# POLITECNICO DI TORINO

Master's Degree in Communication and Computer Network Engineering


Politecnico di Torino

Master's Degree Thesis

# Performance evaluation and design of ML-based solutions for the support of mobile services in 5G systems

Supervisors

Prof. CARLA FABIANA CHIASSERINI

Prof. CLAUDIO ETTORE CASETTI

Candidate

TAHMINEH JAVADZADEH

July 2021

# Summary

The vision of the fifth generation of mobile networks (5G) lies in bringing enhanced performances with respect to the previous mobile technologies. 5G has been planned to provide revolutionary high throughput, extremely low latency, for miscellaneous devices with massive and ubiquitous connections.
To achieve these goals, many advanced technologies have been introduced and utilized in 5G, like massive MIMO, mmWave, efficient Radio Resource Management (RRM) techniques, etc. Among all, an efficient RRM could have a significant impact on effective spectrum utilization, massive connections. One thriving solution is represented by virtual Radio Access Network (vRAN) technology and is profitable in terms of cost and scalability for the mobile network operators. Indeed, in virtualizing the RAN, radio processing intelligence, which was initially performed by purpose-built hardware, will be performed at higher levels of the network. vRAN adds the ability to scale the network resources assigned to the various demanding entities in 5G network. This is achieved by separating networking functions from hardware, a technique that overcomes many technical challenges in the integration with legacy technologies.

Although vRAN has brought flexibility and cost reduction in terms of hardware, it has introduced new challenges. Indeed, by increasing the number of connected devices with heterogeneous demands to vRANs, the network performance will deteriorate. Despite the mentioned issues, the conventional mechanisms might not be sufficient to fulfill the optimal performance and resource allocation. Therefore, a more efficient and intelligent RRM is required, which can dynamically scale and allocate radio resources. Therefore, in recent years Machine Learning (ML) and in particular Reinforcement Learning (RL), has introduced versatile applications in the telecommunication area, and several intelligent resource allocation mechanisms have been proposed.

In this regard, the main objective of this thesis is to design and to study online learning mechanisms based on RL-based and deep RL-based RRM for vRAN, to analyze their performance in typical network scenarios, and to compare the possible

enhancement of both methods. The proposed solutions are aimed at real-time and dynamic RRM according to user demands in vRANs and they are designed to deal with the dynamics associated with the radio environment.

The simulations have been carried out using the Network Simulator 3 to prepare network scenarios for the generation of traffic between vRAN and multiple users receiving different channel conditions. For the main scenario, it is considered to implement not only multi-user but also non-stationary users, in presence of one eNodeB. This scenario allows analyzing the proposed approaches under more complex situations. The first approach is grounded on a differential semi-gradient State-Action-Reward-State-Action (SARSA) mechanism, and the second approach is based on Deep Q Learning (DQL) mechanism for real-time RRM in vRAN. The RL agent is responsible to receive the channel conditions, which are considered as Channel Quality Indicator (CQI) and transmission buffer head of line delay. Then following a greedy policy, the agent selects optimal Modulation and Coding Scheme (MCS) values to maximize the average reward which denotes the user's throughput. The proposed RL-based solutions are promising and can meet the demands of the heterogeneous spectrum of users.

# Acknowledgements

Firstly, I would like to thank my parents for their supports in all aspects of my life and sympathetic ear. You are always there for me.

Then I would like to thank the best in my life, Vahid Sadat, who always helped me and didn't let me feel alone in these tough periods.

In addition, I would like to thank my supervisor Professor Carla Chiasserini, her insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

Yours Sincerely
Tahmineh Javadzadeh

# Table of Contents

# List of Tables

# List of Figures

# Acronyms:

**AI:** Artificial Intelligence

**AMC** Adoptive Modulation and Coding

**BBU** BaseBand Unit

**BS** Base Station

**BSR** Buffer Statuse Report

**CAPEX** CAPital EXpenditure

**CN** Core Netwoek

**CQI** Channel Quality Indicator

**DP** Dynamic Programming

**DQL** Deep Q Learning

**DQN** Deep Q Network

**EPC** Evolved Packet Core

**eMBB** Enhanced Mobile Broadband

**GPRS** General Packet Radio Service

**GSM** Global System for Mobile Communication

**ITU** International Telecommunication Union

**LTE** Long Term Evolution

**MCS** Modulation and Coding Scheme

**MDP** Markov Decision Process

**ML** Machine Learning

**MSE** Mean Squared Error

**NFV** Network Function Virtualization

**NN** Neural Network

**OFDM** Orthogonal Frequency Division Multiple Access

**OPEX** OPerational EXpenditure

**p-GW** Packet gateway

**QoS** Quality of Service

**RAN** Radio Access Network

**RB** Resource Block

**Relu** Rectified Linear Unit

**RL** Reinforcement Learning

**RRC** Radio Resource Control

**RRU** Remote Radio Unit

**s-GW** Serving gateway

**SC-FDMA** Single Carrier Frequency Division Multiple Access

**UE** User Equipment

**UMTS** Universal Mobile Telecommunication System

**vRAN** Virtual Radio Access Network

**WAN** Wide Area Network

**WCDMA** Wide-Band Code Division Multiple Access

# Chapter 1

# Introduction

The continuous growth in wireless user devices and data usages increases the intense demand to evolve and innovate new technologies. As it has been described in detail in the Cisco visual networking index [1], by 2023, the number of miscellaneous devices connected to the IP networks will exceed the global population by a factor of 3. Cisco [1] also has expected that sheer volume of data and connections, over 10 percent of global mobile devices, will be dedicated to 5G connections. In addition to an exponential increase in the number of connected devices, they will have various demands on the network. For instance, mobile applications such as video streams or cloud games demand high throughput connections and connected cars require ultra-reliable and low latency communications, etc.

These demands always have made forced to evolve and constantly add new features to communication. Going back to early 1980, the first generation mobile communication known as 1G emerged. 1G was analog communication and only supports voice services. This technology suffered from poor coverage, low sound quality, and many other drawbacks. Despite all the shortages, 1G paved the way for the next generations of mobile technologies.

The second generation is known as 2G mobile networks has been launched under Global System for Mobile Communications (GSM) standard in 1991. 2G emerged as the first digital communication, which improved voice calls, also introduced more features like data services like text messages and multimedia messages. In 2G data rate started with 9.6 kbits/s, finally, it offered 40 kbits/s. GSM technology has continuously tried to offer better services and led to 2.5G known as General Packet Radio Service (GPRS), deployed based on the original GSM system. GPRS supports packet-based services also provides improved data rate, i.e., from 56 Kbits/s up to 384 Kbits/s.

The third generation of mobile networks, 3G mobile cellular system, based on the GSM standards is called universal mobile telecommunications system short for UMTS. 3G is enabled by more complex access technology Wide-band Code Division Multiple Access (WCDMA), which led to introducing increased data transfer capabilities (4 times faster than 2G) and new services such as enhanced video and audio streaming. 3G mobile communication systems have provided voice and paging services, video calling, mobile internet, and a variety of other services. In addition, 3G has offered wide area network (WAN) coverage of peak data rate, 384 kbit/s for non-stationary users and limited data rate, 2 Mbits/s for stationary users. However, providing broadband services would be one of the major goals of the 4G Wireless systems.

The fourth generation of mobile networks, 4G, also known as LTE, firstly has been deployed commercially in 2009. In LTE, one of the major differences with the previous generation is that in downlink orthogonal Frequency Division Multiple Access (OFDMA) and in uplink Single Carrier Frequency Division Multiple Access (SC-FDMA) are used [2]. Because of OFDMA, LTE has high spectral efficiency, and also it is robust against interference. Additionally, data speed in LTE has been increased remarkably to support huge data access by various services. LTE uses scalable channel bandwidths of 5–20 MHz and supports a high peak data rate, more than 100 Mbits/s for high-speed mobility and 1 Gbit/s for low-speed mobility in the downlink.

Some crucial limitations of conventional mobile networks were the motivation to introduce the 5G mobile networks. 5G, the fifth generation of mobile technologies specified by 3GPP, as stated in [3] introduces significant performance improvements depicted in 1.1. Indeed as described in [1], mobile data traffic is increasing extremely, chiefly because of video games and high-quality video streaming, which require a sheer volume of data to be transferred in a significantly short time and with negligible latency. Additionally, the growing number of connections, miscellaneous devices, energy efficiency requirements, reducing operational expenditures, etc., explain the necessities of 5G to bring innovative features. Therefore, 5G introduces remarkable operational performance, e.g., increased spectral efficiency, higher data rates, low latency, ubiquitous coverage, and seamless connection with high-speed mobility. As it is declared in [4] the 4G and 5G comparison of key capabilities is shown in 1.1.
In this regard, three main usage cases are defined by International Telecommunication Union (ITU) [6] that 5G has to provide:

- Enhanced Mobile Broadband (eMBB): One of the core features of 5G targets is the eMBB. It focuses on the data rate and network capacity, to support

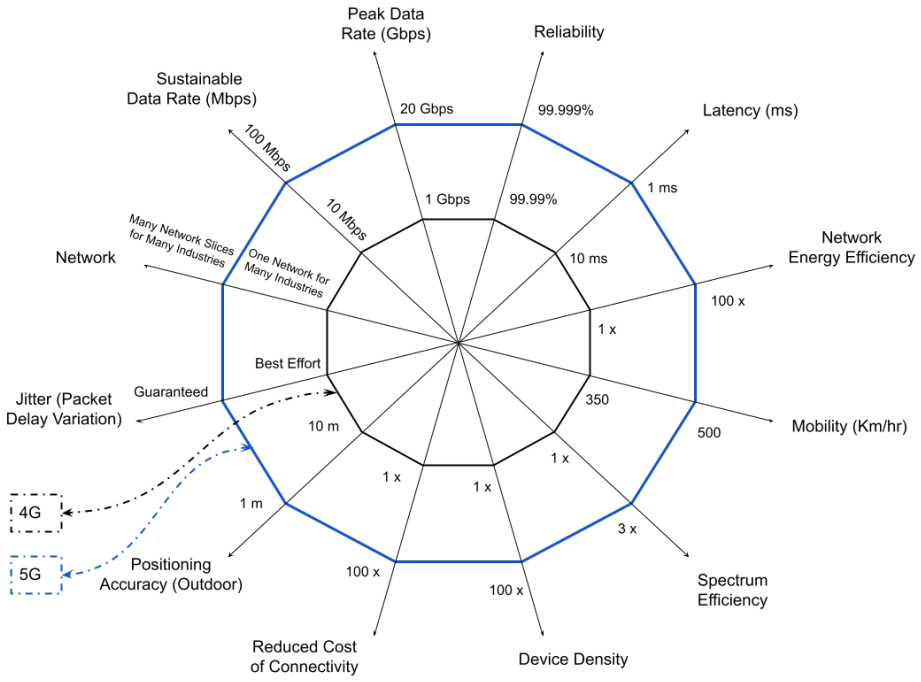



**Figure 1.1:** 5G features [5]

extreme growth of data rates, high user density, and huge traffic capacity for scenarios with massive connections as well as ubiquitous coverage. Some advanced technologies enable 5G to meet these requirements namely mmWave, massive MIMO, and beamforming, etc.

- Massive Machine-type Communications (mMTC): This key feature is introduced mainly for the Internet of Things (IoT) which connects very large numbers of devices, which requires both low data rates and low power consumption. mMTC takes care of scalable and efficient connectivity for a massive number e.g., billions of miscellaneous devices.

- Ultra-reliable and Low Latency Communications (URLLC): This feature is considered the key enabler of 5G to support mission critical applications e.g.,remote healthcare.

5G, as a dynamic and flexible framework of advanced technologies, has been an effective revolution in mobile communication technology which has been developed into both the Radio Access Network (RAN) and Core Network. In this regard, RAN in 5G, with respect to the previous mobile network generations is not limited to be in the same place with base station, and enabled with more intelligent architecture, which makes 5G network architecture much more service-oriented than previous

generations.

To achieve these goals, one thriving technology that is introduced in 5G is Network function virtualization (NFV). NFV enables 5G infrastructure by virtualizing applications i.e., by decoupling software from hardware. Thus, it replaces various network functions with virtualized instances running as software. As a result, complex hardware elements are not required anymore and many other costs related to installation time are eliminated. In the core network, many different functionalities are required, e.g., firewalls, routers, gateways, etc., previously, each vendor used to have proprietary hardware and software for each of these functions. Exploiting NFV, the commercial out-of-shelf (COTS) hardware is introduced. It allows operators to get standardized hardware at a much lower cost, enables different vendors to focus on software, rather than hardware.

Virtual Radio Access Network (vRAN) technology takes advantage of NFV and is profitable in terms of cost and scalability for the mobile network operators (MNO). vRAN enables new opportunities to provide open source development to ease the deployment of new features. Indeed, in virtualizing the RAN, radio processing intelligence which was initially performed by purpose-built hardware, will be performed at higher levels of the network. vRAN adds the ability to scale and adjust the network resources assigned to the various demanding entities in the 5G network.

# Chapter 2

# Radio Access Network of 5G

## 2.1 LTE Overview

Long-Term Evolution (LTE) is a standard for wireless broadband communication, developed by the 3rd Generation Partnership Project (3GPP), known as the Universal Mobile Telecommunication System (UMTS), and is specified in its Release 8 and is the access part of the Evolved Packet System (EPS). Based on [7] 3GPP development targets of the LTE are defined in below:

- Low Latency
- High Throughput
- FDD and TDD in the same platform
- Simple architecture

In addition to the successful development targets of LTE, it has a well-organized protocol stack which is almost resembling those of the 5G NR protocol stack. Thus, to study the LTE technology, it is worth having a quick overview of its protocol stack organized below:

- Physical layer: It takes care of carrying information from the MAC transport channels[1] over the air interface.

