides whether to accept the request or not according to its admission policy.

Suppose that a MetaSlice request is accepted, the Resource Management allocates the accepted MetaSlice to a MetaInstance with the highest similarity index and updates this MetaInstance accordingly. Specifically, if the accepted MetaSlice has dedicated functions, system resources are allocated to initiate these new functions. If the current MetaInstances do not share any function with the new MetaSlice, a new MetaInstance is created for the accepted MetaSlice. When a MetaSlice departs/completes, its resources will be released, and the MetaInstance will be updated accordingly.

In this work, we consider D resource types owned by the MISP, e.g., computing, networking, and storage. Then, the required resources for a MetaSlice m can be represented by a resource vector, i.e., $\mathbf{n}_m = [n_m^1, \ldots, n_m^d, \ldots, n_m^D]$, where n_m^d is the amount of type d resources. In this case, the total occupied resources by all MetaSlices cannot exceed the maximum resources of MISP, i.e.,

$$\sum_{m \in \mathcal{M}} n_m^d \leq N^d, \quad \forall d \in \{1, \ldots, D\}, \tag{5.1}$$

where N^d is the total amount of type d resources of the MISP, and $\mathcal{M}$ is the set of all running MetaSlices in the system. In our proposed solution, the system's available resources and the required resources for the request are two crucial factors for the admission control in MetaSlicing. However, in practice, the future requests' arrival process and its required resources are likely unknown in advance. In addition, the

departure process of MetaSlices (i.e., how long a MetaSlice remains in the system) is also highly dynamic and uncertain. Therefore, in the next section, we will introduce a framework based on the semi-Markov decision process to address these challenges.

5.2 MetaSlicing Admission Control Formulation

In this work, we propose a highly-effective semi-Markov Decision Process (sMDP) framework to address the MetaSlice admission control problem due to the following reasons. First, the sMDP can enable the MetaSlicing's Admission Controller to adaptively make the best decisions (i.e., whether to accept or reject a MetaSlice request) based on the currently available system resources (i.e., computing, networking, and storage) and the MetaSlice request's blueprint (i.e., resource, class and similarity) without requiring complete information about the surrounding environment (e.g., arrival and departure processes of MetaSlices) to maximize the MISP's long-term revenue. Second, in practice, MetaSlice requests can arrive at any time, so the admission decision needs to be made as soon as possible. As such, the decision epoch in the considered problem changes rapidly. However, the conventional Markov Decision Process (MDP) only takes an action at each time slot with an equal time period, making it unable to capture real-time events, e.g., request arrival [98]. In contrast, the sMDP makes a decision whenever an event occurs so that it can perfectly capture the real time of MetaSlicing. Finally, the MetaSlice's lifetime is highly uncertain. Upon a MetaSlice departs from the system, its occupied resources are released, and the system state transits to a new state immediately. Again, the conventional MDP is unable to capture this transition as it works in a discrete-time fashion.

To that end, the sMDP will be used in our framework to enable the MISP to make real-time decisions and maximize its long-term revenues. Technically, an sMDP can be defined by a set of five components, including (i) the decision epoch t_i, (ii) the

Table 5.2 : Table of Notations

Notation	Description
Q	Maximum number of packets in the data queue (Packets)
$\mathbf{n}_m$	Required Resource Vector of MetaSlice m
n_m^d	The amount of type d resources required by MetaSlice m
$\mathbf{n}_u$	the available resources
n_u^d	The number of available resources type d
N^d	The total amount of type d resources of the MISP
$\mathcal{A}$	The action space
$\mathcal{S}$	The state space
$\mathcal{T}$	The transition probability
r	The immediate reward function
i	The class ID
j	The similarity index
$\mathbf{e}$	The event vector
a_s	The action at state s
$\mathbf{x}$	The number of ongoing MetaSlices
x_i	The number of MetaSlices in class i that are running
λ_i	The mean of Poisson distribution
$1/\mu_i$	The mean of the exponential distribution
w_d	The weight corresponding to resource type d
n_o^d	Number of type d resources occupied by this requested MetaSlice
$\pi(s_g)$	The action derived by policy π at decision epoch g
τ_g	The interval time between two consecutive decision epochs
$\overline{\mathcal{T}}_\pi$	The limiting matrix corresponding to policy π
$\mathcal{F}_m$	The function configuration of a MetaSlice m

state space $\mathcal{S}$, (iii) the action space $\mathcal{A}$, (iv) the transition probability $\mathcal{T}$, and (v) the immediate reward function r. In the following sections, we will explain how this framework can capture all events in the MetaSlice system and make optimal decisions for the MISP.

5.2.1 Decision Epoch

The decision epochs are defined as points of time at which decisions are made [98]. In our real-time MetaSlicing system, the Admission Controller must make a decision once a MetaSlice request arrives. Therefore, we can define the decision epoch as an interval between the occurrence of two consecutive requests.

5.2.2 State Space

Aiming to maximize revenue for the MISP with limited resources, several important factors need to be considered in the system state space. First, the current system's available resources and the resources required by the current MetaSlice request are the two most important factors for the Admission Controller to decide whether it accepts this current request or not. Second, since the income for leasing each MetaSlice class is different, the class ID i of a requested MetaSlice is another crucial information. Third, the similarity index j of a requested MetaSlice reflects the similarity between a requested MetaSlice and the ongoing MetaInstances. Recall that the higher the value of j is, the higher similarity between the requested MetaSlice and the running MetaInstances is, leading to lower occupied resources when deploying this new request. Hence, the similarity index is also an important factor for the MetaSlice admission decision.

We denote the available resources of the system by a vector $\mathbf{n}_u = [n_u^1, \ldots, n_u^d, \ldots, n_u^D]$ where D is the total number of resource types, and n_u^d denotes the number of available resources of type d. Similarly, the request's required resources are denoted by $\mathbf{n}_m = [n_m^1, \ldots, n_m^d, \ldots, n_m^D])$, where n_m^d is the number of requested resources of type d. Given the above, the system state space can be defined as follows:

$$\begin{aligned} \mathcal{S} \triangleq \Big\{ \big(n_u^1, \ldots, n_u^d, \ldots, n_u^D, n_m^1, \ldots, n_m^d, \ldots, n_m^D, i, j\big): \\ n_u^d \text{ and } n_m^d \in \{0, \ldots, N^d\} \forall d \in \{1, \ldots, D\}; \\ i \in \{1, \ldots, I\}; j \in [0, \ldots, J] \Big\}, \end{aligned} \tag{5.2}$$

where N^d is the maximum resources of type d, I is the total number of classes in the system, and J is the maximum similarity index of a MetaSlice derived by the MetaSlice Analyzer. By this design, the system state is presented by a tuple, i.e., $\mathbf{s} \triangleq (\mathbf{n}_u, \mathbf{n}_m, i, j)$, and the system can work continuously without ending at a

terminal state, at which the system stops working [99].

Recall that in the sMDP framework, the system only transits from state $\mathbf{s}$ to state $\mathbf{s}'$ if and only if an event occurs (e.g., a new MetaSlice request arrival). We define the event as a vector $\mathbf{e} \triangleq [e_1, \ldots, e_i, \ldots, e_I]$, where $e_i \in \{-1, 0, 1\}$. Specifically, $e_i = -1$ if a MetaSlice class i departs from the system, $e_i = 1$ if a new MetaSlice request class-i arises, and $e_i = 0$ otherwise (i.e., no MetaSlice request class-i arrival or departure). Thus, the set of all possible events is given as follows:

$$\mathcal{E} \triangleq \big\{\mathbf{e} : e_i \in \{-1, 0, 1\}; \sum_{i=1}^{I} |e_i| \leq 1\big\}. \tag{5.3}$$

Note that there is a trivial event $\mathbf{e}^* \triangleq (0, \ldots, 0)$ meaning that no MetaSlice request of any class arrives or departs.

5.2.3 Action Space

If a MetaSlice request arrives at state $\mathbf{s}$, the Admission Controller can decide whether to accept or reject this request to maximize the long-term revenue for the MISP. Thus, the action space at state $\mathbf{s}$ can be defined by:

$$\mathcal{A}_{\mathbf{s}} \triangleq \{0, 1\}. \tag{5.4}$$

In particular, if the requested MetaSlice is accepted, the action at state $\mathbf{s}$ is equal to one, i.e., $a_{\mathbf{s}} = 1$. Otherwise, $a_{\mathbf{s}} = 0$.

Recall that a user needs to subscribe to a Metaverse tenant to access a MetaSlice, i.e., a Metaverse application. As such, we assume that users' requests are queued at the corresponding Metaverse tenant before being forwarded to the MISP. If the MISP rejects a user's request, it will return to the queue at the Metaverse tenant subscribed by this user until it is served or the user decides to stop the request. Since

the lifetime of a MetaSlice is limited, and resources are released whenever a MetaSlice departs, a rejected request is likely served eventually. It is important to note that this work does not focus on managing the queue, and we assume that the queue management is independently controlled by each Metaverse tenant. In addition, a Metaverse tenant can rent resources from several MISPs to provide alternative access for their subscribed users if the current MISP runs out of resources. Again, this problem is also out of the scope of this work. Thus, this is a potential direction for future works.

5.2.4 State Transition Probabilities

This sub-section analyzes the sMDP's dynamic by characterizing the underlying Markov chain's state transition probabilities. Since the sMDP is based on the semi-Markov Process (sMP) that consists of a renewal process and a Continuous-time Markov Chain (CTMC) $\{X(t)\}$, the uniformization method can be used to derive the state transition probabilities $\mathcal{T}$ [98, 139, 140]. Specifically, the uniformization transforms the CTMC into a corresponding stochastic process $\{\bar{X}(t)\}$ whose transition epochs are derived from a Poisson process at an uniform rate, whereas state transitions follow a discrete-time Markov chain $\{X_n\}$. These two processes, i.e., $\{\bar{X}(t)\}$ and $\{X(t)\}$, are proven to be probabilistically equivalent [98].

In practice, similar to many communication systems, e.g., mobile phone systems, we do not know when a user request comes and leaves the system. Thus, we can consider that the arrival process of class-i requests follows the Poisson distribution with mean λ_i while the departure of class-i MetaSlice follows an exponential distribution with mean $1/\mu_i$, as those in [139]. In this way, the parameters of the uniformization

method are defined as:

$$z = \max_{\mathbf{x} \in \mathcal{X}} \sum_{i=1}^{I} (\lambda_i + x_i \mu_i), \tag{5.5}$$

$$z_{\mathbf{x}} = \sum_{i=1}^{I} (\lambda_i + x_i \mu_i), \tag{5.6}$$

where each element of vector $\mathbf{x} \triangleq [x_1, \ldots, x_i, \ldots, x_I]$ represents the number of on-going MetaSlices in the corresponding class (i.e., x_i is the number of MetaSlices in class i that are running simultaneously in the system), and $\mathcal{X}$ is the set containing all possible values of $\mathbf{x}$. Since the total resources of on-going MetaSlices cannot exceed the total number of system resources, we have

$$\sum_{i=1}^{I} \sum_{m=1}^{x_i} n_m^d \leq N^d \quad \forall d \in \{1, \ldots, D\}. \tag{5.7}$$

Now, the events' probabilities are determined based on z and $z_{\mathbf{x}}$ as follows:

- The probability of a class i request occurring in the next event $\mathbf{e}$ is λ_i / z.
- The probability of a class i MetaSlice departing in the next event $\mathbf{e}$ is $x_i \mu_i / z$.
- The probability of trivial event (i.e., no MetaSlice request of any class arrives or departs) arising in the next event $\mathbf{e}$ is $1 - z_{\mathbf{x}} / z$.

Then, we can obtain the state transition probabilities $\mathcal{T} = \{P_{\mathbf{ss}'}(a_{\mathbf{s}})\}$ with $\mathbf{s}, \mathbf{s}' \in \mathcal{S}$ and $a_{\mathbf{s}} \in \mathcal{A}_{\mathbf{s}}$, i.e., the probability that the system moves between states by taking actions.

5.2.5 Immediate Reward Function

To maximize the MISP's long-term revenue, the income from leasing resources to Metaverse tenants should be captured by the immediate reward function. Here, we

consider that the revenues for leasing resources for different classes are different since different classes may have different requirements such as reliability and delay. Recall that in our proposed Metaverse system, MetaSlices can share some functions with others, leading to differences in resource occupation even between MetaSlices from the same class. As such, even if two MetaSlices give the same revenue, accepting a MetaSlice that requires fewer resources will benefit the provider in the long term. Therefore, the number of resources required by a MetaSlice is another key factor.

To this end, the immediate reward function can be defined as follows:

$$r(\mathbf{s}, a) = \begin{cases} r_i - \sum_{d=1}^{D} w_d n_o^d, & \text{if } e_i = 1 \text{ and } a = 1, \\ 0, & \text{otherwise,} \end{cases} \tag{5.8}$$

where r_i is the revenue from leasing resources for a MetaSlice class i, and n_o^d is the number of type d resources occupied by this requested MetaSlice. The trade-offs between these factors are reflected by weights, i.e., $\{w_i\}_{d=1}^{D}$. Equation (5.8) implies that if slices have the same income (i.e., they are in the same class), accepting requests with fewer resource demands can help the provider to maximize the long-term revenue.

Note that a weight w_d reflects the price for renting one unit of resources type d (e.g., 1 TB storage) from the Metaverse Infrastructure Service Provider (MISP). In the literature, resource pricing is a well-investigated problem, which can be addressed by various methods such as optimization theory [141] or game theory [142]. Generally, resource prices can be determined by the amount of remaining resources, i.e., the higher amount of remaining resources type d is, the lower its weight is. In practice, the resources are priced based on MISPs' business strategies, e.g., [143,144]. Note that optimizing weights for resource types is out of scope in this work. In addition, the simulations in Section 5 are performed to study the impact of immediate

reward on the system performance.

5.2.6 Optimization Formulation

Since the statistical characteristics (e.g., arrival rate and departure rate of a MetaSlice) of the proposed system are stationary (i.e., time-invariant), the policy π for the meta-controller can be described as the time-invariant mapping from the state space to the action space, i.e., $\pi : \mathcal{S} \rightarrow \mathcal{A}_{\mathbf{s}}$. This study aims to find an optimal policy for the MetaSlicing's Admission Controller that maximizes a long-term average reward function $R_\pi(\mathbf{s})$, which is defined as an average expected reward obtained by starting from state $\mathbf{s}$ and following policy π as follows:

$$\mathcal{R}_\pi(\mathbf{s}) = \lim_{G\rightarrow\infty} \frac{\mathbb{E}\big[\sum_{g=0}^{G} r(\mathbf{s}_g, \pi(\mathbf{s}_g))|\mathbf{s}_0 = \mathbf{s}\big]}{\mathbb{E}\big[\sum_{g=0}^{G} \tau_g|\mathbf{s}_0 = \mathbf{s}\big]}, \forall \mathbf{s} \in \mathcal{S}, \tag{5.9}$$

where G is the total number of decision epochs, $\pi(\mathbf{s}_g)$ is the action derived by π at decision epoch g, and τ_g is the interval time between two consecutive decision epochs. The existence of the limit in $\mathcal{R}_\pi(\mathbf{s})$ is proven in Theorem 5.1. Thus, given the currently available resources and the MetaSlice request's information, the optimal policy π^* can give optimal actions to maximize $\mathcal{R}_\pi(\mathbf{s})$, thereby maximizing the long-term revenue for the infrastructure provider.

Theorem 5.1. *Given that the state space $\mathcal{S}$ and the number of decision epochs in a certain period of time are finite, we have:*

$$\mathcal{R}_\pi(\mathbf{s}) = \lim_{G\rightarrow\infty} \frac{\mathbb{E}\big[\sum_{g=0}^{G} r(\mathbf{s}_g, \pi(\mathbf{s}_g))|\mathbf{s}_0 = \mathbf{s}\big]}{\mathbb{E}\big[\sum_{g=0}^{G} \tau_g|\mathbf{s}_0 = \mathbf{s}\big]} \tag{5.10}$$

$$= \frac{\overline{\mathcal{T}}_\pi r(\mathbf{s}, \pi(\mathbf{s}))}{\overline{\mathcal{T}}_\pi y(\mathbf{s}, \pi(\mathbf{s}))}, \qquad \forall \mathbf{s} \in \mathcal{S}, \tag{5.11}$$

where $r(\mathbf{s}, \pi(\mathbf{s}))$ is the expected immediate reward and $y(\mathbf{s}, \pi(\mathbf{s}))$ is the expected interval between two successive decision epochs when performing action $\pi(\mathbf{s})$ at state

$\mathbf{s}$, *and* $\overline{\mathcal{T}}_\pi$ *is the limiting matrix of the embedded Markov chain corresponding to policy* π*, which is given based on the transition probability matrix of this chain, i.e.,* $\mathcal{T}_\pi$*, as follows:*

$$\overline{\mathcal{T}}_\pi = \lim_{G\to\infty} \frac{1}{G} \sum_{g=0}^{G-1} \mathcal{T}_\pi^g. \tag{5.12}$$

Proof. The proof of Theorem 5.1 is presented in Appendix A.1. □

Note that the underlying Markov chain of our sMDP model is irreducible (i.e., the long-term average reward is independent of the starting state), which is proven in Theorem 5.2.

Theorem 5.2. *The long-term average reward* $R_\pi(\mathbf{s})$ *for any policy* π *is well-defined and independent of the starting state, i.e.,* $R_\pi(\mathbf{s}) = R_\pi, \forall \mathbf{s} \in \mathcal{S}$.

Proof. The proof of Theorem 5.2 is presented in Appendix A.2. □

Since the limiting matrix $\overline{\mathcal{T}}_\pi$ exists and the sum of probabilities that the system moves from one state to others is one, we have $\sum_{\mathbf{s}'\in\mathcal{S}} \overline{\mathcal{T}}_\pi(\mathbf{s}'|\mathbf{s}) = 1$. Given the above, the MetaSlice admission control problem can be formulated as follows:

$$\begin{aligned} \max_{\pi} \quad & \mathcal{R}_\pi = \frac{\overline{\mathcal{T}}_\pi r(\mathbf{s}, \pi(\mathbf{s}))}{\overline{\mathcal{T}}_\pi y(\mathbf{s}, \pi(\mathbf{s}))}, \\ \text{subject to:} \quad & \sum_{\mathbf{s}'\in\mathcal{S}} \overline{\mathcal{T}}_\pi(\mathbf{s}'|\mathbf{s}) = 1, \quad \forall \mathbf{s} \in \mathcal{S}. \end{aligned} \tag{5.13}$$

Note that while the problem studied in this work involves resource management for Metaverse, it may seem at first glance that our problem formulation is similar to conventional resource allocation problems, such as virtualized network function (VNF) resource allocation or network slicing [145–147]. However, they are fundamentally different. In particular, our proposed resource management framework, MetaSlicing, leverages the similarities in functions between Metaverse applications

to improve system resource utilization significantly. In contrast, conventional resource allocation approaches do not consider function similarities between applications when allocating resources [54, 56, 57]. Therefore, it makes our mathematical optimization problems clearly different from conventional resource allocation. We first propose the MetaSlice decomposition to divide a Metaverse application into different functions that can operate and be initialized independently. Then, we introduce the MetaInstance technique to group Metaverse applications into different groups according to their function similarities in which some functions can be shared by multiple Metaverse applications. For that, we propose a method to identify the similarities in functions between MetaSlices. Finally, an admission control problem can be adopted to allocate resources for requests of MetaSlices based on system's available resources and similarity between the MetaSlice request and the ongoing MetaSlices in the system. Clearly, in addition to to optimizing different types of resources as in conventional VNF resource optimization, we need to consider other important factors related to the deployment of Metaverse applications in practice, i.e., decomposition of Metaverse applications (i.e., MetaSlices) and identifying similarities between MetaSlices. Thus, the considered problem requires a fundamentally different approach than conventional VNF resource allocation approaches.

It is worth mentioning that due to the discrete action space, (5.9) and (5.13) are not convex, making them challenging to solve by conventional methods. In the following section, we will discuss our proposed solution that can help the Admission Controller to obtain the optimal admission policy π^* to maximize the long-term average reward function, i.e., $\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathcal{R}_\pi$.

