Machine Learning for Wireless Communication

From next-generation spectrum sharing frameworks to communication-aware learning agents

Miguel Camelo

Supervisor **prof. dr. Steven Latré**

Thesis submitted in fulfilment of the requirements for the degree of doctor in computer science
Faculty of Science | Antwerp, 2024

University of Antwerp

University of Antwerp

Faculty of Science

Machine Learning for Wireless Communication

From next-generation spectrum sharing frameworks to communication-aware learning agents

Thesis submitted in fulfilment of the requirements for the degree of doctor in computer science at the University of Antwerp

Miguel Camelo

Antwerp, 2024

Supervisor
prof. dr. Steven Latré

Contact
Miguel Camelo

University of Antwerp
Faculty of Sciences
IDLab - Department of Computer Science
Sint-Pietersvliet 7, 2000 Antwerp, Belgium
M: miguel.camelo@uantwerpen.be

Abstract

As services and networks continue to grow in complexity, the demand for network management automation rises, encompassing not only the services but also the network functions themselves. The main goal behind such networks' management, orchestration, and control systems is to optimize the network resources (e.g., spectrum, networking, computing, etc.) to achieve efficiency while satisfying users' requirements. These kind of networks are the so-called Autonomous Networks (ANs). The recent advances in Artificial Intelligence (AI), and more precisely in Machine Learning (ML), can provide the enablers to allow these AN to manage the resources autonomously and be able to learn and adapt themselves in very complex and dynamic environments optimally. Learning and adaptation are the fundamental capabilities to provide the network of capabilities to perform advanced autonomous tasks such as self-healing, self-diagnosing, and self-provisioning.

In this context, the radio access domain is undergoing a radical transformation towards a fully Autonomous Domain (AD). Wireless networks have experienced significant expansion and are now crucial segments within the more extensive network infrastructure. This transformation is primarily focused on two key aspects of research. Firstly, these networks face inherent complexities due to factors such as user mobility and the unreliability of wireless links. Moreover, the demanding requirements of 5G and future networks, including ultra-low latency (1-10 ms), exceptional reliability (up to five nines), high throughput (up to 20 Gbps), and flexible resource allocation, amplify the need for advanced management techniques to tackle these intricate challenges effectively.

Secondly, a shortage of available spectrum hinders the deployment of 5G technologies. Traditional static frequency plans, which allocate exclusive spectrum portions for single usage or individual users, have become obsolete. While a significant portion of the allocated spectrum remains underutilized, the spectrum used by everyday communication technologies is overutilized. Global efforts are underway to update the exclusive-usage spectrum allocation model and make additional spectrum available for broadband data, thereby increasing spectrum reuse. Critical enabling technologies, such as Cognitive Radios (CRs), continue to evolve providing capabilities to intelligently adapt radios to their environment aiming to achieve Dynamic Spectrum Access (DSA) optimally.

DSA for spectrum sharing is a good example where there is a continuously growing interest in how to enable the intelligent operation of the radio networks, which is mainly driven by the increased complexity of the network's management compared with the legacy versions, imposes a totally different way to perform operations that were formerly performed purely through human intervention (error pruned), mathematical optimization (too slow), or first-generation AI such as expert/ruled-based systems (deriving the rules are not anymore straightforward). It is here where the new generation of AI, basically driven by Deep Learning (DL) techniques in ML during the last 5-10 years, has

provided a new set of algorithms and techniques to empower the new generation of data analytic and intelligent decision engines to achieve the vision of fully ANs.

To successfully implement and deploy radio access ADs, significant novel contributions and innovations are required in selecting, designing, and deploying AI/ML algorithms managing the available resources (spectrum, networking, computing, storage). This is fundamental as new challenges emerge: deciding when to use traditional management algorithms, ML-based models, or even hybrid approaches; deciding where they have to be deployed (radio vs. fog vs. edge); and how to manage their life cycle (from data harvesting to intelligent decision-making). However, when applying AI to solve problems in networking and to run in networks, it is crucial to focus not only on the optimization problem to solve but also on the design of the learning algorithm itself, considering both requirements when using a given technique and challenges when we want to deploy them in networks.

Until a few years ago, the dominant trend among most AI practitioners for networking was using "vanilla" versions of the ML algorithms to empower their controllers and orchestrators. However, this approach did not account for the specificities and unique requirements of networking functionalities, resulting in it not being a sensible choice. More recently, there has been a growing adoption of specific techniques to design tailored solutions that include aspects such as sustainability (energy consumption and carbon footprint), reliability (algorithms performance in unseen situations), scalability (centralized vs. decentralized deployments), resource awareness (model size vs. accuracy), training efficiency (labels vs. accuracy), communication awareness (communication overhead vs. convergence), or responsiveness (real-time vs. non-real-time training), which can deliver solutions that are (semi-) optimal and suitable to run in networks.

This dissertation investigates some of these challenges from two complementary points of view. Firstly, on the radio network side, we investigate the challenges associated with creating a novel spectrum-sharing framework that is built on top of the concept of radio access ADs, i.e., the radio access network provides autonomy, abstraction, and collaboration, such that we can go beyond the state-of-the-art spectrum sharing systems that are mainly centralized and database-assisted. We approach these challenges by designing a two-tier architecture that enables efficient spectrum sharing, built on top of the concept of Collaborative Intelligent Radio Network (CIRN) and with the guarantee of incumbent protection. Compared to new approaches like Citizens Broadband Radio Service (CBRS) and Licensed Shared Access (LSA), our system requires no central infrastructure to control and grant access to the shared spectrum. Moreover, this is the first work that proposes a system to protect the incumbent's transmissions in a collaborative environment to the author's best knowledge.

We also provide a general framework that enables the development of Traffic Classification (TC) algorithms optimized for wireless networks. Building on top of it, a procedure based on DL to perform TC on spectrum samples is proposed. This procedure enables the management algorithms running at the Gateway (GW) nodes (or beyond) to perform better by having a broader view of the traffic flowing in the shared spectrum. To the best of our knowledge, this is the first framework that allows the development of Radio Access Technologies (RAT)-agnostic spectrum-based TC algorithms. Both frameworks are built on top of a DL-based Technology Recognition (TR) algorithm that is also designed and developed in this dissertation. This algorithm is a fundamental enabler for the proposed

frameworks since it allows them to be ultimately radio technology agnostic. The two-tier framework, which includes the TR algorithm, has been tested and validated in Colosseum, the world's largest Radio Frequency (RF) channel emulator built for the DARPA Spectrum Collaboration Challenge (SC2) competition. Using the Defense Advanced Research Projects Agency (DARPA) Colosseum, we provided strong evidence about the robustness of the proposed approach.

Secondly, on the ML side, we investigate two main challenges when designing DL and Reinforcement Learning (RL) algorithms: 1) how to ensure training efficiency (labels vs. accuracy) when they are used to solve networking problems, and 2) how to guarantee scalability when they need to be deployed in a distributed networking infrastructure. In the first case, we approach this challenge by making the label-efficient TR. For this, the proposed DL algorithm empowering TR is based on Semi-supervised Learning (SSL) that minimizes the need for labeling large data sets of spectrum data (raw In-phase and Quadrature (IQ)), which is a time-consuming and challenging task. In the second case, we approach this challenge by providing a novel approach that allows any table-based Parallel Reinforcement Learning (PRL) algorithm to run RL-based applications with minimal communication overhead in a distributed environment. To the authors' best knowledge, this is the first work that focuses on solving the communication overhead of distributing PRL algorithms without requiring any a priori information about the environment, making the agents communication-aware by design.

Samenvatting

Naarmate diensten en netwerken in complexiteit blijven groeien, neemt de vraag naar automatisering van netwerkbeheer toe, waarbij niet alleen de diensten maar ook de netwerkfuncties zelf worden omvat. Het belangrijkste doel achter dergelijke beheer-, orkestratie- en controlesystemen van netwerken is om de netwerkmiddelen (bijvoorbeeld spectrum, netwerken, computing, enz.) te optimaliseren om efficiëntie te bereiken terwijl aan de behoeften van gebruikers wordt voldaan. Dit type netwerk wordt de zogenaamde Autonomous Networks (ANs) genoemd. De recente vooruitgang in Artificial Intelligence (AI) en meer specifiek in Machine Learning (ML) kan de mogelijkheden bieden om deze ANs in staat te stellen de bronnen autonoom te beheren en zich optimaal aan te passen in zeer complexe en dynamische omgevingen. Leren en aanpassen zijn de fundamentele capaciteiten om het netwerk in staat te stellen geavanceerde autonome taken uit te voeren, zoals zelfherstel, zelfdiagnose en zelfvoorziening.

In deze context ondergaat het domein van radiotoegang een radicale transformatie naar een volledig Autonomous Domain (AD). Draadloze netwerken hebben een aanzienlijke expansie doorgemaakt en zijn nu cruciale segmenten binnen de bredere netwerkinfrastructuur. Deze transformatie richt zich voornamelijk op twee belangrijke onderzoeksaspecten. Ten eerste worden deze netwerken geconfronteerd met inherente complexiteiten door factoren zoals gebruikersmobiliteit en de onbetrouwbaarheid van draadloze verbindingen. Bovendien versterken de veeleisende vereisten van 5G en toekomstige netwerken, waaronder ultra-lage latentie (1-10 ms), uitzonderlijke betrouwbaarheid (tot vijf negens), hoge doorvoersnelheid (tot 20 Gbps) en flexibeleallocatie van middelen, de nood aan geavanceerde beheertechnieken om deze ingewikkelde uitdagingen effectief aan te pakken.

Ten tweede wordt de huidige implementatie van 5G-technologieën belemmerd door een tekort aan beschikbaar spectrum. Traditionele statische frequentieplannen, die exclusieve delen van het spectrum toewijzen voor enkelvoudig gebruik of individuele gebruikers, zijn verouderd. Terwijl een aanzienlijk deel van het toegewezen spectrum onderbenut blijft, wordt het spectrum gebruikt door alledaagse communicatietechnologieën overbenut. Wereldwijde inspanningen zijn gaande om het exclusieve spectrumallocatiemodel te actualiseren en extra spectrum beschikbaar te stellen voor breedbandgegevens, waardoor het hergebruik van spectrum toeneemt. Kritische ondersteunende technologieën, zoals Cognitive Radios (CRs), blijven evolueren en bieden mogelijkheden om radio's intelligent aan te passen aan hun omgeving, met als doel optimale Dynamic Spectrum Access (DSA) te bereiken.

DSA voor spectrumdeling is een goed voorbeeld waarbij er een continu groeiende interesse is in hoe de intelligente werking van de radionetwerken mogelijk gemaakt kan worden. Dit wordt voornamelijk aangedreven door de toegenomen complexiteit van het netwerkbeheer in vergelijking met de oudere versies, wat een totaal andere manier van

opereren vereist. Vroeger werden deze operaties uitgevoerd door puur menselijke interventie (foutgevoelig), wiskundige optimalisatie (te traag) of eerstegeneratie AI, zoals expert-/regelgebaseerde systemen (waarbij het afleiden van regels niet meer vanzelfsprekend is). Hier komt de nieuwe generatie AI, voornamelijk aangedreven door Deep Learning (DL) technieken in ML tijdens de afgelopen 5-10 jaar, in beeld. Deze heeft een nieuwe reeks algoritmen en technieken geleverd om de nieuwe generatie van data-analyse en intelligente besluitvormingsmachines te versterken, waarmee de visie van volledig ANs bereikt kan worden.

Om radiotoegang ADs succesvol te implementeren en in te zetten, zijn significante nieuwe bijdragen en innovaties vereist bij het selecteren, ontwerpen en implementeren van AIs/MLs-algoritmen voor het beheer van beschikbare middelen (spectrum, netwerken, rekenkracht, opslag). Dit is essentieel omdat er nieuwe uitdagingen ontstaan: beslissen wanneer traditionele beheeralgoritmen, op ML gebaseerde modellen of zelfs hybride benaderingen moeten worden gebruikt; beslissen waar ze moeten worden ingezet (radio versus fog versus edge); en hoe hun levenscyclus moet worden beheerd (van gegevensverzameling tot intelligente besluitvorming). Bij het toepassen van AI om netwerkproblemen op te lossen en in netwerken te draaien, is het echter cruciaal om niet alleen te focussen op het op te lossen optimalisatieprobleem, maar ook op het ontwerp van het leeralgoritme zelf, waarbij zowel de vereisten bij het gebruik van een gegeven techniek als de uitdagingen bij netwerkimplementatie worden overwogen.

Tot voor kort was de dominante trend onder de meeste AI-beoefenaars voor netwerken het gebruik van "standaard" versies van ML-algoritmen om hun controllers en orchestrators te versterken. Deze benadering hield echter geen rekening met de specifieke kenmerken en unieke eisen van netwerkfunctionaliteiten, wat resulteerde in een zinloze keuze. Onlangs is er een groeiende acceptatie van specifieke technieken om op maat gemaakte oplossingen te ontwerpen die aspecten omvatten zoals duurzaamheid (energieverbruik en koolstofvoetafdruk), betrouwbaarheid (prestaties van algoritmen in onvoorziene situaties), schaalbaarheid (gecentraliseerde versus gedecentraliseerde implementaties), middelenbewustzijn (modelgrootte versus nauwkeurigheid), trainings-efficiëntie (labels versus nauwkeurigheid), communicatiebewustzijn (communicatie-overhead versus convergentie) of reactievermogen (real-time versus niet-real-time training), die oplossingen kunnen bieden die (semi-) optimaal zijn en geschikt zijn om in netwerken te draaien.

Deze dissertatie onderzoekt enkele van deze uitdagingen vanuit twee complementaire perspectieven. Ten eerste onderzoeken we aan de kant van het radionetwerk de uitdagingen die gepaard gaan met het creëren van een nieuw kader rond spectrumdeling gebaseerd op het concept vanradiotoegang ADs, dat wil zeggen, het radionetwerk biedt autonomie, abstractie en samenwerking, zodat we verder kunnen gaan dan de bestaande spectrumdelingssystemen die voornamelijk centraal zijn en ondersteund door een database. We benaderen deze uitdagingen door een tweelaagse architectuur te ontwerpen die efficiënte spectrumdeling mogelijk maakt, gebaseerd op het concept van Collaborative Intelligent Radio Network (CIRN) en met de garantie van bescherming van de huidige gebruikers. In vergelijking met nieuwe benaderingen zoals Citizens Broadband Radio Service (CBRS) en Licensed Shared Access (LSA), vereist ons systeem geen centrale infrastructuur om de toegang tot het gedeelde spectrum te regelen en te verlenen. Bovendien is dit het eerste werk dat een systeem voorstelt om de transmissies van de huidige gebruikers in een samenwerkende omgeving te beschermen, naar beste weten van de auteur.

Daarnaast bieden we ook een algemeen kader dat de ontwikkeling van Traffic Classification (TC) algoritmen optimaliseert voor draadloze netwerken. Op basis hiervan wordt een procedure voorgesteld op basis van DL om TC uit te voeren op spectrumsamples. Deze procedure stelt de beheeralgoritmen die draaien bij de Gateway (GW)-knooppunten (of verder) in staat beter te presteren door een breder beeld te hebben van het verkeer dat in het gedeelde spectrum stroomt. Voor zover ons bekend, is dit het eerste kader dat de ontwikkeling van spectrumgebaseerde TC-algoritmen mogelijk maakt die onafhankelijk zijn van Radio Access Technologiess (RATs). Beide kaders zijn gebouwd op basis van een op DL gebaseerd Technology Recognition (TR) algoritme dat ook is ontworpen en ontwikkeld in deze dissertatie. Dit algoritme is een fundamenteel instrumentvoor de voorgestelde kaders, aangezien het hen in staat stelt uiteindelijk technologie-agnostisch te zijn. Het tweelaagse kader, inclusief het TR-algoritme, is getest en gevalideerd in Colosseum, 's werelds grootste Radio Frequency (RF) kanaalemulator gebouwd voor de DARPA Spectrum Collaboration Challenge (SC2) competitie. Met behulp van de Colosseum van het Defense Advanced Research Projects Agency (DARPA) hebben we sterke bewijzen geleverd van de robuustheid van de voorgestelde aanpak.

Ten tweede onderzoeken we aan de kant van ML twee belangrijke uitdagingen bij het ontwerpen van DL- en Reinforcement Learning (RL)-algoritmen: 1) hoe de trainingsefficiëntie (labels versus nauwkeurigheid) te waarborgen wanneer ze worden gebruikt om netwerkproblemen op te lossen, en 2) hoe schaalbaarheid te garanderen wanneer ze moeten worden ingezet in een gedistribueerde netwerkinfrastructuur. In het eerste geval benaderen we deze uitdaging door de label-efficiënte TR te maken. Hiervoor is het voorgestelde DL-algoritme dat TR versterkt gebaseerd op Semi-supervised Learning (SSL), dat de noodzaak minimaliseert om grote datasets met spectrumgegevens (ruwe In-phase and Quadrature (IQ)) te labelen, wat een tijdrovende en uitdagende taak is. In het tweede geval benaderen we deze uitdaging door een nieuwe aanpak te bieden die elk op tabellen gebaseerd Parallel Reinforcement Learning (PRL) algoritme in staat stelt RL-gebaseerde toepassingen uit te voeren met minimale communicatie-overhead in een gedistribueerde omgeving. Naar beste weten van de auteur is dit het eerste werk dat zich richt op het oplossen van de communicatie-overhead bij de distributie van PRL-algoritmen zonder enige a priori informatie over de omgeving, waardoor de agenten van nature communicatiebewust zijn.

Acknowledgements

Writing a Ph.D. thesis book is undoubtedly challenging, but writing this section has been the hardest task for me, as it brought back so many memories that cannot fit into a single book. However, below you will find the shorter version of my acknowledgments.

I would like to start by thanking my advisor, Steven Latré, the person who gave me the opportunity to secure a research position in Belgium after my first Ph.D. After our first interview, I knew I was meeting my possible new boss. Over the years, I discovered that Steven was not only a brilliant researcher and an inspiring leader but also a professional role model and, last but not least, someone you can really trust. Thank you, Steven, for being the person you are.

When I joined IDLab-UAntwerp (formerly MOSAIC), Steven was a new professor leading a group that included both new and experienced postdocs, as well as a new generation of Ph.D. students. Thank you, Bart S., Esteban, Glenn, Pedro, and Serena, for the shared moments during those initial years and for remaining my friends today. Over the years, many more people have been part of this journey, whether through engaging research discussions, simply sharing pleasant times during coffee breaks (from the cold server room at Middelheim to Café René at the Beacon), or supporting me at any level until today. The following is not an exhaustive list, but these are the people I can recall: Arno, Bart L., Catherine, Céline, Daniel, Dimitri, Ensar, Esra, Henrique, Inton, Jakob, Jeremy, Johan B., Johann M., Lee, Lynn, Maarten, Michelle, Najat, Nina, Patrick, Raf, Ruben, Tom, Thomas, Werner, Xhulio, and Yorick. I extend my gratitude to all of you.

I would like to extend my sincere thanks also to my colleague Adnan Shahid, who introduced me to the world of AI/ML for spectrum management and supported me during those initial steps in this field. In addition, many thanks to my colleagues during the DARPA SC2 competition from Ghent, IMEC, and Rutgers University for the great adventure and interesting talks.

On a more personal note, I would like to thank my family and closest friends. First, I want to thank my lovely wife Christine, for being the reason I started an adventure in Belgium and for standing by my side each and every day for the last twelve years, as well for giving me the most precious gift: our daughter, Helena. I love you both with my hearth!. Of course, I extend this gratitude to my in-laws, for accepting me as one of their own. I am deeply grateful for their hospitality and the way they have integrated me into their lives.

Me gustaría continuar con mi familia en Colombia, porque cada logro en mi vida es un reconocimiento a todo el amor que me han dado. Un agradecimiento especial también va para mi mamá, mi tía Gloria y mi tía Liliana por su amor y apoyo incondicional cada día de mi vida. Obviamente, este agradecimiento se extiende a todo el resto de mi familia: mi hermana, mis tíos y tías, y todos los que siempre han estado presentes. También estoy profundamente agradecido con mis abuelos Marina y Baudelino por haber sido mis mayores apoyos desde que tengo memoria. Agradezco su continuo apoyo y amor, aun cuando ya no estén a nuestro lado.

De manera similar, me gustaría expresar mi sincera gratitud hacia mis buenos amigos Paola y Nelson, y recientemente David, por recordarme las buenas vibras de mi país cada vez que hablamos. También quisiera agradecer a mi amigo Carlos D. por sus excelentes consejos (tanto profesionales como personales) que siguen siendo valiosos hasta hoy. Y no olvidar a mis mejores amigos, Ferney, Julian y Luis Eduardo, por su apoyo y amistad incondicional. ¡Muchas gracias!

Último, pero no menos importante, gracias a Dios, no solo por darme fuerza en esos momentos difíciles en mi vida cuando necesitaba un apoyo divino, sino también por darme la oportunidad de tener la mejor familia y amigos que uno pudiera desear.

May 2024, Antwerp
Miguel Camelo

"If all difficulties were known at the outset of a long journey, most of us would never start out at all." – Dan Rather

Dedicated to my grandparents Marina and Baudelino, who taught me the meaning of unconditional love. Algun dia nos volveremos a ver mis viejos.

Table of Contents

List of Figures

List of Tables

Acronyms

5G-NR 5G New Radio.

AD Autonomous Domain.

ADC Analog-to-Digital Converter.

AE Autoencoder.

AI Artificial Intelligence.

AL Alamouti.

AMC Automatic Modulation Classification.

AN Autonomous Network.

AP Access Point.

API Application Programming Interface.

APM Adaptive Power Management.

ASI Automatic Signal Identification.

ATSSS 3GPP Access Traffic Steering Switching and Splitting.

B5G Beyond 5G.

BPSK Binary Phase Shift Keying.

CAGR Compound Annual Growth Rate.

CB Control-Broadcast.

CBR Constant Bit Rate.

CBRS Citizens Broadband Radio Service.

CCK Complementary Code Keying.

CI Collaborative Interface.

CIL CIRN Interaction Language.

CIR Collaborative Intelligent Radio.

CIRN Collaborative Intelligent Radio Network.

CNN Convolutional Neural Network.

Conv Convolutional.

CP Control Plane.

CPU Central Processing Unit.

CQI Channel Quality Indicator.

CR Cognitive Radio.

CS-RL Constant-Share Reinforcement Learning.

CTW Context Tree Weighting.

CU Centralized Unit.

CWT Continuous Wavelet Transform.

DAC Digital-to-Analog Converter.

DAE Deep Autoencoder.

DARPA Defense Advanced Research Projects Agency.

DCI Downlink Control Information.

DDC Digital Down Converter.

DFN Deep Feedforward Networks.

DFS Dynamic Frequency Selection.

DL Deep Learning.

DMA Direct Memory Access.

DNN Deep Neural Network.

DP Data Plane.

DPI Deep Packet Inspection.

DRL Deep Reinforcement Learning.

DSA Dynamic Spectrum Access.

DSB Double Sideband.

DSM Dynamic Spectrum Management.

DSP Digital Signal Processing.

DSSS Direct-Sequence Spread Spectrum.

DT Decision Trees.

DU Distributed Unit.

DUC Digital Up Converter.

EHF Extreme High Frequencies.

ETSI European Telecommunications Standards Institute.

FB Feature-Based.

FCC Federal Communications Commission.

FFT Fast Fourier Transform.

FIFO First In, First Out.

FIR Finite Impulse Response.

FNN Feedforward Neural Networks.

FPGA Field-programmable Gate Array.

FSL Few-Shot Learning.

FWA Fixed Wireless Access.

GB Gradient Boost.

GL Gossip Learning.

GMFA Generalized Mean Ambiguity Function.

GP Gaussian Processes.

GRU Gated Recurrent Units.

GW Gateway.

HDLC High-level Data Link Control.

HMM Hidden Markov Model.

ICDE Intelligent Control and Decision Engine.

ImRAT Intelligent multi-RAT.

IoT Internet of Things.

IoV Internet of Vehicles.

IP Internet Protocol.

IPP Incumbent Protection Policy.

IQ In-phase and Quadrature.

IR Intelligent Radio.

ITU International Telecommunication Union.

K-NN k-Nearest Neighbours.

L1 Physical Layer.

L2 Link Layer.

L7 Application Layer.

LB Likelihood-Based.

LBW Learning By Watching.

LR Learning Rate.

LSA Licensed Shared Access.

LSTM Long Short-Term Memory.

LTE Long Term Evolution.

LTE-PDCCH LTE Physical Downlink Control CHannel.

MAC Medium Access Control.

MARL Multi-Agent Reinforcement Learning.

MC Multiple Carrier.

McF-TDMA Multi-Concurrent-Frequency Time-Division Multiple Access.

MCS Modulation and Coding Scheme.

MDP Markov Decision Process.

Meta-L Meta-learning.

MF-TDMA Multiple Frequencies Time Division Multiple Access.

MILCOM Military Communications.

MIMO Multiple-Input and Multiple-Output.

ML Machine Learning.

MLOps Machine Learning Model Operationalization Management.

MLP Multi-Layer Perceptron.

MTL Multi-Task Learning.

NB Naïve Bayes.

near-RT near-real-time.

NMS Network Monitoring Service.

NN Neural Network.

OFDM Orthogonal Frequency-Division Multiplexing.

OOBE Out-Of-Band Emissions.

PDU Protocol Data Unit.

PHY Physical Layer.

PRL Parallel Reinforcement Learning.

PSR Packet Success Rate.

QAM Quadrature Amplitude Modulation.

QL Q-Learning.

QoS Quality of Service.

QPSK Quadrature Phase Shift Keying.

QT Q-Table.

RaF Random Forest.

RAN Radio Access Network.

RAT Radio Access Technologies.

ReLU rectified linear unit.

RF Radio Frequency.

RF-MON Radio Frequency Monitor.

RFIC Radio Frequency Integrated Circuits.

RFNoC RF Network-on-Chip.

RIC RAN Intelligent Controller.

RL Reinforcement Learning.

RNN Recurrent Neural Network.

RRU Remote Radio Unit.

RSSI Received Signal Strength.

RSUPP Repeated Spectrum Usage Pattern Prediction.

SARL Single-Agent Reinforcement Learning.

SARSA State–action–reward–state–action.

SAS Spectrum Access System.

SC Single Carrier.

SC2	DARPA Spectrum Collaboration Challenge.
SCLD	Single Carrier Linear Digital.
SDAE	Stacked Denoising AutoEncoder.
SDR	Software Defined Radio.
SDRL	Supervised Deep Reinforcement Learning.
Self-SL	Self-Supervised Learning.
SG	Stochastic Game.
SGD	Stochastic Gradient Descent.
SHF	Super High Frequencies.
SIGINT	Signals Intelligence.
SINR	Signal-To-Interference-Plus-Noise Ratio.
SL	Supervised Learning.
SM	Spatial Multiplexing.
SNR	Signal-to-Noise Ratio.
SoC	System-on-Chip.
SSB	Single Sideband.
SSDRL	Semi-Supervised Deep Reinforcement Learning.
SSL	Semi-supervised Learning.
STFT	Short Time Fourier transform.
SVM	Support Vector Machines.
TC	Traffic Classification.
TD	Temporal Difference.
TDMA	Time Division Multiple Access.
TDWR	Terminal Doppler Weather Radar.
TL	Transfer Learning.
TR	Technology Recognition.
UDM	User Data Management.
UHD	USRP Hardware Driver.
USL	Unsupervised Learning.
USRP	Universal Software Radio Peripheral.

UT	User's Terminal.
VAE	Variational AutoEncoder.
VHF	Very High Frequency.
VM	Virtual Machine.
WLAN	Wireless Local Area Network.
WNIC	Wireless Network Interface Card.
WPA	Wi-Fi Protected Access.
WSN	Wireless Sensor Network.

Chapter 1
Introduction

1.1 Research Context

Beyond 5G (B5G) communication networks are expected 1) to increase data rates significantly, 2) to provide ultra-low latency and enhanced connectivity of a massive number of devices, and 3) to bring improvements in network energy efficiency. Traditionally, planning, implementation, and management of these networks and their services have been primarily performed as a manual activity with some limited automated assistance. However, it is recognized that this approach needs to be revised. It is expected that the arrival of a new generation of networking systems that **can provide network management automation capabilities such as self-configuration, self-healing, self-optimizing, and self-evolving** [1, 2].

In this context, one critical network domain that is experiencing a **radical transformation toward a fully Autonomous Domain (AD) [2, 1] is the radio (or wireless) access domain** [3, 4, 5]. Wireless technologies are becoming omnipresent and providing access to the Internet to millions of users and machines with an increased network capacity to support the ever-increasing number of devices and applications. New technologies for 5G and beyond, such as 802.11ax and 5G New Radio (5G-NR), are designed to support challenging requirements such as extreme-low latency (1-10 ms), ultra-high reliability (up to five nines), enhanced throughput (up to 20 Gbps), and flexible resource usage. However, the current deployment of 5G technologies is being limited due to a **shortage of the available spectrum** [6], which is mainly due to the obsolescence of the traditional static frequency plan based on providing access to single usage or a single user, which has granted exclusive use of a specific portion of the spectrum.

Currently, most of the allocated spectrum is underutilized, and the part mainly used by the technologies we use for daily communication is over-utilized. This exclusive-usage spectrum allocation model is being updated by several global efforts to make additional spectrum available for broadband data and **increase spectrum reuse** [7] and current progress in critical enabling technologies like Cognitive Radios (CRs) that provides a framework to adapt the radio intelligently their environment aiming to exploit old and new available spectrum so the performance of the communication network is improved as a whole. The set of techniques that are used to achieve this objective with CRs are grouped under the term Dynamic Spectrum Access (DSA), also known as Dynamic

Spectrum Management (DSM). The concept of DSM draws principles from the fields of cross-layer optimization, Artificial Intelligence (AI), Machine Learning (ML), network information theory, and game theory, among others. Examples of common tasks in DSA include link adaptation, bandwidth management, multi-user Multiple-Input and Multiple-Output (MIMO), interference management, dynamic spectrum sharing and reuse, and channel bonding, among others.

Of course, the scenarios where DSA is required are not new, e.g., the Federal Communications Commission (FCC) Spectrum Policy Task Force [8] proposed spectrum policy reform where secondary users can utilize those unused frequency bands dynamically licensed to the primary users. However, it has been the recent developments in Software Defined Radios (SDRs) platforms and AI that are making it possible to develop feasible systems to enable the **intelligent operation** of the radio networks and realizing a true CR system. Moreover, DSA is becoming a must-have functionality for future wireless communication systems driven by the increased complexity of the network's management compared with the legacy versions, which imposes a **totally different way to perform operations that were formerly performed purely through human intervention** (error pruned), mathematical optimization (too slow), or first-generation AI such as expert/ruled-based systems (deriving the rules are not anymore straightforward).

With a large portion of real-world problems having the property that it is significantly easier to collect the data (or, more generally, identify a desirable behavior) than to write the program or rules explicitly [9], the current generation of AI systems are acquiring their knowledge by extracting patterns from data, capability known as ML. In the last 5-10 years, Deep Learning (DL), which is a type of ML algorithm that uses multiple layers to extract higher-level features from the raw input progressively, has emerged as a viable approach to building AI systems that can solve complex problems in complicated, real-world environments. DL approaches achieve great power and flexibility by **learning to represent the world as a composition of concepts from a large amount of data**, where each concept is defined in relation to simpler concepts. As the primary enabler of autonomous operation is data, ML is then a crucial technology to empower the new generation of **data analytics and intelligent decision engines** that present self-dynamic capabilities to create innovative business and advanced network operations in a closed-loop fashion. Throughout this dissertation, we will use the term ML to indicate (semi-)supervised ML models that rely on complex DL architectures and are trained with large amounts of input data, such as Deep Neural Network (DNN). In the case of an ML algorithm that is not based on DL architectures or AI algorithms that are not ML, we will refer to the specific algorithm for clarity.

CRs are now a real possibility thanks to recent developments in SDR platforms. These radios offer a freely programmable computer that empowers developers to implement diverse air interfaces and signal processing functions using software within a single device. Hence, they transmit and receive data using software functionalities instead of hardware. Typically, SDRs are composed of programmable general-purposes Digital Signal Processing (DSP) or Field-programmable Gate Array (FPGA) chips. This underlying infrastructure supports versatile calibration and enables the utilization of Application Programming Interfaces (APIs) for optimizing overall performance. Moreover, SDRs play a significant role in implementing decisions orchestrated by CR. These decisions encompass dynamic adjustments to the network structure, frequency allocation, modulation schemes, coding techniques, and other pertinent parameters following the cognitive ra-

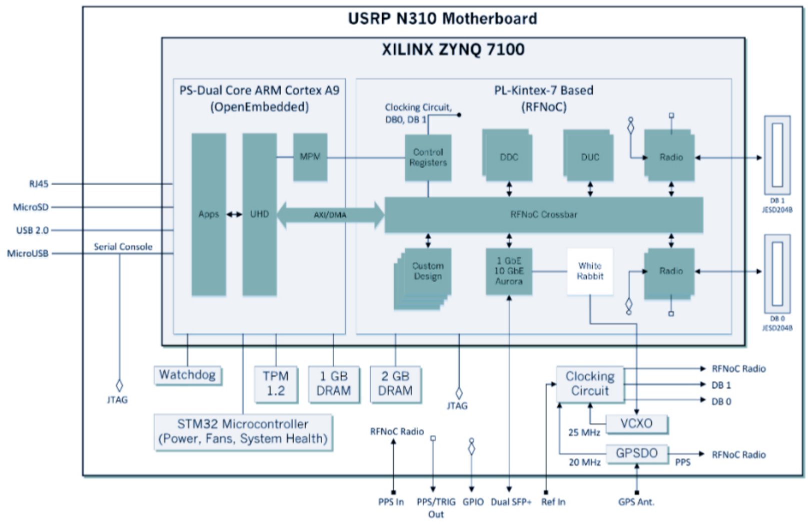


Figure 1.1: A general Ettus Universal Software Radio Peripheral (USRP) architecture based on the Ettus N310 [10].

dio cycle (spectrum sensing, cognitive management, and subsequent control actions). Through this cycle, radio nodes evolve from simply executing predefined protocols to becoming **intelligent agents that possess awareness of the radio domain**, learn user preferences, and exhibit self-programmable capabilities. Figure 1.1 shows a well-known SDR from Ettus Research.

Today, SDR is a dominant industry standard thanks to the advent of Radio Frequency Integrated Circuitss (RFICs) from companies like Analog Devices and cost-effective DSP-intensive FPGAs from companies like Xilinx, where the continue innovation in semiconductor and software technology are driving higher development productivity and more cost-effective products for radios that are now evolving to become frequency-agile intelligent communication systems based on SDRs[11]. In Figure 1.2, we can see how the shift towards SDRs has evolved over the past three decades, evolving from the initial group of industries, such as Signals Intelligence (SIGINT)), electronic warfare, test and measurement, public-safety communications, spectrum monitoring, and Military Communications (MILCOM), which transitioned from traditional hardware radio systems to SDR-based ones, towards 5G and beyond, many of them not necessarily label their industry as SDR but offering/requiring such kind of capabilities [12].

Nevertheless, how far are we from having radio networks that autonomously collaborate and reason about how to share the Radio Frequency (RF) spectrum, thereby avoiding interference and jointly exploiting opportunities to achieve the most efficient use of the available spectrum? This was the main question that the DARPA Spectrum Collaboration Challenge (SC2) asked the participants [13, 14]. As an answer to this, a collaboration among researchers from the University of Antwerp, the University of Ghent, and Rutgers University) designed and developed SCATTER, a Collaborative Intelligent Radio Network (CIRN) that provides autonomous DSA capabilities and uses efficient collaboration information across multiple radio systems to facilitate coexistence in the same spectrum band. Figure 1.3 shows a high-level view of the SCATTER Collaborative In-

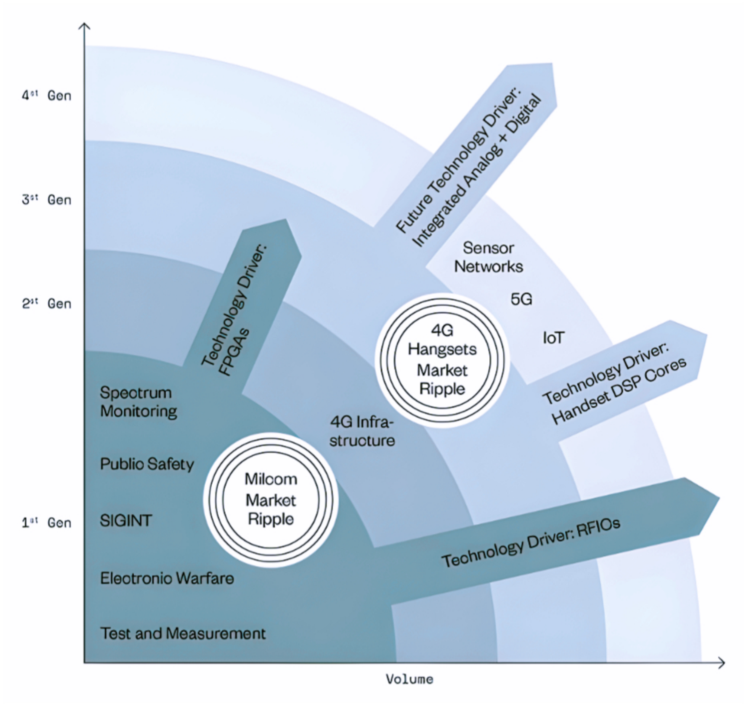


Figure 1.2: Generational evolution of SDRs and their radio industry market over time presented in [11].

telligent Radio (CIR) system architecture [15] and Figure 1.4 shows an example of a set of CIRNs sharing a spectrum. The primary user is the radar, and CIRNs collaborate to access and use the spectrum while protecting the radar efficiently. This problem is further elaborated in Chapter 4.