[1]transport channels They determine how and with what characteristics the information is transmitted over the radio interface.

- Medium Access Control (MAC) layer: The MAC layer is responsible for mapping between logical channels[2] and transport channels, multiplexing/ demultiplexing MAC Service Data Units(SDU) from logical channels into transport Blocks (TBs). As well, it is responsible for error correction through the Hybrid Automatic Repeat Request known[3] (HARQ).

- Radio Link Control (RLC) layer: The RLC layer, specified by 3GPP, is the second radio link protocol used in UMTS, LTE, and 5G. It holds three modes of operation: Transparent Mode (TM), Unacknowledged Mode (UM), and Acknowledged Mode (AM). RLC Layer in each operation mode performs different tasks, e.g., the RLC in AM and UM mode is responsible for transferring of upper layer Packet Data Units (PDUs), Concatenation, segmentation and reassembly of RLC SDUs, reordering of RLC data PDUs, duplicate detection, etc.

- Packet Data Convergence Protocol (PDCP) layer: The PDCP Layer, specified by 3GPP, for UMTS, LTE, and 5G is located in the radio protocol stack and responsible for Header compression and decompression of IP data, Transfer of data and ciphering, and deciphering of user plane and control plane data, integrity protection and integrity verification of control plane data, etc.

- Radio Resource Control (RRC) layer: The RRC layer, specified by 3GPP for UMTS, LTE, and 5G is a layer three protocol and performs between UE and BS connection establishment. The key tasks of the RRC layer are connection establishment and release functions, broadcast of system information, point-to-point radio bearers establishment, etc.

- Non-Access Stratum (NAS) layer: NAS enables supporting the mobility of UEs and session management to establish and maintain IP connectivity.

The high-level architecture of LTE is comprised of three main components:

- User Equipment (UE): An UE is defined as an end-user same as the definition in UMTS and GSM.

- eUTRAN: eUTRAN is known as Radio Access Network (RAN) and handles the radio communications between UEs and the core network. A RAN is

[2]These channels define what type of information is transmitted over the wireless channel, either data or control messages. The MAC provides services to the RLC in the form of logical channels.

[3]Hybrid automatic repeat request (hybrid ARQ or HARQ) is a combination of high-rate forward error correction (FEC) and automatic repeat request (ARQ) error-control.

composed of one component known as Evolved NodeB (eNodeB). An eNodeB can connect to the core network and also other eNodeBs by different interfaces.

- Evolved Packet Core (EPC): The core network of the system is called EPC. It is composed of multiple elements such as Mobility Management Entity(MME), Serving Gateway (SGW), Packet-Data Network Gateway(P-GW), etc. Also, it takes care of performing switching authentication and end to end IP connections.

Although the LTE RAN has been evolved significantly with respect to the previous generations, some key problems come to light. The most crucial issue is that the hardware components are fully vendor-specific and have fixed capacity with little flexibility and dimensioning, which causes high costs for any upgrading. As it has been declared in [1], the rising sheer volume of data demand, ubiquitous coverage, and diverse requirements on various Key Performance Indicators (KPI), force Network Mobile Operators (MNOs) to boost the convenient networks.

## 2.2 Virtual Radio Access Network

Virtualization [8] is a solution that brings successful solutions to the open challenges of previous mobile generations. Virtualization mitigates the overall cost of equipment and their management by increasing the hardware utilization, decoupling functionalities from infrastructure, easier migration to newer services, and flexible management.

vRAN transforms the radio network intelligence from hardware to software and runs on COTS servers, i.e., vRAN transforms softwarized radio access points (RAP) referring to a base station, eNodeB, gNodeB, etc., into computing infrastructure in remote servers[9]. Further, vRAN brings many benefits for MNOs, indeed, MNOs can upgrade their network instantaneously with quick deployment, and at the same time, with convenient hardware maintenance operate their network more efficiently. VRAN can support and handle both LTE and 5G simultaneously and offers flexible network operations. In addition, as stated in [10] dynamic spectrum sharing can be applied to enable the coexistence between LTE and 5G in the same spectrum, which brings further operational efficiency and a smooth journey to 5G.

While the traffic pattern is inconstant and a wide range of services characterized by diverse key performance indicators (KPIs), the legacy networks use key algorithms e.g., channel estimation, error correction real-time signal, and packet processing. Using the virtualized solution, all these algorithms will be performed through the software on the COTS servers to provide the same level of service and

KPIs to the operators. Thus the virtualized solution brings flexibility and scaling, depending on the traffic pattern or traffic demands. In addition key benefits of vRAN are listed below:

- Efficient scaling and pooling network resources: Basically the network can be scaled based on the user demands, traffic patterns, and on a higher scale, at cell requirements. As well, the network can be scaled in and out depending on whether it is off-peak hours or peak time hours. Additionally, the other aspect is having pooling gains in terms of sharing the resources across the different cell sides, and implementing certain algorithms to optimize and run the network efficiently.

- Application virtualization for optimized service performance: Numerous automation and orchestration comes in to optimize the virtualized network.

- Create more flexible and cost-efficient RAN: The goal is that the vRAN will bring cost-efficiency in terms of CAPital EXpenditures (CAPEX) and Operating expenses (OPEX) saving for the operators as well as flexible architectures.

### 2.2.1 vRAN Architecture

To study the vRAN architecture, it is better apprehensible to start with the difference between traditional RAN and vRAN:

Figure 2.1 just shows the difference between traditional RAN and virtual RAN. The traditional RAN used to have all the hardware and software in the same place in the base station, and antennas used to connect via RF cabling on the top of the tower. Indeed, along with the antennas, there is a Remote Radio Unit (RRU), whose duty is RF functionalities. Further, there is another device, called the Base Band Unit (BBU), whose main duty is RRC, coding, modulation, etc. The BBU takes the signals from the RRU and then forwards them to the next place in the chain called the Central Unit (CU). The CU is connected to the multiple BBUs within a small region and concentrates them to the core network.

On the other hand, for the VRAN, still, RRU is along with the antennas and is just hardware, and the difference is that instead of BBU, there is a Distribution Unit (DU) connected to CU, ultimately is connected to the core network. In this regard, the BBU is just a fixed piece of hardware, proprietary with fixed capacity, and DU and CU run on COTS software, and then VNF is installed on that. Also, it provides the same functionalities as the BBU but using software on COTS.

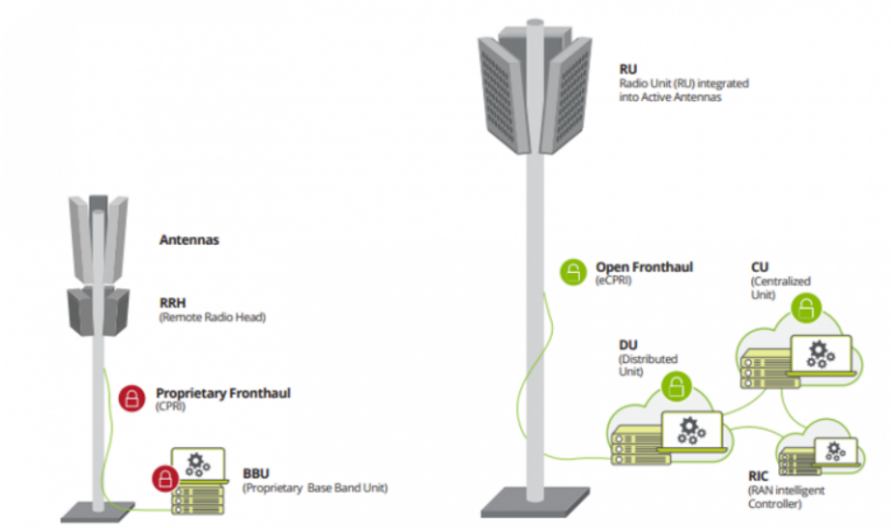



**Figure 2.1:** vRAN vs traditional RAN

While deploying vRAN brings significant benefits, makes the network more agile, and minimizes the requirement of expensive dedicated hardware, some challenges arise [10]:

- software vs hardware competencies requirements
- increase fiber density
- network architecture changes

In addition, one other unavoidable challenge is an unprecedented increase in the number of connected devices which try to concurrently access the vRAN, as well, several applications compete for resources. These issues deteriorate the efficiency of radio functions [10]. In this regard, efficient radio resource management (RRM) is required to handle these demands, especially in heterogeneous networks. Efficient RRM can effectively support the diverse demands and rapidly varying network and channel conditions.

## 2.3 Radio Resource Management

RRM and radio resources are challenging issues that by shifting the 5G paradigm they must be taken into account more than previous mobile generations. RRM takes care of assigning radio and network resources to be used for wireless communications. Roughly speaking, resource allocations must be designed in such a

way to optimize the amount of successfully transmitted information to users in a network [11]. In addition, 5G integrates many advanced technology components e.g. massive MIMO, MM-wave communication, network slicing, vehicular networks, broad frequency bands, etc., which makes it highly complex than in previous mobile generations. Thus the 5G requires mechanisms for RRM to satisfy optimization domains and meet latency demands since efficient RRM with traditional rule-based algorithms is particularly challenging.

Cellular networks both 5G and 4G exploit Adaptive Modulation and Coding (AMC). This mechanism refers to the selection of the appropriate modulation and coding scheme (MCS) based on the channel quality experienced by each UE. The AMC aims to keep the block error rate (BLER) below a predefined threshold. As stated in [12] the BLER target in 4G is fixed at 10 %, however, 5G systems will cover a wider spectrum of services, requiring potentially different BLER targets [13]. In other words, by selecting different combinations of modulation types and code rates, spectrum efficiency will be improved, and it is possible to scale in and out the radio resources based on different channel conditions.

The AMC is a successful solution introduced to deal with the time-varying nature of the wireless channel under mobility. To reach this goal, periodically, each UE measures the channel quality and maps this information into a 4bit index called Channel Quality Indicator (CQI), and transmits this information to the base station (BS). The BS based on the CQI reported by each UE defines the best MCS correspondingly. Typically, each CQI is associated with a given signal-to-noise ratio (SNR). Then the BS sends MCS values to UEs, using Downlink Control Information (DCI) embedded into the physical downlink control channel (PDCCH). MCS values are denoted by an integer number $\in \{1,2,..28\}$ and each value is characterized by different modulation order and transport block size. The mapping between MCS index modulation order is described in table 2.1.

| MCS index | Modulation scheme |
|---|---|
| $I_{MCS} \in \{1,2,..,9\}$ | QPSK |
| $I_{MCS} \in \{10,11,..,16\}$ | 16-QAM |
| $I_{MCS} \in \{17,18,..,28\}$ | 64-QAM |

**Table 2.1:** Mapping between MCS indices and modulation order [14]

# Chapter 3

# Network Simulator 3

## 3.1 Ns-3 Platform

Network Simulator 3 [15], known as ns-3 is a discrete-event network simulator. It has been used to simulate and implement different network technologies for research and educational purposes. ns-3 is an open-source simulator written mainly in C++ and Python, provided with high modular implementation. ns-3 with comprehensive tools paves the way for researchers, students, and developers to simulate most of the telecommunication and network parts. The ns-3 supports researches on both IP and non-IP-based networks, involving models for wifi, WiMAX, LTE, and 5G NR. ns-3 has some great features that are superior to other simulators and are listed below:

- It has been developed to support virtual networks' elements such as nodes, channels, applications, and also event schedulers, topology generators, timers, random variables, and other objects. These components enable ns-3 to support discrete-event network simulation of both internet-based and non-Internet-based network systems.
- The useful feature that ns-3 supports is the ability to trace, log and compute statistics on the simulation output.
- ns-3 has an integrated Gnuplot library for visualization and provides PCAP output that enables analyzes with other tools like Wireshark. It also has excellent support for animation of network simulation, such as NetAnim.

There are four abstracts terms that have specific meaning for ns-3 simulator:

**Node**

On the internet, each computing device that can connect to the internet is called a host or end-user. But in ns-3, as a network simulator, a node is used for a more generic term. The corresponding class: Node includes methods for handling these computing devices in simulations.

### Application

Software typically exploits various computer resources and a user runs an application that uses the resources controlled by the software, while an operating system is used to enable this collaboration. In ns-3 there is no definition of an operating system, despite that, the application concept has been provided by class: Applications which is run on nodes to drive simulations, i.e., the Application generated some activity similar to user programs. For an instance, UDPEchoClientApplication and UdpEchoServerApplication in the form of client and server, serve as an Application on top of UDP on ns-3 Node. Besides, in this class, users can develop applications by extending the existing Application class using object-oriented programming.

### Channel

Usually, channels are media used to connect a computer to the network to transmit data flows. In ns-3 the class Channel provides an abstraction of media that connects a Node to a communication sub-network. various specialized Channels have been provided in ns-3 such as CsmaChannel, PointToPointChannel and WifiChannel.

### Net Device

To connect computers to the network, two main components are required, a network cable and a hardware device, knowing as a peripheral card in the case that owned Network Interface called NIC. The latter must be installed on the computer using software drivers so-called device drivers. In other words, devices and network devices are handled by a device driver and network device driver respectively. In ns-3, both elements are provided in a single entity and called Net Device and represented in class NetDevices. Connections to Nodes and Channels are managed via NetDevices class. Moreover, specialized versions of NetDevices have been provided by the ns-3, such as CsmaNetDevice, PointToPointNetDevice, LteEnbNetDevice and LteUeNetDevice.

### Topology Helper

The Last abstract component is Topology Helper that handles and manages the connections of different objects. Indeed, Topology Helper classes connect

NetDevices, Nodes, and Channels, also is responsible to assign IP addresses, etc.