5.3 AI-based Solution with MetaSlice Analysis for MetaSlice Admission Management

This section presents our proposed approach for the MetaSlice admission management. We first discuss the main steps in the MetaSlice analysis to determine the similarity between a requested MetaSlice and ongoing MetaSlices. Then, we propose a Deep Reinforcement Learning (DRL)-based algorithm for the Admission Controller to address the real-time decision making requirement and the high uncertainty and dynamics of the request's arrival and MetaSlice departure processes, which are, in practice, often unknown in advance. Thanks to the self-learning ability of DRL, the Admission Controller can gradually obtain an optimal admission policy via interactions with its surrounding environment without requiring complete knowledge of the arrival and departure processes of MetaSlices in advance.

5.3.1 MetaSlice Analysis

Recall that the major role of the MetaSlice Analyzer is to analyze the request's MetaBlueprint to determine the similarity between the requested MetaSlice and the ongoing MetaInstances. Then, the similarity report is used to assist the Admission Controller in deciding whether to accept or reject a request. This work uses the function configuration to decide the similarity index since it clearly shows the relationship between the requested MetaSlice and the ongoing MetaInstance.

We consider that the proposed framework supports F types of functions. Then, we can denote the function configuration of a MetaSlice m by a set $\mathcal{F}_m$ as follows:

$$\mathcal{F}_m = \left\{ \mathbf{f}_1^m, \ldots, \mathbf{f}_f^m, \ldots, \mathbf{f}_F^m \right\}, \tag{5.14}$$

where the configuration of function f is represented by a vector with size K, i.e., $\mathbf{f}_f \in \{0,1\}^K$. We define a trivial function configuration vector $\mathbf{f}^* \triangleq (0, \ldots, 0)$ mean-

ing that a function is not required by the MetaSlice. Note that a MetaInstance also maintains a function configuration set and updates it whenever the MetaSlice is initialized or released.

Given two function configuration sets $\mathcal{F}_1$ and $\mathcal{F}_2$, the similarity index can be given as follows:

$$j(\mathcal{F}_1, \mathcal{F}_2) = \frac{1}{F} \sum_{f=1}^{F} b(\mathbf{f}_f^1, \mathbf{f}_f^2), \tag{5.15}$$

where b can be any similarity function used for vectors such as Jaccard and Cosine similarity functions [148]. Here, we use Jaccard similarity that is defined as follows:

$$b_{\text{Jaccard}}(\mathbf{f}_f^1, \mathbf{f}_f^2) = \frac{\mathbf{f}_f^1 \cdot \mathbf{f}_f^2}{||\mathbf{f}_f^1||^2 + ||\mathbf{f}_f^2||^2 - \mathbf{f}_f^1 \cdot \mathbf{f}_f^2}, \tag{5.16}$$

where the numerator is the dot product of two vectors and $|| \cdot ||$ is the Euclidean norm of the vector, which is calculated as the square root of the sum of the squared vector's elements.

Recall that the similarity index j is one of the four elements of the state, and in reinforcement learning, an agent determines its action based on the observation of the state. Thus, the similarity index plays two important roles in the MetaSlicing framework. First, it supports the Admission Controller based on a reinforcement learning algorithm with precious information for making decisions. Second, based on the accepted request's similarity index, MetaSlicing's resource management determines to put the accepted MetaSlice request in an existing MetaInstance or create a new MetaInstance.

5.3.2 Deep Dueling Double Q-learning based-Admission Controller

In RL, Q-learning is widely adopted due to its simplicity in implementation and convergence guarantee after the learning phase [149]. Nevertheless, using a table to estimate the optimal values of all state-action pairs $Q^*(\mathbf{s}, a)$, i.e., Q-values, hinders

the Q-learning from being applied in a high-dimensional state space as the problem considered in this work with hundred thousand states [101]. In addition, the usage of the Q-table is only feasible when values of states are discrete, but the similarity score in the considered state can be a real number. These challenges are addressed by deep Q-network (DQN) algorithm, in which a Deep Neural Network (DNN), instead of a Q-table, is used to approximate $Q^*(\mathbf{s}, a)$ for all state-action pairs [150]. However, both Q-learning and DQN have the same problem of overestimation when estimating Q-values [31]. This issue makes the learning process unstable or even results in a sub-optimal policy if overestimations are not evenly distributed across states [102]. To that end, this thesis proposes to leverage D3QL, discussed in Section 2.2.4, for the MetaSlicing Admission Controller's algorithm, namely iMSAC, that can address these above issues effectively by leveraging three innovative techniques: (i) the memory replay mechanism, (ii) the Dueling neural network architecture, and (iii) the double deep Q-network (DDQN) algorithm. The iMSAC model is illustrated in Figure 5.2.

Note that most recently proposed DRL algorithms are on-policy methods, e.g., Asynchronous Advantage Actor-Critic (A3C) and Proximal Policy Optimization (PPO), and aim to overcome the limitation of DQN in problems with continuous action space [151, 152]. As such, they may not perform well in problems with discrete action space, such as the considered problem in this work. For instance, the study in [153] shows that the Dueling DDQN outperforms the A3C in most of 57 Atari games with discrete action spaces. Similarly, results in [154] show that the performance of PPO and Dueling DDQN are similar in a discrete action problem, and the run time of PPO is almost two times greater than that of Dueling DDQN. Moreover, the authors in [155] demonstrate that PPO's performance is heavily influenced by its code-level implementation. Hence, the PPO may require significant effort to optimize hyper-parameters, which is not the focus of our work. In contrast,

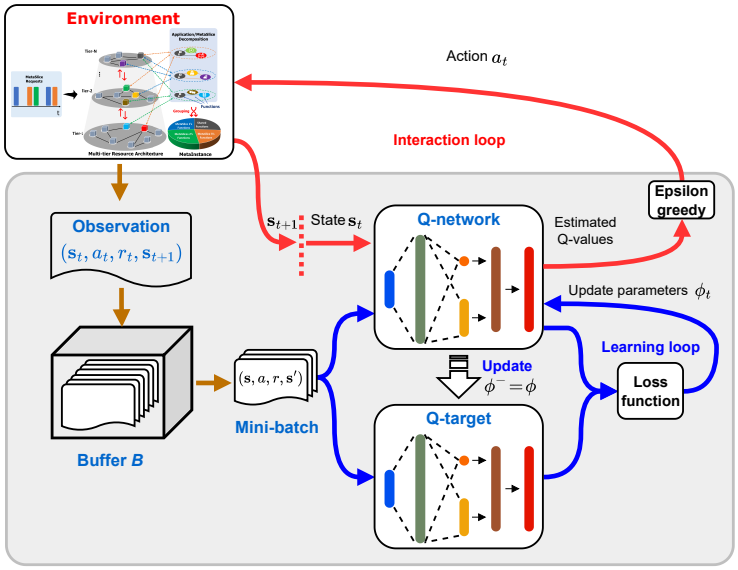


Figure 5.2 : The proposed iMSAC-based Admission Controller for the MetaSlicing framework.

Dueling DDQN can achieve good performance with typical hyper-parameters, as those in [34, 150]. More importantly, since A3C and PPO are on-policy methods, they must use up-to-date experiences collected by their current policies to learn an optimal policy [99]. As such, old experiences are dropped after each learning iteration, making the sample efficiencies of these methods low. Here, the sample efficiency presents how much an RL algorithm can get the most use of every experience. In contrast, the Dueling DDQN (the fundamental component of D3QL) is an off-policy method that can leverage experiences collected from any other policies [99]. Given the above observations, the Dueling DDQN is the most appropriate learning approach for the considered problem as it can perform well with typical hyper-parameters and has high sample efficiency.

The computational complexity of the proposed iMSAC is mainly determined by the training process of a deep neural network, i.e., Q-network that represents the admission policy. As iMSAC is based on D3QL, its discussion of the optimality and computational complexity was discussed in Section 2.2.5. Since the Q-network only

consists of conventional components, e.g., fully connected layers and tanh activation, the decision latency is very marginal since decisions can be made mostly instantly by feeding the state to the Q-network. In this work, simulations are performed on a typical laptop with the AMD Ryzen 3550H (4 cores at 2.1 GHz) and 16GB RAM. We record that the average decision latency is 6.2 μs. In practice, several low latency-constrained applications have adopted DNN, such as Tesla Autopilot [106] and ALVINN [107], and thus they clearly demonstrate the applicability of iMSAC for real-time systems.

5.4 Performance Evaluation

5.4.1 Simulation Parameters

The parameters for our simulation are set as follows, unless otherwise stated. We consider that the system supports up to nine types of functions, i.e., $F=9$. Each MetaSlice consists of three different functions and belongs to one of three classes, i.e., class-1, class-2, and class-3. In the configuration set $\mathcal{F}$, we set $K=1$. Here, we set λ_1, λ_2, and λ_3 to 60, 40, and 25 requests/hour, respectively, and its vector is denoted by $\boldsymbol{\lambda} = [60, 40, 25]$. The average MetaSlice session time is 30 minutes, i.e, $\mu_i = 2,\ \forall i \in \{1, 2, 3\}$. The revenues r_i for accepting a MetaSlice request from class i are set as $r_1 = 1, r_2 = 2, r_3 = 4$. Note that our proposed algorithm, i.e., iMSAC, does not require the above information in advance. It can adjust the admission policy according to the practical requirements and demands (e.g., rental fees, arrival rate, and total resources of the system) to maximize a long-term average reward. Therefore, without loss of generality, the system has three types of resources, i.e., computing, storage, and radio bandwidth. Each Metaverse function is assumed to require a similar amount of resources as those of the Network Slice in 5G Network Slicing [156], e.g., 40 GB for storage, a bandwidth of 40 MHz, and 40 GFLOPS/s for computing.

In our proposed algorithm, i.e., iMSAC, the settings are as follows. For the ϵ-greedy policy, the value of ϵ is gradually decreased from 1 to 0.01. The discount factor γ is set to 0.9. We use Pytorch to build the Q-network and the target Q-network. They have the same architecture as shown in Figure 2.4. During the learning process, typical hyperparameters of DNN are selected as those in [150] and [31], e.g., the learning rate of the Q-network is set at 10^{-3} and the target-Q network's parameters are copied from the parameters of Q-network at every 10^4 steps.

Recall that our proposed solution consists of two important elements, i.e., the intelligent algorithm iMSAC and MetaSlice analysis with the MetaInstance technique (MiT). With MetaInstance, functions can be reused, leading to a significant improvement in resource utilization. Meanwhile, the iMSAC can help the Admission Controller to obtain an optimal policy without requiring the complete information about the arrival and departure of MetaSlice in advance. Therefore, we compare our proposed solution, i.e., iMSAC+MiT, with three counterpart approaches: (i) iMSAC, (ii) Greedy policy [99] where the MetaSlicing's Admission Controller accepts a request if the system has enough resources for the request, and (iii) Greedy policy with the MetaInstance technique, i.e., Greedy+MiT. Recall that since the iMSAC approach does not leverage MetaInstance and is a RL-based resource management, it can be straightforwardly adopted to various scenarios, such as network slicing. Thus, performance of iMSAC can provide us insights on how existing admission control approaches perform in Metaverse.

5.4.2 Simulation Results

Simulations are conducted to gain insights into our proposed solution, i.e., iMSAC+MiT. First, we will investigate the convergence rate of our proposed algorithm iMSAC. Then, we evaluate the proposed solution in different scenarios to study

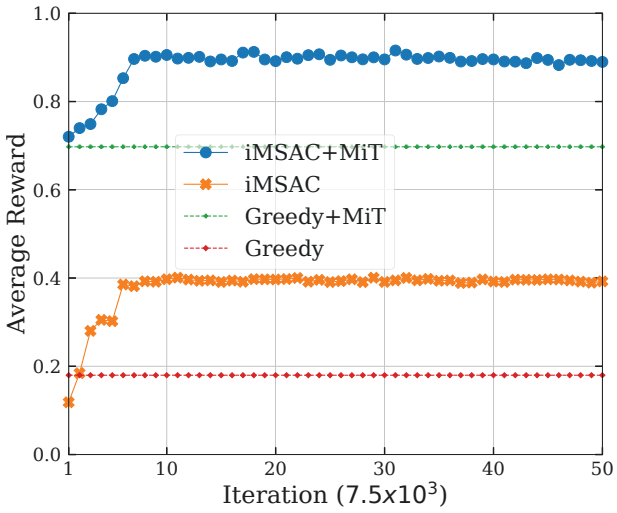


Figure 5.3 : Convergence rate of iMSAC.

impacts of important system parameters, e.g., the available system resources, immediate rewards reflecting the revenue of the MISP, and the maximum number of MetaSlices sharing one function that is one of the most important parameters of the MISP.

5.4.2.1 Convergence Rate

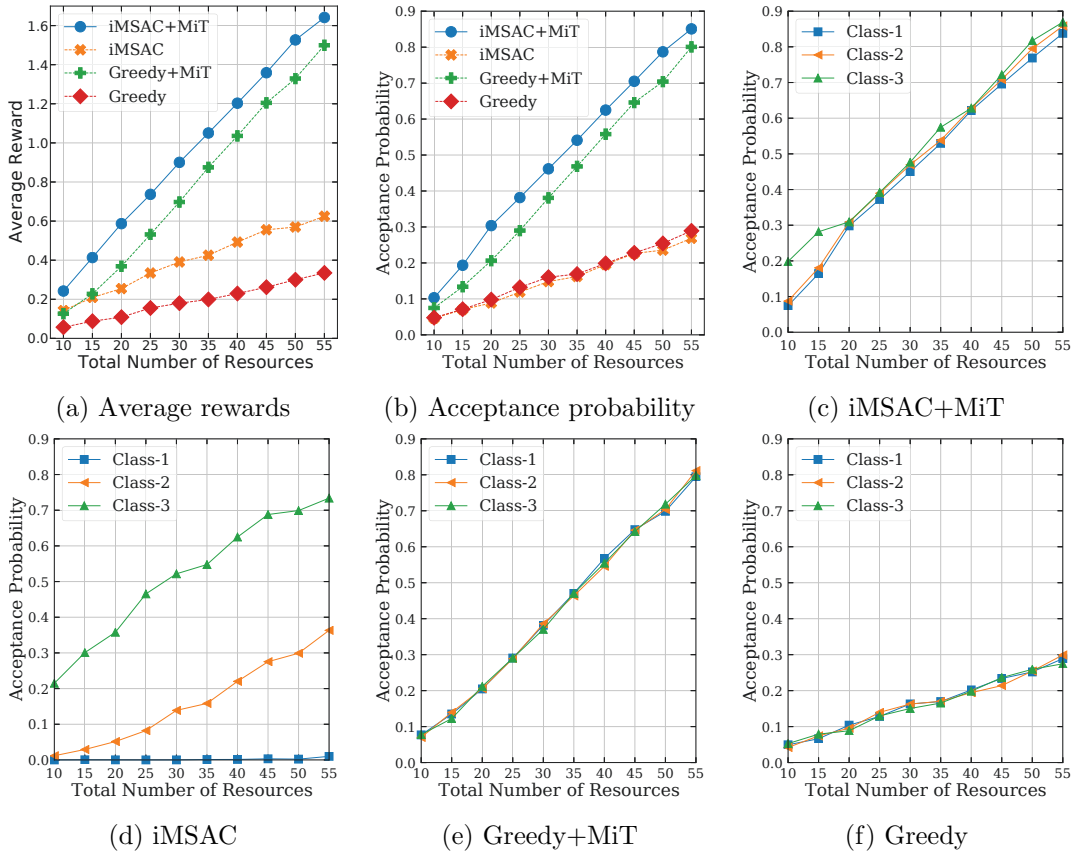
Figure 5.3 shows the convergence rates of our proposed iMSAC algorithm in two scenarios with and without the MetaInstance technique. In this experiment, we set storage, radio bandwidth, and computing resources to 1200 GB, 1200 MHz, and 1200 GFLOPS/s, respectively. In other words, the system can support up to 30 functions in total. The average rewards obtained by Greedy+MiT and Greedy are also presented for comparisons. Specifically, the learning curves of iMSAC+MiT and iMSAC have a very similar trend. As shown in Figure 5.3, both of them gradually converge to the optimal policy after 6×10^4 iterations. However, the iMSAC+MiT's average reward is stable at 0.9, which is 2.25 times greater than that of the iMSAC. Similarly, the Greedy+MiT's average reward is much greater (i.e., 3.5 times) than that of the Greedy. Thus, these results clearly show the benefits of the iMSAC and

the MetaInstance technique. In particular, while the MetaInstance can help to maximize the resource utilization for the system, the iMSAC can make the Admission Controller learn the optimal policy to maximize the long-term average reward.

5.4.2.2 Performance Analysis

We now investigate the robustness of our proposed solution, i.e., iMSAC+MiT, in different scenarios. First, we vary the storage, radio, and computing resources from 400 GB, 400 MHz, and 400 GFLOPS/s to 2200 GB, 2200 MHz, and 2200 GFLOPS/s, respectively. In other words, the total number of functions supported by the system is varied from 10 to 55. The policies of iMSAC+MiT and iMSAC are obtained after 3.75×10^5 learning iterations. In this scenario, two metrics for evaluating the Admission Controller's performance are average reward and acceptance probability since they clearly show the effectiveness of the admission policy in terms of the income for MISP (i.e., average reward) and the service availability for end-users. Figure 5.4(a) clearly shows that as the total amount of system resources increases, the average rewards of all approaches increase. This is due to the fact that the higher the total resource is, the higher number of MetaSlice the system can host, and thus the greater revenue the system can achieve. It is observed that the proposed algorithm iMSAC+MiT always obtains the highest average reward, up to 80% greater than that of the second-best policy in this scenario, i.e., Greedy+MiT. Similarly, Figure 5.4(b) shows that iMSAC+MiT achieves the highest acceptance probability for an arriving MetaSlice request, up to 47% greater than that of the Greedy+MiT, i.e., the second-best policy. In addition, Figures 5.4(a) and (b) demonstrate the benefit of MetaInstance. In particular, it helps the system to increase the average rewards and acceptance rates of both iMSAC and Greedy by up to 396% and 222%, respectively.

To gain more insights, we look further at the acceptance probability for each class

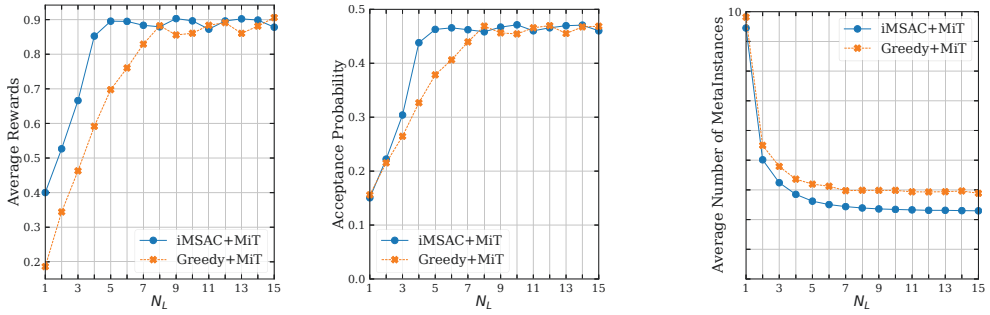


(a) Average rewards (b) Acceptance probability (c) iMSAC+MiT

(d) iMSAC (e) Greedy+MiT (f) Greedy

Figure 5.4 : Vary the total number of system resources.

of MetaSlice. As shown in Figures 5.4(c) and (d), for the iMSAC+MiT and iMSAC, the acceptance probabilities of class-3 are always higher than those of other classes when the number of resources increases. Meanwhile, Greedy+MiT and Greedy accept requests from all the classes at almost the same probability, as depicted in Figures 5.4(e), and (f). Recall that the arrival rate of class-3 is the lowest value (i.e., $\lambda_3 = 25$), while the immediate reward for accepting requests class-3 is the greatest value, i.e., $r_3 = 4$. Thus, the proposed algorithm iMSAC can learn and adjust its policy to achieve the best result. More interestingly, for the iMSAC+MiT results, the acceptance probability of class-3 has significant gaps (up to 50% greater than those of other classes) when the total resources are little (i.e., less than 20), as shown in Figure 5.4. This stems from the fact that if the available resources are low, the Admission Controller should reserve resources for future requests from class-3 with

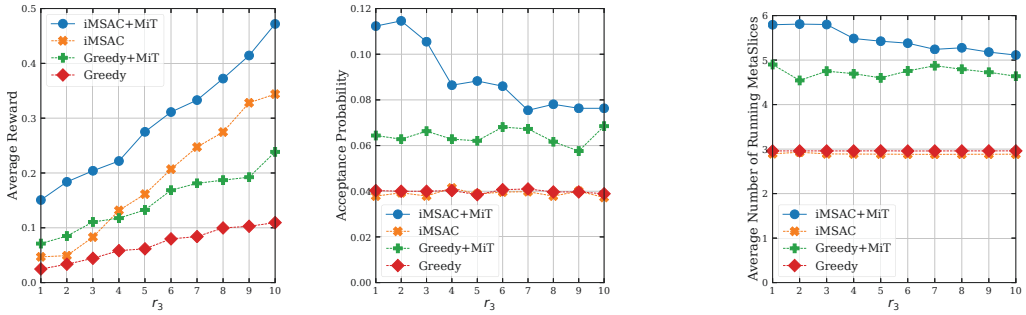


(a) Average rewards (b) Acceptance probability (c) Average No. of MetaInstances

Figure 5.5 : Varying N_L

the highest reward. In contrast, if the system has more available resources, the Admission Controller should accept requests from all classes more frequently. This observation is also shown in Figure 5.4(d), where the MetaInstance is not employed.