It is important to highlight that during the development of this dissertation, I actively participated in the design and development of the CIRN SCATTER for the SC2 competition. My contributions were primarily focusing on the control layer of the radio, designing ML-based algorithms to solve complex tasks for DSA scenarios such as blind Technology Recognition (TR) for SIGINT (see Chapter 3), active incumbent protection (see Chapter 4), flow control mechanism (see [16]), spectrum prediction (see [17]), tasks where traditional radios are not able to cope with. The algorithms for those tasks were evaluated in the Defense Advanced Research Projects Agency (DARPA) Colosseum testbed [18, 19], the testbed used for the SC2. Colosseum is the world's largest RF emulator designed to support research and development of large-scale, next-generation radio network technologies in a repeatable and highly configurable RF environment. It combines 128 SDR with a massive digital channel emulator backed by an extensive FPGA routing fabric. Accessible as a cloud-based platform, Colosseum also provides other resources to create real-time, large-scale radio environments, such as traffic generation, timing, and GPS

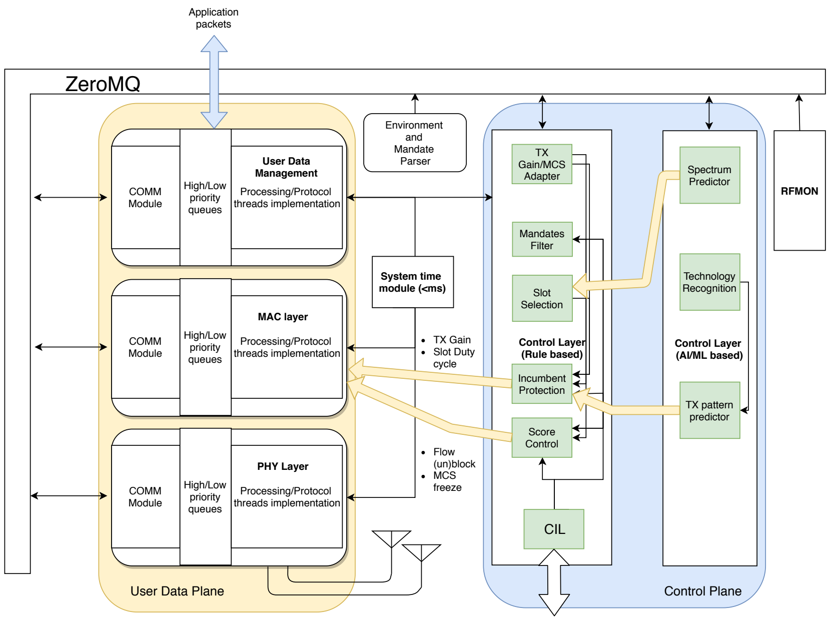


Figure 1.3: As described in [15], SCATTER CIRN has three main blocks. The first is the user data plane, where SCATTER performs data management at different layers of the radio stack. The second one is the control plane, where the rule-based and the ML-based control algorithms were running. The third one is the Radio Frequency Monitor (RFMON), which offers real-time spectrum monitoring through a continuous stream of Fast Fourier Transform (FFT) samples.

synthesis. Moreover, their real-time channel emulator provides emulation of realistic RF channel conditions for 256 radios (128 SDR, each one with two transmitters and two receivers. After the SC2, Colosseum is owned and operated by the Institute for the Wireless Internet of Things at Northeastern University, with research priorities on 5G and 6G wireless systems, AI for wireless systems, space internet, smart and connected implantable medical devices, smart cities, oceans, and ports, unmanned aerial vehicles for civil and national defense [20].

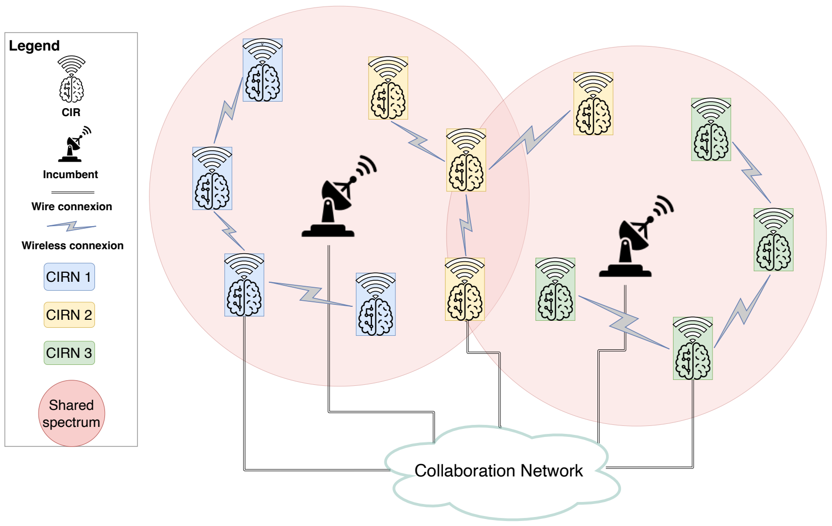


Figure 1.4: Notice that from a high-level perspective, a CIRN is a realization of an AD for radio networks since it should provide the same operational principles of 1) *autonomy*, i.e., they govern its behavior in support of business goals, 2) *abstraction*, i.e., they hide the details of domain implementation, operations and the functions of the domain elements from its users, 3) *collaboration among ADs*, i.e., cooperate based on the intent mechanism to fulfill business and customer needs [1].

1.2 Problem Statement

In the previous section, we can see that network management and optimization require advanced ML algorithms to deal with the ever-increasing complexity of network management. Moreover, these algorithms are expected to be deployed and run on heterogeneous gateways, controllers, and orchestrators across different ADs [21] to make autonomous decisions. Based on data, these algorithms will be able to automatically manage the composite mosaic of (radio) network functions and associated resources consumed by various network services (e.g., network slices) and exploited by different tenants to craft intelligence for networking.

As a candidate to enable radio access AD, CRs provide a set of closed-loop systems that manage the resources within them, where each closed control loop can observe its environment and functionality thanks to **novel cognitive capabilities, reason about those observations in the current situation, and take actions towards a set of well-defined goals**. This process allows CRs to adjust their behavior depending on the user needs and business goals when the environment changes, all with minimal human intervention.

To successfully implement and deploy radio access ADs, significant novel contributions and innovations are required in selecting, designing, and deploying ML algorithms that manage the available resources (spectrum, networking, computing, storage). This is fundamental as new challenges emerge: deciding when to use traditional management algorithms, ML-based models, or even hybrid approaches; deciding where they have to be deployed (radio vs. fog vs. edge); and how to manage their life cycle (from data harvesting to intelligent decision-making).

However, when applying ML to solve problems in networking and to run in networks, it is crucial to focus not only on the optimization problem to solve but also on **the design of the learning algorithm itself, considering both requirements when using a given technique and challenges when we want to deploy them in networks**. Until a few years ago, the dominant trend among most ML practitioners for networking was using "vanilla" versions of the ML algorithms to empower their controllers and orchestrators. However, this approach did not account for the specificities and unique requirements of networking functionalities, resulting in it not being a sensible choice. However, more recently, there has been a growing adoption of specific techniques to design tailored solutions that include fundamental aspects in design such as sustainability (energy consumption and carbon footprint), reliability (algorithms performance in unseen situations), **scalability (centralized vs. decentralized deployments)**, resource-awareness (model size vs. accuracy), **training efficiency (labels vs. accuracy)**, **communication-awareness (communication overhead vs. convergence)**, or responsiveness (real-time vs. non-real-time training) that can deliver solutions that are (semi-) optimal and suitable to run in networks.

This dissertation investigates some of these challenges from two complementary points of view. **On the radio network side**, we investigate the challenges associated with creating a novel spectrum sharing framework that is built on top of the concept of radio access ADs, i.e., the radio access network provides autonomy, abstraction, and collaboration, such that we can go beyond the state-of-the-art spectrum sharing systems that are mainly centralized and database-assisted. **On the ML side**, we investigate two main challenges when designing DL and Reinforcement Learning (RL) algorithms: 1) how to ensure training efficiency (labels vs. accuracy) when they are used to solve networking problems, and 2) how to guarantee scalability when they need to be deployed in a distributed networking infrastructure. To be more precise, the following research problems were identified concerning **AI-based algorithms to support and realize novel spectrum sharing frameworks for radio access ADs**:

[P_1] **Centralized multi-tier spectrum sharing models do not scale**: The Citizens Broadband Radio Service (CBRS) and the Licensed Shared Access (LSA) models are initiatives that provide multi-tier spectrum sharing frameworks in the reallocated spectrum. In these frameworks, **the incumbent, i.e., the primary user or technology that used the spectrum exclusively in the past**, has to be protected against interference caused by the new technologies sharing the same spectrum. For example, CBRS offers three-tiered access to users via an automated frequency coordinator, known as a Spectrum Access System (SAS), which guarantees that once a higher priority user is transmitting, the lower ones must vacate the spectrum to avoid interference. Although the multi-tier models are an initial step to mitigate spectrum scarcity via spectrum sharing, they still suffer several fundamental problems. Firstly, a single point of control or coordination can become a bottleneck as the number of users and devices increases [22, 23, 24]. Secondly, as these models rely on well-defined rules and mechanisms [25], they must be optimized to ensure fair and efficient use of the spectrum [26]. Thirdly, these models require a massive overhaul of the centralized infrastructure to support changes in environmental conditions, regulations, and policies, which can be a very time-consuming and bureaucratic task that slows down the deployment of new technologies or services that rely on shared spectrum [27]. Finally, accommodating wireless communications within some specific frequency bands (e.g., radar spectrum) requires novel spectrum-sharing paradigms since existing spectrum-sharing approaches are not designed for all coexistence scenarios [28, 29].

[P_2] **CR technologies can not share and reuse spectrum efficiently as they work in isolation and have only local information**: To achieve optimal spectrum usage, CRs will provide the capabilities to dynamically learn and apply the best policy that determines their spectrum allocation. Simultaneously, they must protect the incumbents and minimize interference with other technologies. However, adding learning capabilities to the radios is insufficient to protect the incumbent. The uncertainty about the spectrum state is higher if only local spectrum measures are used, and there is no mechanism for feedback on the radios' decisions. One potential solution to this challenge involves implementing centralized multi-tier spectrum sharing models. However, as discussed in C_1, it is important to note that these models may not be scalable, limiting their usability in highly dynamic and complex environments.

[P_3] **Traffic Classification (TC) systems are not designed to support the spectrum management decision-making processes for radio networks in a shared spectrum**: The TC task is assumed to be performed on traffic that belongs to the same network domain and over a byte/protocol representation of the packet at the Link Layer (L2) (or above). These assumptions limit the capabilities of TC systems in radio networks using shared spectrum, e.g., in unlicensed bands. The users' traffic from one radio network domain can be negatively impacted by users' traffic transmissions from other radio networks without being noticed by the TC system, as demonstrated in [30]. Contrary to a wired network, co-located radio transmissions in the same spectrum band can generate Physical Layer (L1) packets that are not detected by a receiver performing the TC task. Examples of these cases are when the transmitter is using a radio technology that cannot be demodulated and decoded by the receiver (i.e., different technology) or when the transmission can be demodulated and decoded (i.e., same technology) but the decoded traffic is already encrypted in L2 (e.g., wireless devices belong to a different radio network domain and its network is secured).

Concerning the **design of DL and RL algorithms for spectrum sharing and/or to run on a distributed infrastructure, e.g., an ad-hoc or mesh radio network**, the following research problems were identified:

[P_4] **Collecting spectrum data to train DL models for spectrum sensing is easy, but labeling is hard**: Traditionally, TR is done by domain experts, which use carefully designed hand-crafted rules to extract features from the radio signals. On the contrary, state-of-the-art approaches based on DNNs can extract features directly from raw input data and automatically perform the recognition task on those features. However, DNN-based approaches have two main drawbacks: 1) they are mainly trained in a supervised way, which implies that the whole data used for training must be labeled, and 2) their training algorithms, such as Stochastic Gradient Descent (SGD) [31], require a large amount of data to obtain a good performance [32].

Otherwise, the resulting trained model may suffer severe overfitting problems [33]. Generally, assigning labels to data can be expensive, e.g., very time-consuming, and/or some of the data might not have any labels due to incomplete knowledge of the ground truth class labels, e.g., the radio technologies to be classified are entirely unknown. On the contrary, sensing the spectrum using modern radios allows the collection of a large amount of unlabeled data at no cost.

[P_5] **DL algorithms have outperformed traditional optimization methods in solving several networking problems when considering learning-related performance metrics like accuracy, but ML practitioners tend to neglect equally important metrics related to the execution of these algorithms like inference time**: In ML, there is a prevailing emphasis on learning metrics (e.g., accuracy, loss) as the primary metric for evaluating ML models. While achieving high performance in such metrics is undoubtedly crucial from a purely ML perspective, this often neglects equally critical performance metrics that are very dependent when applied to specific domains or tasks. One particularly overlooked aspect is the inference time. In real-world applications, the inference time can be as critical as accuracy, especially in scenarios where real-time or low-latency decision-making is required. Additionally, the choice of DL architecture, whether it is a DNN, Recurrent Neural Network (RNN), or other variants, can have a high impact on inference time, scalability, and energy consumption, which can produce a model with unprecedented capacity but useless to solve the task in real-time.

[P_6] **Table-based RL algorithms are not designed to run in decentralized networking environments as they are not communication-aware**: Many closed-loop control problems are solved using RL-based algorithms as they introduce a natural way to learn a task by interacting with its environment [34]. When used to solve a task in a distributed environment, e.g., routing in radio networks [35] or driving a robot in a warehouse [36], they reduce the learning time by sharing information among multiple instances of the algorithm (i.e., multiple RL agents) [37]. An example of them is Parallel Reinforcement Learning (PRL), a type of Multi-Agent Reinforcement Learning (MARL), which reduces the learning time by leveraging parallelization [38]. In general, PRL can reduce the learning time at a rate proportional to the number of agents [39]. However, the execution time, i.e., actual time, is reduced at a slower rate due to the communication overhead between agents and the shared Q-Table (QT), the wasted learning of using overlapping search strategies, and processing and storage constraints of the infrastructure. Moreover, this communication overhead can highly reduce any efforts performed by other algorithms directly optimizing resource utilization.

1.3 Research Questions

In the previous section, we identified five problems grouped into two complementary views: spectrum sharing in radio networks and ML. This approach is natural in this dissertation as the identified problems lie in the intersection of these two major research areas. Next, a set of research questions will be defined regarding spectrum sharing techniques and the design of ML algorithms for networking and/or running on networks to investigate and develop solutions to address the identified problems above. Each one of the following research questions targets one (part) of the problem statements described in the previous section:

[Q_1] **Can we design an architecture for a spectrum-sharing system that does not require any central infrastructure to control and grant access to a shared spectrum?**: An architecture that does not require centralized authority controlling and granting access to the shared spectrum is crucial to guarantee scalability in the number of users/incumbents compared to architectures such as CBRS and LSA. Moreover, it is required to allow secondary users to use any available spectrum not used by the incumbent inside the shared spectrum band to maximize spectrum usage. However, they should not expect interference protection to maintain decentralization. Out-of-band communication should be provided among incumbent and secondary users to share information collaboratively, aiming to reduce uncertainty about the environment (radio spectrum) while providing a mechanism to augment the data used to learn and give feedback on the decisions made. To complement, radios should be intelligent and fully autonomous to dynamically learn and apply the best policy determining their spectrum allocation. Simultaneously, they must protect the incumbent and minimize interference with other technologies.

[Q_2] **Can we formulate the TR problem for DL-based algorithms such that they can use labeled and unlabeled data and design robust systems that can deal with different amounts of them?**: In general, traditional methods such as Likelihood-Based (LB) and expert Feature-Based (FB) engineering combined with pattern recognition have been outperformed by supervised DL methods in the task of TR. Supervised DL methods remove the need for expert knowledge about the environment and the signal features used for classification by using the power of automatic feature abstraction. However, it requires the whole data set to be labeled. In the case of the technologies to be recognized and the environment being entirely unknown, the labeling task becomes time-consuming and challenging. To overcome these limitations, it is required to design a DL architecture that 1) can separate the feature extraction from the classification task so we can use a large amount of unlabeled data to extract the features of the signals, and 2) reduce the use of domain expert knowledge so that only a small portion of the entire dataset has to be labeled for the classifier while still obtaining a good performance, which is not the case of supervised DL architectures.

[Q_3] **Can we design a general framework that enables the development of TC algorithms optimized for wireless networks?**: TC systems that are byte-based have limited capabilities to be functional on wireless networks. We need to move from TC systems that work at a byte representation of the packet to a generic framework for TC at the spectrum level. Moreover, it should be wireless technology agnostic (e.g., spectrum sensing + TR + packet assembly) since using an L1 packet as a classification object allows a classification at any layer for any technology as this object contains the whole information carried by the transmitted packet. In this way, the traffic generated by any other wireless device sharing the same spectrum can be monitored, detected, assembled, and classified in real-time, even if it is encrypted, belongs to a different network domain, or uses various wireless technologies.

[Q_4] **Can we provide communication-awareness capabilities to PRL agents to reduce the communication overhead while deploying them in distributed infrastructure**

without requiring any a priori information about the deployment environment?: RL-based algorithms solving a single problem collaboratively in a distributed environment can not assume that the partitioning and distributing of the state-action space is given by using an a priori domain knowledge, as it is far from realistic. It is required to design an algorithm that dynamically creates loosely coupled partitions of the QT, representing the problem space and assigning each partition to the agent exploiting it the most. If the partitioning algorithms can provide an optimal co-allocation of storage and processing, i.e., the learning agents update mainly the states in the partition assigned, then the communication cost can be minimized. Moreover, this functionality will reduce the usage of resources such as spectrum, bandwidth, and energy during the learning process, which is only achievable by providing this as built-in functionality.

1.4 Research Hypotheses

Deploying radio networks that can handle the increasing demands on network capacity of new applications and services while guaranteeing their Quality of Service (QoS) requirements will depend on the radios' capabilities to be aware of their spectrum environment, sharing and reusing it optimally. Centralized multi-tier spectrum sharing models such as CBRS and LSA can not scale in the number of users/incumbents since the need for a centralized authority controlling and granting access adds a lot of technical and administrative overhead. Simpler models like Dynamic Frequency Selection (DFS) are more straightforward to deploy than CBRS and LSA. However, they lack 1) a mechanism to exploit the spectrum not used by the incumbent and, if it does not work correctly, 2) a mechanism to incentivize and enforce the incumbent protection.

CRs are the primary candidates as a technology enabler to design a spectrum-sharing architecture that removes the need for a centralized authority to ensure scalability. However, a CR works mainly with local information obtained by its spectrum sensing functional block; therefore, the uncertainty about the spectrum state is very high. This is also increased as most of the deployments of CRs need to address the problem of how to provide feedback about the radio's decisions. Here is where collaboration plays a key role. Collaboration among CR networks and incumbents reduces that uncertainty while providing a mechanism to augment the data used to learn and give feedback on the decisions made.

A set of interconnected radios with these capabilities create CIRNs [19]. More precisely, a CIRN, i.e., AI-based autonomous radio technologies, or CIR, that exchange explicit information to solve joint problems via collaboration, can share and reuse spectrum efficiently without coordination and with the guarantee of incumbent protection. A realization of a CIRN will be a step beyond modern CR networks since CIRN can reduce the uncertainty about spectrum measures using the collaborative information and self-learning and self-adapting the radio operation parameters based on experiences, requirements to have an AD for the radio access as envisioned by the European Telecommunications Standards Institute (ETSI) Autonomous Network (AN) program [1].

CIRN could be the key breaking idea to provide the capabilities to define a scalable decentralized multi-tier model for next-generation spectrum sharing that can maximize the spectrum's use even if there are incumbents to protect. Based on this, let us state the following hypothesis:

> [H_1]: **Data-driven TR is a practical approach to enable spectrum sharing in CIRNs. Even with a small number of labeled measurements, acceptable classification performance can be achieved.**

CIRNs are empowered by ML-based algorithms such as spectrum sensing, blind TR for SIGINT, reasoning about spectrum usage, i.e., TC, and decision-making, e.g., channel selection and power control to mitigate interference. These algorithms must be designed and deployed to perform their expected tasks in the radios. DL architectures are expected to be at the heart of the algorithms empowering CIRN as they can extract features directly from raw input data and automatically perform the recognition task on those features, something that traditional approaches based on domain experts can not do anymore.

But why DL? Why not the traditional statistical ML approach? To exemplify the need for DL approaches when working on raw spectrum data, we can see that designing the features for TCs that use spectrum-based packets (or L1 packets) will be almost impossible since 1) L1 packets are modulated, coded, and sometimes encrypted before being transmitted. As a result, transmitting the same user's L2 packet may result in a very different spectrum view of the packet using either the same wireless technology, e.g., due to different Modulation and Coding Schemes (MCSs), or a different one, e.g., due to different digital multi-carrier transmission schemes. Moreover, how can an expert draft the features that can be used to differentiate a set of L1 packets belonging to YouTube vs. Twitch transmissions?

Notice also that DL architectures are challenging to train and get good performance if there is not enough labeled data. This challenge can be mitigated through the use of semi-supervised learning techniques, which combine a small amount of labeled data with a larger pool of unlabeled data to improve the learning process. By leveraging the large and easily collectible unlabeled spectrum data available, Semi-supervised Learning (SSL) could effectively reduce the dependency on extensive labeled datasets, making it a practical approach for spectrum sharing in CIRNs. These techniques enable the DL models to learn underlying patterns and features from the unlabeled data, which are then fine-tuned using the limited labeled examples.

When deploying CIRNs, their performance in distributed environments heavily relies on the efficiency of exchanging learning information among nodes in terms of latency and amount. This challenge is exacerbated when the learning algorithm is based on RL, which tends to have poor performance and low scalability in distributed environments due to often overlooked communication overhead, especially in wireless environments, which introduce higher latency. This leads us to state our second hypothesis for this dissertation:

> [H_2]: **To efficiently employ data-driven methods in CIRNs, it is crucial to leverage collaborative learning algorithms capable of operating over distributed infrastructures with minimal communication overhead, thereby ensuring practical implementation and scalability.**

Table 1.1: A summary of this dissertation and the links between problem statements, research hypotheses, and research questions.

Research	**Hypotheses**	
Questions	**H1**	**H2**
Q1	P1, P2	P1, P2
Q2	P4	–
Q3	P3, P5	–
Q4	–	P6

To summarize, Table 1.1 presents the relationships between problem statements, hypotheses, and research questions. Hypothesis H_1 envisions a next-generation spectrum sharing framework with incumbent protection based on the concept of CIRNs and data-driven enabling technologies such as label-efficient TR and real-time spectrum-based TC, directly addressing the problem statements P_1, P_2, P_3, P_4, and P_5. A positive answer to their research questions Q_1, Q_2, and Q_3 will strongly support hypothesis H_1.

Hypothesis H_2 focuses on leveraging collaborative learning algorithms to efficiently employ data-driven methods in CIRNs, addressing problem statement P_6. Although P_1 and P_2 do not directly support H_2, the scalability issues of centralized spectrum-sharing models (P_1) and the inefficiency of CR technologies (P_2) while learning in isolation highlight the need for distributed and scalable solutions. These challenges underscore the importance of developing communication-aware RL algorithms that can function effectively in decentralized environments. Therefore, a positive answer to Q_1 will support H_2 by establishing the feasibility of a decentralized, scalable framework for spectrum sharing, which is a prerequisite for implementing collaborative learning algorithms with minimal communication overhead. Consequently, addressing research question Q_4 positively will provide the evidence needed to validate hypothesis H_2.

This dissertation is composed of seven chapters. Figure 1.5 summarizes the organization of this book and shows how each research contribution, presented in detail in Section 7.1, links to different chapters, problem statements, and research questions. Following this introduction, Chapter 2 introduces some terminology and background required to navigate this book's different topics. In particular, we introduce concepts for wireless communication systems, e.g., radio spectrum and CRs, and ML, e.g., different learning techniques. The following four chapters present the four contributions described above.

Chapter 3 presents a SSL-based TR that provides the flexibility of exploiting a large amount of unlabeled data while improving the classification accuracy by using a limited labeled data set in CR. This contribution C_1 tackles research problem P_4 and acts as an enabler technology to address the research problems P_1, P_2, and P_3. Chapter 4 presents a two-tier model for spectrum sharing based on AI-based autonomous wireless radio technologies that exchange explicit information to solve joint problems via collaboration and, as a result, can share and reuse spectrum efficiently without coordination and with the guarantee of protecting legacy technologies. The realization of this framework is our contribution C_2 and addresses the research problems P_1, P_2, and P_5. Notice that P_5 was partially covered by this contribution as the original TR module was modified to achieve near real-time operation as required by the proposed spectrum sharing model.

1.5 Dissertation Outline

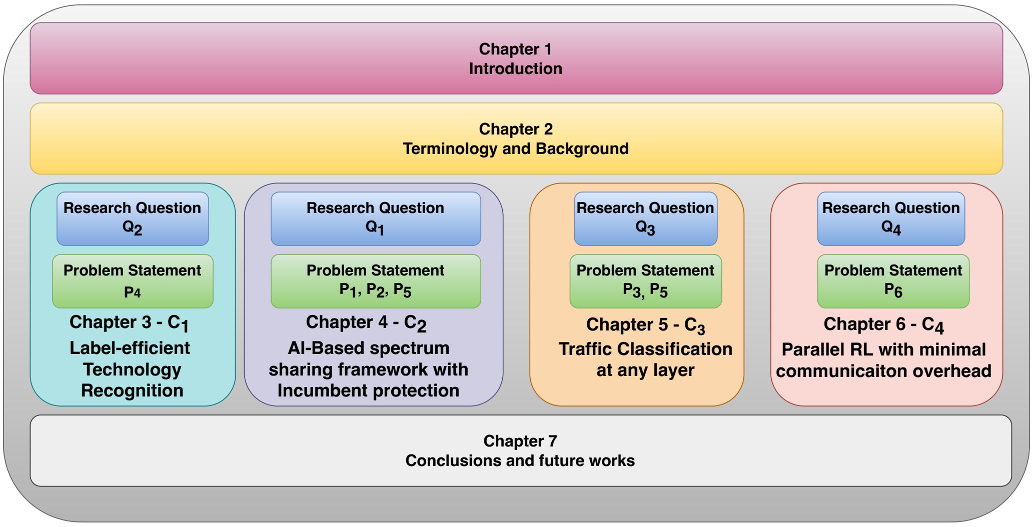

Chapter 1
Introduction
Chapter 2
Terminology and Background
Research Question Q2
Problem Statement P4
Chapter 3 - C1
Label-efficient Technology Recognition
Research Question Q1
Problem Statement P1, P2, P5
Chapter 4 - C2
AI-Based spectrum sharing framework with Incumbent protection
Research Question Q3
Problem Statement P3, P5
Chapter 5 - C3
Traffic Classification at any layer
Research Question Q4
Problem Statement P6
Chapter 6 - C4
Parallel RL with minimal communicaiton overhead
Chapter 7
Conclusions and future works

Figure 1.5: Organization of this book and relationship with problem statements, research questions, and contributions. This book is organized into seven chapters covering two functional blocks for CIRN, such as TR in Chapter 3 and TC at any layer in Chapter 5, a spectrum sharing framework based on CIRN with incumbent protection mechanism in Chapter 4, and a PRL algorithm that can run on distributed environments with minimal communication overhead in Chapter 6.

On the side of contributions to ML-based solutions to support the novel spectrum sharing framework for radio access ADs, Chapter 5 introduces and develops our contribution C_3 to address the research challenge P_3 and P_5. This contribution provides a general framework to achieve TC at any layer on the radio network together with the ML-based algorithms that perform it, including the impact of the input data and the designed architecture in terms of inference time for real-time applications. Complementing contribution C_1 in the topic of the design of DL and RL algorithms for networking and/or to run on networks, Chapter 6 proposes a novel approach that allows any table-based PRL algorithm to be deployed in distributed environments with minimal communication overhead and without requiring any a priori information about the environment, i.e., the agents are communication-aware by design. This contribution C_4 explicitly overcomes the limitations of these algorithms as described in the research problem P_6. Finally, we present our main contributions, open challenges, and future research prospective in Chapter 7.

1.6 Publications

The results obtained during this research have been presented and published in several international peer-reviewed scientific journals and conferences. Furthermore, two patent applications were submitted for contributions related to Parallel Reinforcement Learning and Technology Recognition. The following list provides an overview of the publications as first author and patent applications during the research development of this thesis.

1.6.1 O: Patent Applications

- **M. Camelo**, A. Shahid, J. Fontaine, F. A. P. de Figueiredo, E. De Poorter, I. Moerman, and S. Latré, European patent application for "A NEURAL NETWORK FOR IDENTIFYING RADIO TECHNOLOGIES" filed at the European Patent Office (EPO) on September 6, 2019, with application number EP 19195811.5.
- **M. Camelo**, M. Claeys, and S. Latré, European patent application for "EXPLORING AN UNEXPLORED DOMAIN BY PARALLEL REINFORCEMENT" filed at the European Patent Office (EPO) on October 12, 2018, and received the application number EP 18200069.5.

1.6.2 A1: Journal publications indexed by ISI Web of Science "Science Citation Index Expanded"

- **M. Camelo**, P. Soto and S. Latré, "A General Approach for Traffic Classification in Wireless Networks Using Deep Learning," in IEEE Transactions on Network and Service Management, vol. 19, no. 4, pp. 5044-5063, Dec. 2022, doi: `10.1109/TNSM.2021.3130382`. [IF 4.758].
- **M. Camelo**, R. Mennes, A. Shahid, J. Struye, C. Donato, I. Jabandzic, S. Giannoulis, F. Mahfoudhi, P. Maddala, I. Seskar, I. Moerman, and S. Latré, "An AI-Based Incumbent Protection System for Collaborative Intelligent Radio Networks," in IEEE Wireless Communications, vol. 27, no. 5, pp. 16-23, October 2020, doi: `10.1109/MWC.001.2000032`. [IF 12.777]
- **M. Camelo**, M. Claeys and S. Latré, "Parallel Reinforcement Learning With Minimal Communication Overhead for IoT Environments," in IEEE Internet of Things Journal, vol. 7, no. 2, pp. 1387-1400, Feb. 2020, doi: `10.1109/JIOT.2019.2955035`. [IF 10.238]

1.6.3 P1: Proceedings included in the ISI Web of Science "Conference Proceedings Citation Index - Sciences"

- **M. Camelo**, Camelo, T. D. Schepper, P. Soto, J. Marquez-Barja, J. Famaey and S. Latré, "Detection of traffic patterns in the radio spectrum for cognitive wireless network management," 2020 IEEE International Conference on Communications (ICC), 2020, pp. 1-6, doi: `10.1109/ICC40277.2020.9149077`.
- **M. Camelo**, A. Shahid, J. Fontaine, F. A. P. de Figueiredo, E. De Poorter, I. Moerman, and S. Latre, "A semi-supervised learning approach towards automatic wireless technology recognition," 2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN), 2019, pp. 1-10, doi: `10.1109/DySPAN.2019.8935690`.

- **M. Camelo**, J. Famaey and S. Latré, "A Scalable Parallel Q-Learning Algorithm for Resource-Constrained Decentralized Computing Environments," 2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC), 2016, pp. 27-35, doi: `10.1109/MLHPC.2016.007`.

Chapter 2

Terminology and Background

One of the main aspects of the research presented in this book is that it falls in the intersection between wireless communication systems and Artificial Intelligence (AI). To make this book self-contained, this chapter provides all the basic terminology and background information that will serve as fundamentals of all the chapters across this book. The content of this chapter is partially based on information that can be found in different seminar books and papers on the various research topics of this book as follows:

- Section 2.1: General concepts on wireless communication [40, 41], signal processing [42, 43] , Cognitive Radio (CR) [40, 44, 45, 46, 47], and Collaborative Intelligent Radio Networks (CIRNs) [15, 19, 16, 48].
- Section 2.2: Machine Learning (ML) and Deep Learning (DL) [33], Semi-supervised Learning (SSL) [49], Single-Agent Reinforcement Learning (SARL) [50], and both Multi-Agent Reinforcement Learning (MARL) and Parallel Reinforcement Learning (PRL) [37, 51, 38].

2.1 Wireless Communication Systems

Wireless communication systems have evolved from the first practical radio wave-based wireless telegraph system created by Guglielmo Marconi until very advanced Collaborative Intelligent Radio Networks (CIRNs) that can reduce the uncertainty about spectrum measures by a combination of collaborative information and self-learning techniques, which results in automatically adapt the radio operation parameters based on pure experiences with better performance compared to state of the art Cognitive Radios (CRs). In this section, we introduce some of the essential terminology and background on wireless communication systems, signal processing, and CRs that support the research presented in the different chapters of this book.

2.1.1 Radio Spectrum and Communication Systems

Open space is the medium used in wireless communications to transfer information via electromagnetic waves. The spectrum of such electromagnetic waves is organized into frequency bands to group different types of wireless systems. Table 2.1 shows the wavelengths and frequencies of electromagnetic waves as defined by the International Telecommunication Union (ITU).

Table 2.1: Radio bands of the electromagnetic spectrum defined by the ITU. Beyond ultraviolet are X-rays and Gamma-rays.

Electromagnetic waves	Acronym	Frequency	Wavelength
Extremely low frequency	ELF	30–300 Hz	10–1,000 km
Super low frequency	SLF	300–3,000 Hz	1–100 km
Very low frequency	VLF	3–30 kHz	100–10 km
Low frequency	LF	30–300 kHz	10–1 km
Medium frequency	MF	300–3,000 kHz	1,000–100m
High frequency	HF	3–30MHz	100–10m
Very high frequency	VHF	30–300MHz	10–1m
Ultra high frequency	UHF	300–3,000MHz	100–10 cm
Super high frequency	SHF	3–30 GHz	10–1 cm
Extreme high frequency	EHF	30–300 GHz	10–1mm
Tremendously high frequency (Terahertz radiation)	THF	300–3,000 GHz	1–0.1mm
Infrared rays		43,000–416,000 GHz	7–0.7μm
Visible light		430,000–750,000 GHz	0.7-0.4 μm
Ultraviolet		750,000–3,000,000 GHz	0.4–0.1 μm

Each one of these ranges provides different properties when transmitting radio waves. Radio waves at lower frequencies, e.g., below 300 MHz, tend to follow the earth's surface, while higher ones propagate in a straight line. The range from 0 (or direct current, DC) to Super High Frequencies (SHF) is commonly used for communication and other purposes like radar, radio astronomy, spectroscopy, industrial, medicine, science, and others. In contrast, the range of Extreme High Frequencies (EHF) and beyond is less used due to environmental difficulties caused by attenuation due to atmospheric effects and technical challenges in the radio transmitters and receivers, such as wave generation, amplification, detection, and modulation. Finally, over 1000 GHz, waves become optical and are traditionally limited to optical fibers.

We need to build a communication system to transfer information from one point to another. Figure 2.1 shows a block diagram of a general communication system. This system starts with the source, which generates either analog signals, e.g., speech, or digital data, e.g., multimedia. The information generated by the source is then passed to the source encoder, which generates binary data from the source. This binary data is passed through a channel encoder to improve the receiver's probability of correctly reproducing the transmitted binary data. The channel-encoded data stream is then modulated to generate the waveform transmitted across a channel such as a telephone line, an optical fiber, or a radio link over the air. Of course, a channel can be subject to different phenomena, such as noise and fading. This process is reversed at the receiver to restore the source of information.

The wireless and guided (wired) electromagnetic wave channels are the two more common transmission channels. Examples of wireless channels are the atmosphere or the free space. As these channels are open, they are usually affected by different noise sources and attenuation effects. On the other hand, a waveguide channel provides a medium in which a wave can propagate with little energy loss. A coaxial cable line and optical fiber are well-known types of guided wave channels.

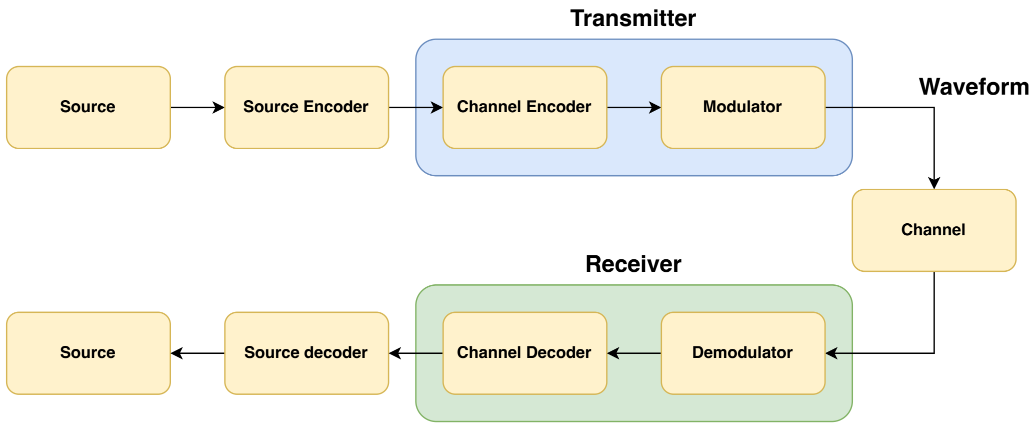


Figure 2.1: Block diagram of a general communication system.

2.1.2 Modulation and the Sampling Process

As indicated above, the modulation block generates the waveform transmitted across the channel. In this sense, modulation is the act of changing the carrier signal's properties (amplitude, phase, frequency), i.e., the electromagnetic wave that will be transmitted in a controlled way to transfer data across a channel or to obtain desired signal properties. Figure 2.2 shows how the modulation block generates a waveform from a binary data stream, and the demodulation block generates a data stream from a received waveform. It is here where Analog-to-Digital Converters (ADCs) and Digital-to-Analog Converters (DACs) play the role of the bridges between the analog (radio spectrum) and the digital domain (bits).

Once a signal has been captured in a digital form, various signal processing techniques can work on it. The conversion from a continuous-time signal to a discrete-time signal is achieved by performing the **sampling** process, where some form of an algorithm (or set of algorithms) generates a sequence of discrete values (numbers) representing the original signal. The value of the given signal in a specific time and/or space is called a **sample**, and the speed at which these samples are collected is called the **sampling frequency** or **sampling rate** f_s (i.e., one sample is taken each $1/f_s$ seconds).