## 3.2 LENA Module

The module that has been used in this thesis is the LENA module developed by [16]. The proposed LTE module supports the essential features of LTE. In this module, basic implementation of LTE devices has been provided which is composed of MAC layer, Physical layers, and also, various propagation models. Moreover, QoS management, scheduling algorithms in the MAC layer, and frequency reuse techniques have been provided. The LTE module has been composed of two main parts:

**The LTE model**

It provides the LTE radio protocol stack, including RRC, PDCP, RLC, MAC, and PHY. The UE and eNodeB hold all These entities. As the LTE model plays the role of Radio Access Point (RAP), it supports evaluation of RRM, QoS aware packet scheduling, inter-cell interference coordination, and dynamic spectrum analysis of LTE systems. As in real LTE networks, scheduling and RRM do not work with IP packets, rather work with RLC PDUs. Moreover, the HARQ scheme, PDCP layer, RLC layer, and RRC layer functionalities have been implemented the same as the real LTE network. Moreover, the simulations can be scaled up to a large number of eNodeBs and UEs and can be configured with different cells with different carrier frequencies and bandwidths.

**The EPC model**

The LTE core network interfaces, protocols, and entities have been provided in this model. SGW, PGW, and MME nodes include all these entities and protocols, also the eNodeB has a portion of them. The EPC model provides end-to-end IP connectivity over the LTE model between the UE and a remote host. It also enables the connections of the UEs to the internet, through the RAN of the eNodeBs that are connected to the SGW and PGW. Any ns-3 application that works on top of TCP or UDP can be used within EPC to model realistic applications.

Figure 3.1 depicts a schematic of the LTE-EPC simulation model. The model represents the entities and network interfaces in the EPC model and the LTE model in ns-3.

In the following sections, the functionality and specifications of different modules

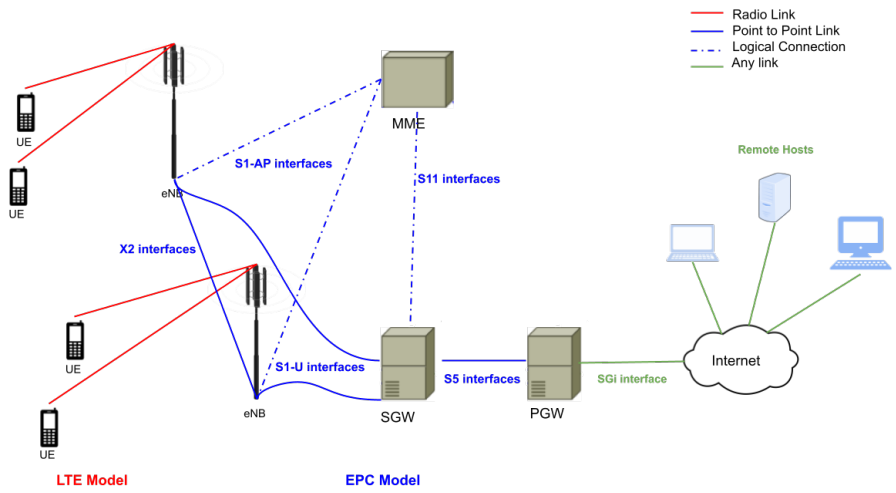



**Figure 3.1:** Overview of the LTE-EPC simulation model

in the downlink are described because, in this thesis, the main goal is to evaluate the performance of downlink transmissions.

### 3.2.1 MAC Entity

As the Functionality of the MAC layer is different in UEs and eNodeBs, then the Mac entity is implemented in two different classes: LteUeMac and LteEnbMac. The Mac entity provides an interface between the upper layers and the Physical layer. The main task of the Mac entity is the creation of the transport blocks composing fragments provided by RLC. In addition, The Adaptive Modulation and Coding module resides in the Mac entity in LteEnbMac class. Also the latter is responsible for radio resource allocation, which means that the uplink and the downlink packet scheduler reside in this class. Whereas each UE must prepare its CQI, thus, LteUeMac class is enabled to create CQI feedback.

**The Adaptive Modulation and Coding**

The Adaptive Modulation and Coding (AMC) module enables the functionalities of the AMC defined in the LTE standard [12]. Indeed, each UE periodically sends CQI based on the channel quality measurement and sends it to eNodeB. Then AMC module uses the CQI to find the most suited MCS for the UE (described in section 2.3).

In the ns-3, the AMC module resides in LteEnbMac class and provides an unique

mapping among cqi and the MCS, the mapping function used in the ns-3 follows the table 3.1. Generally, two types of AMCs have been provided in the ns-3. One is based on the model described in [14]: LteAmc :: PiroEW2010 and the other which is modified version of [14] is: LteAmc :: MiErrorModel .
In the LteAmc :: PiroEW2010 model, spectral efficiency is required and is calculated by expression 3.1c. Then the value of spectral efficiency is discretized by the received CQI from the UE. Afterward, the floor of the value is calculated and is mapped to the corresponding MCS index. The expression 3.1c represents the calculation of spectral efficiency $\eta_i$ of a UE, where $i$ denotes a generic UE, and $\gamma_i$ is its SINR.

$$BER = 0.00005 \tag{3.1a}$$

$$\Gamma = \frac{-ln(5 \times BER)}{1.5} \tag{3.1b}$$

$$\eta_i = \log_2(1 + \frac{\gamma_i}{\Gamma}) \tag{3.1c}$$

In the LteAmc :: MiErrorModel model, MCS index is selected for the actual physical layer performance based on the reported CQI. In this model, when Physical Downlink Shared Chanel (PDSCH) TB with the MCS and code rate correspondent to a specific CQI index can be received and meet the error probability threshold, i.e., error probability threshold is 0.1, then the CQI is assigned.

**CQI feedback**

In the ns-3, Wideband CQI and inband CQI are periodical measurements i.e., every Transmission Time Interval (TTI) are generated. Based on the standard, wideband CQI is a single value of channel state which represents the status of all RBs in use. On the other hand, inband CQI is a set of values, each of which represents the channel state for each RB.

Generally, the eNodeB sends reference symbols over the entire downlink bandwidth of each TTI. To generate the CQI feedback, each UE uses data collected from the reference symbols and exploits them to measure SINR, then the measured SINR will be mapped to a CQI. Then each UE will report the CQI to the target eNodeB. However, in the ns-3, SINR calculation in downlink is done according to two different methods:

- CtrlMethod :: Signal power from the reference signals in the ns-3 simulation is equivalent to the PDCCH. This signal power is combined with the interference

| Interval for $\eta_{i,j}$ | CQI Index | Modulation Scheme |
|---|---|---|
| $\leq 0.15$ | 1 | 4-QAM |
| $0.15 \div 0.23$ | 2 | 4-QAM |
| $0.23 \div 0.38$ | 3 | 4-QAM |
| $0.38 \div 0.60$ | 4 | 4-QAM |
| $0.60 \div 0.88$ | 5 | 4-QAM |
| $0.88 \div 1.18$ | 6 | 4-QAM |
| $1.18 \div 1.49$ | 7 | 16-QAM |
| $1.49 \div 1.91$ | 8 | 16-QAM |
| $1.91 \div 2.40$ | 9 | 16-QAM |
| $2.40 \div 2.73$ | 10 | 64-QAM |
| $2.73 \div 3.32$ | 11 | 64-QAM |
| $3.32 \div 3.90$ | 12 | 64-QAM |
| $3.90 \div 4.52$ | 13 | 64-QAM |
| $4.52 \div 5.12$ | 14 | 64-QAM |
| $\geq 5.12$ | 15 | 64-QAM |

**Table 3.1:** Mapping function: from the spectral efficiency to the modulation orders [14]

power from the PDCCH to calculate the SINR. Indeed, in this method, the goal is to consider any neighboring eNodeB as interference, regardless of whether the eNodeB is using any PDSCH or the power and RBs used in PDSCH. In this thesis, this model has been exploited because only one cell and one eNodeB have been considered.

- MixedMethod :: Concerning the CtrlMethod : method, signal power from the reference signal is combined with the interference power from the PDSCH. In this method, interference only is considered from the eNodeBs that are actively sending and receiving data on PDSCH. So, this method allows the generation of inband CQIs because the different amounts of interference can be calculated on different RBs.

## 3.3 Ns-3-ai Platform

Artificial intelligence algorithms are playing a critical role in wireless radio technology paradigms for the next generation networks. Next generation networks

are supposed to support high data rates, low latency, and new applications requiring intelligent adaptive learning and decision making. In recent research[17], [18] summarized the essential accomplishments that Artificial intelligence techniques have achieved in telecommunication in comparison with classical approaches.

An AI framework is a tool that helps to develop algorithms and train machine learning models. The AI frameworks facilitate the complexity of developing machine learning algorithms by providing existing models, as well, Python is the main programming language used by mainstream frameworks. Both TensorFlow [19] and PyTorch [20] are two popular AI frameworks that provide a comprehensive API among all deep learning frameworks.

ns-3-ai [21] provides an interface between ns-3 and several Python-based AI frameworks like TensorFlow and PyTorch to develop Machine Learning and Deep Learning algorithms. ns-3-ai uses shared memory mechanisms for transmitting data, which outperforms the other methods by reducing transmission time and supporting the exchange of huge data [21]. The interaction between ns-3 and AI frameworks must be done by transferring data in multiple processes since the global variable of any process cannot be accessed by other processes. Thus shared memory as a practical solution has been adjusted to communicate between processes. The ns-3-ai module composed of two main components:

- The ns-3 interface developed by C++ as core module: It supports transferring data from C++ program to Python programs.
- The AI interface Developed by Python: It provides fast development of Deep Learning and Reinforcement Learning algorithms.

### 3.3.1 Ns-3-ai System Architecture

Figure 3.2 illustrates the architecture of the ns-3-ai platform and its collaboration with the ns-3 simulator. Based on this architecture it can be perceived that the ns-3 simulator and AI frameworks run in separate processes, therefore, data transmission is done through shared memory and in two directions:

- The ns-3 simulator -> the AI frameworks: A network topology is set up within the ns-3 simulator to generate data which is required to be sent to the AI algorithm to train.
- The AI frameworks -> the ns-3 simulator: An AI algorithm is provided using the AI frameworks to train the model using the data sent by the ns-3

simulator. Afterward, the data from the trained model have to be sent to the ns-3 simulation for testing the data.

Thus, the ns-3-ai module must provide interconnection for the ns-3 and the AI frameworks, and it is done by using shared memory and transferring data through this memory pool. This shared memory is accessible by AI frameworks and ns-3. the latter has the most control over the shared memory pool. The high-level

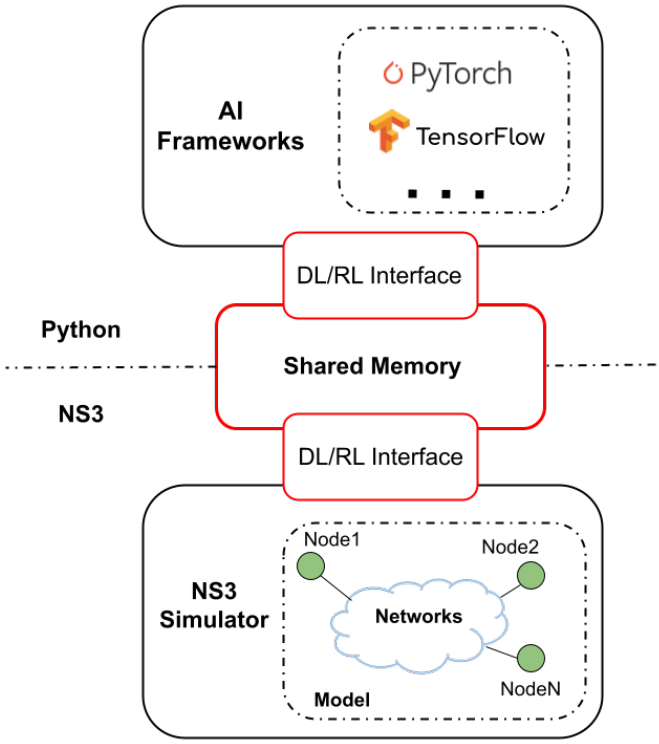



**Figure 3.2:** System architecture [21]

interface of ns-3-ai supports various dimensions and types of data, also self-defining data types such as structures to be transmitted in both sides, the ns-3 simulation i.e., c++ side, and AI frameworks i.e., Python side. Further, for some algorithms e.g., Reinforcement Learning(RL) and Deep Learning (DL) sophisticated data structures and interfaces have been developed.

The DL module in ns-3-ai is developed by exploiting PyTorch and Tensorflow frameworks. On the Python side, well-defined data structures are used which carry information such as features of data-set, target, and predicted values, further, on the C++ side, several functions are defined to extract features of data sets and feedback prediction results, etc.

The RL module as well is developed by exploiting PyTorch and Tensorflow frameworks. In this module, on the Python side, more or less two data structures are defined, one of them enables receiving data from the ns-3 simulation, i.e., environment states, and the other enables sending the data to the ns-3 simulation, i.e., actions. On the other hand, on the C++ side, two functions are defined: set and get, which dealing with modifying the environment based on the received actions and delivering the environment state to the ns-3-ai side. In this research, the RL module is exploited which is described in detail in section 5.