Next, we investigate one of the most important factors in the MetaSlicing framework, which is the maximum number of MetaSlices that share the same function, denoted by N_L. In this experiment, we set the resources the same as those in Figure 5.3, and other settings are set the same as those in Section 5.4.1. Figure 5.5(a) shows that as the value of N_L increases from 1 to 15, the average rewards obtained by our proposed solution iMSAC+MiT and Greedy+MiT first increase and then stabilize at around 0.88. Remarkably, when the value of N_L is small (i.e., less than 8), the iMSAC+MiT's average reward is always greater than that of Greedy+MiT, up to 122%. The reason is that as N_L increases, meaning that more MetaSlices can share the same function, the number of MetaSlices that can be deployed in the system increases. In other words, the MetaSlicing system capacity increases according to the increase of N_L. As such, the Admission Controller can accept more requests to obtain greater rewards when N_L increases. In addition, the thresholds in average rewards for both approaches originate from the fact that the arrival and departure processes of class i follow the Poison and exponential distributions with a fixed mean λ_i and $1/\mu_i$.

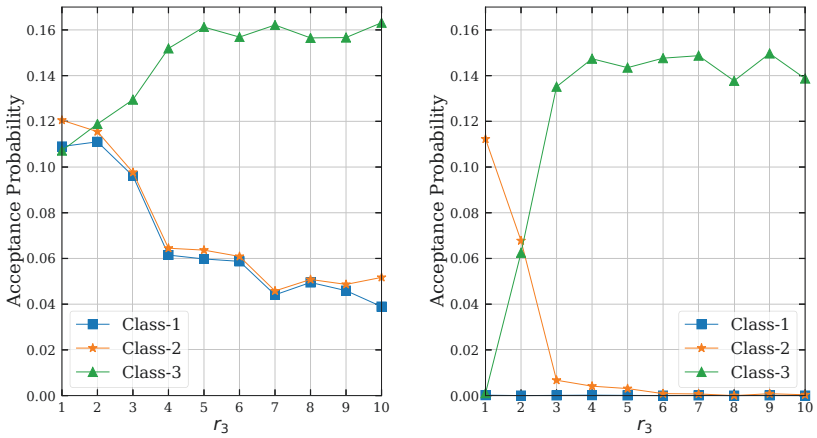


(a) Average rewards (b) Acceptance probability (c) Average No. of running MetaSlices

Figure 5.6 : Vary the immediate reward of class-3.

In terms of the acceptance probability for a MetaSlice request, similar observations can be made in Figure 5.5(b). Specifically, our proposed solution maintains higher request acceptance probabilities (up to 34%) than that of the Greedy+MiT when N_L is less than 8. As N_L increases, the acceptance probabilities of both approaches increase, then they are stable at around 0.46 when $N_L > 4$ for iMSAC+MiT and $N_L > 7$ for Greedy+MiT. The reasons are similar as those in Figure 5.5(a). In particular, the higher the system capacity is, the higher the request acceptance probability is. Unlike the above metrics, the average numbers of MetaInstances decrease for both approaches as the value of N_L increases from 1 to 15, as shown in Figure 5.5(c). The reason is that an increase of N_L can result in increasing the number of MetaSlices in a MetaInstance, thereby decreasing the number of MetaInstances in a system with a fixed capacity. The above observations in Figure 5.4 and Figure 5.5 show the superiority of our proposed approach compared with others, especially when the system resources are very limited.

We continue evaluating our proposed solution in the case where the immediate reward of class-3, i.e., r_3, is varied from 1 to 10. In this experiment, we set the storage, radio, and computing resources to 400 GB, 400 MHz, and 400 GFLOPS/s, respectively. The arrival rate vector of MetaSlice is set to $\boldsymbol{\lambda} = [60, 50, 40]$ to explore the robustness of our proposed solution. In Figure 5.6(a), as r_3 increases, the average



(a) iMSAC+MiT (b) iMSAC

(c) Greedy+MiT (d) Greedy

Figure 5.7 : The acceptance probability per class when varying the immediate reward of class-3.

rewards obtained by all approaches increase. In particular, the results demonstrate that our proposed solution, i.e., iMSAC+MiT, consistently achieves the highest average reward, up to 111% greater than that of the second-best, i.e., Greedy+MiT when $r_3 = 1$. Interestingly, when r_3 is small (i.e., less than 4), the iMSAC's average rewards are lower than those of the Greedy+MiT. However, when r_3 becomes larger than or equal to 4, the average rewards obtained by iMSAC are higher than those of the Greedy+MiT.

Similarly, Figures 5.6(b) and (c) show that our proposed solution always obtains the highest values compared to those of other approaches in terms of the acceptance

probability and average number of running MetaSlices when r_3 increases from 1 to 10. Interestingly, even with a decrease in the acceptance probability and average number of running MetaSlices, the average rewards obtained by the iMSAC+MiT increase as r_3 is varied from 1 to 10, as shown in Figure 5.6. The reason is that when the immediate reward of class-3 is very high (e.g., $r_3 = 10$) compared to those of class-1 and class-2 (i.e., 1 and 2, respectively), the iMSAC+MiT reserves more resources for the future requests of class-3.

We now further investigate the above observations when varying the immediate reward of requests class-3 by looking deeper at the acceptance probability per class for each approach. Figures 5.7(a)-(d) illustrate the acceptance rate per class according to the policies obtained by the proposed and counterpart approaches. In Figure 5.7(c), the Greedy+MiT's acceptance probabilities for all classes are almost the same, at around 0.06, when the immediate reward of class-3 increases from 1 to 10. A similar trend is observed for the Greedy but at a lower value, i.e., 0.04, in Figure 5.7(d). In contrast to Greedy and Greedy+MiT, iMSAC+MiT's acceptance probability for class-3 increases while those of other classes decrease as r_3 increases from one to 10, as shown in Figure 5.7(a). More interestingly, when the immediate reward of class-3 requests is small (i.e., $r_3 < 2$), class-3 requests have the lowest acceptance probability compared to those of other classes. However, when the immediate reward of class-3 requests is larger than or equal to 2, class-3 requests will achieve the highest acceptance probability compared with those of other classes. Moreover, when $r_3 > 4$, the acceptance probability for class-3 requests obtained by the iMSAC+MiT is stable at around 0.16.

Similar to the iMSAC+MiT, the iMSAC's acceptance probability for class-3 requests also increases until reaching a threshold with a lower value, i.e., around 0.14, compared to that of the iMSAC+MiT. Furthermore, the acceptance probability of class-3 requests obtained by the iMSAC-base solutions is up to four-time greater

than those of the Greedy-based solutions. Thus, the iMSAC+MiT and iMSAC can obtain a good policy in which the acceptance probability of a class increases if its reward increases compared with the rewards of other classes, and vice versa. Note that our proposed solution does not need complete information about the MetaSlice's arrival and departure processes in advance. However, as observed, the proposed solution can always achieve the best results in all scenarios when we vary important parameters of the system.

The above findings underscore the efficacy of the proposed MetaSlicing framework in managing multi-tier resource architecture to meet requirements of Metaverse applications. The system must balance between accepting more requests (increasing acceptance probability) and optimizing the allocation of these resources to maximize rewards. The integration of MiT with the intelligent iMSAC algorithm enables a dynamic and efficient allocation of resources, significantly outperforming conventional approaches and demonstrating the potential for real-time, high-performance Metaverse application management.

5.5 Conclusion

In this chapter, we have proposed two innovative techniques, i.e., the application decomposition and the MetaInstance, to maximize resource utilization for the Metaverse built on a multi-tier computing architecture. Based on these techniques, we have developed a novel framework for the Metaverse, i.e., MetaSlicing, that can smartly allocate resources for MetaSlices, i.e., Metaverse applications, to maximize the system performance. Moreover, we have proposed a highly effective framework based on sMDP together with an intelligent algorithm, i.e., iMSAC, to find the optimal admission policy for the Admission Controller under the high dynamics and uncertainty of resource demands. The extensive simulation results have clearly demonstrated the robustness and superiority of our proposed solution com-

pared with the counterpart methods as well as revealed key factors determining the system performance. As Metaverse continues to evolve and expand, there will be enormous potential research topics that we can explore. For example, elastic resource allocation approaches can be developed to further enhance the efficiency of using MetaSlices. In particular, allocated resources of a MetaSlice can be scaled dynamically according to the resource demand of this MetaSlice during operation, thus further improving resource utilization. Another topic that we can further consider is methods for ensuring interoperability between different MetaSlices in the Metaverse. This could include developing standards and protocols for communication between different MetaSlices and mechanisms for managing conflicts and ensuring consistency between Metaslices.

Chapter 6

Countering Eavesdroppers with Meta-learning-based Cooperative Ambient Backscatter Communications

The previous chapter has addressed the resource management challenges in 6G networks to facilitate emerging Metaverse applications. This chapter introduces a novel lightweight framework using ambient backscattering communications, an emerging technology enabling 6G networks [1], to counter eavesdroppers. In particular, our framework divides an original message into two parts. The first part, i.e., the active-transmit message, is transmitted by the transmitter using conventional RF signals. Simultaneously, the second part, i.e., the backscatter message, is transmitted by an ambient backscatter tag that backscatters upon the active signals emitted by the transmitter. Notably, the backscatter tag does not generate its own signal, making it difficult for an eavesdropper to detect the backscattered signals unless they have prior knowledge of the system. Here, we assume that without decoding/knowing the backscatter message, the eavesdropper is unable to decode the original message. Even in scenarios where the eavesdropper can capture both messages, reconstructing the original message is a complex task without understanding the intricacies of the message-splitting mechanism. A challenge in our proposed framework is to effectively decode the backscattered signals at the receiver, often accomplished using the maximum likelihood (MLK) approach. However, such a method may require a complex mathematical model together with perfect channel state information (CSI). To address this issue, we develop a novel deep meta-learning-based signal detector that can not only effectively decode the weak backscattered signals without requir-

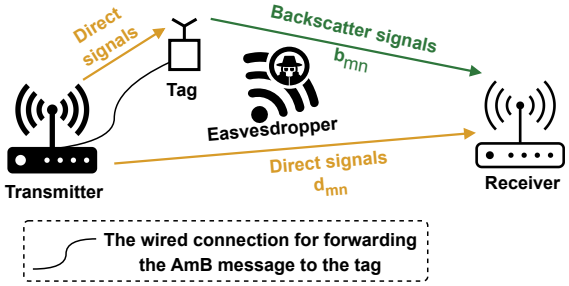


Figure 6.1 : Anti-eavesdropping attack system model.

ing perfect CSI but also quickly adapt to a new wireless environment with very little knowledge. Simulation results show that our proposed learning approach, without requiring perfect CSI and complex mathematical model, can achieve a bit error ratio close to that of the MLK-based approach. They also clearly show the efficiency of the proposed approach in dealing with eavesdropping attacks and the lack of training data for deep learning models in practical scenarios.

This chapter is organized as follows. Our proposed anti-eavesdropping system and the channel model are discussed in Sections 6.1 and 6.2. Then, Sections 6.3 and 6.4 present the MLK-based detector and our proposed DL-based detector for the AmB signal, respectively. Our proposed deep meta-learning-based approach is presented in Section 6.5, and Section 6.6 discusses our simulation results. Finally, Section 6.7 wraps up this chapter with a conclusion.

6.1 System Model

This work considers a wireless network with the presence of an eavesdropper, as shown in Figure 6.1. Here, the transmitter has a single antenna, while the receiver is equipped with M antennas. The eavesdropper aims to wiretap the transmitted signals to gather the information in this channel. To cope with the eavesdropper, this work deploys a low-cost and low-complexity tag equipped with the AmB technology, allowing it to send data by backscattering ambient RF signals without producing active signals as in conventional active transmissions. Specifically, the AmB tag

has two operation states, including (i) the absorbing state, where the tag does not reflect incoming signals, and (ii) the reflecting state, where the tag reflects incoming signals. In this way, the AmB tag can transmit data without using any active RF components. Before sending a message to the receiver, the transmitter first encodes it into two messages: (i) an active message for the transmitter and (ii) an AmB message forwarded to the AmB tag via the wired channel. Note that the use of a single-antenna transmitter stems from our study's aim to develop a lightweight anti-eavesdropping framework, similar to [82, 83]. If beamforming were to be performed at a multiple-antenna transmitter, it would necessitate feedback from the AmB tag, leading to increased system complexity.

The active message is then directly transmitted by the transmitter to the receiver using the conventional RF transmission method. At the same time, when the transmitter transmits the active message, the AmB tag will leverage its RF signals to backscatter the AmB message [69, 157]. Thus, the receiver will receive two data streams simultaneously, one over the conventional channel d_{mn} and another over the backscatter channel b_{mn}, as depicted in Figure 6.1. Note that instead of producing active signals, the AmB tag only backscatters/reflects signals. Therefore, wiretapping AmB signals is intractable unless the eavesdropper has prior knowledge about the system configuration, i.e., the usage of AmB and the exact backscatter rate. As a result, our system can provide a new deception strategy for data transmissions. Specifically, the eavesdropper is attracted by the active signals generated by the transmitter, so it pays less attention to (or is even unaware of) the presence of AmB communications. Here, we assume that without obtaining information from the AmB message, the eavesdropper cannot decode the information from the original message. Moreover, even in the worse case in which the eavesdropper knows the presence of AmB communication in the system, they cannot obtain the original message straightforwardly without the knowledge about the system's message-splitting

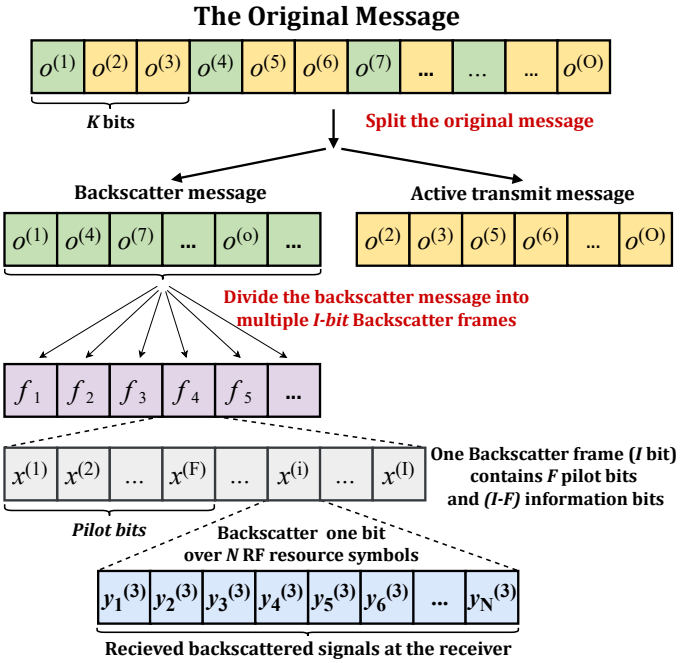


Figure 6.2 : Message-splitting mechanism.

mechanism. As a result, our proposed approach can deal with eavesdropping attacks in wireless systems.

The detail of our proposed encoding mechanism is depicted in Figure 6.2. Specifically, at the beginning, an encoding mechanism is used to split the original message into two parts, i.e., backscatter message and active transmit message. Note that our framework here and following analysis can adopt any encoding mechanism, and the design of encoding mechanism is out of the scope of this work. Since the AmB rate is usually lower than the active transmission rate, the AmB message's size can be designed to be smaller than the active message's size. As such, the AmB message is constructed by taking bits at every K bits of the original message. By doing so, the system security under the presence of the eavesdropper can be significantly improved because the eavesdropper is unable to derive the message splitting mechanism and the backscatter message. Note that to improve the detection performance, we incorporate F pilot bits into the AmB frame, as depicted in Figure 6.2. The specific utilization and significance of these pilot bits will be extensively discussed in detail

Table 6.1 : Table of Notations

Notation	Description
M	The number of antennas
I	The number of bits in each backscatter frame
d_{mn}	The conventional channel
b_{mn}	The backscatter channel
F	The number of pilot bits
y_{mn}	The signal received at the m-th antenna at the n-th time
$\mathbf{y}$	The receiver's total received signals
σ_{mn}	The noise following CSCG
P_t	The transmitter's transmit power
P_{tr}	The average power received by the receiver
λ	The wave length
L_r, L_b	The transmitter-receiver and transmitter-tag distances
Gt, Gr	The antenna gains of the transmitter and the receiver
s_{tn}	The signal transmitted by the transmitter at time instant n
f_{dm}, f_{bm}	The fading of the transmitter-receiver and transmitter-tag-receiver links
P_b	The average power received by the AmB tag
G_b	The antenna gain of the AmB tag
g_r	The Rayleigh fading of the transmitter-tag link
l_n	The active signals from the transmitter at the AmB tag
e	The state of the AmB tag
s_{bn}	The backscattered signals
γ	The reflection coefficient
α_{dt}	The average SNRs of transmitter-to-receiver channel
α_{bt}	The average SNRs of transmitter-tag-receiver channel
$\mathbf{f}_d$	The channel response vectors of transmitter-to-receiver channel
$\mathbf{f}_b$	The channel response vectors of transmitter-tag-receiver channel
$Z(\cdot)$	The binary entropy function
θ_0 and θ_1	The probability of backscattering bit 0 and bit 1, respectively
$\mathcal{V}$	A realization of $\mathbf{y}$
$p(\mathcal{V}\|e)$	The conditional probability density function
$\mathbf{Y}^{(i)}$	The sequence of received signal corresponding to the period of the i-th AmB symbol

in Section 6.4.1.

6.2 Channel Model

This section presents the channel model of our considered system in detail. In particular, the AmB rate should be significantly lower than the sampling rate of the transmitter's signals so that the receiver can decode the backscattered signals with low BERs [69, 70, 88]. Formally, the transmit signals' sampling rate is assumed to

be N times higher than the AmB rate. In other words, each bit of the backscatter message is backscattered over N transmitter symbols. In the considered system, the receiver has M antennas ($M \geq 1$). Let y_{mn} denote the signal that the m-th antenna of the receiver receives at the n-th time. As illustrated in Figure 6.1, y_{mn} comprises (i) the active signal transmitted on the direct link, (ii) the signals backscattered by the AmB tag on the backscatter link, and (iii) noise from the surrounding environment. As such, y_{mn} can be expressed as follows:

$$y_{mn} = \underbrace{d_{mn}}_{\text{direct link}} + \underbrace{b_{mn}}_{\text{backscatter link}} + \sigma_{mn}, \tag{6.1}$$

where σ_{mn} is the noise following the unit-variance and zero-mean circularly symmetric complex Gaussian (CSCG), denoted by $\sigma_{mn} \sim \mathcal{CN}(0, 1)$. In the following, the signals on the direct and backscatter links are formally formulated.