To perform the conversion from analog to digital, a critical problem is determining how fast we must sample a given signal to recover the original signal faithfully later. According to the Nyquist sampling theorem, the minimum sampling frequency must be at least twice that of the highest frequency component present in the original signal. This value is also called the **Nyquist rate** B. As a result, high computing requirements exist to achieve such sampling rates over signals with a high bandwidth (e.g., signals in the Very High Frequency (VHF) band and above). It is here where complex sampling plays a fundamental role.

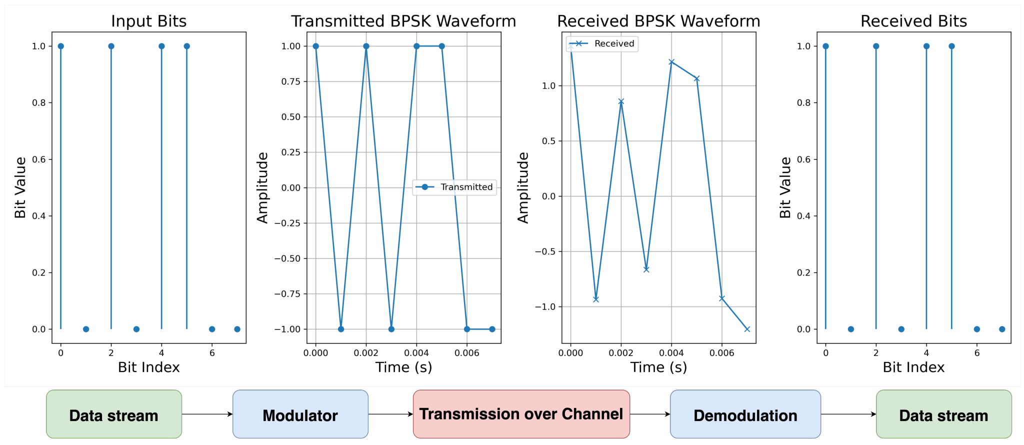


Figure 2.2: Modulation and demodulation processes as mechanisms to convert a data stream to a waveform and then recover a data stream from the waveform, respectively.

Let us consider a radio signal $X(t)$ that can be represented as follows:

$$X(t) = a(t) * \cos 2\pi f_c t + \theta(t) \tag{2.1}$$

where $a(t)$ is the amplitude modulation function, $\theta(t)$ is the phase modulation function, and f_c is the carrier frequency. Figure 2.3 shows different types of modulation based on the change of some of these functions. Now, by applying the angle sum trigonometric identity, we can decompose $X(t)$:

$$\begin{aligned} X(t) &= a(t)\cos[\theta(t)]\cos(2\pi f_c t) - a(t)\sin[\theta(t)]\sin(2\pi f_c t) \\ X(t) &= I(t)cos(2\pi f_c t) - Q(t)sin(2\pi f_c t) \end{aligned} \tag{2.2}$$

where the $I(t)$ and $Q(t)$ are termed as the **in-phase** and **quadrature-phase** modulation functions, respectively. In this representation, $X(t)$ is a **bandpass** or **low pass** signal since the $I(t)$ and $Q(t)$ functions are modulating the carrier. We can also decompose the bandpass signal into a low pass modulation function multiplied by a complex exponential (carrier) and derive a new **baseband equivalent** modulation function X_{bb}, also known as the **IQ samples**:

$$\begin{aligned} X_{bb}(t) &= I(t) + jQ(t) \\ e^{jx} &= \cos(x) + j\sin(x) \\ X(t) &= Real\{X_{bb}(t)e^{j2\pi f_c t}\} \end{aligned} \tag{2.3}$$

Figure 2.4 uses a Binary Phase Shift Keying (BPSK) modulation as an example to show the concepts described above. Note that the Nyquist rate for a signal with no frequencies $\geq B$ can be reduced to just B (complex samples/sec) instead of $2B$ (real samples/sec) as established by the Nyquist theorem. In practice, this approach is widely used in digital communication systems due to its simplicity for representing mathematical operations,

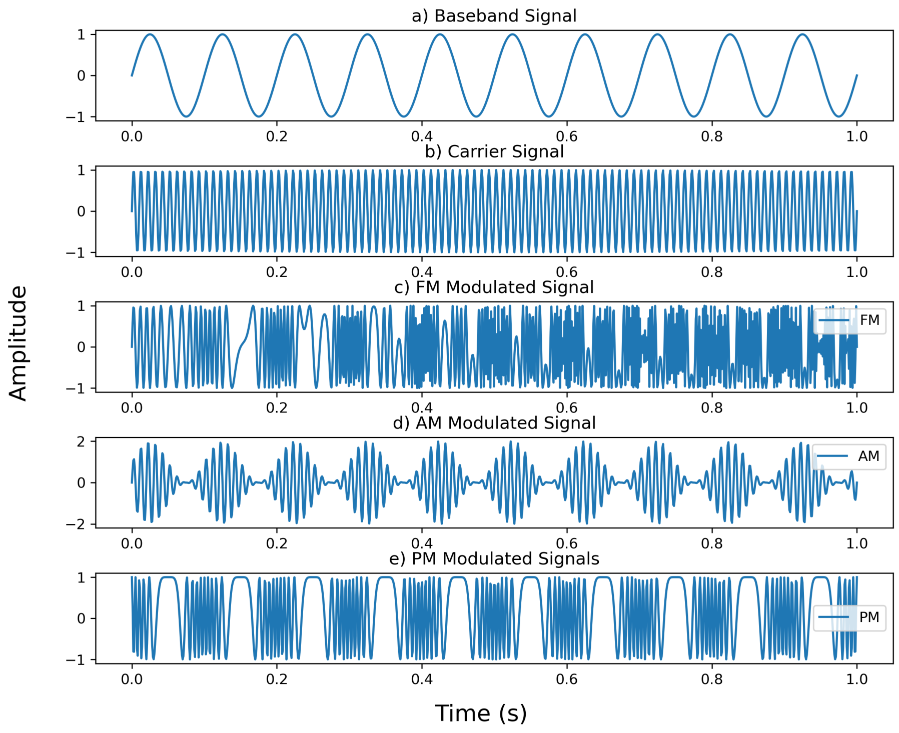


Figure 2.3: Different types of modulations. A modulated signal is a signal (b) in which one or more of its properties, such as its frequency (c), amplitude (d), or phase (d), have been varied in order to carry information from another signal (a).

its flexibility to generate any modulation scheme based on different values of $I(t)$ and $Q(t)$, and lower computing requirements for sampling as it needs half as many complex-valued samples as the original number of real samples with no information lost, and the original $s(t)$ can be recovered, if necessary, with no error. Equivalent, complex modulation allows doubling the data rate for a given signal bandwidth by using the sine and cosine as an orthogonal basis to transmit independent information on both the in-phase and quadrature-phase signals.

To finalize, let us use Figure 2.5 as a visual example to show the advantages of using complex vs. real signals for modulation. Real signals have equivalent (mirrored) positive and negative frequency spectrums, transmitting the same information in both sidebands. This type of transmission is called Double Sideband (DSB). On the other hand, complex signals have independent positive and negative spectra; therefore, each sideband can transmit unique information. This type of transmission is called Single Sideband (SSB).

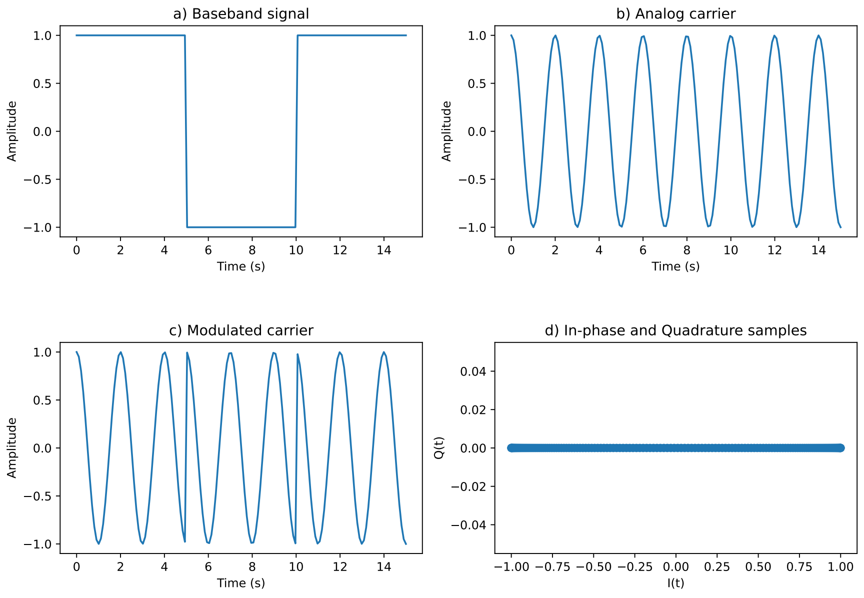


Figure 2.4: A BPSK signal is built from a) a digital baseband signal and b) an analog carrier. The resulting c) BPSK modulated carrier can be represented by its In-phase and Quadrature (IQ) values, where only the In-phase value changes.

2.1.3 Cognitive Radios

The radio spectrum is a finite resource; three main aspects govern it: location, time, and frequency. Traditionally, spectrum is allocated based on frequency bands to different radio technologies in a fixed manner. However, this approach is not scalable since 1) some bands are overcrowded while others are rarely used, and 2) it may take years before authorities change them to support new technologies or use cases. The under-utilization of the electromagnetic spectrum leads us to think in terms of *spectrum holes*, which can be defined as a band of frequencies assigned to a primary user, but, at a particular time and specific geographic location, the band is not being utilized by that user. Spectrum efficiency can be enhanced by enabling secondary users, i.e., those not currently served, to access available unused spectrum holes in the appropriate location and timeframe. In 2002, the Federal Communications Commission (FCC) Spectrum Policy Task Force [8] proposed spectrum policy reform where secondary users can utilize those unused frequency bands licensed to the primary users, also known as the **incumbent**. CR technologies have been suggested as a solution to optimize spectrum utilization by leveraging the presence of these vacant spectrum segments. Formally, we can define a CR as follows:

A *CR* is an intelligent wireless communication system that is aware of its surrounding environment (i.e., the outside world) and uses the methodology of understanding-by-building to learn from the environment and adapt its internal states to statistical variations in the incoming RF stimuli by making corresponding changes in specific operating pa-

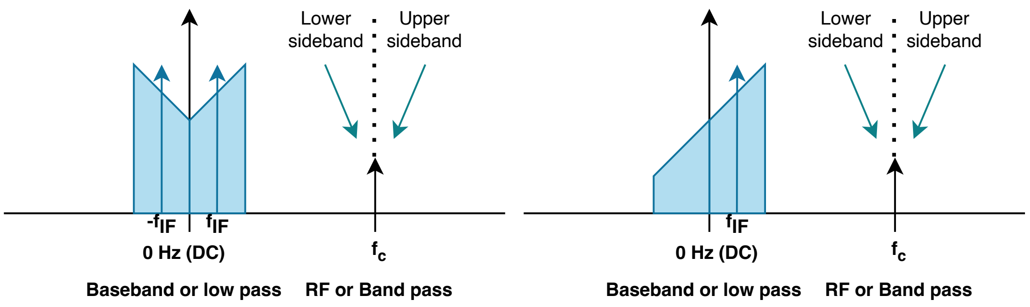


Figure 2.5: Double sideband and sideband effects when using real (left) and complex signals (right) in transmissions, respectively.

rameters (e.g., transmit-power, carrier-frequency, and modulation strategy) in real-time, with two primary objectives in mind:

- Highly reliable communication whenever and wherever is needed
- Efficient utilization of the radio spectrum

This definition highlights six pivotal concepts: awareness, intelligence, learning, adaptivity, reliability, and efficiency. Implementing this comprehensive set of capabilities is achievable today, owing to remarkable advancements in Digital Signal Processing (DSP), networking, Machine Learning (ML), computer software, and hardware. Beyond these cognitive attributes, a CR also possesses the capability of reconfigurability. This latter capacity is made possible through Software Defined Radios (SDRs), which serves as the foundation for CR systems.

SDRs ideas started in the early 1990s, and the main objective of these platforms is to provide a freely programmable computer so that developers can implement all kinds of air interfaces and signal processing functions using software in one device and be able to adapt it using the appropriate Application Programming Interface (API). Typically, SDRs are composed of programmable general-purposes DSP or Field-programmable Gate Array (FPGA) chips. The platform also provides the mechanism for adequate calibration and APIs, so it can be adapted for optimal performance. Figure 2.6 shows the high-level hardware block diagram of a well-known SDR platform provided by the Ettus Universal Software Radio Peripheral (USRP). The Ettus RF Network-on-Chip (RFNoC) framework is also a good example of a set of well-defined software APIs which allows making FPGA acceleration more accessible to build functions such as Fast Fourier Transform (FFT) and Finite Impulse Response (FIR) filters.

The importance of SDRs in CR is that these radio platforms provide the mechanism to enforce the decisions made by the CR, such as dynamically changing the network, frequency, modulation, coding, and any other parameters that are required to allow a flexible spectrum usage according to its environment. Now, CRs make decisions following the **cognitive radio cycle**, which is shown in Figure 2.7. This cycle allows transforming radio nodes from blind and static execution of predefined protocols to intelligent agents that are radio-domain-aware, learn user preferences, and can program themselves.

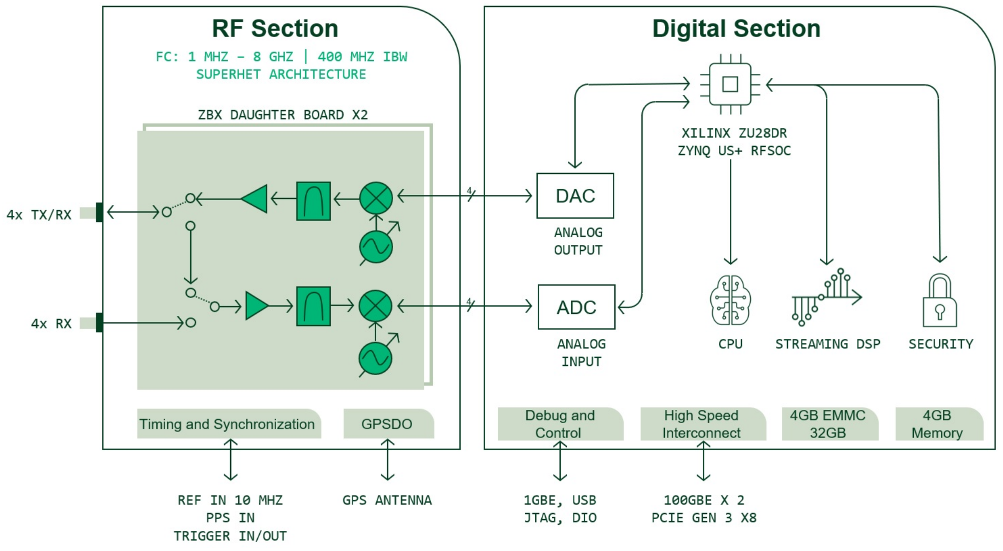


Figure 2.6: High-level hardware block diagram of the commercial-grade NI Ettus USRP X410 SDR [52].

The cognitive cycle is composed of three major components:

1. *Spectrum sensing*: Spectrum sensing refers to the task of estimating the radio channel parameters such as transmission channel characteristics, interference/noise level, spectrum occupancy, etc. Spectrum sensing is mainly done in the frequency and time domain. However, it can also be done in code and phase domains. With spectrum sensing, a CR can perform radio-scene analysis (detection of spectrum holes) and channel identification (estimation of channel-state information, channel capacity forecasting).

2. *Cognition/management*: There is a need to capture the best available spectrum to meet the user communication requirements. CRs should decide on the best spectrum band to meet Quality of Service (QoS) requirements on all spectrum bands.

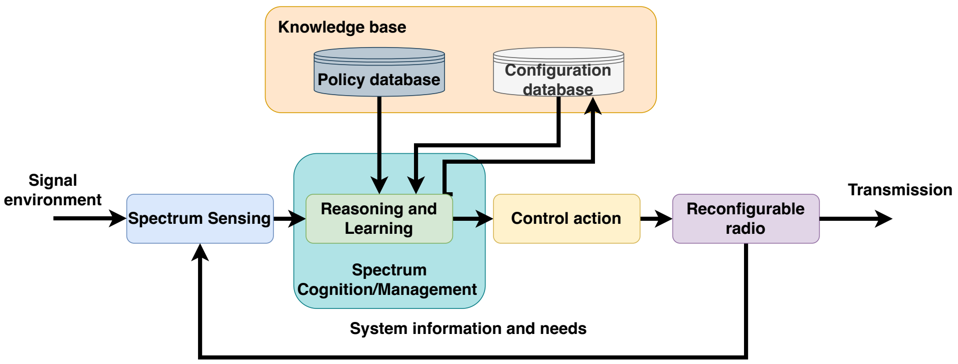


Figure 2.7: The CR cycle.

Spectrum analysis, spectrum detection, dynamic spectrum management, routing, and QoS provision are among the tasks included in the cognition/management functions.

3. *Control actions*: Finally, the control action results in executing the decisions made by the cognition/management block, such as changes in the transmit power, changes in central frequency and bandwidth, and rate control.

As CR is considered a closed-loop system, the receiver must also give helpful information to the transmitter (e.g., sensing spectrum information, quantized channel capacity, etc.). The design of the cognitive cycle is traditionally addressed as a cross-layer design problem, where the physical layer implements sensing, cognition, and adaptation, whereas the Medium Access Control (MAC) layer and above implements cooperation.

An important aspect in Figure 2.7 is that it follows the traditional CR cycle, where the reasoning and learning capabilities are part of the spectrum cognition and management block. However, novel spectrum sensing algorithms can go beyond monitoring and include learning and reasoning capabilities as well. Examples of these advanced spectrum sensing algorithms are blind Technology Recognition (TR) for Signals Intelligence (SIGINT) presented in Chapter 3 and Chapter 3, while the spectrum cognition/management block in a more traditional sense can be exemplified as the second step of the algorithm presented in Chapter 4 and communication-aware learning from Algorithms 3 and 4 in Chapter 6. This advanced integration of learning and reasoning beyond the typical CRs framework is now being used on more sophisticated CR networks such as the SCATTER CIRN, which are presented next.

2.1.4 Collaborative Intelligent Radio Networks (CIRN)

The DARPA Spectrum Collaboration Challenge (SC2) was designed to encourage researchers to develop intelligent systems that collaboratively, rather than competitively, adapt in real-time to the fast-changing, congested spectrum environment—redefining the conventional spectrum management roles of humans and machines to maximize the flow of Radio Frequency (RF) signals. The primary goal of DARPA Spectrum Collaboration Challenge (SC2) was to imbue radios with advanced ML capabilities so that they could collectively develop strategies that optimize the use of the wireless spectrum in ways not possible with today's intrinsically inefficient approach of pre-allocating exclusive access to designated frequencies.

In response to this challenge, a team of researchers from the University of Antwerp, the University of Gent, and Rutgers University designed, implemented, and validated an advanced wireless end-to-end communication system called SCATTER. The SCATTER system was designed as a CIRN in order to achieve the main goal of the entire challenge: *to encourage teams to include Artificial Intelligence (AI) elements in their networks to help with increasing spectral efficiency and coexistence with other teams and their radios, as well as to promote collaboration between teams, mainly in terms of spectrum usage and occupancy, to achieve the common goal of increasing the combined throughput of all teams*. This resulted in the architecture presented in Figure 2.8 that shows the general architecture of our software design and summarizes the interaction across modules.

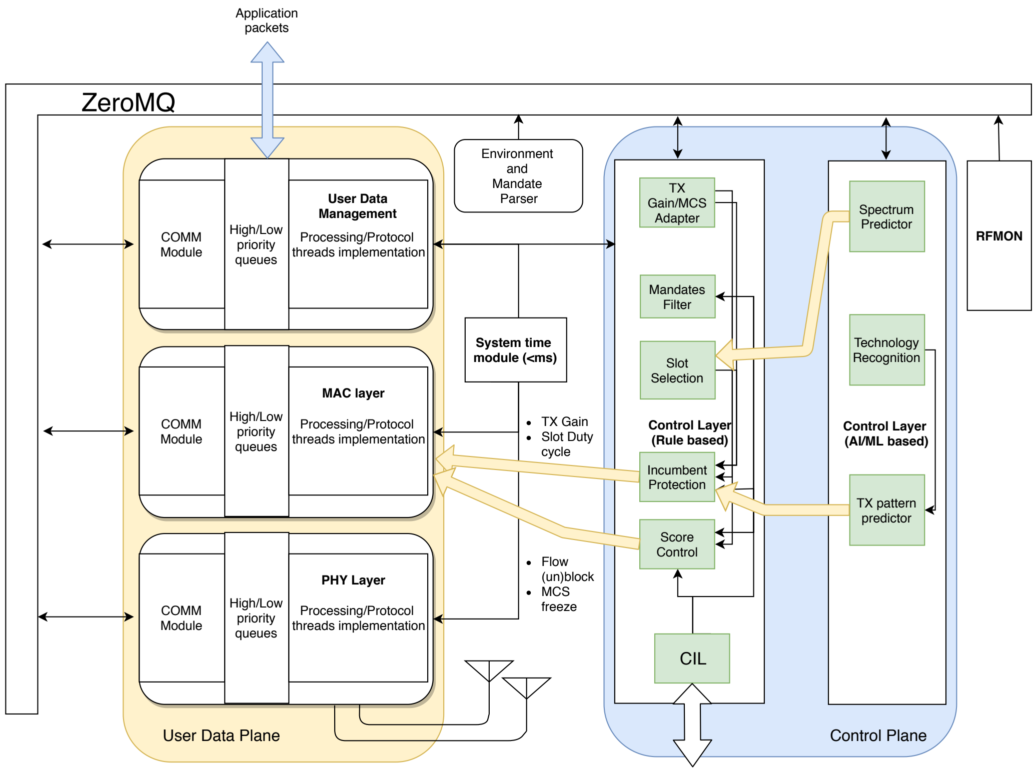


Figure 2.8: SCATTER system architecture.

SCATTER system has been designed to split every functionality within different modules, all connected through a common data bus where all information is exchanged. The primary purpose of this approach is *i*) to provide an abstraction layer that allows developers to code modules using different programming languages, choosing the most convenient one depending on the feature: while most of the modules are written in C/C++, the ML modules are written in python using third party frameworks such as TensorFlow[1] and CuPy [2] to offload matrix computations to the GPU; *ii*) to offer a fail-safe system: as each module is an individual system process, in case one of them crashes, SCATTER system can restart that process during runtime; and *iii*) to follow a plug-in approach: replacing, deleting, or adding a specific functionality is just a matter of attaching another process to the data bus. Therefore, the API for any information exchange or supported functionality is defined as a message template that two or more modules can exchange. For example, if our decision engine needs to trigger an action, regardless of the submodule that initiates it, it will result in the same action. This is especially useful for our Control Plane (CP) modules, where multiple actions are simultaneously being taken from different modules.

As shown in Figure 2.8, SCATTER has three main blocks: the data plane, the control plane, and the Radio Frequency Monitor (RF-MON) module. The Data Plane (DP) is composed of the User Data Management (UDM), MAC, and Physical Layer (PHY) modules. The CP comprises two main modules: the rule-based and the ML-based control layers. Each

[1]`https://www.tensorflow.org`
[2]`https://cupy.chainer.org`

module in the data path follows a template object architecture, offering basic input and output of data, priority buffering, and the necessary reserved space for module-specific functionalities implementation and support for communication with the CL.

The RF-MON module offers real-time spectrum monitoring in the form of a continuous stream of FFT samples. A system time module synchronizes all modules based on the system clock. Finally, some additional blocks are included to support specific requirements in the context of the SC2 challenge: the CIRN Interaction Language (CIL) module, used to interact with other networks, and the environment and traffic flow QoS parser, responsible for accepting input on required settings of the RF front-end and data traffic types along with their QoS characteristics.

2.1.4.1 SCATTER Data Plane and RF-MON

As described below, the DP is composed of three main modules: the UDM, MAC, and PHY modules. Let us describe them in more detail:

SCATTER PHY: The high-level architecture of the SCATTER PHY is depicted in Figure 2.9. The figure illustrates the different software/hardware layers composing the SCATTER PHY and the threads within each one of them. Red dashed arrows indicate data paths, while black arrows indicate control/information interaction between threads. The SCATTER PHY is implemented as a SDR and is built upon the srsLTE library [3], evolving beyond the existing Long Term Evolution (LTE) features. It communicates to a USRP X family of SDR devices[4] for pass-band conversion and transmission using the USRP Hardware Driver (UHD) software API. As can be seen in the figure, the individual PHY modules are connected to the ZeroMQ [5] (Data/Control) module, which interconnects the SCATTER PHY with the MAC layer through the ZeroMQ bus. This module manages the exchange of control and statistics messages between the SCATTER PHY and MAC layer.

Communication with the SCATTER PHY is implemented through a well-defined interface designed with Google's Protocol Buffers (protobuf)[6] for data serialization coupled with the ZeroMQ messaging library for distributed exchange of control, statistics and data messages. Implementing the ZeroMQ push-pull pattern allows the local or remote MAC layer's real-time configuration of several parameters and reading of several pieces of information/statistics provided by the SCATTER PHY. Based on the ZeroMQ logic, PHY and MAC layers can exchange control and data messages following a non-blocking communication paradigm. The SCATTER PHY was designed to be completely decoupled and independent of the MAC layer module, not posing any constraints on hardware, software, and/or programming language adopted by it.

In addition, the SCATTER PHY contains the following set of main features:

- *Orthogonal Frequency-Division Multiplexing (OFDM) waveform*: We adopt OFDM as the SCATTER PHY waveform. OFDM is a mature technology implemented in a

[3]`https://www.srslte.com/`
[4]`https://www.ettus.com/product/category/USRP-X-Series`
[5]`https://zeromq.org/`
[6]`http://code.google.com/p/protobuf/`

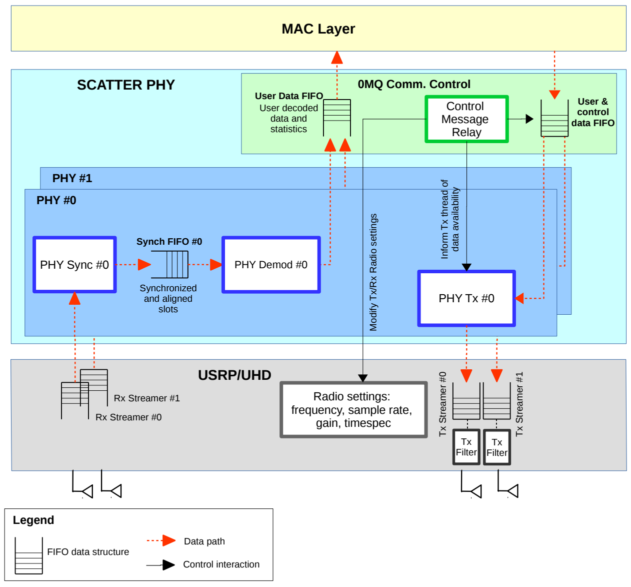


Figure 2.9: High-level architecture of the SCATTER PHY.

wide range of products due to its several advantages such as robustness to severe multi-path fading, low implementation complexity, easy integration with Multiple-Input and Multiple-Output (MIMO), simple channel estimation, etc.

- *Bursty transmissions*: with discontinuous transmissions, it is possible to improve the use of available spectrum and to coordinate its usage with other networks/radios in an opportunistic/intelligent/collaborative way.
- *Dual-Concurrent PHYs*: having two physical interfaces simultaneously transmitting and receiving at independent frequencies enables Multi-Concurrent-Frequency Time-Division Multiple Access (McF-TDMA) scheme to be implemented by the MAC layer. The ability to allocate concurrent slots allows for more flexible spectrum utilization, as vacant disjoint frequency chunks can be concurrently used.
- *FPGA-based filtered transmissions*: filtering the transmitted signal effectively minimizes Out-Of-Band Emissions (OOBE), allowing better spectrum utilization by enabling radios to have their transmissions closer to each other in the frequency domain.
- *Out-of-Band Full-Duplex operation*: both PHYs operate entirely independently, meaning that transmission (TX) and reception (RX) modules can transmit and receive at different channels, set different gains, and use different PHY bandwidths.
- *Timed-commands*: this feature allows the configuration of the exact time in the future to (i) start transmission and (ii) change TX/RX frequencies/gains.

SCATTER MAC: The SCATTER MAC protocol is based on an enhanced Multiple Frequencies Time Division Multiple Access (MF-TDMA) scheme, taking advantage of our dual concurrent PHY support and the separated RX and TX channels offered per PHY. Since two slots can be active simultaneously using PHY0 and PHY1, we can support a McF-TDMA table where two slots can be allocated for TX and two for RX at any given

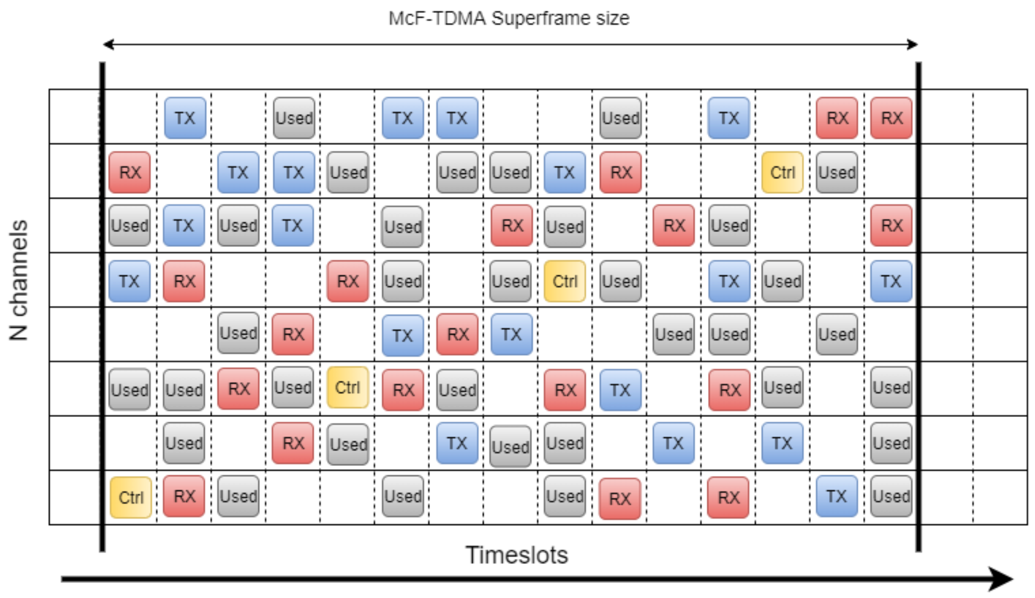


Figure 2.10: McF-TDMA slot table structure and possible states.

time slot per node. By employing a McF-TDMA scheme, the CIRN can utilize the entire offered spectrum if needed. MAC layer maintains a McF-TDMA table per node and updates it with every successful slot allocation/removal procedure. An example of a McF-TDMA table is shown in Figure 2.10.

MAC layer is also responsible for exchanging slot allocation/removal control messages and notifying neighbor nodes about any newly allocated/released time-frequency slots. Based on the latency and throughput requirements of an incoming flow, MAC layer initiates one/multiple slot allocation procedures towards the destination node to serve the incoming traffic. As a protection mechanism against slot allocation/removal procedure failures, MAC layer periodically broadcasts table status to neighboring nodes to align all McF-TDMA tables.

The MAC protocol is based on three generic operations:

- Support of a distributed slot setup protocol with any neighboring node that can perform allocation, deletion, and move of any [channelX-timeslotY] slot. Each allocated slot serves the traffic of a single link and all flows belonging to it.
- Maintain a schedule McF-TDMA table [slot, channel, type of slot, node] that keeps track of the assignment of channel-time slot tuple to transmit or receive to/from a specific node as well as Control-Broadcast (CB) slots.
- Support a semi-static Control-Broadcast slot allocation scheme. CBs are based on slotted aloha medium access if there is a large number of nodes in the network or are divided into mini-slots, where each mini-slot is used by one node, offering a Time Division Multiple Access (TDMA) like medium access. The CB scheme adapts to the channelization of the available bandwidth during boot time to avoid the possibility of CB slots interfering across the available spectrum.

Apart from the basic MAC operation, several other features exist in the MAC layer to support QoS requirements and link robustness. A double layer of buffering exists to support aggregation/fragmentation of incoming network layer Protocol Data Units (PDUs) in order to fit in the PHY PDU size (based on the running Modulation and Coding Scheme (MCS) on the link) and avoid wasted space. Also, re-transmissions are supported based on aggregated acknowledgments from the receiver, informing the sender about successfully receiving a packet. Re-transmission maximum retries are dynamically calculated per packet to ensure that the latency requirements of every packet will not be violated while data packets are dropped proactively if it is found that it is not possible to be delivered in time to the destination.

SCATTER User Data Management (UDM): Our decision engine is designed to track radio performance constantly. To achieve this, UDM reports to the CP several metrics per flow, such as the number of packets per second and average packet size. These metrics are crucial for CP to understand how close the node is to fulfilling the flow QoS requirements based on reported incoming traffic and, therefore, correctly quantify the success of our system. UDM monitors all incoming traffic flows in runtime and reports the required monitoring information to several submodules of the CP. UDM also performs buffering when specific bursty types of traffic are injected into our system, taking into account the latency characteristics of the bursty incoming traffic and reshaping the traffic flow to a Constant Bit Rate (CBR)-like flow.

SCATTER RF-MON: The RF-MON module is very important to the whole system as it gives the CP a local insight into the spectrum usage by enabling CP to access spectrum sensing measurements. These measurements are used to train ML algorithms employed to understand the environment better, optimize the spectrum usage, and cooperatively work with other networks, entirely agnostic for other network's characteristics.

RF-MON continuously monitors the whole competition bandwidth, which can go up to 40MHz in an SC2 scenario. Performing this compute-intensive task on the FPGA reduces Central Processing Unit (CPU) load. It also reduces the amount of data to be transferred from USRP to the host, as only periodic snapshots (time-averaged spectral energy) are sent to the host. This custom FPGA module, along with transmit FIR filters, was built and integrated within the SCATTER system using Ettus Research RFNoC framework as shown in Figure 2.11. As SCATTER uses dual-concurrent PHY, samples from the second radio are split into two streams, with one stream feeding RF-MON and the other stream feeding the RX1 decoding pipeline.

2.1.4.2 Control Plane (CP) and Intelligent Decision Engine

The CP of the SCATTER system holds the intelligence of the system, making use of all the knobs exposed by the MAC layer in addition to the ground truth vision provided by the RF-MON. Those are the primary enablers of our decision engine. Any required decision is taken by the sub-modules that constitute our decision engine: from MCS and transmission gain adaptation, slots scheduling, Incumbent Protection, up to traffic prioritization and score control. Let us describe the rule-based sub-modules in the CP that support our decision engine.

TX Gain and MCS adaptation: In the SCATTER system, link adaptation is controlling

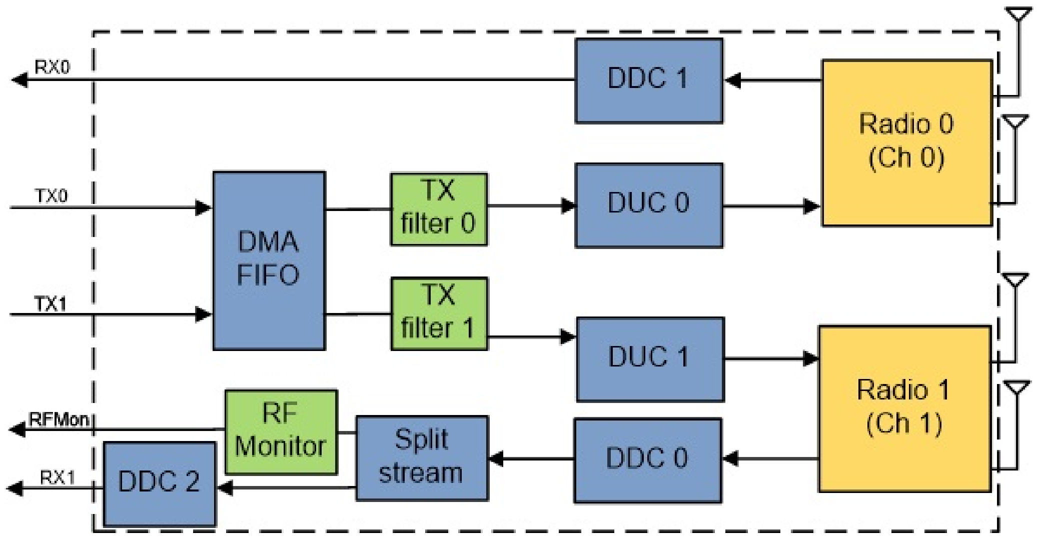


Figure 2.11: FPGA configuration with dual PHY and RF-MON. The two PHYs are composed of their own Digital Down Converter (DDC) and Digital Up Converter (DUC), and Direct Memory Access (DMA)-First In, First Out (FIFO) to transfer data between the FPGA target and host processor.

two main aspects of a link between two nodes, the MCS and the TX gain used. The link adaptation plays a two-folded role: *i*) finding optimal settings for initializing and keeping links stable with high Packet Success Rate (PSR) while not interfering, *ii*) while providing the first level of interference countering when other teams' radios interfere with a specific link. This means that the link adaptation algorithm must be able to adapt TX gain and MCS fast enough when interference is detected through PSR and link statistics like Received Signal Strength (RSSI) and Channel Quality Indicator (CQI). As all data packets are acknowledged (ack) from the receiver to the sender, the acks were used to push receiver-side statistics to the sender, thus closing a fast control loop between the sender and the receiver. This control loop is the core of the link adaptation algorithm.

