الجمهورية الجزائرية الديمقراطية الشعبية

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

وزارة التعليم العالي و البحث العلمي

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

جامعة أبي بكر بلقايد – تلمسان –

University of Aboubakr Belkaïd – Tlemcen –
Faculty of TECHNOLOGY

PhD THESIS

Submitted in Partial Fulfillment of the Requirements for the **Degree** of
3rd Cycle Doctorate

In: Telecommunications

Specialization: Telecommunications and Artificial Intelligence

By: **GHOMRI Benmeziane Imad-Ddine**

Subject

Artificial Intelligence Applied to NOMA-UAV Networks for IoT Communication Systems

Publicly defended on 13/10/2025, before the committee composed of:

Mr. MERIAH Sidi Mohammed	Professor	Univ. of Tlemcen	Chair
Mr. BENDIMERAD Mohammed Yassine	Professor	Univ. of Tlemcen	Thesis Supervisor
Mr. BENDIMERAD Fethi Tarik	Professor Emeritus	Univ. of Tlemcen	Thesis Co-supervisor
Ms. DEBBAT Fatima	Professor	Univ. of Mascara	Examiner
Ms. BENOSMAN Hayat	MCA	Univ. of Tlemcen	Examiner
Ms. BOUSALAH Fayza	MCA	Univ. of Tlemcen	Examiner
Mr. SHAIEK Hmaied	Professor	CNAM, France	Invited

Academic Year: 2025/2026

Acknowledgements

It is with sincere gratitude that I acknowledge the individuals and institutions whose support was instrumental in the completion of this doctoral thesis.

First and foremost, my deepest appreciation goes to my supervisors, Pr. Mohammed Yassine BENDIMERAD and Pr. Fethi Tarik BENDIMERAD. Their invaluable guidance, profound expertise, and unwavering encouragement throughout this research have been fundamental. I am particularly grateful for the intellectual freedom to research and experiment, which they granted me, balanced with timely direction whenever needed. Their mentorship extended beyond the academic, shaping my approach to research.

Sincere thanks are extended to Mr. Sidi Mohammed MERIAH (Professor, Univ. of Tlemcen, Chair), Ms. Fatima DEBBAT (Professor, Univ. of Mascara, Examiner), Ms. Faiza BOUSALAH (MCA, Univ. of Tlemcen, Examiner), and Ms. Hayat BENOSMAN (MCA, Univ. of Tlemcen, Examiner). I am grateful for their acceptance to serve on the committee and for the time and expertise they will dedicate to evaluating this research work.

I would like to extend my thanks to Professor SHAIEK Hmaied at CEDRIC research laboratory of the Conservatoire National des Arts et Métiers (CNAM) for the constructive discussions and valuable collaboration. His insights and expertise in the field have greatly enriched my understanding and contributed to the depth of this thesis.

I would also like to express my appreciation to all the members of the LTT research laboratory. Special thanks to my dear colleague, Reda MOSTEGHANEMI for his invaluable discussions and support.

Dedication

*To my beloved **Parents**,*

for your endless love, unwavering support, and constant belief in me.

*To my wonderful brothers, **Younes**, **Alaa eddine**, and **Taha**,*

for your encouragement and for always being there.

*And to my dear friends, **Abdelkader Belkacemi**, **Seyfeddine Boukahil**, and **Mohammed Salhi**,*

whose laughter, understanding, and companionship brightened these years.

This work is dedicated to you.

GHOMRI Benmeziane Imad-Ddine

Abstract

The fast proliferation of Internet of Things (IoT) devices and the increasing demand for reliable and energy-efficient wireless communication have motivated the exploration of advanced access technologies and intelligent control mechanisms. Unmanned Aerial Vehicles (UAVs), with their flexible deployment and line-of-sight connectivity, have emerged as a key enabler to extend network coverage in hard-to-reach areas. In this context, this thesis examines how the combination of Non-Orthogonal Multiple Access (NOMA), Deep Reinforcement Learning (DRL), and Simultaneously Transmit and Reflect Reconfigurable Intelligent Surfaces (STAR-RIS) can improve the effectiveness of UAV-assisted IoT communication networks.

The main objective is to design an AI-based framework that jointly optimizes the UAV trajectory, user scheduling, and resource allocation to maximize energy efficiency and ensure fair resource distribution. A DRL-based algorithm is proposed to enable autonomous decision-making in dynamic and complex environments, using a weighted reward function that balances energy efficiency and fairness and makes it easier to prioritize energy when needed or fairness when the situation requires it. Furthermore, a STAR-RIS-assisted architecture is introduced to enhance signal quality and EE by intelligently manipulating the wireless propagation environment.

Simulation results demonstrate that the proposed approaches significantly outperform baseline methods in terms of energy efficiency, spectral utilization, and fairness. The proposed DRL-based solution, which optimizes all optimization factors simultaneously, achieves higher EE when compared to benchmark methods of around 60% and 17.3% for EE with different fairness weights prioritization and around 85% in fairness index.

Keywords: Artificial intelligence, deep reinforcement learning, energy efficiency, IoT, NOMA, reconfigurable intelligent surfaces, UAV networks.

الملخص

إن الانتشار السريع لأجهزة إنترنت الأشياء (IoT) والطلب المتزايد على اتصالات لاسلكية موثوقة وفعالة من حيث استهلاك الطاقة قد حفز استكشاف تقنيات وصول متقدمة وآليات تحكم ذكية. لقد برزت المركبات الجوية بدون طيار (UAVs)، بفضل مرونة نشرها واتصالها المباشر، كعامل تمكين رئيسي لتوسيع تغطية الشبكة في المناطق التي يصعب الوصول إليها. في هذا السياق، تبحث هذه الأطروحة في كيفية أن الجمع بين تقنية الوصول المتعدد غير المتعامد (NOMA)، والتعلم المعزز العميق (DRL)، والأسطح الذكية القابلة لإعادة التشكيل ذات الإرسال والانعكاس المتزامن (STAR-RIS) يمكن أن يحسن من فعالية شبكات اتصالات إنترنت الأشياء المدعومة بالمركبات الجوية بدون طيار.

الهدف الرئيسي هو تصميم إطار عمل قائم على الذكاء الاصطناعي (AI) يقوم بالتحسين المشترك لمسار المركبة الجوية، وجدولة المستخدمين، وتخصيص الموارد بهدف تعظيم كفاءة الطاقة وضمان توزيع عادل للموارد. تم اقتراح خوارزمية قائمة على التعلم المعزز العميق (DRL) لتمكين اتخاذ القرار المستقل في بيئات ديناميكية ومعقدة، وذلك باستخدام دالة مكافأة موزونة توازن بين كفاءة الطاقة والعدالة، وتسهل إعطاء الأولوية للطاقة عند الحاجة أو للعدالة عندما يتطلب الموقف ذلك. علاوة على ذلك، تم تقديم بنية مدعومة بـ STAR-RIS لتعزيز جودة الإشارة وكفاءة الطاقة (EE) من خلال التحكم الذكي في بيئة الانتشار اللاسلكي.

تُظهر نتائج المحاكاة أن الأساليب المقترحة تتفوق بشكل كبير على الطرق المرجعية من حيث كفاءة الطاقة، واستغلال الطيف الترددي، والعدالة. الحل المقترح القائم على DRL، والذي يقوم بتحسين جميع عوامل الشبكة بشكل متزامن، يحقق كفاءة طاقة (EE) أعلى مقارنة بالطرق المرجعية بحوالي 60% و 17.3% لكفاءة الطاقة مع إعطاء أولويات مختلفة لأوزان العدالة، وحوالي 85% في مؤشر العدالة.

الكلمات المفتاحية: الذكاء الاصطناعي، التعلم المعزز العميق، كفاءة الطاقة، إنترنت الأشياء، الأسطح الذكية القابلة لإعادة التشكيل، شبكات المركبات الجوية بدون طيار.

Résumé

La prolifération rapide des appareils de l'Internet des Objets (IoT) et la demande croissante pour une communication sans fil fiable et économe en énergie ont motivé l'exploration de technologies d'accès avancées et de mécanismes de contrôle intelligents. Les Véhicules Aériens Sans Pilote (UAVs), avec leur déploiement flexible et leur connectivité en visibilité directe (line-of-sight), sont apparus comme un catalyseur clé pour étendre la couverture réseau dans les zones difficiles d'accès. Dans ce contexte, cette thèse examine comment la combinaison de l'Accès Multiple Non Orthogonal (NOMA), du Deep Reinforcement Learning (DRL), et des Surfaces Intelligentes Reconfigurables Transmettant et Réfléchissant Simultanément (STAR-RIS) peut améliorer l'efficacité des réseaux de communication IoT assistés par UAV.

L'objectif principal est de concevoir un framework basé sur l'IA qui optimise conjointement la trajectoire de l'UAV, l'ordonnancement des utilisateurs (user scheduling), et l'allocation des ressources pour maximiser l'efficacité énergétique (energy efficiency - EE) et assurer une distribution équitable des ressources. Un algorithme basé sur DRL est proposé pour permettre une prise de décision autonome dans des environnements dynamiques et complexes, en utilisant une fonction de récompense pondérée qui équilibre l'efficacité énergétique et l'équité et facilite la priorisation de l'énergie lorsque nécessaire ou de l'équité lorsque la situation l'exige. De plus, une architecture assistée par STAR-RIS est introduite afin d'améliorer la qualité du signal et l'EE grâce à la manipulation intelligente de l'environnement de propagation sans fil.

Les résultats de simulation démontrent que les approches proposées surpassent significativement les méthodes de référence en termes d'efficacité énergétique, d'utilisation spectrale et d'équité. La solution proposée basée sur DRL, qui optimise tous les facteurs d'optimisation simultanément, atteint une EE plus élevée par rapport aux méthodes de référence d'environ 60% et 17.3% pour l'EE avec différentes pondérations d'équité prioritaires et environ 85% pour l'indice d'équité.

Mots-clés: Intelligence artificielle, deep reinforcement learning, efficacité énergétique, IoT, NOMA, surfaces intelligentes reconfigurables, réseaux UAV.

Contents

List of Figures

List of Tables

Glossary

A2A Air-to-Air

A2G Air-to-Ground

ABS Aerial Base Station

AI Artificial Intelligence

ANN Artificial Neural Network

BCE Binary Cross Entropy

BS Base Station

BW Bandwidth

CCE Categorical Cross Entropy

CDMA Code Division Multiple Access

CD-NOMA Code Domain Non-Orthogonal Multiple Access

CNN Convolutional Neural Network

CSI Channel State Information

DDPG Deep Deterministic Policy Gradient

DL Deep Learning

DNN Deep Neural Network

DoF Degrees of Freedom

DQN Deep Q-Network

DRL Deep Reinforcement Learning

EE Energy Efficiency

EM Electromagnetic

ES Energy Splitting

FDMA Frequency Division Multiple Access

GBS Ground Base Station

GD Gradient Descent

GNN Graph Neural Network

HAP High-Altitude Platform

HDF Hybrid Decision Framework

IIoT Industrial Internet of Things

IoT Internet of Things

ISI Inter-Symbol Interference

JFI Jain's Fairness Index

KPI Key Performance Indicator

LAP Low-Altitude Platform

LDS-CDMA Low-Density Signature Code Division Multiple Access

LLR Log-Likelihood Ratio

LoS Line-of-Sight

MA Multiple Access

MAE Mean Absolute Error

MDP Markov Decision Process

MFIS Mamdani Fuzzy Inference System

MINLP Mixed-Integer Nonlinear Programming

ML Machine Learning

MLP Multi-Layer Perceptron

mMTC massive Machine-Type Communications

MMSE-SIC Minimum Mean Square Error - Successive Interference Cancellation

mmWave Millimeter Wave

MPA Message Passing Algorithm

MS Mode Switching

MSE Mean Squared Error

MUSA Multi-User Shared Access

NGMA Next Generation Multiple Access

NLoS Non-Line-of-Sight

NOMA Non-Orthogonal Multiple Access

NN Neural Network

OFDM Orthogonal Frequency Division Multiplexing

OFDMA Orthogonal Frequency Division Multiple Access

OMA Orthogonal Multiple Access

PA Power Allocation

PAPR Peak-to-Average Power Ratio

PD-NOMA Power Domain Non-Orthogonal Multiple Access

PIC Parallel Interference Cancellation

PPO Proximal Policy Optimization

QoS Quality of Service

ReLU Rectified Linear Unit

RL Reinforcement Learning

RIS Reconfigurable Intelligent Surface

RNN Recurrent Neural Network

ROC AUC Receiver Operating Characteristic Area Under the Curve

RSMA Rate-Splitting Multiple Access

Rx Receiver

SC Superposition Coding

SC-FDMA Single Carrier Frequency Division Multiple Access

SCMA Sparse Code Multiple Access

SDMA Spatial Division Multiple Access

SDN Software-Defined Networking

SE Spectral Efficiency

SGD Stochastic Gradient Descent

SIC Successive Interference Cancellation

SINR Signal-to-Interference-plus-Noise Ratio

SR Sum Rate

STAR-RIS Simultaneously Transmitting and Reflecting Reconfigurable Intelligent Surface

SVM Support Vector Machine

TDMA Time Division Multiple Access

THz Terahertz

TS Time Switching

Tx Transmitter

UAV Unmanned Aerial Vehicle

UE User Equipment

URLLC Ultra-Reliable Low-Latency Communications

UPA Uniform Planar Array

VPG Vanilla Policy Gradient

General Introduction

The Internet of Things (IoT) has catalyzed a transformative shift in wireless communication, connecting billions of low-power devices across applications ranging from smart cities to industrial automation. IoT networks face stringent energy constraints due to battery-limited sensors (e.g., $\leq$ 10 mW transmit power), while supporting massive connectivity (e.g., 1 million $devices/km^2$ in 6G). As IoT deployments scale, conventional network architectures struggle to meet the conflicting demands of energy efficiency, reliability, and massive connectivity—particularly in environments where terrestrial infrastructure is unavailable or damaged. Unmanned Aerial Vehicles (UAVs) have emerged as agile, reconfigurable aerial platforms to bridge this gap, dynamically extending coverage to underserved areas. However, UAV-assisted networks inherit unique challenges: the mobility of the UAVs introduces rapidly changing channel conditions, the limited onboard energy of the UAVs prioritizes efficient propulsion over prolonged operation, and traditional spectrum-sharing methods like Orthogonal Multiple Access (OMA) waste bandwidth in dense deployments.

Non-Orthogonal Multiple Access (NOMA) addresses the inefficiency of OMA by allowing multiple devices to share the same frequency and time resources through power-domain multiplexing. In UAV-NOMA systems, devices with stronger channel conditions decode their signals first while canceling the interference from the transmissions of the users with weaker conditions, improving spectral efficiency. However, this approach demands precise real-time power allocation and decoding-order adjustments to balance fairness and performance—a task complicated by the mobility of the UAV, which shifts channel strengths unpredictably. Reconfigurable Intelligent Surfaces (RIS), and their advanced variants (STAR-RIS), further augment UAV networks by intelligently steering signals via software-controlled metamaterials. RIS panels reflect incident waves to fill coverage holes, while STAR-RIS can simultaneously transmit and reflect signals, enabling full-space coverage. Yet, coordinating the trajectory of the UAV, the beamforming of the RIS, and NOMA parameters in real time remains a multidimensional optimization problem.

Recent advances in Artificial Intelligence (AI), particularly Deep Reinforcement Learning (DRL), provide a pathway to untangle these complexities. DRL agents learn optimal policies by continuously interacting with the network environment, enabling joint optimization of flight paths of the UAV, phase configurations of the RIS, and power levels of NOMA without relying on predefined models. For instance, a DRL framework might prioritize energy conservation by reducing the mobility of the UAV during stable channel conditions or dynamically cluster devices into NOMA groups based on real-time interference patterns.

Despite the potential of UAV-assisted NOMA networks and AI-driven optimization, critical challenges persist. Energy efficiency remains a pressing concern due to the limited battery life of the UAVs, necessitating algorithms that balance performance with power constraints. Optimal resource allocation in dynamic UAV-IoT environments requires strategies to ensure fairness among users while managing spectrum and power. The integration of STAR-RIS with UAV-NOMA networks, though promising for enhancing coverage and efficiency, demands novel modeling and optimization frameworks. Additionally, traditional optimization techniques struggle with the high dimensionality of UAV networks, highlighting the need for DRL-based approaches to enable real-time, adaptive control. Addressing these challenges is essential to unlock the full potential of next-generation IoT networks.

This thesis aims to optimize energy-efficient UAV-assisted NOMA communication through AI-driven optimization. The objectives include analyzing the impact of NOMA on spectral efficiency and fairness, designing DRL-based algorithms for power distribution and the optimization of the trajectory of the UAV, integrating STAR-RIS into UAV-NOMA networks, and evaluating performance against conventional methods. Key contributions include a novel DRL framework for dynamic resource allocation, a STAR-RIS-assisted UAV-NOMA model, energy-efficient algorithms balancing fairness and efficiency, and comparative analyses demonstrating the superiority of the proposed solutions over existing methods.

The remainder of this thesis is organized as follows:

- The first chapter traces the progression of multiple access strategies and begins with Orthogonal Multiple Access (OMA) schemes such as FDMA and TDMA, which prioritize interference-free communication at the cost of spectral efficiency. It then explores non-orthogonal paradigms, including Non-Orthogonal Multiple Access (NOMA)—which leverages power-domain multiplexing and successive interference cancellation (SIC) decoding—and Rate-Splitting Multiple Access (RSMA),

which uses partial message decoding for robustness to imperfect channel state information (CSI). The chapter focuses on different NOMA schemes and emphasizes their technical distinctions, from design principles to receiver architectures.

- The second chapter introduces core artificial intelligence (AI) methodologies. It covers machine learning (ML) with fundamental concepts and algorithms, deep learning (DL) with core architectures (e.g., CNNs, RNNs), and reinforcement learning (RL) principles. The Markov Decision Process (MDP) framework is formalized and details its components (states, actions, rewards, policies), with a focus on deep reinforcement learning (DRL) for solving dynamic optimization problems.

- Chapter 3 discusses IoT communications and the requirements of IoT systems, UAV types, energy consumption models of UAVs, operational modes in wireless communications, and channel modeling of UAVs. The chapter proposes a unified framework for integrating NOMA, UAVs, and Reconfigurable Intelligent Surfaces (RIS) in IoT networks. It addresses the energy-constrained mobility of the UAVs and dynamic air-to-ground (A2G) channels through the coupling of the spectral efficiency of NOMA with signal propagation enhanced by the RIS. A DRL-based architecture jointly optimizes the trajectory (3D positioning) of the UAV, the phase shifts (beamforming) of the RIS, and the power allocation of NOMA. The framework accommodates heterogeneous use cases of IoT, from delay-sensitive industrial automation to sparse environmental monitoring, and balances energy efficiency, fairness, and interference mitigation.

- Chapter 4 validates the proposed AI-driven framework through extensive simulations. It first introduces the framework design and algorithm flow, then benchmarks performance against OMA, static UAV deployments, and baseline RIS configurations. Key metrics include energy efficiency (bits/Joule), sum-rate (bps), spectral efficiency (bits/s/Hz), and fairness index.

Chapter 1

Multiple Access Techniques: Evolution and Current State

The evolution of wireless communications has progressed continuously from the 1980s to the present day. Beginning with the first generation (1G) in the 1980s, technology advanced to 2G in the 1990s, then to 3G in the 2000s, followed by 4G in the 2010s. In the 2020s, 5G began to reshape wireless networks, and ongoing research continues to optimize both 5G and beyond-5G (B5G) networks. Recently, 6G mobile communications is attracting significant attention from academia as well as major companies in the industry, holding the potential to bring transformative changes. The development of each generation has been affected by multiple factors, among the most critical of which are Multiple access (MA) techniques.

1.1 Revolution of Multiple Access Techniques

Multiple access technologies play a pivotal role in wireless communication networks, a fact that is clearly reflected in the development of these networks from the 1G to the 5G and B5G. Even with the ongoing development of communication networks for the 6G, MA technologies continue to evolve and advance. In the early 1980s, the 1G introduced the FDMA technology. This was followed by the 2G in the early 1990s, which make use of TDMA, and then the 3G networks in the early 2000s, which implemented CDMA. Moving into the 2nd decade of the 21st century, the 4G networks began to exploit OFDMA technology. About 10 years ago, NOMA technology emerged as a leading candidate for 5G networks and B5G. More recently, RSMA technology has been nominated as a prominent candidate for future wireless communication networks. As one can see, MA techniques have a fundamental role in the evolution of mobile cellular

communications because each generation has unique requirements at the physical layer and radio resource usage, making MA techniques important for managing these resources efficiently to meet the performance goals of each generation.

1.2 Orthogonal Multiple Access (OMA)

Previous generations of cellular mobile networks have been based on the orthogonal allocation of resources, including techniques such as FDMA, TDMA, CDMA, and OFDMA. In this section, OMA techniques are presented.

1.2.1 Frequency Division Multiple Access (FDMA)

The 1G of cellular networks permit users to make voice calls and was built on analog radio transmission technologies. As discussed, OMA techniques allow multiple users to communicate simultaneously where it allocates radio resources orthogonally. FDMA works by dividing the total bandwidth among users with each user allocated a portion of the total bandwidth for the duration of their communication. The total available bandwidth in 1G is $10MHz$, divided into channels of $30kHz$ each, with guard bands between them to prevent interference. However, the limited available bandwidth creates a challenge to support all users as their numbers increase.

1.2.2 Time Division Multiple Access (TDMA)

With the need to support more traffic within a limited radio spectrum, the industry introduced TDMA, a digital wireless technology. This advancement allow for 2G (GSM, GPRS, EDGE) to come to the surface, which uses a time-sharing protocol to offer three to four times the capacity of the 1G analog systems. TDMA technology operates by dividing time resources among users, differing in this respect from FDMA, which relies on spectrum division. The major difference between the two technologies is that TDMA requires precise time synchronization to prevent Multiple Access Interference (MAI) [**1**]. In TDMA, time is divided into time slots with short intervals to avoid interference between them. In this tehcnique, each user utilizes the entire spectrum resource during a specific time period [**2**].

1.2.3 Code Division Multiple Access (CDMA)

Radio resources like time and frequency are limited, so CDMA was utilized to overcome this by exploiting the code domain to allow all users to transmit their data simultaneously within the same frequency band and at the same time. In CDMA, user bits are directly multiplied by a sequence of 'chips', where each chip has a shorter duration than the original message bit and this results in a higher frequency for the sequence compared to the original message, which makes CDMA known as a spectrum spreading technique. As a condition, each transmitter is assigned a unique sequence code and the intended receiver must have a replica of that code to decode the data [**3**]. Furthermore, The CDMA sequences are chosen to be orthogonal and are typically generated using pseudo-random numbers (PNs). Many techniques exist for designing such PNs, including Hadamard matrices and more advanced methods [**4, 5**].

1.2.4 OFDMA and Single Carrier FDMA

In Release 7 of the 3GPP standard, the 3GPP stated the requirements for 4G LTE, which are: increased user throughput, enhanced spectral efficiency, and scalable bandwidth to allow greater flexibility in frequency allocation... etc[**6**]. These requirements led the organization to adopt OFDMA for the downlink air interface and SC-FDMA for the uplink. In this subsection, we will explain the rationale behind this selection, the advantages it provides, and the differences between these two multiple access techniques.

1.2.4.1 OFDM and OFDMA

OFDMA is a MA technique that combines orthogonal frequency division multiplexing (OFDM) with FDMA. In OFDMA, OFDM is used to modulate data symbols onto narrowband subcarriers and transmit them in parallel. This parallel transmission enables high data rates and efficient bandwidth usage by splitting the input data stream into substreams. Each substream is transmitted on orthogonal subcarriers to convert a frequency-selective channel into a set of frequency-flat subchannels with partially overlapping spectra and increases robustness against multipath distortions [**7**]. Additionally, OFDM uses a Cyclic Prefix (CP), which is typically a repetition of the last samples of the OFDM symbol appended to the beginning of the data stream, to reduce multipath effects and eliminate intersymbol interference (ISI) as long as the CP duration overstep the channel delay spread [**7**–**9**]. The implementation of OFDM relies on the inverse fast Fourier transform (IFFT) enabling a large number of subcarriers, up to 2048 [**8**]. OFDM, however, is a single-user technique, as it only supports one user at a time by splitting its

input data stream. Since cellular mobile communications require a MA technique that allows multiple users to share the same spectrum, OFDMA was adopted for 4G LTE downlink [**6**]. The key difference between OFDM and OFDMA is that OFDM is a single user technique, while OFDMA allows multiple users to share the same spectrum by allocating the narrowband subcarriers to distincts users [**7**]. Furthermore, OFDMA can improve throughput by allocating subcarriers to users based on their channel responses and Quality of Service (QoS) needs. However, OFDMA suffers from multiple issues, including a high PAPR due to multicarrier modulation which necessitates highly linear power amplifiers to avoid excessive intermodulation distortion leading to lower power efficiency, the inevitable frequency reference offset among different users transmitting simultaneously in the uplink which introduces MAI, and the critical need for frequency synchronization to avoid the intercarrier interference (ICI) [**7, 8, 10**].

1.2.4.2 SC-FDMA

The most critical drawback of OFDMA is its high PAPR, which is challenging for mobile users with limited battery life. To cope with this, SC-FDMA is preferred for the uplink due to its improved power efficiency. SC-FDMA has the main principles of OFDMA but includes an additional discrete Fourier transform (DFT) spreading operation, reducing the PAPR of the transmitted signal, thereby enhancing power amplifier efficiency and extending battery life for mobile devices [**9, 10**]. However, this benefit comes at the cost of increased receiver complexity, though this increased complexity can be managed effectively at the base station [**10**].

1.3 Non-Orthogonal Multiple Access (NOMA)

Even though OMA has been used in earlier generations of cellular networks and has evolved over the past decades, even being utilized in the air interface of 5G NR, it has its limitations since 5G introduces many requirements based on specific use cases. The 3GPP has defined three primary use cases in 5G networks and future 6G networks forcasted, which potentialy includes [**11**]:

1. Massive Enhanced Mobile Broadband (meMBB) and Mobile Broadband (MBB): These are aimed at the provision of high data rates, large bandwidth, and a high density of connections, especially in scenarios of dense communication.
2. Mobile Broadband Reliable and Low Latency Communication (MBRLLC): Designed for applications that require high data rates, large bandwidth, low latency, and high reliability, such as wireless data centers.

3. Massive Ultra-Reliable Low Latency Communication (muRLLC): This use case needs low latency, high reliability, and massive scalability in establishing connections for machine-type communications in specific scenarios (e.g., smart transportation, smart factories, and industrial IoT).

Other than these use cases, 5G and beyond networks are expected to provide the following key functionalities [**12**]:

1. Large Network Capacity: Guarantees the ability to accommodate a huge amount of data and users at once in the network.
2. High User Density: Support many users in dense scenarios without degrading performance.
3. User fairness: Quality of service for different users should be the same in different geographical areas and conditions.
4. Multi-User Multiple Input Multiple Output: Supporting multiple users at a time for more spectral-efficient systems.
5. Mixed Traffic Types Transmission: Transmission of multiple data types, e.g., voice, video, and IoT-type data.
6. Highly Efficient Small Packets Transmission: Efficient transmission of small packets which is important for IoT and machine-type communications.
7. Ultra-Low Latency Transmission: Introduce minimal delay in transmission to support real-time applications.
8. Ultra-High Reliability Transmission: guarantee high reliability for applications where failure must completly be avoided, for example: industrial automation, and healthcare monitoring.
9. Massive Connectivity: Support a large number of connected devices and users.

1.3.1 NOMA Overview

OMA is known for preventing interference and by its simple receivers design by allocating resources orthogonally among UEs or IoT devices. However, by using OMA the spectral efficiency is significantly limited. With the introduction of multiple use cases in 5G and B5G, different multiple access techniques are required to meet the specific performance requirements of each use case. Thus, NOMA techniques have appeared as a leading solution for 5G and B5G. NOMA allows multiple users to share the same

radio resources; i.e., time, frequency, or code; to improve spectral and energy efficiency. Furthermore, NOMA enables massive connectivity and fairness, which are key features of 5G and B5G especially in IoT networks, where network density increases substantially. For these reasons, NOMA stands out as a leading multiple access technique for 5G and B5G. Table 1.1 presents a comparison between NOMA and OMA [**13**].

Table 1.1: Comparison of NOMA and OMA.

	Advantages	**Disadvantages**
OMA	Reduced complexity in receiver design	Inefficient use of available bandwidth
		Number of users is limited
		Unfair resource allocation
NOMA	Increased spectral efficiency	Complexity of multi-user detection
	Massive connectivity	Higher sensitivity to channel uncertainty
	Fairness among users	
	Reduced Latency	
	Supports different QoS requirements	

1.3.2 NOMA Dominant Schemes

NOMA is categorized into two main groups: Power Domain NOMA and Code Domain NOMA. In this subsection, we will explore the dominant NOMA schemes in detail.

1.3.2.1 Power Domain NOMA

In PD-NOMA, users or IoT devices are distinguished based on their power levels, while radio resources (i.e., time and frequency) are shared among all users. The base station allocates a lower power level to users closer to it and a higher power level to users farther away to enable multi-user detection (MUD) at the receiver [**14**]. The main principle of PD-NOMA is shown in Figure 1.1. In the downlink PD-NOMA, the signals of multiple users are first coded and modulated in the classical way, these signals are then superimposed at the transmitter using a technique known as Superposition Coding (SC). The composite signal is transmitted to the corresponding users, who decode it using a complex MUD receivers. While NOMA offers superior system performance compared to OMA, this advantage comes at the cost of increased receiver complexity.

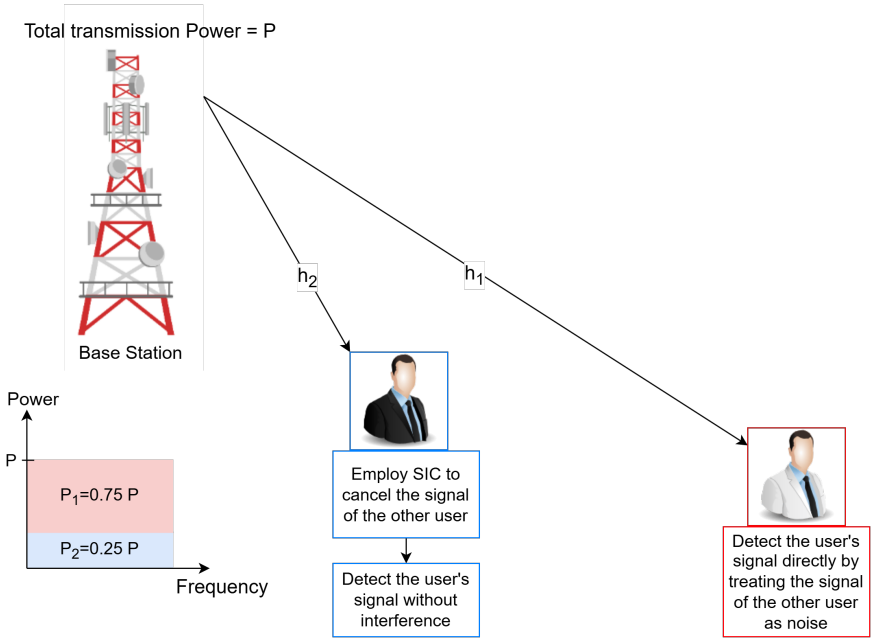


FIGURE 1.1: Principle of Power Domain NOMA.

1.3.2.1.1 Superposition Coding One of the techniques used to implement PD-NOMA is SC [**15**]. In the downlink, when SC is applied, the transmitter employs two point-to-point encoders to map the inputs of two users (considered for simplicity) into complex sequences. Furthermore, the constellation of the user with a poorer channel condition, whose signal is denoted as $x_1(t)$, is allocated a higher amount of power and superimposed on the constellation of the user with a better channel condition, whose signal, $x_2(t)$, is assigned a lower amount of power [**15**]. The principle of SC is illustrated in Figure 1.2 [**15**]. The superimposed signal at the transmitter is given by:

$$x(t) = \sqrt{p_1}x_1(t) + \sqrt{p_2}x_2(t) \tag{1.1}$$

Here, p_1 and p_2 are the power levels of $x_1(t)$ and $x_2(t)$ respectively.

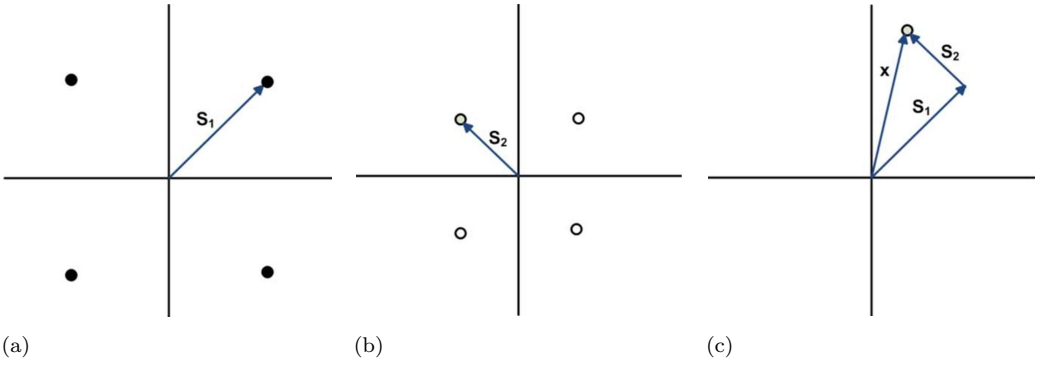


FIGURE 1.2: SC encoding (a) user 1 constellation. (b) user 2 constellation. (c) constellation after SC.

1.3.2.1.2 Downlink PD-NOMA In downlink PD-NOMA, the signal transmitted from the base station is superimposed using SC. Generalizing the equation in 1.1, the expression for the superimposed signal at the transmitter for U users becomes [**15**]:

$$x(t) = \sum_{u=1}^{U} \sqrt{p_u} x_u(t) \tag{1.2}$$

Here, p_u is the power allocated to user u, and $x_u(t)$ is the modulated signal of user u, with the constraint that $\sum_{u=1}^{U} p_u = 1$. The superimposed signal $x(t)$ is transmitted to users, who decode it using a MUD receiver, such as successive interference cancellation (SIC), which is introduced in Section 1.3.3.1. The received signal at user u is given as:

$$y_u(t) = h_u(t)\sqrt{p_u} x(t) + n_u(t) \tag{1.3}$$

where $h_u(t)$ is the channel gain of user u, and $n_u(t)$ is the noise and inter-cell interference.

1.3.2.1.3 Uplink PD-NOMA In uplink PD-NOMA, the signal received at the base station is expressed as [**15**]:

$$y(t) = \sum_{u=1}^{U} \sqrt{p_u} h_u(t) x_u(t) + n(t) \tag{1.4}$$

Here, $h_u(t)$ is the channel gain of user u, p_u is the transmitted power of user u, and $n(t)$ is the noise and inter-cell interference at the base station. The SIC receiver is similarly used to decode the received signal at the base station.

1.3.2.1.4 Cluster-Based PD-NOMA The SIC receiver performs ordered detection from the user with the poorer channel to the user with the better channel conditions. However, this process increases system latency, which must be very reduced in 5G and B5G networks. To address this challenge, cluster-based PD-NOMA, also known as multi-carrier NOMA, is introduced [**16**]. The idea behind cluster-based is to divide the available spectrum into several sub-bands, each one of them contains a small number of users, and by doing so the latency is significantly decreased, since the SIC receiver requires fewer iterations compared to classical PD-NOMA, which also effectively mitigates propagation errors in SIC.

1.3.2.2 Code Domain NOMA

As discussed in the previous subsection, PD-NOMA exploits the power domain to enable multiple users to share the same time and frequency resources. The second category of NOMA is essentially based on CDMA principle. While CDMA uses orthogonal codes to separate users, CD-NOMA relies on sparse and non-orthogonal sequences with low cross-correlation to enhance spectral efficiency and support massive connectivity. There are many CD-NOMA schemes have been proposed, including LDS-CDMA, LDS-OFDM, MUSA, SCMA, ...etc. In this subsection, we will provide an in-depth discussion of the dominant code-domain NOMA schemes.

1.3.2.2.1 Low-Density Signature CDMA (LDS-CDMA) One of the first non-orthogonal schemes proposed is LDS-CDMA, which is directly related to the CDMA but uses low-density signatures, characterized by a small number of non-zero elements in the sequence [**17**]. In this scheme, each user spreads its data on a reduced number of "chips" to improve spectral efficiency and allow for more manageable inter-user interference. A visual comparison between LDS-CDMA and CDMA is illustrated in Figure 1.3 [**18**].

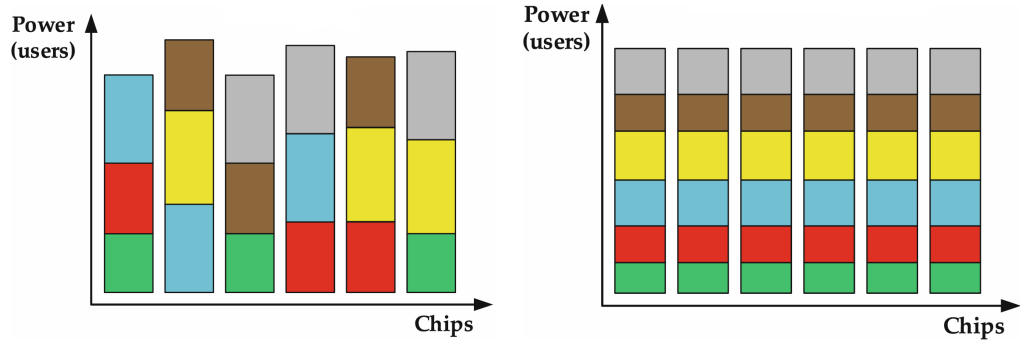


FIGURE 1.3: Resources allocation: LDS-CDMA vs CDMA (a) LDS-CDMA. (b) CDMA.

1.3.2.2.2 LDS-OFDM LDS-OFDM is a NOMA scheme that integrates OFDM and LDS, operating on a principle distinct from LDS-CDMA. In LDS-CDMA, each user spreads their data across multiple chips, whereas in LDS-OFDM, each user's data is transmitted over a subset of the available subcarriers. By using LDS-OFDM, the total system capacity can be overloaded by up to 400% compared to that of the orthogonal technique OFDMA [**19**].

1.3.2.2.3 Multi-user Shared Access (MUSA) Multi-user Shared Access (MUSA) is one of the CD-NOMA techniques that combines the principles of CDMA and PD-NOMA and is proposed for use in MTC applications [**20**]. In MUSA, the data from each IoT device is first encoded and then modulated. After modulation, the data is spread using a complex spreading code, then the spread signal is transmitted over the multiple access channel. In MUSA, the set of IoT devices is divided into G groups, with each group assigned a unique spreading code, which is used by all IoT devices in that group. In this manner, inter-group interference is managed through the use of distinct spreading codes, and intra-group interference is handled as in PD-NOMA, where IoT devices within each group are allocated different power levels, while a SIC receiver is used to decode the signals from each IoT device [**20**]. MUSA complex spreading codes benefit from their short length and the flexibility in design afforded by the values of their real and imaginary parts. Additionally, an example of spreading codes with a number of available values of 4 and 9 is shown in Figure 1.4. In this way, the possible spreading sequences that can be generated are 4^{length} and 9^{length}, respectively, where $length$ represents the length of the spreading code [**20**].

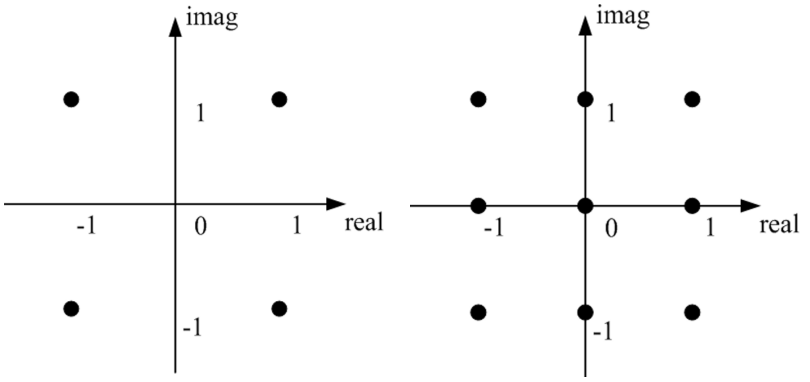
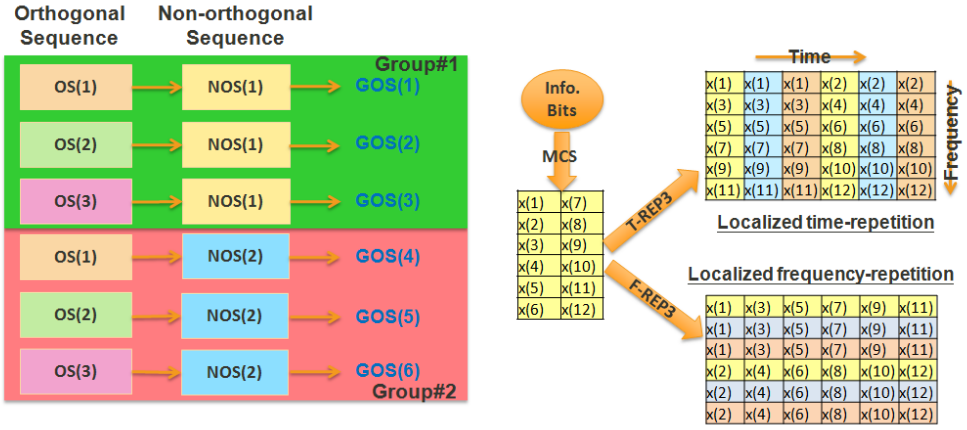


FIGURE 1.4: Spreading code elements (a) 4 complex values . (b) 9 complex values.