6.2.1 Direct Channel

On the direct channel, i.e., the transmitter-receiver link, the average power received by the receiver is formulated by

$$P_{tr} = \frac{\kappa P_t G_t G_r}{{L_r}^{\upsilon}}, \tag{6.2}$$

where P_t is the transmitter's transmit power, and $\kappa = (\frac{\lambda}{4\pi})^2$ in which λ is the wavelength, υ denotes the path loss exponent. L_r is the transmitter-receiver distance. The antenna gains of the transmitter and the receiver are denoted by G_t and G_r, respectively. At time instant n, we denote the signal transmitted by the transmitter as s_{tn}. Then, at the receiver's m-th antenna, the direct link signal is given by

$$d_{mn} = f_{dm}\sqrt{P_{tr}}s_{tn}, \tag{6.3}$$

where f_{dm} represents the Rayleigh fading such that $\mathbb{E}[|f_{dm}|^2] = 1$ [88]. Note that the proposed AmB tag uses the AmB communication technology to backscatter s_{tn} without using any dedicated energy sources. Since the AmB tag does not know the active message, s_{tn} appears as random signals [82,88,158]. Therefore, we can assume that $s_{tn} \sim \mathcal{CN}(0, 1)$.

6.2.2 Ambient Backscatter Channel

As mentioned, the AmB tag backscatters the transmitter's active signals to transmit AmB messages. In the following, we present the formal formulation of the AmB signals at the receiver. Firstly, the average power received by the AmB tag can be defined by

$$P_b = \frac{\kappa P_t G_t G_b}{{L_b}^{\upsilon}}, \tag{6.4}$$

where G_b denotes the antenna gain of the AmB tag, and L_b denotes the transmitter-tag distance. Let g_r denote the Rayleigh fading of the link from the transmitter to the AmB tag, then the active signals from the transmitter at the AmB tag is given by

$$l_n = \sqrt{P_b} g_r s_{tn}. \tag{6.5}$$

As discussed above, the key idea of the AmB communication is to absorb or reflect surrounding ambient RF signals to convey information without generating any active signals. As such, we denote e as the state of the AmB tag. Specifically, $e = 1$ when the tag reflects the transmitter's signals, i.e., transmitting bits 1, and $e = 0$ when the tag absorbs the transmitter's signals, i.e., transmitting bits 0. Because each backscattered bit is backscattered over N transmitter's symbols, state e of the AmB tag remains unchanged during this period. The backscattered signals then can be given by

$$s_{bn} = \gamma l_n e, \tag{6.6}$$

where γ is the reflection coefficient. It is worth noting that γ captures all properties of the AmB tag such as load impedance and antenna impedance [158]. Let f_{bm} represent the Rayleigh fading of the AmB channel, and L_e denotes the AmB tag-to-receiver distance. Here, we can assume that $\mathbb{E}[|f_{bm}|^2] = 1$ and $\mathbb{E}[|g_r|^2] = 1$ without loss of generality [88]. Then, the signals in the AmB link received by the m-th antenna is given by

$$\begin{aligned} b_{mn} &= f_{bm}\sqrt{\frac{G_b G_r \kappa}{{L_e}^{\upsilon}}} s_{bn} \\ &= f_{bm}\sqrt{\frac{G_b G_r \kappa}{{L_e}^{\upsilon}}} \gamma e\Big(g_r \sqrt{P_b} s_{tn}\Big) \\ &= f_{bm} e\Big(g_r \sqrt{\frac{\kappa|\gamma|^2 G_b G_r P_b}{{L_e}^{\upsilon}}} s_{tn}\Big) \\ &= f_{bm} e\Big(g_r \sqrt{\frac{\kappa|\gamma|^2 G_b G_r \kappa P_t G_t G_b}{{L_e}^{\upsilon} {L_b}^{\upsilon}}} s_{tn}\Big) \\ &= f_{bm} e\Big(g_r \sqrt{\frac{\kappa|\gamma|^2 P_{tr} {G_b}^2 {L_r}^{\upsilon}}{{L_b}^{\upsilon} {L_e}^{\upsilon}}} s_{tn}\Big). \end{aligned} \tag{6.7}$$

By denoting $\tilde{\alpha}_r = \frac{\kappa|\gamma|^2 {G_b}^2 {L_r}^{\upsilon}}{{L_b}^{\upsilon} {L_e}^{\upsilon}}$, (6.7) can be rewritten as

$$b_{mn} = f_{bm} e\Big(g_r \sqrt{\tilde{\alpha}_r P_{tr}} s_{tn}\Big). \tag{6.8}$$

6.2.3 Received Signals

Now, we can obtain the received signals at the receiver's m-th antenna by substituting (6.3) and (6.8) to (6.1) as follows:

$$\begin{aligned} y_{mn} &= \underbrace{d_{mn}}_{\text{direct link}} + \underbrace{b_{mn}}_{\text{backscatter link}} + \sigma_{mn}, \\ &= f_{dm}\sqrt{P_{tr}} s_{tn} + f_{bm} e\Big(g_r \sqrt{\tilde{\alpha}_r P_{tr}} s_{tn}\Big) + \sigma_{mn}. \end{aligned} \tag{6.9}$$

Let $\alpha_{dt} \triangleq P_{tr}$ and $\alpha_{bt} \triangleq \tilde{\alpha}_r P_{tr}$ denote the average signal-to-noise ratios (SNRs) of the transmitter-to-receiver link and the transmitter-tag-receiver channel (i.e., backscatter link), respectively. Hence, (6.9) is rewritten as follows:

$$y_{mn} = \underbrace{f_{dm}\sqrt{\alpha_{dt}}s_{tn}}_{\text{direct link}} + \underbrace{f_{bm}e\Big(g_r\sqrt{\alpha_{bt}}s_{tn}\Big)}_{\text{backscatter link}} + \sigma_{mn}. \tag{6.10}$$

Since the receiver has M antennas, the channel response vectors corresponding to the backscatter and the direct channels and are respectively given by

$$\begin{aligned} \mathbf{f}_b &= [f_{b1}, \ldots, f_{bm}, \ldots, f_{bM}]^{\mathsf{T}}, \\ \mathbf{f}_d &= [f_{d1}, \ldots, f_{dm}, \ldots, f_{dM}]^{\mathsf{T}}. \end{aligned} \tag{6.11}$$

Then, the receiver's total received signals is expressed as

$$\mathbf{y}_n = \underbrace{\mathbf{f}_d\sqrt{\alpha_{dt}}s_{tn}}_{\text{direct link}} + \underbrace{\mathbf{f}_b e\Big(g_r\sqrt{\alpha_{bt}}s_{tn}\Big)}_{\text{backscatter link}} + \boldsymbol{\sigma}_n, \tag{6.12}$$

where $\boldsymbol{\sigma}_n = [\sigma_{1n}, \ldots, \sigma_{mn}, \ldots, \sigma_{Mn}]^{\mathsf{T}}$.

This work considers that there are I bits in each backscatter frame, i.e.,

$$\mathbf{x} = [x^{(1)}, \ldots, x^{(i)}, \ldots, x^{(I)}]. \tag{6.13}$$

To enhance the receiver's decoding process, F pilot bits are placed in each backscatter frame, so there are $I-F$ information bits in each backscatter frame. We consider that the AmB tag and the receiver know these bits in advance and use them to estimate the AmB tag-receiver channel coefficients. More details about the use of pilots will be discussed in Section 6.4. We assume that during one AmB frame, the AmB tag-receiver channel is invariant [82]. Since each backscatter bit is backscattered over N symbols transmitted by the transmitter, we can express the receiver's

received signals during the i-th AmB symbol duration by:

$$\mathbf{y}_n^{(i)} = \mathbf{f}_d\sqrt{\alpha_{dt}}s_{tn}^{(i)} + \mathbf{f}_b e^{(i)}\left(g_r\sqrt{\alpha_{bt}}s_{tn}^{(i)}\right) + \boldsymbol{\sigma}_n^{(i)}, \tag{6.14}$$

where the backscatter state $e^{(i)}$ equals backscatter bit $x^{(i)}$, i.e., $e^{(i)} = x^{(i)}$, $\forall i = 1, 2, \ldots, I$, and $n = 1, 2, \ldots, N$.

Note that for the AmB tag, the transmitter's signals appear as random signals and are unknown in advance. Therefore, the AmB rate's closed-form (denoted R_{AmB}) cannot be obtained [82, 157]. For that, in Theorem 6.1, we provide an alternative method to derive the AmB system's idealized throughput model when $N = 1$. Let θ_0 and θ_1 denote the probability of backscattering bit 0 and bit 1, respectively. $\mathcal{V}$ denotes a realization of $\mathbf{y}$, and $p(\mathcal{V}|e)$ denotes the conditional probability density function. Theorem 1 is as follows.

Theorem 6.1. *The maximum achievable rate of the AmB tag is given as*

$$R_{AmB}^{*} = Z(\theta_0) - \int_{\mathcal{V}} \big(\theta_1 p(\mathcal{V}|e=1) + \theta_0 p(\mathcal{V}|e=0)\big) Z(\mu_0) d\mathcal{V}, \tag{6.15}$$

where $\mu_0 = p(\mathcal{V}|e=0)$, and $Z(\cdot)$ represents the binary entropy function.

Proof. The proof of Theorem 6.1 is provided in Appendix B.1. □

It can be observed from (6.15) that the maximum achievable rate R_{AmB}^{*} depends on the probability of backscattering bit 0 θ_0. We will explore this dependency through our simulations in Section 6.6. Let $\mathbf{Y}^{(i)} = [\mathbf{y}_1^{(i)}, \ldots, \mathbf{y}_N^{(i)}]^{\mathsf{T}}$ denote the sequence of received signal corresponding to the period of the i-th AmB symbol. We describe our proposed AmB detectors that use MLK and DL to recover the original bits $x^{(i)}$ from these sequences of received signals in the following sections.

6.3 AmB Signal Detector based on Maximum Likelihood

Recall that signals backscattered by the AmB tag can be regarded as the active signals' background noise, making them very challenging to be detected [63, 80]. This section presents the AmB signal decoding based on MLK. Note the MLK-based detector can be considered as an optimal signal detector, so it can provide the system's upper-bound performance for comparison purpose.

6.3.1 Received Signals' Likelihood Functions

The aim of MLK-based detector is to derive the received signals' likelihood functions. In particular, if the AmB tag transmits bits "0" (corresponding to $e^{(i)} = 0$), the receiver only receives signals solely from the transmitter-to-receiver channel. Whereas, if the AmB tag transmits bits "1" (corresponding to $e^{(i)} = 1$), the receiver will receive signals from both direct and AmB channels. Consequently, the channel statistical covariance matrices for these scenarios are expressed as [82], [158]

$$\begin{aligned} \mathbf{R}_1 &= (\mathbf{k}_1 + \mathbf{k}_2)(\mathbf{k}_1 + \mathbf{k}_2)^{\mathrm{H}} + \mathbf{I}_M, \text{ if } e^{(i)} = 1, \\ \mathbf{R}_0 &= \mathbf{k}_1 \mathbf{k}_1^{\mathrm{H}} + \mathbf{I}_M, \text{ if } e^{(i)} = 0, \end{aligned} \tag{6.16}$$

where $\mathbf{k}_1 = \mathbf{f}_d\sqrt{\alpha_{dt}}$, $\mathbf{k}_2 = g_r\mathbf{f}_b\sqrt{\alpha_{bt}}$, $(\cdot)^{\mathrm{H}}$ is the conjugate transpose operator, and $\mathbf{I}_M$ is the identity matrix with size $M \times M$. It is important to note that the estimation of CSI can be performed by various techniques which are well studied in the literature [159, 160]. Similar to [82], the noise and the transmitter's RF signals are assumed to follow the CSCG distribution*. As a result, $\mathbf{y}_n^{(i)}$ also follows the CSCG [83], i.e.,

$$\mathbf{y}_n^{(i)} \sim \begin{cases} \mathcal{CN}(0, \mathbf{R}_0), & \text{if } e^{(i)} = 0, \\ \mathcal{CN}(0, \mathbf{R}_1), & \text{if } e^{(i)} = 1. \end{cases} \tag{6.17}$$

*Note that this information is not required by our proposed DL-based AmB signal detector.

Now, the conditional probability density function (PDFs) of the received signals $\mathbf{y}_n^{(i)}$ given the backscatter state $e^{(i)}$ is computed by

$$\begin{aligned} p(\mathbf{y}_n^{(i)}|e^{(i)}=0) &= \frac{1}{\pi^M|\mathbf{R}_0|}e^{-\mathbf{y}_n^{(i)\mathrm{H}}\mathbf{R}_0^{-1}\mathbf{y}_n^{(i)}}, \\ p(\mathbf{y}_n^{(i)}|e^{(i)}=1) &= \frac{1}{\pi^M|\mathbf{R}_1|}e^{-\mathbf{y}_n^{(i)\mathrm{H}}\mathbf{R}_1^{-1}\mathbf{y}_n^{(i)}}, \end{aligned} \tag{6.18}$$

where $|\cdot|$ and $(\cdot)^{-1}$ represent the determinant and inverse of a matrix, respectively. From (6.18), we now can express the likelihood functions $\mathcal{H}(\cdot)$ for the sequence of received signals as follows [82, 88]:

$$\begin{aligned} \mathcal{H}(\mathbf{Y}^{(i)}|e^{(i)}=1) &= \prod_{n=1}^{N}\frac{1}{\pi^M|\mathbf{R}_1|}e^{-\mathbf{y}_n^{(i)\mathrm{H}}\mathbf{R}_1^{-1}\mathbf{y}_n^{(i)}}, \\ \mathcal{H}(\mathbf{Y}^{(i)}|e^{(i)}=0) &= \prod_{n=1}^{N}\frac{1}{\pi^M|\mathbf{R}_0|}e^{-\mathbf{y}_n^{(i)\mathrm{H}}\mathbf{R}_0^{-1}\mathbf{y}_n^{(i)}}. \end{aligned} \tag{6.19}$$

6.3.2 Maximum Likelihood Detector

Let $\tilde{e}^{(i)}$ be the estimation of $e^{(i)}$. From (6.19), the MLK hypothesis in (6.20) can be used to obtain the backscattered state $e^{(i)}$ [82], [158].

$$\tilde{e}^{(i)} = \begin{cases} 1, & \mathcal{H}(\mathbf{Y}^{(i)}|e^{(i)}=0) < \mathcal{H}(\mathbf{Y}^{(i)}|e^{(i)}=1), \\ 0, & \mathcal{H}(\mathbf{Y}^{(i)}|e^{(i)}=0) > \mathcal{H}(\mathbf{Y}^{(i)}|e^{(i)}=1). \end{cases} \tag{6.20}$$

Based on (6.19) and (6.20), the MLK hypothesis can be given by

$$\tilde{e}^{(i)} = \begin{cases} 1, \prod_{n=1}^{N}\frac{1}{\pi^M|\mathbf{R}_0|}e^{-\mathbf{y}_n^{(i)\mathrm{H}}\mathbf{R}_0^{-1}\mathbf{y}_n^{(i)}} < \prod_{n=1}^{N}\frac{1}{\pi^M|\mathbf{R}_1|}e^{-\mathbf{y}_n^{(i)\mathrm{H}}\mathbf{R}_1^{-1}\mathbf{y}_n^{(i)}}, \\ 0, \prod_{n=1}^{N}\frac{1}{\pi^M|\mathbf{R}_0|}e^{-\mathbf{y}_n^{(i)\mathrm{H}}\mathbf{R}_0^{-1}\mathbf{y}_n^{(i)}} > \prod_{n=1}^{N}\frac{1}{\pi^M|\mathbf{R}_1|}e^{-\mathbf{y}_n^{(i)\mathrm{H}}\mathbf{R}_1^{-1}\mathbf{y}_n^{(i)}}. \end{cases} \tag{6.21}$$

By applying logarithmic operations to (6.21), we can derive the following expression

$$\tilde{e}^{(i)} = \begin{cases} 1, & \sum_{n=1}^{N} \mathbf{y}_n^{(i)\mathrm{H}} (\mathbf{R}_0^{-1} - \mathbf{R}_1^{-1}) \mathbf{y}_n^{(i)} > N \ln \frac{|\mathbf{R}_1|}{|\mathbf{R}_0|}, \\ 0, & \sum_{n=1}^{N} \mathbf{y}_n^{(i)\mathrm{H}} (\mathbf{R}_0^{-1} - \mathbf{R}_1^{-1}) \mathbf{y}_n^{(i)} < N \ln \frac{|\mathbf{R}_1|}{|\mathbf{R}_0|}. \end{cases} \tag{6.22}$$

Then, the backscattered bit $x^{(i)}$ can be derived based on $\tilde{e}^{(i)}$. However, for that, MLK requires perfect CSI for both conventional and AmB transmissions, making its less efficient in practice, especially for AmB communications where backscattered signals are very weak [82].

6.4 Deep Learning-Based Signal Detector

As discussed in Section 6.3, the traditional approaches for backscattered signal detection (i.e., MLK-based detectors) require accurate channel statistical covariance matrices. However, the uncertainty of the wireless environment makes it difficult for the system to estimate these matrices accurately. To that end, we develop a DL-based signal detector that can overcome this challenge. We first discuss the data preprocessing procedure, aiming to construct a dataset for the training phase of the DL detector, and then our proposed DNN architecture is presented in detail.