Traffic Flow Management: This module is responsible for sorting the offered traffic (also known as mandates), aiming to maximize the efficiency of our system. To this end, we defined a set of benefit-cost functions for each type of traffic to calculate its benefit. The most beneficial traffic flows are selected first to be enabled in order to reach our target score. For every type of mandate, our traffic management keeps track of its status and evaluates its success. If the system cannot stabilize a flow due to bad channel conditions, the node blocks it and picks the next most beneficial mandate on the list. A detailed description of the whole traffic flow management is presented in [16].

CIL support: A node acting as gateway is the unique entity of a CIRN that is connected to CIRN Interaction Language (CIL) network. This node collects mandate performance reports and spectrum usage from each node, packs this information in a single report, and sends it to other networks. When other networks share any related information, the CIL module parses the messages and passes relevant values to sub-modules of the CP for further processing.

Slot selection: The CP selects a slot between two nodes, combining the information from multiple input sources. In order to combine this information, the input data must be normalized. To this end, we designed a slot selection system using multiple filters, representing each input source as a filter. For each filter output, the value is normalized between 0 and 1 and a slot's final "Goodness" value. The slot selection system supports two classes of filters: i) MUL-filters, where the values are multiplied by a factor and then increase the impact of the filter during the slot allocation procedure, e.g., the Incumbent-presence filter downgrades the overlapping MAC slots with the incumbent's spectrum region or the external McF-TDMA filter to make sure two nodes do not select the same slot; ii) ADD-filters, with a summation effect to the slot Goodness. The SCATTER system implements the following filters:

- External McF-TDMA filter (MUL): To ensure two nodes do not select the same slot, we implemented a filter to turn off slots used by other team members.
- Slot prediction filter (ADD): the slot prediction module, which will be described later in this section, provides a value that indicates the probability of a slot being used in the future.
- Channelizing filter (ADD): to be more predictable for other competitors, we try to allocate our slots in the same frequency channels. This filter gives a value for slots in a channel based on the number of slots already allocated in that channel.
- Incumbent Protection - presence filter (MUL): This filter marks slots that (partially) overlap with the transmission of incumbent technologies. The filter will downgrade the slots in the overlapping spectrum. As a result, the system will give priority to the slots in the non-overlapping spectrum.
- Zero-filter (MUL): In the case of active incumbents, a zero-filter is used to disable the slots in the active incumbent spectrum to achieve maximum protection. The goal is to avoid selecting slots in the spectrum where the active incumbent resides. The filter receives the slot values from the TR module. Chapter 4 provides a detailed description of the protection system implemented.
- Historic spectrum usage from other CIRN filter (ADD): This filter uses the spectrum usage information from other teams provided via CIL to lower the Goodness value of parts of the spectrum used by them. The filter will downgrade the parts of the spectrum that the other teams need to improve their performance while trading off our performance under certain limits.

During a typical slot allocation procedure between 2 nodes, the TX node, after applying all filters, selects slots with the highest Goodness from available slots and then proposes a subset of these slots to the RX node. The RX node selects the best slot out of the proposed set based on its filters and reports the selected slot back to the TX node for allocation between both nodes. In addition to the rule-based sub-modules, the CP also includes the following ML-based sub-modules that create the Intelligent Decision Engine:

Spectrum Prediction: As detailed in [17, 53], we have designed and implemented a Convolutional Neural Network (CNN) to learn and predict the usage of the spectrum. The model was trained offline with historical spectrum data as a bootstrapping step. In addition to the knowledge learned during the offline training, the model is also trained in an online fashion by accessing and using data from the RF-MON block in order to quickly learn, recognize, adapt, and predict the (possibly new) behavior of other networks, aiming to avoid interference with other transmissions in real-time. The outcome of this model is a matrix of values with the same shape as the McF-TDMA scheduling grid. These values are used as a filter in our slot selection module to enhance the view of our nodes when they select, negotiate, and allocate slots.

Technology Recognition (TR): Identifying what is in the spectrum is critical to making better decisions about accessing the spectrum. The TR module uses RF-MON data, which is framed according to our McF-TDMA scheduling grid, to discriminate different types of transmitting signals. As presented in Chapter 3, TR was initially designed to recognize different transmitting technologies and background noise based on raw IQ samples. A modified version of it for high performance and real-time classification was able to recognize five types of radio signal signatures based on FFT samples: radar, jammer, SCATTER, other teams, and noise, as presented in Chapter 4. Both versions were implemented using Deep Neural Network (DNN) architectures.

TX pattern prediction: The modified version of TR was used to provide advanced spectrum sensing capabilities in real-time to detect the incumbent's transmission. As part of a novel multi-tier spectrum-sharing framework to mitigate spectrum scarcity via spectrum sharing, SCATTER CIRN provides an autonomous incumbent protection system that allows the radios to learn and predict the time slots and frequencies where the active incumbent will transmit in the near future concerning our McF-TDMA scheduling grid. The TX pattern prediction sub-module can learn the transmission patterns of the incumbent online in time and frequency so that it can adapt to any chance of its patterns in real-time. This outcome is passed to the incumbent protection filter, marking the slots in our scheduling grid that can not be used at a given moment. As a result, our design allows efficient sharing and reuse of spectrum without centralized coordination in contrast to state-of-the-art approaches such as Citizens Broadband Radio Service (CBRS) and Licensed Shared Access (LSA). This functionality is further explained in Chapter 4.

Notice that ML-based modules in SCATTER, mainly Deep Learning (DL)-based, were a fundamental component to achieve the main objective of the Defense Advanced Research Projects Agency (DARPA) SC2. As we will see through this book, these algorithms are very promising to empower the new generation of intelligent decision engines that present self-dynamic capabilities to create innovative business and network operations in a closed-loop fashion. In the next section, we will provide the background on ML that facilitates reading this book.

2.2 Machine Learning

ML represents a transformative field of AI that has garnered widespread attention and applications across various domains. At its core, ML empowers computers with the ability to acquire knowledge and improve their performance without being explicitly programmed. A ML algorithm is an algorithm that can learn from data. Traditionally, these algorithms can be broadly categorized into three major types: Supervised Learning (SL), Unsupervised Learning (USL), and Reinforcement Learning (RL). These categories delineate the fundamental approaches through which machines can extract patterns, make predictions, and optimize decision-making processes, laying the foundation for advancements in fields as diverse as image recognition, natural language processing, and autonomous radios.

More recently, DL, which is a type of ML technique, has emerged as a viable approach to building AI systems that can solve complex problems in complicated, real-world environments. DL approaches achieve great power and flexibility by learning to represent the world as a composition of concepts, where each concept is defined in relation to simpler concepts. This approach allows computer machines to build more abstract representations from less abstract ones. Figure 2.12 shows how AI, ML, and DL relate to each other. In this section, we introduce all the essential terminology and background on AI and ML that are used to support the research in the different chapters throughout this book.

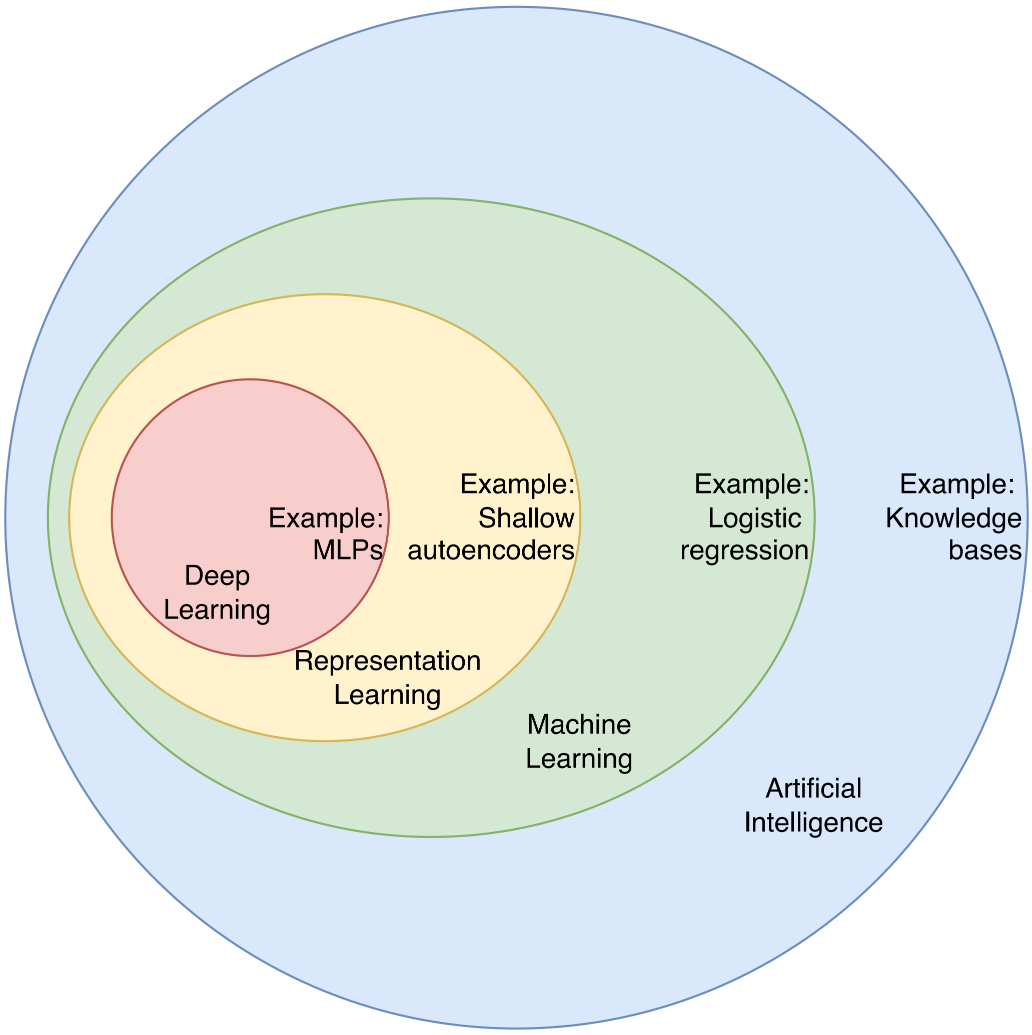


Figure 2.12: Relationships between AI, ML, and DL.

2.2.1 Unsupervised and Supervised Learning

Let $X = \{x_1, x_2, \ldots, x_N\}$ be a set of n examples (or points), where $x_i \in X$ for all $i \in [n] := \{1, \ldots, n\}$. Typically, it is assumed that points are drawn independently and identically distributed from a common distribution on X. The goal in **Unsupervised Learning (USL)** algorithms is to learn useful properties of the structure of X. Now, let $Y = \{y_1, y_2, \ldots, y_N\}$ be the set of labels or targets. In **Supervised Learning (SL)**, the goal is to learn a mapping from X to Y given a training set of pairs (x_i, y_i), where y_i is the label of the ith example x_i for all $i \in [N] := \{1, 2, \ldots, N\}$. The task is well-defined since a mapping can be evaluated through its predictive performance on test examples. When $Y = \mathbb{R}$ or $Y = \mathbb{R}^d$, i.e., when labels are continuous, the task is called **regression**. On the other hand, if y takes values from a finite set, i.e., discrete labels, then the task is called **classification**.

Roughly speaking, USL involves observing several examples of a random vector $x \in X$ and attempting to implicitly or explicitly learn the probability distribution $p(x)$ or some interesting properties of that distribution. On the other hand, SL involves observing several examples of a random vector x and an associated value or vector y, and learning to predict y from x, usually by estimating the predictive density $p(y|x)$, also known as **discriminative** SL. Alternatively, a SL algorithm can try to model the class-conditional density $p(x|y)$, also known as **generative** SL, by some USL procedure. Then, we can infer the predictive density by applying the Bayes theorem:

$$p(y|x) = \frac{p(x|y)p(y)}{\int_y p(x|y)p(y)dy} \tag{2.4}$$

2.2.2 Semi-supervised Learning

As shown in Figure 2.13, Semi-supervised Learning (SSL) is an approach that falls between USL and SL. In this family of learning algorithms, the set X is divided in two subsets $X_s = \{x_1, x_2, \ldots, x_L\}$, for which their corresponding labels $Y_s = \{y_1, y_2, \ldots, y_L\}$ are provided, and $X_u = \{x_{L+1}, \ldots, x_N\}$, for which no labels are provided such that $X = \{x_1, x_2, \ldots, x_L, x_{L+1}, \ldots, x_N\}$. Of course, if SSL still requires labeled data as in SL, is SSL still meaningful? If we assume that the distributions of examples, which will be elucidated using the unlabeled data, are relevant to the learning problem, then the answer is "yes". More formally, an important prerequisite to use SSL is that the knowledge on $p(x)$ that one gains through the unlabeled data set has to carry information that helps the inference of $p(y|x)$. If this is not the case, then SSL will not provide any improvement over SL, and it could even be the case that the use of unlabeled data degrades the prediction accuracy by misguiding the inference.

Like other learning algorithms, SSL requires certain assumptions to hold to work. As an example, SL learning algorithms also rely on assumptions such as the **smoothness assumption of SL** that indicates that if two points x_1, x_2 are close, then so should be the corresponding output y_1, y_2. Of course, without such assumptions, it would never be possible to generalize from a finite training set to a set of possible infinite many unseen cases. Let us now introduce the assumptions that need to be held by SSL to work correctly:

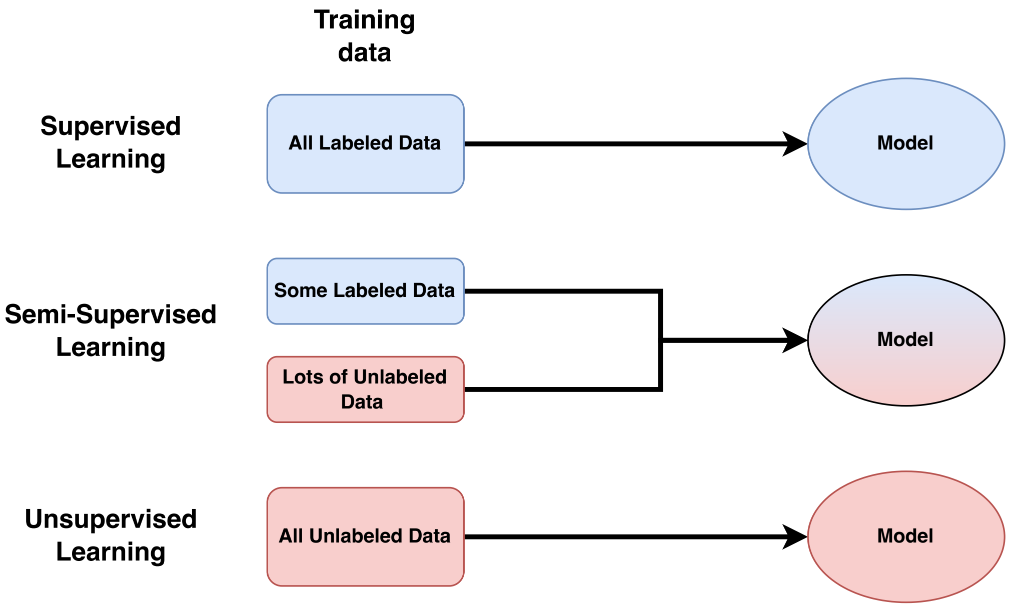


Figure 2.13: Comparing supervised, unsupervised, and semi-supervised learning.

- *Semi-supervised smoothness assumption*: If two points x_1, x_2 in a high-density region are close, then so should be the corresponding outputs y_1, y_2.

  By transitivity, this assumption implies that if two points are linked by a path of high density (e.g., if they belong to the same cluster), their outputs are likely to be close. If, on the other hand, they are separated by a low-density region, then their outputs need not be close.

- *The cluster or low-density separation assumption*: If points are in the same cluster, they are likely to be of the same class. Equivalent, the decision boundary should be like in a low-density region.

  Suppose we knew that the points of each class tended to form a cluster. This assumption indicates that unlabeled data could aid in finding the boundary of each cluster more accurately since one could run a clustering algorithm and use the labeled points to assign a class to each cluster. This is reasonable based on the sheer existence of classes: if there is a densely populated continuum of objects, it may seem unlikely that they were ever distinguished into different classes.

- *Manifold assumption*: The (high-dimensional) data lie (roughly) on a low-dimensional manifold.

  A well-known problem of many statistical methods and learning algorithms is the so-called curse of dimensionality. It is related to the fact that volume grows exponentially with the number of dimensions, and an exponentially increasing number of examples is required for statistical tasks such as the reliable estimation of densities. A related problem of high dimensions, which may be more severe for discriminative methods, is that pairwise distances tend to become more similar and thus less expressive. If the data lie on a low-dimensional manifold, the learning algorithm can essentially operate in the space of the corresponding dimension, thus avoiding the curse of dimensionality.

- *Vapnik's principle*: When trying to solve some problem, one should not solve a more complex problem as an intermediate step.

  A problem related to SSL was introduced by Vapnik as the so-called **transductive learning**[54]. In this problem, one is given a (labeled) training set and an (unlabeled) test set. The main idea of this approach is to perform predictions only for the test points. This contrasts to **inductive learning**, where the goal is to output a prediction function defined on the entire space X. Now, suppose label predictions are only required for a given test set. In that case, transduction can be argued to be more direct than induction: while an inductive method infers a function $f : X \to Y$ on the entire space X, and afterward returns the evaluations $f(x_i)$ at the test points, transduction consists of directly estimating the finite set of test labels, i.e., a function $f : X_u \to Y$ only defined on the test set. Note that transduction is not the same as SSL: some SSL algorithms are transductive, but others are inductive.

We can see across the literature that most SSL algorithms can be seen to correspond to or implement one or more of these assumptions. In general, SSL can be divided into four main classes corresponding to the previous four assumptions: change of representation, low-density separation, graph-based methods, and generative models.

2.2.3 Reinforcement Learning and its Parallelization

Notice that some machine learning algorithms do not just learn from a fixed data set. This is the case for Reinforcement Learning (RL) algorithms that interact with an environment, so there is a feedback loop between the learning system and the information it uses to learn.

2.2.3.1 Single and Multi-Agent Reinforcement Learning

RL is a branch of Machine Learning (ML) that allows an entity, called the learning agent, to solve a problem by trial and error. Problems in RL are formulated as Markov Decision Processs (MDPs). A finite MDP is a tuple $\{S, A, T, R\}$ where S is the set of environment states, A is the set of agent actions, both finite, and $T : S \times A \times S \to [0, 1]$ and $R : S \times A \times S \to \mathbb{R}$ are the transition probability and the reward functions, respectively. In this model, the state $s \in S$ describes the environment at each discrete time step t. After observing the state $s_t \in S$, the agent takes action $a_t \in A$, and the environment changes to a new state $s'_t \in S$ according to the function T. Simultaneously, the agent receives a reward $R_r \in \mathbb{R}$ that evaluates the immediate effect of the action taken according to the function R. However, the reward says nothing about the long-term effects of the action. Figure 2.14 shows how an agent interacts with the environment in the framework of MDPs.

One kind of MDPs assumes that there are terminal states, i.e., states that once an agent is there, it can not leave it. Under this assumption, the learning process is separated into episodes, which start in an initial state and end in a terminal state. A policy $\pi : S \to A$ determines an agent's action at any state. The primary objective of the agent is to find a

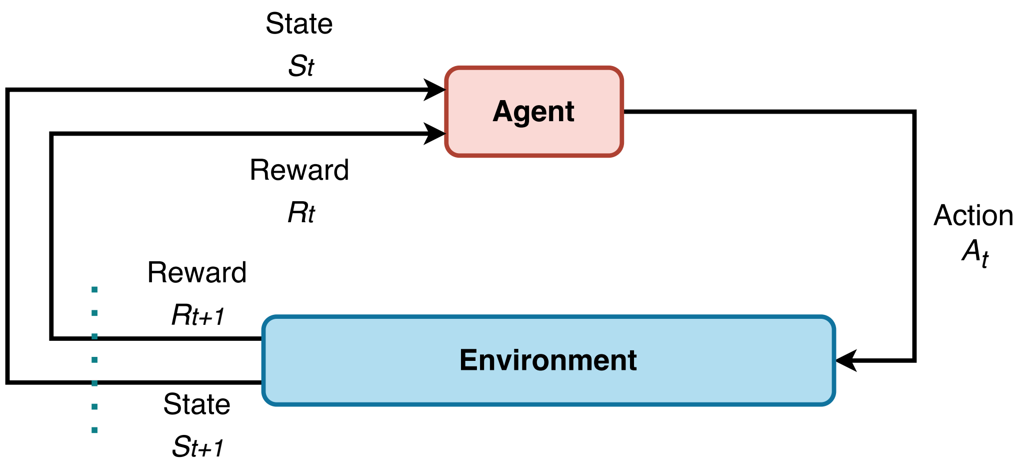


Figure 2.14: An agent-environment interaction in a MDP.

policy that maximizes, for every $s \in S$, the state-action value function $Q^{\pi} : S \times A \to \mathbb{R}$, also known as the Q-function:

$$Q^{\pi}(s,a) = E\left\{ \sum_{t=0}^{\infty} \gamma R_t \middle| s = s_0, a = a_0, \pi \right\} \tag{2.5}$$

Where the discount factor $\gamma \in [0,1)$ is used to bound the sum, which otherwise might grow unbounded. The Q-function represents the reward accumulated by the agent in the long run for taking action a in state s and following the policy π. To obtain the optimal Q-function, $Q^*(s,a)$, equation (2.5) can be written as a Bellman optimally equation as follows.

$$\begin{aligned} Q^*(s,a) = \sum_{s \in S} T(s,a,s')\big[R(s,a,s') + \\ \gamma \max_{a'} Q^*(s',a')\big] \end{aligned} \tag{2.6}$$

Intuitively, the Bellman optimality equation allows expressing the expected total reward for an agent by taking a in s in terms of the optimal value from the next state. After computing Q^* for every pair $(s,a) \in S \times A$, the optimal policy, also known as greedy, can be computed as $\pi^*(s) = arg\max_{a'} Q^*(s',a)$.

Depending on the availability of the MDP, RL algorithms can be divided into model-based and model-free approaches [50]. Model-based RL algorithms find optimal policies by interacting with a model of the environment, which is either provided as an MDP or estimated from data (sampling). However, these learning algorithms have only practical use in problems with small state-action space. On the contrary, model-free RL algorithms do not require the MDP and learn through actual experience with the (real) environment.

Temporal Difference (TD) learning is a family of model-free RL algorithms that learn directly by interacting with the environment and updating estimates of Q^{π} based on other learned estimates. TD methods can be divided into On-Policy or Off-Policy according to how the actions are selected during learning. On-Policy TD methods learn the policy used to take action. Off-Policy TD learn one policy, called the target policy, while taking actions following another policy, called the behavior policy. In other words, Off-Policy TD algorithms separate exploration via the behavior policy that selects actions and generated behaviors from the control via the target policy whose Q-function is the objective of the learning process. Q-Learning (QL) [55] is an example of the Off-Policy TD RL algorithms that is the basis of many Single-Agent Reinforcement Learning (SARL) and Multi-Agent Reinforcement Learning (MARL) algorithms [56, 37]. QL is one of the most efficient and simpler RL algorithms that updates its estimates $Q(s,a)$ based on sample $< s, a, s', r >$, at time t, with:

$$Q(s_t, a_t) + \alpha \cdot \left[r_{t+1} + \gamma \cdot \max_{a'} Q_t(s', a') - Q_t(s_t, a_t)\right] \tag{2.7}$$

A generalization of the MDPs with multiple agents is defined by the Stochastic Game (SG) framework. A SG is a tuple $\{S, A_1, \dots, A_n, T, R_1, \dots, R_n\}$, where n is the number of agents, S is the set of environment states, A_i, $i \in [1, n]$, are the set of actions available to agent i, which generate the joint action set $\mathbf{A} = A_i \times \cdots \times A_n$, $T : S \times \mathbf{A} \times S \longrightarrow [0, 1]$ is the transition probability function, and $R : S \times \mathbf{A} \times S \longrightarrow \mathbb{R}$ is the reward function.

Depending on different choices in this framework, several types of MARL problem descriptions and possible solutions can be derived. In our case, we are interested in fully collaborative stochastic games with independent learners and a single shared policy [38]. This model, also known as Parallel Reinforcement Learning (PRL), assumes that all agents are independent, i.e., they only observe their local actions, they have the same reward function $R_1 = \ldots = R_n$, to maximize the joint return and update a unique policy. These assumptions reduce the problem to an MDP, whose action space is the joint action space of the SG. Assuming independent agents is more practical since the observability of joint actions is hard to meet in real environments [57].

2.2.3.2 The Parallel Reinforcement Learning Problem

Traditionally, SARL algorithms are sequential: an agent perceives its current state $s \in S$ by sensing the environment and then selects an action $a \in A$. As a result of the selected action, the agent moves to a new state $s' \in S$ and receives a reward $r \in R$. To learn the optimal policy, the agent interacts with the environment for a finite number of steps. The time that requires the agent to learn can be measured in two ways: learning and actual time.

Definition 2.2.1. Learning time is the number of algorithm iterations that an agent requires to find an optimal policy.

Definition 2.2.2. Execution time is the actual time that an agent requires to find an optimal policy.

In SARL, the agent requires a long learning time and a tabular representation of the q-values to guarantee convergence to an optimal policy. PRL aims to reduce the learning time of SARL algorithms by increasing the number of agents solving the task and sharing their experiences. Kretchmar introduces the PRL problem by using the well-known n-armed bandit task and shows the complexities of sharing information among multiple

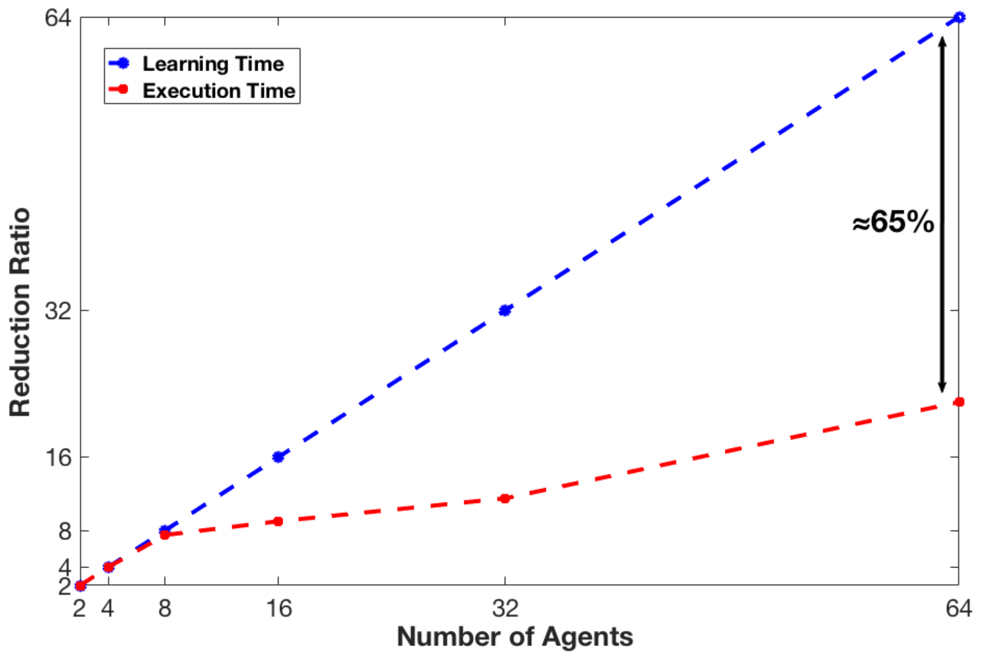


Figure 2.15: Reduction rate in the learning and execution time of the Constant-Share Reinforcement Learning (CS-RL) [38] algorithm solving the Cliff problem (see example 6.6 in [50]) in a grid of size 100x100.

RL agents running in parallel and without using a common shared Q-Table (QT) [58]. More specifically, the learning challenge in PRL is the problem of designing strategies for sharing and merging knowledge among the agents.

In previous work, Whitehead proved that the learning time, in terms of the number of training episodes, can be reduced proportionally to the number of agents solving the task if they update the same policy/QT [39]. However, in both cases, the execution time, in terms of actual time, is reduced at a rate lower than n due to the added communication overhead and the wasted learning of using overlapping search strategies [58, 38, 39]. Therefore, a scalability problem in PRL requires developing strategies that simultaneously reduce both the learning and execution time while increasing the number of agents solving a given task. Formally, we can define the communication cost as follows:

Definition 2.2.3. Communication cost (hardware-dependent) In distributed environments, RL agents running in parallel are interconnected via a communication network. The communication cost of a PRL algorithm with n agents is defined as the ratio between the cumulative (actual) time spent by exchanging messages among agents and its execution time. This measurement has no units.

Definition 2.2.4. Communication cost (hardware-independent) The communication cost of a PRL algorithm with n agents is defined as the ratio between the cumulative number of messages exchanged by the agents over the network and its learning time. This measurement has messages/iteration as a metric unit.

Figure 2.15 shows the reduction ratio in the learning and the execution time, i.e., learning/execution time of n agents concerning a single agent, of a naive implementation of the CS-RL algorithm [38] solving a RL problem using up to 64 agents on a unique server with 20 cores, i.e., no communication overhead due to a computer network but inter-processes messaging in the server. While the learning time is reduced linearly to the number of agents solving the problem, the execution time was reduced to 65% less with 64 agents, which will be much worse in a fully distributed scenario.

While the learning time in PRL depends on the strategy and frequency of sharing experiences between agents, the execution time depends on the strategies to optimize the available processing, storage, and communication resources. While distributed and large-scale infrastructures, such as the Internet of Things (IoT), can provide the processing and storage to guarantee the scalability of the PRL algorithms, it is required to design new strategies that minimize the cost of exchanging information among agents over the network. At the same time, these strategies should be independent of the type of RL agent to provide flexibility at the moment of selecting the right RL algorithm behind the IoT application.

2.2.4 Neural Networks and Deep Learning

Feedforward Neural Networks (FNN), also known as Deep Feedforward Networks (DFN), or Multi-Layer Perceptrons (MLPs), is the most perfect example of **Deep Learning (DL)**. A FNN aims to approximate some functions f^*. In the case of a classifier, $y = f^*(x)$ maps an input example x to a label y. It defines a mapping $y = f^*(x;\theta)$ and learns the value of the parameters θ, resulting in the best function approximation.

These models are called **feedforward** as information flows through the evaluated function f from x, moving across the intermediate computations required to get f, and then to the output y. In FNN, there are no **feedback** connections where the (intermediate or final) output of the model is fed back into the model itself. An example of a well-known specialized FNNs is the **Convolutional Neural Network (CNN)**, which is used for object recognition directly from images. Another example of an extension of FNNs to include feedback connections is called **Recurrent Neural Network (RNN)**, which is now used for processing temporal sequences and provides the means to power many natural language applications by allowing modeling relationships between sequences and other sequences rather than just fixed inputs.

The term **networks** is associated with FNN since they typically are represented as a composition of many different functions whose relationships can be described via a directed acyclic graph. For example, we can have three functions $f^{(1)}$, $f^{(2)}$, and $f^{(3)}$ connected sequentially to form $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$, where $f^{(1)}$ is called the **first layer** of the network, $f^{(2)}$ the **second layer**, and so on. The first layer of the model is called the **input** layer, while the last layer is called the **output** layer. The number of layers that are part of such composition gives the **depth** of the model, terminology where DL and **Deep Neural Network (DNN)** emerge from.

The objective of training a FNN is to drive $f(x)$ to match $f^*(x)$. The training data provides noisy, approximate examples of $f^*(x)$ evaluated at different training points, where each x is associated with a label $y \approx f^*(x)$. Note that the training examples specify what the output layer must do at each point x presented to the input layer, i.e., evaluating

f^* at x should produce a value close to its associated y. However, the behavior of the other layers is not directly specified by the training data, i.e., the learning algorithm must decide how to use the intermediate layers to implement the best f^* such that it can output each expected output. However, the training data does not say what each layer should do. This is why the intermediate layers are also called **hidden** layers. Traditionally, each hidden layer is vector-valued, so the dimensionality of these hidden layers determines the **width** of the model. Figure 2.16 shows a simplified version of a DNN model.

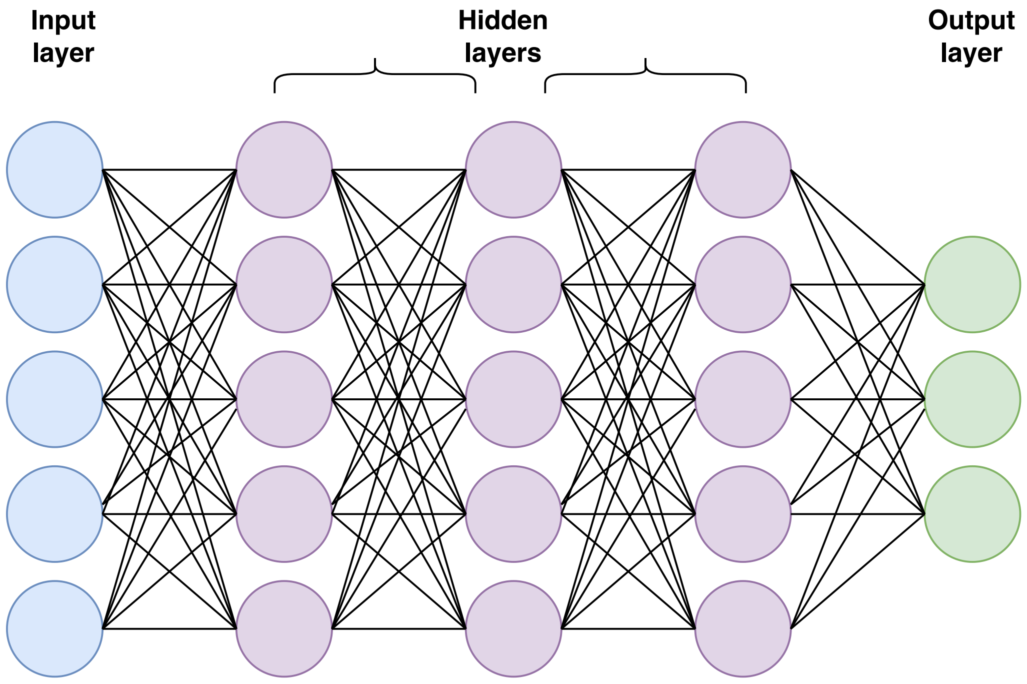


Figure 2.16: A DNN model showing the input, hidden, and output layers.

Finally, FNN are called **neural** because they are loosely inspired by neuroscience. Each element of the hidden layer may be interpreted as analogous to a neuron in the brain. In this way, instead of thinking of a layer as a vector-to-vector function, a layer can be seen as parallel **computing units** able to act in parallel, where each one represents a vector-to-scalar function. Again, the analogy with the brain is that each unit resembles a neuron because it receives input from many other units and computes its activation value. However, modern Neural Networks (NNs) research is guided by disciplines like mathematics and engineering, where the goal is not to mimic the model of the brain but to create function approximation machines that are designed to achieve statistical generalization, models that occasionally can get some insights from what we know about the brain. With this goal in mind, the activation functions are commonly selected to be nonlinear so that by the composition of multiple layers, in theory, we could approximate any function [59].

Nevertheless, why DL is being the most used ML approach to solve many complex Artificial Intelligence (AI) tasks? Traditionally, simple ML models can solve complex tasks by designing the correct set of features to extract what represents the input data for that task. The main goal behind designing features, or algorithms for learning features, is to separate the factors of variation that explain the observed data. For example, a helpful feature of a singer's vocal range is the shape and structure of each individual's vocal folds. It, therefore, gives a vital clue as to whether the singer is a bass, tenor, alto, or soprano. However, knowing what features should be extracted for many learning tasks is challenging. For example, given a picture, how can we describe what exactly a cat is in terms of pixel values? Many characteristics make this task difficult. For example, what is the shape of a "general" cat? How do we geometrically define it? Moreover, this is even more complicated as there are different types of cats, different environments where the cat may be while capturing the picture, and there are different visual effects that may impact the picture itself. In other words, a significant source of difficulty in many real-world AI applications is that many factors of variation influence every single piece of data we can observe.

Manually designing features for a complex task requires much human time and effort, from weeks to months (for very specialized and easy problems), up to decades for an entire community of researchers. ML can be used to solve this problem by not only learning the mapping between inputs and outputs but also the representation itself. This approach is called **Representation Learning**, which often performs much better than can be obtained with hand-designed representations. Moreover, it can even be the case that obtaining such representation is as difficult as solving the original problem.

DL solves this central problem in representation learning by introducing representations that are expressed in terms of other, more straightforward representations. Therefore, DL allows the computer to build complex concepts out of simpler concepts. In this way, DL system can represent the concept of a cat by combining simpler ideas such as corners and contours, which are defined in terms of edges, and so on. A perfect example of a DL model for learning representations is the **Autoencoders (AEs)**, which is a combination of an **encoder** network that converts the input data into a different representation and a **decoder** network that transforms the new representation back into the original format. AE are trained to preserve as much information as possible when the input is run through the encoder and then the decoder. Still, they are also trained to make the new representation have various nice properties, e.g., remove noise from the input example.

In practice, building DNNs requires addressing several design decisions needed to deploy them. First, training a DNN requires choosing the **optimizer**, which is the algorithm used to change the attributes of the DNN to create the best f^*, the **cost function**, which is used to measure the performance of the model finding a relation between the input and output, and the form of the output units, which is linked to the type of ML task that is being solved. In this part, gradient-based learning algorithms will play a role as they allow DNNs to compute the gradients of complicated functions. The back-propagation algorithm and its modern generalizations can be used to calculate these gradients efficiently. Secondly, we must also design the architecture of the DNN, which includes how many layers the network should contain, how these layers should be connected, how many units should be in each layer, and what type of units. Finally, we need to choose the activation functions that will be used to compute the values of the layers. DL can safely be regarded as the study of models involving more composition of learned functions or concepts than traditional machine learning.