# Chapter 4

# Machine Learning Background

## 4.1 Machine Learning Algorithm

Machine Learning (ML) is one of the most substantial artificial intelligence branches that in recent years has been successfully applied in many fields. Machine Learning generally can be defined as automatic computational procedures which can use series of past experiences to learn a task, to improve the performance of a system, or to make precise models to infer or forecast information [22]. So, the ML can address studying and flourishing computational models. Indeed, by providing context-related data from an environment, we can train a model to forecast, predict the future or make a decision, while the model does not know all knowledge about the environment. In other words, a model can be adapted to statistical learning to learn about the environment and tries to find the best decisions, values, or actions to reach a particular goal. Continuously train the model to get improved by reducing potential errors using different optimization models[23]. When it comes to talking about the problems that can be tackled using ML, it is widespread and popular in different areas, lately, it also has played a significant role in telecommunication fields and has provided high potential innovations to the Telecom value chain.

Besides the most significant and brilliant ML paradigm, one most important fact is studying and mastering different ML algorithm categories and trying to find the best fit for a particular problem. The following sections describe an overview of each ML category.

### 4.1.1 Supervised Learning

Supervised Learning is one of the widely used ML algorithms, which its most common application is in classification and regression. The supervised learning algorithm aims at creating and learning models, that find matches of labels with the so-called training data set. Then, training data set is used to train a model. Indeed, the training data set includes labeled input data that pair with desired outputs and are used to find a map between the input data and the output data. Afterward, the obtained model is exploited to make a prediction based on evidence, for a new set of data, in the presence of uncertainty. To name examples of Supervised learning, we can name their application in predictive maintenance problems [24], financial applications[25] and also biological applications like cancer detection [26].

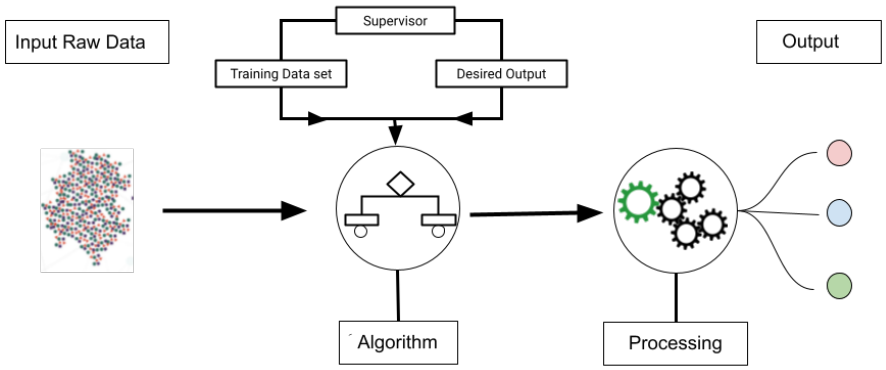



**Figure 4.1:** General schematic of supervised learning algorithm

### 4.1.2 Unsupervised Learning

On the contrary, unsupervised learning does not deal with labels, and there is no need to supervise the model. This category usually addresses finding all kinds of unknown patterns in data or groupings in data. clustering such as K-means clustering, anomaly detection, and pattern recognition are the most well-known applications of unsupervised learning. In clustering, the main goal is to specify different categories in a set of data that does not specify by any label. Rather it can be done by defining the similarity measures for the clusters. There are cases that we have to handle enormous data sets that hardly can be labeled or we have no prior knowledge about how many or what classes that the data set is divided into. Therefore, we need clustering to gain some insight into the structure of the data, this is where Unsupervised Learning comes in. However, one of the important

issues that can be mentioned related to unsupervised learning is that it is hard to get accurate information regarding sorting data.

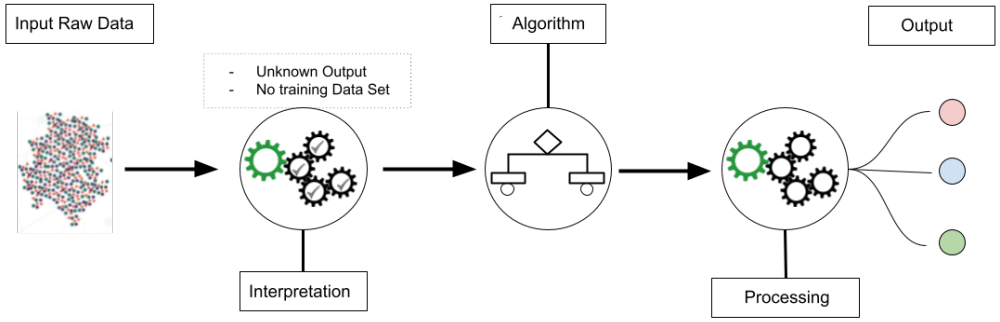



**Figure 4.2:** General schematic of unsupervised learning algorithm

### 4.1.3 Reinforcement Learning

Reinforcement Learning (RL) is one of three machine learning paradigms, alongside supervised learning and unsupervised learning. The goal in using reinforcement learning is to enable an agent (learner) to learn a policy in an environment by trial and error. The agent uses feedback from its actions to maximize the notion of cumulative reward [27].
Reinforcement learning will be discussed further and in more detail in section 4.3.


**Figure 4.3:** General schematic of reinforcement learning algorithm

## 4.2 Deep Learning

Deep Learning (DL), also known as the deep neural network is an artificial intelligence function and part of the ML family. DL is based on Artificial Neural Networks (ANN) with representation and feature learning. DL seeks to mimic the

human brain in processing data, by exploiting the hierarchical structure of neural networks, which allows it to take non-linear approaches, processing data across a series of layers.

ANN are nonlinear computational models composed of interconnected processing units, known as, artificial neurons. The artificial neurons are organized into three layers: input, hidden, and output layers.

- Input layer: This layer is made of $n_1$ neurons, which corresponds to network inputs.
- Hidden layer: This layer is composed of one or multiple hidden layers and each can be of a dimension of $n_2$ neurons.
- Output layer: It includes $n_3$ neurons corresponding to each network output.

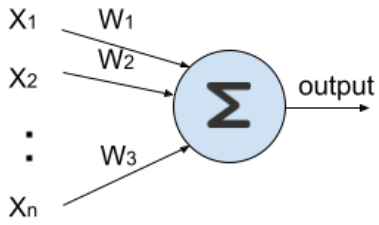



**Figure 4.4:** The structure of an artificial neuron

As it is represented in figure 4.4 each neuron receives n inputs signal $x_i$ from multiple neurons, then weights them separately,i.e., connection weights of $w_i$, then, the neuron sums them up and passes the sum through a function to produce the output.

The simplest neural network unit, the so-called Perceptron was introduced by Frank Rosenblatt in 1957, mostly aimed at supervised learning of binary classifiers. This algorithm enables neurons to learn and process in the training set, one at a time. The expression 4.1 describes the learning rule of perceptron algorithm through this learning rule, a binary decision is made, i.e., if the sum exceeds a threshold, the neurons will fire, otherwise not. The output can be either 1 and 0 or 1 and -1, depending on the activation function.

$$Y_i = \begin{cases} 0 & \text{if } \sum_{i=1}^{n} X_i Wi < threshold \\ 1 & \text{if } \sum_{i=1}^{n} X_i Wi > threshold \end{cases} \tag{4.1}$$

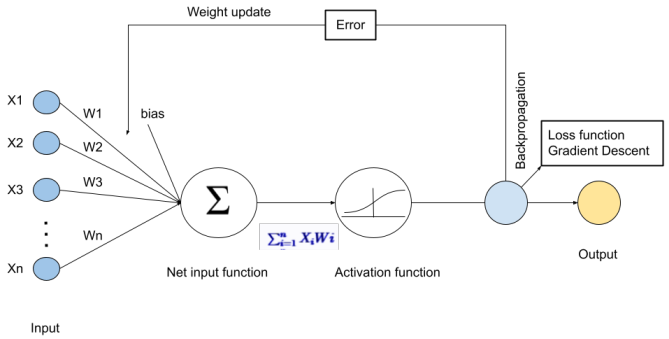
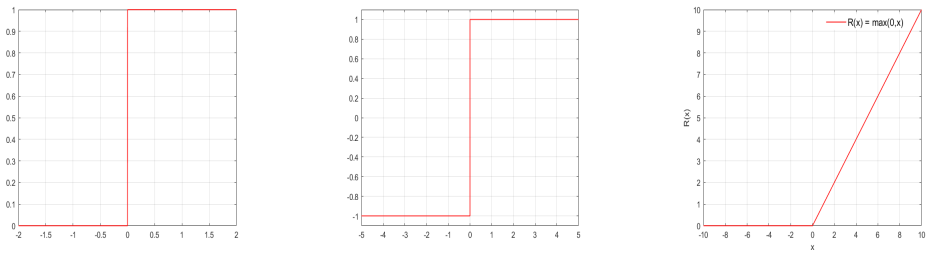



**Figure 4.5:** Neural network architecture

**Activation function** Generally, an activation function in a neural network defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network. The activation function of Perceptron applies step function which is represented in figure 4.5. It converts numerical inputs to +1 and 0, i.e., the Step function gets triggered above a certain value of the neuron output or it is deactivated, i.e. its output is not considered for the next hidden layer. On the other hand, the Sign function outputs +1 or -1 depending on whether the neuron's output is greater than zero or not. A bias is an extra scalar that is added to the neuron to shift the value of the output.




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

**Figure 4.6:** Activation function,(a) step function (b) sign function (c) Relu function

The step function is not useful when there are multiple classes. That is one of the limitations of the binary step function. In addition, the gradient of the step function is zero which causes a restraint in the backpropagation process. Indeed,

when the gradient of the function is zero, the weights and biases don't update. In addition to the step function, there are more sophisticated activation functions whose output is not binary values like sigmoid, Relu, and tanh, etc.

For example rectified linear activation function or ReLU depicted in 4.6c, is the most used activation function in the world right now. Since it is used in almost all convolutional neural networks or DL. As it is depicted in 4.6c, the ReLU is half rectified (from bottom). f(x) is zero when x is less than zero and f(x) is equal to x when x is above or equal to zero.

**Gradient descent**

Gradient descent algorithms are commonly used to train ML models and neural networks. Gradient denotes the slop of a function, and simply measures changes in all weights that address the changes in error. The higher the gradient, the steeper the slope, and the faster a model can learn. But if the slope is zero, the model stops learning. from the mathematical point of view, the gradient is a vector of partial derivatives of specific input values with respect to a target function. The goal of gradient descent is to minimize the cost function or the error between predicted and actual value. To achieve this goal, at first, the gradient, the first-order derivative of the function at that point must be computed, and after that one step is made in the opposite direction of the gradient, by learning rate, then it is multiplied the gradient at that point.

In order to do this, it requires two data points—a direction and a learning rate. These factors determine the partial derivative calculations of future iterations, determines that they gradually arrive at the whether local or global minimum i.e. point of convergence. More detail on these components can be found below:

**Learning rate** also referred to as step size or the alpha. Learning rate denotes the size of the steps that are taken to reach the optimum value. This is typically a small value, and it is evaluated and updated based on the behavior of the cost function. Applying large values of learning rates leads to larger steps which increase the risk of overshooting the minimum. Conversely, choosing small values of learning rate leads to small step sizes. While it has the advantage of more precision, the number of iterations compromises overall efficiency as this takes more time and computations to reach the minimum.

**Loss function** or cost function measures the error in terms of difference between actual y and predicted y at its current position. Indeed, loss function provides feedback to the model, by adjusting the parameters to minimize the error

and find the local or global minimum. It continuously iterates, moves along the direction of steepest descent (or the negative gradient) until the cost function minimizes or gets almost close to zero. At this point, the model will stop learning. It is worth mention that, however the terms, cost function, and loss function, are used exchangeably, there is a slight difference between them. The loss function refers to the error of one training example, while a cost function calculates the average error across an entire training set.

Additionally, there are three popular types of gradient descent that mainly differ in the amount of data they use:

- Batch gradient descent, also known as vanilla gradient descent, calculates the error for each example within the training dataset. In this algorithm, the model gets updated just after all training examples are evaluated. Although batch gradient descent has computational efficiency and produces a stable error gradient and stable convergence, it can achieve sub-optimal solutions and also requires the entire training dataset for each training cycle.

- Stochastic Gradient Descent (SGD) updates the parameters for each training example one by one. This behavior makes SGD faster than batch gradient descent, also provides a detailed rate of improvement. On the other hand, frequent updates lead to high computationally expensive than the batch gradient descent approach. Additionally, the frequency of those updates can result in noisy gradients.

- Mini-batch gradient descent is a combination of the concepts of SGD and batch gradient descent. In this method, the training dataset is split into small groups called mini-batches, and then an update is performed for each of those batches. This behavior brings robustness and efficiency to the learning process. Common mini-batch sizes range between 50 and 256, and it is the most common type of gradient descent within deep learning.

**Backpropagation**

Backpropagation or simply "backprop," is an algorithm for calculating the gradient of a loss function with respect to variables of a model. Backpropagation is used to train neural network models to calculate the gradient for each weight in the network model. The gradient is then used by an optimization algorithm to update the model weights.

The algorithm was developed explicitly for calculating the gradients of variables in graph structures working backward from the output of the graph toward the

input of the graph, propagating the error in the predicted output that is used to calculate the gradient for each variable.

## 4.3 Reinforcement Learning

As described in section 4.1.3, RL is one of the ML paradigms, whose underlying idea is similar to human life, i.e., human actions turn out to be good or bad actions, which reinforces or punishes the human's behavior which caused taking an action. So, the so-called software agents try to make the best actions based on a policy, to maximize the so-called cumulative reward. The agent is in interaction with the environment, and by exploring the environment tries to discover the best policy. Figure 4.7 depicts a generic schematic of RL Algorithm.