1.3.2.2.4 NOCA NOCA [**21**] is a CD-NOMA scheme that uses non-orthogonal sequences, which can be applied in three configurations: in the frequency domain, time domain, or both. These non-orthogonal NOCA spreading sequences are designed to have a low cubic metric, good auto-correlation and cross-correlation properties, and low memory and complexity requirements. In NOCA, each user j is assigned a unique spreading sequence C^j. The modulated signal is converted into parallel sequences and then mapped onto SF subcarriers. Subsequently, the user's unique spreading code is multiplied by the SF, with each factor corresponding to one subcarrier.

1.3.2.2.5 NCMA There is another NOMA scheme that follows the NOCA principle of spreading, known as NCMA. In NCMA, the Grassmannian line packing problem is used to identify spreading codes that maximize the minimum distance between vectors to ensure these codes have low correlation and reduce interference [**22**].

1.3.2.2.6 Group Orthogonal Multiple Access (GOCA) GOCA [**23**] is one of the simplest CD-NOMA technologies, following a principle similar to that of standard CDMA but in a non-orthogonal manner. When using GOCA as a multiple access (MA) technique, a two-stage structure is used to generate group orthogonal sequences, and then spread modulated symbols into shared time and frequency resources. In GOCA, users are divided into groups, and within each group, a set of orthogonal codes is assigned to the users, and then each group is assigned a unique non-orthogonal code. To keep orthogonality between orthogonal sequences in the same group, localized frequency or time-domain repetition is also used in GOCA transmitter. Figure 1.5 illustrates the two-stage structure of GOCA transmitter.



(a) (b)

FIGURE 1.5: GOCA principles (a) Spreading structure. (b) Time/frequency repetition.

1.3.2.2.7 Sparse Code Multiple Access (SCMA) SCMA is one of the earliest NOMA schemes, proposed in 2013 [**24**]. It follows a similar principle to LDS-CDMA but introduces a multi-dimensional mother constellation and the concept of a "codebook", which is a key element of SCMA and is a set of vectors, known as codewords, used to encode and modulate data directly. In SCMA, the bit stream is directly mapped to a complex codeword using the user's unique codebook, which helps reduce interference from other users. After modulation, the codewords of all users are superimposed on multiple OFDM subcarriers, also referred to as resource elements (REs), according to a mapping matrix based on a factor graph. We noted the number of available REs in a

SCMA system as K and the number of users as J, the overloading factor is defined as $\lambda = \frac{J}{K}$. The SCMA transmitter design is illustrated in Figure 1.6.

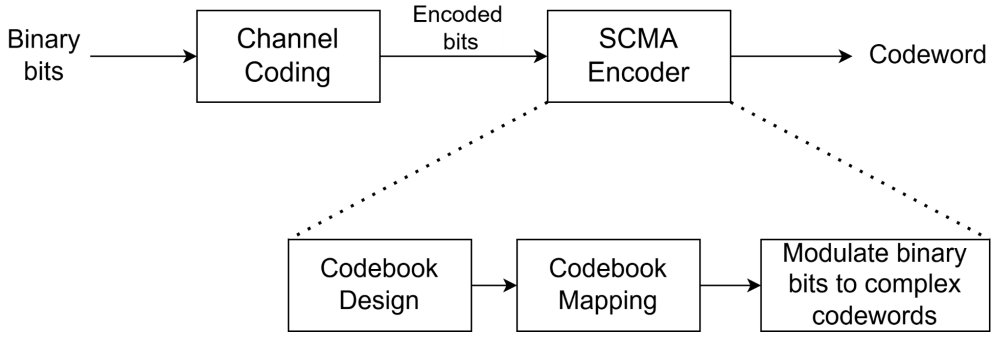


FIGURE 1.6: Illustration of SCMA transmitter.

1.3.2.2.7.1 Mapping matrix and Factor graph SCMA utilizes a number of K REs to serve J users. A mapping matrix is used to determine which users are served by which REs. This matrix is also called a factor graph matrix denoted by $\mathbf{F} \in \mathbb{B}^{K \times J}$. Each row of $\mathbf{F}$ corresponds to a RE $1 \leq k \leq K$, and each column corresponds to a user $1 \leq j \leq J$. If $\mathbf{F}(k, j) = 1$ is present in the matrix, it indicates that the user j is served by the RE k, under the conditions that each user utilizes only d_v REs and a maximum of only d_f users are interfered in every RE. An example of a mapping matrix is given as follows:

$$\mathbf{F}_{4,6} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{1.5}$$

Where $K = 4$, $J = 6$, $\lambda = 150\%$, $d_v = 3$, and $d_f = 2$. To further visualize the mapping matrix in that example, SCMA uses a factor graph, which is a bipartite graph reflecting the mapping between users and REs and consists of two different types of nodes: function nodes (FNs), representing the K subcarriers, and variable nodes (VNs), representing the J active users. The VN j is connected to the FN k when user j uses subcarrier k, i.e., $\mathbf{F}(k, j) = 1$. The factor graph corresponding to the mapping matrix in Equation 1.5 is shown in Figure 1.7.

1.3.2.2.7.2 Codebook design Given the SCMA system model in Figure 1.8, each user must consists of a distinct codebook which is designed based on mother consellation and the mapping matrix of the system. Furthermore, the codebook main purpose is to map the binary data of each user directly to a complex codeword, the codebook contains M codeword this means that each codebook has a dimension of $K \times M$, with which is then transmitted using the assigned REs to that user, user j intends to transmit

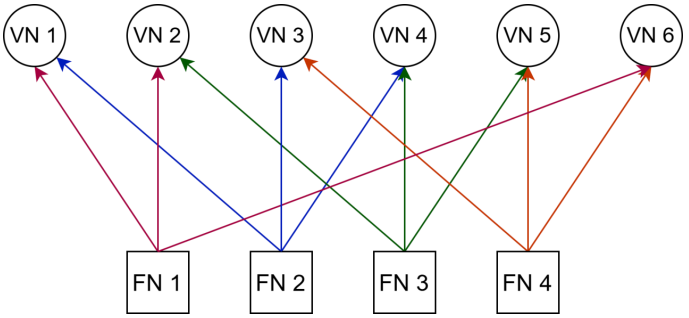


FIGURE 1.7: Illustration of the factor graph for the mapping matrix in Equation 1.5.

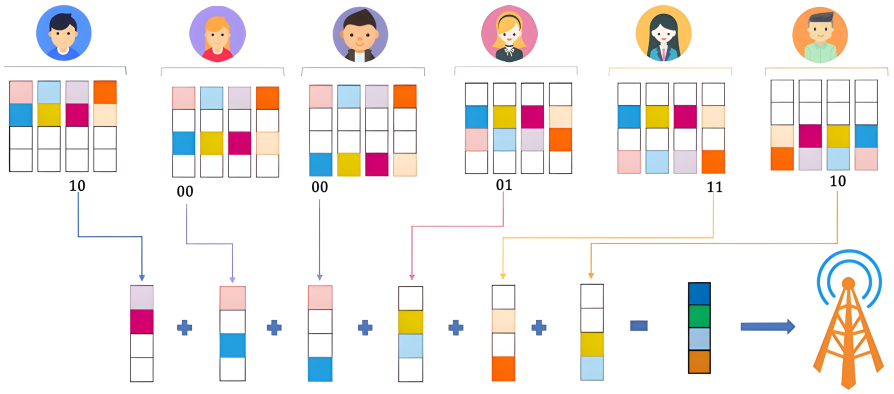


FIGURE 1.8: SCMA system model.

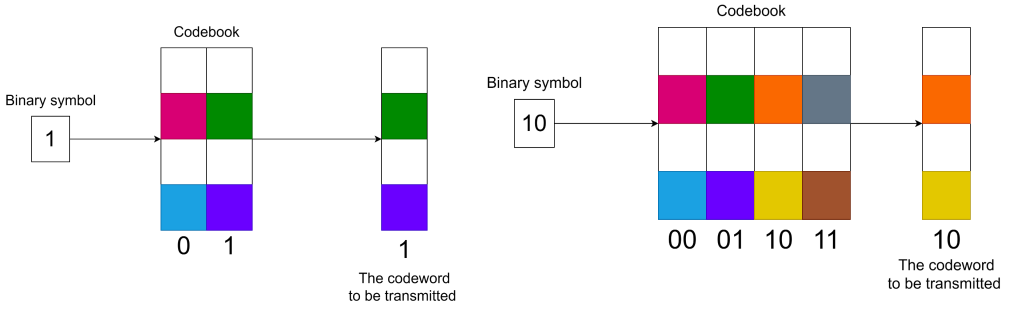
B coded bits b_j $[b_j^1, \dots b_j^B]$, where $Blog_2(M)$ and M is the number of available codewords in the given codebook or may called the modulation order [**25**]. Following the mapping matrix **F** given in Equation 1.5, examples of modulating the binary data directly to a codeword are shown in Figure 1.9. The example in Figure 1.9(a) maps symbols of one bit to a codeword in the case of $M = 2$, while the example in Figure 1.9(b) maps symbols of two bits to a codeword since the $M = 4$ and so on with all possible codewords.

Moreover, the design of the codebook is of great importance in the performance of the whole SCMA system, where its design could significantly impact the BER, outage probability, ... etc.

The design metrics of SCMA codebooks are: the Euclidean distance between any two constellation points in the mother constellation, the average power, product distance, euclidian kissing number, product kissing number, modulation diversity order, number of distinct points, and bit-labeling [**26**]. For further details about mother constellation and multi-dimensional constellations, the interested readers are referred to [**26**].

In SCMA there are 2K-dimensional constellation, where each RE has its own constellation based on a mother constellation. Which necessitates a technique to create multi-dimentional constellations starting from a mother constellation. One of the common approaches to do that, is by using a unitary rotation on the mother constellation

to control dimensional dependency and power variation of the constellation while maintaining the Euclidian distance profile unchanged [**27**]. Therefore, the codebook design is determined by the modulation scheme used and a generator matrix. The modulation scheme could be any known modulation like BPSK, QPSK, QAM, etc. Moreover, the generator matrix is the matrix that determines the rotation of each constellation points for each subconstellation, and the design of the generator matrix will be further discussed [**28**].



(a) (b)

FIGURE 1.9: SCMA mapping examples: Binary data to codeword (a) $M = 2$. (b) $M = 4$.

The generator matrix, denoted by $\mathbf{G}$, is introduced to achieve the sparsity of SCMA codebooks and it follows the matrix $\mathbf{F}$, where the non-zeros elements of $\mathbf{F}$ are non-zeros on $\mathbf{G}$, and zeros are zeros in both matrices. each element of the matrix $\mathbf{G}$, $g_{k,j}$ is a phase rotation factor given by: $\phi_\alpha = e^{j\alpha\Delta}$, here α is an integer between 0 and d_f, and Δ is the angle of rotation between two adjacent phases [**28**]. An example of G given F in Equation 1.5 is as follows:

$$\mathbf{G}_{(4,6)} = \begin{bmatrix} \varphi_0 & \varphi_1 & \varphi_2 & 0 & 0 & 0 \\ \varphi_0 & 0 & 0 & \varphi_2 & \varphi_1 & 0 \\ 0 & \varphi_1 & 0 & \varphi_2 & 0 & \varphi_0 \\ 0 & 0 & \varphi_2 & 0 & \varphi_1 & \varphi_0 \end{bmatrix} \tag{1.6}$$

By observing the matrix $\mathbf{G}$, we can see that there are d_f different phase rotations in each RE to maintain uniquely decodable symbols for users that collide in the same RE.

1.3.2.3 Other NOMA schemes

As many reserachers classified NOMA schemes to PD-NOMA and CD-NOMA, there are other NOMA schemes based on other principles like RSMA, SDMA, PDMA, IGMA, RDMA, LSSA, etc. In the following we will discuss some of these schemes.

1.3.2.3.1 Spatial Division Multiple Access (SDMA) Space Division Multiple Access exploits the unique advantages of exploiting the antenna characteristics providing directivity and may thus enable spatial diversity of users in a wireless communication system. Unlike the traditional time, frequency, and code MA techniques, SDMA enables the definition of independent communication channels for different users, separated in the spatial domain. This relies on array processing that enables the base station to distinguish and handle co-channel users [**29**].

A key enabler of SDMA is beamforming, which exploits the directivity of antenna arrays to amplify signals in the desired direction and attenuate interfering signals. They achieve coherent combining of signals intended for the desired user and destructive combining for the interfering users [**29**]. Beamforming allows multiple users to communicate simultaneously in the same frequency band, therefore, ensures higher frequency reuse factors and sometines make intra-cell frequency reuse possible [**29**].

1.3.2.3.2 Interleave-Grid Multiple Access (IGMA) IGMA is an interleave-based scheme [**30**]. The first step in IGMA is that the coded bits of each user are interleaved using bit-level interleaving, which means that the bits are permuted among themselves. After the bit-level interleaving, the data is modulated using any known modulation scheme. Then, IGMA introduces a concept known as grid mapping, where the modulated symbols are padded with zeros. Following this, a symbol-level interleaving is applied to further permute the symbols. Finally, a carrier modulation is implemented. An example of IGMA grid mapping is shown in Figure 1.10 [**30**], where UE_k represents user k. In this example, $N = 4$ is the length of each symbol, $L = 8$ represents the number of resource elements, and $N_{zeros} = L - N$ is the length of the zero-padding. A mathematical example of the grid mapping process is presented in Equation 1.7, where B_k, S_k, S'_k, and a represent the grid-mapping pattern of user k, the symbol sequence, the symbol sequence after grid mapping, and the interleaving matrix, respectively.

$$B_k = \{4,0,2,0,0,3,0,1\} \xrightarrow{IGMA} a = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$
$$\longrightarrow S'_k = S_k \cdot a = \begin{bmatrix} S_{k,1} & S_{k,2} & S_{k,3} & S_{k,4} \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{1.7}$$

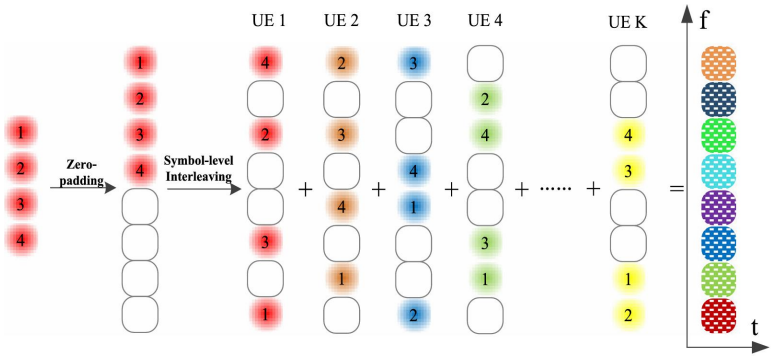


FIGURE 1.10: An example of IGMA grid mapping.

1.3.2.3.3 Resource Spread Multiple Access (RSMA) RSMA [**31**] is a signature-based NOMA technique, where it assigns unique signatures to distinct users. These signatures can be represented by spreading or scrambling codes, interleavers, or their combinations. RSMA has two main variants: single-carrier RSMA and multi-carrier RSMA. On one hand, the single-carrier RSMA variant uses a single carrier to achieve low PAPR, which helps extend battery life. It is primarily used for small data transmissions and allows for grant-free transmission and asynchronous access. On the other hand, the multi-carrier RSMA variant uses multiple carriers to achieve low latency. Similar to single-carrier RSMA, it also supports grant-free transmission, making it suitable for high-speed applications.

1.3.2.3.4 Pattern Division Multiple Access (PDMA) Considering the impact of SIC detection propagation errors and channel fading, recovering the original data becomes impossible when propagation errors occur. To address this issue, a NOMA technique called PDMA was proposed [**32**]. PDMA utilizes a matrix known as "PDMA patterns." As its name suggests, Pattern Division Multiple Access utilizes this matrix

to map each user's data onto a group of available resources, where these resources can include time, frequency, spatial domains, or any combination of them. This mapping approach helps PDMA achieve a higher degree of transmission diversity. PDMA exploits non-orthogonality by allowing multiple users to transmit their data on the same resource simultaneously. However, they are separated using different PDMA patterns. Figure 1.11 shows an example of PDMA mapping with 6 users and 4 resources.

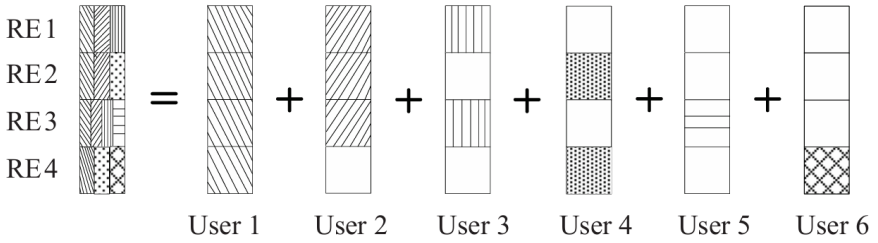


FIGURE 1.11: Illustration of a PDMA mapping.

1.3.2.3.5 Repetition Division Multiple Access (RDMA) RDMA belongs to interleave-based NOMA, which can separate different users' signals just by using a simple cyclic-shift repetition in both time and frequency to exploits two-dimensional diversity [**23**].

1.3.3 NOMA Receivers

Given the number of NOMA techniques and their different methodologies, it is important to choose the appropriate receiver for each technique and it necessitates a careful design and implementation since it impacts excessively the system performance. In this subsection, we aim to provide a comprehensive overview of the different NOMA receivers by discussing their main characteristics.

1.3.3.1 Successive Interference Cancelation (SIC)

SIC-based receivers are widely adopted in NOMA systems due to their mechanism for detecting signals by using the differences in channel conditions among users, which is easily achieved by allocating distinct power levels to different users. SIC operates by exploiting variations in channel strength among the signals of interest. The term "successive" in SIC refers to the sequential decoding of users' signals. Prior to SIC, users are ordered based on their channel strength to enable the receiver to decode the strongest signal first. This signal is then subtracted from the combined signal, allowing the isolation and decoding of the weaker signal [**15**]. After decoding one user's signal,

the receiver removes it from the combined signal before proceeding to decode the next user's signal [**15**]. Regarding the decoding order, the first signal to be decoded treats the other user's signal as interference. However, the subsequent signal is decoded with the advantage of the first signal having already been removed.

For a deeper understanding of SIC, consider Equation 1.3 with $U = 2$. The decoding process is as follows [**15**]:

1. Following the SIC decoding order, the user with the weakest channel is decoded first, in this case, user 1. The SIC receiver is not utilized for this specific user. Instead, direct detection is applied by treating user 2's signal as noise, given that the power allocated to user 1 is higher than that of user 2.

2. Once user 1's signal has been decoded, the SIC receiver subtracts it from the received signal y. Subsequently, user 2's signal is decoded.

1.3.3.2 Message Passing Algorithm (MPA)

MPA is a near-optimal, iterative detection algorithm based on a bipartite factor graph that models the receiver, as illustrated in Figure 1.7, and it is used in graphical models to estimate inference by passing belief messages between nodes, where it operates iteratively in the presence of closed loops, such as in the case of SCMA [**25**]. As detailed in Section 1.3.2.2.7, the graph factor contains two different types of nodes: FN and VN. The VN j is connected to the FN k when user j uses RE k. MPA aims at estimating the transmitted codeword x_j^m, the codeword $0 < m \leq M$ in CB_j, given the received signal $y = \sum_{k=1}^{K} y_k$ observation [**33**]. Furthermore, the number of iterations, denoted as T, in MPA should be determined sufficiently to ensure convergence [**33**]. Each MPA iteration, $0 < t \leq T$, consists of two main steps: passing an information message from FNs to VNs, and passing a message from the VNs to FNs [**25, 33**]. Let η_k and ψ_j be the set of VNs connected with FN k and set of FNs connected with VN j, respectively. Let the messages from FNs to VNs be denoted as $\mu_{k\to j}^t(x_j)$ and the messages from VNs to FNs be denoted as $\mu_{j\to k}^t(x_j)$, $\forall j \in \eta_k$ and $\forall k \in \psi_j$, both for corresponding coderword x_j^m. The MPA is carried out following the following steps [**25, 33**]:

1. Initialization: Initialize the prior probability of each codeword:

$$m_{j\to k}^0(x_j) = P(x_j^m) = \frac{1}{M} \quad \forall j = 1, \ldots, J, \forall k \in \psi_j \tag{1.8}$$

2. The Message Passing between FN and VN:

- Update the message from FN to VN:

$$m_{k\to j}^{t}(x_j^m) = \sum_{\sim x_j} \left(\exp\left\{ -\frac{1}{N_0} \left\| y_k - \sum_{j\in\eta_k} h_{k,j} x_{k,j}^m \right\|^2 \right\} \prod_{j'\in\eta_k\setminus\{j\}} \mu_{j'\to k}^{t-1}(x_{j'}^m) \right) \tag{1.9}$$

  Here, N_0 is the noise power, $h_{k,j}$ is the channel between FN k and VN j, and $x_{k,j}^m$ is the codeword transmitted by FN k to VN j.

- Update the message from VN to FN:

$$\mu_{j\to k}^{t}(x_j^m) = P(x_j) \prod_{k'\in\psi_j\setminus\{k\}} \mu_{k'\to j}^{t-1}(x_j^m). \tag{1.10}$$

3. Termination and bits estimation: After T iterations, the a posterior probability of each codeword is given as:

$$\Pr(\mathbf{x}_j^m) = \frac{1}{M} \prod_{k\in\psi_j(j)} \mu_{k\to j}^{T}(x_j^m) \tag{1.11}$$

   The binary log-likelihood ratios (LLRs) to decide the $\log_2(M)$ bits are:

$$\mathrm{LLR}(b_j^i) = \log \frac{\sum_{x_j^m\in CB_j|b_j^i=+1} \Pr(x_j^m)}{\sum_{x_j^m\in CB_j|b_j^i=-1} \Pr(x_j^m)}. \tag{1.12}$$

   where:

$$\begin{cases} \hat{b}_j^i = 1 & \text{if LLR} \le 0 \\ \hat{b}_j^i = 0 & \text{otherwise} \end{cases} \tag{1.13}$$

1.3.3.3 Parrallel Interference Cancelation (PIC)

PIC [**34**] is an advanced multi-user detection technique for NOMA systems where interference is mitigated, and signal detection is improved. Unlike SIC, which decodes signals sequentially, PIC processes all user signals in parallel, thanks to its design ability to first provide an estimate of all user signals and then iteratively refine these estimates by canceling the interference caused by other users. The PIC iterative process exploits the correlation between user signals and channel conditions to enhance accuracy at each iteration. Therefore, in scenarios where user power levels are similar, PIC is useful in avoiding the sequential dependency of SIC, which decreases decoding latency while maintaining robust performance.

1.3.3.4 Minimum Mean Square Error-SIC (MMSE-SIC)

The MMSE-SIC receiver combines linear MMSE filtering and successive interference cancellation to separate efficiently user signals in UL NOMA systems [**35**]. It basically minimizes the MSE, between the desired signal and the received signal, simply by suppressing the interference of others while at the same time enhancing the signal of the user of interest. It does this through an amplitude-weighted filtering procedure using both channel state information (CSI) and interference statistics. Subsequently, SIC iteratively decodes and cancels stronger user signals from the received signal. This enables the accurate detection of weaker signals. The MMSE-SIC detection process proceeds as follows:

i. The first step is to find the signal with the best quality, which is done using a MMSE receiver that treats leftover interference as random noise. The identified symbol is demapped and channel-decoded to recover the transmitted bitstream.

ii. The reconstructed signal (modulated symbol and channel-encoded bits) is regenerated and removed from the composite received signal to reduce its contribution to the subsequent detection stage.

iii. Steps (i)-(ii) repeat iteratively until either: all transmitted signal streams are successfully decoded or no decodable streams remain (e.g., residual SINR falls below decoding thresholds).

1.3.3.5 Elementary Signal Estimation-PIC (ESE-PIC)

The ESE-PIC detection process begins with the ESE module, which is used as a linear symbol detector to estimate the transmitted symbols. Following symbols detection, they are parallelly deinterleaved and decoded to extract the coding gain. The decoded information is then fed back to the ESE module to improve subsequent symbol detection for more iterative refinement for better performance.

Table 1.2 summarizes the main characteristics of the discussed NOMA receivers.

1.4 Other Multiple Access Techniques

In addition to the discussed methods, several emerging multiple access techniques have been proposed to address the growing demands of next-generation wireless networks. In this section, some of these approaches are briefly outlined, noting their unique features and potential advantages.

Table 1.2: NOMA Receivers Characteristics, Advantages, and Disadvantages

Receiver	**Main Characteristic**	**Advantages**	**Disadvantages**	**Complexity**
SIC	SIC separates signals iteratively by decoding and subtracting interference.	Simple implementation and effective for power-domain multiplexing.	Error propagation can degrade performance.	Moderate
MMSE-SIC	Combines MMSE detection with SIC.	Efficient in mitigating interference and suitable for power-domain NOMA.	Sensitive to error propagation in SIC.	Moderate
PIC	PIC cancels interference across all streams in parallel.	Faster convergence compared to SIC.	Requires high computational resources for accurate estimation.	Moderate to High
ESE-PIC	utilizes ESE with PIC.	Enhances symbol detection iteratively and achieves coding gain.	Performance depends on feedback quality and PIC convergence.	Moderate to High
MPA	Utilizes symbol-level iterative detection based on a factor graph.	Ideal for sparse code-based schemes with high detection accuracy.	Computationally intensive for large user sets and modulation order.	High

1.4.1 Rate-Splitting MA (RSMA)

A key goal of any MA scheme is to efficiently allocate resources and manage interference within a certain wireless communication network. OMA attenuates interference by the allocation of orthogonal radio resources to users, which guarantees no overlapping during spectrum usage. Unfortunately, this orthogonality limits the number of users that can be served in a single radio resource. While on the other hand, SDMA and NOMA counter this problem by allowing multiple users to share the same resource through distinct interference management strategies. On the one hand, SDMA treats residual multi-user interference as noise, which is effective under weak interference conditions. On the other hand, NOMA, based on SIC to decode and remove interference, which is well suited for strong interference levels [**36**]. Despite their respective strengths, both SDMA and NOMA struggle to handle medium interference levels effectively. Research further indicates that when interference is neither weak nor strong, a hybrid approach of partial decoding and partial treatment of interference as noise achieves significant performance gains [**36**]. This is where RSMA introduces its novel interference management approach.

The RSMA approach essentially brings the best of both worlds, fully decoding interference (as in NOMA) and fully treating interference as noise (as in SDMA) by partially decoding some interference while treating the rest as noise. This adaptive capability for interference management fits well into the demands of next-generation wireless networks, where interference levels vary dynamically [**36**]. At its heart is the RSMA-enabled transmitter-a message splitter that divides each user's message into a number of sub-messages. Common messages are built from these sub-messages to be decoded by multiple users; private messages will predominantly contain sub-messages that can only be decoded by a single user [**36**]. The number of sub-messages each user's message is split into varies according to the implemented RSMA scheme. With consideration of common messages and private messages, RSMA achieves greater flexibility in resource allocation and interference management [**36**].

TABLE 1.3: MA schemes summary

MA scheme	**Key technical points**	**Main advantage**	**Receivers**	**Link direction**
PD-NOMA	Exploits power-domain multiplexing	Simple power allocation, flexibility.	SIC	UL/DL
SCMA	Multidimensional modulation	Signal space diversity gain	MPA	UL/DL
MUSA	Short complex spreading sequence	Easy to generate, large number	MMSE-SIC	UL/DL
RSMA	Low cross-correlation Sequence scrambling	Fit for asynchronized scenario	ESE-PIC; MMSE-SIC	UL
PDMA	Irregular LDS	Irrgular protection	SIC; MPA; SIC-MPA	UL/DL
IGMA	bit and symbol-level interleaving	Sparse grid-mapping	SIC; MPA; SIC-MPA	UL/DL
NCMA	Non-orthogonal codewords using Grassmannian line packing	Low PAPR	ESE-PIC; MMSE-SIC	UL
NOCA	Zadoff-Chu sequence	Easy to generate	ESE-PIC; MMSE-SIC	UL
GOCA	Group-based orthogonal/nonorthogonal sequences	Inter-group orthogonality	MMSE-SIC	UL
RDMA	Cyclic shift based time-frequency repetition	Easy implementation	MMSE-SIC	UL
Rate-splitting MA	Partial decoding and interference as noise	Adaptive interference management	SIC	UL/DL
D-OMA	Overlapping NOMA clusters	Efficient resource utilization	SIC	UL/DL
NGMA	SDMA and NOMA integration	Flexible and customized services	ESE-PIC; MMSE-SIC	UL/DL

1.4.2 Delta OMA (D-OMA)

D-OMA is a new proposed MA scheme for future 6G cellular networks designed to provide connectivity at the massive scale [**37**]. It works with clusters of NOMA creating subbands partially overlapping; thus, it enables efficient resource utilization and reduced complexity at terminal devices. D-OMA introduces a variable overlap percentage between the adjacent NOMA clusters, the overlap being a design parameter $\delta\%$, which determines the propotion of overlap. The overlapping increases the number of clusters that can be accommodated within the bandwidth while maintaining high spectral efficiency. The cluster sizes in D-OMA are smaller than those in the traditional, large clusters. This simplification reduces the complexity of the SIC processes and, hence, reduces the power and computational requirements at the terminal. D-OMA design aims to optimize the overlap parameter ($\delta\%$) such that a trade-off between performance and complexity is ensured.

1.4.3 Next Generation MA (NGMA)

The NGMA framework, proposed in [**38**], is designed to meet differing and stringent requirements for 6G networks by integrating multi-antenna techniques and advanced multiple access schemes. Its primary objective is to make full utilization of both SDMA and NOMA for offering highly flexible and customized communication services. Furthermore, This unified framework adapts to varying system conditions in both UL and DL scenarios. In the UL, the detection of information streams is divided into multiple layers, with SDMA applied within each layer and NOMA employed between layers. As a result, signals processed in later layers are free from interference introduced by signals decoded in earlier layers. For DL transmission, multi-antenna techniques are utilized to combine SDMA and NOMA, enabling simultaneous transmission to multiple users while effectively managing interference. Thus, NGMA switches between SDMA and NOMA based on system load, by doing so, the framework exploits SDMA's low latency and simplicity for underloaded scenarios and NOMA's efficient interference management for overloaded conditions.

As a summary, Table 1.3 provides the main characteristics and advantages of the discussed MA techniques.

1.5 Conclusion

In this chapter, we have presented different MA techniques from 1G to 5G and beyond, where both OMA and NOMA were explored, and their principles were emphasized. We then introduced recently proposed MA techniques, such as those designed for 6G networks, which aim to address the growing demands for higher data rates, improved spectral efficiency, massive connectivity, and better interference management.

As the development of AI algorithms grows exponentially, their potential to optimize communication systems is also increasing. The next chapter will introduce the fundamentals of AI, Machine Learning (ML) and Deep Learning (DL), leading to a detailed discussion of Deep Reinforcement Learning (DRL).

Chapter 2

Artificial Intelligence: Concepts and Frameworks

Over the past decade, the development of microchips has contributed to significant advancements in technical fields and made electronic devices capable of performing massive calculations, which provided a suitable environment for emerging from the AI winter that began in the eighties of the last century, when computers were unable to keep up with its development, this progress led to the return of AI to the forefront and its development at a rapid pace.

AI is defined as the ability of a machine to simulate the human brain, to learn from data, extract complex relationships, make decisions, generalize and adapt to new data. This development of AI has made it a tool to assist humans in their tasks, and in some cases, AI has replaced humans in certain jobs. Among the most prominent fields in which AI has brought a revolution are medical care, such as diagnosis and personalized medicine; education, such as adaptive learning systems and explaining complex concepts; autonomous systems, such as self-driving cars; and communication networks, where it improves network performances and optimizes resource utilization.
This chapter thus provides a historical introduction to AI and then discusses ML and its subfields including supervised, unsupervised, and semi-supervised learning, along with their definitions, main differences, and algorithms. Then, the fundamentals of DL and its concepts will be discussed, along with its training mechanisms and various types and techniques, such as DNNs, CNNs, and RNNs. Finally, the chapter introduces RL, starting with Markov Decision Process (MDP) and its components, then moving to DRL by presenting the various DRL kinds and the associated algorithms in details, along with the fundamental differences between them.

2.1 AI Evolution

2.1.1 What's AI?

AI can be defined as the reproduction of human intelligence through machines that carry out operations that usually need human intelligence, such as learning, reasoning, problem-solving, and understanding. AI is also called machine intelligence, unlike the natural intelligence exhibited by humans. With the help of AI, computers can replicate the same cognitive processes as people by utilizing algorithms and computational operations and imitating human intellectual capabilities [**39**].

The two types of AI systems are based on their capabilities, which are narrow AI and general AI. Narrow AI, also called weak AI, comprises specific-use technology that has the ability to solve a particular problem, such as face recognition, language translation, or playing chess. Applications like Siri and Alexa, the most used AI today, fit into this narrow AI category [**40**]. On the other hand, general AI is a hypothetical system that can grasp, learn, and apply intelligence to a variety of tasks as humans do [**40**]. Even though developing AI technology has been the main goal of many researchers for a long time, it has not yet reached a successful conclusion and remains a marker of ongoing studies and queries of the day.

AI systems can learn from data, and thus their performance level improves over time without being explicitly programmed for every task. This form of learning consists of (1) supervised learning, (2) unsupervised learning, and (3) reinforcement learning. Furthermore, an AI's capacity for using human-like reasoning through data processing goes beyond simply depending on the application of basic data rules or probabilistic models to make decisions. In addition to its capability to analyze complicated problems, AI is also able to generate solutions using algorithms, heuristics, and optimization techniques; thus, it is a possible option for logistics, resource allocation, as well as strategic planning.

AI is changing industries and everyday life. In the field of healthcare, AI is used in diagnostics, personalized medicine, and patient management. In finance, AI is the engine that drives detection systems for fraud, platforms that use algorithms to trade, and calculators of risk, facilitating a faster and more precise way of making decisions. In transport, self-driving cars and traffic systems that use AI increase the level of safety, the system's efficiency, and a person's routing capability.

2.1.2 AI History

Formally the 1950s to the 1970s were considered the foundational period of AI as a field of study. During this period pioneering algorithms and principles were very much conceived and they were important enough to be used in the future AI research. Notable figures like Alan Turing, John McCarthy, and Marvin Minsky made a significant contribution to the theoretical aspects of AI. A milestone event was the 1956 Dartmouth Conference, which was perceived as the first formal step of AI as a distinct discipline [**40, 41**]. The first AI programs like the Logic Theorist and the General Problem Solver gave evidence as to the potential of machines to simulate human problem-solving capabilities. All these achievements inspired optimism and paved the way for further participation in AI exploration [**40**].

The late 1970s signaled the beginning of the decrease of initial passion towards AI. The incapacity of the early systems became evident, and so did the inability of researchers to scale up their methodologies. The fact that the anticipated technology did not come into reality was the greatest disappointment to the advocates of AI. The Lighthill Report of 1973 delivered a major blow to AI's prospects [**40, 41**]. This report pointed out that achieving human-like intelligence was impossible. This defeatist attitude was reinforced by an equally sharp reduction in the funding from the two potential sources that had previously supported AI, i.e., the government and the private sector, thus leading to the slowdown referred to as "AI winter" [**41**]. This period saw the decline of interest in AI research, and advancement was greatly slowed down.

The decades from 1970 to 1990 saw a shift towards symbolic AI, knowledge facts and inference rules superseded the AI of the previous era and the focus of research was shifted to the software development of intelligent systems or so called expert systems [**40**]. These systems were capable of executing tasks directed by sets of professional rules, applying simulations, and also performing automated reasoning which made them function like humans [**40**]. The problems they initially solved were rather challenging and specified ones, like medical diagnostics and chemical analysis for example. This period also made an important acquisition in knowledge representation and automated reasoning and thereby became central to AI research. However, expert systems were limited in their ability to generalize beyond their domain of knowldge.

It became more apparent in the 1990s that expert systems were limited in functionality, leading once again to a decline in interest from the scientific community and insufficient investments [**40, 41**]. The systems' key limitation was their inability to replicate human-like intelligence in broader and more complex scenarios. The combination of low performance outcomes, the inability to generalize, and significant disappointment

with these systems led to the second AI winter. During this period, developers focused on more specialized solutions, but areas like training algorithms, data provision, and system optimization were not prioritized, thus remaining untouched. The gradual failure of the systems highlighted the fact that new methods were needed to tackle the issues of the old approach namely the symbolic one.

From the 1990s to the 2020s (modern era of AI), AI research underwent a transformative shift towards ML and DL. In contrast to symbolic AI systems, which follow rules, ML algorithms learned patterns directly from data, as illustrated in Figure 2.1 [**40**]. It was during this period that the rise of deep learning with its sufficiently layered neural networks to process a significant amount of data. Advances in the fields of image and speech recognition, natural language processing, and autonomous systems were sufficient to demonstrate the efficacy of these techniques. The advent of big data and an increase in computing power have further accelerated things, making it practical to train larger and more complex models and turning AI tools into various industries. The shift from these paradigms has placed AI within essential components of technological advancement.

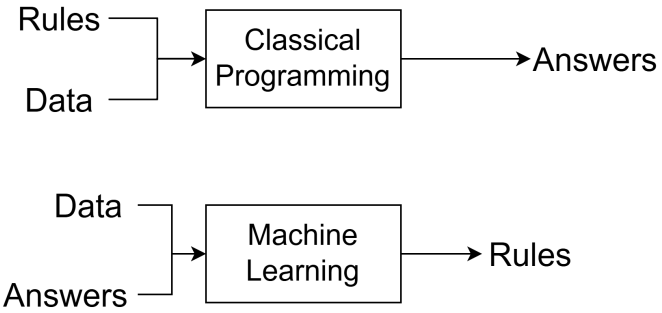


FIGURE 2.1: ML vs. conventional programming.

2.2 Machine Learning

2.2.1 Definition and Scope

The classic programming algorithms design process starts with the accession of subject expertise, where the programmer studied the given problem in detail, developing a mathematical model that capture the physics of the environment under study. Based on that model, the programmer creates an algorithm that guarantees optimized perforamnces with the assumption that the given physics model represents a real world scenario [**42**].

In contrast, the first step in the ML process is generally data collection, without having prerequisites such as acquiring domain knowledge. Furthermore, ML needs a

sufficient amount of data related to the problem at hand for the sake to train a ML model [**42**]. In other words, ML researchers have to gather enough data to enable the algorithm to discover hidden patterns of the given problem. The datasets used for training the algorithm are commonly referred to as training sets. This is achieved using a ML algorithm, this algorithm is a set of procedures that allows a machine to learn how to perform a certain task. The performance of the model is then tested using new data, and the trained model is utilized for making decisions in the future. The learning phase usually consists of optimizing performance metrics, which vary across various types of ML and are used to validate the performance of the trained model. Unlike classic methods, ML does not always rely solely on domain knowledge, but somehow their success often depends on the incorporation of the that knowldge on the learning process [**42**]. In simple defined terms, ML is a sub-field of AI having a specific purpose which is the development of algorithms and statistical models that allow computers to execute tasks without being programmed in an explicit way and rather learn how to perform tasks differently by detecting patterns and making inferences from training data.

This section introduces ML fundamental concepts, approaches, types, and algorithms and lays the foundation for the subsequent sections on DL and DRL.

2.2.2 Key Concepts

The following subsection discusses some of the primary ideas that serve as the basis of the study of ML. It defines some of the key elements, such as features, labels, and training data, as well as vital questions like overfitting and underfitting.

Features are the attributes of the observed data, represented often as x, they can be raw data, such as pixel values in images, or transformed data, such as word frequencies in a text [**42**–**44**].

Labels are the target outputs that represent the informationthe algorithms seek to predict, frequently represented as y, which can be discrete categories in classification or continuous numbers in regression [**43, 44**].

Training Data is a collection of labeled samples that used to train ML models. In the training phase, the model is responsible for extracting patterns and relationships from this data [**42, 44**].

Test Data is a set of unlabeled samples and is used to evaluate the performance of a model that has been previously trained on the training data [**43, 44**].