Chapter 3

Label-Efficient Automatic Wireless Technology Recognition

The content of this chapter has been partially published in:

- M. Camelo, A. Shahid, J. Fontaine, F. A. P. de Figueiredo, E. De Poorter, I. Moerman, and S. Latré, "A semi-supervised learning approach towards automatic wireless technology recognition," 2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN), 2019, pp. 1-10, doi: `10.1109/DySPAN.2019.8935690`.
- M. Camelo, A. Shahid, J. Fontaine, F. A. P. de Figueiredo, E. De Poorter, I. Moerman, and S. Latré, European patent application for "A NEURAL NETWORK FOR IDENTIFYING RADIO TECHNOLOGIES" filed at the European Patent Office (EPO) on September 6, 2019, with application number EP 19195811.5.

This chapter presents the Technology Recognition (TR) module, an Artificial Intelligence (AI)-based functionality that provides advanced capabilities for spectrum sensing and is used as an enabling technology for the research presented in Chapters 4 and 5. From the wireless domain perspective, TR enables next-generation Cognitive Radios (CRs) to sense the spectrum and reason what kind of technologies are transmitting, even if the identified technologies are a-priori unknown. This capability is not part of traditional approaches that require knowledge about specific features of the radio signals to identify them. On the AI side, where most of the novelty is present, the proposed approach uses Semi-supervised Learning (SSL) to tackle one of the well-known challenges using Deep Learning (DL) models that are trained in a supervised way: high classification accuracy requires a considerable amount of labeled data.

3.1 Introduction

According to Cisco, global mobile data traffic would increase seven-fold between 2017 and 2022, from 12 to 77 Exabytes (EB) per month, thereby increasing the Compound Annual Growth Rate (CAGR) by 46 percent [60]. Recently, data has shown that the total global mobile data traffic, excluding traffic generated by Fixed Wireless Access (FWA), reached 93 EB per month at the end of 2022 and is projected to grow by a factor of 3.5 to reach 329 EB per month in 2028 [61]. Such increased traffic cannot be accommodated even by spectrum extension. In this regard, the radio spectrum becomes valuable since many wireless technologies share the same spectrum. However, most spectrum is underutilized, while radio technologies suffer poor performance due to interference in overutilized ones. In such a coexisting environment, Cognitive Radio (CR) systems will play a significant role in solving this problem [62]. Therefore, providing intelligence to the radios so they can reason about using and sharing the available spectrum efficiently and defining new spectrum access strategies is of fundamental importance.

Inside CR, Dynamic Spectrum Access (DSA) provides the capability to share the spectrum among multiple technologies opportunistically. One critical problem that DSA faces is identifying if some technology is accessing the same spectrum and then taking appropriate measures to combat the performance degradation due to interference. This problem is termed the Technology Recognition (TR) problem, which refers to identifying radio signals of wireless technologies without requiring any signal pre-processing such as channel estimation and timing and frequency synchronization [63]. Traditionally, TR is done by domain experts, who use carefully designed hand-crafted rules to extract features from the radio signals. On the contrary, state-of-the-art approaches based on Deep Neural Networks (DNNs) can extract features directly from raw input data and automatically perform the recognition task on those features. However, DNNs approaches have two main drawbacks: 1) they are mainly trained in a supervised way, which implies that the whole data used for training must be labeled, 2) their training algorithms, such as Stochastic Gradient Descent (SGD) [31], require a large amount of data to obtain a good performance [32] otherwise, the resulting trained model may suffer severe overfitting problems [33].

Generally, assigning labels to data can be expensive, e.g., very time-consuming, and/or some of the data might not have any labels due to incomplete knowledge of the ground truth class labels, e.g., the radio technologies to be classified are entirely unknown. On the contrary, sensing the spectrum using modern radios allows for collecting a large amount of unlabeled data at no cost. Therefore, it is of utmost importance to formulate the TR problem that uses labeled and unlabeled data and designs robust systems that can deal with different amounts of them.

Semi-supervised Learning (SSL) is a Machine Learning (ML) technique that learns from unlabeled data by extracting a good representation of the data distribution and then using it to solve the supervised problem with a reduced number of labels [49]. Given the number of wireless technologies that already exist and the new ones under development, there is a need for efficient spectrum usage via collaboration and coexistence. For this, CR systems require a new TR approach that allows exploiting a large amount of unlabeled data while improving the classification accuracy by using a limited labeled data set.

In this chapter, we propose an SSL-based TR system that can work on raw In-phase and Quadrature (IQ) samples and does not require the whole data set to be labeled, which is a time-consuming and challenging task. The proposed solution uses Deep Autoencoder (DAE), which requires an unlabeled data set and only a few labeled examples. Moreover, its performance was evaluated in the Defense Advanced Research Projects Agency (DARPA) Colosseum testbed [18, 19] against a DNN architecture trained using

Supervised Learning (SL). We show that the proposed scheme outperforms the DNN recognizing sixteen different and unknown radio technologies while only requiring a limited labeled data set. Finally, this chapter provides a critical functional block to support the ideas behind the Chapters 4 and 5.

The rest of this chapter is structured as follows. Firstly, Section 3.2 introduces some of the most relevant work on TR. Secondly, the proposed SSL approach is presented in Section 3.3, including the architectural design and training pipeline. Thirdly, the data set generation and its details are described in Section 3.4. Fourthly, the experimental results and performance evaluations are provided in Section 3.5. Finally, conclusions are presented in Section 3.6.

3.2 Related Works

This section presents some of the most relevant work on TR. For a more exhaustive review of the general radio signal identification problem, we recommend [44] and [64] to the readers.

TR has been mainly applied to the identification of communication systems based on the differentiation of their channel method access, e.g., Single Carrier (SC) vs. Multiple Carrier (MC) [65], and it has been extended to classify various wireless communication technology standards, e.g., WiMAX vs. LTE [66]. Similar to other related tasks in radio signal identification, traditional approaches for TR are based on Likelihood-Based (LB) and Feature-Based (FB) using high-order statistics features such as moments, cumulants, and cyclic cumulants. Karami et al. propose an algorithm to identify Spatial Multiplexing (SM) and Alamouti (AL)-coded Orthogonal Frequency-Division Multiplexing (OFDM) signals for Multiple-Input and Multiple-Output (MIMO) systems based on second-order signal cyclostationarity [63]. This algorithm only requires the cross-correlation of the received signals on multiple antennas to discriminate between these two classes of OFDM systems. Firdaoussi et al. propose a method to obtain the Generalized Mean Ambiguity Function (GMFA) of the received signal and use it to discriminate between OFDM signals and Single Carrier Linear Digital (SCLD) in channels with additive white Gaussian noise [67].

For the identification of wireless technology standards, Bouzegzi et al. propose an algorithm that can discriminate among different technologies such as WiMax, WiFi, and DVBT by exploiting the fact that these technologies are based on OFDM but differ from their intercarrier spacing used in OFDM [68]. The algorithm estimates the intercarrier spacing based on the maximum-likelihood principle. The proposed algorithm does not need a training sequence and is more robust than autocorrelation-based methods under small-length cyclic prefixes and a multipath environment. Al-Habashna et al. propose an algorithm based on second-order cyclostationarity properties of the LTE and WiMAX technologies and use them as discriminating features for classification [66]. The proposed algorithm does not require carrier, waveform, and symbol timing recovery information. This approach also provides immunity to phase, frequency, and timing offsets. The previous approaches for TR require expert knowledge for either modeling the signals and the environment (LB methods) or selecting the required features (FB methods). Therefore, they cannot be used to identify unknown radio technologies.

More recently, several approaches based on Deep Learning (DL) have been proposed to solve the TR task using raw time, frequency, and time-domain data. Kulin et al. propose a Convolutional Neural Network (CNN) to identify single transmissions of ZigBee, WiFi, and Bluetooth radio technologies using raw IQ and the Fast Fourier Transform (FFT), the amplitude and phase of the raw IQ samples [69]. Without requiring any feature engineering, the proposed models achieved an accuracy above 80% in scenarios with Signal-to-Noise Ratio (SNR) $> -10dB$, above 95% accuracy using raw IQ samples, and near 100% with FFT and amplitude/phase in scenarios with SNR> 5 dB.

Biter et al. propose a CNN that can recognize 802.x standard compliant technologies using a time-frequency representation of the spectrum for a wide range of SNRs [70]. This model outperforms standard feature-based classification methods regarding classification accuracy and can detect and identify these technologies when they overlap in time. Finally, Yi et al. propose a real-time external interference source classification method for a ZigBee-based wireless sensor network using a CNN classifier and Received Signal Strength (RSSI) values as input data [71]. As interference, the model can identify WiFi beacons, WiFi video streaming, WiFi file transfer, Bluetooth iBeacon, and microwave oven RSSI traces. The proposed model can achieve an accuracy of over 93%, detecting the different interference classes with minimal computational resources.

In general, traditional methods such as LB and expert FB engineering combined with pattern recognition have been outperformed by supervised DL methods in the task of TR. Supervised DL methods remove the need for expert knowledge about the environment and the signal features used for classification by using the power of automatic feature abstraction. However, it requires the whole data set to be labeled. Labeling becomes time-consuming and challenging for both the technologies to be recognized and the environment to be entirely unknown. To overcome these limitations, in this chapter, we propose a SSL approach for TR that separates the feature extraction from the classification task in the DL architecture, so the use of unlabeled data is maximized. At the same time, the proposed approach minimizes the use of domain expertise knowledge by requiring only a small portion of the entire data set to be labeled to obtain a good performance, which is not the case with supervised DL models.

It is worth noting that while this chapter explores SSL, several other learning approaches have emerged in recent years that can also improve label efficiency during training. Self-Supervised Learning (Self-SL) [72] leverages large amounts of unlabeled data to generate predictive signals. Transfer Learning (TL) [73] allows the adaptation of pre-trained models to new tasks with minimal labeled data. Few-Shot Learning (FSL) [74] and Meta-learning (Meta-L) [75] enable rapid generalization from limited examples, addressing the challenges of scarce data. Despite these recent advances and shifts in TR research, such as distributed learning [76], SSL using DAE remains the state of the art for achieving label-efficient TR [77]. Therefore, the solution proposed in this chapter serves as a baseline against which novel architectures can be compared. These novel architectures have the potential to further reduce the need for labeled data by utilizing new learning paradigms or more advanced DAE architectures [78, 79].

3.3 A Semi-supervised system for Technology Recognition

In this section, we present the proposed SSL approach by first reasoning about how to define TR as an SSL learning problem and then derive the proposed core system along with the description of the main components.

3.3.1 Automatic Signal Identification as a classification problem

Given a classification problem with an input vector set X and their corresponding target variables set Y, the objective is to find a function f that predicts $y \in Y$ given a new value for $x \in X$, where y represents L class labels.

$$\begin{aligned} f : \mathbb{R}^n &\rightarrow 1, ..., K \\ y &= f(x) \end{aligned} \tag{3.1}$$

When the function f is used to map a given signal $s(t)$ to a set S of signal classes (labels) without requiring any pre-processing of the signal, then the classification problem is termed Automatic Signal Identification (ASI) [63]. Traditionally, this problem has been studied in two main research lines: Automatic Modulation Classification (AMC), where Y is the set of modulation schemes to be identified, and TR, where Y is the set of wireless radio technologies to be identified, e.g., generic medium access technologies such as SC vs. MC, standard wireless technologies such GSM, WiMAX, LTE, WiFi, Bluetooth, and ZigBee, among other, or more recently, unknown technologies such as the ones participating in the DARPA Spectrum Collaboration Challenge (SC2).

In ASI, three approaches have been mainly used in the literature: LB, FB, and supervised DL. The first two use signal processing and pattern recognition methods to identify signals. On the other hand, DL methods are Artificial Intelligence (AI) algorithms that learn by representing the world as a nested hierarchy of concepts, with each concept defined in relation to more straightforward concepts and more abstract representations computed in terms of less abstract ones [33]. DL algorithms allow building the mathematical function f as a combination of many simpler functions. To date, most of the research on TR has focused on using DNNs in a supervised way. However, this method has a drawback: building f for a complex task as TR requires a large data set of training examples with their corresponding labels. Otherwise, the training with supervised learning techniques on a small labeled data set often results in learning the training data but severely failing to predict the correct class of unseen data (overfitting) [33]. A more detailed review of related works on this subject is presented in Section 3.2.

In real environments, we often can collect large amounts of (unlabeled) IQ samples. However, labeling all of them may be costly, e.g., in terms of time, and/or some of the data might not have any labels at all due to incomplete knowledge of the ground truth class labels, e.g., the radio technologies to be classified are entirely unknown and increases the complexity of the labeling task. Therefore, solving the TR problem for unknown technologies in unknown environments, which is the case of SC2, is challenging and requires another approach that allows DL models to use unlabeled data to bootstrap the learning and minimize the number of labels required to solve the supervised learning task efficiently. It is here where SSL will play a key role.

As explained in Section 2.2, SSL allows algorithms to learn with fewer labels compared to DL-based SL approaches. To use SSL algorithms for recognition, it is required that the knowledge acquired about the distribution of the examples from the unlabeled data set, i.e., $p(x)$, is helpful to infer $p(y|x)$. Otherwise, semi-supervised learning may decrease the performance of the SL classifier by misguiding it during the learning process. SSL tries to use the unlabeled data to learn valuable information about the data and then use it to fine-tune a classifier with a reduced number of labels. By extending this approach and generalizing it to be used as a system for TR, we can use a reduced number of labels, in comparison to a DL model, with reasonable accuracy on the recognition task, even if the technologies are entirely unknown and no information about the environment is provided.

3.3.2 Spectrum Manager Framework

Figure 3.1 shows a spectrum manager framework that elaborates where the results of TR can be used. The framework comprises a spectrum manager, which makes spectrum decisions, and N radios, which represent unknown wireless technologies. The goal of the spectrum manager is to assist the N unknown wireless technologies in making spectrum decisions by first identifying them and then doing frequency domain analysis. In order to enable this, the spectrum manager executes the following tasks in the listed manner: a) *training*, b) *validation*, c) *frequency domain analysis*, and d) *spectrum decision*. In this work, we focus on the lower two blocks, i.e., *training* and *validation*, to enable TR for CR systems.

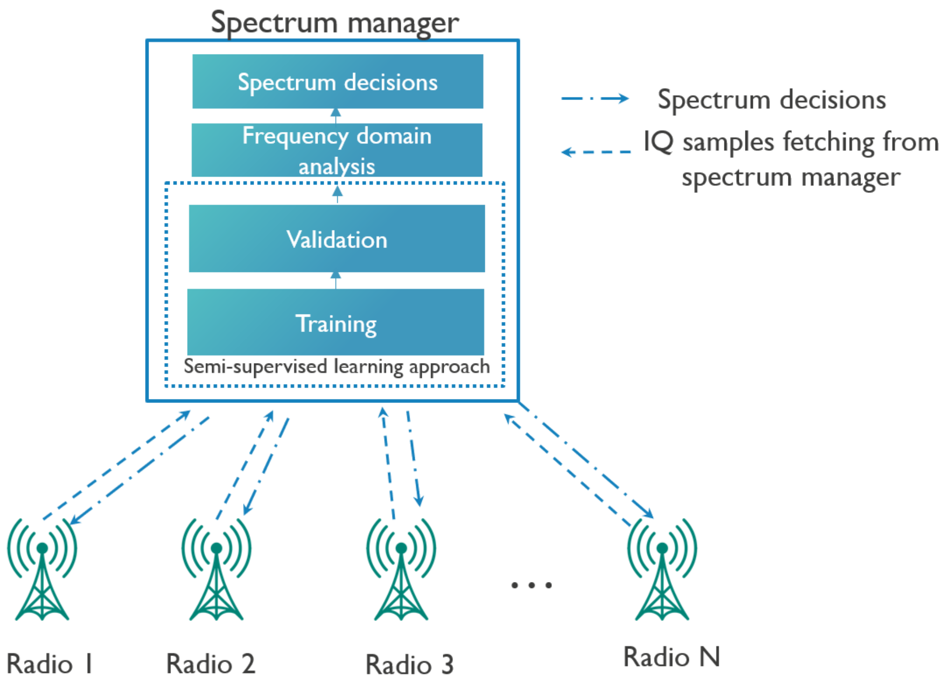

Spectrum manager
Spectrum decisions
Frequency domain analysis
Validation
Training
Semi-supervised learning approach
Spectrum decisions
IQ samples fetching from spectrum manager
Radio 1
Radio 2
Radio 3
Radio N

Figure 3.1: A spectrum management framework.

Focusing on the TR block, the *training* task is used to train a model in a semi-supervised way with raw IQ samples of the N radios using a DAE. A detailed description of the semi-supervised approach and its implementation is given in Section 3.3 and Section 3.4, respectively. Once the model is trained, in the *validation* task, it can identify the N unknown wireless technologies. In the *frequency domain analysis* task, frequency domain analysis of the identified technologies is done by extracting spectrum occupancy information of the technologies. Finally, in the *spectrum decision* task, the radio uses the extracted spectrum efficiency information to define actions, such as changing the radios' center frequencies and assigning a collision-free time slot for transmissions, so that fair coexistence can be achieved. Once the spectrum decisions are made, they are notified to the N radios via control channels.

3.3.3 System description

Sensing and capturing over-the-fly radio signals in the form of IQ samples is simple and can be performed using Software Defined Radio (SDR) platforms. However, IQ samples labeling is a difficult task for the following reasons: 1) an expert is required to identify and label each captured sample, and 2) in unknown environments, the number of unknown signals increases the complexity of the labeling task. The proposed approach decouples the feature extraction via unsupervised learning and the classification tasks via supervised learning while keeping the high expressiveness of DL models. The overall workflow of the proposed semi-supervised learning approach is shown in Figure 3.2. Below is a description of each block and the actual implementation details.

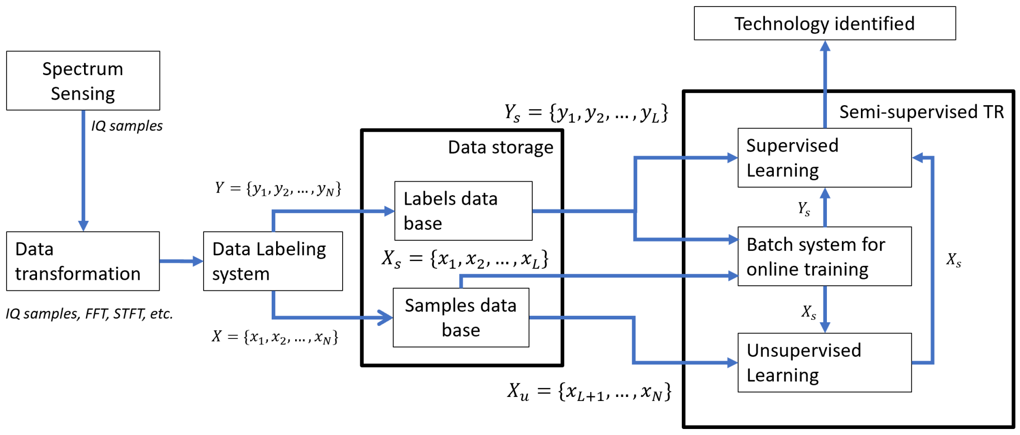

Technology identified
Spectrum Sensing
IQ samples
$Y_s = \{y_1, y_2, \ldots, y_L\}$
Semi-supervised TR
Data storage
Supervised Learning
$Y = \{y_1, y_2, \ldots, y_N\}$
Labels data base
Y_s
Data transformation
Data Labeling system
$X_s = \{x_1, x_2, \ldots, x_L\}$
Batch system for online training
X_s
IQ samples, FFT, STFT, etc.
Samples data base
X_s
$X = \{x_1, x_2, \ldots, x_N\}$
Unsupervised Learning
$X_u = \{x_{L+1}, \ldots, x_N\}$

Figure 3.2: Proposed semi-supervised learning approach workflow for TR.

Spectrum sensing: This module is responsible for sensing the spectrum and capturing IQ samples that the subsequent blocks will further process.

Data Transformation: Depending on the model to be trained, the original IQ samples, which are time domain representations of radio signals, can be transformed into other domains such as frequency or time-frequency. In this work, we focus on the IQ samples representation as it does not require further processing, which can be seen as the rawest version of spectrum data.

Data labeling system: In this block, two steps are performed: a) sample selection and b) labeling of the samples. Since the proposed architecture is semi-supervised, selecting representative samples of the radio technologies that must be identified is essential. Here, domain expert knowledge or in combination with pseudo labeling is used. This block stores all the samples and the labels associated with the labeled samples.

Data storage: This block comprises two databases: 1) sample and 2) label databases. IQ samples are stored in the sample database, while the label database stores the labels of a reduced set of examples. Depending on the kind of data and the training strategy, the databases are connected to one or more blocks: supervised learning (sample database and label database), unsupervised learning (sample database), and the batch system (sample database and label database).

Batch system for online training: In offline training, the input data is created by selecting a portion of the data from the sample database via some predefined strategy, e.g., uniform random selection. In online training, on the other hand, the input can be provided by a batch system that takes data from the sample database and uses it to retrain a model.

Semi-supervised classification: This block receives the sensed data and performs the classification task. The block also receives a limited labeled data set from the data labeling system block. Based on the labeled and unlabeled data sets, different learning algorithms can be used in the supervised and unsupervised learning blocks and how they interact to perform the SSL task.

Technology Recognized: This is where the proposed architecture indicates to which class a given capture sample belongs. Note that a class label may be as simple as the name of the technology. However, it can also be more expressive and contain information about the spectrum utilized over time, central frequencies, duty cycle, etc.

The proposed workflow is flexible to support a range of SSL algorithms, training methods, and input types. The selection of the semi-supervised approach mainly depends on various factors, including the amount of available data, the number of labels, the complexity of the radio signals to be identified, and the need for offline or online training capabilities, etc.

3.3.4 Semi-Supervised Learning using Deep Autoencoders

The SSL TR block shown in Figure 3.2 was implemented using a DAE [80]. The resulting architecture of the DAE for TR is shown in Figure 3.3. DAEs are DNNs trained to copy its input to its output. A DAE is composed of two parts: an encoder that maps $h = f(x)$, where h is known as the code, and a decoder that produces a reconstruction $r = g(h)$. In practice, DAEs are not trained to get $x = g(f(x))$ but to obtain an h that contains only useful information about x. To do that, h is constrained so that its dimension is smaller than x. This kind of DAE is called under-complete. The learning process of a DAE can be defined as:

$$\text{minimize } L(x, g(f(x))) \tag{3.2}$$

where L is a loss function indicating how similar is the input x and the reconstructed output $g(f(x))$. Under-complete DAE aims to learn only important data distribution

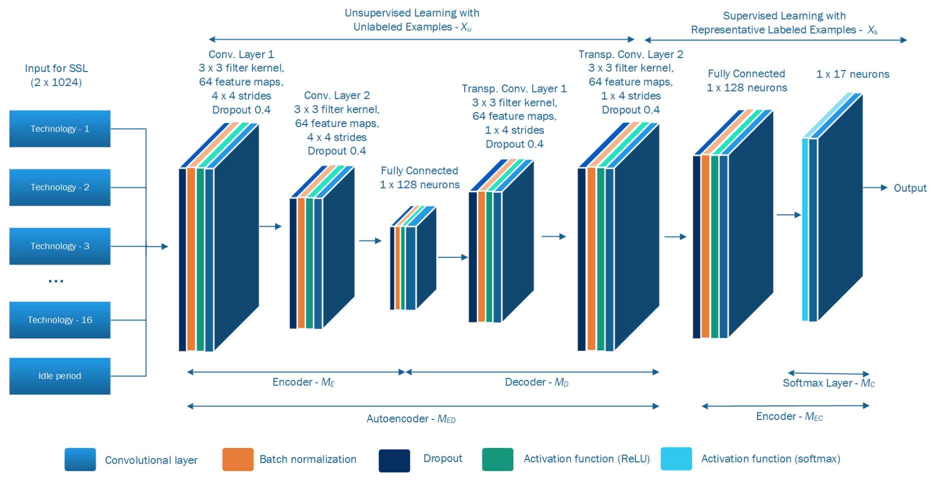

Unsupervised Learning with Unlabeled Examples - X_u
Supervised Learning with Representative Labeled Examples - X_s
Input for SSL (2 x 1024)
Technology - 1
Technology - 2
Technology - 3
...
Technology - 16
Idle period
Conv. Layer 1 3 x 3 filter kernel, 64 feature maps, 4 x 4 strides Dropout 0.4
Conv. Layer 2 3 x 3 filter kernel, 64 feature maps, 4 x 4 strides Dropout 0.4
Fully Connected 1 x 128 neurons
Transp. Conv. Layer 1 3 x 3 filter kernel, 64 feature maps, 1 x 4 strides Dropout 0.4
Transp. Conv. Layer 2 3 x 3 filter kernel, 64 feature maps, 1 x 4 strides Dropout 0.4
Fully Connected 1 x 128 neurons
1 x 17 neurons
Output
Encoder - M_E
Decoder - M_D
Autoencoder - M_{ED}
Softmax Layer - M_C
Encoder - M_{EC}
Convolutional layer
Batch normalization
Dropout
Activation function (ReLU)
Activation function (softmax)

Figure 3.3: SSL algorithm implemented using DAE.

features. To enforce learning good features and avoid learning to copy the input to the output, denoising DAE uses a different loss function to discourage learning the identity function as follows:

$$\text{minimize } L(x, g(f(\bar{x}))) \tag{3.3}$$

where $\bar{x}$ is a copy of x that has been corrupted with some noise. In this way, the DAE does not learn to map $x \rightarrow x$ but undo the corruption by learning the structure of $p(x)$ [81].

For SSL, DAE provides a two-step training process: First, we train the DAE M_{AE}, which is composed of the encoder M_E and decoder M_D networks, in an unsupervised way using only X_u. Second, after the unsupervised learning, we create a train a classifier M_{EC} using the encoder M_E together with a SoftMax classifier M_C in a supervised way using the reduced labeled data set X_s.

During the supervised training, M_E is used as a feature extractor for M_C. This step provides an initial bootstrapping on the classification task. Then, a fine-tuned step is performed, i.e., all layers in M_{EC} are retrained to increase the accuracy of the resulting model. Algorithm 1 shows the pseudo-code of the training procedure.

The DAE block of our system was designed by following a CNN architecture [82]. While traditional DNNs are built by connecting a series of fully-connected layers, CNN connects the neurons of a given layer, called the Conv layer, with only a few numbers of neurons of the next layer to reduce the computational complexity of the learning. Note that this kind of Neural Networks (NNs) has been shown to perform well with IQ samples as input [83], which motivates us to follow the same design pattern.

The encoder of the DAE comprises two Conv layers with rectified linear unit (ReLU) activation function, each followed by Batch Normalization and a Dropout layer for regularization. We use strides> 1 instead of Max Pooling layers. Note that the dropout layers allow the DAE to behave as a denoising DAE to improve its capacity as a feature extractor [84]. Figure 3.3 shows an overview of the resulting architecture and the parameters of the

Algorithm 1 SSL procedure using DAE.

```
Require: Unlabeled data set: $X_u$
Require: Labeled data set: $X_s$
Require: (Optional) Trained Autoencoder network: $M_{AE}$
Require: (Optional) Trained Encoder-Classifier network: $M_{EC}$
 1: if $M_{AE}$ Exists then
 2:     UnFreezeEncoderWeights($M_{AE}$)
 3: else
 4:     $M_{AE}$ = CreateModel($M_E$,$M_D$)
 5: while Unsupervised training do
 6:     TrainAutoencoder($M_{AE}$, $X_u$)
 7: if $M_{EC}$ do not Exist then
 8:     $M_{EC}$ = CreateModel($M_E$,$M_C$)
 9: while Supervised training do
10:     FreezeWeightsEncoder($M_{EC}$)
11:     TrainClassifier($M_{EC}$, $X_s$)
12:     UnFreezeWeightsEncoder($M_{EC}$)
13:     TrainClassifier($M_{EC}$, $X_s$)
14: return Trained models: $M_{AE}$ and $M_{EC}$
```

Conv layers. The specific parameters of each layer, such as the number of filters, strides, dropout rate, etc., were determined using hyperparameter swapping. The proposed encoder configuration generates an intermediate code of size 128, e.g., a reduction factor of 16x.

Similarly, the decoder part follows the same pattern but in reverse order and replaces the Conv layers with Transposed Conv. The DAE contains 1M of trainable parameters. The Autoencoder (AE) was trained using batches of size 128, the Adam optimizer [85] with a learning rate of 0.0004, and binary cross-entropy as the loss function for reconstruction. We implemented our model in Keras [86] with TensorFlow [87] as the back-end, and it was training during 200 epochs: 100 epochs in unsupervised mode, 50 epochs only classifier (phase 1), and 50 epochs fine-tuning (phase 2).

The supervised part of the architecture is composed of the encoder part of the DAE in addition to two dense layers, one with 128 neurons and the second with 17 neurons, and a SofMax activation layer for classification. The resulting model (encoder+dense layers and classification) has 500k and 18k trainable parameters in phase 1 and phase 2, respectively. This model was trained using the same parameters as the DAE except that the loss function was categorical cross-entropy, and the learning rate was reduced to 0.004.

3.3.5 Baseline using CNN

We implemented and trained a CNN architecture to be used as a baseline for comparison with the proposed SSL approach using DAE. This architecture comprises three Conv layers with ReLU activation function, each followed by a max-pooling, batch normalization, and a Dropout layer for regularization. The CNN model was fine-tuned to have a high performance using the entire data set, and it was trained during 100 epochs to guarantee

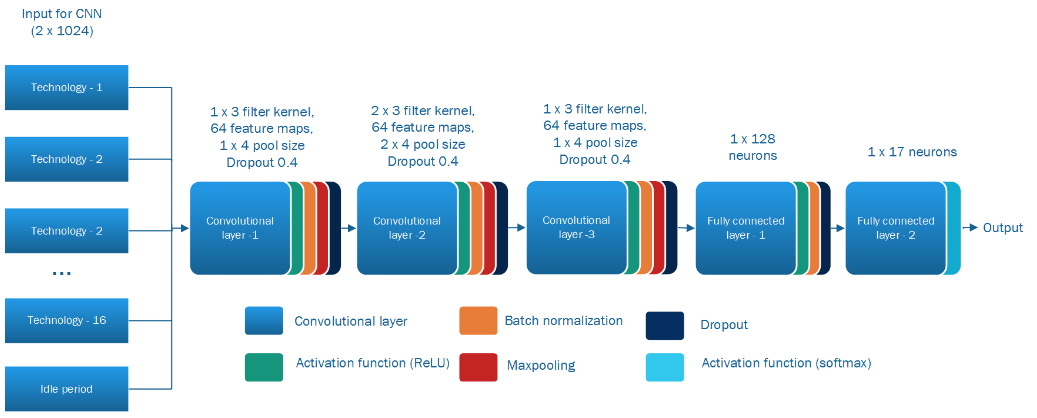

Input for CNN (2 x 1024)
Technology - 1
Technology - 2
Technology - 2
...
Technology - 16
Idle period
1 x 3 filter kernel, 64 feature maps, 1 x 4 pool size Dropout 0.4
2 x 3 filter kernel, 64 feature maps, 2 x 4 pool size Dropout 0.4
1 x 3 filter kernel, 64 feature maps, 1 x 4 pool size Dropout 0.4
1 x 128 neurons
1 x 17 neurons
Convolutional layer -1
Convolutional layer -2
Convolutional layer -3
Fully connected layer - 1
Fully connected layer - 2
Output
Convolutional layer
Batch normalization
Dropout
Activation function (ReLU)
Maxpooling
Activation function (softmax)

Figure 3.4: Base-line SL algorithm implemented using CNN.

a fair evaluation. Figure 3.4 shows an overview of the resulting CNN model.

3.4 Data set generation

We generated the data set in the DARPA Colosseum [18], the testbed used for the DARPA's three-year SC2 on smart radios and spectrum sharing. The SDRs available in Colosseum are Ettus Universal Software Radio Peripheral (USRP) X310 with UBX 160 USRP Daughterboards. Given this hardware configuration, the Radio Frequency (RF) monitor module is implemented as a thread running along with the Physical Layer (PHY) of our radio stack. The RF monitor uses the USRP 's Radio stream # 1, where the RX channel is used for spectrum sensing, and the TX channel supports dual PHY transmissions. Figure 3.5 shows how the RF monitor block interacts with the USRP and TR blocks. Note that this module exclusively serves the ML algorithms running in our radio, and it sends the IQ samples for TR to our data storage for labeling and future use during offline training.

During phase 2 of the competition (2018), twenty technologies participated and used the Colosseum's capabilities to train their Intelligent Radio (IR). We played 55 games in a 6 Mhz bandwidth scenario with a constant 60-dB path loss among all the nodes. Each game was played against random technologies, and we managed to collect IQ samples of 16 out of 19 technologies (excluding us) and noise (idle period). The IQ values are stored using 16-bit binary integers. The RF monitor block was set to collect samples at 23.04Msps, giving us 43.04 ns space between IQ samples. Figure 3.6 shows the resulting amplitude and spectrograms from some of the collected IQ samples for each unknown technology. A particular case was Technology 1, which was transmitting out-of-band.

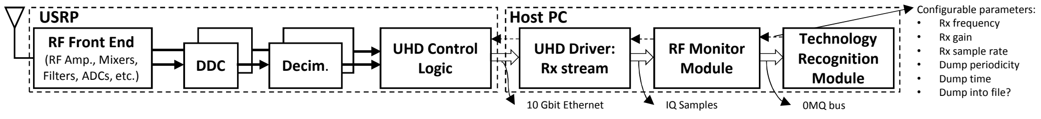
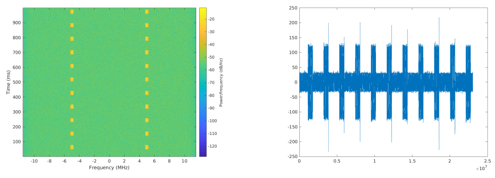
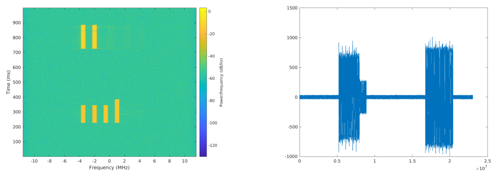


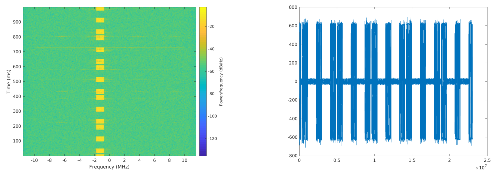
Figure 3.5: RF Monitor module and its connections.



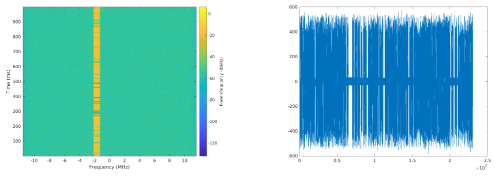
Technology 1



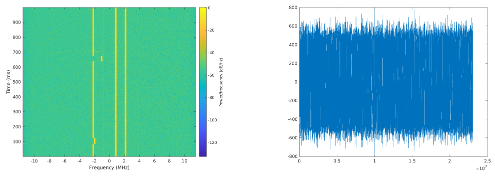
Technology 2



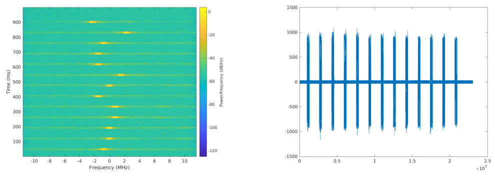
Technology 3



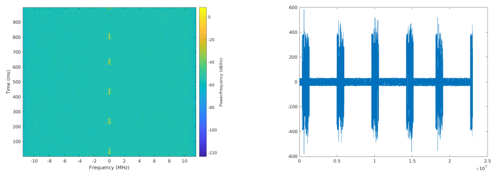
Technology 4



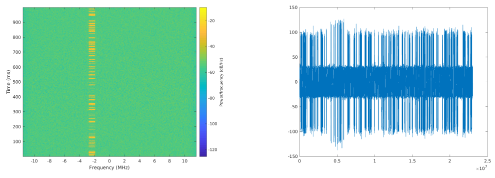
Technology 5



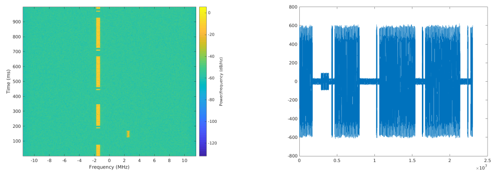
Technology 6



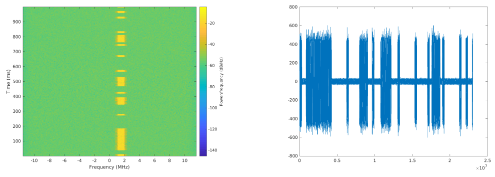
Technology 7



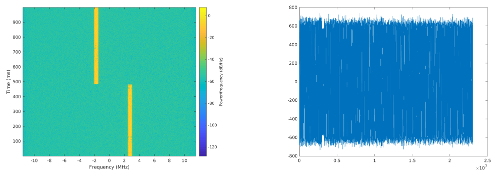
Technology 8



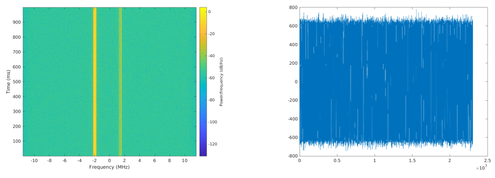
Technology 9



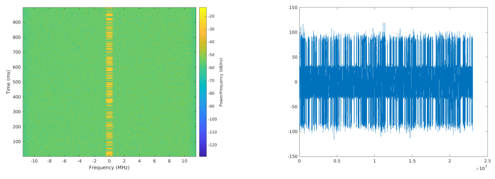
Technology 10



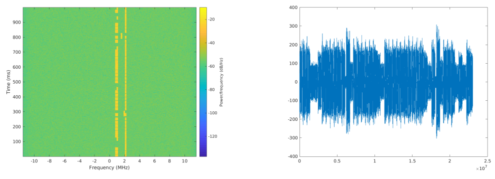
Technology 11



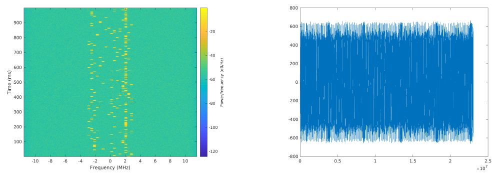
Technology 12



Technology 13



Technology 14


Technology 15

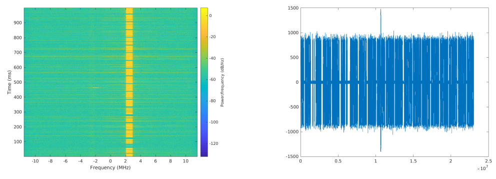

Technology 16

Figure 3.6: Time and time-frequency signatures of the wireless technologies to be recognized.