6.4.1 Data Preprocessing

In practice, the raw data (e.g., received signals) may not be qualified to be used directly to train a DL model as it can lead to a poor or even useless trained model. Therefore, data preprocessing is an essential step before training. In this work, the data preprocessing steps are as follows. Recall that $\mathbf{y}_n^{(i)}$ denotes the received AmB signals, and $\mathbf{Y}^{(i)}$ is the sequence of $\mathbf{y}_n^{(i)}$ corresponding to the i-th AmB symbol period. Instead of directly using $\mathbf{Y}^{(i)}$ for model training, the sample covariance matrix $\mathbf{S}^{(i)}$ of the received signals, defined by (6.23), can be used as the training

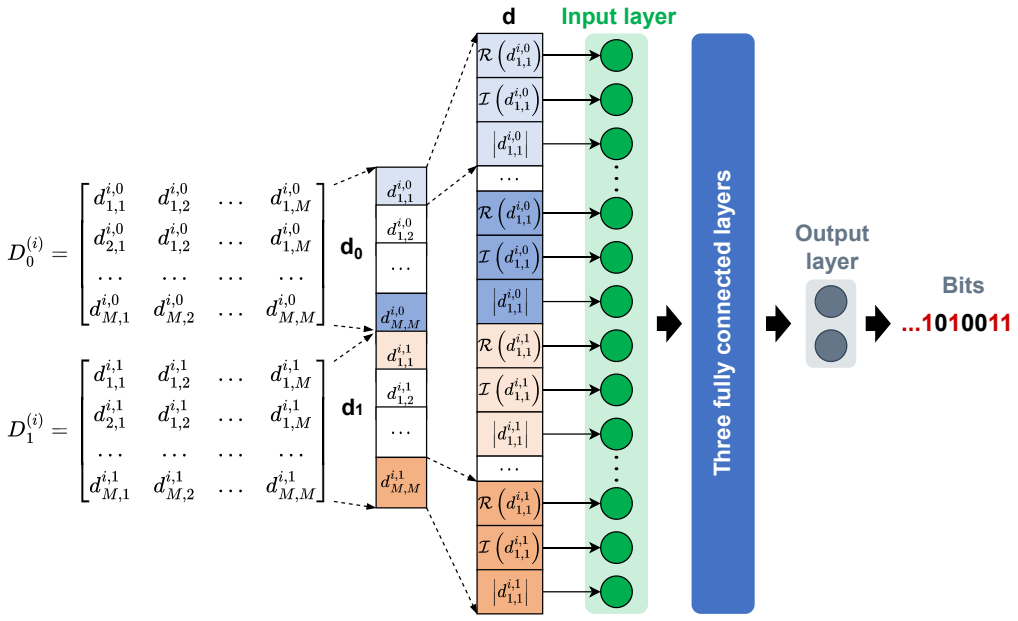


Figure 6.3 : Our proposed AmB signal detector based on DL.

data for DL models.

$$\mathbf{S}^{(i)} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{y}_n^{(i)}\mathbf{y}_n^{(i)^{\mathrm{H}}}. \tag{6.23}$$

The main reason is that the sample covariance matrix can capture the relationship between received signals corresponding to an AmB symbol period, thus providing more insights on received signals than the signals themselves. In addition, the performance of a signal detector is proportional to the number of RF symbols corresponding to an AmB symbol, i.e., N. In other words, the higher value N is, the better performance the detector can achieve. Typically, N is often set to a higher value than the number of antennas [82, 83, 88], and thus the size of $\mathbf{S}^{(i)}$ is smaller than that of $\mathbf{Y}^{(i)}$. Therefore, training the DNN model with the sample covariance matrix $\mathbf{S}^{(i)}$ can not only preserve all crucial information of the received signals but also reduce the training data size without affecting the learning efficiency [82, 88]. Note that conventional MLK-based detectors explicitly require accurate estimation of channel coefficients (i.e., $\mathbf{k_1}$ and $\mathbf{k_2}$), resulting in high computational complexity, especially for a system with multiple antennas [161]. To alleviate this challenge, pilot

bits are leveraged to indirectly capture the channel coefficients, thus improving the accuracy of the covariance matrix estimation. Specifically, we use a sequence of F pilot bits where the first $F/2$ bits are "0", and the rest are "1". The detector knows these bits in advance so that the channel coefficients can be implicitly obtained by inspecting the received signals corresponding to them. In this work, the averages of $\mathbf{S}^{(i)}$ of received signals corresponding to pilot bits "1" and "0" are leveraged to indirectly capture the channel coefficients, which are given as follows:

$$\begin{aligned} \bar{\mathbf{R}}_1 &= \frac{2}{FN}\sum_{i=1}^{F/2}\sum_{n=1}^{N}\mathbf{y}_n^{(i)}\mathbf{y}_n^{(i)\mathrm{H}}, \quad \text{when } e^{(i)}=1, \\ \bar{\mathbf{R}}_0 &= \frac{2}{FN}\sum_{i=1}^{F/2}\sum_{n=1}^{N}\mathbf{y}_n^{(i)}\mathbf{y}_n^{(i)\mathrm{H}}, \quad \text{when } e^{(i)}=0. \end{aligned} \tag{6.24}$$

In this way, the detector can use $\bar{\mathbf{R}}_0$ and $\bar{\mathbf{R}}_1$ to effectively detect AmB signals without requiring channel coefficients explicitly. In particular, if the AmB tag backscatters bit "0", the sample covariance matrix $\mathbf{S}^{(i)}$ would be similar to $\bar{\mathbf{R}}_0$ but different from $\bar{\mathbf{R}}_1$. In contrast, if bit "1" is backscattered, the sample covariance matrix $\mathbf{S}^{(i)}$ would be similar to $\bar{\mathbf{R}}_1$ but different from $\bar{\mathbf{R}}_0$. It is worth noting that these similarities and differences between $\mathbf{S}^{(i)}$ and the averaged covariance matrices (i.e., $\bar{\mathbf{R}}_0$ and $\bar{\mathbf{R}}_1$) can be better represented by multiplying $\mathbf{S}^{(i)}$ with the inverse of these matrices. As such, the data for training our DL model is given as follows:

$$\begin{aligned} \mathbf{D}_1^{(i)} &= \mathbf{S}^{(i)}\bar{\mathbf{R}}_1^{-1}, \\ \mathbf{D}_0^{(i)} &= \mathbf{S}^{(i)}\bar{\mathbf{R}}_0^{-1}. \end{aligned} \tag{6.25}$$

By doing so, the proposed AmB signal detector based on DL can obtain adequate information about the channel coefficients via $\mathbf{S}^{(i)}$, $\bar{\mathbf{R}}_0$, and $\bar{\mathbf{R}}_1$ to learn and detect backscattered bits.

To create input data for the DL model, the matrices $\bar{\mathbf{R}}_0$ and $\bar{\mathbf{R}}_1$ are processed

as follows. Recall that the receiver has M antennas, $\mathbf{D}_0^{(i)}$ and $\mathbf{D}_1^{(i)}$ are $M \times M$ matrices. Let d_{ij}^0 and d_{ij}^1 be the elements of $\mathbf{D}_0^{(i)}$ and $\mathbf{D}_1^{(i)}$, respectively, with $i, j \in \{1, 2, \ldots, M\}$. First, the matrices $\bar{\mathbf{R}}_0$ and $\bar{\mathbf{R}}_1$ are flattened into two vectors, i.e., $\mathbf{d}_0$ and $\mathbf{d}_1$, each has $M \times M$ elements, as shown in Figure 6.3. Then, $\mathbf{d}_1$ is concatenated at the end of $\mathbf{d}_0$ to form vector $\mathbf{d}$ with $2M^2$ elements. Finally, the input data for the DL model is constructed by taking (i) the real part, (ii) the imaging part, and (iii) the absolute of each element of the combined vector $\mathbf{d}$, as illustrated in Figure 6.3. Thus, the input data can allow the DL model to effectively leverage all important information of the received signals, thereby improving learning quality.

6.4.2 Deep Neural Network Architecture

This work designs a DNN at the signal detector to decide whether the signals received by the receiver correspond to bit "1" or bit "0" of an AmB frame. Aiming to develop a lightweight anti-eavesdropping solution, we only consider a small footprint DNN with a simple and typical architecture. In particular, our DNN consists of an input layer, a small number (e.g., three) of fully connected (FC) hidden layers, and an output layer, as illustrated in Figure 6.3. Since the input vector data's size is $M \times M \times 2 \times 3$, the input layer consists of $M \times M \times 2 \times 3$ neurons. After the input layer receives the input data, it will be sequentially forwarded to FC layers, and each consists of multiple neurons. In FC layers, each neuron connects to all neurons in the previous layer. For example, each neuron in the first FC layer connects to all neurons in the input layer. The output of the last FC layer is applied to the soft-max function at the output layer to obtain the category probabilities p of which the received signals corresponds to bit "0" and bit "1". Finally, the output layer uses these probabilities to determine the input's class, i.e., "0" or "1". In DNNs, a neuron applies a mathematical function, such as a sigmoid, tanh, or rectified linear unit (ReLU), to the sum of its input and bias to get its output. Here, the

connections between neurons are represented by weight values. The input to a neuron is calculated as the weighted sum of the outputs of neurons in the previous layer that are connected to it. The training process aims to optimize the DNN parameters θ (i.e., weights and bias) to minimize the loss $\mathcal{L}(f(\mathbf{d};\theta),\psi)$, which is the error between the DNN output $f(\mathbf{d};\theta)$ and the ground truth ψ, as follows:

$$\min_{\theta} \quad \mathbb{E}\big[\mathcal{L}(f(\mathbf{d};\theta),\psi)\big], \tag{6.26}$$

where the cross-entropy loss is used at the output layer. To minimize the loss $\mathcal{L}(f(\mathbf{d};\theta),\psi)$, this thesis proposes to use SGD (discussed in Section 2.1) to optimize the DNN's parameters. Thanks to this design, the proposed DL based-receiver can accurately predict transmitted bits in a backscatter frame even for weak AmB signals

Although DL allows receivers to effectively detect AmB signals without requiring accurate CSI, it faces several shortcomings. First, the training process of DL often requires a huge amount of received signals for the training process to achieve good performance [89]. Thus, this makes DL less efficient in practice, especially when data is expensive and contains noise due to the environment's dynamic and uncertainty, as in the considered wireless environment. Second, conventional DL may face the over-fitting problem, i.e., performing excellently on training datasets but very poorly on test datasets, if the training dataset does not contain enough samples to represent all possible scenarios. Due to the dynamic nature of the wireless environment, the channel conditions may vary significantly over time. For example, a moving bus may change channel conditions from LoS to NLoS and vice versa. Consequently, real-time data may greatly differ from training data, making these above problems more severe. Third, wireless channel conditions can also be very different at different areas due to their landscapes, so the DL model trained at one area may not perform well at other areas. As such, different sites may need training different models

from scratch, which is time-consuming and costly [90]. For that, the next section will discuss our approach based on meta-learning that can effectively alleviate these challenges.

6.5 AmB Signal Detector based on Meta-Learning

Recently, meta-learning is emerging as a promising approach to address the shortcomings of DL discussed in the previous section. The reason is that meta-learning has an ability of learning to learn, i.e., self-improving the learning algorithm [89–91, 162]. The main idea of meta-learning is to train the model on a collection of tasks, enabling it to acquire generalization capabilities. As a result, the trained model can quickly perform well in a new task only after a few update iterations, even when provided with a small dataset specific to the new task [91]. The meta-learning algorithms can be classified into two groups [89]. The first group aims to encode knowledge in special and complex DNN architectures, such as Convolutional Siamese Neural Network [163], Matching Networks [164], Prototypical networks [165], and Memory-Augmented Neural Networks [162]. As such, this group requires more overhead for operations [91], and thus it may not be suitable for our lightweight framework. In the second group, learning algorithms aim to learn the model's initialization. Thus, this approach does not require any additional elements or a particular architecture, making it well fit in our framework. Therefore, the model initialization is proposed to use in our framework.

A classical approach of the second group (i.e., learning the model's initialization) is that the model is first trained with a large dataset from existing tasks and then fine-tuned with the new task's data. However, this pre-training method could not guarantee that the learned initialization is good for fine-tuning, and many other techniques need to be leveraged to achieve good performance [89]. Recently, the work in [91] proposes Model-Agnostic Meta-Learning (MAML) that directly optimizes

Algorithm 6.1 The Meta-learning based AmB Signal Detection

1: Initialize model f with parameters θ.
2: **for** *episode = 1, 2, ...* **do**
3: Sample task τ uniformly, and obtain the corresponding training dataset $\mathcal{D}_\tau$
4: **for** step = 1 to p **do**
5: Based on dataset $\mathcal{D}_\tau$, perform a step of stochastic optimization (SGD or Adam), starting with parameters θ, resulting with parameters $\bar{\theta} = U_\tau^p(\theta)$.
6: **end for**
7: Update $\theta \leftarrow \theta + \eta_o(\bar{\theta} - \theta)$
8: **end for**

model's parameters during the learning process with a collection of similar tasks, thus guaranteeing a good initialization of the model. However, MAML needs to perform second-order derivatives through the optimization process, hence requires high complexity. To overcome this problem, the Reptile algorithm is proposed in [89]. This algorithm only uses a conventional DL algorithm, such as stochastic gradient descent (SGD) and Adam, making it less computationally demanding while still maintaining a similar performance level of MAML [89]. For that reason, this work adopts the Reptile algorithm. The detail of the meta-learning-based AmB signal detection is provided in Algorithm 6.1. In particular, a model f (initiated with parameters θ) is trained with a set $\mathcal{T}$ of similar tasks [89, 91]. As an example, let us consider a set of image classification tasks. The first task involves classifying images into different animal categories, specifically the tiger, dog, and cat classes. On the other hand, the second task focuses on classifying images into another set of animal categories, namely the elephant, bear, and lion classes. Similarly, this work considers a set of AmB signal detection tasks, each under a particular channel model, e.g., Rayleigh, WINNER II, and Rician.

Generally, meta-learning comprises two nested loops. In the inner loop, a base-learning algorithm (e.g., SGD or Adam) solves task τ, e.g., detecting the AmB signal under the Rician channel. The objective of the inner loop's learning is to

minimize the loss $\mathcal{L}_\tau$. Note that a base-learning algorithm can be a typical DL algorithm, such as stochastic gradient descent (SGD). Thus, the inner loop is similar to the training process of the proposed DL-based AmB signal detector discussed in Section 6.4.2. After p steps of learning, the resulting parameters is denoted by $\bar{\theta} = U_\tau^p(\theta)$, where $U_\tau^p(\cdot)$ denotes the update operator. Then, at the outer loop, a meta-learning algorithm (e.g., Reptile) updates the model parameters as follows:

$$\theta \leftarrow \theta + \eta_o(\bar{\theta} - \theta), \tag{6.27}$$

where η_o is the outer step size controlling how much the model parameters are updated. By doing so, the generalization in learning is improved. Note that if $p = 1$, i.e., performing a single step of gradient descent in the inner loop, Algorithm 6.1 becomes a joint training on the mixture of all tasks, which may not learn a good initialization for meta-learning [89].

Our proposed meta-learning algorithm comprises two nested loops. The inner loop is similar to stochastic gradient descent (SGD), a traditional deep learning algorithm, and thus, shares the same computational complexity. Therefore, assuming that the outer loop consists of K iterations, the computational complexity of our meta-learning approach would be K times that of SGD. However, our simulation results demonstrate that meta-learning requires a significantly smaller dataset (1/200 the size) to achieve a comparable performance level to that of a traditional deep-learning algorithm in a new environment. In the next section, our proposed framework will be evaluated in various scenarios to get insights into its performance.

6.6 Simulation Results

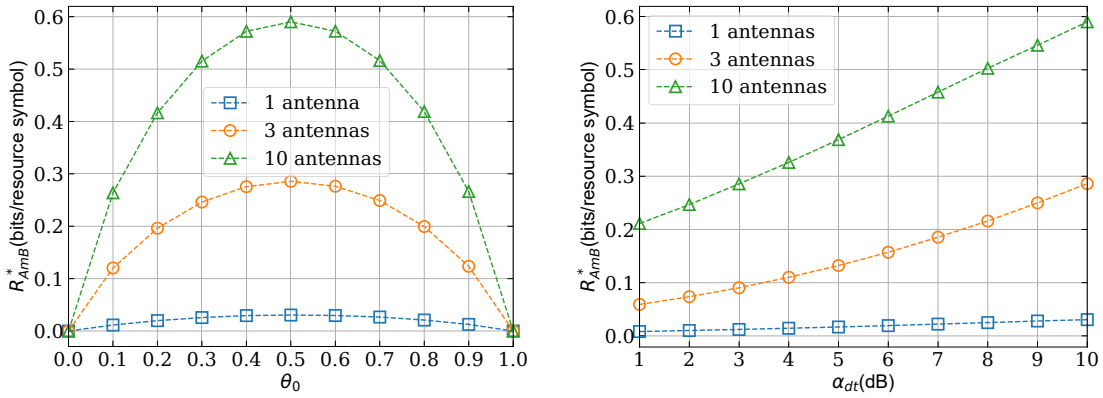
6.6.1 Simulation Settings

To evaluate our proposed solution, we conduct simulations in various scenarios under the following settings unless otherwise stated. For the transmission aspect, the AmB rate is 50 times lower than the transmitter-to-receiver link rate, i.e., $N = 50$, and the AmB message consists of $I = 100$ bits. Since the transmitter-receiver link SNR α_{dt} significantly depends on many factors, such as transmit power, antenna gain, and path loss, we vary it from 1 dB to 9 dB in the simulations to examine our proposed solution in different scenarios. The tag-receiver link SNR α_{bt} is often quite weak, and thus it is set to -10 dB [158].

In this work, the system is simulated using Python, and the DL model is built using the Pytorch library. Specifically, the Rayleigh fading follows the zero-mean and unit-variance CSCG distribution, as suggested in [158]. Note that in our system, the AmB tag does not know the transmitter's signals (i.e., RF resource signals) sending from the transmitter to the receiver. In addition, this work focuses on signal detection for the AmB link. As such, the RF resource signals can be assumed to follow the zero-mean and unit-variance CSCG distribution, similar to those in [82,83, 158]. For evaluating different aspects of our proposed solution, we use three metrics, i.e., maximum achievable AmB rate and BER for the transmission performance and guessing entropy for the security analysis.

6.6.2 Maximum Achievable Backscatter Rate

To get insights on the maximum achievable AmB rate R^*_{AmB} of the proposed system, we perform simulations in three settings, where the receiver has $1, 3$ and 10 antennas, as shown in Figure 6.4. First, in Figure 6.4(a), the prior probability of backscattering bit "0", i.e., θ_0, is varied to observe the maximum achievable AmB



(a) (b)

Figure 6.4 : The maximum achievable AmB rate when varying (a) θ_0 and (b) the transmitter-to-receiver link SNR, i.e., α_{dt}.

rate, i.e., the idealized throughput R^*_{AmB} which is given by Theorem 6.1. The results are obtained by 10^6 Monte Carlo runs. Figure 6.4(a) shows that the maximum achievable AmB rate R^*_{AmB} increases when the receiver has more antennas. This is because the receiver can achieve a higher gain with more antennas. As such, the received backscatter signals are enhanced, thereby reducing the impacts from fading and interference originating from the direct link and the surrounding environment. Importantly, R^*_{AmB} is maximized at $\theta_0 = 0.5$. Therefore, we set $\theta_0 = 0.5$ in the rest of simulations. Next, we vary the transmitter-to-receiver link SNR, i.e., α_{dt}, from 1 dB to 9 dB, as shown in Figure 6.4(b). It can be observed that as the active link SNR α_{dt} increases, the maximum achievable backscatter rate R^*_{AmB} increases. The reason is that when the transmitter increases its transmit power, the signal arriving at the AmB tag is stronger, making the backscattered signals stronger.

6.6.3 Anti-eavesdropper and Security Analysis

It is worth noting that our proposed framework can counter eavesdropping attacks in two folds. Firstly, a part of the original message is hidden in the AmB channel, appearing as background noise for conventional signal detectors. Secondly, the proposed message encoding mechanism makes reconstructing the original mes-

sage non-trivial unless the eavesdropper knows this and at the same time it can decode information from both active and AmB channels.

In particular, without knowledge about the system in advance (i.e., the settings of AmB transmission), the eavesdropper is not even aware of the existence of the AmB message, thus guaranteeing the confidentiality of the original message. It is worth emphasizing that even if the eavesdropper is aware of the AmB message, it is still challenging to capture the AmB message. Suppose that the eavesdropper leverages the AmB signal detector circuit proposed in [70], which requires different values of resistors and capacitors in the circuit for different backscatter rates. As such, if the eavesdropper deploys the AmB circuit but does not know the exact backscatter rate, it still cannot decode the backscatter signals [70]. Iteratively testing every possible rate is impractical. It can be considered that the eavesdropper can use MLK- or DL-based detectors for detecting AmB signals. However, MLK-based approaches require complete information on signal distribution and perfect CSI, while DL-based methods need a large amount of data and time to detect AmB signals properly. Given the above, the probability that the eavesdropper can successfully capture the AmB message is marginal in practice.

To further evaluate the security of our proposed system, we consider the worst-case in which the eavesdropper knows the exact backscatter rate in advance, and thus they can capture the AmB message as well as the active message. However, since the message encoding technique is unknown, the eavesdropper still does not know how to construct the original message based on the active and AmB messages. They only know that combining these two messages can decode the original message. As such, they must determine the position of all I bits of the AmB message in the original message. To quantify the security of our proposed anti-eavesdropping solution in this case, this study considers the guessing entropy metric [81].

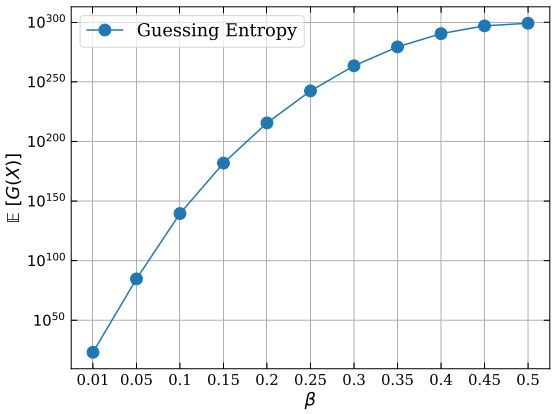


Figure 6.5 : The upper bound of the expected number of guesses, i.e., $\mathbb{E}[G(X)]$, vs. the message splitting ratio β.