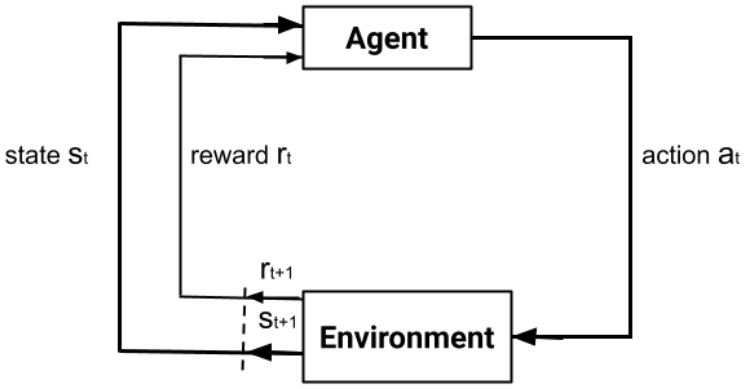



**Figure 4.7:** The generic schematic of RL algorithm

One of the most simple and basic examples of RL applications is cartpole game. The cartpole is a pendulum that contains a wight above its pivot. It is unstable but it can be controlled. In the cartpole game the goal is to keep balance the pole as long as it is possible. Figure 4.8 represents the cartpole general schematic.

The position of the cartpole is known as state and the game starts from the pole in upright condition, i.e., the current position of the cartpole. The state comprising the angle of pole, velocity, and position of the card, and depending on the action the agent takes it leads to different other states. The cartpole is controlled by inserting a force of +1 or -1 to the card to keep the pole upright, i.e., the agent moves the card to right or left to prevent the pole from falling over. For every time step, the agent receives a reward for its action, a reward of +1denotes that in that time step the pole has remained balance. In addition, the vertical

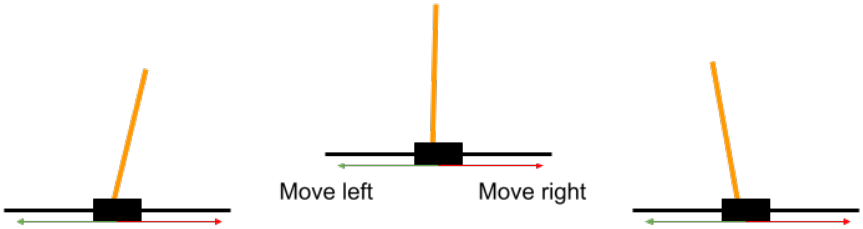



**Figure 4.8:** The Cartpole schematic

degree of the pole determines the end of each episode, e.g., if the pole is more than 15 degrees from vertical, the episode ends. The game is repeated in various episodes, in which agents can learn more about the dynamics of the environment.

By this explanation, the difference between reinforcement learning and supervised learning and unsupervised learning will be clear. Indeed, in supervised learning, the algorithm uses a training data set whose correct answer keys are used to train the model. Whereas in Reinforcement learning, there is no training data set, the agent must explore the environment to find the best action in each state of the environment.

Moreover, the difference between unsupervised and RL algorithms can be described in terms of their goals. Unsupervised learning's goal is to learn similarities and differences between provided samples. RL's goal is to find an optimal policy to be able to make the best action for each particular state to maximize the feedback value receives from the environment.

### 4.3.1 Main Elements of RL

Besides an agent and an environment, as two main elements of RL, it has other elements as well: a policy, a reward, a value function, and a model of the environment. The latter is optional in some RL approaches [28]. Moreover, states and actions are integral parts of each reinforcement learning algorithm that must be specified properly [29].

State: A state is represented with $s_i \in S$, denotes the current situation of the environment. The state space has N dimension, $S = \{s_1, s_2, s_3, ..., s_N\}$.

Action: An action $a_j \in A$, denotes what the agent can do in any state, and the

action space with M dimension can be described as $A = \{a_1, a_2, a_3, ..., a_M\}$.

- Policy: A policy $\pi(.)$ determines a behavior that the agent has at the given time, i.e., the policy makes a map between the current state at time t $s_t \in S$ into action $a \in A$. The policy is an essential part of the agent and in some cases, it is a simple lookup table so-called Q_table, and in some cases, it has immense computation time.
  In general, the policy can be categorized into two main groups of function: deterministic, and stochastic. A deterministic policy can be defined as a function of the form $\pi_d() : S \rightarrow A$, in which for every state $s \in S$ there is $a \in A$ clear defined action that will be taken. On the other hand, a stochastic policy is function of the form $\pi_s(a|s) \in [0,1]$. It means that for every state $s \in S$ there is no clear defined action to take but there is a probability distribution for the action $a \in A$ to take from that state $s \in S$.

- Reward: A reward signal describes environment feedback according to the action $a \in A$ that the agent made at state $s \in S$,indeed, at each time step, the agent observes an state $s_i \in S$, performs an action $a_i \in A$ and receives immediate reward from the environment and transit to the next state $s_{i+1} \in S$. So, the reward signal defines the good and bad actions for the agent. While the agent's main goal is to maximize the cumulative reward over the long run, it uses the reward to alter the policy if the corresponding received reward is low, in order to change the action in that situation in the future.

- Value Function: A value function concerns how good is the action in long run, whereas, as discussed previously, a reward signal is an immediate response of the environment. The notion of "how good" here is defined in terms of future rewards that can be expected. The expected future reward also is known as expected return from time t, can be described as cumulative discounted rewards, following the expression 4.2:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3}, ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{4.2}$$

  where $\gamma \in [0,1]$ is discounted factor and the idea behind using the discount factor is not only penalizing future reward when we have infinite R but also, discounting with a value $\in [0,1]$ to avoid divergence of expected reward.
  Therefore, we can define the state value as the expected value of the discounted return if we are in the state $s \in S$ at time t:

$$V_\pi() = \mathbb{E}_\pi[R_t|S_t = s] \tag{4.3}$$

  In addition to value function we need to define state-action values, also known as Q values, $Q_\pi(s, a)$ denotes the discounted return $R_t$ that the agent should

have expected, starting from the state $s \in S$, getting the action $a \in A$ under policy $\pi$:

$$Q_\pi(s, a) = \mathbb{E}_\pi[R_t|S_t = s, A_t = a] \tag{4.4}$$

Approximation of the value function is done repeatedly to get closer to the current policy, in the meantime, the policy improves since, the optimal value function produces the maximum expected return, by reproducing the value function based on the target policy:

$$V_\pi() = \sum_{a \in A} Q_\pi(s, a)\pi(a|s) \tag{4.5}$$

where $\pi$ is target policy. So, optimal policy achieves the optimal value function $V_*()$:

$$V_*() = \max_\pi V_\pi() \tag{4.6}$$

$$Q_*(s, a) = \max_\pi Q_\pi(s, a) \tag{4.7}$$

- Model: The final element of RL is the model of the environment, generally, the model seeks to imitate the behavior of the environment. As mentioned this is an optional element of RL algorithms and is exploited by the so-called model-based approaches.
  On the other hand, in other RL algorithm categories, model-free approaches only exploit the three first elements. Both approaches are described in the following sections.

### 4.3.2 Model_Based Approaches

Model-based RL is defined as any Markov Decision Process (MDP) which exploits a model and then learns to approximate a policy function. In other words, model-based algorithms can be considered as Dynamic Programming (DP) which is categorized into two approaches, Policy Iteration and Value Iteration. These approaches use the model's predictions or distributions of the next state and reward in order to calculate optimal actions. Specifically in DP, the model must provide state transition probabilities, and expected reward from any state, action pair which turns to high complexity. In addition, a model-based algorithm is an algorithm that uses a transition function and a reward function in order to estimate the optimal policy.

### 4.3.3 Model_Free Approaches

In model_free approaches, there is no dependency on the environment model during the learning process. Indeed, the transition probability distribution which is main element in model-based RL, and also the reward function which is associated with the MDP are not used. The model_free approaches rely on stored values of state action pairs, which are maximum returns that the agent can expect for each action that has been taken from each state over many trials, in other words, model_free approaches are based on trial and error mechanisms.

There are two major model_free approaches namely: Monte Carlo (MC) and Temporal Difference (TD). MC approach learns directly from complete episodes with no bootstrapping, i.e., MC is based on averaging complete return. In other words, we want to estimate $v_\pi(s)$ in an episode that is computed under policy $\pi$ passing through state s, this can be done either by averaging the returns following the first occurrence of s, or averaging the returns following all occurrence of s. One drawback of MC is that it can only be applied to episodic MDP where all episodes must terminate.

Further, Temporal-Difference (TD) methods can be considered as a combination of MC and DP approaches, since similar to MC, TD uses experiences to solve the prediction problem and also, similar to DP, uses some variation of generalized policy iteration. TD(0), Q Learning, and SARSA are the simplest and most used algorithms of TD learning algorithms that are explained in the following sections.

### 4.3.4 TD(0)

The TD uses experience to solve the prediction problem, same as MC method, i.e., Value function $V$ is updated for the states $s_t$ using experience following a policy $\pi$. However, in contrast to MC, $TD(0)$ at next time step,$t+1$ immediately forms a target and makes update using the observed reward $R_{t+1}$ and estimates the $v(S_{t+1})$, rather than waiting for entire episode.

$$V(S_{t+1}) = V(S_t) + \alpha[V_{target}(S_{t+1}) - V(S_t)] \tag{4.8}$$

$$V_{target}(S_{t+1}) = R_{t+1} + \gamma V(S_{t+1}) \tag{4.9}$$

The equation 4.9, is an estimate of the true value of $V(S_t)$ , and also called the TD target. Indeed it is a bootstrapping method, since a Q value is used to update another Q value. The equivalent update can be described in 4.10:

$$V(S_{t+1}) = V(S_t) + \alpha[G_t - V(S_t)] \tag{4.10}$$

Where the $G_t$ is the total discounted return at time t, assuming a start from state $s$, then taking action $a$, using the current policy until the episode ends.
In addition, equation 4.11 represents the error in TD(0), which measures the difference between the estimated value and the target value. Further, the error in TD(0) takes care of the error in the estimation made at time $t + 1$, since the error in TD(0) depends on the next state and reward.

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \tag{4.11}$$

### 4.3.5 SARSA

State-Action-reward-State-Action for short SARSA, is a well-known on-policy method i.e., on-policy methods have the same target policy and behavior policy. In SARSA algorithm rather than defining and exploiting state-value functions, the action-value function is defined. Typically in SARSA algorithm, TD method is exploited as described in 4.8, $q_\pi(s, a)$ for the behavior policy $\pi$ and all states $s$ and actions $a$ must be estimated.

$$Q(S_i, a_i) = Q(S_i, a_i) + \alpha[R_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(S_t, a_t)] \tag{4.12}$$

The expression 4.12 represents the update procedure in SARSA algorithm. The update is based on the action that is taken i.e., on-policy method, and is done after every transition from state $s_t$ . In this method, continuously both the $q_\pi$ is estimated for the behavior policy $\pi$ and $\pi$ is changed toward the optimal policy with respect to $q_\pi$.

### 4.3.6 Q Learning

Q-Learning algorithm is one of the most popular off-policy TD algorithms, i.e., off-policy algorithms have different target policies than behavior policy. Unlike the SARSA algorithm, the Q-Learning, independent of the policy, approximates the optimal action-value function. In particular, Q-Learning uses the update equation 4.13 derived from 4.11:

$$Q(s_i, a_i) = Q(s_i, a_i) + \alpha[R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{4.13}$$

In Q-Learning algorithm, a Q table is defined that contains the values of action-values. If the state space or action space is large, it is difficult to manage the table and the learning process becomes inefficient, as a result, Q-Learning cannot

learn the optimal policy. In this regard, Deep Q-Learning (DQL) is introduced to overcome the problems of Q-Learning.

## Deep Q Learning

The main difference between DQL and Q-Learning is the implementation of the Q table. Indeed, DQL exploits NN knowing as Deep Q Network (DQN) rather than Q table. So, the DQL maps input states to action Q-value pairs instead of mapping state-action pairs to Q-values. Also, figure 4.9 shows the architecture of DQL, in which the DQN is replaced with a Q table and approximates the Q-values.
If using a nonlinear function approximator in RL algorithms, as noted in [30], may

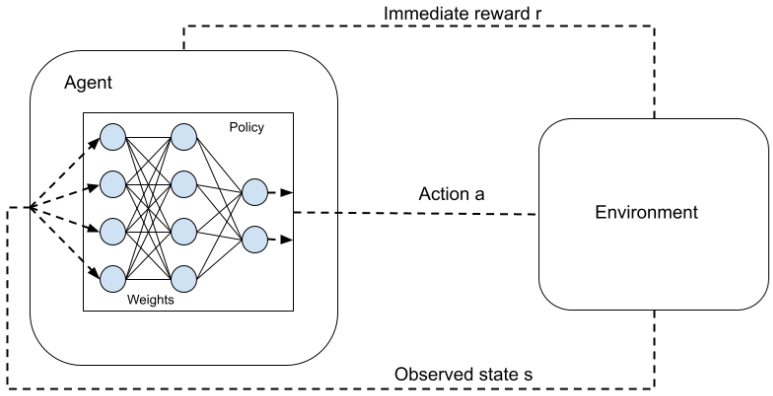



**Figure 4.9:** The architecture of DQL algorithm [30]

turn into an unstable average reward. In another word, it causes that optimal policy cannot be reached with small changes of Q-values. Therefore, two mechanisms i.e., experience replay and target Q-network, are introduced to address these problems.

- Experience replay: In this algorithm, an experience replay or replay memory is a buffer with limited space that can accommodate agents' experiences as a tuple including: $[s_i, a_i, r_i, s_{i+1}]$. Then, random samples know as mini baches of the tuples from replay memory are selected to train the NN. It is worth mentioning that the reason for random selection is to eliminate the possible correlations between observations to obtain stable learning.