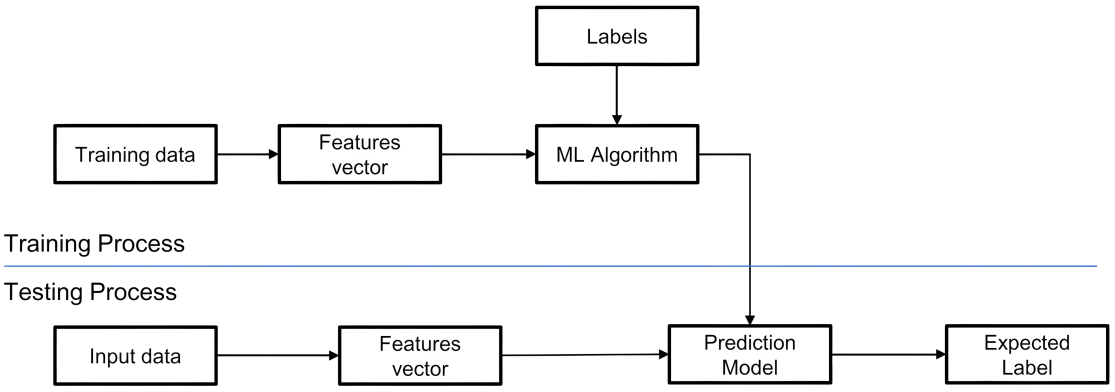


FIGURE 2.2: Supervised Learning.

Overfitting presents the modeling phenomenon that occurs when a model learns not only the underlying patterns but also the noise present in the training data. Such a model usually performs well on training data but poorly on unseen test data. As a result, it is very effective on the training set, but very ineffective on unseen data, leading to a substantial gap between training error and test error. Overfitting usually happens when the capacity of the model is too high, enabling it to fit the training very well [**45**].

Underfitting is a concept used to explain the situation where the model is too simple to discover the patterns in the training data. As a result, it performs poorly on both the training set and unseen data, leading to high training error. Underfitting typically happens when the model has insufficient capacity or complexity to learn from the training data effectively [**45**].

2.2.3 ML Types and Algorithms

2.2.4 Types of Machine Learning

2.2.4.1 Supervised Learning

Supervised learning specifically entails the learning of a mapping function from labeled input data to the output labels. The model is able to predict outputs for previously unseen data by inferring relationships between different data. The most common tasks in supervised learning are the classification and regression of data. Figure 2.2 illustrates the principal of supervised learning.

2.2.4.1.1 Classification The classification is a process in which the instances are put into predefined categories. Some of the instances of this are as for example, choosing the type of object from a given list of objects by using the images, like a mug example or

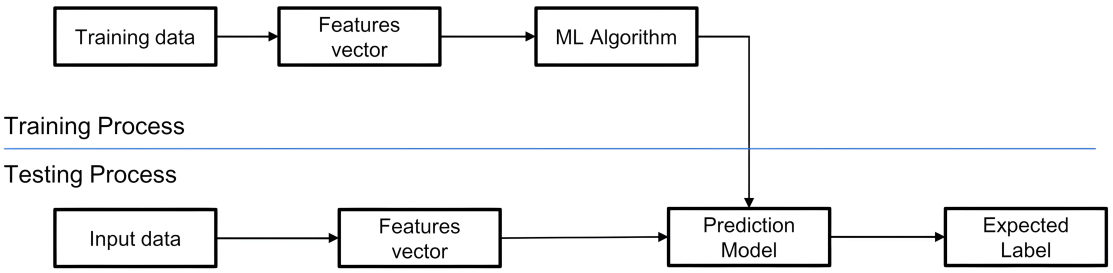


FIGURE 2.3: Unsupervised Learning.

spotting an anomaly in the system through transaction analysis [**43, 44**]. The algorithms commonly used for building such systems include simple logistic regression, decision trees, support vector machines (SVM), and neural networks [**44, 46, 47**].

2.2.4.1.2 Regression The regression is the process of predicting the output variable that takes a continuous value. For example, estimating the price of a house after examining several aspects of the problem, such as location, area, etc., while launching marketing campaigns to ensure that prices are maintained would be one of the bits of systemic data that this example could predict through regression analysis. Various algorithms that are commonly used are, for instance, linear regression, decision trees, or Bayesian networks [**46**].

2.2.4.2 Unsupervised Learning

Unsupervised learning schemes an aim to find unknown rules in unlabeled data. Without instructions, learning is done using input data only. Examples consist of dimensionality reduction and clustering [**43, 44**]. Figure 2.3 illustrates the principal of unsupervised learning.

2.2.4.2.1 Clustering the clustering is the task of putting identical data into classes without knowing the classes (labels). Algorithms like k-means, hierarchical clustering, and Gaussian mixture models are common [**46**].

2.2.4.2.2 Dimensionality Reduction the dimensionality reduction is a process in which the number of features in the dataset is decreased while the essential part of the information is retained, where it serves as a preprocessing method to prepare high dimensional data for other learning tasks. The commonly used methods include principal component analysis (PCA), autoencoders, and manifold learning [**46, 47**].

2.2.4.3 Reinforcement Learning

Reinforcement learning involves an agent interacting with an environment and learning by receiving rewards or penalties based on its actions, which is seeking to optimize its decision-making strategy [**46**]. Some of RL use cases include self-driving cars, competitive game, robotics. The regularly used algorithms are Q-learning, and its extension by the use of deep networks.

2.2.5 Machine Learning Algorithms

ML algorithms are grouped into three cetegories based on their use cases, thus in this subsection, we will provide brief overview of popular algorithms, grouped by their learning type.

2.2.5.1 Supervised Learning Algorithms

- **Linear Regression**: This is a method that estimates the linear relation between the input and the output [**44**]. It is particularly useful in predicting continuous variables [**46**].
- **Logistic Regression**: A common application of logistic regression is estimating the probability that a certain instance belongs to a particular class, frequently used in binary classification [**44, 46**].
- **Decision Trees**: This is another ML algorithm which enables decision-making that is easy to follow and comprehend, as it builds a tree-like structure [**46**].
- **Support Vector Machines (SVM)**: SVMs are algorithms that optimize a hyperplane separating the data. The core strength of SVMs is their effectiveness in dealing with high-dimensional spaces data, and their versatility offered by the kernel trick [**46, 47**].

2.2.5.2 Unsupervised Learning Algorithms

- **k-Means**: k-means is a popular method that divides data into k clusters; it is a simple yet widely used algorithm[**46, 47**].
- **Hierarchical Clustering**: This method of clustering is used for gaining a deeper knowledge of the relationships between the samples included in it, which is done by creating a hierarchy of clusters [**46, 47**].

- **Gaussian Mixture Models**: They are applied when data is believed to be a mixture of multiple Gaussian distributions, and they are quite capable of differentiating between clusters of various shapes in most cases [**46**].
- **Principal Component Analysis (PCA)**: PCA is a method that can identify the essential factors that influence most variability in the data, as well as enabling dimensionality reduction [**46, 47**].

2.2.5.3 Reinforcement Learning Algorithms

- **Q-learning**: Q-learning is an off-policy method; thus it attempts to find the optimal action-value function [**46, 47**].
- **Actor-Critic Methods**: This method is a combination of the value-based and the policy-based, hence this method is useful in complex environments [**46**].
- **Deep Reinforcement Learning**: Its function is to apply deep neural networks to solve RL; thus, it is useful for high-dimensional spaces [**46, 47**].

Choosing the most appropriate algorithm depends on the problem at hand and the available data is important. For example, linear regression is very simple and less complex in its computation, although, it can only represent a linear relationship. Furthermore, neural networks are very powerful, but they require large datasets.

2.2.6 ML Models Evaluation Metrics

ML evaluation metrics are important for estimating the performance of ML models. Therefore, the choice of a metric depends on the nature of the problem at hand. Different ML evaluation metrics are [**46**]:

Accuracy: represented by the proportion of correct predictions. However, it may be misleading for imbalanced datasets [**44, 46, 47**].

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{2.1}$$

Precision: represented by the proportion of true positive predictions among all positive predictions. A high precision means fewer false positives [**44, 46, 47**].

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.2}$$

Recall: It measures the proportion of true positive predictions among all actual positive instances. A higher recall means fewer false negatives [**44, 46, 47**].

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2.3}$$

F1-Score: Calculated as the harmonic mean of precision and recall, that captures trade-off between precision and recall [**46, 47**].

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.4}$$

ROC AUC: ROC stands for Receiver Operating Characteristic and AUC stands for Area Under the Curve. It assesses how well the model can differentiate between classes over different thresholds, useful for imbalanced datasets [**46, 47**].

$$\text{Area Under the Curve (AUC)} = \int_0^1 TPR(FPR^{-1}(t))\,dt \tag{2.5}$$

Here, TPR is the True Positive Rate, and FPR is the False Positive Rate.

The choice of evaluation metric depends on the nature of the task at hand and the priorities. For example, with imbalanced datasets, accuracy is not the right metric; instead metrics such as precision, recall, or F1 score are better choices [**44**].

2.2.7 Methodology

2.2.7.1 Data Preprocessing

Data preprocessing is an important step of preparing raw data for ML algorithms [**46**]. Further, the quality of ML model results depends on data quality. The common techniques for data preprocessing are:

Data Cleaning: The treatment of missing data, errors rectification, and elimination of duplicates and outliers.

Normalization: The process of unifying the scales of features so that there is no bias towards variables with larger numerical values, such as normalizing the data in a specific range like [0, 1], or [-1, 1]. The common min-max normalization and standardization are the typical processes in normalization.

Feature Scaling: is to modify the values of features to a set range without creating distortion in value differences. The aim is to prevent the model from favoring variables with the largest numerical values.

Handling Missing Data: Among the used techniques, we note the removal of rows with missing values and the replacement of missing values with mean, median, etc.

2.2.7.2 Feature Engineering

Feature engineering refers to the process of selecting, transforming, or creating new features that are based on existing knowledge and available data [**46**]. Some of the common techniques are the following:

Feature Selection: Reduce the number of features by removing irrelevant or redundant features from the input raw [**44**].

Feature Extraction: New features from input data are created that capture essential patterns in a low-dimensional representation [**47**].

Feature Transformation: Is the changement of existing features to a form identical to what models require, e.g. encoding categorical features into the corresponding numerical representations, polynomial feature expansions.

2.2.7.3 Model Selection and Hyperparameter Tuning

Model selection dictates the type of algorithm to use in a given situation, thus a sound knowledge of various algorithms and data is necessary [**46**]. There is no single solution for everything, thus the algorithm most fitting should be the one to be chosen.

Hyperparameter tuning is the process of identifying the best values of the parameters related to the learning algorithm (e.g., learning rates, number of hidden layers, regularization strength, etc. See subsection 2.3.2). The right values can have a substantial effect on performance [**46**]. Some common methods are:

- **Grid Search** [**45**]: The grid is a discrete n-dimensional space made of possible values of the parameters and the method checks every single combination of values stating, e.g. if alpha and beta can be 0.3 or 0.5, then it will try with 0.3/0.3, 0.3/0.5, 0.5/0.3 and 0.5/0.5 with the rest of the parameters fixed for every single case, thereby taking the best combination.
- **Random Search** [**45**]: It randomly picks some parameter combinations that are selected from the parameters space.
- **Bayesian optimization**: Employs the Bayesian method which is combined with the previously gained knowledge to assess all the hyperparameter search space

possibilities with the current performance tests to assess how successful the combinations are or if there are new combinations that haven't been tried. Through this method, the trial testing is minimized and the best candidate can be selected faster.

2.2.7.4 Cross-Validation

In order to estimate the cross-validation performance of the ML model, the whole training data is divided into many partitions so that the model is trained on a portion of the data only and is validated using the managed testing data on the other part. The goal behind using cross-validation is to get an unbiased estimation of the performance of the model [**46**]. Some of the used techniques are:

- **k-Fold Cross Validation:** By using this method the dataset is divided into k equal parts (folds). The power of the model is trained with k-1 parts and the rest is the test. The process is repeated k times and each time, one different part is held out. Finally, the performance is computed by averaging the values obtained.
- **Stratified Cross Validation:** It is a variant of k-fold cross-validation that ensures that the distribution of classes in each fold is the same as the one in the whole dataset; it works with imbalanced datasets as well.

2.3 Deep Learning

DL is a subfield of ML, which implies that all concepts within ML remain valid in DL, but not vice versa. DL utilizes DNNs to train models, particularly for large datasets, achieving high performance and serving as a powerful computational tool.

2.3.1 Neural Networks

Artificial Neural Networks (ANN) or "Neural Networks (NN)" are ML architectures that simulate the mechanism of learning in biological brains. The human nervous system contains cells, which are referred to as neurons [**48**]. Axons and dendrites are used to link the neurons in the human brain, and the areas that join axons and dendrites are called synapses [**48**]. These connections are illustrated in Figure 2.5(a). The strengths of synaptic connections often change in response to external stimuli [**48**]. This change is how learning takes place in biological organisms [**48**]. ANN neurons are interconnected in layers, where the first layer is the input layer; here, data enters the NN; the last layer

is the output layer, which provides the results; and the layers in between are hidden layers, which handle the complex processing, and the strength of their connections is represented by weights that are learned during training.

2.3.1.1 Neurons

The biological mechanism mentioned above is simulated in ANNs, which contain computation units that are referred to as neurons. Each input to a neuron is scaled with a weight w, which affects the function, z, computed at that unit. An ANN computes a function of the inputs by propagating the computed values from the input neurons to the output neuron and using the weights w and biases b as intermediate parameters, where each neuron has its own bias and every edge has its own weight [**48**]. Learning happens by updating the weights that connect the neurons as well as the biases within each neuron. Just as external stimuli are needed for learning in biological organisms to learn a targeted behavior, the external stimulus in ANNs is provided the training data containing examples of input-output pairs of the function to be learned [**48**]. For mathematical representation, The function a computed at a single neuron is given by:

$$a = f(\mathbf{w} \cdot \mathbf{x} + b) \tag{2.6}$$

In Equation 2.6, $\mathbf{x}$ represents the input vector, which corresponds to the raw data vector with all its features, $\mathbf{w}$ is the weight vector to be updated, b is the neuron bias, and f is the activation function. The activation function is a non-linear function that introduces non-linearity into the model, which is essential for learning complex patterns in the data. The most commonly used activation functions are discussed in subsection 2.3.2.

2.3.1.2 Layers

The NN layers are a core hyperparameter that affects considerably the model's performance. The layers are the building blocks of a NN, where they are interconnected with each other. Furthermore, the layers are divided into three different types: input, hidden, and output. NNs are classified on three types based on their number of layers, a single-layer NN which is also called a perceptron, a multi-layer NN which includes an input layer, one or more hidden layers, and an output layer, the NN that incude only one hidden layer is called a shallow NN, and the NN that consists of multiple hidden layers referred to as a Deep NN (DNN) [**49**].

2.3.1.2.1 Perceptron A perceptron is a mathematical model that is similar to the function of one biological neuron. It takes inputs, then applies weights, and outputs a result based on a threshold decision rule. The perceptron was introduced during the first wave of NNs research in the 1950s–1960s; at that time, it was used as it could learn a decision boundary separating two linearly separable classes. The perceptron was unable to handle problems that were not linearly separable (for example, the XOR problem), as noted in the Perceptrons book [**50**]. The development of this perceptron opened the path for DNNs which overcame its limitations in handling more complex problems through the addition of other hidden layers and the use of nonlinear activation functions. A perceptron example is illustrated in Figure 2.4.

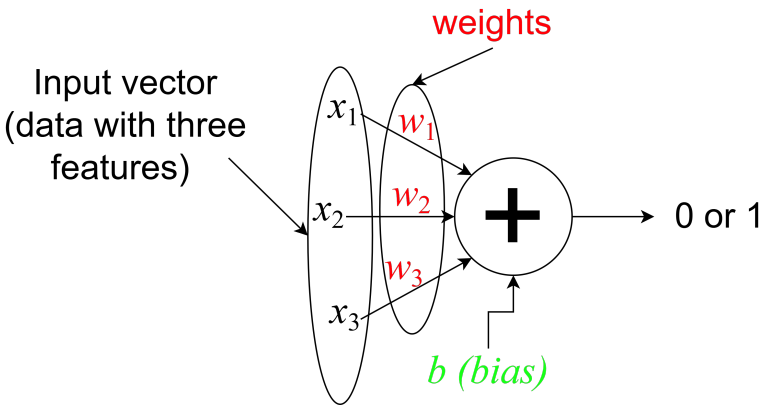


FIGURE 2.4: A Perceptron example.

2.3.1.2.2 MultiLayer Perceptron (MLP) MultiLayer NNs (or MLPs) are the developed form of the perceptron, where they consist of multiple hidden layers with a variable number of neurons in each layer of the NN. An MLP has an input layer where the raw data is provided, so its dimension is equal to the number of features in that provided data, followed by one or more hidden layers with a variable number of neurons, and an output layer that outputs the approximated results, where the dimension of the output layer is based on the problem to be solved or the task the model is trained on. As previously stated, the number of layers and neurons in each layer are hyperparameters that influence the model's performance and therefore require careful tuning to achieve optimal results. Given the number of layers in the NN as L and N_l as the number of neurons in layer l, where $l = 0, \ldots, L-1$. When $l = 0$, it refers to the input layer, and N_0 equals the number of features. Similarly, when $l = L-1$, it refers to the output layer, and $N_{L-1} = m$, where m is the number of classes or outputs. Starting by presenting a single neuron in a hidden layer l, the function $a_n^{[l]}$ computed at that neuron is given as:

$$a_n^{[l]} = f(z_n^{[l]}) = f((\mathbf{w}_n^{[l]})^T \cdot \mathbf{a}^{[l-1]} + b_n^{[l]}) \tag{2.7}$$

Here, $\mathbf{w}_n^{[l]}$ is the weights vector of the current neuron n in layer l, and its column vector is obtained using the transpose T, $\mathbf{a}^{[l-1]}$ is the vector of the output of the previous layer, and $b_n^{[l]}$ is the bias of the current neuron. A neuron representation in a hidden layer is illustrated in Figure 2.5.

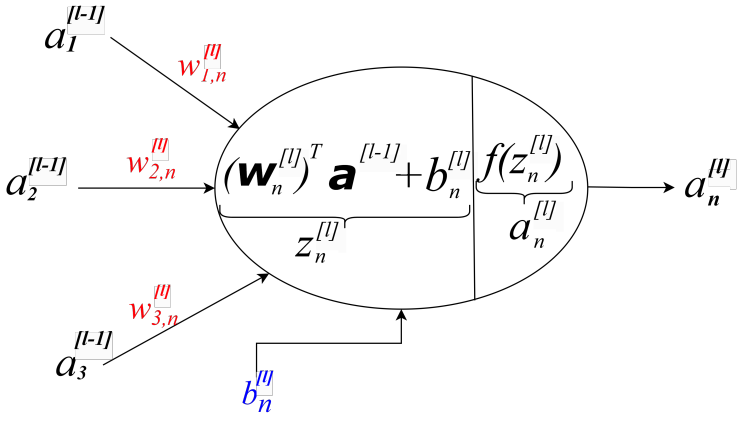


FIGURE 2.5: A single neuron representation in a hidden layer.

Furthermore, an MLP consists of multiple layers, and each layer contains multiple neurons. The feedforward process, based on the example in Figure 2.6, given $\mathbf{x}$, is as follows:

$$\begin{aligned}
\mathbf{z}^{[1]} &= \mathbf{w}^{[1]} \cdot \mathbf{x} + \mathbf{b}^{[1]} \rightarrow \mathbf{a}^{[1]} = f(\mathbf{z}^{[1]}) \\
\mathbf{z}^{[2]} &= \mathbf{w}^{[2]} \cdot \mathbf{a}^{[1]} + \mathbf{b}^{[2]} \rightarrow \mathbf{a}^{[2]} = f(\mathbf{z}^{[2]}) \\
\mathbf{z}^{[3]} &= \mathbf{w}^{[3]} \cdot \mathbf{a}^{[2]} + \mathbf{b}^{[3]} \rightarrow \mathbf{a}^{[3]} = f(\mathbf{z}^{[3]}) \\
\mathbf{z}^{[4]} &= \mathbf{w}^{[4]} \cdot \mathbf{a}^{[3]} + \mathbf{b}^{[4]} \rightarrow \mathbf{a}^{[4]} = f(\mathbf{z}^{[4]})
\end{aligned} \tag{2.8}$$

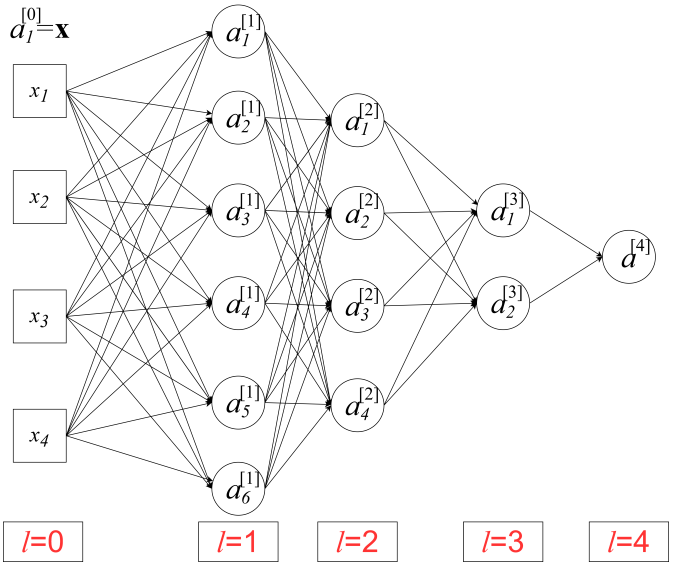


FIGURE 2.6: An MLP example ($L = 5$).

2.3.2 Core Principles for Training Deep Networks

The training of DNNs is a complex process that requires the understanding of its core principles, such as activation functions, loss functions, learning rate, forward propagation, backpropagation, regularization techniques, and optimization algorithms. These principles are discussed in the following subsections.

2.3.2.1 Activation Functions

In the context of NNs, an activation function is a mathematical function applied to the output of a neuron (or node) in a network, and it represents an important hyperparameter. It introduces non-linearity into the model, which allows NNs to learn complex patterns and relationships in data.

The non-linearity introduced by an activation function is important in training NNs because it allows the network to learn non-linear relationships in the data and, therefore, enables the training of a complex model [**51**]. Without non-linearity, even deep networks are equivalent to a single linear layer, which would significantly limit their power. As gaols of activation functions, they should be differentiable, non-linear, and have a range of values that are not too large or too small [**52**]. The most commonly used activation functions are [**53**]:

- **Sigmoid**: Sigmoid activation function, a common choice in the early development of NNs, maps any real-valued input into the range [0, 1]. Mathematically, it is defined as:
$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.9}$$
- **Softmax**: Softmax function is an extension of the Sigmoid function, typically used in the output layer of a NN for multi-class classification problems. It transforms a vector of real numbers into a probability distribution over predicted output classes. The Softmax function for the i-th class is given by:
$$\sigma_i(x) = \frac{\exp(x_i)}{\sum_{j=1}^{K} \exp(x_j)} \tag{2.10}$$
where K is the number of classes.
- **Tanh**: Tanh (Hyperbolic Tangent) function is another used activation function that addresses some of the limitations of the Sigmoid function. It maps input

values into the range [-1, 1], providing a zero-centered output:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.11}$$

- **ReLU**: ReLU function has become the go-to activation function for many deep learning models due to its simplicity and effectiveness. It is defined as:

$$\text{ReLU}(x) = \max(0, x) \tag{2.12}$$

- **Leaky ReLU**: To address the "dying ReLU"[1] problem, the Leaky ReLU was introduced. It modifies the ReLU function by multiplying the input by a small positive constant for negative inputs:

$$\text{LeakyReLU}(x) = \max(\alpha x, x) \tag{2.13}$$

  where α is the small positive constant, usually around 0.01.

- **Swish**: Swish activation function is a more recent proposed function that has shown promise in improving the performance of NNs in some cases when compared to ReLU and its variants. It is defined as:

$$\text{Swish}(x) = x \cdot \sigma(\beta x) \tag{2.14}$$

  where σ is the sigmoid function and β is a learnable parameter.

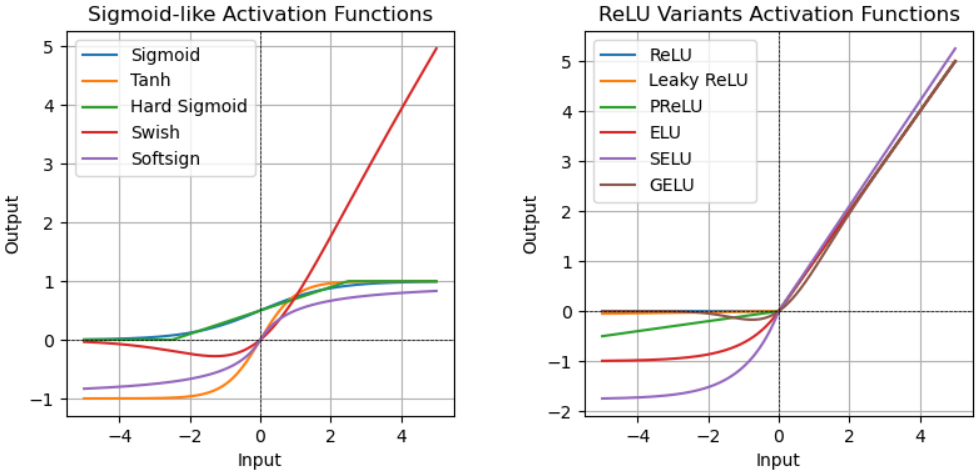
A visualization of some activation functions is presented in Figure 2.7.

2.3.2.2 Loss Functions

To optimize an objective function in any optimization problem, we either maximize or minimize it. In DL, and in ML in general, the objective function may also be referred to as a cost function or loss function [**45**].

Furthermore, a loss function is a mathematical function that quantifies the value the model is trained to minimize or to maximize. In this thesis, we will not introduce all loss functions available; instead, we will present the common ones for the sake of brevity.

[1] When using ReLU, the output might always be zero for some neurons.



(a) (b)

FIGURE 2.7: Activation functions (a) sigmoid-like. (b) ReLU variants.

2.3.2.2.1 Loss Functions for Classification Tasks

- **Binary Cross Entropy (BCE)**: this loss function is used in the binary classifiers, where the output should be 0 or 1, and it is formulate as follows:

$$\text{BCE} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right] \tag{2.15}$$

- **Categorical Cross Entropy (CCE)**: the CCE extends the BCE to a multi-class classifiers and it is usually used with softmax activation function. CCE loss function is given as follows:

$$\text{CCE} = -\sum_{i=1}^{n}\sum_{k=1}^{K} y_{ik} \log(\hat{y}_{ik}) \tag{2.16}$$

- **Hinge loss**: This loss function is primarly used for SVM, and for binary classification where data is labeled as -1 or +1. It expressed Mathematically as follows:

$$\text{Hinge Loss} = \sum_{i=1}^{n} \max(0, 1 - y_i \hat{y}_i) \tag{2.17}$$

2.3.2.2.2 Loss Functions for Regression Tasks

- **Mean Squared Error (MSE)**: Measures the average squared difference between the predicted values ($\hat{y}$) and actual values (y). It works well for normally distributed data but is less effective for handling outliers. The MSE loss function is

expressed as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.18}$$

- **Mean Absolute Error (MAE)**: Calculates the average absolute difference between predictions and actual values. Unlike MSE, MAE is less sensitive to outliers but may suffer from slower convergence during optimization. It is given as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.19}$$

- **Huber loss**: Combines the strengths of MSE and MAE to balance sensitivity to outliers and speed of convergence. It is commonly used when data has small noise with occasional outliers. Huber loss is expressed as:

$$L(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{if } |y - \hat{y}| \leq \delta, \\ \delta|y - \hat{y}| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases} \tag{2.20}$$

  Here, The parameter δ is a threshold or cutoff point.

2.3.2.3 Learning rate

The learning rate is an ML hyperparameter, which is widely denoted as α, aand is responsible for controlling the optimization algorithm's step size (the optimization algorithms are detailed in Subsection 2.3.2.6). The learning rate is a key hyperparameter that significantly affects the learning process and model performance; hence, its tuning is very important, and it is the most crucial parameter because it affects the convergence speed [**45**]. While a smaller learning rate may contribute to a stable training process, but lead to slower convergence, whereas a higher learning rate speeds up convergence, but the model may overshoot the convergence point. For the model to find that point, a well-tuned learning rate is therefore essential.

There are several techniques for scheduling learning rate behavior, such as step decay, exponential decay, cyclic learning rate, inverse decay, and others [**45, 48**].

2.3.2.4 Froward Propagation and Backpropagation

The process of passing messages among nodes and layers in DNN training contains forward propagation and backpropagation. On one hand, forward propagation is used to generate the output based on the input that is fed to a neuron and pass it forward until

the final output of the network. On the other hand, the process of backpropagation is responsible for computing the gradient of the loss function concerning the weights and biases of the NN and using a gradient optimization algorithm to update these elements.

2.3.2.4.1 Forward Propagation (Feedforward) This process computes the output of the neural network by passing the input data through its layers. The primary goal is to calculate the predicted output $\hat{y}$ for a given input x. Given equations in 2.7 and 2.8, the forward propagation process input is $\mathbf{a}^{[l-1]}$ and it outputs are $\mathbf{a}^{[l]}$ and $\mathbf{z}^{[l]}$ for each layer l, where it forward $\mathbf{a}^{[l-1]}$ to the next layer and cache $\mathbf{z}^{[l]}$ for future calculations in backpropagation.

2.3.2.4.2 Backpropagation (Backward) Backpropagation refers to two concepts: the mathematical method used to calculate derivatives and the training algorithm that updates network and biases to maximize the model's evaluation metrics. Backpropagation is used in DL to discover the structure of the data in large datasets to indicate to a model how it should change its weights and biases [**54**].

The backpropagation can be applied repeatedly to propagate gradients through all NN neurons, starting from the output layer and working all the way back to the input layer. Once the gradients are computed, the gradients with respect to the weights and the biases of each node are then calculated directly, where these gradients are used to update the weights with an optimization algorithm [**54**]. A visualization of feedforward and backpropagation processes is presented in Figure 2.8.

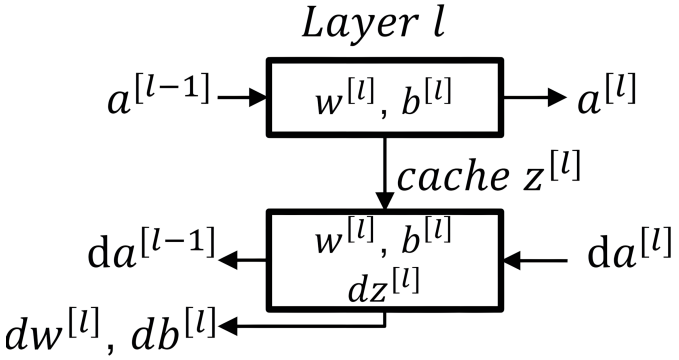


FIGURE 2.8: Illustration of feedforward and backpropagation at a layer of a NN.

2.3.2.5 Regularization

Referring to the problem of overfitting discussed in Subsection 2.2.2, a solution is needed to prevent it from happening. To deal with this issue, a technique called

regularization has been employed, which includes various approaches to prevent overfitting, each of which is designed to avert that problem differently, including L1 and L2 regularizations, dropout, etc.

2.3.2.5.1 L1 and L2 Regularization L1 and L2 are two regularization methods that add a penalty term to the implemented loss function; this term is added to reduce the impact of the weights on the system since they are learned to fit the training data well only, therefore preventing the model from generalizing to validation and test data. However, they differ in how the penalty term is applied in each of them. L1 turns many weights to zero, which makes it useful for feature selection use cases, and its formula is given as $L = L_{\text{original}} + \lambda \sum_i |w_i|$; here, λ is the regularization parameter that controls the penalty strength. Next, L2 regularization scales weights towards zero, but it rarely makes them exactly zero; thus, it reduces the impact of irrelevant features and typically works better than L1 in preventing the overfitting problem. It can be expressed as $L = L_{\text{original}} + \lambda \sum_i w_i^2$.

2.3.2.5.2 Dropout Dropout is a technique used to reduce NN overfitting, where it works by randomly turning off some neurons and their connections during training so that the network does not rely too much on specific ones [**55**]. Dropout is represented by a hyperparameter p, which determines the probability for neurons to be dropped in each layer. During each training step, neurons are randomly chosen to be deactivated using p, which creates different network setups each time. As a result, individual neurons may experience occasional deactivation during training cycles. This dynamic suppression forces the network to build robust, generalizable feature hierarchies instead of memorizing training artifacts.

2.3.2.5.3 Early Stopping As discussed in Subsection 2.2.7.4, the training process divides the data into training and validation sets, where the training set is used to train the model and the validation set is used to evaluate the model during training after each epoch. The validation and training performance continuously increase as the training process progresses; however, at some point, the training performance keeps rising while the validation performance starts to decrease; therefore, the model starts to overfit at this stage. To avoid overfitting in this situation, a method called early stopping is used to stop the training process when the validation performance starts to decline [**56**].

2.3.2.5.4 Data Augmentation The lack of data in some cases can lead to model overfitting. Datasets are usually limited in practical scenarios and we cannot always

have enough data, data augmentation is used to increase the amount of data by applying simple transformations to the existing data, such as rotation, scaling, flipping, cropping, and even adding noise, to create new data samples [**56**]. This regularization technique helps the model generalize better and prevent both overfitting and underfitting.

2.3.2.5.5 Group Normalization Group Normalization (GN) is a normalization technique developed to tackle certain issues of standard batch normalization that happen in cases where the batch sizes are small. For instance, if there are 32 channels, GN may split them into 4 groups of 8 channels each. Then, the normalization is applied to each group separately, which helps to reduce the dependency on batch size [**57**]. In order to preserve the representational capacity of the original features, GN applies a learnable scale and shift to each channel after normalizing, much as in batch normalization.

2.3.2.6 Optimizers

All DL techniques require some form of optimization. Optimization is the process of minimizing or maximizing a cost function $J(\theta)$ by modifying θ. Most optimization problems are typically expressed in terms of minimizing $J(\theta)$. To maximize, use a minimization technique that minimizes $-J(\theta)$ [**45**]. Derivatives are useful for minimizing a function because they indicate how to change θ to achieve a small improvement in y [**45**]; thus, this is why the backpropagation algorithm is adopted in NNs to calculate gradients and, therefore, allows the optimizer to optimize the model parameters.

The distinction of the DNN training optimizer is the critical determinant for the performance of DL models. The most common first-order optimization techniques used in training DL models are the calculation of the gradient of the cost function with respect to the weights and biases. The goal of using an optimizer is to obtain weights and biases that will help the NN make optimal predictions. The optimizer starts at an arbitrary position, which means that the parameter values are arbitrary. The initial values are nevertheless random, but not all these values offer a good result; hence, many approaches are available for weight initialization, including but not limited to zero initialization, random initialization, Xavier initialization, and He initialization. In this subsection, we will discuss the most common optimizers used in DL.

2.3.2.6.1 Gradient Descent (GD) GD is the first optimizer used in DL, and it is the simplest optimization algorithm. It works by updating the weights and biases in the opposite direction of the gradient of the loss function (L) with respect to the weights

and biases. The update rule for the weights and biases when using GD is given as:

$$\theta = \theta - \alpha \cdot \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta L_i, \tag{2.21}$$

Here θ represents the weights or biases, α is the learning rate, N is the number of samples, and $\nabla_\theta L_i$ is the gradient of the loss function with respect to the weights or biases.

2.3.2.6.2 Stochastic GD (SGD) The GD uses all the samples in every single optimization epoch, which is a time- and resource-consuming process. To address this drawback, SGD uses only a mini-batch of samples at a time; consequently, it applies the same updating rule as GD but with the summation over only a mini-batch of samples.

2.3.2.6.3 RMSProp As we mentioned in Subsection 2.3.2.3, where we introduce the learning rate, it is one of the most important hyperparameters that influences model performance and therefore has to be properly tuned to achieve the best results. For the moment, there are two ways to change the learning rate during training: using different algorithms to change the learning rates of each network parameter and applying the same learning rate to the whole network, with the exception of updating it through the training epochs. One of the algorithms that tunes the learning rate for each network parameter is RMSProp, which is an adaptive method of computing the average of squared gradients by using the running average over the latest time interval. The equations provided below are the mathematical representations for determining the new network parameter values:

$$v_t = \beta v_{t-1} + (1-\beta) g_t^2, \quad \theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{v_t + \epsilon}} g_t \tag{2.22}$$

where β is the decay rate (typically between 0.9 and 0.999), α is the learning rate, and ϵ prevents division by zero.

2.3.2.6.4 Adam The Adaptive Moment Estimation (ADAM) [**58**] method can simply be defined as an advanced, first-order gradient-based optimizer that applies the stochastic objective functions governing the process of its training. Just like other sophisticated methods for outer parametric optimization, such as ADAM, it has many benefits: it is easy to work with, takes few resources, and does not depend on the scaling of the gradients. The main idea underlying the algorithm is the calculation of separate adaptive learning rates based on the estimation of the first and second moments of the gradients. Another key point is that, through this, a mechanism is created that allows ADAM to adaptively explore areas of the objective space, even the areas of the

objective function that are sparse and/or noisy, which is one of the main characteristics of non-convex optimization in ML. ADAM updates parameters following the equations below:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t, \tag{2.23}$$

where $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$ is the bias-corrected first moment estimate, $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$ is the bias-corrected second moment estimate, m_t is the first moment estimate, v_t is the second moment estimate, α is the learning rate, and ϵ is a small constant to prevent division by zero.

2.3.3 Deep Learning Architectures and Models

2.3.3.1 The Simple ANN

MLP is one of the simplest types of artificial neural network where the data flows in one direction—from input to output—without any loops or cycles. MLPs are particularly suitable for structured, tabular data and simple mapping tasks like classification and regression. It is just the same as presented in Subsection 2.3.1.2.2.

2.3.3.2 CNN

DNNs can be complicated structures formed from the basic structure of the human brain. However, Convolutional Neural Networks (CNNs), which represent a special kind of NNs, have different characteristics. CNNs have been developed to deal with inputs in grid formats such as images or time series. Their architecture and functionality are inspired by the way the visual cortex processes visual stimuli; thus, they have become the most powerful tool, especially when it comes to tasks where spatial hierarchies are involved.

2.3.3.2.1 CNN Operations

2.3.3.2.1.1 Convolution [59] CNN applies convolution operations using filters (also called kernels) that slide over the input data to extract local features. Filters are one of the main components of CNNs, which are square matrices that have dimensions $f \times f$, where f is an integer and is usually a small number, like 3 or 5. We could hard-code filters, but a much better way is to treat the filter as parameters to be learned. Each filter detects specific patterns, such as edges, textures, or shapes.

2.3.3.2.1.2 Padding [52] There are two problems when using CNNs, and they are: images shrink, and the information on the borders of images is not preserved as well as the information in the middle. To address this, padding with a border of zeros is added to the input image before the convolution operation.

2.3.3.2.1.3 Stride [52] The stride is simply the number of rows or columns you move your region when selecting the elements; thus, it determines how far the filter moves during convolution. A stride of 1 means the filter slides one pixel at a time; a stride of 2 skips every other pixel.

2.3.3.2.1.4 Pooling [59] There is a type of layer in CNNs called a pooling layer, which reduces the dimensionality of feature maps to make the network more efficient and prevent overfitting. It is used to reduce the spatial dimensions of feature maps while retaining important information and discarding noise. The pooling layer has two types: max pooling and average pooling. On one hand, max pooling selects the maximum value in a region. On the other hand, average pooling computes the average value in a region.

CNNs are DL architectures developed for grid-like data, such as images. They use convolutional layers to extract features, pooling layers to minimize complexity, and fully connected layers to make final predictions. An example of a CNN is shown in Figure 2.9, the example includes a convolutional layer (padding, stride, and filter), and a pooling layer [**60**]. While CNNs are extremely good at image classification, segmentation, and object recognition, they require large datasets and significant computer resources to train. Common architectures such as LeNet, AlexNet, and ResNet have expanded the scope of computer vision tasks.

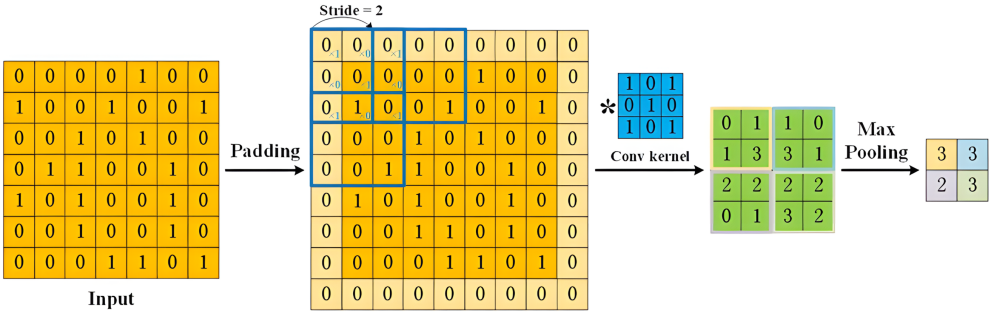


FIGURE 2.9: An example of a CNN architecture.