Suppose that the original message has P bits, the probability that the eavesdropper successfully finds the positions of I bits in the original message by guessing is given by

$$p_I = \frac{1}{C_I^P} = \frac{I!(P-I)!}{P!}. \tag{6.28}$$

To correctly identify the position of I bits, the eavesdropper needs to ask a number of questions of the kind "does this correct?". Suppose that the eavesdropper follows an optimal guessing sequence, the average number of these questions defines the guessing entropy [81]. We can consider that the position of I bits is analogous to the key k_I for reconstructing the original message. Let X denote the random variable corresponding to the selection of key k_I in the key set $\mathcal{K}$, and X is determined by probability distribution $\mathcal{P}_k$. Note that in this work, the key set consists of all possible keys, and thus the size of $\mathcal{K}$ is C_I^P. One of the optimal brute-force attacks is the scheme where the eavesdropper knows $\mathcal{P}_k$, and it sorts the key set $\mathcal{K}$ in the descending order of key selection probability p_k to form a guessing sequence $\bar{\mathcal{K}}$ [81]. Then, it iteratively tries a key in the sorted key set $\bar{\mathcal{K}}$. For that, the guessing entropy is defined as follows:

$$G(X) = \sum_{1 \le i \le |\bar{\mathcal{K}}|} i p(x_i), \tag{6.29}$$

where i in the index in $\bar{\mathcal{K}}$, and $p(x_i)$ corresponds to the selection probability of a key at index i. In [81], the upper bound of the expected number of guesses, i.e., $\mathbb{E}[G]$, for the eavesdropper with the optimal guessing sequence is given as follows:

$$\mathbb{E}[G(X)] \leq \frac{1}{2}\left[\sum_{i=1}^{|\bar{\mathcal{K}}|}\sqrt{p(x_i)}\right]^2 + \frac{1}{2}. \tag{6.30}$$

In Figure 6.5, we vary the message splitting ratio $\beta = P/I$ to observe the guessing entropy of our framework. Here, $\beta = 0.1$ means that 10% of original message bits are transmitted by the AmB tag, and the remaining bits are transmitted by the transmitter. Since the AmB rate is significantly lower than the transmitter rate [82, 158], the message splitting ratio β is less than 0.5 in practice. As shown in Figure 6.5, when the ratio β increases from 0.01 to 0.5, the guessing entropy also increases. It is stemmed from the fact that as the size of the AmB message increases, the probability of guessing successfully in one trial decreases, as implied by (6.28) when $\beta < 0.5$. It is worth mentioning that guessing entropy is the average number of questions asked by the eavesdropper to successfully construct the original message. As such, the greater the value of $\mathbb{E}[G(X)]$, the higher the security level is.

6.6.4 The Learning Process of DL-based AmB Signal Detector

The settings for the DL-based approaches are as follows. Note that since the architecture of DNN can significantly affect the performance of the DL-based AmB signal detector, it must be designed thoughtfully. For example, a DNN with more layers may perform better but demands more resources and takes longer time for training. Thus, this work designs a simple and lightweight DNN while still achieving good performance of detecting AmB signal. Particularly, our DNN has an input layer, four FC layers, and an out layer. The number of neurons in the input layer depends on the dimension of the input data $\mathbf{d}$, which is $M \times M \times 2 \times 3$ where M is

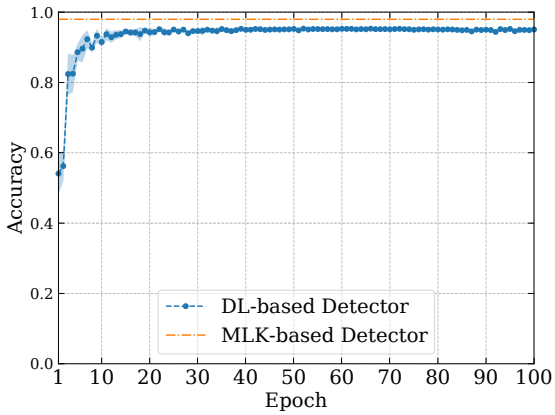


Figure 6.6 : The convergence of DL-based AmB signal detector when $\alpha_{dt} = 1$ dB and the receiver has 10 antennas.

the number of the receiver's antennas. The first, second, and third FC layers have $600, 1000$, and 600 neurons, respectively. The training dataset is obtained from 10^4 Monte Carlo runs.

Figure 6.6 shows the convergence rate of our proposed DL-based AmB signal detector when the direct link SNR $\alpha_{dt} = 1$ dB and the receiver has 10 antennas. To train the DNN, we use the SGD with the learning rate of 0.001 [95]. The batch size is set to $1,000$ data points, and thus each training epoch consists of 10 learning iterations. In Figure 6.6, the accuracy of the MLK-based detector is also presented as a baseline. It can be observed that after 30 learning epochs, our proposed DL-based detector converges into a reliable model that can achieve an accuracy close to that of the MLK-base detector. Note that MLK-based detectors are considered the optimal signal detection scheme, but it is impractical due to high computing complexity and the requirement of perfect CSI [166].

6.6.5 BER Performance

Now, we investigate the system's BER performance when varying the direct link (i.e., transmitter-to-receiver) SNR α_{dt} from 1 dB to 9 dB. The training process of our proposed DL-based detector is conducted in the same way described in 6.6.4

with two settings, i.e., providing (i) the estimated CSI based on pilot bits, namely DL-eCSI, and (ii) the perfect CSI, namely DL-pCSI. We perform simulations with three antenna configurations for the receiver, i.e., 1, 3, and 10 antennas. To obtain reliable results, both the MLK-based and the DL-based detectors are evaluated by performing Monte Carlo fashion with 10^6 runs. Note that this work only focuses on the AmB link (i.e., the tag-receiver link), and thus we only obtain the BER of the AmB link since the BER of the direct (transmitter-receiver) link can be close to zero with advanced modulation and channel coding techniques.

Figure 6.7(a) shows that when the direct link SNR, i.e., α_{dt}, increases from 1 dB to 9 dB, the BERs of all approaches decrease. The rationale behind is that the stronger the transmitter's active signals are, the stronger the signals backscattered by the AmB tag (i.e., AmB signals) are, so the lower the system BER is. It is also observed that the system performance improves (i.e., a decrease in BER) as the number of receiving antennas increases. This is because the received signals at the receiver are enhanced by antenna gain, which is typically proportional to the number of antennas. Similar to the observation in Section 6.6.4, the BERs of our DL-based detector are close to those of the MLK-detector, an optimal signal detector, in all cases with 1, 3, and 10 antennas. Thus, these clearly show the effectiveness of our proposed DL-based approach.

Next, we vary the tag-receiver SNR, i.e., α_{bt}, from -25 dB to -5 dB, while the transmitter-receiver SNR α_{dt} is 7 dB, as shown in Figure 6.7(b). Similar to observations in Figure 6.7(a), when α_{bt} and the number of antennas increase, the BERs of all approaches reduce. The rationale behind this is that when the received signal is stronger and the antenna gain is higher, the detector can attain a lower BER, indicating better performance. Again, the gap observed in BERs between our proposed DL-based detector and the MLK method is marginal. Interestingly, the DL-eCSI can achieve comparable BER results compared with that of DL-pCSI,

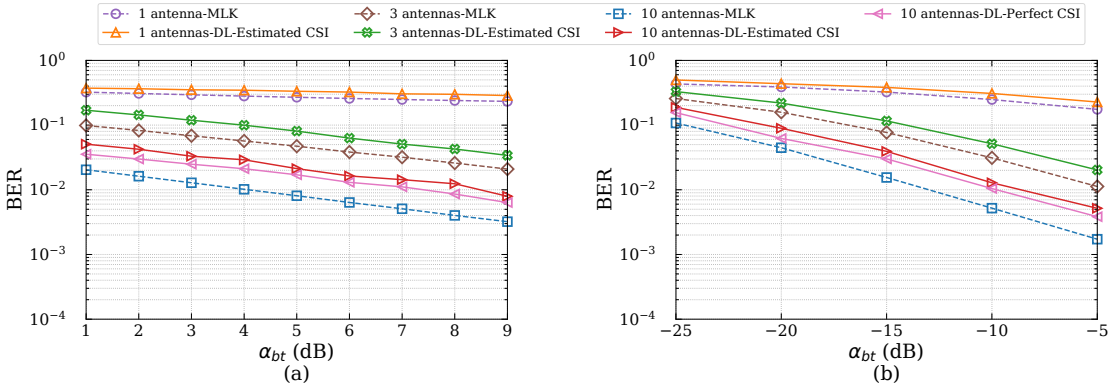


Figure 6.7 : Varying (a) the transmitter-receiver SNR α_{dt} and (b) the tag-receiver SNR α_{bt}.

as shown in Figures 6.7(a) and (b). Thus, these results show that our proposed DL-based detector can perform well in practice with only estimated CSI.

6.6.6 Meta-Learning for AmB Signal Detection

Now, we evaluate the proposed meta-learning approach for the AmB-signal detector with the following setups. We consider that the target task is detecting AmB signals under Rayleigh fading to better demonstrate the comparison between meta-learning-, DL-, and MLK- based approaches. The set of tasks for the learning process, i.e., training tasks, consists of (i) detecting AmB signals under Rician fading and (ii) detecting AmB signals under WINNER II fading [167]. Algorithm 6.1 is used to train the DNN model, i.e., meta-model, with the training tasks' datasets. Here, each training task's dataset contains 500 data points, and the outer step size η is set to 0.1, similar to [89]. Then, the trained meta-model is trained with the dataset collected from the target task (i.e., detecting AmB signals under Rayleigh fading) by using SGD. For these simulations, we select two types of baselines: (i) DL- and (ii) MLK-based approaches. Three DL models are trained with different numbers of data points, including 50, 10^3, and 10^4, in a similar procedure in Section 6.6.4. In all approaches, there are 10 antennas in the receiver.

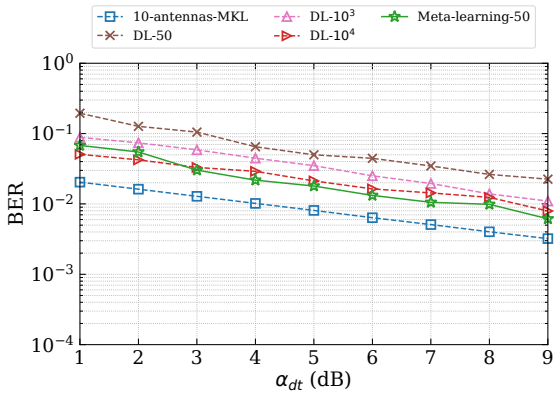


Figure 6.8 : The BER performance of different learning approaches.

First, we compare the BER performance between our proposed meta-learning-based approach and other baselines when varying the transmitter-receiver SNR α_{dt}, as shown in Figure 6.8. Similar to the observations in Figures 6.7(a) and (b), as the signal's strength increases (i.e., an increase of SNR), all approaches achieve a lower BER, indicating better performance. It is observed that a DL model trained with more data points results in a better performance. In particular, among DL-based approaches, DL with 10^4 data points achieves the lowest BER, and DL with 50 data points gets the highest BER. The reason is that more data points can gain more knowledge to the DL model. Thus, it can reduce the over-fitting issue, i.e., performing well on a training dataset but poor on a test dataset, thereby improving the system performance [95]. Interestingly, when the transmitter-receiver SNR $\alpha_{dt} \geq 3$ dB, meta-learning with only 50 data points can achieve the lowest BER compared with DL-based approaches that require up to 10^4 data points. Whereas, when α_{dt} is low (less than 3 dB), meta-learning's BER is slightly higher than that of the DL with 10^4 data points but still lower than those of DL with 10^3 and 50 data points. These observations are stemmed from the fact that meta-learning has a generalization ability that can help a DNN model to learn a new task quickly with few data points [89].

To further investigate the reliability of these learning approaches, we run each

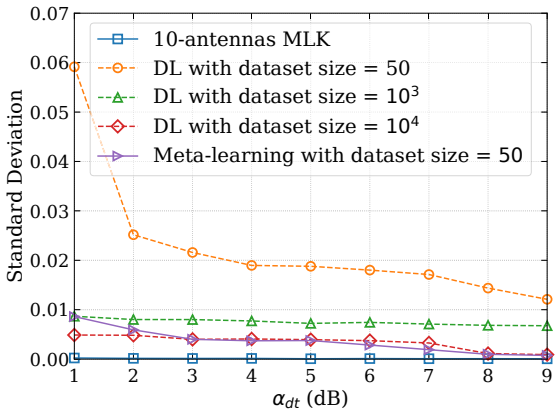


Figure 6.9 : Reliability of learning process with different training datasets' sizes.

learning approach 20 times to obtain the standard deviation of its BER results, as shown in Figure 6.9. The simulation settings are set as those in Figure 6.8. Generally, the standard deviations of all approaches decrease as the transmitter-receiver SNR α_{dt} increases from 1 dB to 10 dB. This is because the received signals contain more noise at a lower SNR, making the learning more challenging and leading to high deviation of results, i.e., more unstable results. In particular, at $\alpha_{dt} = 1$, the standard deviation of DL with 50 training data points is about 10 times higher compared with that of the best learning-based approach i.e., DL with 10^4 data points. Again, when $\alpha_{dt} \geq 3$ dB, the proposed meta-learning-based approach achieves comparable results as those of DL with 10^4 data points. Notably, when $\alpha_{dt} \geq 8$ dB, the results of meta-learning and DL with 10^4 data points close to the results of MLK. When the α_{dt} decreases to less than 3 dB, the meta-learning still obtains very good results, one of the best approaches in terms of reliable performance.

Finally, to get insight into how the amount of training data can impact meta-learning, we set the size of each dataset collected from a training task to D_t and then vary D_t from 100 to 500 data points. Here, the transmitter-receiver SNR α_{dt} is set to 5 dB. Figure 6.10 shows the BER performance of models trained by meta-learning with 50, 100, 200, 400, and 10^3 data points from the target task, namely

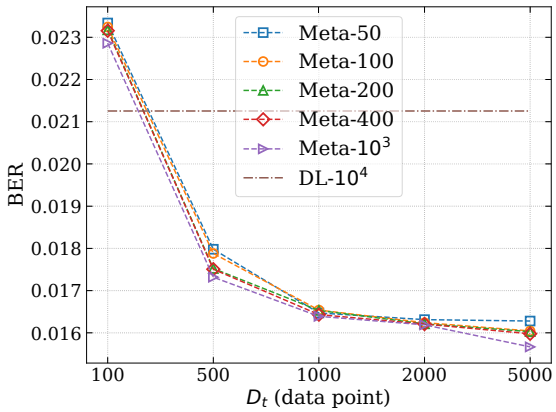


Figure 6.10 : Varying the size of training tasks' datasets.

Meta-50, Meta-100, Meta-400, and Meta-10^3. For comparing with the DL-based approach, this figure also presents the result of the model trained from scratch by SGD (as in Section 6.6.4) at $\alpha_{dt} = 5$ with 10^4 data points of the target task, namely DL-10^4. Similar to observations in Figure 6.8, all approaches achieve lower BERs as the training data size increases, as shown in Figure 6.10. Interestingly, when the meta-learning training data points are adequate (e.g., $D_t \leq 500$), even with only 50 target task's data points, meta-learning approaches outperform the DL approach training with 10^4 target task's data points. Given the above, it clearly shows the superiority of meta-learning to DL approaches in terms of data efficiency, making it more applicable in practical systems.

6.7 Conclusion

This chapter has introduced a novel anti-eavesdropping framework that can enable secure wireless communications leveraging the low-cost and low-complexity AmB technology. In particular, an original message was divided into two messages, and they are transmitted over two channels, i.e., direct transmission channel and AmB communication channel. Since the AmB tag only backscatters the RF signals to transmits data, instead of generating active signals, the eavesdropper is unlikely aware of the existence of AmB transmissions. Even if eavesdroppers can capture

AmB signals, our proposed splitting message introduces much more difficulties for the eavesdroppers to derive the original message. To effectively decode AmB signals at the receiver, we have developed a signal detector based on DL. Unlike MLK-based solutions, our detector did not rely on a complex mathematical model nor require perfect CSI. To make the DL-based detector to quickly achieve good performance in new environments, we have developed the meta-learning technique for the training process. The simulation results have shown that our proposed approach can not only ensure the security of the data communications in the presence of eavesdroppers but also can achieve the BER performance that is comparable to the MLK-based detector, which relies on perfect CSI.

Chapter 7

Conclusions and Potential Research Directions

7.1 Conclusion

In this thesis, we have presented our works in developing advanced machine learning-based solution for various aspects of 6G networks, including NTNs, ICAS, resource management for the Metaverse, and security, with the goal of achieving effective and secure solutions.

In the first study, we have tackle the problem in UAV-based systems, which is an important component of NTNs in 6G. Specifically, we have developed a novel Deep Dueling Double Q-learning with Transfer Learning algorithm (D3QL-TL) that jointly optimizes the flying speed and energy replenishment activities for the UAV to maximize the data collection performance of a UAV-assisted IoT system. The proposed algorithm effectively addresses not only the dynamic and uncertainty of the system but also the high dimensional state and action spaces of the underlying MDP problem with hundreds of thousands of states. In addition, the proposed TL techniques (i.e., experience transfer, policy transfer, and hybrid transfer) allow UAVs to "share" and "transfer" their learned knowledge, resulting in a decrease of learning time and an improvement of learning quality. The simulation results have showed that our proposed solution can significantly improve the system performance (i.e., data collection and energy usage efficiency) and has a remarkably lower computational complexity compared with other conventional approaches.

In the second study, we have addressed the challenge in the Integrated Communications and Sensing (ICAS) technology that plays a critical role in AVs, a use case

of 6G. In particular, we have developed a novel MDP-based framework that allows an ICS-AV to automatically and adaptively decide its optimal waveform structure based on the observations to maximize the overall performance of the ICS system. Then, we have proposed an advanced learning algorithm, i.e., i-ICS, that can help the ICS-AV gradually learn an optimal policy through interactions with the surrounding environment without requiring complete knowledge about the environment in advance. As such, our proposed approach can effectively handle the environment's dynamic and uncertainty as well as the high dimensional state space problem of the underlying MDP framework. The extensive simulation results have clearly shown that the proposed solution can strike a balance between communication efficiency and sensing accuracy, thereby consistently outperforming the benchmark methods in different scenarios.

In the third study, we have addressed the resource management for Metaverse, a new service supported by 6G. We have proposed two innovative techniques, i.e., the application decomposition and the MetaInstance, to maximize resource utilization for the Metaverse built on a multi-tier computing architecture. Based on these techniques, we have developed a novel framework for the Metaverse, i.e., MetaSlicing, that can smartly allocate resources for MetaSlices, i.e., Metaverse applications, to maximize the system performance. Moreover, we have proposed a highly effective framework based on sMDP together with an intelligent algorithm, i.e., iMSAC, to find the optimal admission policy for the Admission Controller under the high dynamics and uncertainty of resource demands. The extensive simulation results have clearly demonstrated the robustness and superiority of our proposed solution compared with the counterpart methods as well as revealed key factors determining the system performance. As Metaverse continues to evolve and expand, there will be enormous potential research topics that we can explore. For example, elastic resource allocation approaches can be developed to further enhance the efficiency of

using MetaSlices. In particular, allocated resources of a MetaSlice can be scaled dynamically according to the resource demand of this MetaSlice during operation, thus further improving resource utilization. Another topic that we can further consider is methods for ensuring interoperability between different MetaSlices in the Metaverse. This could include developing standards and protocols for communication between different MetaSlices and mechanisms for managing conflicts and ensuring consistency between Metaslices.

Our fourth study has introduced a novel anti-eavesdropping framework that can enable secure wireless communications leveraging the low-cost and low-complexity AmB technology, which is an emerging technology in 6G. In particular, an original message was divided into two messages, and they are transmitted over two channels, i.e., direct transmission channel and AmB communication channel. Since the AmB tag only backscatters the RF signals to transmits data, instead of generating active signals, the eavesdropper is unlikely aware of the existence of AmB transmissions. Even if eavesdroppers can capture AmB signals, our proposed splitting message introduces much more difficulties for the eavesdroppers to derive the original message. To effectively decode AmB signals at the receiver, we have developed a signal detector based on DL. Unlike MLK-based solutions, our detector did not rely on a complex mathematical model nor require perfect CSI. To make the DL-based detector to quickly achieve good performance in new environments, we have developed the meta-learning technique for the training process. The simulation results have shown that our proposed approach can not only ensure the security of the data communications in the presence of eavesdroppers but also can achieve the BER performance that is comparable to the MLK-based detector, which relies on perfect CSI.