- Target Q network: In DQL, two DQN are defined whose architectures are the same. The idea behind using two DQN is that, during training, the Q values which is used to update Q network may be shifted. The latter issue leads to the destabilization of the algorithm. To solve this issue, one extra DQN is defined called target network. The weight of the target network periodically is frozen

and after N steps these weights are updated by the DQN weights. Therefore, the Correlation between target and Q values decreases, which stabilizes the algorithm.

# Chapter 5

# Experimental Setup

This chapter will go through the implementation and evaluation of the proposed RL-based frameworks for vRAN. Two model-free RL algorithms are designed in order to deal with RRM in vRAN and then their performance in typical network scenarios is analyzed and the possible enhancement of both methods is compared. In this regard, firstly the configuration of the simulation platform is discussed whereby the simulation scenario is explained. Then the detail of the presented approach is demonstrated. Eventually, the system testing and evaluation process are described.

## 5.1 Configuration of Simulation Platform

The simulations that will be obtained in this chapter use the ns-3 simulator with the LENA module and also ns-3-ai module described in section 3.2 and 3.3. It has been chosen to consider LENA module which is the complete implementation of LTE. Further, the generalization to 5G NR can be simply done, since the radio access scheme in both generations, i.e., 4G and 5G, are quite analogous. It is also chosen to address the downlink traffic in LTE link which corresponds to the Physical Downlink Shared Channel (PDSCH).

As it is explained in section 3.2, the LENA module is provided with all the integral elements of LTE architecture, UE, RAN, and EPC. The main scenario is designed to have multiple UEs have connected to one eNodeB in one cell. So, the location and mobility model of UEs are set in such a way that provides the situation whereby the UEs receive different SINRs and channel conditions. As well, the mobility of UEs mimics more common situations in real networks. In this regard, 5 UEs located in different distances (descending order) connected to eNodeB. So, GridPositionAllocator model is set, which enables UEs to locate in a

grid-topology. Also, the RandomWalk2dMobilityModel model is set to enable UEs to move with speed 3 m/s at random directions. Figure 5.1 represents a high-level view of the network topology.

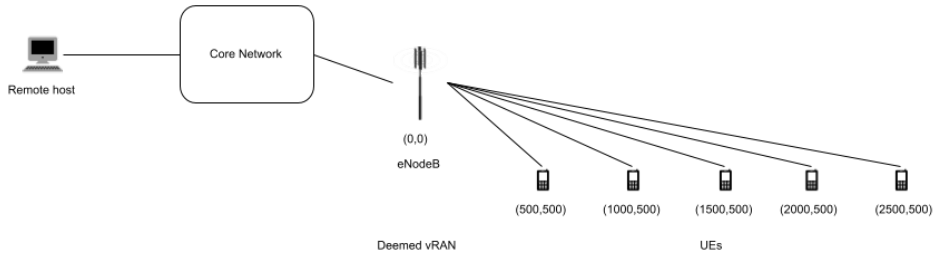



**Figure 5.1:** High-level view of the network topology

Further, a remote end-user, known as a remote host, is provided and is connected to the core network. The connection is set by a point-to-point wired connection to the PGW using the PointToPointHelper topology helper. This connection is provided by 100 Gbit/s and 0 loss and delay to eliminate the effects, e.g., bottleneck, that this link might have on the downlink channel between eNodeB and UEs. The remote host is enabled with applications that generate traffic to be sent to UEs. The key parameters of the application are:

- Packet size ($p_{size}$): 1024 byte
- Data rate ($\delta$)= 10 mbyte/s
- Traffic type: CBR traffic
- Protocol type: TCP

Additionally, the inter-packet intervals $\gamma$ is calculated by expression 5.1:

$$\gamma = p_{size} \times 8/\beta \tag{5.1}$$

Where the $\beta = \delta \times 1e + 6$ denotes the bit rate.

Above all, the parameters that are specified in the designing of the RAN are provided in table 5.1. In this experiments, other parameters that are specified are the scheduler type, the channel propagation model and AMC model, etc:

- Scheduler: Round Robin (RR) scheduler is set in MAC entity, for the sake of simplicity and also in RR all active UEs get a fair share of resources.

| Simulation time | 300 seconds |
|---|---|
| TCP application data rate | 10 Mbit/s |
| Propagation loss model | TwoRayGroundPropagationLossModel |
| MAC entity scheduler | rrMacScheduler |
| AMC model | Piro |
| eNodeB Tx power | 30 dB |
| NoiseFigure | 5dB |
| Frequency Band LTE | 15 MHz- 75 RBs |
| LTE carrier frequency | 2.1 GHz |
| eNodeB Tx power | 30 dB |
| RLC mode | UM |

**Table 5.1:** Simulation parameters

- propagation model: Two-Ray ground propagation model calculates the path losses between a eNodeB antenna and UEs antenna when they are in line of sight (LOS). Generally, the two antennas each have different heights. Therefore, in the simulation, the maximum height of eNodeB and UEs are set to 10 and 2 meters respectively. The other parameter is system loss which is set to 2 dB.

- AMC model: As described in section 3.2.1 the AMC model is set to PiroEW2010

## 5.2 Design of RL-based Frameworks

Two well-known model-free Rl algorithms: DQL and SARSA, are designed and applied in this thesis to improve and optimize the RRM in vRANs. Briefly, the key RL parameters that are used in the experiments are described below:

- Context space: The environment variables that have an effective impact on efficient resource allocation to UEs are: CQI and transmission buffer head of line delay (TXBHLD). Indeed, CQI is used because it is a quantized representation of Signal to Noise Ratio (SNR) as a high correlated parameter in resource provisioning. Besides CQI, the status of transmission buffer including the head of line delay and amount of data waiting to be transmitted toward users are also effective in resource provisioning. In this solution, head of line delay is considered for the sake of simplicity. Thus, context space $S \in \mathbb{R}$ includes context vectors $s_n = \{CQI_n, TXBHLD_n\}$, where n is the $n^{th}$ monitoring slot. It is worthwhile to recall that in each monitoring slot $n \in N$

the context vector $s_n$ is observed, sequentially an action $a_n$ is taken and the corresponding reward $r_n(s_n, a_n)$ is received.

- Action space: The action space $A$ with dimention of M=10 includes choices for the eligible MCS values. Here in this implementation the MCS values defined by the 3GPP standard are quantized to reduce the cardinality of the action space for the sake of simplicity. Thus an action $a_n$ in the $n^{th}$ monitoring slot will be selected from $A = \{6,10,12,14,16,18,20,24,26,28\}$.

- Reward signal: It is chosen to consider the throughput as the KPI. So, to meet this KPI requirement at each UE, it is essential to provide traffic flow with radio resources such that the normalized values of throughputs are always maximized.

- Action Selection: To estimate the action-values it is chosen to implement $\epsilon$-greedy action selection policy. This algorithm aims to select the best actions to maximize the total reward over time. In this regard, two main parameters must be set: $\epsilon$ and decay factor which are set to 0.5 and 0.99 respectively. The algorithm of action selection is defined based on the algorithm below:

**Algorithm 1** Choose_Action Algorithm

**Parameters:** $s_i$ and $a_i$: state and corresponding action in i-th monitoring slot

```
function CHOOSEACTION(s_i)
    if random_uniform() ≥ ε then
        a_i ← MaxQ(s_i, .)
    else
        a_i ← random action from possible action space
    end if
    return a_i
end function
```

Before explaining the formulation of RL frameworks, it should be stated that making decisions is operated under two different durations. In other words, at the beginning of each decision period, the agent decides the MCS values, and it does not change the selected MCSs until the end of the decision period. A decision period comprises N monitoring slots, recalling that in each monitoring slot, the agent observes a context vector $s_i$, takes action $a_i$, and receives reward signal $r_i$. Indeed, the duration of each observation period is of 1 TTI duration, i.e., 1 TTI is equal to 1 ms, and accordingly, each decision period comprising N monitoring slots is of N TTI duration which is N ms. Here in this work, it is chosen to consider: N=10 and N=100.

### 5.2.1 SARSA Framework Formulation

As it is described in section 4.3.5, SARSA is an on-policy TD control method that uses TD for updating. In addition, here in this problem, we need to deal with non-episodic tasks then it is required to adopt average reward setting rather than discounting factor.

It is worthwhile to spend a bit of time explaining that the interaction of the RL agent and the environment can fall into two main categories: non-episodic tasks and episodic tasks. The latter is a basic solution and described in section 4.3.5 using discount factor in updating procedure. On the other hand, the non-episodic tasks address the situations wherein the interaction between the RL agent and the environment continues and there is no termination state. In such conditions, the discounting effect will be eliminated, and instead, the term of average reward will be added. The latter leads to the same importance between delayed reward and immediate reward for the RL agent.

In the non-episodic tasks, the average reward term is obtained as a difference between rewards and average reward, as shown in the expression 5.2:

$$G_t = R_{t+1} - r_\pi + R_{t+2} - r_\pi + R_{t+3} - r_\pi + ... \tag{5.2}$$

As described in [28], $G_t$ is called differential return and the corresponding value function is called differential value function. In this setting the TD error is re-written as expression 5.3:

$$\delta = R_{t+1} - r_\pi + Q_(s_{t+1}, w_{t+1}) - Q_(s_{t+1}, w_t) \tag{5.3}$$

In addition the corresponding weight vector is re-written by expression 5.4:

$$w_{t+1} = w_t + \alpha\delta_t \nabla q(s_{t+1}, a_t, w_t) \tag{5.4}$$

In our problem, UEs continuously send CQI to the RL agent and the RL agent continuously acts on the possible solutions, then the RL agent must follow the average reward setting. Thus, the target action-value is calculated based on expression 5.5, as stated in [31], is the bootstrapping estimate of the action-values of the next state:

$$Q_{t+1} = r(s_t, a_t) - r_\pi + Q_(s_{t+1}, a_{t+1}, w) \tag{5.5}$$

Then the difference of target action-value and current action-value enables the learning procedure. Afterward, using the error value, calculated based on expression 5.3 refers to TD, enables the updating of both average reward value $r_\pi$ and weight

vector $w$. In the latter update, the gradient descent mechanism is used, however, due to impacts of weight vector on bootstrapping target, the gradient descent is biased therefore is called semi-gradient method. the following algorithm described the entire procedure of the proposed SARSA algorithm.

**Algorithm 2** Main_Procedure SARSA

**Parameters:** $s_i$, $s_{i+1}$: states at steps $i$ and $i+1$, $a_i$, $a_{i+1}$: actions at steps $i$ and $i+1$, $\alpha$: learning rate, $r_\pi$: average reward, $\epsilon$: epsilon, $\mathcal{L}$: loss, $N$: decision making periodicity, Initialize the action_value function $Q$ with random weights $w$,

**procedure** MAIN()
  **for** <the $i^{th}$ decision period> **do**
    **for** <monitoring slot j in decision period i, $j \in \{1, .2, ..., N\}$> **do**
      **if** i=1 **then**
        **if** j=1 **then**
          observe $s_j$, $a_j \leftarrow a_i \leftarrow CHOOSEACTION(s_j)$
        **else**
          observe $s_j$, $a_j \leftarrow a_i$
        **end if**
      **else**
        **if** j=1 **then**
          $s_j = s_i$, $a_j \leftarrow a_i \leftarrow CHOOSEACTION(s_j)$
        **else**
          observe $s_j$, $a_j \leftarrow a_i$
        **end if**
      **end if**
      $r_j \leftarrow$ receive the reward
    **end for**
    $s_{i+1} = \sum_{j=1}^{N} \frac{S_j}{N}$ ▷ compute mean context vector over the i-th decision period
    $r_i = \sum_{j=1}^{N} \frac{r_j}{N}$ ▷ compute average reward over the i-th decision period
    $a_{i+1} \leftarrow CHOOSEACTION(s_{i+1})$
    $\mathcal{L} = r_i - r_\pi + Q(s_{i+1}, a_{i+1}, w) - Q(s_i, a_i, w)$ ▷ temporal Difference
    $r_\pi = r_\pi + \alpha\mathcal{L}$ ▷ Estimate average reward
    $w = w + \alpha\mathcal{L} \bigtriangledown Q(s_i, a_i, w)$ ▷ update the weights
    $\epsilon = \epsilon * 0.99$
    $s_i = s_{i+1}$
    $a_i = a_{i+1}$
  **end for**
  **end procedure**

### 5.2.2 DQL Framework Formulation

The next RL-based RRM is implemented based on DQL. As it is described in section 4.3.6, DQL algorithm follows an off-policy method that uses NN to approximate the action-value function. NN not only enables DQL to be implemented on the non-episodic tasks Like SARSA algorithm but also helps the algorithm to learn more effectively when dealing with infinite states. Additionally, DQL applies two effective techniques: two separated NNs for target and predicted action-values and also replay memory, which can ensure considerable convergence and stability in the learning process.