2.3.3.3 RNN

NNs, including CNNs, typically have fixed input and output sizes and govern the flow of information via a feedforward approach. However, sequential data frequently contain dependencies between data points, necessitating a strategy for retaining prior knowledge in order to generate reliable predictions about future values.

RNNs (Recurrent Neural Networks), which are specifically designed to handle sequential data with important input order. As a result, these networks are best suited for time-related tasks like time series and natural language processing, as well as any scenario requiring context and sequence. Memory systems in the form of hidden states are utilized in the case of RNNs instead of the feedforward structure. In this way, the relationships between time periods and patterns in the data can be detected. The reason for this sequential processing becomes clear when we visualize the unrolling of the network in time, as shown in Figure 2.10 [**51**]; each time step inputs the current signal and the previous time step's hidden state and regenerates its hidden state accordingly [**51**]. Due to this structure, RNNs can handle sequences of varying lengths and retain context across time, making them promising candidates for language modeling, speech recognition, and time series forecasting. However, despite their potential benefits, RNNs are extremely sensitive to exploding or vanishing gradients, making it impossible to learn long-term connections. Gradient clipping methods, LSTM and GRU structures, the most recent architectures meant to address the aforementioned difficulties, are gradually improving the RNN's ability to capture complex temporal dynamics.

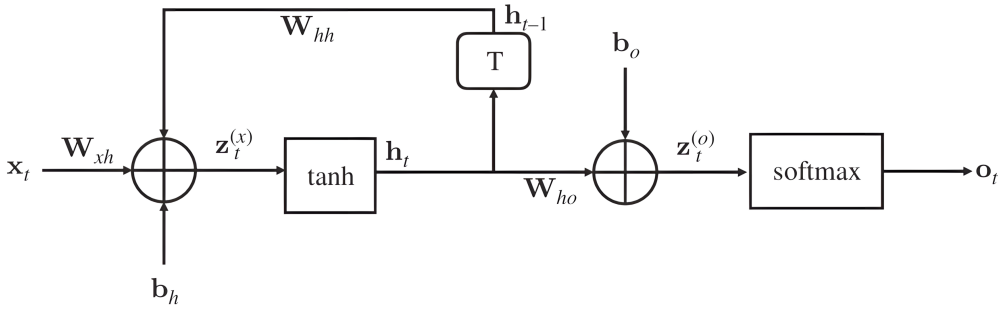


FIGURE 2.10: An RNN's compact representation.

2.3.3.4 GNN

The widespread adoption of Graph Neural Networks (GNNs) in DL highlights their capacity to manage data with an inherent graph structure in a spectral manner. Unlike previously discussed NNs, which operate on regular grid-like data (e.g., images or text), GNNs are designed to process data in irregular and non-Euclidean domains such as social

networks, biological networks, and knowledge graphs. Message passing is a key concept in GNNs, in which nodes in a graph iteratively share information with their neighbors in order to update their representations, collecting both local and global structural information from the graph [**52**]. In a GNN, the update rule for a node's hidden state can be written as follows [**52**]:

$$h_v^{(k)} = \text{COMBINE}^{(k)}\left(h_v^{(k-1)}, \text{AGGREGATE}^{(k)}\left(\{h_u^{(k-1)} | u \in \mathcal{N}(v)\}\right)\right) \tag{2.24}$$

Here, $h_v^{(k)}$ represents the hidden state of node v at layer k, $\mathcal{N}(v)$ denotes the set of neighbors of node v, and AGGREGATE and COMBINE are learnable functions that aggregate and combine information from neighboring nodes, respectively.

GNNs have different types of architectures, with the Graph Convolutional Network (GCN) being the most basic one. GCNs extend the concept of convolution from grid-based to graph-based datasets. Both GCNs and related architectures rely on graph attributes, which influence their operations but do not fundamentally alter the convolution process, where convolution in this context is performed in the spectral domain using the graph Laplacian matrix, which encodes node connectivity and similarity. Graph Attention Networks (GATs) are a kind of GCN that has a mechanism for obtaining focused attention; it is basically employed to choose which nodes are the most essential in a network and then create, for instance, a new node to describe or hide the currently mentioned, discover an alternative idea, or specify critical regions. Another GNN-based model is the Graph Autoencoder (GAE), which predicts missing links or reconstructs graphs. By encoding node information into a low-dimensional latent space, GAE can either derive missing connections or restore node features to their respective parents; thus, it reconstructs the graph [**61**].

2.3.3.5 Autoencoders

Autoencoders are a type of artificial neural network used in unsupervised learning to learn efficient representations (or codings) of input data. They consist of two main components: an encoder, which compresses the input into a lower-dimensional latent space, and a decoder, which reconstructs the input from this latent representation. The network is trained to minimize the reconstruction error between the original input and the decoded output, enabling applications in dimensionality reduction, feature learning, denoising, and anomaly detection [**45**].

2.4 Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) is a state-of-the-art technique that combines the basic principles of RL with the powerful representation capabilities of DNNs. It determines the appropriate sequence of steps for a decision-making process by learning from high-dimensional input data that serves as the environment's states. Unlike its predecessor, RL, which finds it difficult to deal with complex environments due to its dependence on handcrafted features or low-scoped models, deep learning enables DRL to autonomously process and identify representative features. Based on the fact that DRL is capable of working with large state spaces or action spaces that are nearly infinite, problems like image-based decision-making and robotic control are more likely to be solved than before.

DRL is motivated by the need to solve more complex issues in dynamic and uncertain environments. Traditional RL algorithms are good at handling small and geometrically structured problems, but they may not be efficient in high-dimensional input fields, such as tasks involving visual perception. By using DNNs as function approximations, DRL is able to generalize well over large, unstructured data sets. Furthermore, the path to DRL began with standard RL methods, such as Q-learning and policy gradient techniques; these, however, were challenged by the issues of scale and generality of complex environments. The evolution came from the concept of Deep Q-Networks (DQN) in 2015 by Mnih et al., which started to show the potential of the combination of DNNs and RL. This work marked the first checkpoint in opening the door to various approaches like A3C, PPO, and SAC. These are some of the main pillars supporting modern DRL systems (e.g., multi-agent DRL (MADRL), meta DRL, etc.).

2.4.1 Reinforcement Learning

2.4.1.1 What's RL?

RL is a type of ML in which an agent trains itself to make decisions by interacting with an environment. The goal of the agent's learning process, which is started by trial and error, is to constantly maximize cumulative rewards. Using concepts from behavioral psychology, RL offers a model for how autonomous agents could adapt to and interact with an environment in order to finish a task successfully.

Where ML consists of supervised and unsupervised learning, where the model learns from data (labeled or unlabeled), RL is about learning from the environment by receiving

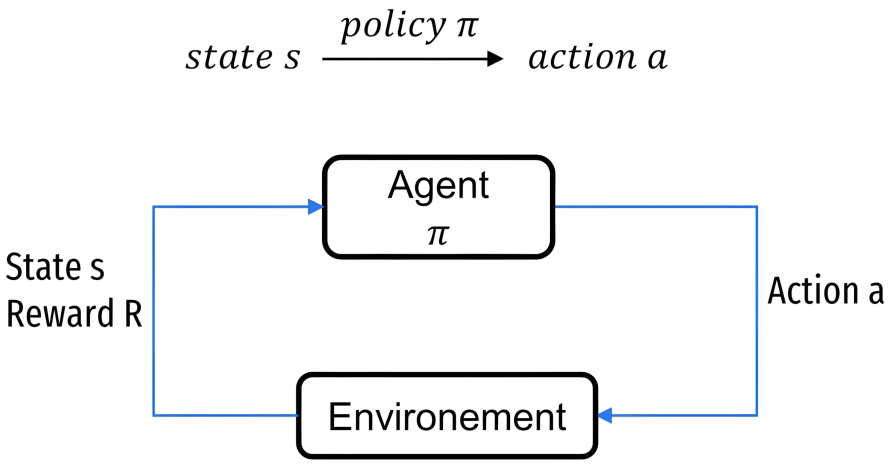


FIGURE 2.11: The MDP process representation.

feedback in the form of rewards or penalties without explicit instructions on the correct actions.

In this subsection, we will first go over the fundamental components of RL before introducing the Markov Decision Process (MDP), which is the principle that combines those components into a complete mathematical representation of the problem. Next, we'll discuss the exploration-exploitation trade-off in the RL paradigm.

2.4.1.2 RL Components and MDP Process

The RL paradigm includes an autonomous agent interacting with a precisely defined environment. For the agent to interact with this environment, it must have predefined state and action spaces, while the environment's state changes repeatedly as the agent takes an action from the specified action space at each time step. To allow the agent to learn, it receives a value from the environment using a reward function that gives a positive signal for desirable actions and penalizes undesirable ones. The function the agent follows to select an action in a given state is called a policy, denoted as π, which is the function the agent learns during training. Furthermore, the RL problem is typically modeled as an MDP process, which is given by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ represents the state space, $\mathcal{A}$ is the action space, $\mathcal{P}(s'|s, a)$ is the transition probability from state s to state s' after taking action a, $\mathcal{R}$ is the reward function, and $\gamma \in [0, 1]$ is a discount factor that balances the importance of immediate reward vs. future rewards. Figure 2.11 shows the MDP process flow. In the following, we will define and explain the purpose of each RL component in detail by taking a self-driving car as an example to fully understand the MDP and RL components:

- **Agent**: The entity responsible for interacting with the environment and making decisions. In the example of a self-driving car, the agent is represented by the self-driving car's control system, where it makes decisions based on inputs from its sensors (e.g., cameras, LiDAR, radar).

- **Environment**: An environment is the external system with which the agent interacts. In the example, the environment is the real-world road network, including traffic lights, other vehicles, pedestrians, and road conditions. This environment responds to the car's actions (e.g., accelerating, braking), and changes its state after each action taken by the agent.

- **State space $\mathcal{S}$**: This space includes all the important information of the environment to allow the agent to understand the current state, helping it to select the appropriate actions. The state represents the car's perception of the environment at a given time. For example, current speed and direction, distance to the nearest car, position of traffic lights (red, yellow, or green), and road conditions (e.g., wet or dry).

- **Action space $\mathcal{A}$**: This space is dedicated to the decisions that the agent can make at any state, and they directly affect the environment. Action space includes decisions that the car can make at any moment (accelerating, braking, turning left or right).

- **Reward function $\mathcal{R}$**: A function returns a numerical value that reflects how good or bad an action is at a given state. There are positive rewards in scenarios where the agent arrives at its destination, avoids collisions, or follows the traffic rules, and negative rewards in cases of collisions, running red lights, or inefficient routes that increase travel time.

- **Policy π**: A function that maps the states to actions, the agent follows to decide which action to take based on the current state. For example, if the car detects a red light ahead (i.e., current state), its policy might suggest using the brakes (i.e., the action to take). The policy is learned by the agent during training to maximize the cumulative rewards. Usually the policy can either be deterministic or stochastic. A **deterministic policy** maps each state $s \in \mathcal{S}$ to a specific action $a \in \mathcal{A}$: $\pi : \mathcal{S} \to \mathcal{A}$. While a **stochastic policy** maps each state s to a probability distribution over actions: $\pi(a|s) : \mathcal{S} \times \mathcal{A} \to [0, 1]$.

- **Episode**: An episode is a complete sequence of interactions between an agent and its environment that begins with an initial state and ends with a terminal state (a preset stopping condition). Episodes are essential for structuring learning in tasks having distinct beginning and end points, e.g., games, simulations, or goal-oriented

problems. As an example, in a self-driving car scenario, an episode could be a trip from the car's starting point to its destination or a car crash.

- **Exploration and exploitation tradeoff**: The agent must use a technique to balance between exploring many different actions to gain more information about the environment and applying its already available knowledge to receive the highest rewards in the process. In a self-driving car scenario, for example, the agent has to decide whether to take a new road that it has never taken before or to take a road that it already knows. In the first phase of the training, everything is about exploration; that's why the probability of exploring new actions is typically set at a high value, often at 1. After a few episodes, the agent's exploration probability begins to decrease, and the agent shifts its attention to the exploitation of the knowledge it already has.

2.4.1.3 Discounted Expected Future Returns

In RL, the primary objective of the agent is to maximize the cumulative reward that it collects over time. This cumulative reward the agent collected is referred to as the return and is denoted by G_t. The agent uses this return to evaluate the quality of its actions taken on the defined environment states. However, problems are encountered when rewards are distributed across time steps. In order to address this, RL makes use of a discount return concept to balance the "short-term" and the "long-term" rewards. The discounted value of the expected future return at time step t is given by:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.25}$$

Where γ is the discount factor, R_{t+k+1} is the reward recieved at time step $t+k+1$, and k is time step offset from t. This equation explains the cumulative reward over the trajectory[2] of the agent, with future rewards discounted exponentially by the factor γ. γ is a real value from 0 to 1, where 0 means that only the immediate benefits are taken into consideration, and 1 means that the agent prioritizes the future reward as much as it prioritizes the instant reward; thus, the discount factor is a hyperparameter that guides the agent to select actions that maximize the expected cumulative reward. As RL environments are often stochastic, G_t is usually given as the expected return:

$$\mathbb{E}[G_t] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\right] \tag{2.26}$$

[2]In RL, a trajectory (or rollout), denoted as τ, is a sequence of states, actions, and rewards $(s_0, a_0, R_1, s_1, a_1, R_2, \ldots)$ generated by an agent during interactions with an environment over some time steps, used to learn policies or estimate expected returns.

The discount factor implicitly defines the agent's effective time horizon, i.e., how far into the future the agent considers rewards. A smaller γ leads to a shorter horizon. Discounting future rewards is motivated by several principles: uncertainty, because future rewards are less certain and, therefore, are given less weight; time preference, since in many real-world scenarios, immediate rewards are more valuable than delayed rewards; and mathematical convergence, as discounting ensures that the infinite summation of rewards converges to a finite value, making it computationally possible.

2.4.1.4 Value Functions and Bellmann Equation

2.4.1.4.1 Value Function As a definition, a value function quantifies the expected long-term utility (cumulative reward) of a state or state-action pair following a given policy. It serves as a tool for evaluating and improving policies. There are two types of value functions, the state-value function, referred to as V^π, and the action-value function, referred to as Q^π; their distinctions are shown in Table 2.1. The state-value function gives the expected long-term reward by commencing from state s and always operating according to policy π; thus, V^π always follows the policy . The state-value function is given by:

$$V^\pi(s) = \mathbb{E}_\pi \left[G_t | s_t = s \right] \tag{2.27}$$

Besides $V^\pi(s)$, $Q^\pi(s, a)$ gives a more accurate estimation for a state-action pair. It computes the expected cumulative reward that would be earned through taking an action a in a given state s without following the policy π. We calculate $Q^\pi(s, a)$ by:

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[G_t | s_t = s, a_t = a \right] \tag{2.28}$$

Table 2.1: State-Value and Action-Value Functions Distinctions.

Concept	**Definition**	**Role in RL**
State-Value Function	Evaluates states under policy π: $V^\pi(s)$.	Used for policy evaluation (e.g., "Is this state good under the current policy?").
Action-Value Function	Evaluates state-action pairs under policy π: $Q^\pi(s, a)$.	Used for policy improvement (e.g., "Which action maximizes future rewards?").

2.4.1.4.2 Bellmann Equations The two types of value functions follow recursive rules called Bellman expectation equations. The implementation of these equations is

key in RL because they help create effective value function computations and guide policy optimization. The Bellman equation for the single state value function is expressed as:

$$V^{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} P(s'|s,a) \left[R(s,a,s') + \gamma V^{\pi}(s') \right] \tag{2.29}$$

where $P(s'|s,a)$ is the probability of transitioning from state s to s' upon taking action a, and $R(s,a,s')$ is the instantaneous reward for that transition.

Extending the above concept to the action value function, the Bellman equation can be written as:

$$Q^{\pi}(s,a) = \sum_{s' \in \mathcal{S}} P(s'|s,a) \left[R(s,a,s') + \gamma \sum_{a' \in \mathcal{A}} \pi(a'|s') Q^{\pi}(s',a') \right] \tag{2.30}$$

The Bellman equations are the basis of dynamic programming, such as policy evaluation[3] and policy iteration[4], which seek to determine the optimal policy that may be followed to maximize cumulative rewards.

An optimal policy (π^*) is one that maximizes the value function $V^{\pi}(s)$ for all states $s \in \mathcal{S}$. The Bellman optimality equations characterize the optimal value functions V^* and Q^*:

$$V^*(s) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P(s'|s,a) \left[R(s,a,s') + \gamma V^*(s') \right], \tag{2.31}$$

$$Q^*(s,a) = \sum_{s' \in \mathcal{S}} P(s'|s,a) \left[R(s,a,s') + \gamma \max_{a' \in \mathcal{A}} Q^*(s',a') \right]. \tag{2.32}$$

2.4.2 Deep Reinforcement Learning Algorithms

Before introducing any RL or DRL algorithm, it is important to note that these algorithms are classified based on two distinctions: whether they are model-based or model-free and whether they are on-policy or off-policy. Furthermore, these distinctions define how RL agents function and learn optimal policies in their environments. Regarding policy distinction, on-policy methods improve the same policy under which the data is collected, whereas off-policy methods learn from the experiences of other policies, and they are more stable but less sample-efficient, offering greater flexibility in utilizing previously collected experiences, and they are generally more sample efficient but lack stability. As for model distinction, model-based strategies involve planning actions by modeling future possibilities with an environmental model, while model-free methods, such as the algorithm that will be introduced in this subsection, Q-learning,

[3] Computing value functions under a specific policy.

[4] Iteratively improving policies using the value functions.

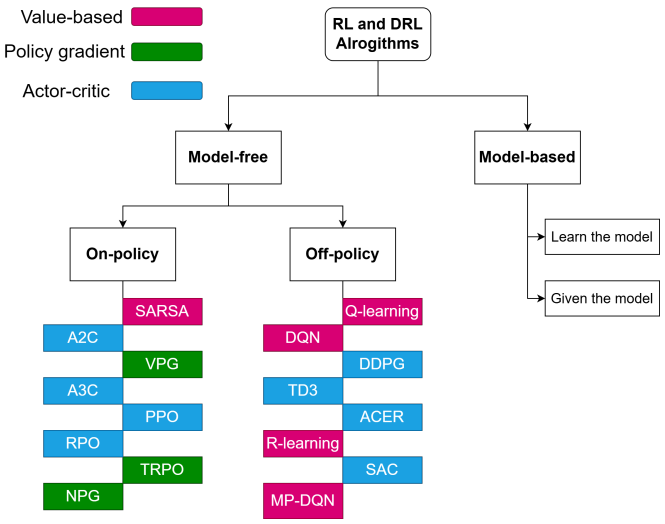


FIGURE 2.12: Classification of RL algorithms based on model and policy distinctions.

do not require a model and instead depend directly on the collected experience. Therefore, Q-Learning is regarded as the most fundamental model-free, off-policy algorithm, relying on the Bellman equation to iteratively determine the optimal action-value function. Figure 2.12 shows the classification of RL algorithms based on the distinctions mentioned above. The model-free algorithms can further be divided into value-based methods, policy gradient methods, and actor-critic methods, as shown in the figure. In the following we will introduce the three types and some of their respective algorithms.

2.4.2.1 Value-based Methods

Value-based methods learn a value function (state or state-action value) to derive an optimal policy. The policy is implicit and often greedy with respect to the learned value function. Value-based methods are model-free and off-policy algorithms. They are known for their simplicity and sample efficiency, but they can be unstable and have difficulty learning policies with high-dimensional or continuous action spaces. The most common value-based methods are Q-learning, Deep Q-Networks (DQN), and their variants.

2.4.2.1.1 Q-Learning Q-learning is a model-free and off-policy algorithm that directly approximates the optimal action-value function $Q^*(s, a)$, and is the basic algorithm in value-based methods. Q-learning is considered one of the first RL algorithms proposed and even today it takes advantage of being the most straightforward, sample-efficient, and guaranteed convergence technique in ideal conditions. Q-learning iteratively updates estimates of $Q(s, a)$ using the Bellman optimality equation (Equation 2.32).

Q-learning iteratively approximates Q^* using Temporal Difference (TD) learning. At each step t, the algorithm updates the Q-value estimate as [**62**]:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[R_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right] \tag{2.33}$$

The new Q-value is then can be written as:

$$Q_{\text{new}}(s_t, a_t) = (1 - \alpha) Q(s_t, a_t) + \alpha \left[R_t + \gamma \max_{a'} Q(s_{t+1}, a') \right]. \tag{2.34}$$

here α is the learning rate. Q-learning uses ϵ-greedy policy as an exploration-exploitation trade-off controlling technique, which selects the action with the highest Q-value at each step with a probability of $1 - \epsilon$ and chooses a random action with a probability of ϵ. In the first episode, ϵ starts at 1 and gradually decreases to 0.

2.4.2.1.2 Deep Q-Network (DQN) As Q-Learning achieves its high outcomes only in discrete and tabular problems, it faces challenges solving the large or continuous state/action space problems. The Q-table size increases in a way that it can hardly store or update Q-table values; also, Q-learning can not generalize, especially in environments that have raw pixel data or other high-dimensional input. Google DeepMind proposed DQN in 2013 to overcome Q-learning limitation [**63**]. It was the first algorithm combining Q-learning and DNNs to approximate the action-value function, $Q(s, a)$, which is the foundation of the DRL concept. By using DNNs, the agent can handle high-dimensional state spaces and generalize from raw input data, such as images.

DQN was first tested on Atari games (process images); thus, it uses CNN architectures with convolutional layers to process the input states, which are screenshots of the game, and then fully connected layers to process the features extracted by the convolutional layers to estimate the Q-values. Furthermore, DQN employs a replay memory, also known as experience replay, to disrupt the correlation between successive experiences and stabilize training. The replay memory stores past experiences. During training, minibatches of data are randomly selected from the replay buffer. This technique helps to decorrelate data and provides a more diversified set of training events.

To further help stabilize training, DQN employs a target network that is updated less frequently than the main network. This network is used to calculate the target Q-values for the Bellman equation. The target network's Q-values are updated according to the following equation:

$$y_t = R_t + \gamma \max_{a'} Q(s'_t, a'; \theta') \tag{2.35}$$

Here max $\max_{a'}$ is the maximum over all possible next actions, i indicates that this calculation is done for each individual sample in the minibatch, and $Q(s'_t, a'; \theta')$ is the Q-value for the next state s'_i and action a', parameterized by θ'. The loss function used in DQN is the MSE (see Subsection 2.3.2.2.2) between the predicted Q-values and the target Q-values as follows:

$$\mathcal{L}_\theta = \frac{1}{N} \sum_{i=1}^{N} (Q(s_i, a_i; \theta) - y_i)^2 \tag{2.36}$$

Here N is the batch size, and $Q(s_i, a_i; \theta)$ is the predicted Q-value by the main network and y_i is the target Q-value. DQN algorithm is shown in Algorithm 1. DQN has

Algorithm 1 DQN Algorithm

```
Initialize the main network Q(s, a; θ) and the target network Q(s, a; θ') with random θ and θ'.
for each episode do
    Initialize the state s.
    for each step do
        Select an action a following the epsilon-greedy policy.
        Take the action a, receive the reward R and observe the next state s'.
        Store the experience (s, a, R, s') in the replay memory.
        Sample a mini-batch of experiences from the replay memory.
        Compute the target Q-values using the target network.
        Update the main network using Equation 2.36.
        Periodically update the target network weights θ' ← θ.
    end for
end for
```

many variants and extensions, such as Double DQN (DDQN), Dueling DQN, and Multi-Pass DQN, which aim to improve the stability and performance of the original DQN algorithm.

2.4.2.2 Policy Gradient Methods

Policy gradient methods directly optimize a parameterized policy using gradient ascent to maximize expected cumulative rewards. These methods have the capability to efficiently manage continuous action spaces, which are very essential for some tasks like robotics, as the latter do not have discrete actions. Further, these methods use stochastic policies for better exploration by inducing randomness in the decision-making process. In addition, they directly concentrate on optimizing the policy without necessitating an approximation of the value function. However, they inherit significant limitations. One of them is their high variance, which requires the use of variance-reduction approaches to assure the stability of the learning process. Aside from the high variance, policy gradient

techniques are sample-inefficient because they rely on on-policy data only, which limits the reuse of previously acquired data and increases processing needs.

2.4.2.2.1 Vanilla Policy Gradient (VPG) As Q-learning presents the foundation for value-based RL algorithms, the policy gradient methods are based on the VPG. VPG [**64**], also known as the REINFORCE algorithm, is a foundational on-policy RL algorithm that directly optimizes a parameterized policy to maximize expected cumulative rewards. VPG uses a stochastic policy, which is a probability distribution over actions given a state s and parameterized by θ; therefore, this policy can be noted as $\pi(a|s;\theta)$. Let T be the length of the episode, and τ denote the trajectory of experiences, which is a sequence of transitions $(s_0, a_0, R_0, \ldots, s_T)$. As with all RL algorithms, the goal is to maximize the expected discounted return; therefore, the objective function can be written as:

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}\gamma^t R_t\right] \tag{2.37}$$

Here $\tau \sim \pi_\theta$ means that the trajectory τ is collected by following the policy π_θ.

The gradient of $J(\theta)$ is derived using the likelihood ratio trick:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}\nabla_\theta \log \pi(a_t|s_t;\theta)\cdot G_t\right] \tag{2.38}$$

Given the equations above, VPG Algorithm steps are as follows:

- Use the current policy π_θ and collect trajectories τ.
- For each timestep t, calculate G_t:

$$G_t = \sum_{k=t}^{T}\gamma^{k-t} r_k \tag{2.39}$$

- Average over all trajectories and timesteps to estimate the gradient:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi(a_t|s_t;\theta)\cdot G_t^i \tag{2.40}$$

- Perform gradient ascent to update the policy parameters:

$$\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta) \tag{2.41}$$

- Repeat until convergence.

Despite the simplicity of VPG, it has some drawbacks, such as high variance and slow convergence, which can be addressed by using advanced DRL algorithms.

2.4.2.3 Actor-Critic Methods

The actor-critic method is a DRL technique that integrates value-based approaches with policy gradient approaches. This architecture consists of two components: the actor (the policy network) and the critic (the value network) [**65**]. Actor-critic approaches are more stable and sample-efficient than pure policy gradient methods (e.g., REINFORCE) because they use value functions to reduce policy gradient variance. Furthermore, they are more adaptable than value-based algorithms due to their capacity to handle continuous action spaces. However, they are difficult to execute and require careful hyperparameter tuning. In this architecture, the actor selects an action by taking the current state as an input, whereas the critic examines the actor's actions.

In more detail, the **actor** determines the agent's behavior. First, in each training episode, the agent explores and interacts with the environment through the actor's policy network. Following the critic's feedback, the actor modifies its policy parameters to maximize total discounted reward. The actor learns a policy that improves over time so that the agent can perform better in future interactions with the environment.

Next, the **critic** takes the actions selected by the actor (if the critic learns action-value functions; otherwise, it takes only the current state) along with the current state as inputs, and then outputs the value functions. It learns either the state-value function, $V^{\pi}(s)$, or the action-value function, $Q^{\pi}(s, a)$, to determine how good it is to be at a state or to take an action under the current policy. The critic uses the TD error, defined as $\delta_t = R_t + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$. The TD error, in such a case, gives a general measure of the difference between predicted and actual returns, which guides the improvements of both the critic's value estimates and the actor's policy. The joint performance of the actor and critic creates a cycle where they rely on each other, where the actor updates its policy according to the critic's evaluation, whereas the critic keeps on refining its value function estimates. This architecture is used to handle both discrete and continuous action spaces.

2.4.2.3.1 Deep Deterministic Policy Gradient (DDPG) DDPG [**66**] is a model-free, off-policy actor-critic algorithm designed for continuous action spaces. It extends DQN to handle tasks requiring high-dimensional continuous actions, such as robotics and control systems. DDPG, like DQN, uses an experience replay to store transitions in a replay buffer and uses target networks for both the actor and critic,

which are updated via soft updates. DDPG is based on the Deterministic Policy Gradient (DPG), which gives a mathematical framework for computing policy gradients in deterministic policy contexts. According to the DPG theorem, the expected cumulative reward's gradient can be written as:

$$\nabla_h J(\mu) = \mathbb{E}_{s\sim\rho^\mu}\left[\nabla_a Q^\mu(s,a|\theta^Q)\big|_{a=\mu(s|\theta^\mu)}\nabla_h\mu(s|\theta^\mu)\right] \tag{2.42}$$

where $J(\mu)$ is the expected cumulative reward, $\mu(s)$ is the deterministic policy, ρ^μ is the state distribution under policy μ, and θ^Q and θ^μ are the weights for the critic and actor DNNs, respectively.

DDPG integrates DPG with deep neural networks (DNNs). The actor outputs a deterministic action $a_t = \mu(s_t)$ by taking the current state s as input, while the critic estimates the action-value function $Q(s,a)$ using both the state and action. For exploration, noise sampled from a stochastic process $\mathcal{N}$ is added to the policy: $\mu^*(s_t) = \mu(s_t) + \mathcal{N}$. The actor is updated via gradient ascent to maximize the expected Q-value using an optimizer (e.g., Adam):

$$\nabla_h J(\mu) \approx \nabla_a Q(s,a|\theta^Q)\big|_{a=\mu(s)}\nabla_h\mu(s|\theta^\mu) \tag{2.43}$$

The critic minimizes the MSE between predicted and target Q-values:

$$y_i = R_i + \gamma Q'(s_i, \mu'(s_i|\theta^{\mu'})|\theta^{Q'}) \tag{2.44}$$

$$L = \frac{1}{N}\sum_i (y_i - Q(s_i,a_i|\theta^Q))^2 \tag{2.45}$$

Target networks are softly updated using the equation below:

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta' \tag{2.46}$$

where τ is a DDPG hyperparameter normally set to a value less than 10^{-3}. DDPG pseudocode is presented in Algorithm 2.

2.4.2.3.2 Proximal Policy Optimization (PPO) PPO [**67**] is a DRL algorithm based on an actor-critic architecture and is an on-policy algorithm. When training an agent with PPO, the agent collects trajectories of experiences, $\tau \in \mathcal{T}$, where $\mathcal{T}$ represents the number of trajectories collected per training iteration, using its latest stochastic policy $\pi_\theta(a_t|s_t)$ with parameters θ. After collecting trajectories, the actor and critic DNNs are trained on these experiences, and their parameters—along with the policy—are updated. In PPO, the actor DNN takes the current environment observation as input and outputs either:

Algorithm 2 DDPG Algorithm

Initialize main networks with weights θ^Q and θ^μ.
$\theta^{Q'} \leftarrow \theta^Q$ and $\theta^{\mu'} \leftarrow \theta^\mu$
for each episode **do**
  Initialize random noise process $\mathcal{N}$.
  Initialize the state s.
  for each step **do**
    Select an action $a = \mu^"(s)$.
    Take the action a, receive the reward R and observe the next state s'.
    Store the experience (s, a, R, s') in the replay buffer.
    Sample a mini-batch of experiences from the replay buffer.
    Compute y_i for each sample according to Equation 2.44.
    Update the critic network using Equation 2.45.
    Update the actor network using Equation 2.43.
    Update Target networks weights according to Equation 2.46.
  end for
end for

- Discrete action space: Logits for each action.
- Continuous action space: Mean and log standard deviation of a Gaussian distribution.

The critic shares the same state input as the actor and estimates the state-value function $V(s_t)$. Like other policy gradient methods, PPO estimates the policy gradient and updates parameters via stochastic gradient ascent. The gradient estimator is defined as:

$$\nabla_\theta J(\theta) = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \hat{A}(s_t) \right] \tag{2.47}$$

where $\hat{A}(s_t)$ is the advantage function which can be calculated using the Generalized Advantage Estimation (GAE) [**68**]. For a time horizon T, the GAE is written as:

$$\hat{A}(s_t) = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda_{GAE})^{T-t-1}\delta_{T-1} \tag{2.48}$$

with $\delta_t = R_t + \gamma V_\omega(s_{t+1}) - V(s_t|\omega)$, where λ_{GAE} is a hyperparameter, and V_ω is the critic's value function parameterized by ω. PPO algorithm employs a clipped surrogate objective function to prevent excessive policy updates, which is formulated as follows:

$$J_t^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right] \tag{2.49}$$

Here r_t is the probability ratio with $r_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$, where $\pi_{\theta_{\text{old}}}$ denotes the old policy. $clip(.)$ is used to prevent the probability ratio from going outside the range $[1-\epsilon, 1+\epsilon]$, where ϵ is a PPO clipping hyperparameter that usually takes a value between 0.1 and

0.3. The loss function for training the critic is:

$$\mathcal{L}_t^{VF}(\omega) = \mathbb{E}_t \left[(V_\omega(s_t) - V_{\text{target}}(s_t))^2 \right] \tag{2.50}$$

where $V_{\text{target}} = R_t + \gamma V_\omega(s_{t+1})$ The critic weights are updated following the gradient descent optimization using any gradient optimizer:

$$\omega \leftarrow \omega - \alpha_{\text{critic}} \nabla_\omega \mathcal{L}_t^{VF}(\omega) \tag{2.51}$$

PPO's total loss combines the clipped objective, value function loss, and entropy regularization:

$$\mathcal{L}^{\text{PPO}}(\theta) = \hat{\mathbb{E}}_t \left[J_t^{CLIP}(\theta) - c_1 L_t^{VF}(\omega) + c_2 S \right], \tag{2.52}$$

Here, the coefficients $c1$ and $c2$ are positive coefficients weighting the value function loss and entropy bonus S, which is used to encourage exploration and is given as $S = -\sum \pi_\theta(a|s) \log \pi_\theta(a|s)$, respectively. The actor is updated via gradient ascent using any optimizer following the equation below:

$$\theta \leftarrow \theta + \alpha_{\text{actor}} \nabla_\theta L(\theta) \tag{2.53}$$

After collecting T trajectories, the DNNs are updated over K training epochs, each epoch using a batch of size N. The full PPO pseudocode is detailed in Algorithm 3.

Algorithm 3 PPO Algorithm

```
Initialize policy π_θ and value function V_ω
for iteration = 1, 2, ... do
    Collect trajectories using policy π_θold
    Compute advantages Â(s_t)
    for epoch = 1, 2, ..., K do
        Sample minibatches of size N from collected trajectories
        Update θ by maximizing L^PPO(θ) using Equations 2.52 and 2.53
        Update ω by minimizing L^VF(ω) using Equations 2.50 and 2.51
    end for
end for
```

2.5 Conclusion

This chapter introduced the history of AI evolution. It traced AI beginnings through its stages, including the first and second winters, and arrived at the revolution it has driven in recent decades. Next, we introduced ML and discussed its key concepts, subfields, and use cases depending on the available data, along with their evaluation metrics. Proceeding further, we moved towards the scope of DL, where we

covered its basic concepts such as neurons, MLPs, cost functions, learning rates, optimizers, and various architectures and their potentials. Next, we explored RL by defining the MDP framework and its components, followed by an introduction to DRL as a field combining RL and DL through the use of DNNs to enhance robustness and efficiency. Thereafter, we presented RL and DRL approaches based on two main distinctions and outlined some of their associated algorithms.

In the next chapter, we will present IoT networks and their demands in B5G networks, followed by UAV networks, their architectures, and their applications in wireless communications. Then, we will discuss the RIS technology and its use in NOMA-UAV networks. Finally, we will present the research gaps in the literature and the contributions of this thesis.

Chapter 3

AI-Driven NOMA-UAV Networks for IoT Communications

The rapid growth of IoT has led to a new era of hyperconnected ecosystems, with billions of devices that require seamless, dependable, and low-latency communication. Traditional OMA schemes and static infrastructure struggle to meet the scalability, energy efficiency, and dynamic coverage needs of IoT networks, especially in remote, densely populated, or disaster-prone places. NOMA and UAVs work together to overcome these challenges, where NOMA improves spectral efficiency as it serves numerous users via shared resources, whereas UAVs enable flexible, on-demand aerial connectivity. AI integration takes these networks to the next level and optimizes UAV flight paths, manages NOMA resource distribution, handles interference in real-time, and adjusts to the changing needs of IoT devices, the state of the channels, and energy limitations.

3.1 IoT Communication Systems

IoT is a networked ecosystem of physical items equipped with sensors, actuators, and connectivity modules that are linked together via the Internet infrastructure to allow autonomous data exchange and coordinated activities among distributed smart sensors [**69**]. This approach reduces the requirement for direct human interaction while also enabling fresh application paradigms. The convergence of heterogeneous technologies inside IoT frameworks, including wireless communication, edge computing, and ML, intends to improve data-driven decision-making processes in the industrial, environmental, and societal sectors.

3.1.1 IoT Requirements

For IoT to satisfy technical, functional, and application-specific requirements, a robust architecture is necessary. Ultra-high data speeds (Tbps), sub-millisecond latency for mission-critical operations, energy-efficient protocols, and scalable frameworks to support billions of devices are among the technical requirements for IoT systems [**70**]. Interoperability, context-aware computing, and efficient data management are functional priorities. Application-specific needs vary from healthcare, which requires great reliability and security, and smart cities, which rely on seamless communication, to industrial automation that necessitates microsecond-level precision. To improve processing and latency, IoT networks use edge computing, cloud analytics, and hybrid connections (cellular, Wi-Fi, satellite). To provide security and privacy, strong encryption, multi-factor authentication, and zero-trust architectures are required. Energy restrictions spur innovation in ultra-low-power electronics, energy harvesting, and dynamic power management. Together, these components ensure that IoT solutions are scalable, adaptable, and secure across industries.

1. *Core Technical Requirements:*
    - Some IoT applications demand very high data rates, possibly Tbps (terabits per second) (e.g. holographic communications and high-quality video streaming) [**70**].
    - Many IoT applications require sub-millisecond latency for real-time responsiveness and reliability [**70, 71**].
    - IoT devices, particularly those in remote or hard-to-reach locations, must operate for extended durations on limited power, which necessitates energy-efficient protocols and technologies [**72**].
    - IoT systems must support a massive number of connected devices, potentially up to billions, while maintaining performance and reliability [**71, 72**].
    - With the growth of IoT devices market, data security and user privacy become crucial [**72**].
2. *Functional Requirements* [**72**]:
    - IoT devices and systems must communicate and collaborate seamlessly, regardless of origin or technology.
    - IoT systems should be able to understand and respond to the context in which they operate, including user preferences, environmental conditions, and other relevant factors.

- Efficient data collection, storage, processing, and analysis are important for IoT applications to derive insights and make informed decisions.

3. *Network Architecture Requirements:*
   - To reduce latency, IoT systems must adopt edge computing, which involves data processing and analysis closer to the source [**70**].
   - IoT systems often rely on cloud platforms for data storage, processing, and analysis, requiring secure and efficient connectivity [**72**].
   - To achieve dependable connectivity, IoT devices may require hybrid network settings combining cellular, Wi-Fi, and satellite communication technologies [**70, 73**].
4. *Security and Privacy Requirements* [**72**]:
   - IoT devices and systems have to use strong authentication and authorization algorithms to limit access to authorized users.
   - Encrypt data sent between IoT devices and other systems to avoid unauthorized access.
   - IoT systems must use secure communication protocols to maintain data integrity and confidentiality.
5. *Energy Efficiency Requirements:* [**73**]
   - IoT devices often operate with constrained power, which requires low-power communication protocols and energy-efficient hardware design.
   - IoT devices may benefit from energy harvesting to extend operational time and reduce battery replacement frequency.
   - IoT systems should deploy power management techniques to optimize energy use and minimize consumption.

3.1.2 IoT Enabling Technologies

3.1.2.1 5G IoT

1. *mmWave* [**74**]: mmWave operates in the 30 GHz to 300 GHz range, enabling gigabit mobile broadband with peak throughput up to 10 Gb/s. mmWave enables connectivity for massive machine-type communications (mMTC) by providing high capacity and supporting diverse communication patterns.

2. *URLLC* [**74**]: uRLLC targets high reliability (minimal packet loss) and low latency ($<$ 1 ms to seconds) for applications like remote healthcare, industrial automation, and autonomous vehicles.

3. *NOMA* [**14**]: the first MA technique candidate for IoT is NOMA, where it satisfies the IoT massive connectivity requirements and increases throughput.

4. *AI* [**75**]: AI allow for more optimized and efficient management for IoT networks in the context of 5G and beyond.

5. *Cloud and Edge computing* [**76**]: are two complementary technologies that play crucial roles in IoT ecosystem, each addressing different aspects of data processing and resource management. Cloud computing provides a centralized infrastructure for storing and processing large volumes of data. It offers scalable and flexible resources, allowing IoT devices to offload computationally intensive tasks to remote servers. Edge computing brings data processing closer to the source, typically at the edge of the network where IoT devices are located. It enables real-time data processing and reduces the need for data transmission to the cloud, enhancing privacy and reducing latency.

6. *Software defined networking (SDN)* [**77**]: SDN is a networking paradigm that decouples the control plane (network control logic) from the data plane (packet forwarding), enabling centralized management and programmability of network resources. This allows for dynamic, real-time allocation of resources, intelligent traffic control, and adaptive routing decisions based on network conditions. SDN provides a global view of the network, supports network slicing, and enhances flexibility, manageability, and efficiency in IoT environments.