Once the batches of IQ samples were collected, a couple of those were visualized as spectrograms and labeled accordingly as different technologies. These limited examples are the representative examples used in the SL classification part of the proposed semi-supervised approach. After removing the noise of the captured signals using a Support Vector Machines (SVM) and subsequently normalizing the whole data set in the range [0,1], the total size of the IQ samples data set was 93 GBs.

The whole data set was transformed into 11.3M examples, where each example corresponds to a pair of 1024 IQ values. For implementing the proposed approach and the baseline, a data set with 1k labels per technology was used for training, validation, and testing. Also, a second data set was created based on the 1k data set via data augmentation to emulate ten different SNR levels. The summary of the TR data set is given in Table 3.1.

Table 3.1: Summary of the TR data set.

Type	**Value**
Total examples	11.3M
Size of example	1024 x 2 x 16 bits
Number of labels	17, 16 technologies + noise
Size data storage	93GB
Training and validation data set	1k examples per label
Augmented data set for SNR test	10k examples per label

3.5 Results

In this section, the proposed SSL approach is compared against the CNN trained in a supervised way in terms of a) convergence performance, b) performance in the presence of different noise conditions, and c) labeling efficiency. For the comparison in the result section, we termed the proposed approach and the baseline as M_{EC} and CNN, respectively.

3.5.1 Algorithm Convergence

To evaluate the performance of DNN models, two performance metrics are often used: the accuracy, which measures the proportion of examples that the model can predict

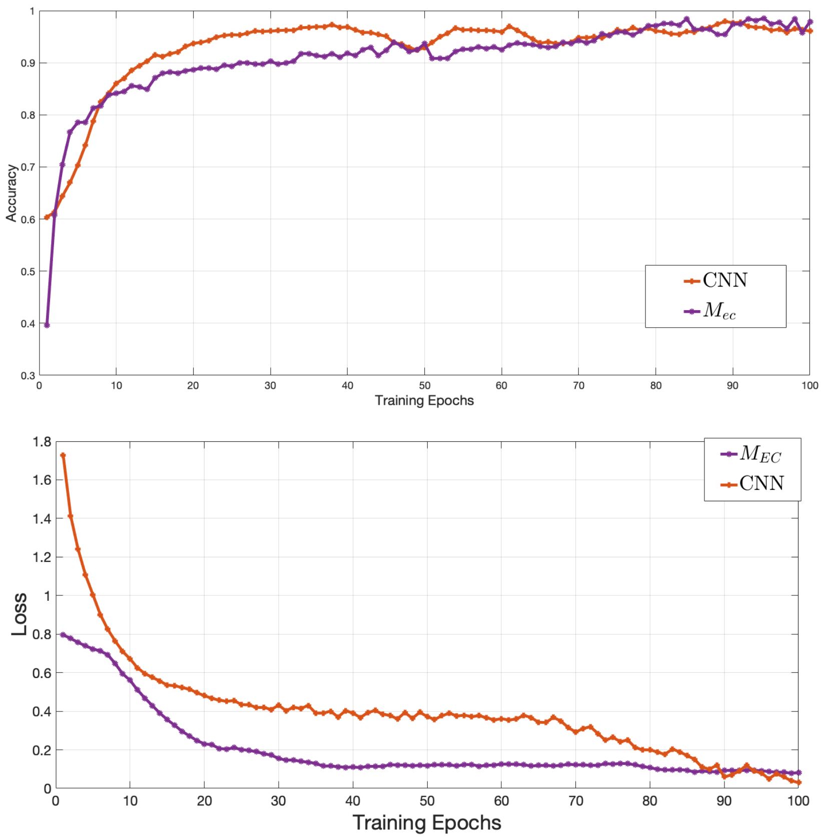


Figure 3.7: Model training convergence: accuracy (top) and loss (bottom) curves using the validation data set.

correctly, and the loss, which quantifies the inconsistency of the predicted value $\hat{y}$ and the actual label y. The original data set of 1k samples per technology was split into training (80%), validation (10%), and test (%10) data sets. This evaluation used the whole training labeled data set for training both models. The impact of reducing the number of labels used for training is analyzed in Section 3.5.3. Figure 3.7 shows the validation accuracy and loss of the models concerning the number of epochs. Here, we limit ourselves to showing only the validation results, not the training results, because we want to show the generalizing performance. The proposed SSL model M_{EC} achieves similar convergence performance as the baseline model CNN trained using SL.

It is important to notice that at epoch 50, when the weights of the encoder M_E are unfrozen for fine-tuning, a slight decrease in the accuracy for a couple of epochs is observed. This is expected because the model M_E needs to be modified for the new classification task. However, later on, it keeps increasing the accuracy over time and reaching the same performance as the CNN model. The loss curves decrease as expected, which indicates correct learning over time. A smooth decaying loss function of M_{EC} shows that the selected hyperparameters, such as learning rate and regularization, help the optimizer to find an optimal point.

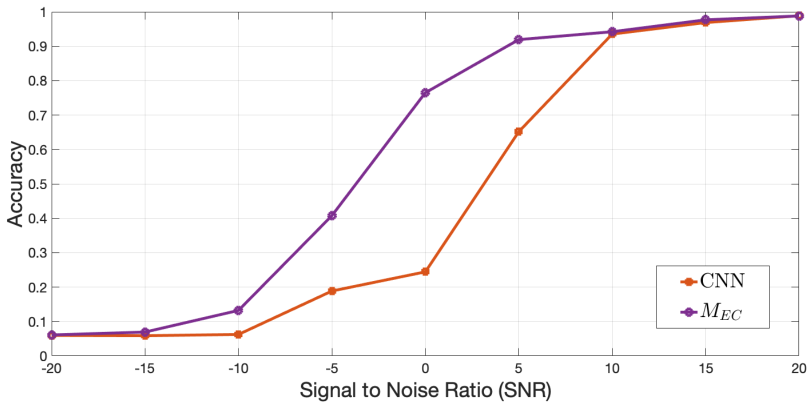


Figure 3.8: Model accuracy at different SNR.

3.5.2 Model performance in the presence of noise

A good accuracy performance of ML models for the TR problem under different noise levels validates the models' efficacy in noisy environments. One way to achieve this is to apply data augmentation techniques to the available training data set, which can be achieved by processing the data set and including different noise levels. However, models that can implicitly learn to be more tolerant of noise without requiring explicit training under noise are more potent in real applications. To evaluate this performance, we generated a new data set by performing data augmentation on the original 1k data set with different levels of SNR. The concern of this augmented data set is to validate the performance of the trained models M_{EC} and CNN and show how they can cope in noisy environments. Figure 3.8 shows the accuracy of both models when we test the augmented data set on the trained model.

The results show that the trained SSL model M_{EC} is more robust in noisy environments than the baseline CNN. While the CNN is trained to accurately predict labels from examples that come from the data distribution, the unsupervised training exploits the properties of denoising DAE. Although we do not use the augmented data set to train the models, which will increase their architecture size, we use dropout layers as a regularizer. This selection has the same effect as adding noise to hidden layers [84]. The unsupervised learning step in the proposed SSL approach exploits the denoising property of DAE, which forces the trained model to be more resilient to noisy examples.

Figure 3.9 shows the two models' confusion matrices at different SNR. For SNR levels lower than $-15dB$, both networks cannot learn any useful information from the raw IQ samples, and their accuracy is very low. Above this value, it is clear that the M_{EC} model is more robust than the CNN-based model. The accuracy at 0 dB is 3x better in our M_{EC} model.

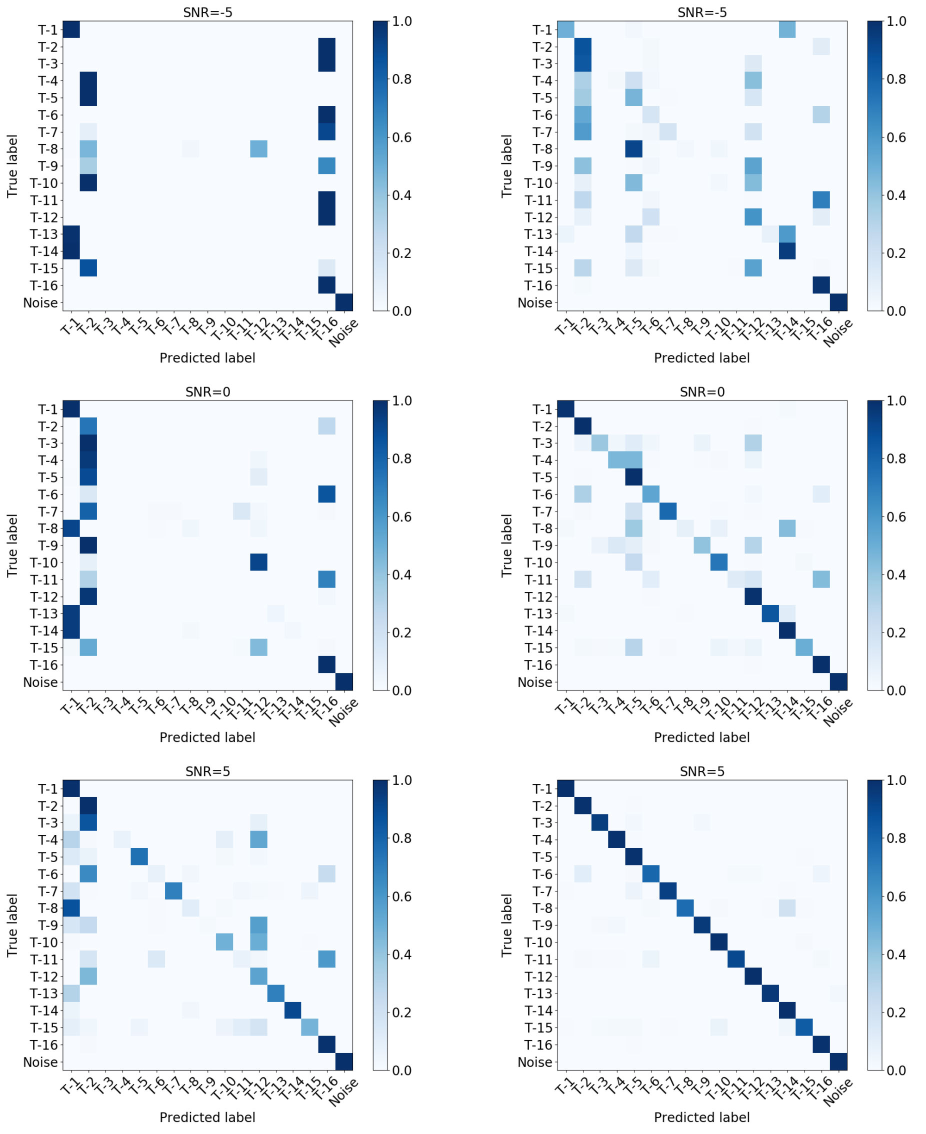


Figure 3.9: Confusion matrices for CNN (left) and SSL algorithm using DAE (right) at different SNR.

3.5.3 Labeling efficiency

The previous evaluations assumed that the M_{EC} model had been trained using the same amount of labels as the CNN to have a fair comparison. Also, both models are fine-tuned by adjusting the hyperparameters individually to achieve high accuracy, i.e., up to 97% on the 1k data set. However, we aim to limit the required number of labeled data sets while still achieving good performance using our approach. The first aspect to verify on the DAE training is if the feature extraction task is performed correctly. This is fundamental to reducing the number of samples of the labeled data set to train the classifier. Figure 3.10 shows a random $q(t)$ example from the test data set and the signal reconstructed by the DAE. Although the signals are not entirely identical, it is clear that the encoder is learning essential features to perform such reconstruction correctly. As the DAE training is unsupervised, the next step is to evaluate the impact of the number of labels during the supervised training.

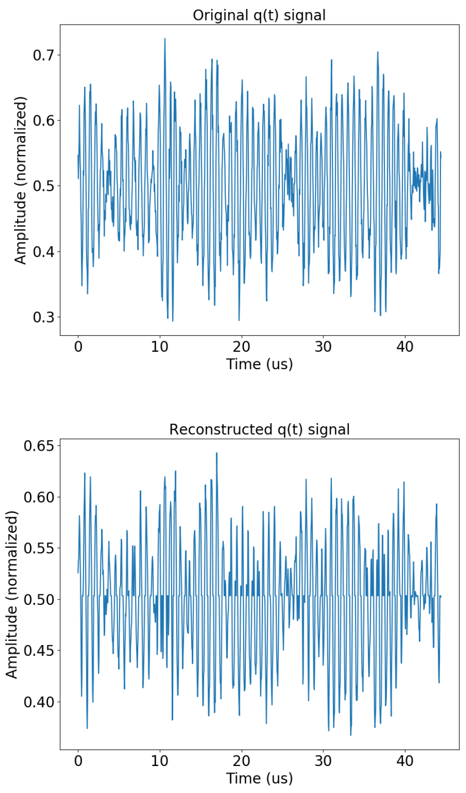


Figure 3.10: DAE reconstruction: Original q(t) signal (top) and reconstructed q(t) signal (bottom).

Figure 3.11 shows the impact of varying the number of labels available for training on the achieved training and validation accuracy of both models. In this evaluation, we train the models with the labeled data sets of different sizes, such as 10%, 33%, 55%, 77%, and 100% of the one used for the initial training. It is clear from the figure that the proposed approach, M_{EC}, takes advantage of the Unsupervised Learning (USL) step to bootstrap the validation accuracy with a limited number of labels. More importantly, the proposed approach achieves an accuracy of $\geq$ 70% with only 10% of the total number of labels, translated to 4.6 times better accuracy than the CNN model using the same amount of labeled data. In other words, M_{EC} generalizes better to unseen data than the CNN with a reduced set of labels. Once more labeled data is available for training, the CNN model increases its validation accuracy. On the other hand, note that both models' training accuracy achieves 100%. This behavior, i.e., overfitting the training data set, is expected since the DL models were designed and optimized for using the whole data set. However, while M_{EC} was able to have an increasing validation accuracy with the reduced data set, the CNN model memorized the reduced data set, and it did not extract features to generalize unseen (validation) data.

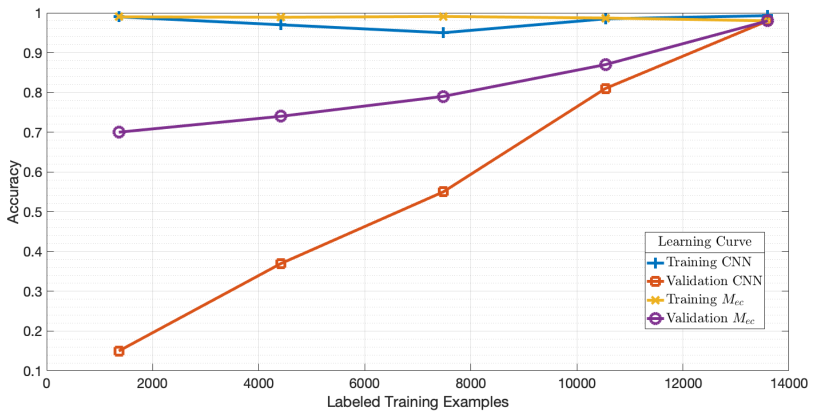


Figure 3.11: Impact of the number of labels used during learning.

3.6 Conclusions

TR will play an essential role in how new wireless technologies make decisions to use the available spectrum efficiently and coexist with any new, legacy, and even unknown technologies. In this chapter, we have proposed a novel SSL approach for TR that minimizes the need for labeling large data sets of spectrum data. In addition, the proposed approach requires only raw IQ samples, which can easily be acquired from low-cost sensing devices.

The evaluation illustrates that the proposed approach can achieve an accuracy of $\geq$ 70% with only 10% of the total number of labels, translated to 4.6 times better accuracy than the considered baseline CNN model using the same amount of labeled data. Besides, we found that the resulting DL model is more robust under corrupted input, e.g., noisy signals, than the CNN-based model, with up to 2x better accuracy at SNR levels of -5dB and up to 3x at 0dB.

Chapter 4

A scalable and decentralized spectrum-sharing framework for Collaborative Intelligent Radio Networks

The content of this chapter has been partially published in:

- **M. Camelo**, R. Mennes, A. Shahid, J. Struye, C. Donato, I. Jabandzic, S. Giannoulis, F. Mahfoudhi, P. Maddala, I. Seskar, I. Moerman, and S. Latré, "An AI-Based Incumbent Protection System for Collaborative Intelligent Radio Networks," in IEEE Wireless Communications, vol. 27, no. 5, pp. 16-23, October 2020, doi: 10.1109/MWC.001.2000032.

As a first step of a cognitive radio cycle (see Section 2.1.3), the Technology Recognition (TR) module presented in the previous chapter serves as a spectrum sensing technique grounded in Machine Learning (ML). This chapter introduces an innovative multi-tier spectrum-sharing framework built upon TR. We have redesigned its architecture for near-real-time operation and enhanced it with new learning and reasoning modules, i.e., the second step of a cognitive radio cycle, to autonomously protect incumbent (primary) technologies. Unlike conventional multi-tier architectures, which rely on centralized authorities and predefined priorities to mitigate interference, our framework empowers autonomous radios to share and reuse spectrum without coordination efficiently. This paradigm shift enables more dynamic and efficient spectrum utilization compared to state-of-the-art approaches being deployed worldwide, such as Citizens Broadband Radio Service (CBRS) and Licensed Shared Access (LSA).

4.1 Introduction

New wireless technologies like 5G require more available radio spectrum to support new applications with high demands on data. However, as presented in Section 3, this growth is expected to keep increasing in the future [61]. As spectrum is, and will continue to be, an obviously essential resource for wireless connectivity to transport such demands of data, dynamic access, and management to additional wideband spectrum as well as efficient utilization of the existing spectrum is of critical importance. However, today, there is a shortage of available spectrum [6] to fulfill such demands, which is mainly due to the obsolescence of the traditional static frequency plan based on providing access to single usage or a single user, which has granted exclusive use of a specific portion of the spectrum in a given geographic location. As a result, most of the allocated spectrum is underutilized, and the part mainly used by the technologies we use for daily communication is over-utilized.

This exclusive-usage spectrum allocation model is being updated by several global efforts to make additional spectrum available for broadband data and increase the spectrum reuse [7]. The Citizens Broadband Radio Service (CBRS) and the Licensed Shared Access (LSA) models are initiatives that provide multi-tier spectrum-sharing frameworks in the reallocated spectrum. In these frameworks, the incumbent, i.e., the technology that used the spectrum exclusively in the past, has to be protected against interference caused by the new technologies sharing the same spectrum. For example, CBRS offers centralized three-tiered access to users via an automated frequency coordinator, known as a Spectrum Access System (SAS), which guarantees that once a higher priority user is transmitting, the lower ones must vacate the spectrum to avoid interference. As part of the SAS, spectrum sensing techniques are fundamental to identify Radio Access Technologiess (RATs) and make decisions based on it.

Although the multi-tier models are an initial step to mitigate spectrum scarcity via spectrum sharing, they still suffer several fundamental problems. Firstly, a single point of control or coordination can become a bottleneck as the number of users and devices increases [22, 23, 24]. Secondly, as these models rely on well-defined rules and mechanisms [25], they are not optimized to ensure fair and efficient use of the spectrum [26]. Thirdly, these models require a massive overhaul of the centralized infrastructure to support changes in environmental conditions, regulations, and policies, which can be a very time-consuming and bureaucratic task that slows down the deployment of new technologies or services that rely on shared spectrum [27]. Finally, accommodating wireless communications within some specific frequency bands (e.g., radar spectrum) requires novel spectrum-sharing paradigms since existing spectrum-sharing approaches are not designed for all coexistence scenarios [28, 29].

As a solution, the DARPA Spectrum Collaboration Challenge (SC2)[14], a three-year competition organized by Defense Advanced Research Projects Agency (DARPA), has shown that Collaborative Intelligent Radio Network (CIRN), i.e., Artificial Intelligence (AI)-based autonomous wireless radio technologies that exchange explicit information to solve joint problems via collaboration, can share and reuse spectrum efficiently without coordination and with the guarantee of incumbent protection [19]. This is a step beyond modern Cognitive Radio (CR) networks since CIRN can reduce the uncertainty about spectrum measures using collaborative information and self-learning and self-adapting the radio operation parameters based on experiences.

Built on top of our Technology Recognition (TR) module, which serves as the first step of the cognitive radio cycle (see Section 2.1.3), this chapter presents the architectural design and experimental validation of a next-generation spectrum sharing framework with incumbent protection capabilities. The proposed architecture does not require any central infrastructure to control and grant access to a shared spectrum based on the concept of CIRN. Moreover, the incumbent protection capabilities are provided by a two-step AI-based algorithm that, combined with collaborative information, can recognize, learn, and proactively forecast the incumbent's transmissions in near real-time. The near real-time operation is achieved by optimizing the architectural design of TR module in terms of input data (from In-phase and Quadrature (IQ) to averaged Fast Fourier Transform (FFT) points) and size of the model.

The rest of this chapter is structured as follows. Section 4.2 introduces the spectrum sharing based on the concept of CIRN. The proposed two-step AI-based algorithm for incumbent protection is presented in Section 4.3. Implementation details and the integration in a CIRN are described in Section 4.4. Section 4.5 provides the experimental results and performance evaluations. Conclusions are presented in Section 4.6.

4.2 Two-tier model framework for CIRN

Let us introduce the six main features of the multi-tier model shown in Figure 4.1 and on which our system is based.

1. It has two tiers: the protected incumbent and the other users.
2. Other users can use any available spectrum not used by the incumbent inside the shared spectrum band.
3. The users have no expectations of any interference protection.
4. There is a wired interconnection network for collaboration where the incumbent reports spectrum power measurements, which determines if non-incumbent transmissions are causing interference to it and its frequency operation details (central frequency and bandwidth).
5. Radio networks have (at least) one radio connected to the collaboration network to receive the incumbent reports and may exchange information with others.
6. Causing interference with the protected incumbent is penalized.

Notice that while features 1-3 are shared with the Dynamic Frequency Selection (DFS) model [7], features 4-5 are novel and are used to incentivize and enforce the incumbent's protection, and feature 6 is shared with CBRS. Specifically, the CBRS model penalizes the radios causing interference to the incumbent by shutting their transmission down. The proposed model is simpler and scales better than the CBRS and LSA frameworks in the number of users/incumbents since it does not require centralized authority controlling and granting access to the shared spectrum. However, it also implies that the incumbents need spectrum sensing capabilities to collaborate and share information about their spectrum power measurements.

On the radios' side, they need to be intelligent and fully autonomous to dynamically learn and apply the best policy determining their spectrum allocation. Simultaneously,

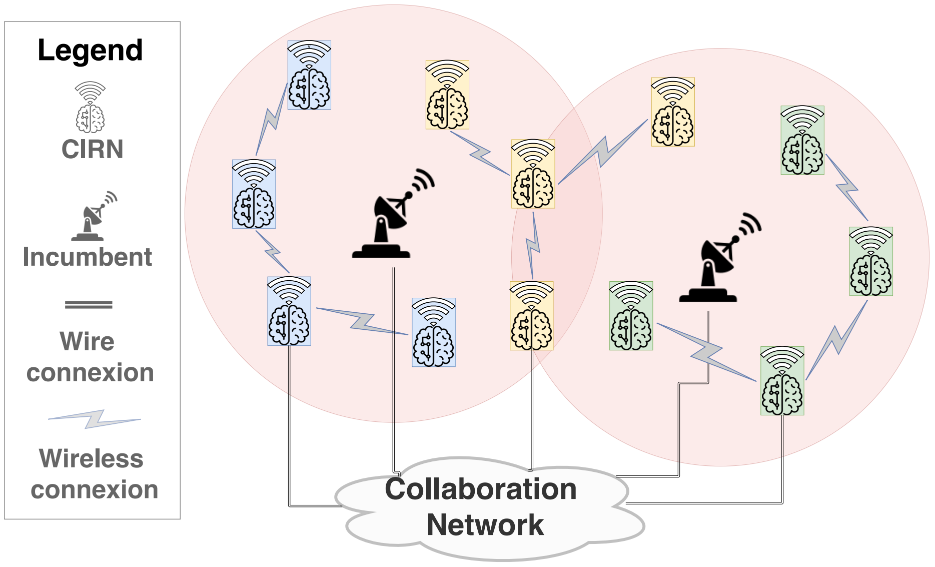


Figure 4.1: Two-tier model framework for incumbent protection using CIRN.

they need to protect the incumbents and minimize interference with other technologies. However, adding learning capabilities to the radios is not enough to protect the incumbent as the uncertainty about the spectrum state is higher if only local spectrum measures are used, and there is no mechanism for feedback on the radio's decisions. Here is where collaboration plays a key role. Collaboration among networks and incumbents reduces that uncertainty while providing a mechanism to augment the data used to learn and give feedback on the decisions made. A set of interconnected radios with these capabilities create CIRNs.

During the SC2, our team designed, implemented, and tested a CIRN called SCATTER [15]. Figure 4.2 provides a simplified view of the SCATTER architecture and the sub-modules of the Intelligent Control and Decision Engine (ICDE), which contains the building blocks for our incumbent protection system. The six fundamental modules are presented below.

Physical Layer (PHY): This layer is implemented as a Software Defined Radio (SDR) with features such as Orthogonal Frequency-Division Multiplexing (OFDM) waveform, bursty transmission, dual-concurrent physical layers, Field-programmable Gate Array (FPGA)-based filtered transmission, out-of-band full-duplex operations, and layer configuration based on timed commands [88].

Medium Access Control (MAC): This layer is based on an enhanced Multiple Frequencies Time Division Multiple Access (MF-TDMA) scheme, which slices the band capacity along with both time and frequency. Its capabilities include the distributed slot-allocation protocol, slot allocation with Quality of Service (QoS) support, and robust mechanisms against failures in slot allocation/removal procedures.

User Data Management (UDM): This layer audits and reports information about the incoming traffic from the user/application space in run-time.

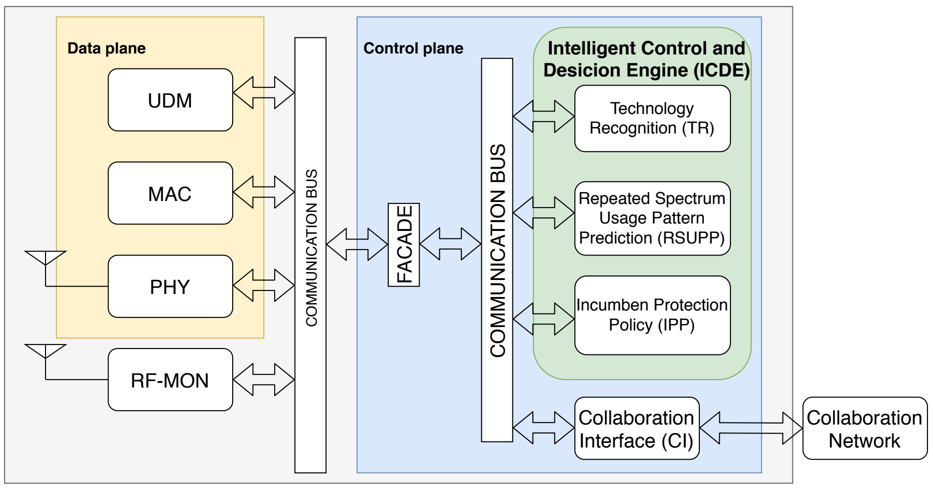

Data plane
UDM
MAC
PHY
RF-MON
COMMUNICATION BUS
Control plane
FACADE
COMMUNICATION BUS
Intelligent Control and Desicion Engine (ICDE)
Technology Recognition (TR)
Repeated Spectrum Usage Pattern Prediction (RSUPP)
Incumben Protection Policy (IPP)
Collaboration Interface (CI)
Collaboration Network

Figure 4.2: SCATTER architecture overview and the ICDE components for incumbent protection.

Radio Frequency Monitor (RF-MON): This FPGA-based module performs the spectrum sensing task in the system.

Collaborative Interface (CI): This module allows communication between the SCATTER CIRN, the protected incumbents, and other CIRNs via a well-defined collaboration protocol or language. It is generally used to send and receive messages from other networks, such as location, actual and predicted spectrum usage, performance metrics, and the incumbent's power measurement reports.

ICDE: This module controls and optimizes the radio's performance by combining rule-based and AI-based algorithms. It uses information such as the spectrum samples from the RF-MON, the slot allocation status and link stats from MAC, traffic flow information from the UDM, and the reports from the protected incumbent and other CIRNs from CI. After combining and consuming the data from these sources, the ICDE module intelligently and dynamically adapts the radio parameters at different layers to improve performance, increase spectrum efficiency, and collaborate with other networks to increase spectrum reuse.

Although the SCATTER CIRN design did not follow any 5G standard, their functional modules can be positioned inside 5G RAN architectures like the one proposed by the O-RAN Alliance [89]. As an example, the ICDE module would be deployed inside the RAN Intelligent Controller (RIC) near-real-time (near-RT) layer as either a standalone third-party application or run as separated AI-trained models that change the behavior of other control functionalities such as the scheduling policies and interference management.

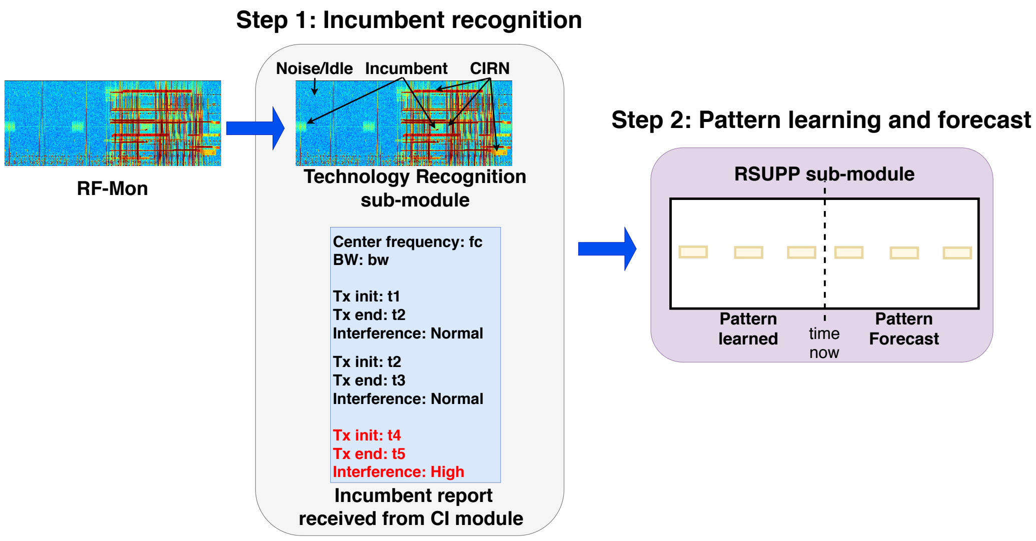


Figure 4.3: Proposed two-step AI-based algorithm for incumbent protection.

4.3 Incumbent protection in SCATTER

Although the DFS scheme is simpler and easier to deploy than CBRS and LSA, it lacks 1) a mechanism to exploit the spectrum not used by the incumbent and if it does not work correctly, 2) a mechanism to incentivize and enforce the incumbent protection. To overcome these limitations, the two-tier sharing spectrum scheme with CIRNs requires that the radios can:

- Share the spectrum in the sub-bands not used by the incumbent while transmitting.
- Share all the available spectrum while the incumbent points away from their radios dynamically.
- Use collaborative information to minimize the probability of harming the protective incumbent and the uncertainties about how others use the spectrum.

In other words, CIRNs need to detect and identify radio technologies, select the right policy to execute according to the identification, and be proactive in maximizing the spectrum usage efficiency. SCATTER radios implement two AI-based sub-modules inside their ICDE that provide identification and spectrum usage pattern prediction to support it.

TR: This sub-module recognizes spectrum signatures of different radio technologies and idle (noise) separately. As input, it consumes the continuous stream of spectrum data collected by the RF-MON module and outputs if a given technology is present in some *spectrum voxel*, where a spectrum voxel is a geometrical realization of the spectrum in terms of time, frequency, location, and power.

Repeated Spectrum Usage Pattern Prediction (RSUPP): Under the assumption that some technologies, like radars, use the spectrum following a fixed time-frequency pattern, this sub-module implements an algorithm that learns and forecasts the pattern of the future incumbent's transmissions in real-time.

Figure 4.3 depicts how these two sub-modules interact to form the proposed two-step algorithm described next. In the first step, the spectrum data provided by the RF-MON module is processed by TR to identify the incumbent's transmissions in the vicinity. This information creates a 2D-binary grid marking the spectrum voxels where the incumbent is detected. Next, this grid is forwarded to the RSUPP sub-module to perform the second step. In this step, the RSUPP 's algorithm learns any periodic pattern of the incumbent and uses it to forecast its future transmissions.

However, what happens if the TR cannot recognize the incumbent due to other users interfering with it? In this case, the probability of recognizing the incumbent by the TR sub-module drops drastically. It is at this point where collaborative information plays a fundamental role during step 1: by augmenting the view generated by TR with spectrum power measurements at the incumbent side, we can reduce the uncertainty about the incumbent existence when TR can not identify it because other interference signals are hiding the incumbent. The only requirement for these spectrum power measures is to provide enough information to determine where (frequency) and when (time) the incumbent suffers interference.

Once the RSUPP has learned the incumbent transmission pattern, it forwards this information to the **Incumbent Protection Policy (IPP)** sub-module inside the ICDE. This sub-module is responsible for translating the learned pattern into a policy to be executed by the MAC layer so as not to interfere with the protected incumbent. Precisely, the IPP controls the duty cycle of the SCATTER voxels/slots that overlap the incumbent's transmission according to the predicted pattern.

Notice that the effectiveness of this two-step algorithm relies on two facts. The first one is that the traditional incumbents, e.g., radars, have waveforms that can be identified and transmitted by following patterns that can be learned. Otherwise, our approach, and any AI-based algorithm, will fail. SCATTER also implements a Deep Learning (DL)-based algorithm inside the ICDE to forecast short-term spectrum occupancy following a more generic approach [53, 15]. However, the algorithm proposed in this work has low complexity and does not require offline training to forecast the incumbent's transmission pattern, which are two requirements to shorten the time to start protecting the incumbent.

The second one is that mechanisms exist to incentivize the appropriate collaboration among different technologies towards a common goal [90]. In our proposal, the primary source of incentives to protect the incumbent is the implicit gain in performance by reducing the interference between the incumbents and the CIRNs. However, this framework also allows the use of an explicit mechanism to enforce the incumbent protection by the CIRNs via collaboration, such as the scoring mechanism implemented in the SC2 (see Section 4.5) or the one presented in [90]. For more comprehensive literature on mechanisms to enforce cooperation and collaboration for dynamic spectrum sharing, we refer the reader to [91].

4.4 System Implementation

During the implementation phase, two main challenges were faced:

- How to identify the protected incumbent in near-real-time?
- How to learn the incumbent's pattern in near-real-time but adapt if it changes dynamically?

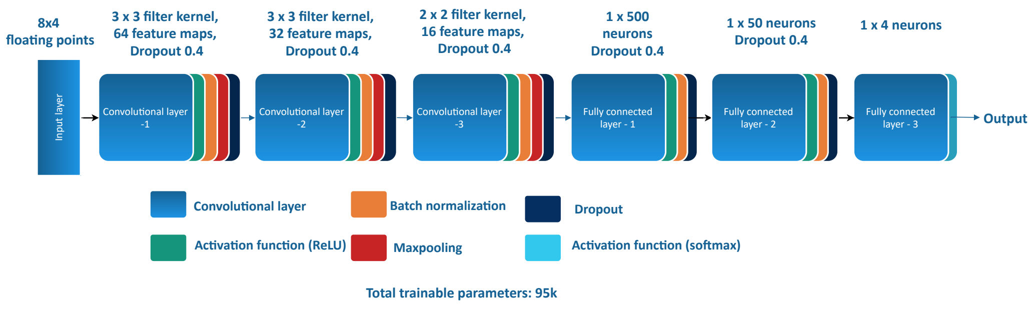

8x4 floating points
3 x 3 filter kernel, 64 feature maps, Dropout 0.4
3 x 3 filter kernel, 32 feature maps, Dropout 0.4
2 x 2 filter kernel, 16 feature maps, Dropout 0.4
1 x 500 neurons Dropout 0.4
1 x 50 neurons Dropout 0.4
1 x 4 neurons
Input layer
Convolutional layer -1
Convolutional layer -2
Convolutional layer -3
Fully connected layer - 1
Fully connected layer - 2
Fully connected layer - 3
Output
Convolutional layer
Batch normalization
Dropout
Activation function (ReLU)
Maxpooling
Activation function (softmax)
Total trainable parameters: 95k

Figure 4.4: Convolutional Neural Network (CNN) model used for TR. It only requires 95k parameters, which is translated in short training time and fast prediction in run time.

In Chapter 3 we showed that CNN architectures could classify radio signals with high accuracy. However, they are not designed to provide near-real-time operation (below 1 second). To address this problem, we design several pre-processing steps to reduce CNN 's complexity for TR. Figure 4.4 shows the implemented CNN architecture. This architecture comprises three convolutional layers, followed by three dense ones. All the layers have rectifier activation functions except the last one, which has a soft-max function for classification. Max pooling is used in convolutional layers for down-sampling the input, while dropout and batch normalization layers are used in convolutional and dense layers to improve generalization and accelerate training.