Together, these above studies illustrate a holistic approach to overcoming the challenges posed by the next generation of wireless networks. They demonstrate

how different components of 6G technology, ranging from aerial data collection and integrated communication-sensing to resource management in virtual environments and secure data transmission, can be optimized using advanced machine learning techniques. Each study, while distinct in its focus, contributes to the overarching goal of creating a robust, efficient, and secure 6G network. They complement and supplement each other by showcasing the versatility of machine learning applications across various 6G network challenges, collectively moving towards fulfilling the broader aim of revolutionizing wireless communication systems with 6G technology.

It is worth noting that our proposed solutions are based on deep neural networks, and thus they cannot guarantee optimality. Additionally, since the thesis explores emerging topics in 6G networks, which are in their infancy, datasets and testbeds are not publicly available. Also, due to our limited resources, our proposed approaches were evaluated in simulated environments, which may not completely reflect real-world conditions. However, the obtained results have still provided insights into the effectiveness of our proposed solutions and the promise of machine learning in 6G networks.

7.2 Future Works

Our results show great potential for leveraging advanced machine learning to enhance various aspects of 6G networks, such as performance, energy efficiency, security, and adaptability. Recent advances in machine learning, such as generative AI and Contrastive Representation Learning, present further opportunities for enhancing many aspects of 6G networks.

- *Interference management:* With the emergence of new services and technologies, heterogeneous infrastructures are expected to be deployed in 6G net-

works, making interferences among wireless systems become more critical. Consequently, conventional solutions based on optimization theory (which often requires complete information about the system) may not be effective in dealing with this over-complexed problem, especially in a dense network like 6G. Thus, there is an urgent need for an effective solution to facilitate the deployment of 6G. Recently, emerging advanced AI methods, e.g., generative AI and meta-learning, are promising approaches to mitigate the interference problem in 6G networks to improve signal quality and overall network performance. For instance, generative AI can be used to simulate various network conditions, including interference scenarios, which can help in training other AI models to automatically adjust network parameters for optimal performance. This could improve signal quality and overall network performance by enabling more adaptive and responsive network management strategies. Therefore, they need to be further explored.

- *Intrusion detection systems:* In future wireless systems, billions of wireless devices will connect simultaneously. Identifying malicious devices presents a significant challenge, as they can disguise their true nature. In this context, predictive AI, which involves forecasting future states based on historical data, can be crucial for identifying potential network failures, performance bottlenecks, or security threats before they occur. In 6G networks, predictive AI can analyze patterns in network traffic, device behavior, and resource utilization to predict and prevent disruptions. However, as more traffic becomes encrypted, device identification becomes more difficult. Thus, an innovative approach is required to effectively address this challenge

- *Security of data and AI models:* Aiming to support the Internet of Everything (IoE) [1, 2], 6G networks indeed consist of a massive number of devices, most

of which have limited resource capabilities in terms of computing, storage, and networking. Thus, it is challenging to use advanced cryptography on such devices to secure data in transit. A weak mechanism makes exchanged data susceptible to being tampered with. If an attacker can intentionally modify the input data of AI models, they can severely poison AI models. Thus, there needs an innovative solution for this issue. A promising solution is to leverage models that can directly operate with encrypted data. By doing so, the need for data decryption and key distribution is eliminated, thus reducing the required computing resources while maintaining security of data transmission. In the literature, this approach is still underexplored, and thus it needs to be further investigated.

Appendix A

Proofs in Chapter 5

A.1 The Proof of Theorem 1

First, we need to prove the existence of the limiting matrix $\overline{\mathcal{T}}_\pi$ given in (5.12). In [168], it is proven that for an aperiodic irreducible chain (as that of our proposed sMDP), the limiting matrix (which is the Cesaro limit of order zero named *C-lim*) of the transition matrix, i.e.,$\mathcal{T}_\pi^g$, exists as follows:

$$\overline{\mathcal{T}}_\pi = \textit{C-lim} = \lim_{G\to\infty} \frac{1}{G} \sum_{g=0}^{G-1} \mathcal{T}_\pi^g. \tag{A.1}$$

Next, because the total probabilities that a given state moves to others is one, i.e., $\sum_{\mathbf{s}'\in\mathcal{S}} \mathcal{T}_\pi(\mathbf{s}|\mathbf{s}') = 1$, we have:

$$\overline{\mathcal{T}}_\pi r(\mathbf{s}, \pi(\mathbf{s})) = \lim_{G\to\infty} \frac{1}{G+1} \mathbb{E}\left[\sum_{g=0}^{G} r(\mathbf{s}_g, \pi(\mathbf{s}_g))\right], \forall \mathbf{s} \in \mathcal{S}, \tag{A.2}$$

$$\overline{\mathcal{T}}_\pi y(\mathbf{s}, \pi(\mathbf{s})) = \lim_{G\to\infty} \frac{1}{G+1} \mathbb{E}\left[\sum_{g=0}^{G} \tau_g\right] \forall \mathbf{s} \in \mathcal{S}. \tag{A.3}$$

It is observed that the long-term average reward $\mathcal{R}_\pi(\mathbf{s})$ in (5.9) is the ratio between $\overline{\mathcal{T}}_\pi r(\mathbf{s}, \pi(\mathbf{s}))$ and $\overline{\mathcal{T}}_\pi y(\mathbf{s}, \pi(\mathbf{s}))$. Note that by the quotient law for limits, the limit of a division of two functions is equivalent to the division of the limit of each function if the limit of a function at the denominator is not equal to zero. Since τ_g, i.e., the interval time between two consecutive decision epochs, is always larger than zero, $\overline{\mathcal{T}}_\pi y(\mathbf{s}, \pi(\mathbf{s}))$ is always larger than zero. Therefore, the long-term average reward $\mathcal{R}_\pi(\mathbf{s})$ exists.

A.2 The Proof of Theorem 2

This proof is based on the irreducible property of the underlying Markov chain, which is proven as follows. As mentioned in Section 5.2.2, the considered state space $\mathcal{S}$ consists of the currently available resources in the system, the required resources, the class ID, and the similarity of the MetaSlice request. Recall that the request arrival and MetaSlice departure follow the Poisson and exponential distributions, respectively. In addition, the requested MetaSlice can have any function supported by the MetaSlicing system, and thus its required resources are arbitrary. Given the above, suppose that the Admission Controller observes state $\mathbf{s}$ at time t, then the system state can move to any other state $\mathbf{s}' \in \mathcal{S}$ after a finite time step. Therefore, the proposed sMDP is irreducible, and thus the long-term average reward function $\mathcal{R}_\pi(\mathbf{s})$ is well-defined regardless of the initial state and under any policy.

Appendix B

Proofs in Chapter 6

B.1 The proof of Theorem 6.1

In this appendix, we prove Theorem 6.1 in a similar way as in [82]. Since the mutual information between the AmB tag's state e, i.e., $\mathcal{M}(e;\mathbf{y})$ and the received signals $\mathbf{y}$ defines the achievable AmB rate, we can obtain the maximum achievable AmB rate by [82]

$$R^*_{AmB} = \mathbb{E}[\mathcal{M}(e,\mathbf{y})], \tag{B.1}$$

where

$$\mathcal{M}(e,\mathbf{y}) = Z(\theta_0) - \mathbb{E}_{\mathcal{V}}[C(e|\mathcal{V})]. \tag{B.2}$$

In (B.2), $Z(\theta_0)$ is the binary entropy function defined by (B.3), and $C(e|\mathcal{V})$ is the conditional entropy of e given $\mathcal{V}$.

$$Z(\theta_0) \triangleq -\theta_1 \log_2 \theta_1 - \theta_0 \log_2 \theta_0. \tag{B.3}$$

Note that since $Z(\theta_0)$ does not depend on the channel coefficients, the maximum achievable AmB rate R^*_{AmB} is given by

$$R^*_{AmB} = \mathbb{E}[\mathcal{M}(e,\mathbf{y})] = Z(\theta_0) - \mathbb{E}_{\mathcal{V}}[C(e|\mathcal{V})]. \tag{B.4}$$

In addition, given $\tilde{\mathbf{y}}$, the posterior probability of e is calculated by

$$p(e=j|\mathcal{V}) = \frac{\theta_j p(\mathbf{y}|e=j)}{\theta_1 p(\mathcal{V}|e=1) + \theta_0 p(\mathcal{V}|e=0)}, \tag{B.5}$$

with $j \in \{0, 1\}$. By letting $\mu_j = p(e = j|\mathcal{V})$, the conditional entropy $C(e|\mathcal{V})$ is expressed by

$$C(e|\mathcal{V}) = -\sum_{j=0}^{1} \mu_j \log_2 \mu_j = Z(\mu_0). \tag{B.6}$$

Consequently, we have [82]

$$\begin{aligned} R^*_{AmB} &= Z(\theta_0) - \mathbb{E}_{\tilde{\mathbf{y}}}[Z(\mu)] \\ &= Z(\theta_0) - \int_{\mathcal{V}} \big(\theta_1 p(\mathcal{V}|e=1) + \theta_0 p(\mathcal{V}|e=0)\big) Z(\mu_0) d\mathcal{V}. \end{aligned} \tag{B.7}$$

Bibliography

[1] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE Network*, vol. 34, no. 3, pp. 134–142, Oct. 2020.

[2] I. F. Akyildiz, A. Kak, and S. Nie, "6G and beyond: The future of wireless communications systems," *IEEE Access*, vol. 8, pp. 133 995–134 030, Jul. 2020.

[3] L. U. Khan, I. Yaqoob, M. Imran, Z. Han, and C. S. Hong, "6G wireless systems: A vision, architectural elements, and future directions," *IEEE Access*, vol. 8, pp. 147 029–147 044, Aug. 2020.

[4] "Loon: Expanding internet connectivity with stratospheric balloons." [Online]. Available: https://loon.com/

[5] "Connecting the world from the sky," Mar. 2014. [Online]. Available: https://about.fb.com/news/2014/03/connecting-the-world-from-the-sky/

[6] P. Papadimitratos, A. L. Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza, "Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation," *IEEE Communication Magazine*, vol. 47, no. 11, pp. 84–95, Nov. 2009.

[7] J. A. Z. al., "An overview of signal processing techniques for joint communication and radar sensing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 6, pp. 1295–1315, Nov. 2021.

[8] "Introducing Meta: A social technology company," https://about.fb.com/news/2021/10/facebook-company-is-now-meta/, (accessed: May 01, 2022).

[9] "Mesh for Microsoft Teams aims to make collaboration in the 'Metaverse' personal and fun," https://news.microsoft.com/innovation-stories/mesh-for-microsoft-teams/, (accessed: May 01, 2022).

[10] M. Xu, W. C. Ng, W. Y. B. Lim, J. Kang, Z. Xiong, D. Niyato, Q. Yang, X. S. Shen, and C. Miao, "A full dive into realizing the edge-enabled Metaverse: Visions, enabling technologies, and challenges," *arXiv preprint arXiv:2203.05471*, 2022.

[11] Meta Quest. Horizon Worlds. (Dec. 10, 2021). Accessed: May 01, 2022. [Online]. Available: https://www.youtube.com/watch?v=02kCEurWkqU&t=64s

[12] Y. Wu, K. Zhang, and Y. Zhang, "Digital twin networks: A survey," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13 789–13 804, May 2021.

[13] X. Yu, D. Owens, and D. Khazanchi, "Building socioemotional environments in Metaverses for virtual teams in healthcare: A conceptual exploration," in *Proceedings of International Conference on Health Information Science*, 2012, pp. 4–12.

[14] C. Kwon, "Smart city-based Metaverse a study on the solution of urban problems," *Journal of the Chosun Natural Science*, vol. 14, no. 1, pp. 21–26, Mar. 2021.

[15] H. Jeong, Y. Yi, and D. Kim, "An innovative e-commerce platform incorporating metaverse to live commerce," *International Journal of Innovative Computing, Information and Control*, vol. 18, no. 1, pp. 221–229, Feb. 2022.

[16] J. Hu, S. Yan, X. Zhou, F. Shu, J. Li, and J. Wang, "Covert communication achieved by a greedy relay in wireless networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 7, pp. 4766–4779, Jul. 2018.

[17] "When can quantum annealing win?" Google Research. https://ai.googleblog.com/2015/12/when-can-quantum-annealing-win.html, (accessed: May 08, 2023).

[18] B. A. Bash, D. Goeckel, D. Towsley, and S. Guha, "Hiding information in noise: Fundamental limits of covert wireless communication," *IEEE Communications Magazine*, vol. 53, no. 12, pp. 26–31, Dec. 2015.

[19] A. Mukherjee, S. A. A. Fakoorian, J. Huang, and A. L. Swindlehurst, "Principles of physical layer security in multiuser wireless networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1550–1573, Feb. 2014.

[20] P. J. Freire, S. Srivallapanondh, A. Napoli, J. E. Prilepsky, and S. K. Turitsyn, "Computational complexity evaluation of neural network applications in signal processing," *arXiv preprint arXiv:2206.12191*, 2022.

[21] X. Zeng, F. Ma, T. Chen, X. Chen, and X. Wang, "Age-optimal UAV trajectory planning for information gathering with energy constraints," in *Proceedings of 2020 IEEE/CIC International Conference on Communications in China (ICCC)*, 2020, pp. 881–886.

[22] O. Bouhamed, H. Ghazzai, H. Besbes, and Y. Massoud, "A UAV-assisted data collection for wireless sensor networks: Autonomous navigation and scheduling," *IEEE Access*, vol. 8, pp. 10 446–11 046, Jun. 2020.

[23] Y. Zhang, Z. Mou, F. Gao, L. Xing, J. Jiang, and Z. Han, "Hierarchical deep reinforcement learning for backscattering data collection with multiple UAVs," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3786–3800, Mar. 2021.

[24] S. F. al., "Energy-efficient UAV-enabled data collection via wireless charging: A reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 10 209–10 219, Jun. 2021.

[25] X. Xu, H. Zhao, H. Yao, and S. Wang, "A blockchain-enabled energy-efficient data collection system for UAV-assisted IoT," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2431–2443, Feb. 2021.

[26] F. Shan, J. Luo, R. Xiong, W. Wu, and J. Li, "Looking before crossing: An optimal algorithm to minimize UAV energy by speed scheduling with a practical flight energy model," in *Proceedings of the 2020 IEEE Conference on Computer Communications (INFOCOM)*, 2020, pp. 1758–1767.

[27] J. Gong, T. Chang, C. Shen, and X. Chen, "Flight time minimization of UAV for data collection over wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 1942–1954, Sep. 2018.

[28] Q. Pan, X. Wen, Z. Lu, L. Li, and W. Jing, "Dynamic speed control of unmanned aerial vehicles for data collection under internet of things," *Sensors*, vol. 18, pp. 1–18, Nov. 2018.

[29] X. Lin, G. Su, B. Chen, H. Wang, and M. Dai, "Striking a balance between system throughput and energy efficiency for UAV-IoT systems," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 519–10 533, Dec. 2019.

[30] K. Li, W. Ni, E. Tovar, and A. Jamalipour, "On-board deep Q-network for UAV-assisted online power transfer and data collection," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 12, pp. 12 215–12 226, Dec. 2019.

[31] 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, no. 1, 2016, pp. 2094–2100.

[32] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *Journal of Machine Learning Research*, vol. 7, pp. 1633–1685, Sep. 2009.

[33] V. M. al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[34] Z. W. al., "Dueling network architectures for deep reinforcement learning," in *Proceedings of The 33rd International Conference on Machine Learning*, 2016, pp. 1995–2003.

[35] G. Naik, B. Choudhury, and J. Park, "IEEE 802.11bd & 5G NR V2X: Evolution of radio access technologies for V2X communications," *IEEE Access*, vol. 7, pp. 70 169–70 184, May 2019.

[36] J. B. Kenney, "Dedicated short-range communications (DSRC) standards in the United States," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, Jul. 2011.

[37] J. Choi, V. Va, N. Gonzalez-Prelcic, R. Daniels, C. R. Bhat, and W. Heath, "Millimeter-wave vehicular communication to support massive automotive sensing," *IEEE Communication Magazine*, vol. 54, no. 12, pp. 160–167, Dec. 2016.

[38] P. Kumari, S. A. Vorobyov, and R. W. Heath, "Adaptive virtual waveform design for millimeter-wave joint communication-radar," *IEEE Transactions on Signal Processing*, vol. 68, pp. 715–730, Nov. 2019.

[39] X. Cheng, D. Duan, S. Gao, and L. Yang, "Integrated sensing and communications (ISAC) for vehicular communication networks (VCN)," *IEEE Internet of Things Journal*, pp. 1–12, Jul. 2022.

[40] J. Hasch, E. Topak, R. Schnabel, T. Zwick, R. Weigel, and M.-w. C. Waldschmidt, "Technology for automotive radar sensors in the 77 GHz frequency band," *IEEE Transactions on Microwave Theory and Techniques*, vol. 60, no. 3, pp. 845–860, Mar. 2012.

[41] T. S. Rappaport, G. R. MacCartney, M. K. Samimi, and S. Sun, "Wideband millimeter-wave propagation measurements and channel models for future wireless communication system design," *IEEE Transactions on Communications*, vol. 63, no. 9, pp. 3029–3056, Sep. 2015.

[42] P. Kumari, N. Gonzalez-Prelcic, and R. W. H. Jr, "Investigating the IEEE 802.11ad standard for millimeter wave automotive radar," *in Proceedings of Vehicular Technology Conference*, pp. 3587–3591, Sep. 2015.

[43] E. Grossi, M. Lops, L. Venturino, and A. Zappone, "Opportunistic radar in IEEE 802.11ad networks," *IEEE Transactions on Signal Processing*, vol. 66, no. 9, pp. 2441–2454, May 2018.

[44] P. Kumari, J. Choi, N. Gonzalez-Prelcic, and R. W. Heath, "IEEE 802.11ad-based radar: An approach to joint vehicular communication-radar system," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 4, pp. 3012–3027, Nov. 2017.

[45] G. R. Muns, K. V. Mishra, C. B. Guerra, Y. C. Eldar, and K. R. Chowdhury, "Beam alignment and tracking for autonomous vehicular communication using IEEE 802.11ad-based radar," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops*, 2019, pp. 535–540.

[46] S. H. Dokhanchi, B. S. Mysore, K. V. Mishra, and B. Ottersten, "A mmwave automotive joint radar-communications system," *IEEE Transactions on Aerospace and Electronic Systems*, pp. 1241–1260, Feb. 2019.

[47] M. Wang, P. Chen, Z. Cao, and Y. Chen, "Reinforcement learning-based UAVs resource allocation for integrated sensing and communication (ISAC) system," *Electronics*, Feb. 2022.

[48] T. Xu, T. Zhou, J. Tian, J. Sang, and H. Hu, "Intelligent spectrum sensing: When reinforcement learning meets automatic repeat sensing in 5G communications," *IEEE Wireless Communications*, pp. 46–53, Feb. 2020.

[49] Q. Zhou, Y. Li, and Y. Niu, "Intelligent anti-jamming communication for wireless sensor networks: A multi-agent reinforcement learning approach," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 775–784, Feb. 2021.

[50] J. Clement, "Most played games on steam 2021, by peak player count," https://www.statista.com/statistics/656278/steam-most-played-games-peak-concurrent-player/, (accessed: May 01, 2022).

[51] "Metaverse: A guide to the next-gen internet," https://www.credit-suisse.com/media/assets/corporate/docs/about-us/media/media-release/2022/03/metaverse-14032022.pdf, (accessed: May 01, 2022).

[52] X. Wang *et al.*, "Characterizing the gaming traffic of World of Warcraft: From game scenarios to network access technologies," *IEEE Network*, vol. 26, no. 1, pp. 27–34, Jan. 2012.