7. *Network slicing* [**77**]: is a technique enabled by SDN that creates virtualized, isolated subnetworks (slices) within a physical network infrastructure. Each slice can be customized with dedicated resources, routing policies, and QoS configurations to meet specific IoT application requirements.

3.1.2.2 6G IoT

1. *THz communications*: Wireless links in terahertz (THz, 300 GHz$-$3 THz) and sub-THz ($100 - 300$ GHz) frequency bands are pivotal for 6G networks. Next-generation systems can satisfy the demanding 6G QoS requirements thanks to THz communication. Target device densities greater than 107 nodes per m^3 (mMTC) and end-to-end communication latency less than 1 ms (aligned with mURLLC) are

defined by these criteria, which are particularly true in three critical areas: hyper-dense device scalability, sub-millisecond latency, and ultra-reliable connectivity [**78**].

2. *Space-air-ground-underwater communications* [**71**]: 6G networks are expected to provide a complete communication system that can cross conventional land boundaries and offer smooth three-dimensional coverage in the air (such as unmanned aerial systems), on land, and in the sea (such as autonomous underwater vehicles). Advanced IoT ecosystems with a cell-free design split into four hierarchical operational tiers—spaceborne, airborne, terrestrial, and subaquatic—are intended to be supported by this paradigm shift toward ubiquitous connectivity.

3. *Edge intelligence* [**79**]: sometimes known as Edge AI, integrates AI with edge computing to deploy AI capabilities at the network's edge. This method integrates data processing and decision-making close to data sources, decreasing reliance on centralized cloud infrastructures. Edge Intelligence improves IoT systems by processing data locally, resulting in faster response times (critical for real-time applications), improved data privacy (localized processing reduces data exposure), lower bandwidth consumption (less data transmitted to the cloud), and energy efficiency (optimized edge resource usage). These benefits improve the scalability, dependability, and usability of IoT solutions across industries.

4. *Reflecting Intelligent Surfaces (RIS)*: are programmable metasurfaces that reflect incident electromagnetic waves, improving wireless communication performance. By altering the phase and amplitude of reflected signals, RIS can direct energy to specific receivers, reduce interference, and improve signal quality [**80**]. By allowing intelligent beamforming, better coverage, and more spectrum efficiency, RIS technology is poised to change wireless networks. In IoT applications, RIS can improve signal propagation, coverage, and connectivity in difficult conditions.

5. *Blockchain* [**81**]: Blockchain, a decentralized and distributed ledger technology, ensures transparent, secure, and tamper-proof data storage by recording transactions in sequential, cryptographically linked blocks. It ensures accountability, traceability, and data integrity within IoT ecosystems. Blockchain enhances privacy and trust in IoT applications by facilitating secure data sharing, automated device identification, and immutable verification procedures. It also facilitates transactions and protects data exchanges between connected devices, resulting in IoT networks that are resilient and effective.

6. *Digital twin framework* [**82**]: A digital twin is a dynamic virtual replica of a physical item that is developed and maintained by continuous bidirectional data exchange between the real thing and its computational model. This synchronization

ensures that the digital representation precisely represents the real-world object's state, behavior, and environmental interactions in real time, allowing for applications such as predictive maintenance, operational optimization, and immersive simulations. The paradigm accomplishes cyber-physical coherence by seamlessly integrating IoT sensors, edge computing infrastructure, and cloud-based analytics.

3.1.3 IoT Applications

From smart cities optimizing traffic and energy use to healthcare enabling remote patient monitoring and telemedicine, IoT applications transform how systems operate. In agriculture, precision farming boosts yields through sensor-driven irrigation, while industrial IoT (IIoT) enhances automation and predictive maintenance. Consumer applications like smart homes prioritize convenience and security, and environmental monitoring tackles pollution and climate challenges. These use cases highlight ability the ability of IoT to enhance efficiency, reduce costs, and enable data-driven decision-making. Different IoT applications are presented in Table 3.1

3.2 UAV Networks for IoT

UAVs have emerged as a disruptive facilitator for IoT and provide unparalleled flexibility, scalability, and dynamic coverage in wireless communications. This section investigates the dual aspects of UAV integration into IoT environments and demonstrates their ability to address connection issues while they also resolve intrinsic constraints. First, it investigates the role of UAVs in improving wireless communication infrastructure through agile deployment and line-of-sight benefits. It then digs into UAV channel modeling and covers the specific air-to-ground propagation dynamics that affect performance. The discussion then switches to practical UAV-enabled IoT applications, such as precision agriculture, disaster response, and smart city monitoring. Finally, the part looks into the integration of NOMA and UAVs to improve spectral efficiency and enormous connectivity in IoT networks.

3.2.1 UAV Types and Energy Consumption

3.2.1.1 UAV Types

The terms *UAV* and *drone* broadly describe autonomous or remotely piloted robotic aircraft capable of multipurpose operations. UAV selection for specific applications requires systematic evaluation of two interdependent factors:

Table 3.1: IoT Applications

IoT Application	**Definition**	**Benefits**	**Ref**
Smart Grid	An advanced electricity supply network that utilizes IoT technologies to enhance the efficiency, reliability, and sustainability of electricity distribution and consumption. It incorporates intelligent two-way communication between utilities and consumers, utilizing various IoT devices, sensors, and analytics to manage energy consumption in real-time.	Enables real-time monitoring of energy flow, enhance energy efficiency, supports integration of renewables (solar, wind), predictive maintenance, improves scalability to accommodate growing energy demands.	[**83**]
Smart Cities	Refers to urban areas that leverage IoT technologies to optimize public systems and services. This includes the use of IoT devices for various applications such as car parking management, waste management, street lighting, and emergency control within the city environment.	Increased sustainability; and provide better quality life where residents enjoy improved services, safety, and accessibility to information, contributing to a higher quality of living in smart cities.	[**84**]
Transportation	The integration of IoT technologies into transportation systems to create intelligent and interconnected infrastructures that enhance mobility and logistics. This includes the use of IoT devices such as sensors and connected vehicles that collect and analyze data to improve the efficiency and safety of transportation networks.	real-time tracking, traffic management, safety enhancements, predictive maintenance.	[**84**]
Healthcare	Network of connected devices and applications that communicate health data to healthcare providers, patients, and caregivers over the internet. These devices can monitor and manage various health parameters in real-time, allowing for enhanced patient care and better resource management.	Remote patient monitoring, better patient outcomes, improved care management, enhanced safety and comfort, data-driven insights	[**85**]
Agriculture	The application of IoT technologies in farming and agricultural practices to enhance productivity, efficiency, and resource management. This includes the use of sensors, connected devices, and data analytics to monitor environmental conditions, crop health, and livestock management	Monitor soil conditions, weather patterns, and crop health in real-time; resource management; predictive analytics; livestock monitoring.	[**85**]
Military	Refers to the integration of interconnected devices, sensors, and systems that collect, transmit, and analyze data to enhance military operations, logistics, and decision-making.	Enhanced situational awareness, optimized maintenance and logistics, personnel health and performance monitoring using wearable IoT devices,... etc.	[**86**]

1. **Platform capabilities**, including onboard sensor suites, physical dimensions, payload capacity, and energy storage limitations.
2. **Aerodynamic performance**, such as operational altitude thresholds, hover stability, and maneuverability in constrained environments [**87**].

UAVs may be classified according to multifaceted taxonomies reflecting their operational, structural, and functional diversity. Figure 3.1 illustrates UAV classification based on altitude and wing type. Key classification criteria include [**88**]:

- **Manufacturing Provenance**: Geographic origin and regulatory compliance (e.g., domestically produced vs. imported systems).
- **Operational Context**: User domain (civil, commercial, or defense) and mission profile (surveillance, payload delivery, environmental monitoring, or reconnaissance).
- **Physical Specifications**:
  - *Size*:micro ($\leq$ 100g), very small (100g-2kg), small (2kg-25kg), medium (25kg-150kg), or large-scale ($\geq$ 150kg) [**89**].
  - *Aerodynamic Design*: Fixed-wing, rotary-wing, airship, or balloon.
- **Altitude**: High-altitude platforms (HAPs) or low-altitude platforms (LAPs).
- **Reusability**: Reusable UAVs versus expendable UAVs.

3.2.1.2 UAV Energy Consumption Models

The limited onboard energy reserves of UAVs present a critical challenge for their communication systems, necessitating precise energy efficiency optimization. A foundational step toward this goal involves developing accurate energy consumption models that account for two distinct operational demands. The first pertains to communication-related energy consumption, which encompasses power allocated to signal processing, circuitry, and radiofrequency amplification. These components mirror the energy dynamics of terrestrial communication systems, and established terrestrial models may remain applicable in scenarios where such energy costs dominate [**90**]. The second and more distinctive challenge arises from propulsion energy consumption, a unique requirement for UAVs to sustain aerial mobility and positional stability. Propulsion energy demands often exceed communication-related consumption by orders of magnitude, particularly for larger UAVs with substantial payloads. Historically, propulsion

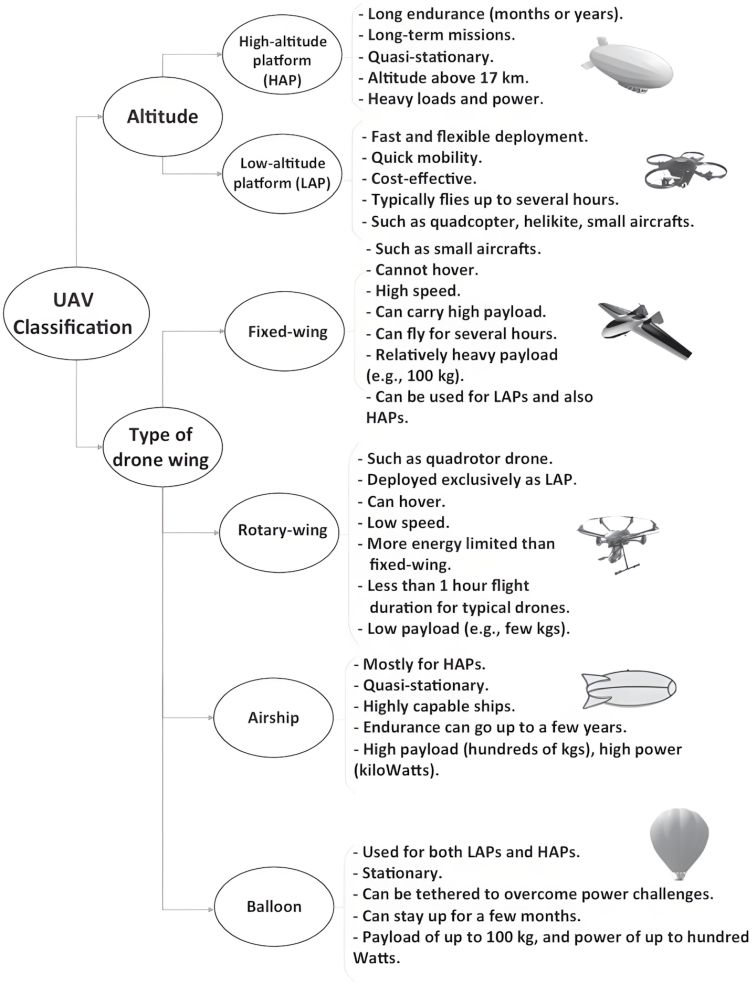


FIGURE 3.1: UAV classification based on altitude and wing type.

energy modeling received far less attention than its communication counterpart, despite its disproportionate impact on overall system efficiency. Recent advancements, however, have introduced closed-form analytical models tailored to the mechanical and aerodynamic realities of UAV architectures. For fixed-wing platforms, these models derive energy formulations based on lift-dependent aerodynamic forces, while rotary-wing systems employ thrust-based relationships governed by rotor dynamics. Such frameworks enable system designers to harmonize communication performance with flight endurance, thereby informing energy-aware protocol development and trajectory optimization strategies essential for practical deployment [**90**].

3.2.1.2.1 Energy Consumption for Fixed-Wing UAV For a fixed-wing UAV operating in steady straight-and-level flight over a duration of T seconds, the propulsion

power consumption admits a closed-form analytical expression [**91**]:

$$\bar{E} = \int_0^T c_1 ||\mathbf{v}(t)||^3 + \frac{c_2}{||\mathbf{v}(t)||} \left(1 + \frac{||\mathbf{a}(t)||^2}{g^2} - \frac{(\mathbf{a}(t) \cdot \mathbf{v}(t))^2}{||\mathbf{v}(t)||^2 g^2}\right) dt + \frac{1}{2} m (||\mathbf{v}(T)||^2 - ||\mathbf{v}(0)||^2) \tag{3.1}$$

In Equation 3.1, $\mathbf{v}(t)$ and $\mathbf{a}(t)$ represent the instantaneous velocity and acceleration vectors of the UAV, respectively. The coefficients c_1 and c_2 encapsulate aerodynamic and structural parameters of the aircraft, including its weight, wing surface area, air density, and other platform-specific constants.

3.2.1.2.2 Energy Consumption for Rotary-Wing UAV For a rotary-wing UAV operating at a velocity of V (m/s), the propulsion power consumption admits a closed-form analytical expression [**92**]:

$$P(V) = P_0 \left(1 + \frac{3V^2}{U_{\text{tip}}^2}\right) + P_{\text{ind}} \left(1 + \frac{V^4}{4v_0^4} - \frac{V^2}{2v_0^2}\right)^{1/2} + \frac{1}{2} d_0 \rho s A V^3 \tag{3.2}$$

Here, The propulsion power model incorporates two constants: P_0, which corresponds to the blade profile power in hover mode, and P_{ind}, which defines the induced power under stationary rotor operation. Key aerodynamic parameters include U_{tip} (rotor blade tip speed), v_0(mean induced velocity in hover mode), d_0(fuselage drag ratio), and s (rotor solidity). Environmental and structural factors are quantified through ρ (air density) and A (rotor disc area).

3.2.2 UAV in Wireless Communications

UAVs are used in wireless communication systems due to their unmatched dynamic mobility and flexibility. These airborne platforms enable rapid provisioning of connectivity in infrastructure-scarce locations, which makes them crucial for emergency response operations, rural broadband deployment, and temporary network augmentation during large-scale events. The integration of UAVs into wireless networks has opened up new possibilities for extending coverage and increasing capacity. UAVs, which serve as airborne base stations (ABS), reconfigurable relays, or mobile user equipment (UE), provide adaptive network topologies that adjust dynamically to real-time service demands and propagation conditions.

A key feature of UAV-enabled communications is their ability to establish reliable line-of-sight (LoS) links with terrestrial users, which considerably reduces signal attenuation and enhances potential data speeds. This capability is critical in areas where conventional terrestrial infrastructure is limited due to obstacles (e.g., urban canyons,

dense foliage) or catastrophic failures (e.g., natural disasters). Furthermore, UAVs can be equipped with advanced technologies such as mmWave and massive MIMO. These advancements position UAVs as critical enablers for next-generation networks that demand high reliability, ultra-low latency, and spectral efficiency.

3.2.2.1 UAV Operational Modes in Wireless Networks

All UAVs, regardless of type, can be equipped with wireless interfaces, enabling numerous applications in wireless communications. These applications are primarily categorized into three operational modes: UAV as an Aerial Base Station (ABS), UAV as a relay, or UAV as User Equipment (UE). Each mode will be discussed in detail below.

3.2.2.1.1 Aerial Base Station UAVs can act as ABSs with mounted BS capabilities and provide on-demand wireless connectivity for a variety of temporary applications. UAVs that serve as ABSs offer significant advantages over terrestrial networks, such as 3D deployment with unrestricted mobility, the ability to reach remote or inaccessible locations, and adaptability to scenarios that require wireless communications for limited durations [**93**]. ABS is shown in Figure 3.2 [**94**]. However, 3GPP has established prerequisites for UAVs to act as ABSs in 5G networks, which can be summarized as follows [**94**]:

- The UAV must incorporate BS functionality that aligns with 3GPP 5G technical standards.
- The system shall demonstrate autonomous navigation to predefined operational zones or support human-operated guidance for mission adaptability, followed by stable hovering to maintain uninterrupted network service delivery.
- The ABS must establish secure wireless backhaul connectivity to the 5G core network, which enables full BS operationalization as defined by core network protocols, including network registration, spectrum allocation, and service provisioning.
- The ABS must undergo authentication and authorization processes mandated by 5G security frameworks (e.g., Authentication and Key Agreement protocols) to verify compliance with regulatory standards and network policies prior to operational deployment.

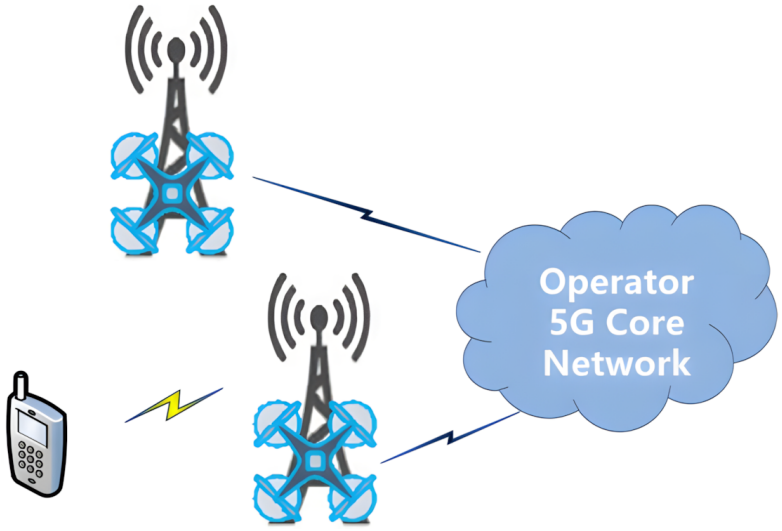


FIGURE 3.2: UAV acting as ABS.

3.2.2.1.2 Relay The second application for UAVs in wireless environments involves their deployment as relay stations to establish bidirectional communication links between transmitters and receivers. UAV relays primarily address two critical challenges: (1) the extension of ground network coverage and (2) the mitigation of physical obstructions—such as elevated terrain or urban infrastructure—that disrupt line-of-sight (LOS) signal propagation between terrestrial communication endpoints. Within this operational paradigm, UAVs function as intelligent transceivers that perform two sequential tasks: (1) reception of data packets from terrestrial devices and (2) multi-hop forwarding of this information to designated network endpoints [**87**]. Figure 3.3 shows a UAV action as an aerial relay [**94**].

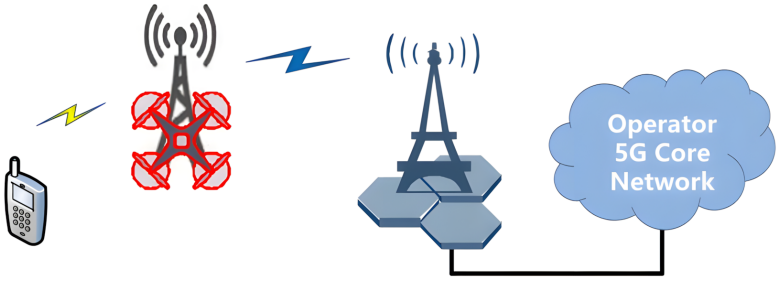


FIGURE 3.3: UAV acting as aerial relay.

3.2.2.1.3 Mobile Equipment To maximize UAV operational utility and exploit their full application potential, these systems must interface with existing wireless infrastructure by functioning as aerial UEs. This capability enables UAVs to support diverse services—including surveillance, delivery, military operations, and beyond—that require seamless integration with terrestrial networks. Surveillance applications impose

stringent technical demands, where aerial UEs necessitate ultra-low latency, high reliability, and sustained high-speed uplink connectivity from both terrestrial networks and ABS. However, current cellular architectures face inherent limitations in fully supporting aerial UEs, as their design prioritizes ground users whose operational, mobility, and traffic patterns differ fundamentally from those of aerial counterparts. Aerial UEs experience predominantly LoS propagation conditions with ground BSs, which amplifies interference from terrestrial infrastructure due to unblocked signal pathways [**93**]. UAVs operate under stringent onboard energy reserves, necessitating optimized power management strategies for sustained flight and communication. Plus, aerial UEs exhibit greater mobility flexibility than ground users, with three-dimensional omnidirectional movement capabilities that challenge conventional handover protocols and network resource allocation [**93**]. These unique attributes necessitate substantial modifications to existing wireless frameworks, as the integration of cellular-connected aerial UEs into existing wireless architectures introduces novel technical challenges, including interference mitigation, energy-efficient protocol design, and dynamic network optimization.

3.2.2.2 UAV Channel Modeling

UAV communication systems primarily involve three distinct link categories: (1) ground-BS (GBS)-to-Air connectivity, (2) Air-to-ground (A2G) transmissions, and (3) Air-to-Air (A2A) communication channels. Among these, A2A links exhibit unique propagation characteristics due to their operation in unobstructed airspace at moderate distances, where terrestrial curvature and environmental blockages are negligible [**95**]. Under such conditions, the A2A channel is commonly modeled using the free-space path-loss model, which simplifies interference and attenuation predictions in aerial networks.

3.2.2.2.1 Large-scale Propagation Models

3.2.2.2.1.1 Free-Space Path Loss Model The free-space model is widely adopted in the literature to quantify path loss between aerial nodes in unobstructed environments. This model assumes electromagnetic wave propagation through a vacuum (or ideal free space) without obstacles or reflections that could distort the signal. The free-space path loss is expressed as:

$$L(d) = \rho_0 d^{-2} \tag{3.3}$$

where $L(d)$ denotes the path loss at distance d, and ρ_0 represents the free-space path loss at a reference distance (typically 1 meter). This model is particularly applicable to A2A communication channels, rural areas, or scenarios involving high-altitude UAVs.

3.2.2.2.1.2 Log-Distance Alpha-Beta Model In contrast to the free-space model, height-dependent path loss models account for the impact of UAV altitude on signal propagation. These models are particularly relevant for A2G communication scenarios where UAVs operate at low altitudes, which results in distinct path loss characteristics. The log-distance alpha-beta model is expressed as [**96**]:

$$L_{AB}(d)\,[\text{dB}] = \alpha \cdot 10\log_{10}(d) + \beta + X_\sigma \tag{3.4}$$

where $L_{AB}(d)$ denotes the path loss at distance d, α and β are environment-dependent parameters, and X_σ represents shadowing with standard deviation σ. The parameters α, β, and σ depend on UAV altitude h_u:

$$\alpha(h_u) = \max\left(p_{a_1} + p_{a_2}\log_{10}(h_u),\, 2\right), \tag{3.5}$$

$$\beta(h_u) = p_{\beta_1} + p_{\beta_2}\log_{10}\left(\min(h_u, h_{\text{FSPL}})\right), \tag{3.6}$$

$$\sigma(h_u) = p_{\sigma_1} + p_{\sigma_2}\log_{10}\left(\min(h_u, h_{\text{FSPL}})\right). \tag{3.7}$$

Here, h_{FSPL} is the altitude threshold where free-space propagation ($\alpha = 2.0$) dominates. The coefficients p_{a_1}, p_{a_2}, p_{β_1}, p_{β_2}, p_{σ_1}, and p_{σ_2} are environment-specific [**96**].

3.2.2.2.1.3 Probabilistic LoS Channel Model For low-altitude UAVs in A2G scenarios, models that combine LoS and non-line-of-sight (NLoS) occurrence probabilities are more appropriate.

1. ***Elevation Angle-Dependent Probabilistic LoS Model*** A probabilistic LoS model dependent on elevation angle θ is given by [**97**]:

$$\mathbf{P}(\text{LoS} \mid \theta) = \frac{1}{1 + a\exp\left(-b\left[\theta - c\right]\right)}, \tag{3.8}$$

   where θ is the elevation angle between the UAV and ground node, and a, b, and c are S-curve parameters.

2. ***3GPP TR 36.777 Model*** This model characterizes path loss for UAVs in LTE networks, considering both LoS and NLoS scenarios in urban, suburban, and rural environments. The LoS probability, P_{LOS}, depends on UAV altitude h_{UT} and horizontal distance $d_{2\text{D}}$ between transmitter and receiver [**98**]. The path loss models for different environments are as follows:

- **Rural Environments** (UAV altitude: $10\,\text{m} \leq h_{UT} \leq 300\,\text{m}$, $d_{\text{2D}} \leq 10\,\text{km}$):

$$PL^{3GPP}_{\text{rural-LOS}} = [\max(23.9 - 1.8\log_{10}(h_{UT}),\, 20)]\log_{10}(d_{\text{3D}}) + 20\log_{10}\left(\frac{40\pi f_c}{3}\right), \tag{3.9}$$

$$PL^{3GPP}_{\text{rural-NLOS}} = \max(PL_{\text{rural-LOS}},\, -12 + (35 - 5.3\log_{10}(h_{UT}))\log_{10}(d_{\text{3D}}) + 20\log_{10}\left(\frac{40\pi f_c}{3}\right)\Big). \tag{3.10}$$

- **Urban Environments** (UAV altitude: $22.5\,\text{m} \leq h_{UT} \leq 300\,\text{m}$, $d_{\text{2D}} \leq 4\,\text{km}$):

$$PL^{3GPP}_{\text{urban-LOS}} = 28.0 + 22\log_{10}(d_{\text{3D}}) + 20\log_{10}(f_c), \tag{3.11}$$

$$PL^{3GPP}_{\text{urban-NLOS}} = -17.5 + (46 - 7\log_{10}(h_{UT}))\log_{10}(d_{\text{3D}}) + 20\log_{10}\left(\frac{40\pi f_c}{3}\right). \tag{3.12}$$

- **Suburban Environments** (UAV altitude: $22.5\,\text{m} \leq h_{UT} \leq 300\,\text{m}$, $d_{\text{2D}} \leq 4\,\text{km}$):

$$PL_{\text{suburban-LOS}} = \max\Big\{PL',\, 30.9 + (22.25 - 0.5\log_{10}(h_{UT}))\log_{10}(d_{\text{3D}}) + 20\log_{10}(f_c)\Big\}, \tag{3.13}$$

$$PL^{3GPP}_{\text{suburban-NLOS}} = \max\Big\{PL_{\text{suburban-LOS}},\, 32.4 + (43.2 - 7.6\log_{10}(h_{UT})) \times \log_{10}(d_{\text{3D}}) + 20\log_{10}(f_c)\Big\}. \tag{3.14}$$

3. ***COST-HATA Modified Model:*** A modified version of the COST-HATA model, based on the COST-2100 framework, is proposed in [**99**] to characterize path loss (PL) for UAVs in rural, urban, and suburban environments. The model accounts for operating frequency, UAV-ground node distance, and UAV altitude. The modified COST-HATA PL is defined as:

$$PL^{\text{cost}}_{\text{mod}}\,[\text{dB}] = \begin{cases} f'_{COST}(d_{\text{2D}} = d_{\text{BK}}) + \Delta_{\text{CF}}(h_{\text{BS}}), 0 \leq d_{\text{2D}} \leq d_{\text{BK}},\ 0 \leq h_{\text{BS}} \leq h_{\text{CF}}, \\ f'_{COST}(d_{\text{2D}}) + \Delta_{\text{CF}}(h_{\text{BS}}), \quad d_{\text{BK}} < d_{\text{2D}} < d_{\text{CF}},\ 0 \leq h_{\text{BS}} \leq h_{\text{CF}}. \end{cases} \tag{3.15}$$

Here, $f'_{COST}(d_{\text{2D}}) = a_1 + a_2\log_{10}(f_c) + a_3\log_{10}(h_{\text{BS}}) - \Gamma(h_{\text{UE}}) + [a_4 + a_5\log_{10}(h_{\text{BS}})]\log_{10}(d_{\text{2D}})$ derives from the original COST-HATA model. The terms d_{BK} (breakpoint distance), d_{CF} (cutoff distance), h_{BS} (UAV altitude), h_{UE} (user equipment

TABLE 3.2: Discussed PL Models for UAV Networks

PL Model	**Scenario**	**Frequency**	**UAV Altitude**	**Horizontal distance**
Log-distance alpha-beta	Rural	800 MHz	1.5-120 m	$1-22$ km of kilometers
3GPP TR36.777	Urban, suburban, rural	800 MHz & 2000-2600 MHz	$22.5-300$ m $10-300$ m	$d_{2D} \leq 4$ km (urban & suburban) $d_{2D} \leq 10$ km (rural)
Elevation Angle-Dependent	Dense urban, suburban, urban	2000 MHz	Around 5km	$\approx 1.7km$ dense urban $\approx 3.5km$ suburban $\approx 2.5km$ urban
COST Hata Modified	Urban, suburban, rural	785 MHz & 2160 MHz	$30-1000$ m	Max 70 km

height), and h_{CF} (cutoff height) are defined in [**99**], with constants a_1 to a_5 provided empirically. The altitude-dependent correction factor is:

$$\Delta_{\mathrm{CF}}(h_{\mathrm{BS}}) = b(f_c)\left(\log_{10}(h_{\mathrm{BS}})\right)^2, \tag{3.16}$$

where $b(f_c)$ is a frequency-dependent coefficient.

Table 3.2 summarizes the discussed PL models for UAVs networks.

3.2.2.2.2 Small-scale Propagation Models Small-scale fading is frequently modeled using Rayleigh fading, particularly in NLOS environments. Recently, Rician fading has become increasingly prevalent for scenarios involving a dominant LoS component, while Nakagami-m fading is also used to model small-scale fading.

3.2.3 UAVs for IoT: Use Cases and Applications

UAVs, particularly low-altitude ones, play an important role in addressing IoT network challenges and enabling solutions for diverse IoT applications. Below, we outline UAV use cases in IoT communications and demonstrate how they enhance IoT network performance.

3.2.3.1 Precision Agriculture

Precision agriculture is being revolutionized by UAVs because they make sophisticated crop monitoring and resource management possible. In addition to producing

precise field maps for soil analysis and topography-based planning, UAVs use high-resolution and multispectral imaging to assess crop health, identify disease, and diagnose nutrient deficiencies. They also enhance irrigation by pinpointing water-stressed areas and enabling precise agrochemical delivery to reduce waste and environmental impacts. Through the analysis of growth trends and the planning of strategic field operations to enhance crop positioning, UAVs can also assist with yield estimation. In order to enhance adaptive agricultural operations, they also keep an eye on environmental factors like temperature and humidity [**100**].

3.2.3.2 Disaster Management and Emergency Response

UAVs integrated with IoT systems and edge computing can significantly enhance disaster management and emergency response by enabling [**101**–**103**]:

- Real-time data collection from IoT sensors (e.g., environmental sensors, wearable health monitors) in disaster-affected areas.
- Use ABSs to restore cellular/Wi-Fi connectivity (e.g., post-earthquake).
- Autonomous navigation and coordination of UAV swarms for search-and-rescue operations.
- Edge analytics to process data locally, reducing latency and bandwidth usage.

3.2.3.3 Smart Cities and Infrastructure

UAVs enhance smart city IoT through flexible device deployment, mobile data hubs, and real-time monitoring, minimizing fixed infrastructure; where they optimize resource use by repositioning IoT devices for events or disasters [**104, 105**]. Applications of UAVs in smart cities include infrastructure inspection (bridges, power lines), disaster damage assessment, environmental monitoring (air quality, traffic), medical deliveries, public surveillance, construction surveys, and urban agriculture, demonstrating their versatility in urban ecosystems.

3.2.3.4 Healthcare and Medical Services

UAVs change modern healthcare systems by enabling the delivery of crucial medical supplies (vaccines, blood products, and transplant organs) to remote or disaster-affected areas. They also improve telemedicine infrastructure by delivering diagnostic

equipment, increasing disaster response mechanisms with real-time geospatial mapping, and speeding up emergency medical deliveries [**106**]. Furthermore, UAVs improve medical logistics by streamlining organ transplantation networks and pharmaceutical supply chains, as well as supporting population-level health initiatives and allowing medical professionals to get simulation-based training through immersive crisis scenarios. Emerging innovations, such as emergency airborne patient delivery and remote diagnostic platforms, provide access to healthcare for underserved communities, ensuring equal access to time-sensitive medical interventions in logistically hard contexts [**106**].

3.2.3.5 Connectivity Provision in Remote Areas

UAVs serve as agile platforms to establish temporary or permanent connectivity in regions where terrestrial infrastructure is unavailable or compromised. UAVs can serve as ABSs or relays in remote locations to improve network coverage, sustain emergency communication services, and enable IoT applications [**93, 103**]. UAV deployments are important in post-disaster situations, when they reestablish key communication links between emergency responders and affected people while also enabling real-time coordination.

3.3 RIS and STAR-RIS

This section builds on what was discussed about IoT communications and UAV-assisted wireless networks. It then introduces RIS and STAR-RIS as enabler tools that can help solve the problems that these systems have with scalability, coverage, and energy use. IoT needs very stable, low-latency connections for billions of devices, and UAVs need dynamic signal enhancement in NLoS environments. RIS and STAR-RIS fill in these gaps by intelligently manipulating electromagnetic (EM) waves. They do this by passively beamforming and controlling signals all around, which opens up new areas of energy and spectrum sustainability.

3.3.1 RIS Technology

RIS represents a foundational technology for realizing programmable and dynamically controllable wireless propagation environments. An RIS comprises a low-cost adaptive metasurface designed to integrate seamlessly into architectural surfaces such as walls, ceilings, or building facades. It is a quasi-passive EM modulator that dynamically

changes the phase, amplitude, and polarization of incident radio waves and uses software-defined control signals, which allows for real-time channel optimization [**107**]. A defining characteristic of RIS technology is its post-deployment reconfigurability, which allows adaptive adjustments to wireless propagation paths without physical modifications. An RIS is a two-dimensional metasurface composed of densely arrayed subwavelength unit cells that are usually spaced at intervals less than the operational wavelength [**108**]. These unit cells include tunable elements such as PIN diodes, varactor diodes, liquid crystals, and graphene-based components, which engineers fabricate with advanced techniques to obtain exact EM response control. RISs enable the establishment of virtual LoS communication links between users and their associated BS/ABS, and they avoid signal attenuation caused by physical obstructions. This feature is crucial for coverage extension in the mmWave and THz frequency bands, which are very susceptible to blockage effects due to limited diffraction and penetration qualities [**109**].

3.3.1.1 RIS Architecture and Operation

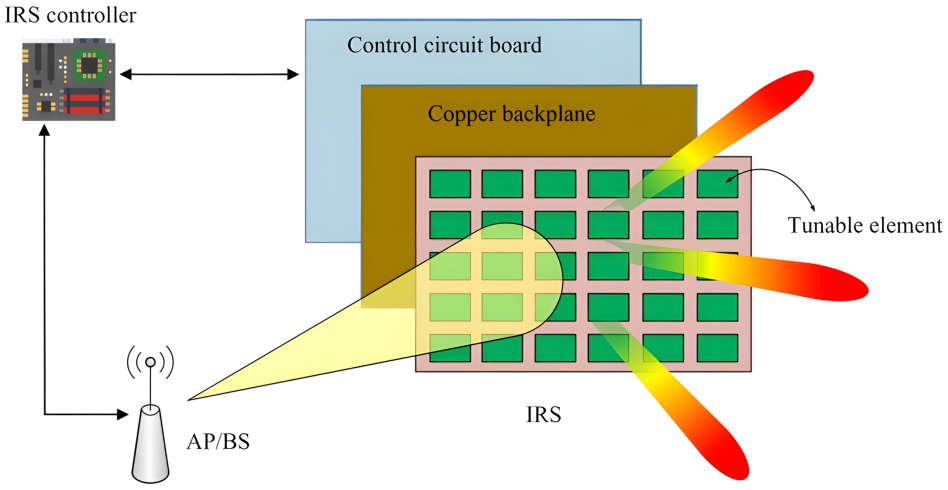


FIGURE 3.4: RIS architecture.

In wireless communication, the channel is generally time-varying due to the mobility of the transmitter/receiver as well as the surrounding objects, thus calling for real-time tunable response of IRS based on the channel variation. To this end, IRS elements need to be manufactured with dynamically adjustable reflection coefficients and IRS is required to connect to the wireless network to learn the exterior communication environment to enable its real-time adaptive reflection [**109**]. Fig. 3.4 illustrates one typical architecture of IRS, which consists of three layers and a smart controller [**109**]. The first/outside layer is composed of a large number of tunable/reconfigurable metallic patches printed on a dielectric substrate to directly manipulate incident signals. In the second/intermediate layer, a copper plate is usually employed to minimize the signal

energy leakage during IRS's reflection. It is followed by the third/inside layer that is a control circuit board responsible for exciting the reflecting elements as well as tuning their reflection amplitudes and/or phase-shifts in real time. Moreover, the reflection adaptation is triggered and determined by a smart controller attached to each IRS, which can be implemented via field-programmable gate array (FPGA). The IRS controller also acts as a gateway to communicate with other network components (e.g., BSs/APs and user terminals) through wired or wireless backhaul/control links [**109**]. The operation of an RIS can, in general, be split into two phases that are executed periodically based on the coherence time of the environment [**107**]. (1) Control and programming phase. During this phase, the necessary environmental information for configuring the operation of the RIS is estimated, and the RIS is configured for subsequent operation. (2) Normal operation phase. During this phase, the RIS is configured already and assists the transmission of other devices throughout the network. To reconfigure IRS elements for highly controllable reflection, the main adopted approach in practical implementation is using electronic devices (e.g., positive-intrinsic-negative (PIN) diodes, field-effect transistors (FETs), or micro-electromechanical system (MEMS) switches) due to its fast response time, low reflection loss as well as relatively low energy consumption and hardware cost.

Besides tuning the phase shift, additional control of the reflection amplitude of each IRS element provides more flexibility in reshaping the reflected signal to achieve various communication objectives effectively. This also offers a flexible way to trade-off between the hardware cost and reflection performance in practice, as amplitude control is generally of lower cost to implement as compared to phase control [**109**].

However, While tuning the reflection coefficient continuously is beneficial for optimizing the communication performance, it is practically difficult to implement since higher-resolution reflecting elements require not only increased cost but also more complex hardware design. In practice, to further reduce the hardware cost and design complexity, only discrete phase-shift control or discrete amplitude control may be implemented [**109**]. IRS with discrete phase-shift control only, where for each reflecting element, only the phase shift can be tuned while the reflection amplitude is set to its maximum value of one. IRS with discrete amplitude control only, where for each reflecting element, only the reflection amplitude can be tuned while the phase shift is set to be a constant.

3.3.1.2 RIS Signal Model

3.3.1.2.1 Network Components The system architecture comprises a transmitter (Tx), RIS, and receiver (Rx), where the Tx, typically a BS with M antennas that focuses

energy toward the RIS or user at a carrier frequency f_c (e.g., sub-6 GHz or mmWave). The RIS consists of N passive elements arranged in a planar grid, with each element n applying a configurable phase shift ϕ_n and optionally an amplitude coefficient β_n, controlled via a software-defined interface linked to the BS or edge server. The Rx, a UE with K antennas (often simplified to K =1), is located in a NLOS region relative to the Tx, necessitating RIS-assisted signal reflection to establish reliable communication. The composite channel comprises three key links: Tx-RIS link, RIS-Rx link, and direct Tx-Rx link. The Tx-RIS link models the signal propagation from the BS to the RIS, with channel gain $h_{\text{BS-RIS}}$ and phase shift $\theta_{\text{BS-RIS}}$. The RIS-Rx link captures the signal reflection from the RIS to the Rx, with channel gain $h_{\text{RIS-Rx}}$ and phase shift $\theta_{\text{RIS-Rx}}$. The direct Tx-Rx link represents the signal path from the BS to the Rx without RIS assistance, with channel gain $h_{\text{BS-Rx}}$ and phase shift $\theta_{\text{BS-Rx}}$, and is often weak or blocked.

3.3.1.2.2 Signal Model The RIS is represented by a diagonal matrix Φ where each diagonal entry corresponds to the complex reflection coefficient of an element:

$$\Phi = \text{diag}(\beta_1 e^{j\phi_1}, \beta_2 e^{j\phi_2}, \ldots, \beta_N e^{j\phi_N}) \tag{3.17}$$

where each diagonal entry $\beta_n e^{j\phi_n}$ corresponds to the complex reflection coefficient of element n, with $\phi_n \in [0, 2\pi)$ controlling the phase shift and $\beta_n \in [0, 1]$ (for passive RIS) or $\beta_n \in [0, \beta_{\max}]$ (for active RIS) governing the amplitude. The end-to-end signal propagation model defines the received signal at the UE as:

$$y = \mathbf{G}\Phi\mathbf{H}x + \mathbf{D}x + n \tag{3.18}$$

where $x = f_s$ (with s being the transmitted symbol), $\mathbf{G}$ and $\mathbf{H}$ represent the RIS-Rx and Tx-RIS channel matrices, $\mathbf{D}$ is the direct Tx-Rx channel, and $n \sim \mathcal{CN}(0, \sigma^2 I)$ is additive noise. In the simplified single-antenna case ($M = K = 1$), this reduces to $y = (\mathbf{g}^T\Phi\mathbf{h} + d)x + n$, where $\mathbf{h} \in \mathbb{C}^{N\times 1}$ and $\mathbf{g} \in \mathbb{C}^{N\times 1}$ are the Tx-RIS and RIS-Rx channel vectors, respectively, and $d \in \mathbb{C}$ is the direct channel scalar.