The resulting model contains about 95k trainable parameters, equivalent to only 20% of the model's trainable parameters presented in Chapter 3 ($95k$ vs. $500k$). The following design choices achieve the low complexity for this model. First, we use averaged FFT samples, which provide a frequency-only spectrum view, instead of raw IQ ones, which give a time-only spectrum view. This operation, implemented in the RF-MON, allows us to use up to 32-averaged 512-point FFT samples while still recognizing the incumbent, i.e., 32x lesser examples to be processed in TR. We use the incumbent's frequency operation information shared via collaboration to compensate for the losses in frequency resolution due to averaging. Second, we align the samples with the SCATTER's MF-TDMA frames before TR receives them, so TR 's processing requirements are lowered. Third, we group all the CIRN radio signatures under the same class, resulting in only four categories: Incumbent, SCATTER, other CIRNs, and idle/noise. Finally, we recognize technologies per SCATTER's channel. As SCATTER uses 7(14) channels of 1.4Mhz in 10(20) MHz band, the CNN input is only 32 floating-point values. TR was implemented in Python and used the TensorFlow library for the CNN model[1].

[1]https://www.tensorflow.org/

With a sample rate of 23.04 Msps, time-slots of 25ms, and a frame of 500ms, each SCATTER voxel contains 35 32-FFT samples. To maximize the incumbent's protection and minimize the number of false-positives identifications per voxel, we define that an incumbent is detected in a SCATTER voxel if more than 15% of the samples are classified as an incumbent with accuracy above 99%.

In scenarios with 10MHz bandwidth, the implemented TR sub-module outputs a 2D-Boolean grid of size 7x20, where the value one means the incumbent is detected in a given SCATTER voxel, and zero otherwise. This grid is sent to the RSUPP sub-module, which simultaneously creates another 7x20 2D grid aligned with the SCATTER's frame using the timestamped reports received from the incumbent. As the RSUPP also receives the grid created by TR, then it combines both grids using an OR operation. To decrease the RSUPP 's computational requirements even more, we learn the incumbent pattern only on the SCATTER's channels that overlap the incumbent transmission. Let us describe how the RSUPP sub-module learns the incumbent's pattern.

We create a boolean string for each channel on which the prediction will be executed. The last string characters are used as a prefix to search in a buffered string that contains a fixed number of previous strings. This buffer creates a probability tree representing all possible incumbent states after the specified prefix. Figure 4.5 shows three possible trees with three different prefixes based on a buffered string, where character A means that the incumbent is detected in that SCATTER voxel, and X otherwise. The root of each tree is the selected prefix, and each node sequence from the root represents sub-strings with the same prefix. Links between nodes are labeled by the number of sub-strings with the same character sequence. In this implementation, we set the buffer size to 1000, the depth of the tree search to 400, and the pattern to find will have a string length between 80 and 400.

This process is done for all the prefixes starting from the end string (reading backward). Notice that increasing the prefix length reduces the number of branches in the probability tree. The algorithm combines the different trees, and if the first n children are reachable with a probability higher than a given threshold following the greedy path, the algorithm determines that it has found a pattern candidate. If the same pattern is predicted a few times in a row, it is used to avoid the incumbent on a given channel. Once a pattern is found, we fix it and use it until we detect changes. Similarly, if the predicted pattern does not match the learned one a few times in a row, we assume the incumbent pattern has changed, and a new one will replace the old pattern.

Notice that the predicted and the learned pattern will not match when 1) the incumbent changes its transmission pattern, or 2) the TR sub-module is not able to detect the incumbent due to interference and no information is received from the incumbent reporting that interference, e.g., due to high packet latency. In the last case, the predicted pattern is generated using only data from the blind TR. In this implementation, we set the times a new pattern candidate has to be found consecutively to replace the old one to three. However, in the case of non-reliable and congested collaboration networks, this value may be increased to maximize the probability of detecting the incumbent with TR before replacing the previously learned pattern.

The described algorithm is a slightly modified version of the Context Tree Weighting (CTW) Markov predictor for binary sequences [92]. The main modification was implementing a circular buffer to detect the periodic pattern of the incumbent's transmission.

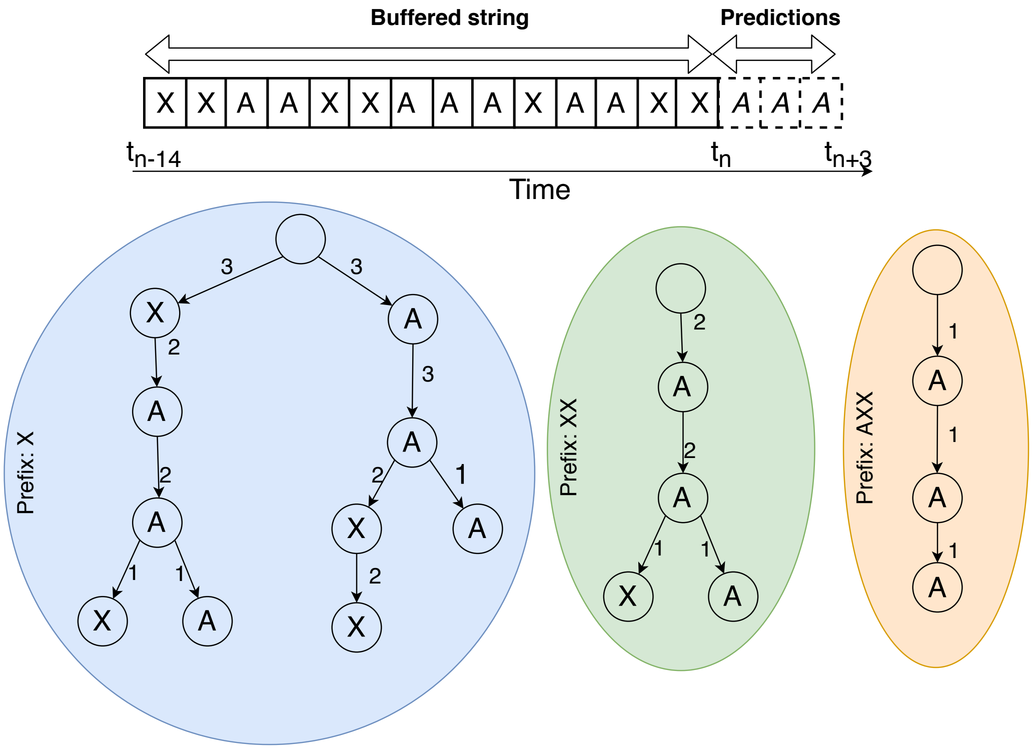


Figure 4.5: Example of creating a probability tree based on a given incumbent detection buffer, where A represents that the incumbent was detected during that specific time slot and X otherwise.

Compared to other approaches used for spectrum prediction, e.g., predictors based on Hidden Markov Model (HMM) [93] or Recurrent Neural Network (RNN)[94], our predictor supports online training and has both low time and space complexities with theoretical performance guarantees [92]. The RSUPP sub-module was implemented in C++.

4.5 Experimental validation

The SC2 challenge provided a scenario to test the CIRNs capabilities to coexist and protect the incumbent. In this scenario, CIRNs were challenged not to harm a Terminal Doppler Weather Radar (TDWR) system, i.e., the incumbent, while sharing and exploiting the available spectrum. This scenario lasted 330 seconds, where all the CIRNs shared 10 MHz of bandwidth. The scenario comprised up to five CIRNs, where one node per network, the gateway, was connected to the collaboration network.

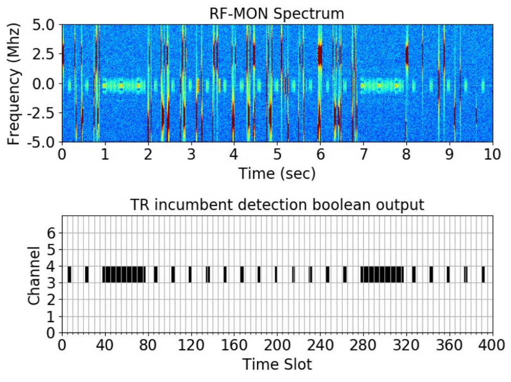
The incumbent to-be protected shares the following information via an implemented collaboration protocol called CIRN Interaction Language (CIL)[95, 96]: incumbent's current Signal-To-Interference-Plus-Noise Ratio (SINR) measurement (dB); a threshold that sets the minimum SINR value on which other transmissions will start harming it; a violation flag that is activated when the threshold is breached; and incumbent's center frequency and bandwidth. The CIL protocol includes two critical design choices to guarantee the scalability of the information sharing: 1) short messages and 2) a limited rate to produce them. For example, an incumbent report has a size less than 0.5Kbit and is generated at 10Hz. Generally, any information shared by peers via the collaboration network was generated with rates between 0.1 and 10Hz, translating into proportional growth to the number of sharing peers.

CIRNs follow a simple procedure to access the network. Once the scenario starts, CIRNs and the protected incumbent(s) register to a collaboration server. Then, peer-to-peer communication is established among CIRNs and the incumbent to begin collaborating. Data plane traffic that CIRNs have to route among their nodes is rewarded with points if their requirements of QoS hold for a determined period. The competition was designed to award the CIRN that scores the most points. However, CIRNs were responsible for computing their score and getting the score of other CIRNs via the collaboration network. This scoring system has two main properties: 1) if any CIRN is scoring points below a threshold, then all the CIRNs will receive the same score as the weakest CIRN, and 2) if the incumbent reports that it is harmed in a period, then the CIRNs get no points during that period. While 1) is used to promote collaboration among CIRNs, motivating them to give spectrum to the weakest network, 2) is used to enforce and incentivize the CIRNs to protect the incumbent.

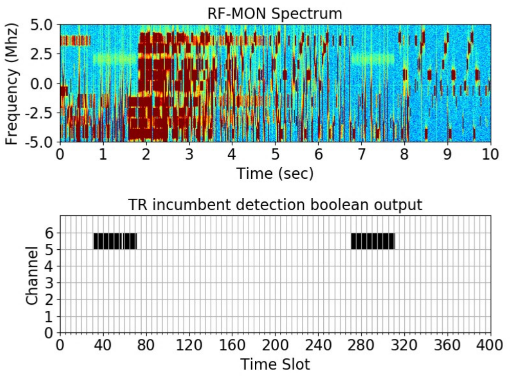
The scenario had three difficulty levels and two incumbent bandwidths. The three levels were *easy*, where the incumbent follows a slow transmission pattern; *medium*, where the incumbent transmits shorter and faster than the easy one; and *hard*, where two incumbents are transmitting in the same band, one with the easy and the other with the fast pattern. For each difficulty, the incumbent could either use the entire available bandwidth, ensuring no other CIRNs can transmit simultaneously, or use half of the available bandwidth so that other CIRNs can transmit in the other half of the spectrum. This section presents two of the executions of this scenario where SCATTER was involved as a) a single CIRN and b) collaborating with multiple CIRNs. For visualization purposes, the relative transmission power of the incumbents received by RF-MON was modified (increased) to allow its visualization.

4.5.1 SCATTER protecting the incumbent alone

In this experiment, SCATTER CIRN was deployed alone to protect two incumbents transmitting in the same channel (hard-level scenario) and using the total bandwidth. Figure 4.6 (a) shows the spectrum view provided by RF-MON and the Boolean 2D grid that the TR module forwards to the RSUPP module. Note that although the incumbents were using the whole bandwidth, it was only detected in the channel overlapping its center frequency. This is a side effect of the average applied to the FFT samples by RF-MON to reduce the amount of processed data. However, the incumbents' reports provide lost information (frequency operation data). Nevertheless, it does not impact the capabilities of TR to recognize the different incumbents correctly. SCATTER radios could learn the incumbent patterns and schedule their transmissions between the transmissions of the incumbents. In this experiment, 98% of the future incumbent's transmissions were correct, and most of the mistakes were at the beginning of the scenario where the RSUPP started to learn.



(a) SCATTER CIRN and two incumbents.



(b) Multiple CIRNs and one incumbent.

Figure 4.6: CIRNs collaborating and protecting the Incumbent. Spectrum received from RF-MON (top) and the incumbent detection output from TR (bottom) aligned to the SCATTER MF-TDMA scheduling (1.4 MHz channels, 25ms per time slot).

4.5.2 SCATTER and multiple CIRN protecting the incumbent

In this experiment, three CIRNs were deployed, including SCATTER CIRN, to protect one incumbent while using half of the available spectrum in the easy-level scenario. Figure 4.6 (b) shows how multiple CIRNs were able to protect the incumbent while maximizing the use of the spectrum. Depending on the traffic demands, CIRNs try to use the available spectrum left by the incumbent efficiently. For example, between seconds 0 and 4, high traffic demand is managed by the CIRNs. However, while the incumbent was transmitting, no CIRNs scheduled transmissions in the 5MHz band (upper band). Note that part of the incumbent was hidden due to a side-band transmission of another radio (time slots between 50 and 60). As a result, TR could not identify the incumbent there. However, this gap was well predicted by the RSUPP as an incumbent transmission. In general, 95% of the future transmissions of the incumbent were predicted correctly. This slight accuracy drop is expected since coexisting with other networks introduces a longer learning time.

4.5.3 Execution time performance

On average, the two-step algorithm took less than 300ms. TR used 200ms to format the pre-processed data received from RF-MON, i.e., transforming raw bytes into python arrays, 100m for performing the TR task, and 50ms to predict the transmission pattern of the incumbent for the next SCATTER's frame. As the RSUPP is implemented in C++, it can efficiently process the received raw bytes. Finally, learning a pattern varies, on average, from 5 seconds in solo scenarios to 30 seconds in coexistence with other CIRNs.

4.6 Conclusions

We presented the architectural design and the experimental validation of an incumbent protection system for the next generation of spectrum-sharing frameworks. Built on top of the concept of CIRN, we designed and implemented a two-tier framework that enables efficient spectrum sharing while protecting the incumbents. Compared to new approaches like CBRS and LSA, our system requires no central infrastructure to control and grant access to the shared spectrum. It only requires that the incumbents collaborate with other networks by sharing information such as location, interference measurements, and frequency operation parameters.

On the radio side, we proposed and implemented an AI-based two-step algorithm that protects the incumbent autonomously. This algorithm uses spectrum data and information the incumbent provides to recognize, learn, and proactively predict the incumbent transmission pattern with an accuracy above 95% in near-real-time. We have experimentally validated the proposed architecture and algorithm using Colosseum, the world's largest Radio Frequency (RF) channel emulator built for the SC2 challenge.

Chapter 5

A General Approach for Traffic Classification in Wireless Networks

The content of this chapter has been partially published in:

- M. Camelo, P. Soto, and S. Latré, "A General Approach for Traffic Classification in Wireless Networks using Deep Learning," in IEEE Transactions on Network and Service Management, doi: 10.1109/TNSM.2021.3130382.

In the previous chapter, we demonstrated that the successful deployment of wireless networks capable of meeting the growing demands of emerging applications and services while ensuring Quality of Service (QoS) depends on the radios' ability to perceive their spectrum environment intelligently, enabling efficient sharing and reuse. The Technology Recognition (TR) module, initially introduced in Chapter 3, has played a vital role as the first step in the cognitive radio cycle and has given the foundation for the development of the Collaborative Intelligent Radio Networks (CIRNs) framework, as showcased in Chapter 4. In this chapter, we introduce a novel framework that empowers the Intelligent multi-RAT (ImRAT) Gateway (GW), a type of Intelligent Radio (IR), to perform Traffic Classification (TC) at any layer within the radio stack, leveraging spectrum data. Additionally, we propose a Deep Learning (DL) architecture for the second step of the cognitive radio cycle, encompassing learning and reasoning. This approach overcomes the limitations of traditional TC systems, which typically assume that all traffic belongs to the same network domain and rely on byte/protocol representations of packets at the Link Layer (L2) (or above).

5.1 Introduction

In the realm of Cognitive Radio (CR) networks and spectrum sharing models, traditional Network Monitoring Service (NMS) have long played a vital role in analyzing network behavior and traffic patterns [97, 98, 99, 99]. These services have historically been instrumental in identifying bandwidth-intensive applications and critical network links, enabling effective decision-making for network management and the assurance of Quality of Service (QoS) for users. With the integration of Machine Learning (ML) techniques, NMS has evolved to encompass automated network traffic analysis, including network state predictions, anomaly detection, malware detection, and Traffic Classification (TC) [100].

TC, a central component of network management, involves determining the origin of network traffic, thereby allowing the inference of the specific application responsible for generating that traffic [100, 101]. The ability to classify traffic into distinct categories provides the foundation for enforcing tailored security and QoS policies. Over time, various approaches have been developed to keep pace with advancements in user applications and communication protocols. In recent years, Deep Learning (DL)-based TC systems have emerged as superior alternatives to traditional methods such as port-based classifiers, Deep Packet Inspection (DPI), and statistical ML-based flow analysis, even in scenarios involving encrypted traffic [102, 103, 104]. However, the traditional TC task is assumed to be performed on traffic that belongs to the same network domain and over a byte/protocol representation of the packet at the Link Layer (L2) (or above). These assumptions limit the capabilities of TC systems in wireless networks using shared spectrum, e.g., in unlicensed bands. This is because users' traffic from one wireless network domain can be negatively impacted by traffic transmissions from other wireless networks without being noticed by the TC system as demonstrated in [30].

Co-located transmissions within the same spectrum band in wireless networks can produce Physical Layer (L1) packets that escape detection by receivers tasked with TC. These situations occur when transmitters utilize incompatible wireless technologies or when decrypted traffic is already secured at the L2 within a distinct network domain. To overcome these limitations, a shift from TC systems operating at the byte level to TC systems functioning at the spectrum level becomes necessary. This approach allows for monitoring, detecting, assembling, and classifying traffic generated by various wireless devices sharing the same spectrum, even when the traffic is encrypted, originates from different network domains, or employs diverse wireless technologies. However, this transition introduces new challenges not present in byte-based TC systems. Spectrum-based packets are subject to modulation, coding, and encryption before transmission, resulting in distinct spectral representations of packets. These variations can occur even within the same wireless technology due to differences in Modulation and Coding Schemes (MCSs) or when different digital multi-carrier transmission schemes are employed.

To this extent, we introduce a novel framework that achieves TC at any layer of the radio network stack, relying on the Technology Recognition (TR) module proposed in Chapter 3 for wireless technology-agnostic spectrum sensing capabilities. Building on top of it, a procedure based on DL to perform TC on spectrum samples is proposed. This procedure enables the management algorithms running at the Gateway (GW) nodes (or beyond) to perform better by having a broader view of the traffic flowing in the shared spectrum. Two DL-based architectures were designed to solve the task of classifying packets directly on spectrum data. To evaluate the performance of the architectures, we create and provide an open source data set containing 802.11 standard-compliant L1 waveforms, the first open and available data set for testing traffic classification at the spectrum level. We believe this data set would foster reproducibility and allow further advances on this topic. The data set and the code associated with this chapter can be obtained in Zenodo[1] and Github[2], respectively.

[1]`https://doi.org/10.5281/zenodo.5208200`
[2]`https://github.com/miguelhdo/tc_spectrum`

The remainder of this chapter is structured as follows. We present the related work in Section 5.2. The general framework for TC is introduced in Section 5.3, and the proposed spectrum-based TC algorithm using DL is presented in Section 5.4. Finally, we show the DL models' performance evaluation results on both coarse-grained and fine-grained traffic classification tasks in Section 5.6 and conclusions in Section 5.7.

5.2 Related work

This section presents some of the most relevant work on TC using DL approaches for encrypted traffic. For a more exhaustive literature review on the general applications of DL in wireless networks and on ML/DL approaches for TC, we refer the reader to [103] and [102, 105], respectively.

5.2.1 Traffic Classification using L2 (and above) classification objects

Over the years, the applications have evolved, and so have the algorithms and techniques used to classify the traffic generated by them [106, 101, 107, 102, 100]. The initial approach was using port numbers. Later, traffic classifiers using DPI techniques were designed to find patterns in the data packets' payload. While port-based classifiers are more accessible and faster than DPI methods, DPI outperformed them at the cost of higher computational requirements. Unfortunately, both approaches are limited to non-encrypted traffic that typically belongs to the same network domain. To avoid this problem, ML approaches were proposed to classify the traffic using packet flows, where flow measurements are used as features. However, it has been shown that its accuracy can be affected by user behavior variations, device OS-specific patterns, and network-specific conditions, among others [107].

Table 5.1: Comparison of our work and other contributions focusing on TC using DL.

Contributions	**Traffic Representation**	**Classification at any layer**	**Classification object as input data**	**End-to-End framework**	**Proposed Model**	**Comparison against other models**	**Comparison against other traffic representation**	**Multi-task learning**	**Data set and availability**
[15]	L2 Packet flow	No	Flow statistics as images	Yes	2D-CNN	SVM, MLP, NB, DT	No	No	Private
[16]	L2 Packet flow	No	Session and Flow statistics	Yes	1D-CNN	2D-CNN	No	No	Public
[17]	L2 Packet flow	No	24 statistical features of a packet	Yes	1D-CNN	1D-CNN trained in supervised mode	No	No	QUIC, Public
[18]	L3/L4 packet	No	Raw bytes	Yes	SDAE, CNN, LSTM-RNN	RaF, MLP	No	No	Restricted
[19]	L2 Packet flow	No	Sequence of packet statistics	Yes	CNN+LSTM-RNN	CNN, LSTM-RNN	No	No	RedIRIS, Public
[20]	L7 packet	No	Raw bytes as images	Yes	VAE	No	No	No	IMT17, Public
[21]	Raw DCI	No	Transport Block Size	Yes	AE + softmax +dense layer.	LSTM-RNN+softmax, LSTM-RNN+dense layer	No	Yes	Private
[23]	Raw DCI	No	Transport Block Size	Yes	LSTM-RNN, 1D-CNN, MLP	SVM, LR, K-NN, RaF, GP	No	No	Private
[8]	L1 packet	Yes	Raw In-phase and Quadrature (IQ) samples	No	LSTM-RNN	No	No	No	Private
[10,24]	L1 packet flow	Yes	Raw IQ samples as images	No	2D-CNN	2D-CNN on raw L2 packets	Yes, raw L2 packet	No	Private
This work	**L1 packet**	**Yes**	**Raw IQ samples**	**Yes**	**2D-CNN**	**GRU-RNN, GB**	**Yes, raw L2 packet and packet length**	**No**	**IDLAB-TC-SPECT, Public**

More recently, TC algorithms based on DL approaches have outperformed ML ones based on traditional algorithms [102, 100, 105]. One example of these algorithms is shown in [108] and [109], where authors proposed an end-to-end TC algorithm based on Convolutional Neural Networks (CNNs) that converts raw traffic into images. In [108], the authors took the first several packets of the traffic flows and their time-series features and identified the application or protocol type that generates them. The Seq2Img model was compared against four popular classifiers such as Support Vector Machines (SVM), Multi-Layer Perceptron (MLP), Naïve Bayes (NB), and Decision Trees (DT). Results showed that all approaches perform equally well when classifying protocols, but Seq2Img is almost 12% more accurate than other models when classifying applications.

To overcome the problem of data labeling, authors in [110] proposed a semi-supervised approach that pre-trains a 1D-CNN model on an unlabeled data set to infer traffic patterns and afterward re-train the model on a labeled data set to confirm those patterns. This way, the amount of labeled data needed in the second step is considerably reduced. The datasets are based on time-series features of a fixed number of sampled packets from traffic flows. Results showed that the pre-training step increases the model's accuracy by up to 10% compared to a model without pre-training. A Stacked Denoising AutoEncoder (SDAE), a CNN, and a Long Short-Term Memory (LSTM) were proposed as classifiers in [111], where Netlog was developed to simplify the data labeling process. Comparison against a Random Forest (RaF), which requires statistical features, and an MLP showed that all DL models performed over 20% better than RaF in terms of accuracy, showing that much more insightful features can be learned from raw data than from the statistical features used by the RaF.

A combination of two CNN layers followed by one LSTM layer with two fully connected layers at the end was proposed in [112]. The time-series features are taken from the headers of the first 20 packets exchanged during the flow lifetime and did not include any information that could identify users (MAC/IP addresses) to ensure data confidentiality. Results showed that the combination obtained the best results in terms of accuracy and F1-score. Autoencoders (AEs) are also a predominant option as models. For instance, in [113], a network traffic flow is also transformed into an image later processed by a semi-supervised model based on a Variational AutoEncoder (VAE). The authors took data from the HTTP sessions (requests and responses) and converted them to a 28x36 image. The proposed VAE uses an MLP encoder and decoder that analyses images in an unsupervised manner as a feature extractor. Then, the extracted features are mapped to an app in a supervised manner. Results show that, even with two features, the network traffic can be effectively discriminated in the unsupervised step, achieving an accurate classification at the supervised step.

As pointed out by the authors in [114], most of the literature in the field of NMS is focused on single-task learning, e.g., each model is designed and trained to solve one specific learning task such as TC, traffic prediction, or anomaly detection. As a solution, Multi-Task Learning (MTL) approaches have been proposed in [114] and [115], where TC is used as one of the learning tasks to leverage helpful information contained in multiple related tasks aiming to improve the generalization capabilities of all them while learning. The authors in [114] used a MTL approach to jointly solve the TC and traffic prediction task using traffic traces containing the Downlink Control Information (DCI) messages carried within the LTE Physical Downlink Control CHannel (LTE-PDCCH) with a time granularity of 1ms. Their proposal is a two-step procedure where they first use an AE to extract common feature representations among tasks and then use the encoder part of it to train a traffic classifier (softmax layer with a softmax activation function) and a traffic predictor (a dense layer with the rectified linear unit (ReLU) activation function) simultaneously. Compared to conventional single-task learning approaches, which do not use AE and tackle classification and prediction tasks separately, the MTL approach always provided the highest performance.

More recently, the authors in [116] proposed a CNN architecture to perform TC without having to decode and/or decrypt any of the transmitted flows. To achieve this, the input to the model is raw physical control channel messages of a mobile network. The input data is obtained by decoding the DCI messages carried within the LTE-PDCCH. Among the information carried by the DCI messages, the authors used the number of allotted resource blocks, the MCS, and the transport block size. The authors claimed that this information should provide sufficient information for learning algorithms to classify the application and service the user is running reliably. The results confirmed such claim and the proposed CNN achieved an accuracy above 98%, outperforming other DL models such as Recurrent Neural Network (RNN) and traditional ML approaches such as SVM and RaF, Learning Rate (LR), k-Nearest Neighbours (K-NN), and Gaussian Processes (GP).

5.2.2 Traffic Classification using L1 classification objects

All the previous works are byte-based approaches, which limits their application on wireless networks (see Section 5.3.1). As a solution, a few spectrum-based traffic classifiers have been proposed in recent years to perform TC on raw spectrum data. Authors in [117] present a DL-based algorithm that classifies traffic patterns of different types of applications directly from the radio spectrum with accuracy $\geq$ 96% and outperforms state-of-the-art methods based on Internet Protocol (IP) packets with DL. They use images representing the spectrum in time and time-frequency as input data for their CNN-based DL architecture. An extension of this work was presented in [118], where a validation with real-life data showed that a model trained with synthetic data could discriminate between different traffic patterns but with a decreasing performance in terms of accuracy. One advantage of this approach was the automatic extraction of the time-dependent features to perform the TC task. However, in contrast to byte-based methods using statistical data from traffic flows, this approach assumes that the spectrum patterns to be classified belong to a single-user and single-flow, which is not the case in real environments.

An alternative to overcome the limitations of [117] is to use L1 packets and perform the TC directly on them. Based on a RNN architecture, the authors in [119] showed that TC on raw spectrum data could be performed on short time-series (a few hundred samples) with an accuracy $\leq$ 85%. This accuracy can be considered low compared to byte-based TC systems if we also consider that the L1 packets were single-modulated with no coding, non-encrypted, and transmitted with a low data rate. One of the reasons for this performance is the use of RNN architectures, such as LSTMs [120, 121], which suffer from inefficient training and low accuracy with large data sequences [122, 123, 124].

Given the limitations of the previous spectrum-based proposals, this chapter introduces a general framework to achieve TC at any layer in the radio stack. On top of this framework, we propose an end-to-end spectrum-based TC procedure that is also Radio

Access Technologies (RAT) agnostic. At the heart of the procedure, we design, train, and evaluate a CNN-based classifier that outperforms an optimized RNN architecture, similar to the one used in [119], by achieving higher accuracy, even on large data sequences, and lower complexity in terms of prediction time. Moreover, the proposed approach removes the need for specialized algorithms for traffic flow separation/aggregation on spectrum data like the one required in [117] as it uses as classification objects the IQ-samples associated to one single L1 packet, which can be realized with simpler algorithms. This framework will allow decision-making engines to enhance the view of the traffic that is passing through the gateway nodes, not only traffic from the same network domain (of which we can obtain a byte representation) but also traffic that is generated by any nearby wireless device and captured directly on the shared spectrum. Table 5.1 provides a comparison of the contributions of this chapter and the analyzed related works that perform TC using DL.

5.3 A general framework for Traffic Classification

This section introduces the main limitations of the byte-based approaches for TC in wireless networks using a shared spectrum and presents a general framework to perform TC at any layer of radio network stack using a spectrum representation of a packet. This framework provides the building blocks to design a DL-based traffic classifier for wireless networks. In the rest of the chapter, the terms spectrum-based packet representation and L1 packet are used as synonymous, similar to byte- or protocol-based representation and L2 (or above) packet.

5.3.1 Limitation of the byte-based frameworks for Traffic Classification

Deploying wireless networks that can handle the increasing demands on network capacity of new applications and services while guaranteeing their QoS requirements will depend on the radios' capabilities to be aware of their spectrum environment, sharing and reusing it optimally. Therefore, including capabilities to sense and understand the environment state is fundamental to dynamically adapting the radio parameters to fulfill users' requirements while optimizing the shared spectrum usage [125, 48].

Large deployments of daily use technologies such as Wi-Fi and 4G/5G are based on simple devices at the users' side and a full management stack at the RAT GW node or beyond. One of the management system's key components is the NMS, which provides various information related to the traffic generated by the users accessing the operator's core network. These services have recently been enhanced with ML techniques to perform automatic network traffic analysis with high accuracy [100, 103, 102]. One of these traffic analysis tasks is TC, which allows inferring the user's application generating the traffic to enforce specific security and QoS policies on it.

To picture the use of TC on wireless networks, let us consider a co-existence scenario in a shared spectrum between two wireless networks, one using Wi-Fi and the other using a private 4G/5G deployment, as shown in Figure 5.1. In this scenario, the management system running on the private 4G/5G deployment wants to prioritize the traffic of user

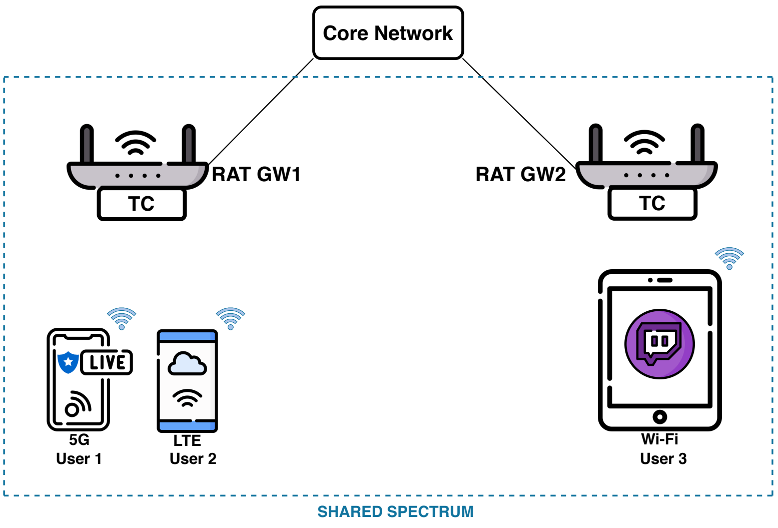


Figure 5.1: Use case scenario where the traffic analysis will be incomplete if a byte-based TC system is used.

1 over user 2 dynamically so it can enforce a given QoS while maximizing the use of its network resources. The GW nodes can perform TC to detect and identify the type of application being executed at any moment by its users. On the other hand, user 3, which belongs to the Wi-Fi network, generates a large traffic volume using a non-priority application. For simplicity, let us assume that a central entity manages both GWs. However, this can easily be extended to independent management domains where information exchange is allowed to enforce collaboration or/and cooperation among them, as demonstrated in our previous work [30].

Traditionally, byte-based TC systems will be located at (or behind) the RAT GW and will classify the traffic sent by the users' devices in the wireless domain of the RAT GW. In the example scenario, GW 1 can classify traffic from users 1 and 2 and use this information to enforce traffic policies, e.g., GW 1 can determine to reduce the bandwidth assigned to user 2 to guarantee QoS on the protected user 1. However, independently of the traffic policies applied at GW 1 to protect user 1, the performance of user 1 can be negatively impacted by the non-priority traffic generated by user 3. This is expected as the GW 1 can not see the traffic generated by user 3 as it will never pass through. Even with a byte-based TC system working on all the users' traffic, it would be difficult to infer that the traffic generated by user 3 negatively impacts user 1 without knowing the users' location.

5.3.2 A Traffic Classification framework at any layer

One possible solution to the abovementioned problem is that the GW 1 recognizes the radio technology that interferes and adapts its behavior accordingly [126, 127, 30]. However, this approach is not enough to increase spectrum efficiency as

1. the interfering/interfered technology may not include a mechanism to adapt or coexist, e.g., to change its frequency band to avoid the interference, and
2. traffic prioritization can be only based on technology and not on the application. Therefore, specific traffic policies such as priority vs. non-priority traffic cannot be applied.

For instance, if the GW 2 can recognize that some external device is transmitting a high-priority/protected traffic, it can enforce a policy over its user 3 to reduce the impact on that external device even if it is not under the same management domain as the other wireless device. Moreover, the TC task cannot be performed at L2 or beyond the central entity, as it requires specialized hardware to demodulate and decode the information of a given technology. Therefore, performing TC to its users' traffic and any other traffic in the same spectrum is crucial for the future deployment of complex and advanced wireless networks.

In general, wired and central-managed wireless networks, using (possibly) multiple technologies, can implement TC at L2 with little effort as all the users' traffic can be obtained at the access points/gateways of the networks and should be the logical choice for implementing a TC system. However, this task is more complex in decentralized multi-technology wireless environments as the traffic from multiple wireless devices flows through the same shared medium (the spectrum). In the case of multiple wireless technologies, it is required to have specialized hardware to demodulate, decode, and decrypt the messages sensed by the receiver to obtain the L2 packet for TC. In the case of the same technology, each wireless device may belong to different wireless domains, which are usually secured, so packets demodulated and decoded can be encrypted but at the cost of requiring more complex classifiers. As a result, a general framework to classify traffic directly at L1 is critical to address these wireless network limitations.

To this end, Figure 5.2 introduces a general framework that allows the classification of different types of traffic using spectrum representation of the data packet. The proposed framework comprises two main blocks: the traffic generators and the Intelligent multi-RAT (ImRAT) GW. Let us describe each one of these blocks in more detail.

Traffic Generators: In general terms, traffic generators are a combination of a user's wireless terminal and the applications that run on it to generate traffic and are transmitted over the spectrum to the ImRAT GW.

1. *User's Terminal (UT)*: This is a wireless device that runs the applications generating the traffic and transmits it to the ImRAT GW. Although these devices may be very complex and use advanced RATs, e.g., CRs [125] or Collaborative Intelligent Radios (CIRs) [15, 48], so they can create fully autonomous and distributed networks, in this framework we assume they are simpler devices that are connected to a (multi) RAT GW node that performs the management tasks.
2. *Applications*: This is any software that runs over the UT and generates traffic. A critical characteristic of these applications is that the generated traffic follows a pattern at any radio stack level. This requirement is fundamental as any Artificial Intelligence (AI)-based algorithm learning to discriminate among different classes needs to find patterns in the data used to learn. For example, a pattern can be learned from the application's protocol generating users' data traffic.

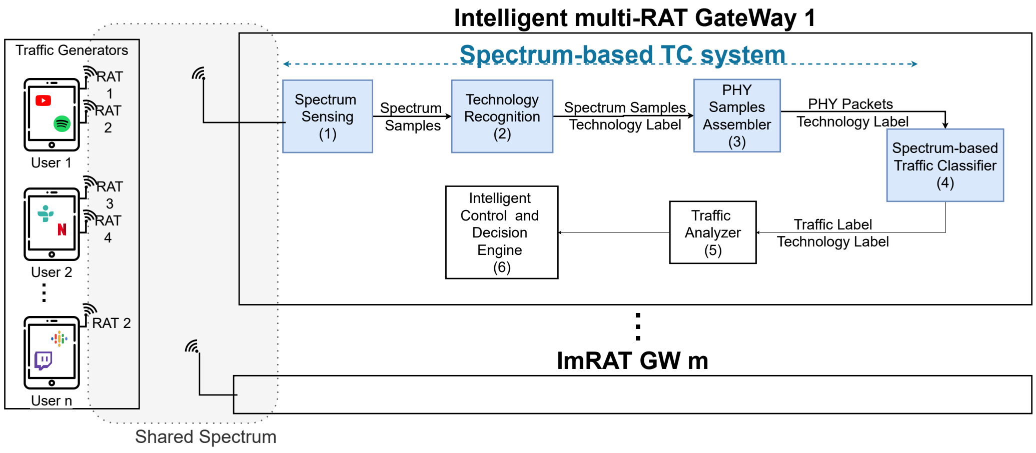

Intelligent multi-RAT GateWay 1
Spectrum-based TC system
Traffic Generators
RAT 1
RAT 2
User 1
RAT 3
RAT 4
User 2
RAT 2
User n
Shared Spectrum
Spectrum Sensing (1)
Spectrum Samples
Technology Recognition (2)
Spectrum Samples Technology Label
PHY Samples Assembler (3)
PHY Packets Technology Label
Spectrum-based Traffic Classifier (4)
Traffic Label Technology Label
Traffic Analyzer (5)
Intelligent Control and Decision Engine (6)
ImRAT GW m

Figure 5.2: Functional diagram of a general framework for traffic classification at any layer.