The RL agent uses action-value function $Q(s_n, a_n)$ which is the expected cumulative future reward (discounted reward) when it is in state $s$ and selects action $a$. The action-value function is given by:

$$Q(s_t, a_t) = \mathbb{E}[\sum_{k=t}^{T} \gamma^{k-t+1} r_t] \tag{5.6}$$

Let $\gamma$ be the discount factor, if it is equal to zero, $Q(s_t, a_t)$ only considers the immediate reward, if it is one, then $Q(s_t, a_t)$ considers sum of the rewards. Here the $\gamma = 0.8$. Recalling that the action-value function can be written as:

$$Q(s_t, a_t) = r_t + \gamma Q(s_{t+1}, a_{t+1}) \tag{5.7}$$

A NN performs the approximation of the $Q(s_t, a_t)$, and let the $w$ standing network weight, the action-value function at observation period $t$ can be written as $Q(s_t, a_t|w)$. Then NN approximates the action-value function and will give the optimal policy $\pi^*(s)$:

$$\pi^*(s) = arg \max_a Q^*(s_t, a_t|w) \tag{5.8}$$

Let the $Q^*(s_t, a_t)$ be the optimal action-value function given by NN approximation. Then the $a_{t+1}$ is followed by $\pi^*(s_{t+1})$. Successively, the target action-value is given by:

$$\tilde{Q}(s_t, a_t|w) = r_t + \gamma Q(s_{t+1}, a_{t+1}|w) \tag{5.9}$$

Afterward, $w$ must be minimized using loss function. Here, Mean Square Error (MSE) is used which performs as expression 5.10:

$$\mathcal{L} = \frac{1}{T} \sum_{t=0}^{T} (\tilde{Q}(s_t, a_t|w) - Q(s_{t+1}, a_{t+1}|w))^2 \tag{5.10}$$

Indeed, the RL agent at decision period $t$, at state $s_t$ obtains action $a_t$ according to expression 5.8 and then observes next state $s_{t+1}$. Then, the tuple ($s_t$,$a_t$,$r_t$,$s_{t+1}$) is stored into replay memory.

**Algorithm 3** Main_Procedure DQL

**Parameters:** $s_i$, $s_{i+1}$: states at steps i and i+1, $a_i$, $a_{i+1}$: actions at steps i and i+1, $\alpha$: learning rate, $\gamma$: discount factor, $\epsilon$: epsilon, $\mathcal{L}$: loss, N: decision making periodicity, $\mathcal{U}$ no. of steps to reset target network's weights, Initialize the $\mathcal{Q}$ and $\tilde{\mathcal{Q}}$ network with random weights $w$ and $\tilde{w}$, Initialize replay memory $\mathcal{D}$ to capacity D,

**procedure** MAIN()
  **for** <each decision period> **do**
    **for** <monitoring slot j in decision period i, $j \in \{1, .2, ..., N\}$> **do**
      **if** i=1 **then**
        **if** j=1 **then**
          observe $s_j$, $a_j \leftarrow a_i \leftarrow CHOOSEACTION(s_j)$
        **else**
          observe $s_j$, $a_j \leftarrow a_i$
        **end if**
      **else**
        **if** j=1 **then**
          $s_j = s_i$, $a_j \leftarrow a_i \leftarrow CHOOSEACTION(s_j)$
        **else**
          observe $s_j$, $a_j \leftarrow a_i$
        **end if**
      **end if**
      $r_j \leftarrow$ receive the reward
    **end for**
    $s_{i+1} = \sum_{j=1}^{N} \frac{S_j}{N}$ ▷ compute mean context vector over the i-th decision period
    $r_i = \sum_{j=1}^{N} \frac{r_j}{N}$ ▷ compute average reward over the i-th decision period
    store $(s_i, a_i, r_i, s_{i+1})$ in $\mathcal{D}$
    select random batch of transitions $(s_t, a_t, r_t, s_{t+1})$ from $\mathcal{D}$
    $\tilde{Q}(s_t, a_t|w) = r_t + \gamma \max_a Q(s_{t+1}, a_{t+1}|w)$
    $\mathcal{L} = MSE(\tilde{Q}(s_t, a_t|\tilde{w}), Q(s_t, a_t|w))$
    Update and optimize $Q$ weights $w$ by stochastic gradient descent
    every $\mathcal{U}$ steps reset $\tilde{\mathcal{Q}}$ with $\mathcal{Q}$ network Weights.
    $\epsilon = \epsilon * 0.99$
    $s_i = s_{i+1}$
  **end for**
  **end procedure**

Then equal to mini-batch size, here it is set to 64, random samples are selected from replay memory, and based on them, the weight $w$ and policy $\pi$ will be updated.

The complete procedure of the DQL algorithm is expressed in the algorithm 3.

### 5.2.3 Design of Neural Network

In this section, we explain the design of NN that is exploited in the DQL framework. As it is explained in the previous section, two NN with the same architecture must be initialized at the beginning of the simulation, calling them: $\mathcal{Q}$ the action-value network and $\tilde{\mathcal{Q}}$, the target action-value network. The logic behind using this strategy is to stabilizing the learning process and also more efficient learning. In this regard, the weight of $\tilde{\mathcal{Q}}$ is frozen, and every $\mathcal{U} = 200$ steps, these weights are updated by weights of $\mathcal{Q}$.

In addition, the size of NN in terms of the number of hidden layers and also the number of neurons in each layer must be taken into account. For example, an inordinately large number of neurons in hidden layers causes overfitting and also it increases the time it takes to train the network. The NNs have been deployed exploiting PyTorch framework through ns-3-ai. In this regard the following structure is considered in the design of the NNs:

- The input layer has a number of neurons equal to the size of samples in mini-batch and is set to 64.
- One hidden layer with size 32 is activated by ReLU function.
- The output layer has a number of neurons equal to the size of action space and is set to 10.

Further, the training of the designed model is driven using the random samples i.e., mini-batch, stored in replay memory. Further, the adopted loss criterion is the MSE loss function, Also, adam optimization,i.e., an alternative to stochastic gradient descent algorithm, is exploited in the learning process. Additionally, the general setting of the implementation are given:

- learning rate = 0.001
- reward discount factor = 0.8
- replay memory size = 2000 trajectories
- Mini-batch size = 64 trajectories
- steps to update target network = 200

## 5.3 Results

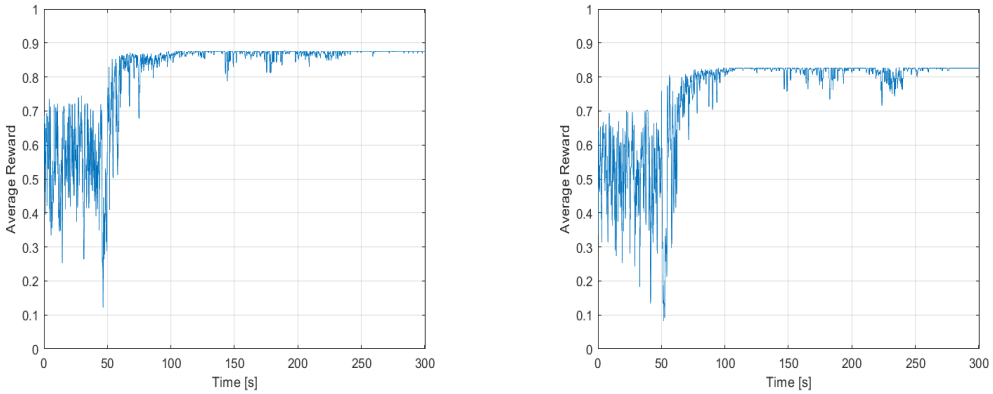
In this section, we evaluate the performance of the proposed RL-based frameworks using the ns-3 simulation in a real-time manner. As expressed in the previous section, the algorithms are operated under two different decision periods, when N=10, the corresponding decision period is 10 ms, when N=100, the corresponding decision period is 100 ms. Thus, (A) we discuss the learning performance of SARSA and DQL algorithms under different decision periods, (B) We assess the variation of performance metrics vs SINR. Additionally, the latter includes a comparison between the result of proposed RL-based frameworks with those of the CQI-prediction method presented in [21].

It is worth spending a bit of time explaining the CQI-prediction model [21]. This approach uses DL and in particular, Long Term Short Memory (LSTM) to predict CQI values received by eNodeB. In a period of times, the CQI values transmitted by UEs are gathered, and based on them the LSTM will be trained, successively rather than receiving CQI from UEs, the CQI's will be predicted until the MSE between predicted CQIs and valid CQIs reported by UEs is less than a threshold. Then eNodeB based on the predicted CQI measures the MCS values through the 3GPP standard.

### 5.3.1 Learning Performance of Proposed RL-based Frameworks

The simulations are carried out for the same duration of 300 seconds, and also all the simulation configurations are considered adequate for the sake of fairness. Then learning performance of the RL algorithms is provided in terms of convergence of average reward values. Since the RL algorithms perform on all UEs, each UE can receive different channel conditions, so, average rewards of the UEs can be different. Thus here only the average case is provided.

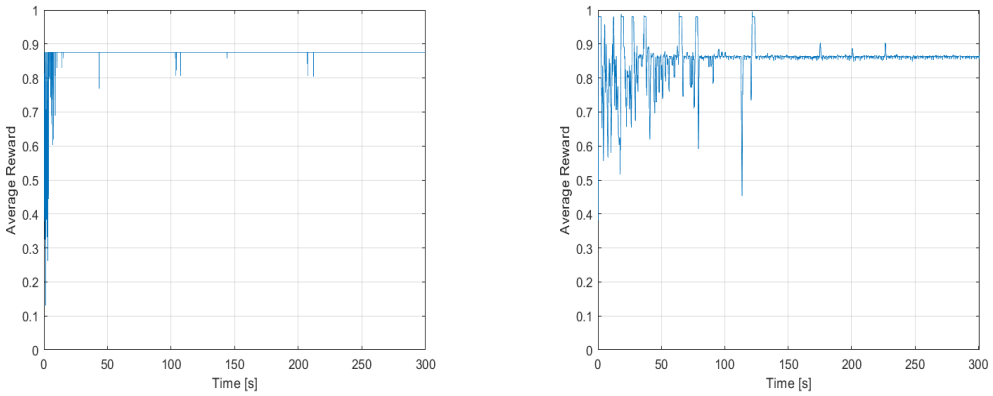
Figure 5.2 represents the learning curves of the DQL framework in terms of convergence of reward values as a function of time. Figure 5.2a depicts the average reward values of DQL under 10 ms decision period. It can be noticed that after 75 seconds, the average reward almost converges. On the other hand, as it can be seen, in Figure 5.2b, DQL under 100 ms decision period convergence takes more time with respect to other cases and it can be seen that some times after convergence there are slight variations. However, it is important to note here that the longer decision period has a higher exploration time with respect to the shorter decision period. It follows that the shorter decision period leads to easier convergence of reward values which leads to fast learning.




**(a)** **(b)**

**Figure 5.2:** DQL algorithm: Learning performance,(a) N=10 decision period (b) N=100 decision period

Figure 5.3 illustrates the learning curves of the SARSA algorithm in terms of convergence of reward values as a function of time. From Figure 5.3a it can be noted that SARSA, when adopting decision period of 10 ms, outperforms all the other solutions. In addition, Figure 5.3b which represents the convergence of SARSA under 100 ms decision period, converges faster with respect to those of DQL under 100 ms decision period.




**(a)** **(b)**

**Figure 5.3:** SARSA algorithm: Learning performance,(a) N=10 decision period (b) N=100 decision period

Based on the learning performance represented in Figures 5.3 and 5.2, we can

conclude that SARSA as an on-policy method performs better with respect to DQL which is an off-policy method.

### 5.3.2 Variation of Performance Metrics Vs SINR

Figure 5.4 shows the average throughput in the function of SINR when adopting the two decision periods for different algorithms. We observe that an increase in SINR causes the throughput to increase, which is as expected. Further, the throughput achieved by SARSA algorithm under 10 ms decision period is higher than the throughput achieved by the CQI-prediction and also DQL. Since the average values of selected MCS values in the RL frameworks are larger with respect to those of CQI-prediction algorithm, the amount of transmitted data with respect to the CQI-prediction method increase.

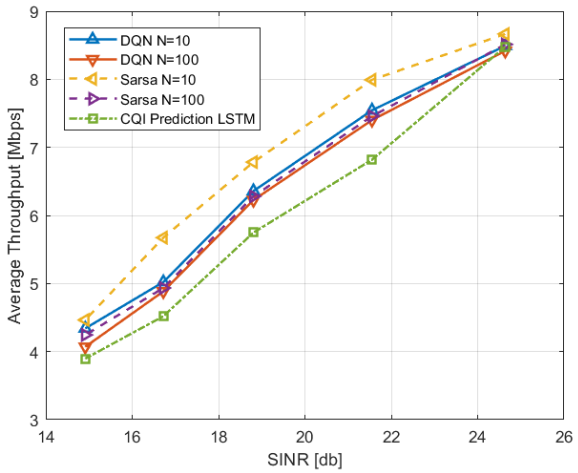



**Figure 5.4:** Variation of throughput vs SINR

Additionally, Figure 5.5 shows the evolution of performance metrics namely: TX buffer size, latency, and loss ratio with respect to SINR. It is worth recalling that, the latency (computed in ms) and loss ratio are measured at the IP layer and the transmission buffer (computed in Kbytes) is measured at the MAC layer.

As it is expected, by increasing the SINR, TX buffer size, latency decrease, and loss ratio decrease. It can be noticed that, in both algorithms, adopting a higher duration of decision period deteriorates the performance of the corresponding algorithm, and it can be noticed in all performance metrics. The reason for this issue is that, by adopting long decision periods, more time is required to train the

RL model, i.e. long exploration time is required. Indeed in the exploration time, most of the MCS values are random.