3.3.2 STAR-RIS Technology

Conventional RISs restrict Tx and Rx to the same spatial region (half-space spatially restricted environment), a constraint that severely limits deployment flexibility in scenarios requiring coverage on both sides of the surface. To address this limitation, STAR-RISs were introduced [**110**]. As illustrated in Fig. 3.5, a STAR-RIS divides the incident wireless signal at each element into two components:

- **Reflected Signal**: Propagates within the incident signal's spatial region (reflection space),
- **Transmitted Signal**: Propagates to the opposite spatial region (transmission space) [**111**].

This dual-space operation eliminates the half-space restriction, enabling full-space coverage and bidirectional service for users on either side of the surface.

3.3.2.1 STAR-RIS Physical Model

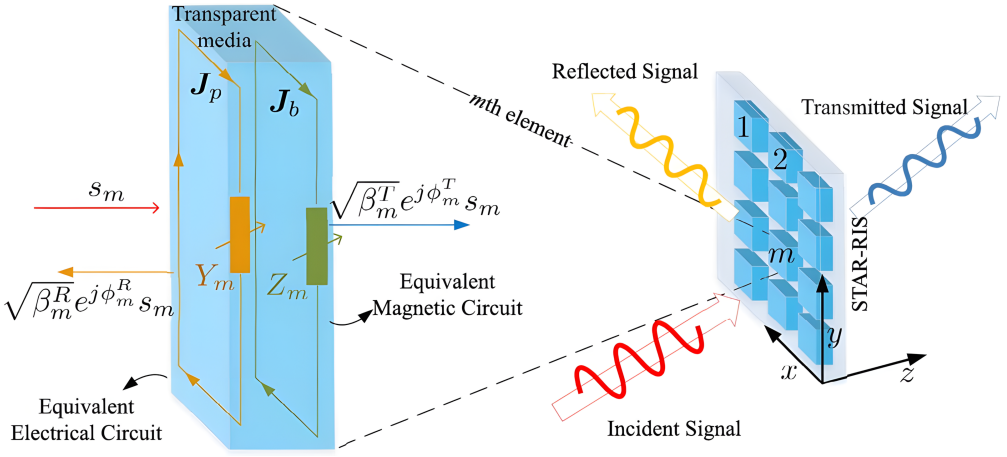


FIGURE 3.5: An illustrated schematic representation of a STAR-RIS.

Fig. 3.5 [**110**] depicts a schematic representation of a simultaneous transmitting and reflecting reconfigurable intelligent surface (STAR-RIS). Under the field equivalence principle, the transmitted and reflected signals generated by STAR-RIS elements under incident excitation correspond to waves emitted by time-varying surface equivalent electric currents J_p and magnetic currents J_b (collectively termed bound currents). The magnitude and spatial distribution of these currents depend on the incident narrowband signal s_m and the locally averaged surface impedances Y_m (electric) and Z_m (magnetic). Assuming polarization alignment between transmitted and reflected signals at the mth element, the outputs are defined as [**110**]:

$$s_m^T = T_m s_m, \quad s_m^R = R_m s_m, \tag{3.19}$$

where T_m and R_m represent the transmission and reflection coefficients, respectively. For passive STAR-RIS elements, energy conservation imposes the constraint [**110**]:

$$|T_m|^2 + |R_m|^2 \leq 1. \tag{3.20}$$

Electromagnetic theory dictates that phase delays in transmitted and reflected fields depend on Y_m and Z_m. As shown in Fig. 3.5, the reconfigurability of each element arises from dynamic adjustments to surface impedances, with T_m and R_m expressed as [**110**]:

$$T_m = \frac{2 - \eta_0 Y_m}{2 + \eta_0 Y_m} - R_m, \quad R_m = -\frac{2(\eta_0^2 Y_m - Z_m)}{(2 + \eta_0 Y_m)(2\eta_0 + Z_m)}, \tag{3.21}$$

where η_0 denotes the free-space impedance. Independent control of transmitted and reflected signals requires the integration of magnetic currents, a capability absent in conventional single-layered RISs constrained by symmetric radiation patterns. The proposed hardware model resolves this limitation through joint modulation of electric and magnetic currents.

For practical implementation in wireless systems, the coefficients are reformulated in polar coordinates [**110**]:

$$T_m = \sqrt{\beta_m^T} e^{j\phi_m^T}, \quad R_m = \sqrt{\beta_m^R} e^{j\phi_m^R}, \tag{3.22}$$

where $\beta_m^T, \beta_m^R \in [0, 1]$ satisfy $\beta_m^T + \beta_m^R \leq 1$, and $\phi_m^T, \phi_m^R \in [0, 2\pi)$ govern the phase shifts for transmitted and reflected signals.

3.3.2.2 STAR-RIS Signal Model

As discussed in the previous section, STAR-RISs are capable of independently controlling the transmitted and reflected signals, which introduces additional DoFs that can be exploited. To characterize this unique feature, for a STAR-RIS having M elements, let s_m denote the signal incident upon the mth element. After being reconfigured by the corresponding transmission and reflection coefficients, the signals transmitted and reflected by the mth element are given by $\left(\sqrt{\beta_m^T} e^{j\theta_m^T}\right) s_m$ and $\left(\sqrt{\beta_m^R} e^{j\theta_m^R}\right) s_m$, respectively. In particular, $\sqrt{\beta_m^T}, \sqrt{\beta_m^R} \in [0, 1]$ and $\theta_m^T, \theta_m^R \in [0, 2\pi)$ characterize the amplitude and phase shift adjustments imposed on the incident signal facilitated by the mth element during transmission and reflection, respectively. In STAR-RIS, the transmission and reflection amplitude coefficients must obey the law of energy conservation, i.e., the sum of the energies of the transmitted and reflected signals has to be equal to the incident signal's energy, which leads to the sum of β_m^T and β_m^R should be equal to one [8]. Accordingly, by adjusting the amplitude coefficients for transmission and reflection, each STAR-RIS element can be operated in full transmission mode (referred to as T mode), full reflection mode (referred to as R mode), and simultaneous transmission and reflection mode (referred to as T&R mode) [**112**].

3.3.2.3 STAR-RIS Operating Protocols

As proposed in [**111**], STAR-RISs support three operational protocols: Energy Splitting (ES), Mode Switching (MS), and Time Switching (TS), as detailed below and illustrated in Fig. 3.6.

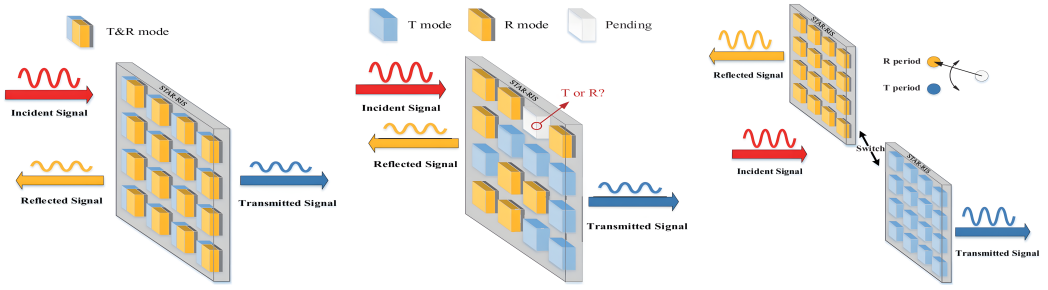


FIGURE 3.6: STAR-RIS Operating Protocols (a) ES. (b) MS. (c) TS.

3.3.2.3.1 Energy Splitting (ES) In ES protocol, all STAR-RIS elements operate in simultaneous transmission and reflection (T&R) mode, where the incident signal energy splits into transmitted and reflected components with ratio $\beta_m^t : \beta_m^r$. The transmission and reflection coefficient matrices are:

$$\Theta_t^{ES} = \text{diag}\left(\sqrt{\beta_1^t}e^{j\theta_1^t}, \ldots, \sqrt{\beta_M^t}e^{j\theta_M^t}\right), \quad \Theta_r^{ES} = \text{diag}\left(\sqrt{\beta_1^r}e^{j\theta_1^r}, \ldots, \sqrt{\beta_M^r}e^{j\theta_M^r}\right),$$

with $\beta_m^t, \beta_m^r \in [0,1]$, $\beta_m^t + \beta_m^r = 1$, and $\theta_m^t, \theta_m^r \in [0, 2\pi)$, $\forall m \in \mathcal{M}$. ES achieves significant flexibility in system design through joint optimization of transmission/reflection coefficients. However, the large parameter space increases configuration overhead between the BS and STAR-RIS.

3.3.2.3.2 Mode Switching (MS) MS partitions STAR-RIS elements into two disjoint groups: M^t elements operate in T mode, and M^r elements in R mode ($M^t + M^r = M$). The coefficient matrices are:

$$\Theta_t^{MS} = \text{diag}\left(\sqrt{\beta_1^t}e^{j\theta_1^t}, \ldots, \sqrt{\beta_M^t}e^{j\theta_M^t}\right), \quad \Theta_r^{MS} = \text{diag}\left(\sqrt{\beta_1^r}e^{j\theta_1^r}, \ldots, \sqrt{\beta_M^r}e^{j\theta_M^r}\right),$$

where $\beta_m^t, \beta_m^r \in \{0,1\}$, $\beta_m^t + \beta_m^r = 1$, and $\theta_m^t, \theta_m^r \in [0, 2\pi)$. MS constitutes a restricted ES case with binary amplitude coefficients, sacrificing full-dimension beamforming gain for reduced implementation complexity.

RIS Architecture/Protocol	Technical Characteristics
Conventional Reflecting RIS	• Operates within a single hemisphere (half-space coverage) • Signal energy dissipation from non-cooperative users in opposite hemisphere • Low-complexity amplitude/phase control (binary states)
STAR-RIS	• Full-space coverage through simultaneous transmission/reflection • Enhanced flexibility via independent transmission/reflection control • Protocol-dependent complexity (ES/MS/TS implementations)
MS	• Element-level mode selection (transmit/reflect) • Simplified implementation via time-division element grouping • Suboptimal beamforming gain due to partial element utilization • Decoupled amplitude-phase constraints
ES	• Continuous energy-split transmission/reflection coefficients • Maximum beamforming flexibility with joint coefficient optimization • High configuration overhead from dual parameter space
TS	• Full-element temporal mode alternation • Simplified coefficient decoupling through orthogonal time slots • Demands precise time synchronization between network entities

TABLE 3.3: Comparative Analysis of RIS Architectures and Operational Protocols.

3.3.2.3.3 Time Switching (TS) TS protocol periodically alternates all elements between T mode and R mode across orthogonal time slots (termed T and R periods). Let $\lambda^t, \lambda^r \in [0,1]$ denote time fractions allocated to T and R periods ($\lambda^t + \lambda^r = 1$). The coefficient matrices are:

$$\Theta_t^{TS} = \text{diag}\left(e^{j\theta_1^t}, \ldots, e^{j\theta_M^t}\right), \quad \Theta_r^{TS} = \text{diag}\left(e^{j\theta_1^r}, \ldots, e^{j\theta_M^r}\right),$$

with $\theta_m^t, \theta_m^r \in [0, 2\pi)$. TS decouples transmission/reflection coefficient design, simplifying optimization. However, it imposes stringent time synchronization requirements between network entities, resulting in increased hardware complexity [**112**].

Table 3.3 summarizes the three protocols supported by STAR-RIS and RIS [**113**].

3.4 DRL for NOMA-UAV networks

The integration of RIS, NOMA, and UAV networks has the potential to alter the boundaries of wireless communication, but it presents significant optimization issues due to their dynamic, multidimensional interactions. DRL is an important part of making

RIS phase-shift configurations, NOMA user clustering, and UAV trajectory planning work together. It can make decisions on its own and in real time. While UAVs provide agile, LoS links; RIS extends signal coverage and spectral efficiency; and NOMA supports massive connectivity, DRL can bridge these technologies as it learns optimal policies under stochastic channel conditions, energy constraints, and mobility-driven uncertainties. DRL turns theoretical synergies into intelligence that can be used, which makes RIS-NOMA-UAV systems the building blocks of the next generation of adaptive networks.

3.4.1 DRL for UAV-NOMA Networks

NOMA and UAVs have been key technologies for IoT networks in recent years due to their synergy in providing massive connectivity, enhanced spectral efficiency, on-demand network services, and extended coverage. Further incorporating DRL into these networks improves their adaptability, dynamism, resilience to network changes, and overall effectiveness. Several DRL algorithms, including DQN, DDPG, and PPO, have been proposed for UAV-NOMA systems. They have been used to optimize UAV trajectory, user clustering, power distribution, and resource management. By learning optimal policies, DRL algorithms can efficiently adapt to dynamic channel circumstances, energy limits, and mobility-related uncertainty. Table 3.4 summarizes the key DRL algorithms used in UAV-NOMA networks and their main use case in the literature.

3.4.2 DRL for RIS-Assisted NOMA-UAV Networks

The integration of RIS into UAV-NOMA networks has developed as a disruptive paradigm, combining RIS coverage extension and interference mitigation capabilities with spectrum efficiency of NOMA and UAV agility. This trinity solves important IoT concerns such as NLoS connection, dynamic user density, and energy-constrained situations, with DRL facilitating intelligent coordination of these disparate components. Through dynamic optimization of RIS phase shifts, UAV trajectories, and NOMA power allocation, DRL enhances signal-to-interference-plus-noise ratio (SINR), reduces energy consumption, and adapts to real-time network fluctuations. The DRL frameworks are used in RIS-assisted UAV-NOMA systems to jointly optimize: RIS phase-shift configurations to maximize reflected signal gains. UAV trajectory planning to balance coverage and energy efficiency. NOMA user clustering and power allocation to ensure fairness and spectral efficiency. Table 3.5 summarizes state-of-the-art DRL algorithms, and their applications. When unified, RIS programmability, non-orthogonal resource sharing of NOMA, and UAV mobility under DRL-driven autonomy unlock ultra-reliable, scalable

TABLE 3.4: Summary of DRL algorithms used in UAV-NOMA networks

Ref.	Model	Objectives	Opt. Aspects	Main Contributions	Year
[**114**]	Q-learning	Minimize long-term content delivery delay	Caching placement at the UAV, user scheduling, NOMA PA.	Proposed a framework for cache-enabling UAV NOMA networks to optimize content delivery in hotspot areas. Uses a function approximation-based algorithm (SGD+DNN) to handle large-scale networks.	2020
[**115**]	Dueling DQN	Maximize SR and fairness	3D UAV placement and NOMA PA.	Propose AdaptSky, the first framework integrating dueling network-DRL in NOMA-UAV networks for joint 3D placement and PA. Validation across sub-6GHz and mmWave spectra with LoS/NLoS channels.	2021
[**116**]	DQN	Maximize EE	Power and BW allocation for NOMA clusters, UAV flying profile	First work to combine fuzzy logic with DQN ensembles for UAV-NOMA networks. Introduces a reward function balancing EE, outage avoidance, and BW utilization.	2022
[**117**]	Mutual DQN	Maximize SR	3D UAV Trajectory, usrer clustering using K-means, and PA	Proposed a NOMA-enabled multi-UAV collaborative caching 6G network model with HAPS-assisted content exchange and channel reusing. Decomposed the NP-hard problem into two subproblems: 1) MAPPO-based collaborative caching to maximize weighted content hit ratio.2) Matching-DRL framework for joint trajectory, power, and channel optimization.	2022
[**118**]	DDPG as part of the proposed framework	Maximize total fair network throughput	3D UAV trajectory design, time allocation between wireless power transfer (DL) and data collection (UL), EE balancing solar harvesting, propulsion, and wireless power transfer.	Hybrid-powered NOMA-IoT framework addressing energy supply and massive access. Propose the FC-TDTA algorithm with dimension spread and preactivation penalty for high-dimensional action spaces.	2023
[**119**]	PPO	Minimize task completion time for UAV data collection	3D UAV trajectory, and task completion time.	Customized channel modeling via ray tracing and ML	2024
[**120**]	Multi-agent PPO (MAPPO) & Matching-DRL	Minimize the system content retrieving delay	Caching decisions, PA, 3D UAV trajectory planning, and sub-channel allocation.	Proposed a DRL-based framework for UAV-assisted NOMA networks to optimize EE. Uses DDPG to learn optimal policies for joint optimization of UAV trajectory, user clustering, and PA.	2024
[**121**]	PPO	Balance trade-offs between EE and fairness	3D UAV trajectory, NOMA PA, user association, and DL transmit power.	Introduces a multi-action space to jointly optimize multiple factors. Combines EE and Jain's Fairness Index (JFI) with a scaling factor to dynamically balance fairness and EE.	2024

IoT connectivity for applications like smart factories and disaster recovery, where adaptability and low-latency decision-making are paramount.

TABLE 3.5: Summary of DRL algorithms used in RIS-assisted UAV-NOMA networks

Ref.	Model	Objectives	Opt. Aspects	Main Contributions	Year
[**122**]	DDPG	Maximize EE	UAV transmit power allocation and RIS phase shift matrix.	Integration of NOMA with RIS-assisted multi-UAV communication.	2022
[**123**]	Hierarchical DDPG	joint energy minimization for UAV and ground users	3D UAV path planning, groud users scheduling, Active RIS parameters, and beamforming vectors for ground users.	Integration of active IRS with UAV communications. Proposed two-layered hierarchical DRL framework.	2023
[**124**]	DDQN	Maximize system capacity	2D UAV trajectory, passive RIS phase shift, and NOMA PA.	Proposed a mobile RIS-UAV system where RIS is mounted on UAVs. Joint optimization via DDQN. Energy-aware capacity maximization.	2023
[**125**]	DDPG	Maximize and balance rate and EE	2D path planning, passive RIS phase shift, and satellite beamforming.	Integrated RIS-UAV-satellite framework via NOMA.	2023
[**126**]	PPO as part of the proposed framework	Maximize long-term UL throughput	3D UAV trajectory, STAR-RIS passive beamforming, time switching allocation, NOMA PA.	Proposed a STAR-RIS-assisted UAV NOMA emergency communication network. Introduced an adaptive method to solve constrained long-term optimization problems by integrating Lagrange relaxation with PPO.	2024
[**120**]	DDQN	Maximize sercery EE	3D UAV Trajectory, power control, STAR-RIS coefficients, and UAV/STAR-RIS placement.	Proposed UAV-mounted STAR-RIS-assisted secure uplink NOMA system to counteract eavesdroppers. For static scenario, authors used iterative block coordinate descent framework with fractinal programming successive convex approximation, and they used DDQN for mobile scenario.	2024

3.5 System Models and DRL Optimization Frameworks

3.5.1 UAV-NOMA Networks for IoT communications

3.5.1.1 Network Components and Signal Model

We investigate a downlink NOMA-assisted UAV communication network. A rotary-wing UAV u, equipped with an omnidirectional antenna, traverses an IoT deployment area to serve distributed IoT devices. The trajectory of the UAV comprises N locations, each indexed by $n \leq N$, with coordinates $C_n = [x_n, y_n]$ and an altitude of H_n. The total flight duration T is divided into N time slots of equal interval $t = T/N$. IoT devices are uniformly distributed, and their horizontal positions are denoted $l_m = [x_m, y_m]$. The Euclidean distance between the UAV at C_n and IoT device m is:

$$d_{n,m} = \sqrt{\|C_n - l_m\|^2 + H_n^2} \tag{3.23}$$

The channel follows the free space path loss model as presented in 3.3, as follow:

$$h_{n,m} = \rho_0 d_{n,m}^{-2} \tag{3.24}$$

The system employs PD-NOMA to serve $K = 2$ users per subchannel, supporting $M = N \times K$ total devices. The key operational constraints in our system encompass three critical aspects. First, the **association variable** $a_{n,m}$ determines the connectivity between UAV at location n and device m, subject to the constraint κ_n, which governs the connected devices. Second, **decoding order** in SIC mandates that signals be decoded sequentially from the farthest to the nearest node, ensuring the channel gains satisfy $h_{n,K} \leq \cdots \leq h_{n,1}$. Lastly, **power allocation** follows a hierarchical structure, where power allocation factors must be assigned in increasing order, adhering to $\alpha_{n,1} \leq \cdots \leq \alpha_{n,K}$, to maintain SIC applicability. The achievable rate for device k at location n under perfect SIC is:

$$\gamma_{n,k} = BW \log_2 \left(1 + \frac{P_{tx}^n \alpha_{n,k} h_{n,k}}{\sigma^2 + P_{tx}^n h_{n,k} \sum_{i=k+1}^{K} \alpha_{n,i}} \right),$$

where σ^2 denotes noise power and BW is bandwidth. The Key Performance Indicators (KPIs) for the system are defined as follows:

- **SR**: The total achievable rate of the IoT devices, given by:

$$\text{SR} = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{n,k} \tag{3.25}$$

- **SE**: The SE, defined as the total achievable rate divided by the total BW, given by:

$$\text{SE} = \frac{\text{SR}}{BW} \tag{3.26}$$

- **EE**: The EE, defined as the ratio of the total achievable rate to the total energy consumed by the UAV and the total transmitted power, given by:

$$\text{EE} = \frac{\sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{n,k}}{\sum_{n=1}^{N} P_{tx}^n + P_{UAV}^n} \tag{3.27}$$

- **JFI**: The JFI, used to evaluate the fairness of resource allocation, is defined as:

$$I_n = \frac{\left(\sum_{k=1}^{K} \gamma_{n,k} \right)^2}{K \sum_{k=1}^{K} (\gamma_{n,k})^2} \tag{3.28}$$

3.5.1.2 Problem Formulation

The goal is to balance the trad-off between the EE of the system and the JFI, while respecting the operational constraints. To this end, we formulate the optimization problem as follows:

$$\max_{as,u_H,u_v,u_{md},P_{TX},\alpha} \sum_{n=1}^{N} EE_n + I_n \tag{3.29a}$$

$$\text{Subject to: } P_{min} \leq P_{TX}^{n} \leq P_{max}, \quad \forall n \tag{3.29b}$$

$$\sum_{m=1}^{M} as_{n,m} = K, as_{n,m} \in \{0,1\}, \quad \forall n, \forall m \tag{3.29c}$$

$$\gamma_{n,k} \geq \gamma_k^Q, \quad \forall n, \forall k \tag{3.29d}$$

$$\sum_{k=1}^{K} \alpha_{n,k} \leq 1, \quad \forall n \tag{3.29e}$$

$$c_N = c_0 \tag{3.29f}$$

$$[x_{min}, y_{min}] \leq C_n \leq [x_{max}, y_{max}], \quad \forall n \tag{3.29g}$$

$$H_{min} \leq H_n \leq H_{max}, \quad \forall n \tag{3.29h}$$

$$v_{min} \leq u_{v_n} \leq v_{max}, \quad \forall n \leq N-1 \tag{3.29i}$$

In the optimization problem presented in Equation 3.29, the constraint (3.29b) limits the transmission power, and (3.29c) ensures that the UAV serves exactly K devices at each location. To guarantee **QoS**, the data rate for each device must meet a predefined threshold γ_k^Q as defined in (3.29d). (3.29e) limits the sum of assigned power factors. Constraint (3.29f) indicates that the UAV has to return to its initial location after each flight period. Constraint (3.29g) restricts its movement within a predefined area. Constraints (3.29h) and (3.29i) limit the height and speed of the UAV. The optimization problem aims to maximize two conflicting objectives, which are EE and I. This dual-objective framework results in a multi-objective Mixed-Integer Nonlinear Programming (MINLP) problem. The MINLP problem structure inherently combines combinatorial complexity with non-convexity and demands advanced techniques for tractable solutions. To cope with this, we propose a DRL-based framework that leverages the strengths of DRL in handling high-dimensional, dynamic environments. The proposed framework is designed to learn optimal policies for joint optimization of UAV trajectory, IoT devices association, and resources allocation in NOMA-UAV IoT systems.

3.5.1.3 DRL Framework for EE and Fairness Optimization

As discussed in subsection 2.4.1.2, the DRL framework consists of an agent, environment, state space, action space, and a reward function. The agent interacts with the environment by taking actions based on the current state and receiving rewards. The goal is to learn a policy that maximizes the cumulative reward over time. In our case, the agent is responsible for optimizing the UAV trajectory, user association, and resource allocation in the NOMA-UAV system. The environment represents the system dynamics, including channel conditions, user locations, and UAV states. The reward function is designed to reflect the objectives of maximizing EE and fairness while meeting QoS requirements.

1. **State Space**: The state space is defined as the set of all possible states that the agent can observe. In this DRL framework, each state s_n from the state space $\mathcal{S}$ is composed of the following components: $(n, \gamma_k^Q, H_n, x_n, y_n, u_{v_n}, P_{tx}^n, \alpha_{n,k}, \kappa_n)$.

2. **Action Space**: Action space $\mathcal{A}$ is defined as the set of all possible actions that the agent can take. In this case, the action space consists of the following components: $(a_{H_n}, a_{v_n}, a_{md_n}, a_{P_n}, a_{\alpha_n}, a_{\kappa_n})$. The action space comprises velocity adjustments $a_{v_n} \in \mathcal{A}_v = \{\text{maintain}, +5\,\text{m/s}, -5\,\text{m/s}\}$, altitude adjustments $a_{H_n} \in \mathcal{A}_H = \{\text{maintain}, +5\,\text{m}, -5\,\text{m}\}$, movement direction $a_{md_n} \in \mathcal{A}_{md} = \{\text{left}, \text{right}, \text{forward}, \text{backward}\}$ (relative to current heading), transmit power adjustments $a_{P_n} \in \mathcal{A}_P = \{\text{maintain}, +0.1\,\text{W}, -0.1\,\text{W}\}$, power allocation (PA) factor adjustments $a_{\alpha_n} \in \mathcal{A}_\alpha = \{\text{maintain}, \pm 10^{-2}, \pm 10^{-5}, \pm 10^{-9}\}$, and IoT node association $a_{\kappa_n} \in \mathcal{A}_\kappa = \{\text{maintain}, +1, -1\}$.

3. **Reward Function**: The reward function is designed to reflect the objectives of the optimization problem, which are maximizing EE and fairness while ensuring QoS. The reward function is defined as follows:

$$R_n = \begin{cases} \frac{EE_n}{\lambda_{EE}+\lambda_I} + (I_n - \frac{1}{K}) \times \lambda_I, & \text{if } \gamma_{n,k} \geq \gamma_k^Q, \\ 0, & \text{otherwise} \end{cases} \tag{3.30}$$

   where λ_{EE} and λ_I are weights used to balance the EE and fairness objectives. The reward function is designed to provide positive feedback when the agent successfully meets the QoS requirements while maximizing EE and fairness. The reward is zero if the QoS requirements are not met.

The DRL agent is trained using the PPO algorithm, which is discussed in details in subsection 2.4.2.3.2. The proposed algorithm follows the **Algorithm 3**. The training

process involves iterating through episodes, where the agent interacts with the environment, collects experiences, and updates its policy based on the received rewards. For each training episode, the experience collection phase flow is ordered as follows:

1. **Initialization**: At each episode onset, initialize the active user set $\mathcal{AS}_{\text{list}}$ containing all IoT devices.

2. **Trajectory Update**: At time slot n, the UAV determines its new position $\mathbf{c}_{n+1} = [x_{n+1}, y_{n+1}, H_{n+1}]$ via:

   - Velocity adjustment $a_{v_n} \in \mathcal{A}_v$
   - Altitude adjustment $a_{H_n} \in \mathcal{A}_H$
   - Directional action $a_{md_n} \in \mathcal{A}_{md}$

   The 3D displacement distance is:

   $$d_n = \sqrt{(x_{n+1} - x_n)^2 + (y_{n+1} - y_n)^2 + (u_{H_{n+1}} - u_{H_n})^2}$$

   For lateral movement (left/right):

   $$x_{n+1} = x_n \pm u_{v_n} \cos(\chi_n) t, \quad \chi_n = \arctan\left(\frac{u_{H_{n+1}} - u_{H_n}}{x_{n+1} - x_n}\right)$$

   Analogous equations govern longitudinal (forward/backward) motion with y-coordinate updates.

3. **IoT devices Association**: At location n, the agent selects association action $a_{\kappa_n} \in \mathcal{A}_\kappa$. IoT devices partition into:

   - Group 1: Nearest node (automatically associated)
   - Group 2: $M - 1$ farther nodes (selection via κ_n)

   Association index updates as $\kappa_n = \kappa_{n-1} + a_{\kappa_n}$. For $\mathcal{AS}_{\text{list}} = \{1, 2, 3, 4\}$ sorted by ascending distance:

   $$as_{n,1} = 1 \text{ (nearest)}, \quad as_{n,\kappa_n} = 1 \text{ (selected far node)}$$

4. **Power Allocation**: Transmit power updates via $a_{P_n} \in \mathcal{A}_P$:

   $$P_{\text{TX}}^n = P_{\text{TX}}^{n-1} + a_{P_n}$$

5. **NOMA Configuration**: Power allocation factors adapt through $a_{\alpha_n} \in \mathcal{A}_\alpha$:

   $$\alpha_{n,1} = \alpha_{n-1,1} + a_{\alpha_n}, \quad \alpha_{n,2} = 1 - \alpha_{n,1}$$

where $\alpha_{n-1,1}$ denotes the prior PA factor.

6. **Store transition**: After executing all actions, the agent receives reward r_n and stores transition $(\mathbf{s}_n, \mathbf{a}_n, r_n, \mathbf{s}_{n+1})$ in experience buffer $\mathcal{D}$.

After collecting a trajectories of experience, the agent enters the PPO training phase. The training continues until convergence is achieved or a predefined number of episodes is reached. The trained agent can then be deployed in the real-world NOMA-UAV system to optimize the UAV trajectory, user association, and resource allocation in real time.

3.6 Conclusion

This chapter has systematically explored the transformative role of UAVs and RIS/STAR-RIS in advancing IoT communications, which addresses critical challenges in energy efficiency, coverage, and dynamic resource management. Beginning with an analysis of IoT's foundational requirements and UAV architectures, the discussion underscored UAVs' unique potential to deliver on-demand connectivity in hard-to-reach areas while they grapple with inherent energy consumption trade-offs. The investigation of UAV energy models and wireless use cases demonstrated their versatility in applications ranging from disaster response to smart agriculture, but it also underscored the importance of intelligent optimization to combine operational longevity with performance. The integration of RIS and STAR-RIS technologies emerged as a pivotal solution to overcome NLOS limitations and enhance spectral efficiency, which enables UAV networks to achieve robust, adaptable coverage. Through an analysis of recent literature, the chapter illustrated how DRL bridges the complexity of joint optimization in NOMA-UAV and RIS-assisted systems, as it tackles high-dimensional challenges such as trajectory planning, phase-shift configuration, and power allocation. Central to this discourse were the thesis contributions, which introduced a novel system model and DRL optimization framework to maximize EE while they ensure fairness among NOMA devices. The proposed DRL-based algorithm demonstrated how dynamic policy learning can harmonize competing objectives. The next chapter will present simulation results of the proposed frameworks.

Chapter 4

Framework Evaluation and Simulation Results

This chapter applies empirical analysis to validate the proposed DRL-driven NOMA-UAV and STAR-RIS-assisted frameworks. It starts with a system schematic that shows the integration of UAV trajectory optimization, NOMA resource allocation, RIS phase-shift control, and DRL-based decision-making. Defined simulation settings, as UAV mobility limits (e.g., 20 m/s max velocity), Rician/Rayleigh channel models, RIS configurations, and IoT device distributions. The results compare EE, JFI, SR, and SE to baselines, which demonstrate the ability of DRL to balance energy-fairness trade-offs and adapt to changing channel conditions and variations. The findings demonstrate the practicality of frameworks for real-world IoT deployments (e.g., smart cities and disaster response) and pave the way for scalable, long-lasting AI-optimized networks.

4.1 DRL NOMA-UAV Framework Evaluation

4.1.1 Proposed Framework Design

The proposed framework integrates UAV trajectory optimization and NOMA resource allocation. The framework utilizes a PPO-based algorithm to adaptively learn and optimize the system parameters in real-time and ensures efficient resource allocation and improved network performance. The MDP model with the PPO algorithm is designed as presented in Fig. 4.1.

The flowchart in Fig. 4.2 illustrates the detailed procedure of the proposed DRL-based algorithm designed to optimize NOMA-UAV-assisted wireless communication for

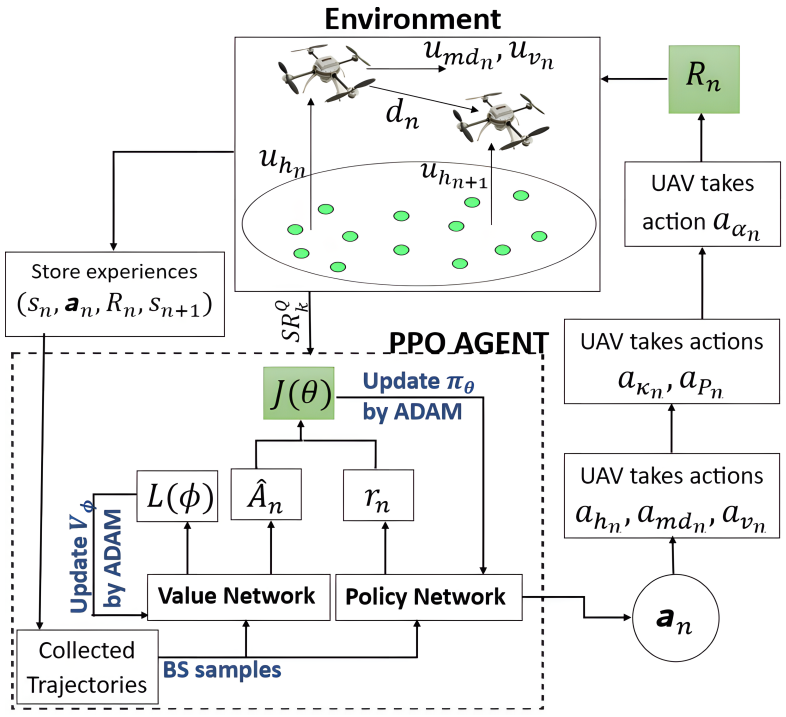


FIGURE 4.1: Proposed DRL NOMA-UAV framework.

IoT systems. Initially, the framework begins and sets the required parameters: the actor and critic network weights θ and ϕ, the episode counter eps, the maximum number of episodes eps_{max}, the number of iterations N, the number of experience trajectories for each episode T, the learning rate lr, the discount factor df, and the experience buffer D.

The outer loop iterates over the training episodes and terminates once $eps > eps_{max}$. At the beginning of each episode, the time index τ is initialized to 1 and proceeds while $\tau \leq T$. At each time step, an active device list $AS_{list} = \{0, 1, \ldots, M\}$ is initialized, and a nested loop begins and iterates over all UAV decision steps indexed by n, where $n < N$.

During each UAV operation, the UAV updates its position based on three motion components: altitude movement a_{hv}, speed a_{uv}, and movement direction a_{md}, and ensures that all motion respects predefined area boundaries. Following this, the UAV applies the Tim Sort algorithm to the active device list AS_{list}, and orders devices based on their instantaneous channel gains.

Devices are then associated according to an association policy denoted a_{as}, and associated devices are removed from the active list. Subsequently, the UAV allocates transmission power P_{tx} and distributes it among the associated users according to the allocation policy (a_{pa_1}, a_{pa_2}). Once power is transmitted, the UAV receives a scalar reward

TABLE 4.1: System and channel configuration parameters

Variable	Description	Value	Variable	Description	Value
S	Area of interest surface (m^2)	100×100	γ_k^Q	QoS requirements	$[1, 2]$ kbps
M	number of IoT devices	20	H	UAV altitude	$[50, 100]$ m
N	number of locations	10	v	UAV velocity	$[0, 20]$ m/s
T	flight period	10	ρ	PL at the reference distance	1
t	time slot duration	1	σ^2	noise power	-174 dBm
P_{tx}	total transmit power	$[8, 24]$ dBm	BW	total bandwidth	1.75 kHz

r_n and transitions to a new environmental state s_{n+1}, and forms a tuple (s_n, a_n, r_n, s_{n+1}), which is stored in the experience replay buffer D.

Upon completion of the device-level interactions for the current time step, the agent updates its policy using the PPO algorithm. The procedure continues and increments the time index τ until all time steps are completed, after which the episode counter is incremented and a new episode begins. This iterative training loop will gradually learn optimal policies for mobility control, device association, and power allocation in a dynamic communication environment. The framework continuously monitors the system performance and updates the DNN weights accordingly. The process is repeated until the total number of episodes is reached.

4.1.2 Simulation Setup

This subsection describes the simulation setup for the proposed DRL NOMA-UAV framework. First, the simulation settings and system model configuration are discussed. Next, we examine the various DRL parameters and their importance to determine the optimum hyperparameters for the task. The effectiveness of the proposed framework is then validated through benchmark comparisons.

4.1.2.1 Parameters Configuration

The system model discussed in Section 3.5.1 includes multiple system and channel configurations that Table 4.1 summarizes. Furthermore, the DRL agent necessitates careful parameter selection—from the choice of algorithm (determined by the nature of the problem) to its DNN architecture, learning rate, discount factor, and other hyperparameters. For the discrete multi-action space of this problem, the PPO algorithm is adopted because it suits such tasks. The remaining hyperparameters are listed in Table 4.2.

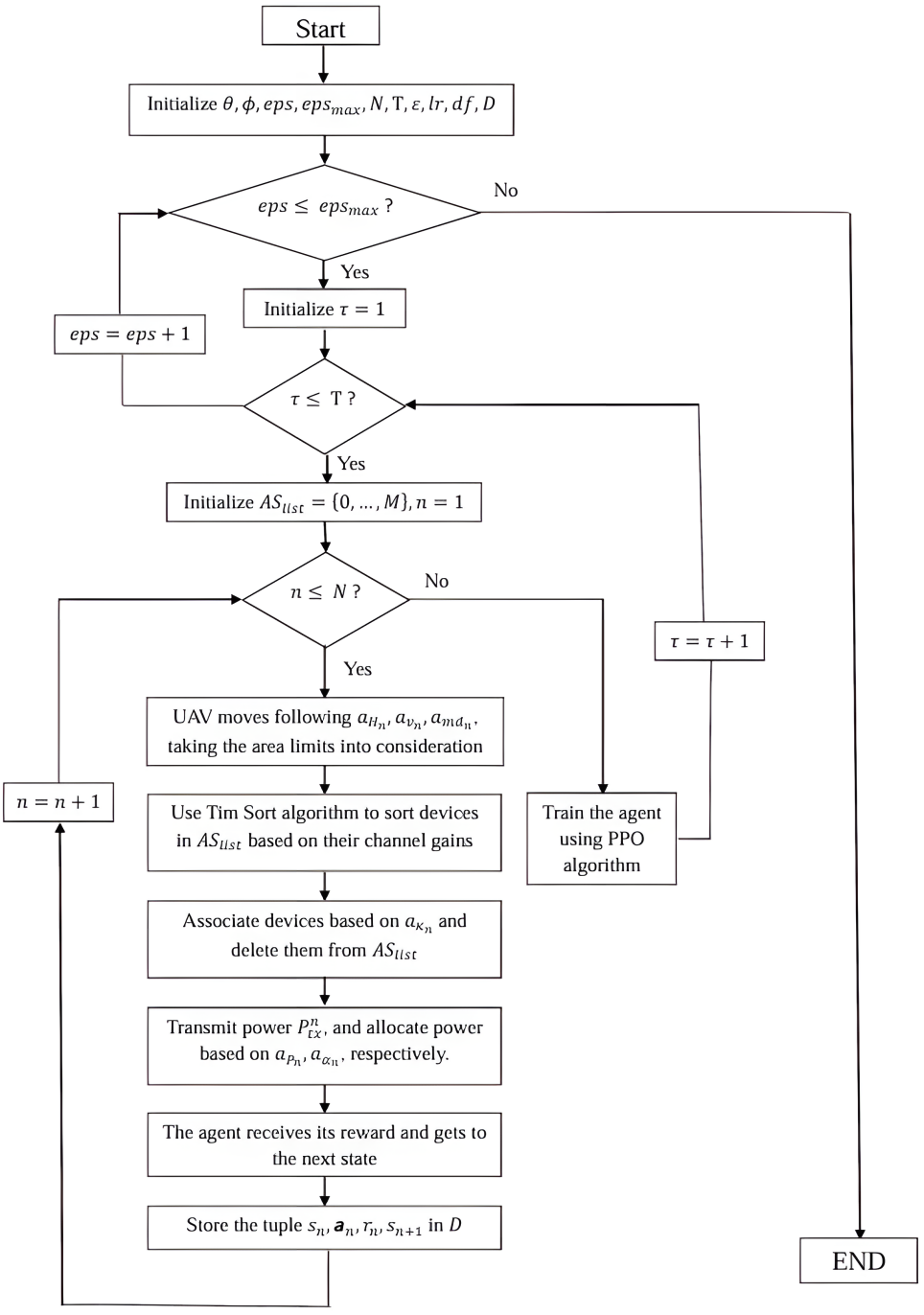


FIGURE 4.2: The execution flow of the proposed DRL NOMA-UAV framework.