[53] J. Zhao, R. S. Allison, M. Vinnikov, and S. Jennings, "Estimating the motion-to-photon latency in head mounted displays," in *Proceedings of the 2017 IEEE Virtual Reality (VR)*, 2017, pp. 313–314.

[54] Y. Jiang, J. Kang, D. Niyato, X. Ge, Z. Xiong, and C. Miao, "Reliable coded

distributed computing for Metaverse services: Coalition formation and incentive mechanism design," *arXiv preprint arXiv:2111.10548*, 2021.

[55] M. Xu, D. Niyato, J. Kang, Z. Xiong, C. Miao, and D. I. Kim, "Wireless edge-empowered Metaverse: A learning-based incentive mechanism for virtual reality," *arXiv preprint arXiv:2111.03776*, 2021.

[56] W. C. Ng, W. Y. B. Lim, J. S. Ng, Z. Xiong, D. Niyato, and C. Miao, "Unified resource allocation framework for the edge intelligence-enabled Metaverse," *arXiv preprint arXiv:2110.14325*, 2021.

[57] Y. Han, D. Niyato, C. Leung, C. Miao, and D. I. Kim, "A dynamic resource allocation framework for synchronizing Metaverse with IoT service and data," *arXiv preprint arXiv:2111.00431*, 2021.

[58] K. Cao, Y. Liu, G. Meng, and Q. Sun, "An overview on edge computing research," *IEEE Access*, vol. 8, pp. 85 714–85 728, May. 2020.

[59] "Google Maps platform documentation," https://developers.google.com/maps/documentation, (accessed: May 01, 2022).

[60] I. Singh, "Your next favorite AR game will be built on Google Maps," https://geoawesomeness.com/google-maps-will-now-power-location-aware-augmented-reality-games/, (accessed: May 01, 2022).

[61] S. Hayden, "'Wooorld' is a multiplayer version of 'Google Earth VR' for Quest, releasing soon," https://www.roadtovr.com/wooorld-google-earth-vr-quest-2/, (accessed: May 01, 2022).

[62] N. Francis, "Collection of the coolest uses of the Google Maps API," https://www.jotform.com/blog/

collection-of-the-coolest-uses-of-the-google-maps-api/, (accessed: May 01, 2022).

[63] Y. Zhang, Y. Shen, H. Wang, J. Yong, and X. Jiang, "On secure wireless communications for IoT under eavesdropper collusion," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 3, pp. 1281–1293, Jul. 2016.

[64] X. Lu, E. Hossain, T. Shafique, S. Feng, H. Jiang, and D. Niyato, "Intelligent reflecting surface enabled covert communications in wireless networks," *IEEE Network*, vol. 34, no. 5, pp. 148–155, Oct. 2020.

[65] R. Soltani, D. Goeckel, D. Towsley, B. A. Bash, and S. Guha, "Covert wireless communication with artificial noise generation," *IEEE Transactions on Wireless Communications*, vol. 17, no. 11, pp. 7252–7267, Nov. 2018.

[66] P. Siyari, M. Krunz, and D. N. Nguyen, "Friendly jamming in a mimo wiretap interference network: A nonconvex game approach," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 3, pp. 601–614, Mar. 2017.

[67] Z. Mobini, M. Mohammadi, and C. Tellambura, "Wireless-powered full-duplex relay and friendly jamming for secure cooperative communications," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 621–634, Mar. 2019.

[68] L. Yang, J. Chen, H. Jiang, S. A. Vorobyov, and H. Zhang, "Optimal relay selection for secure cooperative communications with an adaptive eavesdropper," *IEEE Transactions on Wireless Communications*, vol. 16, no. 1, pp. 26–42, Jan. 2017.

[69] D. T. Hoang, D. Niyato, D. I. Kim, N. Van Huynh, and S. Gong, *Ambient backscatter communication networks*. Cambridge University Press, 2020.

[70] V. Liu, A. Parks, V. Talla, S. Gollakota, D. Wetherall, and J. R. Smith, "Ambient backscatter: Wireless communication out of thin air," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 39–50, Aug. 2013.

[71] D. Bharadia, K. R. Joshi, M. Kotaru, and S. Katti, "Backfi: High throughput wifi backscatter," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 283–296, Aug. 2015.

[72] A. N. Parks, A. Liu, S. Gollakota, and J. R. Smith, "Turbocharging ambient backscatter communication," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 619–630, Aug. 2014.

[73] B. Kellogg, V. Talla, S. Gollakota, and J. R. Smith, "Passive Wi-Fi: Bringing low power to Wi-Fi transmissions," in *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, 2016, pp. 151–164.

[74] A. Wang, V. Iyer, V. Talla, J. R. Smith, and S. Gollakota, "FM backscatter: Enabling connected cities and smart fabrics," in *Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, 2017, pp. 243–258.

[75] P. V. Nikitin, S. Ramamurthy, R. Martinez, and K. S. Rao, "Passive tag-to-tag communication," in *Proceedings of the 2012 IEEE International Conference on RFID (RFID)*, 2012, pp. 177–184.

[76] D. T. Hoang, D. Niyato, P. Wang, D. I. Kim, and Z. Han, "The tradeoff analysis in rf-powered backscatter cognitive radio networks," in *Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM)*, 2016, pp. 1–6.

[77] X. Li, M. Zhao, Y. Liu, L. Li, Z. Ding, and A. Nallanathan, "Secrecy analysis of ambient backscatter NOMA systems under I/Q imbalance," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 12 286–12 290, Oct. 2020.

[78] X. Li, M. Zhao, M. Zeng, S. Mumtaz, V. G. Menon, Z. Ding, and O. A. Dobre, "Hardware impaired ambient backscatter NOMA systems: Reliability and security," *IEEE Transactions on Communications*, vol. 69, no. 4, pp. 2723–2736, Apr. 2021.

[79] X. Li, Y. Zheng, W. U. Khan, M. Zeng, D. Li, G. Ragesh, and L. Li, "Physical layer security of cognitive ambient backscatter communications for green Internet-of-Things," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 3, pp. 1066–1076, Sep. 2021.

[80] N. Van Huynh, D. T. Hoang, D. N. Nguyen, E. Dutkiewicz, D. Niyato, and P. Wang, "Reinforcement learning approach for rf-powered cognitive radio network with ambient backscatter," in *Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[81] S. Boztas, "Comments on "An inequality on guessing and its application to sequential decoding"," *IEEE Transactions on Information Theory*, vol. 43, no. 6, pp. 2062–2063, 1997.

[82] H. Guo, Q. Zhang, D. Li, and Y.-C. Liang, "Noncoherent multiantenna receivers for cognitive backscatter system with multiple RF sources," *arXiv preprint arXiv:1808.04316*, 2018.

[83] H. Guo, Q. Zhang, S. Xiao, and Y.-C. Liang, "Exploiting multiple antennas for cognitive ambient backscatter communication," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 765–775, Feb. 2019.

[84] H. Ye, G. Y. Li, and B.-H. Juang, "Power of deep learning for channel estimation and signal detection in OFDM systems," *IEEE Wireless Communications Letters*, vol. 7, no. 1, pp. 114–117, Feb. 2018.

[85] C. Fan, X. Yuan, and Y.-J. Zhang, "CNN-based signal detection for banded linear systems," *IEEE Transactions on Wireless Communications*, vol. 18, no. 9, pp. 4394–4407, Sep. 2019.

[86] S. Rajendran, W. Meert, D. Giustiniano, V. Lenders, and S. Pollin, "Deep learning models for wireless signal classification with distributed low-cost spectrum sensors," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 3, pp. 433–445, Sep. 2018.

[87] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168–179, Feb. 2018.

[88] N. Van Huynh, D. N. Nguyen, D. T. Hoang, T. X. Vu, E. Dutkiewicz, and S. Chatzinotas, "Defeating super-reactive jammers with deception strategy: Modeling, signal detection, and performance analysis," *IEEE Transactions on Wireless Communications*, vol. 21, no. 9, pp. 7374–7390, Sep. 2022.

[89] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.

[90] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 5149–5169, Sep. 2022.

[91] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the International Conference on Machine Learning*, 2017, pp. 1126–1135.

[92] "Introducing deep learning with MATLAB," Matlab. https://www.mathworks.com/campaigns/offers/next/deep-learning-ebook.html, (accessed: May 08, 2023).

[93] S. Peng, S. Sun, and Y.-D. Yao, "A survey of modulation classification using deep learning: Signal representation and data preprocessing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 7020–7038, Dec. 2022.

[94] S. Ruping, "Incremental learning with support vector machines," in *Proceedings of the 2001 IEEE International Conference on Data Mining*, 2001, pp. 641–642.

[95] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[96] S. Du, J. Lee, H. Li, L. Wang, and X. Zhai, "Gradient descent finds global minima of deep neural networks," in *Proceeding of the International Conference on Machine Learning*, 2019, pp. 1675–1685.

[97] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, pp. 400–407, Sep. 1951.

[98] H. C. Tijms, *A First Course in Stochastic Models*. Wiley, 2003.

[99] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[100] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[101] 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*, vol. 21, no. 4, pp. 3133–3174, May 2019.

[102] S. Thrun and A. Schwartz, "Issues in using function approximation for reinforcement learning," in *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, vol. 6, 1993, pp. 1–9.

[103] S. Halkjær and O. Winther, "The effect of correlated input data on the dynamics of learning," in *Proceedings of the 9th International Conference on Neural Information Processing Systems*, vol. 9, 1996, pp. 169–175.

[104] V. A. Papavassiliou and S. Russell, "Convergence of reinforcement learning with general function approximators," in *Proceedings of IJCAI*, 1999, pp. 748–755.

[105] T. Xu and I. Darwazeh, "Design and prototyping of neural network compression for non-orthogonal iot signals," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, 2019, pp. 1–6.

[106] [Online]. Available: https://www.tesla.com/en_AU/AI

[107] D. A. Pomerleau, "ALVINN: an autonomous land vehicle in a neural network," in *Advances in Neural Information Processing Systems 1*, 1989, p. 305–313.

[108] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, Oct. 2009.

[109] Z. Zhu, K. Lin, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," *arXiv preprint arXiv:2009.07888*, 2020.

[110] C. T. Nguyen *et al.*, "Transfer learning for wireless networks: A comprehensive survey," *Proceedings of the IEEE*, vol. 110, pp. 1073 – 1115, Jun. 2022.

[111] H. Khodr, N. Kouzayha, M. Abdallah, J. Costantine, and Z. Dawy, "Energy efficient IoT sensor with RF wake-up and addressing capability," *IEEE Sensors Letters*, vol. 1, no. 6, pp. 1–4, Dec. 2017.

[112] Q. Wu and R. Zhang, "Common throughput maximization in UAV-enabled OFDMA systems with delay consideration," *IEEE Transactions on Communications*, vol. 66, no. 12, pp. 6614–6627, Dec. 2018.

[113] C. H. Liu, X. Ma, X. Gao, and J. Tang, "Distributed energy-efficient multi-UAV navigation for long-term communication coverage by deep reinforcement learning," *IEEE Transactions on Mobile Computing*, vol. 19, no. 6, pp. 1274–1285, Jun. 2020.

[114] M. A. Abd-Elmagid, A. Ferdowsi, H. S. Dhillon, and W. Saad, "Deep reinforcement learning for minimizing age-of-information in UAV-assisted networks," in *Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[115] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, "Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 2059–2070, Sep. 2018.

[116] N. V. Huynh, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, "Deepfake: Deep dueling-based deception strategy to defeat reactive jammers," *IEEE Transactions on Wireless Communications*, vol. 20, no. 10, pp. 6898–6914, Oct. 2021.

[117] A. G. Andrade and V. Parra, "Evaluation of uwb interference at wimax systems based on a generalized pulse waveform," in *2009 International Conference on Electrical, Communications, and Computers*, 2009, pp. 214–219.

[118] Tech. Rep. [Online]. Available: https://scdn.rohde-schwarz.com/ur/pws/dl_downloads/dl_application/application_notes/1ma220/1MA220_3e_WLAN_11ad_WP.pdf

[119] P. Kumari, W. Heath, and S. A. Vorobyov, "Virtual pulse design for IEEE 802.11ad-based joint communication radar," in *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018, pp. 3315–3319.

[120] M. A. Richard, *Fundamentals of Radar Signal Processing.* Education: McGraw-Hill, 2014.

[121] "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. Amendment 3: Enhancements for very high throughput in the 60 GHz band," Tech. Rep., 2012.

[122] R. W. H. Jr and A. Lozano, *Foundations of MIMO Communication.* Cambridge University Press, 2019.

[123] V. Va, T. Shimizu, G. Bansal, and R. W. H. Jr, *Millimeter wave vehicular communications: A survey.* Now: Foundations and Trends in Networking, 2016.

[124] A. Bazzi, C. Karnfelt, A. Peden, T. Chonavel, P. Galaup, and F. Bodereau, "Estimation techniques and simulation platforms for 77 GHz FMCW ACC radars," *The European Physical Journal - Applied Physics*, vol. 57, Nov. 2011.

[125] H. Rohling and M.-M. Meinecke, "Waveform design principles for automotive radar systems," in *Proceedings of the 2001 CIE International Conference on Radar*, 2001, pp. 1–4.

[126] G. R. Curry, *Radar System Performance Modeling, 2nd ed.* Artech House, 2004.

[127] R. Khalili and K. Salamatian, "A new analytic approach to evaluation of packet error rate in wireless networks," in *Proceedings of the 3rd Annual Communication Networks and Services Research Conference (CNSR'05)*, 2005, pp. 333–338.

[128] K. Nguyen, M. G. Kibria, K. Ishizu, and F. Kojima, "Performance evaluation of IEEE 802.11ad in evolving Wi-Fi networks," *Wireless Communications and Mobile Computing*, vol. 2019, Feb. 2019.

[129] R. S. Sutton and A. G. Barto, *Introduction to reinforcement learning*. Cambridge: MIT press, 1998, vol. 135.

[130] S. Sahhaf *et al.*, "Network service chaining with optimized network function embedding supporting service decompositions," *Computer Networks*, vol. 93, pp. 492–505, Dec. 2015.

[131] B. Alturki, S. Reiff-Marganiec, C. Perera, and S. De, "Exploring the effectiveness of service decomposition in fog computing architecture for the internet of things," *IEEE Transactions on Sustainable Computing*, vol. 7, no. 2, pp. 299–312, Mar. 2019.

[132] A. Wolke, M. Bichler, and T. Setzer, "Planning vs. dynamic control: Resource allocation in corporate clouds," *IEEE Transactions on Cloud Computing*, vol. 4, no. 3, pp. 322–335, Sep. 2014.

[133] S. Abbasloo, Y. Xu, and H. J. Chao, "C2TCP: A flexible cellular TCP to meet stringent delay requirements," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 4, pp. 918–932, Feb. 2019.

[134] "Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges & Call for Action," ESTI. Accessed: Apr. 24, 2022. [Online].

Available: https://docbox.etsi.org/isg/nfv/open/Publications_pdf/White%20Papers/NFV_White_Paper1_2012.pdf

[135] "Description of network slicing concept," NGMN Alliance. Accessed: Apr. 24, 2022. [Online]. Available: https://www.ngmn.org/wp-content/uploads/Publications/2016/161010_NGMN_Network_Slicing_framework_v1.0.8.pdf

[136] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2429–2453, Mar. 2018.

[137] A. Laghrissi and T. Taleb, "A survey on the placement of virtual resources and virtual network functions," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1409–1434, Dec. 2018.

[138] H. Zhang *et al.*, "Network slicing based 5G and future mobile networks: Mobility, resource management, and challenges," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 138–145, Aug. 2017.

[139] R. G. Gallager, *Discrete Stochastic Processes*. U.K.: Kluwer, 1995.

[140] L. Kallenberg, *Markov Decision Processes*. University of Leiden, 2016. Accessed: May 01, 2022. [Online]. Available: http://www.math.leidenuniv.nl/%7Ekallenberg/Lecture-notes-MDP.pdf

[141] G. Wang, G. Feng, W. Tan, S. Qin, R. Wen, and S. Sun, "Resource allocation for network slices in 5G with network resource pricing," in *Proceedings of the 2017 IEEE Global Communications Conference*, 2017, pp. 1–6.

[142] H. Roh, C. Jung, W. Lee, and D.-Z. Du, "Resource pricing game in geo-distributed clouds," in *Proceedings of the IEEE INFOCOM*, 2013, pp. 1519–1527.

[143] "Pricing-cloud services j Microsoft Azure," https://azure.microsoft.com/en-us/pricing/details/cloud-services/, (accessed: Apr 20, 2023).

[144] "Google compute engine pricing," https://cloud.google.com/compute/pricing, (accessed: Apr 20, 2023).

[145] W. Wu, N. Chen, C. Zhou, M. Li, X. Shen, W. Zhuang, and X. Li, "Dynamic RAN slicing for service-oriented vehicular networks via constrained learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2076–2089, Jul. 2021.

[146] X. Shen, J. Gao, W. Wu, K. Lyu, M. Li, W. Zhuang, X. Li, and J. Rao, "AI-assisted network-slicing based next-generation wireless networks," *IEEE Open Journal of Vehicular Technology*, vol. 1, pp. 45–66, Jan. 2020.

[147] W. Wu, C. Zhou, M. Li, H. Wu, H. Zhou, N. Zhang, X. S. Shen, and W. Zhuang, "AI-native network slicing for 6G networks," *IEEE Wireless Communications*, vol. 29, no. 1, pp. 96–103, Feb. 2022.

[148] P. Jaccard, "The distribution of the flora in the alpine zone," *The New Phytologist*, vol. 11, no. 2, pp. 37–50, Feb. 1912.

[149] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992.

[150] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[151] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[152] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[153] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[154] X. Wang, L. Zhang, Y. Liu, C. Zhao, and K. Wang, "Solving task scheduling problems in cloud manufacturing via attention mechanism and deep reinforcement learning," *Journal of Manufacturing Systems*, vol. 65, pp. 452–468, Oct. 2022.

[155] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry, "Implementation matters in deep policy gradients: A case study on ppo and trpo," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020, pp. 1–14.

[156] G. Dandachi, A. De Domenico, D. T. Hoang, and D. Niyato, "An artificial intelligence framework for slice deployment and orchestration in 5G networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 2, pp. 858–871, Nov. 2020.

[157] N. Van Huynh, D. T. Hoang, X. Lu, D. Niyato, P. Wang, and D. I. Kim, "Ambient backscatter communications: A contemporary survey," *IEEE Communications surveys & tutorials*, vol. 20, no. 4, pp. 2889–2922, May. 2018.

[158] Q. Zhang, H. Guo, Y.-C. Liang, and X. Yuan, "Constellation learning-based signal detection for ambient backscatter communication systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 2, pp. 452–463, Feb. 2019.

[159] S. Ma, G. Wang, R. Fan, and C. Tellambura, "Blind channel estimation for ambient backscatter communication systems," *IEEE Communications letters*, vol. 22, no. 6, pp. 1296–1299, Jun. 2018.

[160] W. Zhao, G. Wang, S. Atapattu, R. He, and Y.-C. Liang, "Channel estimation for ambient backscatter communication systems with massive-antenna reader," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 8254–8258, Aug. 2019.

[161] Y. Shen and E. Martinez, "Channel estimation in OFDM systems," *Freescale Semiconductor Application Note*, pp. 1–15, 2006.

[162] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *Proceedings of the International Conference on Machine Learning*, 2016, pp. 1842–1850.

[163] G. Koch, R. Zemel, R. Salakhutdinov *et al.*, "Siamese neural networks for one-shot image recognition," in *ICML Deep Learning Workshop*, vol. 2, no. 1, 2015.

[164] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[165] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[166] D. Wubben, R. Bohnke, V. Kuhn, and K.-D. Kammeyer, "Near-maximum-likelihood detection of MIMO systems using MMSE-based lattice-reduction," in *Proceedings of the 2004 IEEE International Conference on Communications*, vol. 2, 2004, pp. 798–802.

[167] P. Kyosti, "WINNER II channel models," *IST, Tech. Rep. IST-4-027756 WINNER II D1. 1.2 V1. 2*, 2007.

[168] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.