Therefore, by low values of MCSs, throughput decreases, and by higher than

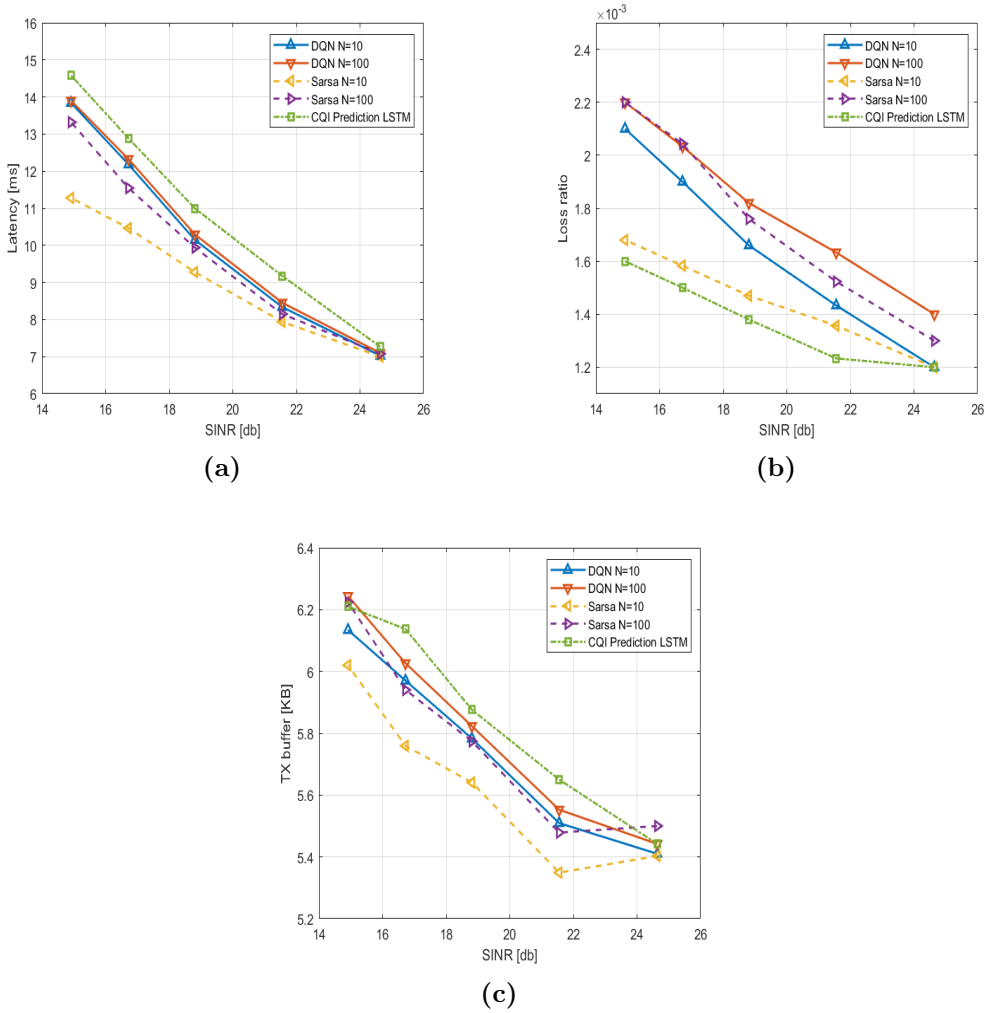



**Figure 5.5:** Variation of performance metrics with SINR,(a) Latency vs SINR (b) Loss ratio vs SINR, (c) Tx buffer size vs SINR

expected MCS values, delay and loss ratio increase. The aforementioned factors impact TX buffer size as well. It is worth recalling that in the simulations presented, for both decision periods, equal simulation times have been considered for the sake of fairness.

One important issue that should be mentioned is that in the CQI-prediction

algorithm, MCS values are obtained based on the regular way defined by 3GPP. The selected values by the CQI-prediction algorithm most of the time- during simulation- are smaller than MCS values that are obtained by RL algorithms. Consequently, the CQI-prediction method will experience lower value of throughputs in comparison with developed RL algorithms. On the other hand, high values of MCS might cause a slightly higher loss ratio in proposed RL algorithms, as Figure 5.5b represents.

# Chapter 6

# Conclusion And Future Work

In this thesis, we studied the benefits of the 5G technology, the new advanced technologies that it has introduced to solve many open issues by the previous mobile generations, and also new challenges that should have been faced. In this regard, we proposed and designed RL-based and deep RL-based solutions to deal with radio resource management in vRAN. Also in the previous chapter firstly we assessed the performance of the solutions in network scenarios including multiple non-stationary UEs, and then we compared the two proposed solutions with respect to each other. With the scenarios and assumptions used in this thesis work, our system evaluation has demonstrated that the proposed RL-based and deep RL-based approaches are promising systems for dynamic radio resource management in vRANs.

Indeed, we observed that the proposed RL-based approach has better performance in comparison with the deep RL-based approach. Indeed, the performance of the proposed algorithms depends on the context space and action space cardinality, while in this thesis work these parameters had limited size, then deep learning just added some complexity and also a delay in the learning process.

Although in this solution the main goal is increasing the users' throughputs, in more advanced solutions it may be chosen to consider other parameters as possible reward value, e.g., loss ratio, latency, etc.

# Bibliography


[1] Cisco White Paper. «Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2010-2015». In: (Feb. 2011). URL: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.pdf (cit. on pp. 1, 2, 7).

[2] Gilberto Berardinelli, Luis Angel Maestro Ruiz de Temino, Simone Frattasi, Muhammad Imadur Rahman, and Preben Mogensen. «OFDMA vs. SC-FDMA: performance comparison in local area imt-a scenarios». In: *IEEE Wireless Communications* 15.5 (2008), pp. 64–72. DOI: 10.1109/MWC.2008.4653134 (cit. on p. 2).

[3] Jeffrey G. Andrews, Stefano Buzzi, Wan Choi, Stephen V. Hanly, Angel Lozano, Anthony C. K. Soong, and Jianzhong Charlie Zhang. «What Will 5G Be?» In: *IEEE Journal on Selected Areas in Communications* 32.6 (2014), pp. 1065–1082. DOI: 10.1109/JSAC.2014.2328098 (cit. on p. 2).

[4] Mahbubul Alam. *Is 5G Friend or Foe for Autonomous Vehicle?* Dec. 2018. URL: https://medium.datadriveninvestor.com/is-5g-friend-or-foe-for-autonomous-vehicle-72ee70800031 (cit. on p. 2).

[5] Mahbubul Alam. *Is 5G Friend or Foe for Autonomous Vehicle?* Dec. 2018. URL: https://medium.datadriveninvestor.com/is-5g-friend-or-foe-for-autonomous-vehicle-72ee70800031 (cit. on p. 3).

[6] *5G - Fifth generation of mobile technologies.* URL: https://www.itu.int/en/mediacentre/backgrounders/Pages/5G-fifth-generation-of-mobile-technologies.aspx (cit. on p. 2).

[7] *A Global Partnership.* URL: https://www.3gpp.org/technologies/keywords-acronyms/98-lte (cit. on p. 5).

[8] Michael Garyantes. «Virtual Radio Access Network opportunities and challenges». In: *2015 36th IEEE Sarnoff Symposium*. 2015, pp. 24–28. DOI: 10.1109/SARNOF.2015.7324637 (cit. on p. 7).

[9] Jose A. Ayala-Romero, Andres Garcia-Saavedra, Marco Gramaglia, Xavier Costa-Perez, Albert Banchs, and Juan J. Alcaraz. «VrAIn: A Deep Learning Approach Tailoring Computing and Radio Resources in Virtualized RANs». In: *The 25th Annual International Conference on Mobile Computing and Networking*. MobiCom '19. Los Cabos, Mexico: Association for Computing Machinery, 2019. ISBN: 9781450361699. DOI: 10.1145/3300061.3345431. URL: https://doi.org/10.1145/3300061.3345431 (cit. on p. 7).

[10] Samsung. *"Virtualized RAN Challenges Debunked"*. URL: https://images.samsung.com/is/content/samsung/p5/global/business/networks/insights/white-paper/virtualized-ran-challenges-debunked/Samsung_Virtualized-RAN-Challenges-Debunked.pdf (cit. on pp. 7, 9).

[11] Sulastri Manap, Kaharudin Dimyati, Mhd Nour Hindia, Mohamad Sofian Abu Talip, and Rahim Tafazolli. «Survey of Radio Resource Management in 5G Heterogeneous Networks». In: *IEEE Access* 8 (2020), pp. 131202–131223. DOI: 10.1109/ACCESS.2020.3002252 (cit. on p. 10).

[12] ETSI TS 136 101. *LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) radio transmission and reception*. 2011 (cit. on pp. 10, 14).

[13] Mar. 2021. URL: https://www.viavisolutions.com/en-us/5g-architecture (cit. on p. 10).

[14] Giuseppe Piro, Nicola Baldo, and Marco Miozzo. «An LTE Module for the Ns-3 Network Simulator». In: *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*. SIMUTools '11. Barcelona, Spain: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2011, pp. 415–422. ISBN: 9781936968008 (cit. on pp. 10, 15, 16).

[15] George F. Riley and Thomas R. Henderson. «The ns-3 Network Simulator.» In: *Modeling and Tools for Network Simulation*. Ed. by Klaus Wehrle, Mesut Günes, and James Gross. Springer, 2010, pp. 15–34. ISBN: 978-3-642-12330-6. URL: http://dblp.uni-trier.de/db/books/collections/Wehrle2010.html#RileyH10 (cit. on p. 11).

[16] Giuseppe Piro, Nicola Baldo, and Marco Miozzo. «An LTE module for the ns-3 network simulator». In: ACM, Apr. 2012. DOI: 10.4108/icst.simutools.2011.245571 (cit. on p. 13).

[17] Guangxu Zhu, Dongzhu Liu, Yuqing Du, Changsheng You, Jun Zhang, and Kaibin Huang. «Toward an intelligent edge: Wireless communication meets machine learning». In: *IEEE communications magazine* 58.1 (2020), pp. 19–25 (cit. on p. 17).

[18] Mowei Wang, Yong Cui, Xin Wang, Shihan Xiao, and Junchen Jiang. «Machine learning for networking: Workflow, advances and opportunities». In: *Ieee Network* 32.2 (2017), pp. 92–99 (cit. on p. 17).

[19] Martin Abadi et al. «TensorFlow: A system for large-scale machine learning». In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 2016, pp. 265–283. URL: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf (cit. on p. 17).

[20] François Chollet. *Deep Learning with Python*. Manning, Nov. 2017. ISBN: 9781617294433 (cit. on p. 17).

[21] Hao Yin, Pengyu Liu, Keshu Liu, Liu Cao, Lytianyang Zhang, Yayu Gao, and Xiaojun Hei. «Ns3-Ai: Fostering Artificial Intelligence Algorithms for Networking Research». In: *Proceedings of the 2020 Workshop on Ns-3*. WNS3 2020. Gaithersburg, MD, USA: Association for Computing Machinery, 2020, pp. 57–64. ISBN: 9781450375375. DOI: 10.1145/3389400.3389404. URL: https://doi.org/10.1145/3389400.3389404 (cit. on pp. 17, 18, 44).

[22] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. 2nd ed. Adaptive Computation and Machine Learning. Cambridge, MA: MIT Press, 2018. 504 pp. ISBN: 978-0-262-03940-6 (cit. on p. 20).

[23] Giuseppe Bonaccorso. *Machine Learning Algorithms: A Reference Guide to Popular Algorithms for Data Science and Machine Learning*. Packt Publishing, 2017. ISBN: 1785889621 (cit. on p. 20).

[24] Jovani Dalzochio, Rafael Kunst, Edison Pignaton, Alecio Binotto, Srijnan Sanyal, Jose Favilla, and Jorge Barbosa. «Machine learning and reasoning for predictive maintenance in Industry 4.0: Current status and challenges». In: *Computers in Industry* 123 (2020), p. 103298. ISSN: 0166-3615. DOI: https://doi.org/10.1016/j.compind.2020.103298. URL: https://www.sciencedirect.com/science/article/pii/S0166361520305327 (cit. on p. 21).

[25] John M. Mulvey. «Machine Learning and Financial Planning». In: *IEEE Potentials* 36.6 (2017), pp. 8–13. DOI: 10.1109/MPOT.2017.2737200 (cit. on p. 21).

[26] Joseph A. Cruz and David S. Wishart. «Applications of Machine Learning in Cancer Prediction and Prognosis». In: *Cancer Informatics* 2 (2006), p. 117693510600200030. DOI: 10.1177/117693510600200030. eprint: https://doi.org/10.1177/117693510600200030. URL: https://doi.org/10.1177/117693510600200030 (cit. on p. 21).

[27] Junyan Hu, Hanlin Niu, Joaquin Carrasco, Barry Lennox, and Farshad Arvin. «Voronoi-Based Multi-Robot Autonomous Exploration in Unknown Environments via Deep Reinforcement Learning». In: *IEEE Transactions on Vehicular Technology* 69.12 (2020), pp. 14413–14423. DOI: 10.1109/TVT.2020.3034800 (cit. on p. 22).

[28] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018. URL: http://incompleteideas.net/book/the-book-2nd.html (cit. on pp. 28, 39).

[29] Faris B. Mismar, Brian L. Evans, and Ahmed Alkhateeb. «Deep Reinforcement Learning for 5G Networks: Joint Beamforming, Power Control, and Interference Coordination». In: *IEEE Transactions on Communications* 68.3 (2020), pp. 1581–1592. DOI: 10.1109/TCOMM.2019.2961332 (cit. on p. 28).

[30] Nguyen Cong Luong, Dinh Thai Hoang, Shimin Gong, Dusit Niyato, Ping Wang, Ying-Chang Liang, and Dong In Kim. «Applications of Deep Reinforcement Learning in Communications and Networking: A Survey». In: *IEEE Communications Surveys Tutorials* 21.4 (2019), pp. 3133–3174. DOI: 10.1109/COMST.2019.2916583 (cit. on p. 33).

[31] Sharda Tripathi, Corrado Puligheddu, and Carla Fabiana Chiasserini. «An RL Approach to Radio Resource Management in Heterogeneous Virtual RANs». In: *2021 16th Annual Conference on Wireless On-demand Network Systems and Services Conference (WONS)*. 2021, pp. 1–8. DOI: 10.23919/WONS51326.2021.9415591 (cit. on p. 39).