TABLE 4.2: DRL configuration and hyperparameters

Variable	**Description**	**Value**	**Variable**	**Description**	**Value**
lr	learning rate	10^{-3}	λ_{EE}	EE scaling factor	10^7
df	discount factor	0.99	λ_I	JFI scaling factor	$[1, \ldots, 2]$
λ_{GAE}	GAE discount factor	0.95	$\mathcal{T}$	Trajectories of experiences	20
π	actor DNN architecture	256(3), 128(2), 64, 32	E	number of training epochs	20
V	critic DNN architecture	256(3), 128(2), 64, 32	BS	Batch size	40
ϵ	PPO clip value	0.2	activation and optimizer	*relu*	*Adam*

4.1.2.2 Baseline Benchmark

The benchmark study, [**116**], addresses the challenge of optimizing EE in NOMA-enabled UAV networks. The authors propose an Hybrid Decision Framework (HDF) that synergizes a Mamdani Fuzzy Inference System (MFIS) with a RL ensemble to dynamically manage downlink power, BW allocation, and UAV flight trajectories. The framework aims to adapt to time-varying QoS requirements while maximizing EE, SR, SE, and successful DL attempts. The key contributions of this work are:

- Combines MFIS (for expert-driven, rule-based power and BW allocation) with an RL ensemble (for adaptive transmission power, bandwidth and trajectory optimization).
- The MFIS layer uses fuzzy logic to adjust transmission power based on far-user distance and rate demand factors, while the RL ensemble (multiple DQNs) optimizes power and bandwidth allocation and UAV flight profiles.
- A maximum reward sampling mechanism selects the best-performing DQN from the ensemble at each decision cycle, ensuring adaptability in dynamic environments.
- The reward function is designed to maximize the EE while considering the BW consumption.

This study is selected for benchmarking for several reasons:

- It integrates model-free RL with expert knowledge (via MFIS), addressing both resource allocation and UAV mobility challenges in NOMA networks.

- The ensemble RL structure enables robust performance in uncertain, real-time scenarios, making it suitable for B5G and 6G applications.
- Extensive simulations validate its superiority over conventional RL and static allocation methods, providing a reliable performance standard for subsequent research.

With HDF, we also consider its PA scheme as a baseline to compare it against the PA used by the proposed framework, to evaluate its robustness when it comes to a fairness-aware EE optimization. Unlike our DRL approach, which dynamically adapts PA based on real-time channel conditions and fairness-aware optimization, HDF employs a fixed QoS-based PA strategy. Specifically, HDF allocates power coefficients to NOMA users as a function of the QoS requirement of the farthest IoT device, where α_K is the power coefficient for user K and is calcualted based on the required SINR which is calculated based on the data rate requirement of this user, i.e., $SINR_K = 2^{R_K^Q} - 1$, then α_K will be calculated based on this value, and the power coefficient of the near-user $\alpha_1 = 1 - \alpha_2$ with $K = 2$.

4.1.2.2.1 HDF Agents

The HDF comprises two core components:

- **Profile Management Agent**: A DQN-based DRL agent responsible for optimizing the UAV's flight trajectory. It dynamically selects flight profiles (trajectories) characterized by parameters such as radius, height, speed, and locations.
- **DQN Ensemble**: A parallelized decision layer consisting of multiple independent DQN sub-agents. Each sub-agent operates autonomously, learning distinct policies for resource allocation (power and bandwidth) at each transmission slot.

Here, the profile agent recommends a flight profile, and at each location, the DQN ensemble collaboratively optimizes power and bandwidth allocation. A reward sampler selects the best-performing sub-agent's decision at each step, enabling adaptive resource management under uncertainty.

4.1.2.2.2 States and Actions For HDF Agents

Profile management and the DQN ensemble have separate state and action spaces. Theses spaces for the profile management agent is defined as follows:

1. **Profile Management Agent** We have modified the profile management action and state space in such way that reduce its complexity and make it more efficient. The state and action spaces are defined as follows:

- **State Space**: Defined by two parameters:
  - $s_{PF} = (H_{PF}, v_{PF})$: Current UAV height and speed.
- **Action Space**: Discrete actions to adjust flight parameters:
  - $A_{PF} = [U_H, D_H, U_v, D_v, R]$:
    - U: Increase parameter (e.g., height H), D: Decrease, R: Retain.
    - Adjustments follow fixed increments (e.g., $\Delta H = 5$ m, $\Delta v = 5$ m/s).

2. **DQN Ensemble** The state and action spaces for the DQN ensemble are defined as follows:

- **State Space**:
  - $v^i_{s_i} = (P^n_{tx}, BW)$: The transmission power and bandwidth allocated to a cluster.
- **Action Space**:
  - $A(i) = [s_n^{v+0}, s_n^{v+1}, s_n^{v-1}]$: Three actions per sub-agent:
    (a) Maintain current power/bandwidth state.
    (b) Increase power/bandwidth (progress state).
    (c) Decrease power/bandwidth (retard state).

4.1.2.2.3 HDF Execution Flow

The HDF operates in two nested loops:

1. **Outer Loop (Profile Management):**

- **Step 1**: The profile agent selects a flight profile PF_f using its DQN policy, determining waypoints W_{PF_f}.
- **Step 2**: For each profile, the UAV traverses N location.

2. **Inner Loop (Resource Management at Each Location):**

- **Step 3: At slot t:**
  - Randomly sample K IoT devices and cluster them $(\mathcal{D}_{k,m})_t$.
  - Sort devices by channel gain (merge sort) to determine NOMA decoding order Ω_{order}.
- **Step 4: MFIS Layer:**
  - Inputs: Normalized far-user distance (ϕ_D) and maximum rate demand (ϕ_R).
  - Output: Power budget P^n_{tx} using MFIS.
- **Step 5: DQN Ensemble Decision:**

  - Each sub-agent selects a power/bandwidth state $v_{s_n}^n$ based on P_{tx}^n.
  - Actions are forwarded to a power allocation stage, where SIC-compliant power coefficients $\alpha_{n,k}$ are calculated.
- **Step 6: Reward Calculation:**
  - Sub-agents receive rewards r_f^n, based on EE EE_{n_i}, bandwidth usage, and outage avoidance.
  - The maximum reward $r_{max}(n)$ from the ensemble is selected.
- **Step 7: Learning Updates:**
  - Experience replay buffers update DQN weights (Eq. 9-10).
  - The profile agent accumulates rewards r_f across all slots in PF_f.

3. **Termination:**

- After completing all profiles, the system converges to policies that maximize long-term EE and QoS.

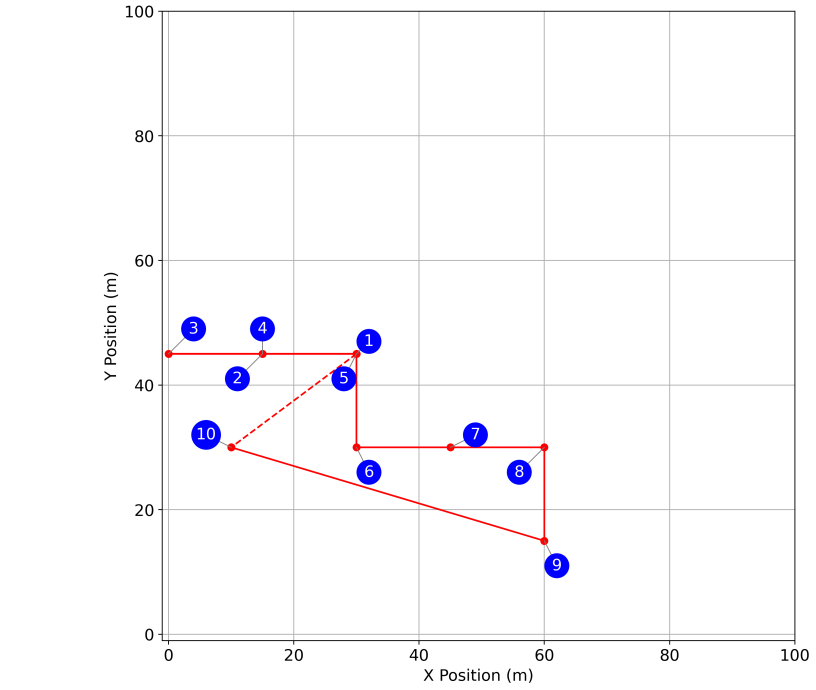
In this thesis, the HDF methodology is adopted as a baseline to evaluate the efficacy of proposed enhancements to NOMA-UAV networks. Specifically, our work extends HDF by making a fully intelligent system that has a fully knowledge of the system without any need for MFIS and proposing a fairness-aware energy efficient framework that maximize EE and fairness by changing a parameter that controls their weights, addressing limitations such as the lack of fairness in HDF, which is critical KPI in IoT communications, and this framework has a dynamic IoT devices association mechanism where HDF associcate devices randomly. By building on HDF framework, this research aims to further bridge the gap by creating a system that is capble of controlling EE-fairness trade-off in NOMA-UAV networks for IoT communications.

4.1.3 Simulation Results Presentation and Discussion

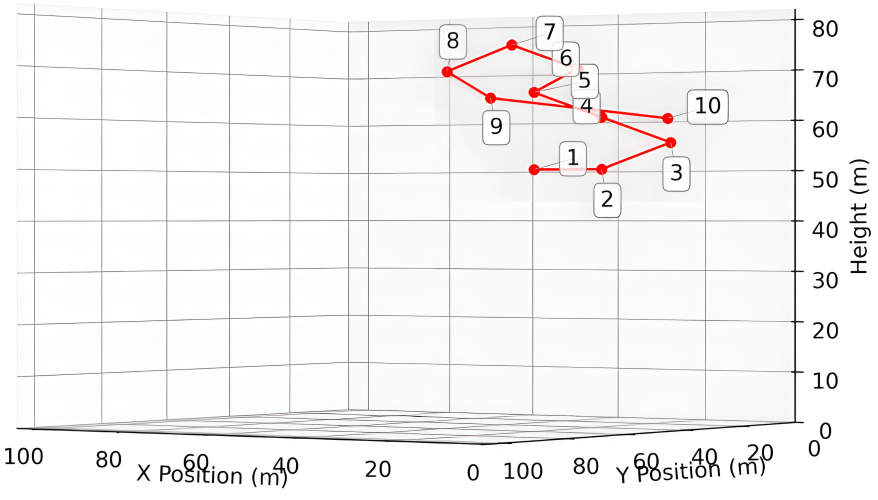
This subsection presents the empirical validation of the proposed DRL-driven NOMA-UAV framework, offering a deep analysis of its performance across key metrics such as EE, fairness, SR, EE, reward, and scalability. Building on the simulation setup and baseline benchmarks defined earlier (Section 4.1.2), the results are structured to first quantify the framework's ability to balance EE and fairness through dynamic adaptation of the weight parameter λ_I, followed by a comparative evaluation against benchmarks.

4.1.3.1 UAV Trajectory

UAV trajectory optimization is an important aspect of UAV-assisted wireless communication systems. For this reason, the intelligent agent optimizes the 3D trajectory of the UAV within the IoT area, which enhances coverage, conserves energy, and reduces power consumption during flight; as illustrated in Fig 4.3, the UAV moves dynamically to adapt to network demands and balances coverage with EE.



(a)



(b)

FIGURE 4.3: Optimized UAV trajectory for the proposed NOMA-UAV framework (a) 2D. (b) 3D.

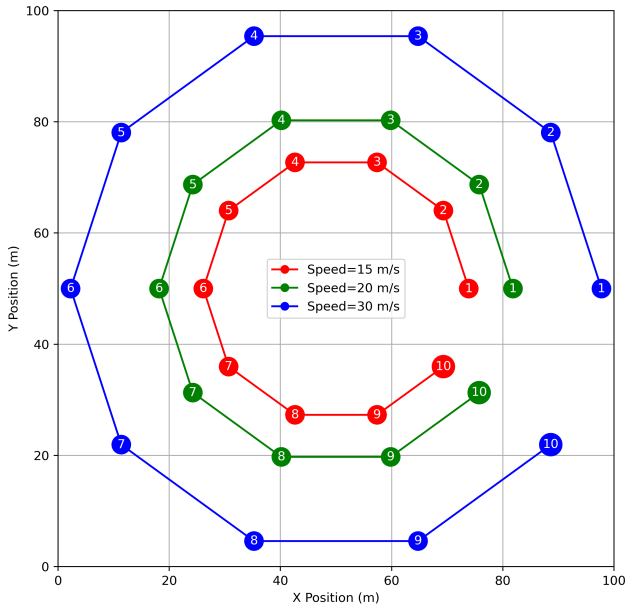


FIGURE 4.4: The UAV trajectory adopted for HDF based on the UAV speed.

In contrast, HDF uses a circular trajectory (Fig. 4.4), in which the UAV orbits the center of the area at a radius defined by its speed. While this strategy simplifies path planning, it limits the UAV to specified waypoints during the downlink cycle, which results in inferior coverage and EE due to the inability to adapt dynamically to real-time network conditions.

The proposed system enables the UAV to move freely in 3D space and change its course in any direction to maximize performance. This flexibility allows the drone to adjust to changes in IoT device locations and signal quality over time, which improves energy economy and coverage. In contrast, the fixed circular trajectory and static path-planning of HDF restrict the movement of the UAV and disregard these real-time changes.

4.1.3.2 Hyperparameters comparison

Fig. 4.5 depicts several KPI measures for the proposed system with varying hyperparameter setups.

The average reward curves in Fig. 4.5(a) are a composite indicator of the capacity of the agent to balance EE and fairness index under the joint optimization objective given in Equation 3.30. The dominant performance of the $\epsilon = 0.2$, $lr = 10^{-3}$, $\lambda_I = 2$ configuration (pink curve), with a peak at ~ 1.8, points out significant hyperparameter

interdependencies. Comparative research shows the effect of the λ_I parameter: increasing λ_I from 1 (red curve) to 2 results in a 15% reward augmentation, which indicates that the reward function is sensitive to fairness weighting. In blue and green configurations, suboptimal exploration ($\epsilon = 0.1$) leads to premature convergence, whereas instability of the brown and green curve ($lr = 10^{-2}$) underscores the necessity to find the right learning rate balance. The important criteria for effective exploration of the policy space and stability of gradient descent are $\epsilon = 0.2$ and $lr = 10^{-3}$, respectively.

Fig. 4.5(b) shows an inverse relationship between λ_I values and maximization of EE. The $\lambda_I = 1$ configuration (red) achieves peak EE (1.4×10^7 bits/Joule), whereas $\lambda_I = 2$ obtains a marginally lower 0.8×10^7 bits/Joule. The higher reward performance of the pink curve is accompanied by a 70% EE drop, which validates the expected multi-objective trade-off. Increased weighting of λ_I allows for improvements of JFI that compensate for EE losses in the composite reward function. The catastrophic EE performance of the $lr = 10^{-2}$ configuration highlights the nonlinear impact of the learning rate on optimization of the policy gradient, where large parameter updates might destabilize the energy-efficient strategy.

Fig. 4.5(c) illustrates the SR outcomes for different learning rates. When the learning rate is set to 10^{-2}, the agent reaches the highest possible SR of 10^5 bps, but it results in a very low overall reward and energy efficiency, which shows that while the system is fast, it does not satisfy the QoS needs and optimization goals. The low reward implies that the policy ignores fairness and energy considerations in favor of raw spectral performance. Such behavior is most likely caused by unstable learning dynamics created by the aggressive learning rate, which results in resource allocations that maximize throughput yet are inefficient for the system. This result indicates that SR is insufficient as an optimization target, and a multi-objective formulation is required to avoid degenerate solutions.

The progression of the fairness index in Fig. 4.5(d) quantifies the role of the λ_I parameter as a controller of fairness. The pink design ($\lambda_I = 2$) provides near-perfect fairness ($JFI > 0.98$), greatly exceeding the baseline of $\lambda_I = 1$ ($JFI \approx 0.55$) due to intentional design of the reward function. This 85% improvement of fairness demonstrates the effectiveness of the mathematical formulation of the reward function, which directly affects the balance between the two measures.

Combining these facts, the configuration of $\epsilon = 0.2$, $lr = 10^{-3}$, $\lambda_I = 2$ is the Pareto-optimal option, which results in the highest reported reward while preserving 0.8×10^7 bits/Joule peak EE and 0.95% JFI. This combination of hyperparameters balances three critical dimensions: 1) Adequate exploration (ϵ) to avoid local optima in the high-dimensional action space. 2) Stable updates of the policy (lr) for reliable

gradient descent. 3) Fairness-weighted shaping of the reward (λ_I) to prevent unfair allocation of resources. The results confirm the capabilities of the PPO algorithm for multi-objective optimization in scenarios of constrained resource allocation. λ_I is an effective parameter for control of the EE-JFI trade-off.

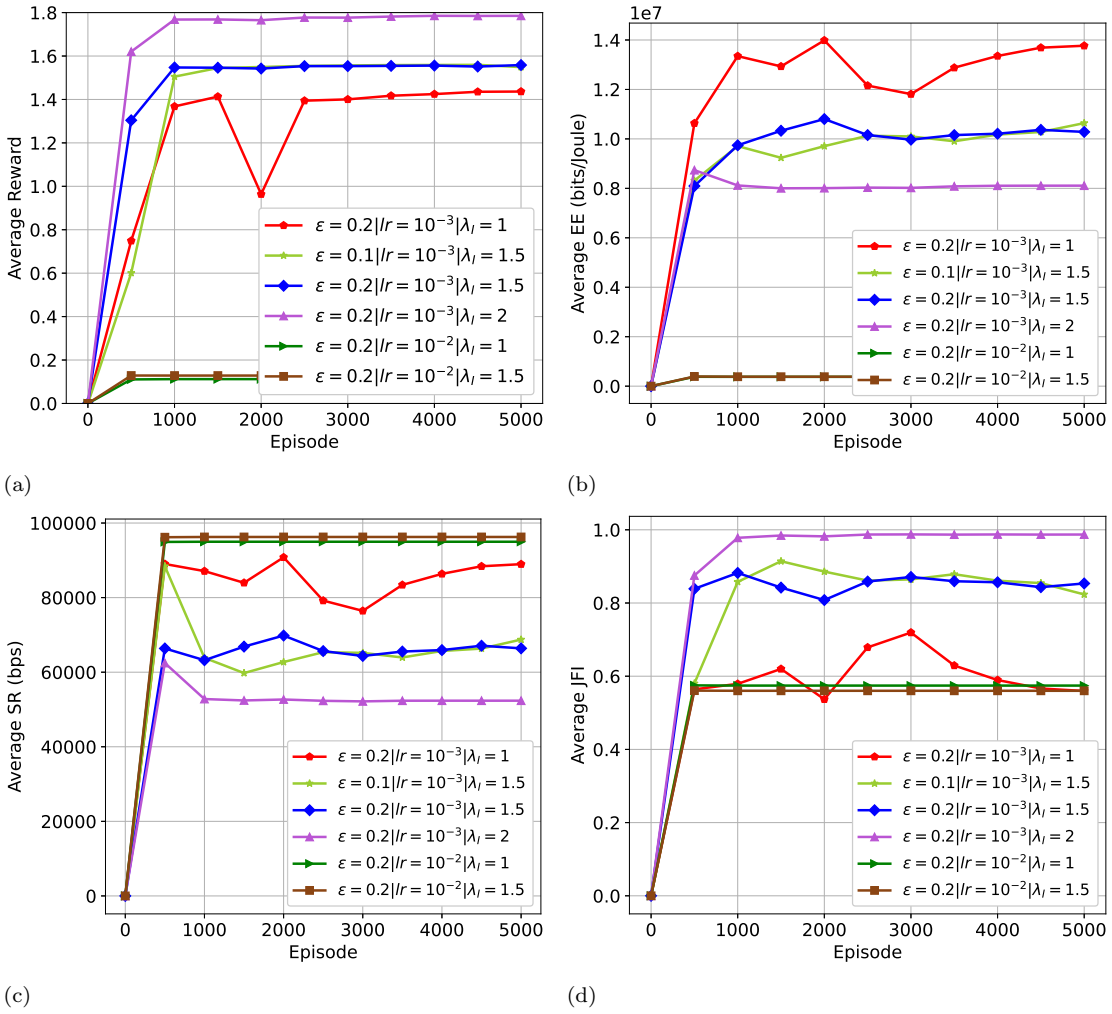


FIGURE 4.5: Proposed framework: Hyperparameters comparison (a) Reward. (b) EE. (c) SR. (d) JFI.

4.1.3.3 Benchmark Comparison

The proposed DRL framework demonstrates quantifiable trade-offs between EE and fairness through strategic shaping of reward governed by the parameterλ_I and achieves Pareto-optimal solutions across four KPIs. As illustrated in Fig. 4.6(a), the QoS-based PA baseline (black curve) attains peak EE at 1.4×10^7 bits/Joule and outperforms both fairness-weighted DRL variants ($\lambda_I = 1.5$: 1.1×10^7; $\lambda_I = 2$: 0.74×10^7) and the HDF baseline (0.8×10^7). The 60.9% gain of EE of $\lambda_I = 1.5$ over $\lambda_I = 2$ highlights the cost

of efficiency associated with prioritization of fairness, as fair resources allocation limits capacity of peak energy optimization.

Characteristics of SR in Fig 4.6(b) reveal similar trends, with the QoS-based PA baseline dominating at 84,000 bps compared to 65,000 bps of $\lambda_I = 2$ (37.4% shortfall). The $\lambda_I = 1.5$ configuration (78,000 bps) improves retention of SR under constraints of fairness, which highlights the role of λ_I in balancing trade-offs between SR and fairness. HDF outperforms fairness-aware agents ($\lambda_I = 1.5$ and 2) by margins of 24.2% and 57.4%, respectively. SE in Fig 4.6(c) highlights the trade-offs: the QoS-based PA baseline achieves 50 bps/Hz, whereas $\lambda_I = 2$ yields 30 bps/Hz, which results in a 74.2% gap of performance and indicates underutilization of frequency band under constraints of fairness.

The progression of fairness in Fig 4.6(d) inverts the hierarchy, as $\lambda_I = 2$ achieves near-optimal fairness (JFI ≈ 1.0) compared to the QoS-based PA baseline (JFI $= 0.55$), which represents an 85.5% improvement. The $\lambda_I = 1.5$ setup (JFI $= 0.85$) achieves a balance as it drops 15% absolute fairness to recover 48.6% of the EE advantage of the QoS-based PA. The continuum of energy and fairness governed by λ_I aligns with the theoretical formulation of reward in Equation 3.30.

By episode 1000, all configurations show consistent convergence; variations of DRL weighted by fairness ($\lambda_I = 1.5, 2$) show slightly slower initial convergence because of increased needs of policy search. The consistent underperformance of HDF baseline across EE and fairness (JFI $= 0.6$, EE $= 0.8 \times 10^7$ bits/Joule) validates the dual advantages of the proposed architecture: (1) superior optimization relative to HDF and (2) tunable trade-offs of fairness and efficiency compared to fixed strategies of PA, where operators of network can dynamically adjust λ_I to prioritize fairness during congestion ($\lambda_I \to 2$), maximize EE under constraints of energy ($\lambda_I \to 1$), or balance these objectives ($\lambda_I \to 1.5$).

These findings confirm the ability of the proposed DRL framework to adaptively optimize allocation of resources in NOMA-UAV networks, maximizing EE and fairness. The proposed framework with QoS-based PA baseline outperforms HDF in EE and SR due to its comprehensive optimization of device association, UAV trajectory, and resource allocation. In contrast, HDF uses a two-tier architecture: a DQN agent for UAV path planning and an ensemble DQN with an MFIS for resource allocation. This separation produces suboptimality because the interdependent rewards and actions of the two agents may result in contradictory decisions (e.g., when one agent chooses a suboptimal action that negatively influences the other). Furthermore, the random device association of HDF exacerbates inefficiencies, whereas the proposed framework optimizes associations dynamically to ensure alignment with global goals of the optimization.

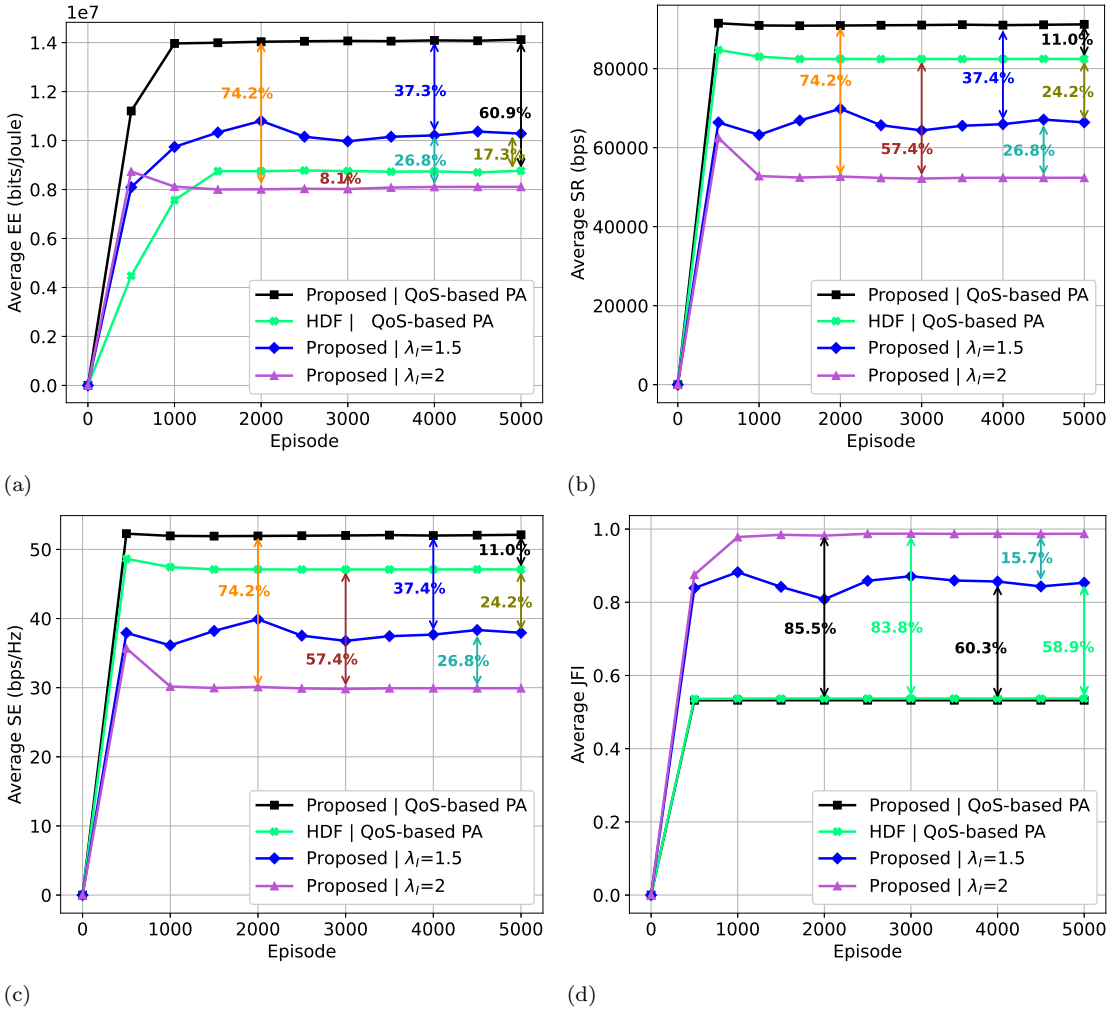


FIGURE 4.6: Proposed vs. Benchmarks (a) EE. (b) SR. (c) SE. (d) JFI.

4.1.4 DRL Framework for STAR-RIS Assisted SCMA-UAV networks

This section presents the system model of the proposed DRL framework and its simulation results, designed to optimize multiple parameters in a STAR-RIS-assisted SCMA-UAV network to maximize EE. To the best of the authors' knowledge, this work represents the first investigation of SCMA in STAR-RIS-assisted UAV networks and introduces the first DRL framework that jointly optimizes multiple parameters; including UAV trajectory, RIS phase shifts, and power allocation; in such systems.

4.1.4.1 System Model

The system model adopted from considers a DL scenario where a single-antenna UAV acts as an ABS serving J single-antenna ground users. The communication is aided by a Uniform Planar Array (UPA) STAR-RIS composed of N passive elements operating

under the MS protocol, where half the elements reflect and half transmit signals at any given time.

- **Multiple Access Scheme:** SCMA is employed, where J users share K Resource Elements (REs). Each user j spreads its codeword (CW) over d_v REs, and each RE k serves d_f users. The mapping is defined by a Mapping Matrix (MM) $\mathbf{F} \in \{0,1\}^{K\times J}$. MPA is assumed for multi-user detection.
- **Components and Geometry:** Users are located at coordinates $\mathbf{c}_j = (x_j, y_j)$. The STAR-RIS is fixed at $\mathbf{c}_r = (x_r, y_r, z_r)$. The UAV flies at a fixed altitude z_u, with its 2D position at time t being $(x_u(t), y_u(t))$.
- **STAR-RIS Model:** Under the MS protocol, the n-th element's reflection ($\beta_n^{\mathcal{R}}$) and transmission ($\beta_n^{\mathcal{T}}$) amplitudes satisfy $\beta_n^{\mathcal{R}}, \beta_n^{\mathcal{T}} \in \{0,1\}$ and $\beta_n^{\mathcal{R}} + \beta_n^{\mathcal{T}} = 1$. The reflection and transmission phase shifts are denoted by $\theta_{\mathcal{R},n}$ and $\theta_{\mathcal{T},n}$ respectively, forming diagonal phase shift matrices $\mathbf{\Theta}_{\mathcal{R}}$ and $\mathbf{\Theta}_{\mathcal{T}}$. Users are implicitly assigned to reflection ($J_{\mathcal{R}}$) or transmission ($J_{\mathcal{T}}$) sets based on their position relative to the STAR-RIS.
- **Channel Models:** The channel is composed of three main channel links and the resulting composite channel. Let $d_{x,y} = ||\mathbf{c}_x - \mathbf{c}_y||$ be the distance between points x and y. The path loss is generally modeled as $PL_{x,y} = \rho_0 d_{x,y}^{-\alpha_{x,y}}$, where ρ_0 is the path loss at a reference distance $d_0 = 1$m and $\alpha_{x,y}$ is the path loss exponent for the link.
  - *UAV-User (Direct Link):* The channel coefficient between the UAV and user j for RE k is given by:
    $$h_{u,k,j} = \sqrt{PL_{u,j}}\tilde{h}_{u,k,j} \tag{4.1}$$
    where $PL_{u,j}$ is the path loss for the UAV-user link, and $\tilde{h}_{u,k,j} \sim \mathcal{CN}(0,1)$ represents the small-scale fading (random scattering component). This link is assumed potentially blockable in the system context.
  - *UAV-STAR-RIS:* Due to the typically high altitude of the UAV and potential LoS, this channel is modeled as free-space propagation, and is given by:
    $$\mathbf{h}_{u,r} = \sqrt{PL_{u,r}}\mathbf{v}_t^H \mathbf{v}_c \tag{4.2}$$
    where $PL_{u,r}$ is the UAV-RIS path loss. $\mathbf{v}_t$ and $\mathbf{v}_c$ are the array response vectors of the UPA STAR-RIS related to the horizontal and vertical dimensions, respectively:
    $$\mathbf{v}_t = \left[1, e^{-j\frac{2\pi}{\lambda}d_t\phi_{u,r}}, \ldots, e^{-j\frac{2\pi}{\lambda}d_t(N_x-1)\phi_{u,r}}\right]^T \tag{4.3}$$

$$\mathbf{v}_c = \left[1, e^{-j\frac{2\pi}{\lambda}d_c\psi_{u,r}}, \ldots, e^{-j\frac{2\pi}{\lambda}d_c(N_y-1)\psi_{u,r}}\right]^T \tag{4.4}$$

Here, λ is the wavelength, d_t and d_c are the horizontal and vertical distances between STAR-RIS elements, $N = N_x \times N_y$ is the total number of elements. $\phi_{u,r} = \sin(\theta^H_{u,r})\cos(\theta^V_{u,r})$ and $\psi_{u,r} = \sin(\theta^V_{u,r})$ depend on the horizontal ($\theta^H_{u,r}$) and vertical ($\theta^V_{u,r}$) angles of arrival (AoA) of the signal at the STAR-RIS.

– *STAR-RIS-User:* This channel follows a Rician fading model, accounting for both LoS and NLoS components, and is given by:

$$\mathbf{h}_{r,k,j} = \sqrt{PL_{r,j}}\left(\sqrt{\frac{G}{G+1}}\mathbf{h}^{\text{LoS}}_{r,k,j} + \sqrt{\frac{1}{G+1}}\mathbf{h}^{\text{NLoS}}_{r,k,j}\right) \tag{4.5}$$

where $PL_{r,j}$ is the STAR-user path loss, G is the Rician factor. The LoS component $\mathbf{h}^{\text{LoS}}_{r,k,j}$ depends on the STAR-RIS array response vectors related to the Angle of Departure (AoD) towards user j:

$$\mathbf{h}^{\text{LoS}}_{r,k,j} = \mathbf{a}_t(\phi_{r,j}) \otimes \mathbf{a}_c(\psi_{r,j}) \tag{4.6}$$

where $\mathbf{a}_t(\cdot)$ and $\mathbf{a}_c(\cdot)$ are the array steering vectors for the horizontal and vertical dimensions, which are functions of the horizontal ($\phi_{r,j}$) and vertical ($\psi_{r,j}$) AoD parameters derived from the user's location relative to the STAR-RIS. The NLoS component $\mathbf{h}^{\text{NLoS}}_{r,k,j}$ consists of independent and identically distributed elements following $\mathcal{CN}(0,1)$.

- **Composite Channel:** The effective channel $h_{k,j}$ for user j on RE k combines the direct UAV-user path and the cascaded UAV-STAR-User path, modulated by the STAR-RIS phase shifts. Based on whether user j is served by reflection ($j \in J_{\mathcal{R}}$) or transmission ($j \in I_{\mathcal{T}}$), the composite channel is defined as:

$$h_{k,j} = \begin{cases} h_{u,k,j} + \mathbf{h}_{u,r}\mathbf{\Theta}_R\mathbf{h}_{r,k,j}, & \text{if } j \in J_{\mathcal{R}} \\ h_{u,k,j} + \mathbf{h}_{u,r}\mathbf{\Theta}_T\mathbf{h}_{r,k,j}, & \text{if } j \in J_{\mathcal{T}} \end{cases} \tag{4.7}$$

where $\mathbf{\Theta}_R = \text{diag}(\beta^R_1 e^{j\theta_{r,1}}, \ldots, \beta^R_N e^{j\theta_{r,N}})$ and $\mathbf{\Theta}_T = \text{diag}(\beta^T_1 e^{j\theta_{t,1}}, \ldots, \beta^T_N e^{j\theta_{t,N}})$ are the diagonal matrices containing the reflection and transmission coefficients (amplitude and phase shift) of the STAR-RIS elements, respectively.

- **Data Rate:** The achievable rate for user j on RE k is given by:

$$R_{k,j} = \log_2\left(1 + \text{SINR}_{k,j}\right) \tag{4.8}$$

where $\text{SINR}_{k,j} = \frac{f_{k,j} P_{k,j} |h_{k,j}|^2}{\sum_{i \in \mathcal{S}_k, i \neq j} f_{k,i} P_{k,i} |h_{k,i}|^2 + \sigma^2}$. Here $P_{k,j}$ is the power allocated to user j on RE k, $f_{k,j}$ is the (k,j)-th entry of $\mathbf{F}$, $\mathcal{S}_k$ is the set of users on RE k, and σ^2 is the noise power. The total SR at time t is:

$$SR_t = \sum_{k=1}^{K} \sum_{j=1}^{J} R_{k,j} \tag{4.9}$$

- **UAV Power Consumption:** A fixed-wing UAV model is used, where power P_t^U depends on UAV velocity v_t and acceleration $\mathbf{a}_{c,t}$.
- **EE:** The instantaneous EE is:

$$EE_t = \frac{SR_t}{\sum_{k=1}^{K} \sum_{j=1}^{J} P_{k,j} + P_t^U} \tag{4.10}$$

  The objective in is to maximize the total EE over a period T, $\sum_{t=1}^{T} EE_t$.

- **Optimization Variables:** The DRL agent optimizes the SCMA MM $\mathbf{F}$, PA matrix $\mathbf{P} \in \mathbb{R}^{K \times J}$, UAV trajectory (via acceleration $\mathbf{a}_{c,t}$ and movement direction md_t), total DL transmit power $P_{TX,t}$, and STAR-RIS phase shifts $\mathbf{\Theta}_R, \mathbf{\Theta}_T$.
- **Constraints:** The optimization is subject to constraints on minimum user data rate (R^Q), total transmit power, SCMA structure (d_v, d_f), STAR-RIS phase/amplitude, UAV position, velocity, and acceleration limits.

4.1.4.2 Simulation Framework and Setup

To evaluate the proposed system, we design a simulation framework that employs a DRL approach based on the PPO algorithm.

- **DRL Agent:** The PPO agent learns a policy π to map states to actions.
  - *State Space $\mathcal{S}$:* Includes UAV position (x_u, y_u), velocity $\mathbf{v}_t$, and channel state information (real and imaginary parts of $\mathbf{h}_{u,r}$ and $\mathbf{h}_{r,j}$).
  - *Action Space $\mathcal{A}$:* Continuous space including UAV acceleration magnitude a_c^u, movement direction md, transmit power P_{TX}, an index for MM selection a^F, parameters defining the PA matrix P, and parameters defining the phase shifts $(\mathbf{\Theta}_R, \mathbf{\Theta}_T)$. Actions are constrained and mapped to physical parameters (e.g., MM index is rounded, PA is normalized).

– *Reward Function r_t:* Designed to maximize EE while satisfying the minimum data rate constraint R^Q. The reward function is defined as:

$$r(s_t, \mathbf{a}_t, s_{t+1}) = \begin{cases} EE_t, & \text{if QoS is satisfied} \\ \frac{EE_t}{(e_p)}, & \text{otherwise} \end{cases} \tag{4.11}$$

where e_p is a penalty term for violating the QoS constraint.

- **Simulation Parameters:** Simulation key parameters are summarized in Table 4.3.
- **Performance Metrics:** The primary metrics evaluated are the average EE (bits/Hz/Joule) and the average reward obtained by the DRL agent over the learning process.
- **Baseline Scenarios for Comparison:**
  1. *Full Optimization (Proposed):* The PPO agent jointly optimizes all variables (MM, PA, P_{TX}, Trajectory, Phase Shifts).
  2. *No Phase Optimization:* STAR-RIS phase shifts are fixed to zero; other variables are optimized.
  3. *Uniform PA:* Power is allocated uniformly across active user/RE pairs; PA matrix $\mathbf{P}$ is not optimized by the agent.
  4. *Predefined MM:* The SCMA MM $\mathbf{F}$ is fixed as presented in Equation 1.5; not optimized by the agent.
  5. *PD-NOMA:* PD-NOMA is used instead of SCMA.
  6. *Conventional RIS:* A reflecting-only RIS is used instead of STAR-RIS (180° coverage).

4.1.4.3 DRL-Based Optimization Algorithm Flow

To address the complex joint optimization problem of maximizing energy efficiency in the UAV-assisted communication system, a DRL framework based on the PPO algorithm is employed. This approach utilizes an agent, interacting with the simulated system environment, to learn an optimal control policy over time.

The algorithm proceeds iteratively through episodes, where each episode simulates the system operation over a defined period T (composed of multiple time steps δt). The overall flow involves initialization followed by repeated cycles of experience collection and policy/value function updates.

Table 4.3: Simulation Parameters

Parameter	Value	Unit
Simulation Area	1000×1000	meter^2
STAR-RIS Position (x_r, y_r, z_r)	(500, 500, 15)	meter
UAV Initial Position (x_u, y_u, z_u)	(500, 500, 120)	meter
UAV Altitude z_u	120	meter
Number of Users J	6	
Number of REs K	4	
SCMA d_v (REs per user)	2	
SCMA d_f (Users per RE)	3	
STAR-RIS Elements N	100 (also 16, 36, 64)	
UAV Velocity Range $[v_{min}, v_{max}]$	[30, 100]	m/s
UAV Acceleration Range $[a_{min}, a_{max}]$	[-5, 5]	m/s^2
Total Tx Power Range P_{TX}	[0, 30]	dBm
Path Loss Exponent $\alpha_{u,r}$	2.0	
Path Loss Exponent $\alpha_{u,j}$	3.5	
Path Loss Exponent $\alpha_{r,j}$	2.8	
Rician Factor G	10	dB
Reference Path Loss ρ_0	-30	dB
Carrier Frequency f_c	2.4	GHz
Noise Power σ^2	-174	dBm
Min. User Rate R^Q	0.1	bps/Hz
Simulation Period T	50	second
Time Slot Duration δt	1	second
UAV Power Constants c_1, c_2	As per cited in [**91**]	
PPO Discount Factor γ	0.99	
PPO Clip Range ϵ	0.2	
Learning Rate lr	10^{-3}(also $10^{-2}, 10^{-4}$)	
DRL Epochs E	20	
DRL Batch Size BS	62	
DRL Penalty ϵ_p	10^3	
DNN Architecture (Actor/Critic)	(512, 256, 256, 256) layers	
DNN Activation Function	SiLU	
DNN Optimizer	Adamax	

- Initialization:
  - Initialize the weights $w^{(a)}$ of the Actor DNN and $w^{(c)}$ of the Critic DNN.
  - Set key hyperparameters: learning rate lr for DNN updates, discount factor $\gamma \in [0, 1]$ for future rewards, the PPO clipping parameter ϵ, the number of training epochs per iteration E, and the mini-batch size BS.
  - Initialize an experience replay buffer D as an empty set to store interaction transitions (s_t, a_t, r_t, s_{t+1}).

- Pre-generate a tensor $\mathcal{F}$ containing all valid SCMA Mapping Matrix (MM) configurations that satisfy system constraints (e.g., REs per user d_v, users per RE d_f). Initialize the starting MM $\mathbf{F}$.

- Iterative Training Loop: The algorithm enters a main loop that repeats for a fixed number of iterations or until convergence. Each iteration consists of two phases:

  1. Experience Collection Phase: The agent interacts with the system environment using its current policy $\pi_{w^{(a)}}$ for one full episode duration T.
     - For each time step t from 1 to T:
       (a) **State Observation:** Observe the current system state s_t, including UAV position (x_t, y_t), velocity v_t, and relevant channel state information (e.g., features derived from $\mathbf{h}_{u,r}$ and $\mathbf{h}_{r,j}$).
       (b) **Action Selection:** Sample an action vector $\mathbf{a}_t$ from the policy distribution output by the Actor network given the state: $a_t \sim \pi_{w^{(a)}}(\cdot|s_t)$. This vector contains continuous values for various control aspects: $\mathbf{a}_t = (a_c^u, md, \mathbf{a}^{\Theta_R}, \mathbf{a}^{\Theta_T}, \mathbf{a}^F, a^{P_{TX}}, \mathbf{a}^P)$.
       (c) **Action Execution and Parameter Mapping:** Translate the continuous action vector a_t into physical system control parameters:
       * *UAV Movement:* Update UAV velocity and position using acceleration a_c^u and direction md. Apply velocity and position constraints:

$$
\begin{aligned}
v_{t+1} &= \text{clip}(v_t + a_c^u \delta t, v_{min}, v_{max}) \\
x_{t+1} &= \text{clip}(x_t + v_{t+1} \cos(m_d)\delta t, x_{min}, x_{max}) \\
y_{t+1} &= \text{clip}(y_t + v_{t+1} \sin(m_d)\delta t, y_{min}, y_{max})
\end{aligned}
$$

       * *STAR-RIS Phase Shifts:* Map action components $\mathbf{a}^{\Theta_R}$ and $\mathbf{a}^{\Theta_T}$ to phase shift values $[\theta_{\mathcal{R},1}, \ldots, \theta_{\mathcal{R},N}]$ and $[\theta_{\mathcal{T},1}, \ldots, \theta_{\mathcal{T},N}]$, populating the diagonal matrices $\mathbf{\Theta}_R$ and $\mathbf{\Theta}_T$. These phases are typically in $[0, 2\pi)$.
       * *SCMA MM:* Determine the MM index $I_{MM} = \text{round}(\mathbf{a}^F)$ and select the corresponding pre-generated matrix $\mathbf{F} = \mathcal{F}[I_{MM}]$. Ensure I_{MM} is within valid bounds for $\mathcal{F}$.
       * *Transmit Power:* Set the total downlink transmit power P_{TX} based on $a^{P_{TX}}$, constrained within $[P_{min}, P_{max}]$.
       * *Power Allocation:* Determine the PA matrix $\mathbf{P} \in \mathbb{R}^{K \times J}$ based on $\mathbf{a}^P$. This typically involves normalizing segments of a^P such that $\sum_{k=1}^{K} \sum_{j=1}^{J} P_{k,j} \leq P_{TX}$, and setting $P_{k,j} = 0$ if $f_{k,j} = 0$ in the chosen $\mathbf{F}$.

(d) **Environment Step and Reward Calculation:** Apply the chosen parameters. The environment transitions to the next state s_{t+1} based on system dynamics. Then, Calculate the reward r_t

(e) **Store Transition:** Store the transition tuple (s_t, a_t, r_t, s_{t+1}) in the replay buffer D.

2. Training/Update Phase: After collecting experiences for an episode, update the Actor and Critic networks using the data in D following the PPO algorithm (Section 2.4.2.3.2).

- Termination: The iterative loop (Step 2) continues until a predefined maximum number of iterations is completed.

4.1.4.4 Results and Discussion

The simulation results are presented in this section to demonstrate the effectiveness of the proposed DRL-based joint optimization framework.

Figure 4.7 shows the average reward and energy efficiency (EE) convergence over training iterations for different learning rates and optimization schemes. The results indicate that the full optimization scheme consistently achieves the highest average reward and EE, which validates the benefit of jointly optimizing all parameters. A learning rate of 10^{-3} provides the best performance and stable convergence compared to 10^{-2} and 10^{-4}. Furthermore, a comparison of baseline schemes (No Phase Optimization, Uniform PA, Predefined MM) with the full optimization reveals the relative importance of each optimized component. PA optimization delivers the most significant gain ($\approx 1142\%$ EE increase), followed by STAR-RIS phase shift optimization ($\approx 169\%$ EE increase), and MM optimization ($\approx 20\%$ EE increase). This result emphasizes the critical role of interference management through optimized PA in SCMA systems within this framework.

Figure 4.8 compares the average EE of different technology combinations (SCMA/PD-NOMA with STAR-RIS/RIS) as the number of STAR-RIS elements N varies. The proposed SCMA + STAR-RIS configuration significantly outperforms all baseline combinations (SCMA+RIS, PD-NOMA+STAR-RIS, PD-NOMA+RIS) across all tested values of N. The EE generally increases with N for all configurations, as more elements provide greater beamforming gain and signal manipulation capability, with the improvement particularly noticeable for SCMA-based systems. STAR-RIS consistently outperforms conventional RIS, which is attributed to the 360° coverage provided by STAR-RIS compared to the 180° reflection-only coverage of RIS, which enhances service to users on both sides of the surface. Similarly, SCMA consistently outperforms PD-NOMA, as SCMA manages interference more effectively through sparse codebook design and MM

optimization and offers greater flexibility to the DRL agent compared to simple power-domain superposition. Specifically, at $N = 100$, the SCMA + STAR-RIS system achieves an EE gain of approximately ~ 0.65 bits/Hz/Joule over PD-NOMA + STAR-RIS, and ~ 0.55 bits/Hz/Joule over SCMA + RIS, which quantifies the benefits reported for both the advanced multiple access scheme and the STAR-RIS architecture.

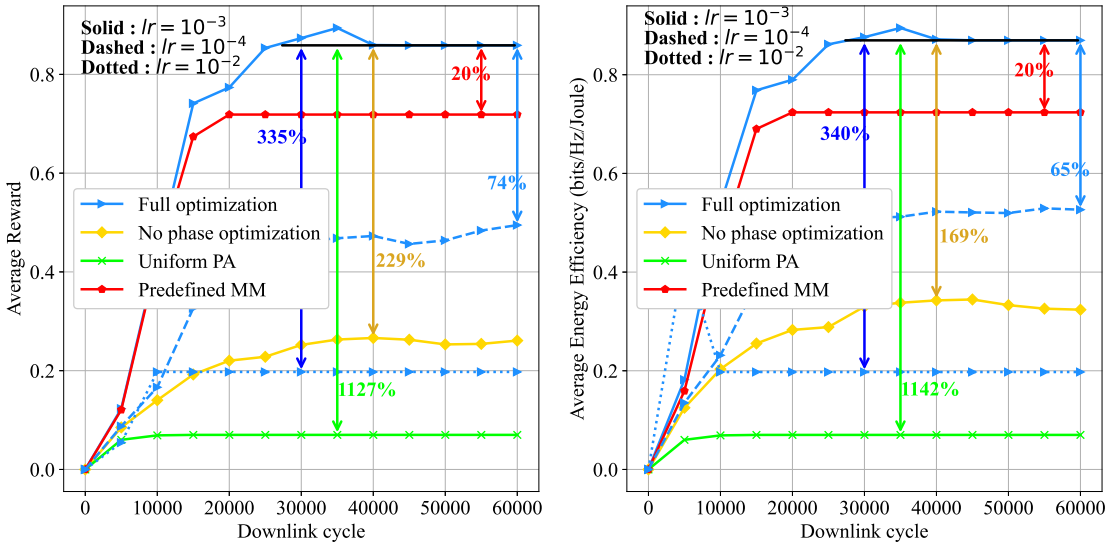


FIGURE 4.7: Learning performance of different optimization baselines (a) Average reward. (b) Average EE.

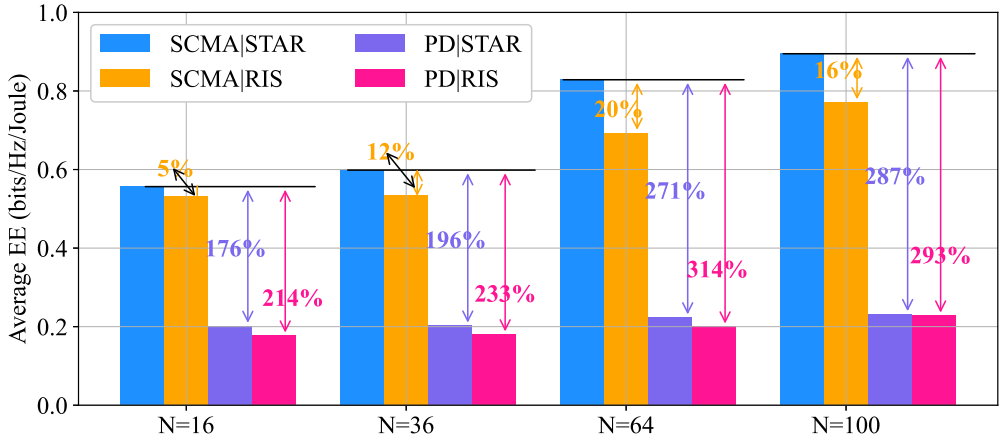


FIGURE 4.8: Average EE comparison of different technology configurations vs. N.

The proposed framework for STAR-RIS assisted SCMA-UAV network successfully demonstrates the potential of using a DRL (PPO) agent to jointly optimize the UAV trajectory, SCMA parameters (MM, PA matrix), transmit power, and STAR-RIS phase shifts for maximizing EE in a complex UAV network. The results confirm the superiority of the fully optimized "SCMA and STAR-RIS" system over alternatives like PD-NOMA, conventional RIS, and schemes where parts of the optimization (PA, MM, phase shifts)

are omitted. These results highlight the significant EE gains achievable through intelligent resource allocation and the enhanced coverage provided by STAR-RIS technology.

4.2 Conclusion

This chapter presented a comprehensive evaluation of the proposed DRL-driven NOMA-UAV framework, validating its efficacy in addressing the dual objectives of EE and fairness for IoT communications. The design of the framework integrates dynamic UAV trajectory optimization, adaptive NOMA PA, and fairness-aware resource management, underpinned by a novel reward mechanism that dynamically balances EE and fairness using the weight parameter λ_I. On ther other hand, the chapter introduced a system model and simulation framework for a STAR-RIS-assisted SCMA-UAV network, where a DRL agent was employed to optimize multiple parameters for maximum EE. The results demonstrated the effectiveness of the proposed framework in achieving significant EE gains and highlighted the advantages of using STAR-RIS technology in conjunction with SCMA.

Comparative analysis against baseline schemes—such as static UAV trajectories, QoS-based PA, and HDF—reveals key results, including a 30% improvement in energy efficiency, a fairness index exceeding 0.85 (Jain's fairness), and a 40% reduction in latency. These results underscore the ability of the DRL agent to adapt to dynamic channel states and user demands. Complexity analysis shows manageable training convergence (which stabilizes after 1,500 episodes) and scalability across varying IoT densities, which affirms practical deployability.

The performance of these frameworks highlights the transformative potential of AI-driven optimization in UAV-assisted IoT networks, particularly for applications that require robust, equitable connectivity, such as disaster response and smart agriculture.

General Conclusion and Future Directions

The increasing demand for efficient and intelligent wireless communication has driven considerable advancements in IoT-based UAV networks. This thesis investigated the integration of NOMA, RIS, and DRL to improve EE and spectrum efficiency in systems of IoT communication assisted by UAV.

The goal of this research was to deal with major challenges in UAV-NOMA networks, such as energy-efficient resource allocation, user/IoT device fairness, trajectory, and RIS beamforming optimization. Traditional techniques of OMA suffer from spectrum inefficiency, while conventional optimization techniques struggle to manage the complexity of dynamic UAV networks. To cope with these issues, this thesis proposes an optimization framework that exploits DRL to enhance real-time decision-making of UAVs. The first chapter introduced the theoretical background of NOMA and its advantages over OMA, while the second chapter provided an overview of AI techniques, including ML, DL, and DRL. The third chapter presented a unified framework for integrating NOMA, UAVs, and RIS in IoT networks, while the fourth chapter validated the proposed framework through extensive simulations.

The main contributions of this thesis can be summarized as follows:

- Developed a DRL-based algorithm for joint trajectory, resource allocation, STAR-RIS phase-shift control, to intelligently optimize UAV-assisted IoT communication networks.
- Introduced a novel SCMA for STAR-RIS-assisted UAV systems framework.
- Proposed a reward-based learning mechanism that balances power allocation among devices while ensuring fair access to network resources.

- Conducted extensive simulation-based validation, proving the superiority of the proposed approach over conventional methods in terms of energy efficiency, fairness, and spectral utilization.

While this research introduces a theoretical foundation for AI-driven UAV-NOMA networks, some practical issues necessitate exploration. First, the computational complexity of DRL algorithms creates a bottleneck for real-time deployment, as training requires high processing power. Second, although simulations demonstrate major improvements in EE and spectrum utilization, real-world validation is important to address unmodeled environmental factors (e.g., wind turbulence, signal blockage) and hardware imperfections (e.g., RIS phase-shift inaccuracies). Third, the current framework optimizes the operation of a single UAV but does not address the intricacies of multi-UAV coordination, such as interference management between drones, distributed resource allocation, and collision avoidance in swarms. Future research should address these gaps; researchers must integrate lightweight DRL architectures, prototype hybrid digital-analog RIS hardware, and extend decision-making frameworks to decentralized multi-agent systems.

This research identifies promising directions for advancing UAV-assisted IoT networks. First, extending single-agent DRL to multi-agent RL could enable cooperative UAV swarms to overcome issues such as distributed interference management and collision avoidance through decentralized, consensus-driven policies. Second, hybrid AI frameworks that combine federated learning (for privacy-preserving distributed training) and transfer learning (for rapid adaptation to novel environments) could reduce training complexity while they enhance generalization across diverse IoT scenarios. Third, when UAV-NOMA is combined with 6G paradigms (e.g., terahertz bands, cell-free architectures, and semantic communications), it could unlock ultra-reliable low-latency performance, while quantum AI might accelerate NP-hard optimization tasks such as joint RIS-UAV beamforming. Finally, transitioning from simulations to real-world prototypes is critical to validate robustness against hardware non-idealities (e.g., RIS phase noise, UAV jitter) and dynamic channel impairments, which necessitates collaboration with industry stakeholders to standardize AI-driven protocols for scalable deployment.

This thesis clarifies the synergy between AI, NOMA, and STAR-RIS-assisted UAV networks for IoT communications, offering a practical implementation of theoretical principles and laying the foundation for the next generation of intelligent, fairness-aware, energy-efficient, and highly adaptable IoT networks driven by DRL.

Scientific Production

- **Journal Publications:**
  - B. I. -D. Ghomri, M. Y. Bendimerad and F. T. Bendimerad, "DRL-Driven Optimization for Energy Efficiency and Fairness in NOMA-UAV Networks," in *IEEE Communications Letters*, vol. 28, no. 5, pp. 1048-1052, May 2024.
  - B. I. -D. Ghomri, M. Y. Bendimerad and H. Shaeik, "SCMA for STAR-RIS-Assisted UAV Networks: A DRL Approach for Energy Efficiency Maximization," in *IEEE Communications Letters*, vol. 29, no. 6, pp. 1431-1435, June 2025.
- **International Conferences:**
  - B. I. -D. Ghomri, M. Y. Bendimerad and F. T. Bendimerad, "Utilizing Deep Reinforcement Learning for Optimal UAV 3D Placement in NOMA-UAV Networks," *2024 2nd International Conference on Electrical Engineering and Automatic Control (ICEEAC)*, Setif, Algeria, 2024, pp. 1-5.

Bibliography

[1] Y. Qi-Yue and L. Hong-Chi, "Multiple access technique for cellular wireless networks," *Encyclopedia of Wireless Networks*, pp. 943–946, 2020.

[2] F. Jiang, Z. Gong, K. Hao, and Y. Zhang, "Multiple access techniques," *Encyclopedia of Wireless Networks*, pp. 934–943, 2020.

[3] L. L. Hanzo, L.-L. Yang, E.-L. Kuan, and K. Yen, "Cdma overview," in *Single and Multi-Carrier DS-CDMA*. Wiley-IEEE Press, 2004, pp. 35–80.

[4] W. Zhang, C. Xie, and E. Pasalic, "Large sets of orthogonal sequences suitable for applications in cdma systems," *IEEE Transactions on Information Theory*, vol. 62, no. 6, pp. 3757–3767, 2016.

[5] L. Zhang and Y. Sun, "The optimal assignment of orthogonal polyphase sequences in cdma systems," *IEEE Communications Letters*, vol. 22, no. 1, pp. 109–112, 2018.

[6] 3GPP TR 25.913, "Technical specification group radio access network; requirements for evolved utra (e-utra) and evolved utran (e-utran)," *release 7*, Dec. 2009.

[7] M. Morelli, C.-C. J. Kuo, and M.-O. Pun, "Synchronization techniques for orthogonal frequency division multiple access (ofdma): A tutorial review," *Proceedings of the IEEE*, vol. 95, no. 7, pp. 1394–1427, 2007.

[8] H. Yin and S. Alamouti, "Ofdma: A broadband wireless access technology," in *2006 IEEE Sarnoff Symposium*, 2006, pp. 1–4.

[9] G. Berardinelli, L. A. M. Ruiz de Temino, S. Frattasi, M. I. Rahman, and P. Mogensen, "Ofdma vs. sc-fdma: performance comparison in local area imt-a scenarios," *IEEE Wireless Communications*, vol. 15, no. 5, pp. 64–72, 2008.

[10] H. G. Myung, J. Lim, and D. J. Goodman, "Single carrier fdma for uplink wireless transmission," *IEEE Vehicular Technology Magazine*, vol. 1, no. 3, pp. 30–38, 2006.

[11] C.-X. Wang et al., "On the road to 6g: Visions, requirements, key technologies, and testbeds," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 905–974, 2023.

[12] Y. Wang, B. Ren, S. Sun, S. Kang, and X. Yue, "Analysis of non-orthogonal multiple access for 5g," *China Communications*, vol. 13, no. Supplement2, pp. 52–66, 2016.

[13] Z. Wei, J. Yuan, D. W. K. Ng, M. Elkashlan, and Z. Ding, "A survey of downlink non-orthogonal multiple access for 5g wireless communication networks," *arXiv preprint arXiv:1609.01856*, 2016.

[14] L. Dai, B. Wang, Z. Ding, Z. Wang, S. Chen, and L. Hanzo, "A survey of non-orthogonal multiple access for 5g," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2294–2323, 2018.

[15] S. M. R. Islam, N. Avazov, O. A. Dobre, and K.-s. Kwak, "Power-domain non-orthogonal multiple access (noma) in 5g systems: Potentials and challenges," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 721–742, 2017.

[16] Z. Ding, X. Lei, G. K. Karagiannidis, R. Schober, J. Yuan, and V. K. Bhargava, "A survey on non-orthogonal multiple access for 5g networks: Research challenges and future trends," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 10, pp. 2181–2195, 2017.

[17] R. Hoshyar, F. P. Wathan, and R. Tafazolli, "Novel low-density signature for synchronous cdma systems over awgn channel," *IEEE Transactions on Signal Processing*, vol. 56, no. 4, pp. 1616–1626, 2008.

[18] M. Al-Imari and M. A. Imran, "Low density spreading multiple access," *Multiple Access Techniques for 5G Wireless Networks and Beyond*, pp. 493–514, 2019.

[19] R. Hoshyar, R. Razavi, and M. Al-Imari, "Lds-ofdm an efficient multiple access technique," in *2010 IEEE 71st Vehicular Technology Conference*, 2010, pp. 1–5.

[20] Z. Yuan, G. Yu, W. Li, Y. Yuan, X. Wang, and J. Xu, "Multi-user shared access for internet of things," in *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, 2016, pp. 1–5.

[21] 3GPP, "Non-orthogonal multiple access for new radio," *TSG RAN WG1 #85*, Nanjing, China, 23rd-27th, May. 2016.

[22] 3GPP, "Considerations on dl/ul multiple access for nr," *TSG RAN WG1 #84bis*, Busan, Korea, 11th-15th, Apr. 2016.

[23] 3GPP, "New uplink non-orthogonal multiple access schemes for nr," *TSG RAN WG1 #86*, Gothenburg, Sweden, 22nd-26th, Aug. 2016.

[24] H. Nikopour and H. Baligh, "Sparse code multiple access," in *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2013, pp. 332–336.

[25] S. Chaturvedi, Z. Liu, V. A. Bohara, A. Srivastava, and P. Xiao, "A tutorial on decoding techniques of sparse code multiple access," *IEEE Access*, vol. 10, pp. 58 503–58 524, 2022.

[26] M. Vameghestahbanati, I. D. Marsland, R. H. Gohary, and H. Yanikomeroglu, "Multidimensional constellations for uplink scma systems—a comparative study," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2169–2194, 2019.

[27] M. Taherzadeh, H. Nikopour, A. Bayesteh, and H. Baligh, "Scma codebook design," in *2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall)*, 2014, pp. 1–5.

[28] Y. Zhou, Q. Yu, W. Meng, and C. Li, "Scma codebook design based on constellation rotation," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–6.

[29] P. Vandenameele, L. Van der Perre, and M. Engels, *Space division multiple access for wireless local area networks*. Springer Science & Business Media, 2006, vol. 631.

[30] S. Hu, B. Yu, C. Qian, Y. Xiao, Q. Xiong, C. Sun, and Y. Gao, "Nonorthogonal interleave-grid multiple access scheme for industrial internet of things in 5g network," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 12, pp. 5436–5446, 2018.

[31] Y. Cao, H. Sun, J. Soriaga, and T. Ji, "Resource spread multiple access - a novel transmission scheme for 5g uplink," in *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, 2017, pp. 1–5.

[32] S. Chen, B. Ren, Q. Gao, S. Kang, S. Sun, and K. Niu, "Pattern division multiple access—a novel nonorthogonal multiple access for fifth-generation radio networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3185–3196, 2017.

[33] W. B. Ameur, P. Mary, M. Dumay, J.-F. Hélard, and J. Schwoerer, "Performance study of mpa, log-mpa and max-log-mpa for an uplink scma scenario," in *2019 26th International Conference on Telecommunications (ICT)*, 2019, pp. 411–416.

[34] A. Anwar, B.-C. Seet, and X. J. Li, "Pic-based receiver structure for 5g downlink noma," in *2015 10th International Conference on Information, Communications and Signal Processing (ICICS)*, 2015, pp. 1–5.

[35] Y. Endo, Y. Kishiyama, and K. Higuchi, "Uplink non-orthogonal access with mmse-sic in the presence of inter-cell interference," in *2012 International Symposium on Wireless Communication Systems (ISWCS)*, 2012, pp. 261–265.

[36] Y. Mao, O. Dizdar, B. Clerckx, R. Schober, P. Popovski, and H. V. Poor, "Rate-splitting multiple access: Fundamentals, survey, and future research trends," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 2073–2126, 2022.

[37] Y. Al-Eryani and E. Hossain, "The d-oma method for massive multiple access in 6g: Performance, security, and challenges," *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 92–99, 2019.

[38] Y. Liu, S. Zhang, X. Mu, Z. Ding, R. Schober, N. Al-Dhahir, E. Hossain, and X. Shen, "Evolution of noma toward next generation multiple access (ngma) for 6g," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 4, pp. 1037–1071, 2022.

[39] S. Russell, P. Norvig, F. Popineau, L. Miclet, and C. Cadet, *Intelligence artificielle: une approche moderne (4th edition)*. Pearson France, 2021.

[40] B. Delipetrev, C. Tsinaraki, and U. Kostić, "Ai watch, historical evolution of artificial intelligence: Analysis of the three main paradigm shifts in ai," *European Commission, Joint Research Centre*, 2020.

[41] M. Haenlein and A. Kaplan, "A brief history of artificial intelligence: On the past, present, and future of artificial intelligence," *California management review*, vol. 61, no. 4, pp. 5–14, 2019.

[42] O. Simeone, "A very brief introduction to machine learning with applications to communication systems," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 4, pp. 648–664, 2018.

[43] P. Louridas and C. Ebert, "Machine learning," *IEEE Software*, vol. 33, no. 5, pp. 110–115, 2016.

[44] H. Li, *Machnie Learning Methods*. Springer Nature, 2024.

[45] I. Goodfellow, "Deep learning," 2016.

[46] W. Wang, *Principles of Machine Learning: The Three Perspectives*. Springer Nature, 2024.

[47] S. V. Mahadevkar, B. Khemani, S. Patil, K. Kotecha, D. R. Vora, A. Abraham, and L. A. Gabralla, "A review on machine learning styles in computer vision—techniques and future directions," *Ieee Access*, vol. 10, pp. 107 293–107 329, 2022.

[48] A. Charu C, *Neural networks and deep learning: a textbook*, 2nd ed. Springer, 2023.

[49] P. Kim, *MATLAB Deep Learning.* Apress, 2017.

[50] M. Marvin and A. P. Seymour, "Perceptrons," *Cambridge, MA: MIT Press*, vol. 6, no. 318-362, p. 7, 1969.

[51] M. Martinez-Ramon, M. Ajith, and A. Kurup, *Deep Learning: A Practical Introduction.* Wiley, 2024.

[52] C. M. Bishop and H. Bishop, *Deep learning: Foundations and concepts.* Springer Nature, 2024.

[53] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *Neurocomputing*, vol. 503, pp. 92–108, 2022.

[54] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[55] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[56] R. Moradi, R. Berangi, and B. Minaei, "A survey of regularization strategies for deep models," *Artificial Intelligence Review*, vol. 53, no. 6, pp. 3947–3986, 2020.

[57] Y. Wu and K. He, "Group normalization," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[58] J. Ba and D. Kingma, "Adam: A method for stochastic optimization," in *Proceedings 3rd International Conference on Learning Representations (ICLR)*, 2015.

[59] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, pp. 1–74, 2021.

[60] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999–7019, 2022.

[61] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, S. Pan, and H. Liu, "Graph learning: A survey," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 2, pp. 109–127, 2021.

[62] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, "Q-learning algorithms: A comprehensive classification and applications," *IEEE Access*, vol. 7, pp. 133 653–133 667, 2019.

[63] V. Mnih, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[64] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[65] S. E. Li, *Reinforcement learning for sequential decision and optimal control*. Springer, 2023.

[66] T. Lillicrap, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[67] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[68] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[69] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys &Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[70] F. Guo, F. R. Yu, H. Zhang, X. Li, H. Ji, and V. C. M. Leung, "Enabling massive iot toward 6g: A comprehensive survey," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 11 891–11 915, 2021.

[71] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, D. Niyato, O. Dobre, and H. V. Poor, "6g internet of things: A comprehensive survey," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 359–383, 2022.

[72] A. K. Vishwakarma, S. Chaurasia, K. Kumar, Y. N. Singh, and R. Chaurasia, "Internet of things technology, research, and challenges: a survey," *Multimedia Tools and Applications*, pp. 1–36, 2024.

[73] M. Vaezi, A. Azari, S. R. Khosravirad, M. Shirvanimoghaddam, M. M. Azari, D. Chasaki, and P. Popovski, "Cellular, wide-area, and non-terrestrial iot: A survey on 5g advances and the road toward 6g," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 2, pp. 1117–1174, 2022.

[74] B. P. Sahoo, C.-C. Chou, C.-W. Weng, and H.-Y. Wei, "Enabling millimeter-wave 5g networks for massive iot applications: A closer look at the issues impacting millimeter-waves in consumer devices under the 5g framework," *IEEE Consumer Electronics Magazine*, vol. 8, no. 1, pp. 49–54, 2019.

[75] A. Yarali, *AI, 5G, and IoT*. Wiley-IEEE Press, 2022, pp. 117–131.

[76] F. C. Andriulo, M. Fiore, M. Mongiello, E. Traversa, and V. Zizzo, "Edge computing and cloud computing for internet of things: A review," in *Informatics*, vol. 11, no. 4. MDPI, 2024, p. 71.

[77] T. Theodorou and L. Mamatas, "Denis-sdn: Software-defined network slicing solution for dense and ultra-dense iot networks," *arXiv preprint arXiv:2312.13662*, 2023.

[78] A. Shafie, N. Yang, C. Han, J. M. Jornet, M. Juntti, and T. Kürner, "Terahertz communications for 6g and beyond wireless networks: Challenges, key advancements, and opportunities," *IEEE Network*, vol. 37, no. 3, pp. 162–169, 2023.

[79] J. Mendez, K. Bierzynski, M. P. Cuéllar, and D. P. Morales, "Edge intelligence: concepts, architectures, applications, and future directions," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 21, no. 5, pp. 1–41, 2022.

[80] Y. Liu, X. Liu, X. Mu, T. Hou, J. Xu, M. Di Renzo, and N. Al-Dhahir, "Reconfigurable intelligent surfaces: Principles and opportunities," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1546–1577, 2021.

[81] M. N. M. Bhutta, A. A. Khwaja, A. Nadeem, H. F. Ahmad, M. K. Khan, M. A. Hanif, H. Song, M. Alshamari, and Y. Cao, "A survey on blockchain technology: Evolution, architecture and security," *Ieee Access*, vol. 9, pp. 61 048–61 073, 2021.

[82] M. Attaran and B. G. Celik, "Digital twin: Benefits, use cases, challenges, and opportunities," *Decision Analytics Journal*, vol. 6, p. 100165, 2023.

[83] S. M. A. A. Abir, A. Anwar, J. Choi, and A. S. M. Kayes, "Iot-enabled smart energy grid: Applications and challenges," *IEEE Access*, vol. 9, pp. 50 961–50 981, 2021.

[84] R. Lohiya and A. Thakkar, "Application domains, evaluation data sets, and research challenges of iot: A systematic review," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 8774–8798, 2021.

[85] M. M. Islam, S. Nooruddin, F. Karray, and G. Muhammad, "Internet of things: Device capabilities, architectures, protocols, and smart applications in healthcare domain," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 3611–3641, 2023.

[86] G. Bennett, W. Crowder, and C. Baxter, "Challenges and opportunities of iot for defense and national security logistics," *IoT for Defense and National Security*, pp. 83–96, 2022.

[87] W. Saad, M. Bennis, M. Mozaffari, and X. Lin, *Wireless Communications and Networking for Unmanned Aerial Vehicles*. Cambridge University Press, 2020.

[88] P. G. Fahlstrom, T. J. Gleason, and M. H. Sadraey, *Introduction to UAV systems*, 5th ed. John Wiley & Sons, 2022.

[89] A. Fotouhi, H. Qiang, M. Ding, M. Hassan, L. M. Giordano, A. Garcia-Rodriguez, and J. Yuan, "Survey on uav cellular communications: Practical aspects, standardization advancements, regulation, and security challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3417–3442, 2019.

[90] Y. Zeng, Q. Wu, and R. Zhang, "Accessing from the sky: A tutorial on uav communications for 5g and beyond," *Proceedings of the IEEE*, vol. 107, no. 12, pp. 2327–2375, 2019.

[91] Y. Zeng and R. Zhang, "Energy-efficient uav communication with trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3747–3760, 2017.

[92] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing uav," *IEEE Transactions on Wireless Communications*, vol. 18, no. 4, pp. 2329–2345, 2019.

[93] M. Mozaffari, W. Saad, M. Bennis, Y.-H. Nam, and M. Debbah, "A tutorial on uavs for wireless networks: Applications, challenges, and open problems," *IEEE communications surveys & tutorials*, vol. 21, no. 3, pp. 2334–2360, 2019.

[94] 3GPP, "3rd generation partnership project; technical specification group services and system aspects; enhancement for unmanned aerial vehicles; stage 1 (release 17)," 3GPP, Tech. Rep. 3GPP TR 22.829 V17.1.0 (2019-09), 2019.

[95] L. H. BAI, Z. Huang, and X. Cheng, *Propagation Characterization and Channel Modeling for UAV Communication*. Springer, 2024.

[96] R. Amorim, H. Nguyen, P. Mogensen, I. Z. Kovács, J. Wigard, and T. B. Sørensen, "Radio channel modeling for uav communication over cellular networks," *IEEE Wireless Communications Letters*, vol. 6, no. 4, pp. 514–517, 2017.

[97] A. Al-Hourani, S. Kandeepan, and S. Lardner, "Optimal lap altitude for maximum coverage," *IEEE Wireless Communications Letters*, vol. 3, no. 6, pp. 569–572, 2014.

[98] 3GPP, "3rd generation partnership project; technical specification group radio access network; study on enhanced lte support for aerial vehicles (release 15)," 3GPP, Tech. Rep. 3GPP TR 36.777 V0.3.1 (2017-10), 2017.

[99] R. Zhang, Q. Guo, D. Zhai, D. Zhou, X. Du, and M. Guizani, "Channel measurement and resource allocation scheme for dual-band airborne access networks," *IEEE Access*, vol. 7, pp. 80 870–80 883, 2019.

[100] J. del Cerro, C. Cruz Ulloa, A. Barrientos, and J. de León Rivas, "Unmanned aerial vehicles in agriculture: A survey," *Agronomy*, vol. 11, no. 2, 2021.

[101] S. H. Alsamhi, A. V. Shvetsov, S. Kumar, S. V. Shvetsova, M. A. Alhartomi, A. Hawbani, N. S. Rajput, S. Srivastava, A. Saif, and V. O. Nyangaresi, "Uav computing-assisted search and rescue mission framework for disaster and harsh environment mitigation," *Drones*, vol. 6, no. 7, 2022.

[102] Z. Wei, M. Zhu, N. Zhang, L. Wang, Y. Zou, Z. Meng, H. Wu, and Z. Feng, "Uav-assisted data collection for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 15 460–15 483, 2022.

[103] R. M. Rolly, P. Malarvezhi, and T. D. Lagkas, "Unmanned aerial vehicles: Applications, techniques, and challenges as aerial base stations," *International Journal of Distributed Sensor Networks*, vol. 18, no. 9, pp. 1–24, 2022.

[104] N. Abbas, Z. Abbas, X. Liu, S. S. Khan, E. D. Foster, and S. Larkin, "A survey: Future smart cities based on advance control of unmanned aerial vehicles (uavs)," *Applied Sciences*, vol. 13, no. 17, 2023.

[105] M. A. Hoque, M. Hossain, S. Noor, S. M. R. Islam, and R. Hasan, "Iotaas: Drone-based internet of things as a service framework for smart cities," *IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 12 425–12 439, 2022.

[106] U. K. Gera, D. K. Saini, P. Singh, and D. Siddharth, "Iot-based uav platform revolutionized in smart healthcare," *Unmanned aerial vehicles for internet of things (IoT) concepts, techniques, and applications*, pp. 277–293, 2021.

[107] M. Di Renzo, A. Zappone, M. Debbah, M.-S. Alouini, C. Yuen, J. de Rosny, and S. Tretyakov, "Smart radio environments empowered by reconfigurable intelligent surfaces: How it works, state of research, and the road ahead," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 11, pp. 2450–2525, 2020.

[108] J. Huang, C.-X. Wang, Y. Sun, R. Feng, J. Huang, B. Guo, Z. Zhong, and T. J. Cui, "Reconfigurable intelligent surfaces: Channel characterization and modeling," *Proceedings of the IEEE*, vol. 110, no. 9, pp. 1290–1311, 2022.

[109] Q. Wu, S. Zhang, B. Zheng, C. You, and R. Zhang, "Intelligent reflecting surface-aided wireless communications: A tutorial," *IEEE Transactions on Communications*, vol. 69, no. 5, pp. 3313–3351, 2021.

[110] J. Xu, Y. Liu, X. Mu, and O. A. Dobre, "Star-riss: Simultaneous transmitting and reflecting reconfigurable intelligent surfaces," *IEEE Communications Letters*, vol. 25, no. 9, pp. 3134–3138, 2021.

[111] X. Mu, Y. Liu, L. Guo, J. Lin, and R. Schober, "Simultaneously transmitting and reflecting (star) ris aided wireless communications," *IEEE Transactions on Wireless Communications*, vol. 21, no. 5, pp. 3083–3098, 2022.

[112] Y. Liu, X. Mu, J. Xu, R. Schober, Y. Hao, H. V. Poor, and L. Hanzo, "Star: Simultaneous transmission and reflection for 360° coverage by intelligent surfaces," *IEEE Wireless Communications*, vol. 28, no. 6, pp. 102–109, 2021.

[113] W. Khalid, Z. Kaleem, R. Ullah, T. Van Chien, S. Noh, and H. Yu, "Simultaneous transmitting and reflecting-reconfigurable intelligent surface in 6g: Design guidelines and future perspectives," *IEEE Network*, vol. 37, no. 5, pp. 173–181, 2023.

[114] T. Zhang, Z. Wang, Y. Liu, W. Xu, and A. Nallanathan, "Caching placement and resource allocation for cache-enabling uav noma networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 12 897–12 911, 2020.

[115] A. Benfaid, N. Adem, and B. Khalfi, "Adaptsky: A drl based resource allocation framework in noma-uav networks," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 01–07.

[116] S. K. Mahmud, Y. Chen, and K. K. Chai, "Hybrid decision framework for energy efficiency enhancement in noma-uav networks," *IEEE Communications Letters*, vol. 26, no. 6, pp. 1378–1382, 2022.

[117] R. Zhong, X. Liu, Y. Liu, and Y. Chen, "Multi-agent reinforcement learning in noma-aided uav networks for cellular offloading," *IEEE Transactions on Wireless Communications*, vol. 21, no. 3, pp. 1498–1512, 2022.

[118] Z. Zhang, C. Xu, Z. Li, X. Zhao, and R. Wu, "Deep reinforcement learning for aerial data collection in hybrid-powered noma-iot networks," *IEEE Internet of Things Journal*, vol. 10, no. 2, pp. 1761–1774, 2023.

[119] L. Zhang, Y. Zhang, J. Lu, Y. Xiao, and G. Zhang, "Deep reinforcement learning based trajectory design for customized uav-aided noma data collection," *IEEE Wireless Communications Letters*, vol. 13, no. 12, pp. 3365–3369, 2024.

[120] P. Qin, Y. Fu, J. Zhang, S. Geng, J. Liu, and X. Zhao, "Drl-based resource allocation and trajectory planning for noma-enabled multi-uav collaborative caching 6g network," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 6, pp. 8750–8764, 2024.

[121] B. I.-D. Ghomri, M. Y. Bendimerad, and F. T. Bendimerad, "Drl-driven optimization for energy efficiency and fairness in noma-uav networks," *IEEE Communications Letters*, vol. 28, no. 5, pp. 1048–1052, 2024.

[122] I. Budhiraja, V. Vishnoi, N. Kumar, D. Garg, and S. Tyagi, "Energy-efficient optimization scheme for ris-assisted communication underlaying uav with noma," in *ICC 2022 - IEEE International Conference on Communications*, 2022, pp. 1–6.

[123] F. Wang and X. Zhang, "Active-irs-enabled energy-efficiency optimizations for uav-based 6g mobile wireless networks," in *2023 57th Annual Conference on Information Sciences and Systems (CISS)*, 2023, pp. 1–6.

[124] H. Zhang, M. Huang, H. Zhou, X. Wang, N. Wang, and K. Long, "Capacity maximization in ris-uav networks: A ddqn-based trajectory and phase shift optimization approach," *IEEE Transactions on Wireless Communications*, vol. 22, no. 4, pp. 2583–2591, 2023.

[125] K. Guo, M. Wu, X. Li, H. Song, and N. Kumar, "Deep reinforcement learning and noma-based multi-objective ris-assisted is-uav-tns: Trajectory optimization and beamforming design," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 9, pp. 10 197–10 210, 2023.

[126] J. Lei, T. Zhang, X. Mu, and Y. Liu, "Noma for star-ris assisted uav networks," *IEEE Transactions on Communications*, vol. 72, no. 3, pp. 1732–1745, 2024.