ImRAT GW: This is a wireless device that acts as a gateway to one or multiple RATs and uses AI-based algorithms to perform wireless management tasks such as the NMS. This device can also be integrated into more advanced Radio Access Network (RAN) architectures like OpenRAN [89] or open5G [128]. The following internal blocks are required for deploying spectrum-based TC algorithms.

1. *Spectrum Sensing*: This sub-module, typically found in CR, obtains samples of the spectrum. The data type will depend on the input data required by the spectrum-based TC. Some formats of these samples are the IQ (time domain), Fast Fourier Transform (FFT) (frequency domain), Short Time Fourier transform (STFT) (time-frequency domain), Continuous Wavelet Transform (CWT) (time-frequency domain). Note that although traditional radio transceivers do not provide access to the received spectrum data, radio platforms running on Software Defined Radio (SDR) [15] or software tools like Nexmon [129] can already provide access to it.

2. *TR*: This sub-module uses spectrum data to recognize the radio signatures of different RATs and idle (noise) separately (see Chapter 3 and [130]). As input, it consumes the continuous stream of spectrum data collected by the spectrum sensing sub-module and outputs if a given technology is present in some part of the spectrum.

3. *Multi-RAT L1 packet assembler*: This sub-module puts together different spectrum samples that belong to the same RAT and creates L1 packets. For this, the labels created by TR are used to find a packet pattern. Note that the spectrum sensing, TR, and this sub-module can be removed if we use the L1 packets already captured by the Wireless Network Interface Card (WNIC) of the multi-RAT before they are demodulated and decoded. However, this implies having one WNIC per technology. One advantage of the spectrum sensing + TR + Multi-RAT L1 packet assembler is that they are technology agnostic and can be easily extended to support new technologies without requiring adding new WNICs to perform the monitoring task.

4. *Spectral-based TC*: This sub-module uses L1 packets to classify the application that generated it. Compared to the byte-based TC, this approach is more complex and needs to be more robust. The features that discriminate traffic classes at different radio stack layers must be extracted from the raw spectrum. For example, a user-level application generating traffic will generally have a very similar IP packet carrying the application payload. However, the same packet at the L1 will be different depending on properties like the RAT (5G vs. WLAN), its version (Wi-Fi 5 vs. Wi-Fi 6), the MCS (BPSK 1/2 vs. QPSK 1/2), etc.

5. *Traffic Analyzer*: This sub-module takes the output from the TCs and automatically generates an analysis of the traffic flowing in the physical medium. The resulting analysis is used to enhance the decision-making engines' view controlling the radio parameters aiming to optimize a(n) (multi-)objective function.

6. *Intelligent Control and Decision Engine (ICDE)*: This sub-module combines the output of different radio systems, e.g., the output of the NMS and the QoS requirements of a given application/user, and intelligently and dynamically adapts the radio parameters at different layers to improve performance and increase spectrum efficiency. An example of this module is being presented in [48, 53].

Notice that although we did not follow any Wireless Local Area Network (WLAN) or 4G/5G standard to realize the ImRAT, their functional blocks can be positioned inside 5G-RAN architectures like the one proposed by the O-RAN Alliance [89] and coexist/coordinate with WLAN Access Points (APs) in the same network domain as established by the 3GPP release 16 with the introduction of the 3GPP Access Traffic Steering Switching and Splitting (ATSSS) [131]. As an example, the Spectrum Sensing, TR, and the L1 packet assembler blocks can be implemented in the Remote Radio Unit (RRU) for fast signal processing and spectrum-based TC empowered by ML models can be deployed in the RAN Intelligent Controller (RIC) co-allocated Distributed Unit (DU)/Centralized Unit (CU) to reduce the amount of data required to move L1 packets to the classifier. In decentralized environments, where multiple networks may run under their management domains, new approaches must be investigated and explored to define mechanisms to incentivize collaboration among networks and enforce policies that optimize global objectives shared by the networks in a fully automated fashion. Examples of such collaborative approaches are the Collaborative Intelligent Radio Network (CIRN), which were developed during the Defense Advanced Research Projects Agency (DARPA) DARPA Spectrum Collaboration Challenge (SC2)[14]. Some of our recent works have demonstrated the capabilities of such networks at the architectural level [15, 53] and validated their performance via experimental results [48]. Theoretical and practical developments of collaborative protocols to support such architecture can be found in [90], and [95, 96], respectively.

This framework allows the deployment of TC systems that work on the spectral representation of transmitted data and realize a general approach for TC for wireless networks at any layer from L1 to the Application Layer (L7). This framework is general as the first two blocks (TR and L1 packet assembler) provide a mechanism to be wireless technology agnostic since using an L1 packet as classification object allows a classification at any layer as this object contains the whole information carried by the transmitted packet. For end-to-end designing, training, and deployment of the ML/DL algorithms behind the sub-modules like TC and TR, this framework can be enhanced by the ideas presented in [100] (byte-based TC) and Chapter 3, respectively.

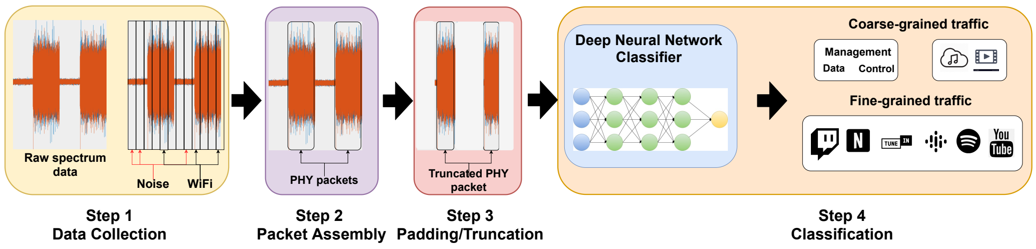


Figure 5.3: TC system using spectrum data. 4 steps compose the system: a) data collection, b) L1 packets filtering/assembly, c) zero padding or data truncation of the time series, d) Fine- or coarse-grained traffic classification.

5.4 Spectrum-based Traffic Classification system based on Deep Learning

TC at spectrum level using an L1 view of the packets enhances the traffic statistics provided by an NMS by including information from any traffic flowing in a shared medium such as a wireless link. However, L1 packets carrying the same payload can be completely different due to several factors:

- Different RATs may use various schemes to transmit (e.g., Orthogonal Frequency-Division Multiplexing (OFDM) vs. Direct-Sequence Spread Spectrum (DSSS)) and modulate/code the data (Binary Phase Shift Keying (BPSK) vs. Quadrature Amplitude Modulation (QAM)).
- Different MCSs produce different packet lengths carrying the same data.
- Same RATs may have different L1 versions, and therefore each version may have its specification (802.11n vs. 802.11ac).
- The L1 layer may encrypt the data before transmitting.

This heterogeneity makes the TC at the spectrum level an arduous task. In fact, it is almost impossible to use ML algorithms that rely on any feature engineering. Here is where DL plays a fundamental role: this ML technique allows the automatic feature extraction on hyper-dimensional data [132]. These capabilities have also been used in the networking domain to perform classification tasks on high dimensional data like spectrum samples for modulation classification [83], and technology recognition [130, 69], or raw bytes and images representing packets or packet flows at L1 or above for TC [104, 133, 117].

As we have shown in Section 5.2, L1 packets (or packet flows) can be treated as 2D time series or images representing time series. Concerning time-series data, there are DL architectures based on RNN, such as LSTMs [120, 121] and Gated Recurrent Units (GRU) [134, 135], that are designed to exploit the structure of sequential data. However, they are difficult to train and have low accuracy with large data sequences [122, 123, 124].

One way to address the limitations of RNN architectures on large data sequences is to use CNNs for performing the automatic feature extraction while shortening the input sequence length before it is fed into the RNN architecture [112]. However, empirical evaluations have shown that even the recurrent layers are no longer needed for capturing the time-series patterns with CNNs [124]. Assuming a generic approach for the selected DL architecture for TC at spectrum level, Figure 5.3 shows a spectrum-based TC procedure built on top of several of the functionalities presented in the general-purpose framework for TC described in Section 5.3. The procedure comprises four main steps: data collection, L1 packet filtering/assembling, zero-padding/truncation, and classification.

- *Data collection:* In this step, the algorithm continuously captures spectrum samples and pre-processes them before being fed to the DL model. The first process to run in this step is the spectrum sensing to capture the samples in a given format, such as IQ or FFT samples. These samples are then normalized and grouped, for example, by using a fixed-sized moving window. The fixed-size samples are then labeled according to the RAT used to transmit them, their absence, i.e., noise, and a mix of them, i.e., interference. This step can be implemented by the spectrum sensing and TR sub-modules of the proposed framework as in [48].
- *L1 packet assembly/filtering:* One crucial aspect of the proposed algorithm is that it assumes that an L1 packet is a self-contained time-series structure where it can find and learn the upper layer protocol's pattern. A combination of IQ samples and labels from the first step provides a mechanism for assembling the L1 packets. More precisely, the IQ samples labeled with a given technology (step 1) are used as a filter to generate different IQ sample flows per technology. Then, the IQ samples labeled as noise are used as a delimiter to assemble them into L1 packets. Cross-correlation with sync words per technology or end-to-end ML approaches for packet detection, such as [136, 137, 138], can be added to increase this step's robustness.
- *Time series Padding/Truncation:* Once an L1 packet has been assembled, or a group/batch of them, we perform the zero-padding for short sequences, and truncation for long sequences, to normalize the length of all L1 packets to a given fix value. This step is important as the training and inference speed of DL algorithms can be improved by using sequences of the same length [139] at the cost of increasing the memory footprint. The optimal L1 packet length for padding/truncation is determined while designing and training the TC 's DL models. This value will depend on the DL architecture, the technology that generates the L1 packet, and the layer on which the features have to be extracted to perform the TC.
- *Fine- or coarse-grained classification:* In this step, the trained DL model consumes the padded/truncated L1 packets processed in the previous step and classify them. This step can be performed by a unique model or a cascade of simpler models for multi-step classification. Let us use TC on WLAN as an example. An L1 packet can be first classified by a model discriminating between the management, control, and data frames at L2 (coarse-grained classification). Then, the L1 packets classified as data frames can be passed through a second model that classifies them between two types of L7 traffic, e.g., music vs. video (coarse-grained classification). Finally, the samples labeled as music can be further classified according to the mobile application that generates it, e.g., Spotify vs. GPodcast (fine-grained classification).

The automatic execution of these four steps provides a complete spectrum-based TC system that is also technology agnostic. Of course, several decision choices will depend on the RATs sharing the spectrum and their hardware capabilities to run this algorithm, the channel bandwidth and sampling rate of the sensing module, the traffic classes to be discriminated, and at which layer their features can be extracted, and the architecture that is used to build the model(s). However, the proposed framework in Section 5.3 and the procedure proposed in this section provide enough flexibility to cover a large number of 5G and beyond use case scenarios using complementary RATs like WLANs and 4G/5G.

5.5 Data set generation and Deep Learning model design

To validate the feasibility of a spectrum-based TC system, we design, implement, and benchmark a DL model based on CNNs. In addition, we also design and implement a baseline model that uses a RNN architecture, similar to the one designed in [119], which is optimized for the provided data set. The rest of this section will describe how the classification tasks are defined, how we create the data set to train the models in the specified tasks, and the DL architecture design. This section focuses on the classifier as this is the core of the procedure proposed in Section 5.4.

It is important to notice that the first two stages of the proposed procedure (data collection and packet assembly) are implemented in an offline fashion for model training and validation via the data set creation (see Section 5.5.1). This decision was made since 1) some of our previous works have already shown prototypes that demonstrate its realization [48, 126], 2) the offline data generation provides a more flexible approach to evaluate the feasibility of performing TC directly on L1 packets, and 3) the proposed classifiers (see Section 5.5.3) have as input a single L1 packet, so time-related features that can be extracted by processing spectral data streams are not required while the data storage requirements are minimized in comparison to [117]. The impact of the third stage (time-series padding/truncation) will be evaluated in Section 5.6.

5.5.1 L1 packets Data set generation

Generating a data set for spectrum-based classification is a difficult task. However, we follow an approach that uses real L2 packets to generate L1 packets in the form of IQ samples, an approach similar to the one proposed in [117]. Without loss of generality, we selected the 802.11 wireless technology for generating the L1 packets. However, the approach described below can also be used to generate L1 packets from other technologies such as Long Term Evolution (LTE) with the same emulation platform[3].

Our decision to use a mixed approach (real packet traces + emulation platform to generate the spectrum samples) is motivated by the recent efforts of standardization bodies like the ITU [140], where multi-level ML pipelines are expected to be connected to emulation/simulation sandboxes to generate data for training and performing preliminary model testing [141]. This approach is important to support use cases like the one presented in this chapter, as generating real spectrum data for training the ML models would require the setup and deployment of infrastructure that is hard to obtain in real

[3]https://www.mathworks.com/products/lte.html

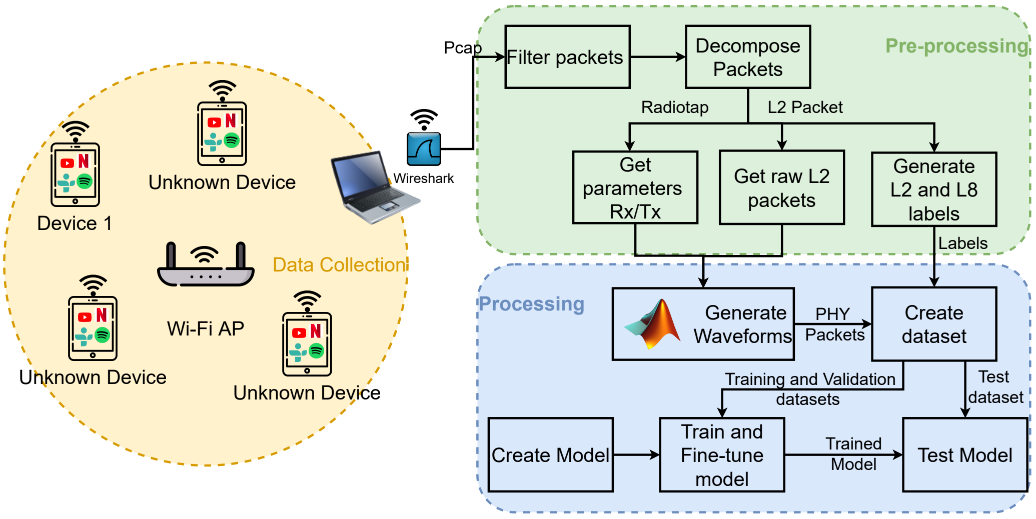


Figure 5.4: Hardware deployment and data flow from capturing traffic and data set creation to model training and validation.

life (e.g., isolated environments, management and control on radio transmitter/receiver at different layers, a mechanism to change the channel conditions, etc.). These sandboxes will provide the required flexibility to generate synthetic data to train and validate ML models while a complete realization of an integrated wireless and ML architecture with closed loops between the ML deployment platform, the sandbox, and the real network will minimize the inaccuracies of the models inside the sandbox and increase the degree of similarity between the sandbox and the real network.

As shown in Figure 5.4, we first perform a data collection step. In this step, we deployed an AP with wired connectivity to the Internet. It was placed in a closed space (living room of a home) where it shares the same channel with other APs deployed in neighboring houses. Our AP was configured to use 802.11n standard, with legacy compatibility, on channel 1 (2.4GHz) with 20 Mhz available bandwidth. Connected to this AP, a mobile device was used to run several L7 applications to generate traffic. Other wireless devices were also connected to the same AP but were not managed and might generate traffic.

This setup provides an easy-to-deploy mechanism to obtain real traffic that is both affected by traffic generated by other wireless devices on the same channel and a large number of variations of the 802.11n protocol stack such as MCS adaptation, L1 diversity (b, g, and/or n due to legacy compatibility of the AP), and L2 packet diversity. Then, a sniffer node (laptop) was used to capture packets over the air without being associated with any WLAN. The captured packets were encrypted as our test network, and networks around it were secured (mainly using Wi-Fi Protected Access (WPA)-2). The collected data were stored in pcap files[4]. Each of these files is named such that we can, later on, retrieve the name of the program/application generating the traffic. The packets in these files create an intermediate data set.

The resulting pcaps were then passed to the pre-processing step. In this step, the L2

[4]https://gitlab.com/wireshark/wireshark/-/wikis/FileFormatReference

packets were filtered to remove non-802.11 packets or packets that could not be accessed by the library used to read the pcap files. On the filtered packets, each packet is decomposed into the Radiotap header[5] and the L2 frame. The Radiotap header, which the host machine adds, obtains the physical layer parameters radios use to transmit/receive the packet. We extracted some information from the L2 packet to generate the labels associated with the L2 type of packet (Management, Control, and Data flags). Then, it was converted to raw bytes. All the captured packets are labeled according to the application/protocol that generated them. As our objective is to show the potential of using a TC system that works on L1 packets, we create labels at L2 and L7. We describe the classification tasks that were defined based on the generated labels in the following subsection.

The L1 data set was then created by combining the raw L2 packets with the information of the Physical Layer (PHY) associated with the L2 packet. For this purpose, we use the Matlab WLAN (2020b) toolbox[6] to generate standard-compliant waveforms of the L1 packets. To simulate the effects of an environment like a room in a house or a small office over the transmitted signal, the generated waveform was passed through an 802.11n (TGn) multipath fading channel with a delay profile model-B [142] with Gaussian noise.

The resulting L1 packets have a measured Signal-to-Noise Ratio (SNR) between 20 and 30dB. The modifications applied to noise-free raw IQ samples, such as adding fading channel effects and Gaussian noise, can be seen as data augmentation techniques. With this approach, it is also possible to generate additional L1 packets with other 802.11 PHY but carrying the same L2 Data Frame and use additional channel conditions. This removes the limitations of creating a data set with such properties on real environments as it will require a highly isolated environment with programmable radios to set the desired parameters at different radio stack layers and with controllable devices that generate different environment states where the label of the generated packet is known.

To the best of our knowledge, this is the first public data set that contains 802.11 standard-compliant L1 waveforms for testing traffic classification at the spectrum level. The waveforms are generated by different 802.11 technologies (b, g, n), which result in different transmission schemes such as DSSS in 802.11b and OFDM in 802.11g/n, different types of L2 frames (management, control and data), and multiple MCS (modulations such as BPSK and Complementary Code Keying (CCK) for 802.11b and BPSK, Quadrature Phase Shift Keying (QPSK), 16-QAM, and 64-QAM for 802.11g/n with coding rates of 1/2, 3/4, and 5/6 according to the standard and modulation selected). Moreover, the payload carried by these L1 packets (information at L2 and above) were generated using real traces of the L7 application running on a mobile device and connected to a secured 802.11 AP with WPA-2. As a result, the provided data set is more realistic and complex than the one used in [119], which is limited to High-level Data Link Control (HDLC), a simpler L2 protocol whose unencrypted waveforms are modulated only with QPSK at a unique data rate of 1Mbps. Finally, it is worth mentioning that the resulting data set contains a single L1 packet per sample, which is equivalent to the expected output of steps 1 and 2 of the proposed framework, where each packet is a sequence of IQ samples. This approach reduces the storage requirements for the data set as any IQ sample that is not part of a L1 packet, e.g., noise, is discarded.

[5]https://www.radiotap.org/
[6]https://www.mathworks.com/products/wlan.html

5.5.2 Traffic Classification tasks

One of the properties to benchmark our approach is the capability to use L1 packets to classify traffic at different layers and granularity, even if the packets are encrypted. For this purpose, the three classification tasks defining the selected labels are described below. Table 5.2 summarizes the proposed traffic classification tasks based on L1 packets.

5.5.2.1 Task 1 - L2 frame characterization

In this coarse-grained task, the TC algorithm uses L1 packets to determine if the transmitted packet is a Management, Control, or Data L2 frame in 802.11.

5.5.2.2 Task 2 - Application characterization

In this coarse-grained task, the TC algorithm uses L1 packets to determine the type of application inside the transmitted packet (e.g., audio or video). As only L2 Data frames carry L7 application data, the algorithm should also discriminate against packets that do not carry data.

5.5.2.3 Task 3 - Application identification

In this fine-grained task, the TC algorithm discriminates between the actual applications generating the L7 traffic.

Table 5.3 shows the number of samples and their distribution in terms of frame type and the physical layer technology used to transmit the packets in the generated data set for task 1. The total number of samples is 466K, where 16% are Management, 54% are Control, and 30% are Data frames. One interesting characteristic of this data set is that each frame type was mainly generated with a different 802.11 physical layer. For example, most of the Management frames were transmitted with 802.11b, which is expected as the APs in 2.4GHz work in compatibility mode and use the oldest technology (802.11b) and lowest MCS to transmit their Beacon frames aiming to increase its visibility and resilience.

Table 5.4 shows that the length distribution is highly associated with the type of frame in terms of packet length (byte and number of IQ samples generated). While most of the Management frames have a mean of 25K IQ samples, Data frames have a mean of 4.6K, and Control only 0.6K. Compared to the packet length at L2, which is the typical representation used by byte-based TC systems, they differ in several orders of magnitude. In this data set, the largest L2 packet did not exceed a length of 1.5Kbytes, which is only 1.1% of the largest L1 packet length found in this data set (131K IQ pairs).

Table 5.2: Description of the proposed classification tasks to evaluate the spectrum-based traffic recognition approach.

Task ID	**Traffic Classification Task**	**Traffic Classification type**	**Input representation**	**Layer on which the task has meaning**	**Number of Classes**	**Classes**
1	**L2 packet type**	Coarse-grained	IQ samples (Layer 1)	L2	3	Management, Control, Data
2	**L7 Application type**	Coarse-grained	IQ samples (Layer 1)	L7	3	Audio, Video, No application type
3	**L7 Application**	Fine-grained	IQ samples (Layer 1)	L7	7	Netflix, Youtube, Twitch, Spotify, Gpodcast, TuneIn, No application

Table 5.3: Sample distribution per task label and per technology within the task labels (task 1)

Total Samples	**Samples per Task Label**	**Technology**		
		802.11b	**802.11g**	**802.11n**
466348	Mgmt: 75156	74662	494	0
	Ctrl: 250967	5340	245627	0
	Data: 72264	5030	337	134858

Table 5.4: L1 and L2 packet length stats per label (task 1).

	L1 (Values in IQ pairs)			**L2 (Values in bytes)**		
	Mgmt	**Ctrl**	**Data**	**Mgmt**	**Ctrl**	**Data**
Mean	25139.37	676.54	4681.98	263.09	21.81	1066.93
Std	3683.24	434.35	3511.11	37.36	7.36	653.55
Min	1600	560	640	30	14	28
Max	37048	4928	131824	397	32	1546

The data set for tasks 2 and 3 is composed of 140K samples, where 67,8% of the packets are L2 Data type, while the rest are Management (10.5%) and Control (21.7%) (see Table 5.5). As a result of the 802.11n encryption, no payload of the Data frames can extract information from higher layers, so traditional approaches like port mapping and DPI will not work. Although tasks 2 and 3 are focused on TC at L7, the proposed TC system uses L1 packets as input. Therefore, the class formed by Management and Control packets provides a way to filter L1 packets that do not carry L7 information. Analyzing Table 5.5, we can see that 27.8% of L1 packets were labeled as Audio while 40% were labeled as Video during the data set generation in terms of coarse-grained labels. Similarly, in terms of fine-grained labels, 10% of the L1 packets in this data set are generated by the Spotify application, 7.2% by TuneIn, 10.6% by Gpodcast, 11.9% by YouTube, 13% by Netflix, and 15.1% by Twitch.

Focusing on task 2, Table 5.6 shows that, on average, L1 packets carrying Audio data are smaller than those carrying Video data. However, this is not the case in L2 representation, where both kinds of packets share similar statistical properties. Similarly, Table 5.7 shows that L1 representation has more variation on the packet length distribution than L2 ones. The variations on L1 packet lengths are mainly due to the changes on MCS. For example, 75% of the Data frames in the TuneIn application were using MCS 7, while this number dropped to 20% in Twitch L1 packets. In fact, 70% of the Twitch L1 packets are using MCS values between 3 and 7. This provides an interesting set of dynamic parameters that make this representation of the data challenging to extract features.

Once the data set is created, the last steps of the processing are executed: model creation, training with validation, and testing, which will be described in the following sub-section.

Table 5.5: Sample distribution per task label and per technology within the task labels (tasks 2 and 3).

Total Samples	**Samples per Class Task 3**	**Samples per Class Task 2**	**Frames**		
			Mgmt	**Ctrl**	**Data**
140665	Spotify: 13822	Audio: 39053	0	0	39053
	Tunein: 10229				
	Gpodcast: 15002				
	Youtube: 16671	Video: 56253	0	0	56253
	Netflix: 18268				
	Twitch: 21314				
	No-App: 45359	No-App-Type: 45359	14805	30554	0

Table 5.6: L1 and L2 packet length stats per label (task 2).

	L1 (Values in IQ pairs)			**L2 (Values in bytes)**		
	Audio	**Video**	**No-App-Type**	**Audio**	**Video**	**No-App-Type**
Mean	5.77K	10.7K	9.5K	1.2K	1.24K	101.92
Std	5.03K	14.2K	11K	565.77	553.12	123.29
Min	640	640	560	28	28	14
Max	38.9K	138.2K	43.4K	1.5K	1546	579

Table 5.7: L1 and L2 packet length stats per class (task 3).

	L1 (Values in IQ pairs)							**L2 (Values in bytes)**						
	Spotify	**Tunein**	**Gpodcast**	**Youtube**	**Netflix**	**Twitch**	**No-App**	**Spotify**	**Tunein**	**Gpodcast**	**Youtube**	**Netflix**	**Twitch**	**No-App**
Mean	5.4K	2.9K	8K	7.3K	10.8K	13.2K	9.5K	1.4K	709.14	1.5K	1.1K	1.3K	1.3K	101.92
Std	2.0K	2K	6.9K	7.1K	12K	18.8K	11K	460.12	675.67	263.5	544.84	545.87	550.61	123.29
Min	640	640	960	640	640	640	560	28	28	78	28	28	28	14
Max	38.9K	38.9K	38.9K	65K	138.2K	138.2K	43K	1.5K	1.5K	1.5K	1.5K	1.5K	1.5K	579

5.5.3 Deep Learning models design and training

As presented in Section 5.2, the literature is quite limited on TC using raw IQ samples [117, 119]. Thus, to realize the proposed framework to perform TC at any radio stack, we designed and implemented a DL model based on CNNs to overcome the limitations of previous works that either use a RNN architecture [119], or require specific procedures to separate different traffic from different users at spectrum level [117]. As proposed in Section 5.3 and realized by the approach presented in Section 5.4, the object classification for TC models are the IQ values associated with single L1 packet. To the authors' best knowledge, this is the first time that a CNN is used to solve TC at the spectrum level. As a baseline, we implemented, fine-tuned, and optimized the RNN architecture proposed in [119] to the data set used in this work. Figure 5.5 shows the two Neural Network (NN) architectures designed for validating our approach. One advantage of the DL models is their ability to perform automatic feature extraction on raw data to discriminate among multiple classes. In fact, it is that property that allows our approach to performing TC at any layer of the radio stack, e.g., L2 packet type or L7 application type, while using the same input representation. Otherwise, expert knowledge is required to determine the raw signal features allowing traffic class discrimination.

The design of the CNN architecture was based on some of our previous experiences solving classification tasks using raw spectrum data such as in [30, 130, 117, 15, 127, 53]. More precisely, we started with a 2D-CNN architecture that worked well in the task of TR with raw IQ samples (see Figure 3.4 in Chapter 3), and then we performed a fine-tuning step where the number of Convolutional (Conv) and Dense layers, the number of filters, the filter kernel size of the Conv layers, the maximum (max) pooling windows size, the dropout rate, and the learning rate were varied. To find the optimal number of Conv and Dense layers, we varied their number between 2 and 4. The lower limit is based on the fact that we need at least two layers to learn non-linear functions, and the upper limit is set to 4, as more than that decreases the model's performance in all the experiments. For the number of filters, we tried values between 8 and 128 (in steps of 8). Values above that limit increase the complexity of the CNN to the point that makes it impractical. To reduce the search space of the optimal configuration, we started by setting the same number of filters on all the layers. We tried to vary this number among the layers when we found a value that yielded a good performance. However, keeping the same number of filters on all the layers provided the best results in our case.

Concerning the filter kernel size, we first tried with values in the set {2, 3, 5, 7}. These values worked well for TR [130] and flow-level traffic recognition using images as input [117]. However, as shown in Section 5.6, this range did not work well for classifying L7 applications (tasks 2 and 3). Therefore, we increased the range of the filter kernel size and explored values between 8 and 64 (in steps of 8). The results indicated that for solving task 1, the CNN architecture only required a small kernel filter size as it was similar to a TR task. This can be explained as the frame type (task 1) is related to the 802.11 standard used to transmit them, as indicated in Table 5.3. In contrast, tasks 2 and 3 require a larger kernel size to learn helpful information at L7 directly from the spectrum. It is important to recall that increasing the kernel size helps to augment the reception field, which is important in classification tasks with large input sequences [143].

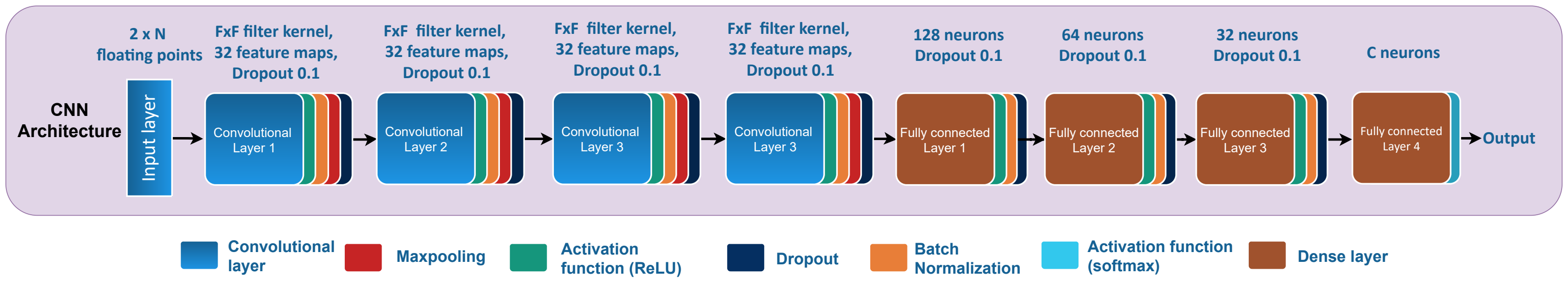
The number of dense layers was also selected in the range between 2 and 4, with a decreasing number of neurons from the inner layer towards the output layer with a maximum value of 512 and a minimum value of 16 in the layer before the output. As can be noticed, we follow the traditional approach of narrowing the network to force it to remove useless information while keeping only the relevant information to reduce computational costs. The resulting CNN architecture comprises four Conv layers, followed by four dense connected layers. All the layers have ReLU activation functions, except the last one with a soft-max function for classification, and are followed by a Dropout layer to improve generalization (reduce overfitting) and a Batch Normalization layer to accelerate training. Conv layers are also followed by Max Pooling layers to down-sample the input.

Figure 5.5a shows the resulting CNN-based architecture proposed in this chapter. It is important to note that during the design and fine-tuning phase, we also tried other architectures that have been used to solve classification tasks with time-series data, such as CNN+RNN [124] and WaveNet [143], which increases the learning capabilities on long sequences by increasing the reception field without increasing the filter kernel size as it is required in tradition Conv layers. However, they did not provide better performance than the developed model for this chapter.

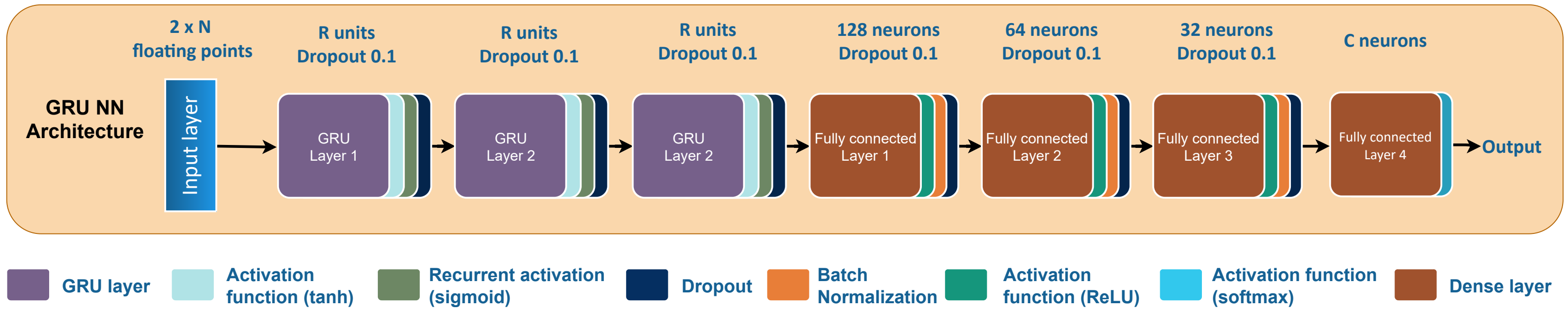
The RNN architecture, designed as the baseline, is inspired by [119] with fine-tuning and optimization steps based on the data set provided in this chapter. Following a similar approach as with the CNN, we varied the number of recurrent layers and the type and number of recurrent units to find the model that performs the best. The number of recurrent layers varied from 1 to 4, achieving the best performance with three recurrent layers, a result that is aligned with [119]. We also varied the type of recurrent units between GRU and LSTM, and we found that GRU had similar or outperformed the LSTM in both execution time and accuracy in all the experiments we run. This result also aligns with previous findings in other comparative studies, such as in [135]. Regarding the number of recurrent units, we searched for the best value between 64 and 384 in steps of 64 units. The upper limit was set at 384 since larger values generate execution times that are prohibited for real-time classification. Again, the best performance was achieved with values in the set {128, 256}.

The dense layers have the same configuration as the CNN architecture. This decision was made to provide similar learning capacities in the classification layers between the two models and allow a more fair comparison. We also tried other configurations for the dense layers in the RNN, but the one used in the CNN always provided the same or better accuracy. As indicated above, the three Recurrent layers have GRU units with recurrent activation function sigmoid and activation tanh. This combination of functions allows a fast implementation to improve performance[7]. Dropout layers also follow the Recurrent layers to improve generalization. Figure 5.5b shows the resulting RNN model, which we call GRU-NN, with the same hyper-parameters for the 3 tasks. Similar to the CNN models, the only hyper-parameter that changes among tasks is the number of GRU units (R), where it increases from 128 in task 1 to 256 in tasks 2 and 3.

[7]`https://keras.io/api/layers/recurrent_layers/gru/`



(a) The resulting 2D-CNN-based architecture proposed in this chapter for TC.



(b) The resulting GRU NN-based architecture used as baseline for TC.

Figure 5.5: DL architectures designed, implemented and evaluated in Section 5.6 for TC at spectrum level.

During the fine-tuning of both models, the hyper-parameters were selected using a multi-round approach of hyper-parameter search over hundreds of executions. The first round used a reduced version of the task 3 data set. Then, the resulting CNN and GRU-NN architectures were used as a baseline for another round of hyper-parameter search but using a reduced version of the data set of tasks 1 and 2. Similarly, it was determined that the best results were obtained using Adam optimizer [85] with a learning rate of 0.001 and a batch size of 64. We use categorical cross-entropy as a loss function and early stopping as a second method for regularization. A CNN and a RNN were created for each task. Given the size of the generated data set and the number of samples per label ($\geq 10K$), we used hold-out cross-validation (i.e., validation with an independent test set) to partition the data set. While the size of the data set allows for maintaining a large number of samples in the training set, we can still guarantee that knowledge about the test set is not leaked into the model, so we can also ensure generalization performance. As a result, the data set was partitioned such that the models were trained with 70% of the samples, while 15% was used for validation and 15% for testing. The models were implemented in Python, Tensorflow 2.1[8] was used to create, train, and evaluate the resulting models, and the hyper-parameter search was performed using hyperas[9]. The training was accelerated using Tesla V100 GPUs in our GPULab facility[10].

5.6 Results and discussion

This section presents the performance evaluations of the two models proposed in Section 5.5. For the L2 Frame type TC (task 1), we balanced the first data set by performing under-sampling. As a result, the evaluation data set contains 75k samples per class, driven by the class with fewer samples (see Table 5.4). Performing the same operation on the second data set to generate the samples used for the L7 Application type TC (task 2) and L7 application TC (task 3), the resulting datasets contains 39K samples per class in task 2 and 10.2K samples per class in task 3 (see Tables 5.4 and 5.4, respectively).

We ran five evaluations on each task, where both models were trained, validated, and tested over different lengths of the input L1 packet in the number of IQ sample pairs (100, 300, 500, 800, 1K, 3K, 5K). Notice that when the length of the L1 packet was shorter than the required input length, we applied a zero-padding operation at the end to adjust it (post-padding). Otherwise, we truncate it to the required length (post-truncation). For each task, we select the results from the best evaluations per model and input size. In the following subsections, we will analyze the models' performance in terms of quality of prediction (accuracy and macro-averaged[11] precision, recall, and F1-score) over the test data set, training time per epoch, and prediction time per sample. These three metrics are good indicators of how good the model is when classifying unseen data, how costly/hard it is to fine-tune and train the model, and the expected execution time when predictions have to be done over a group of samples.

[8]`https://www.tensorflow.org/`

[9]`https://github.com/maxpumperla/hyperas`

[10]https://doc.ilabt.imec.be/ilabt/gpulab/index.html

[11]The metrics are calculated for each label and find their unweighted mean. We used the unweighted mean as all the datasets are already balanced.

5.6.1 L2 Frame characterization task (Task 1)

In this coarse-grained TC task, the models must identify if a given L1 packet carries an L2 frame of type Management, Control, or Data. This was the most straightforward task among the three proposed in this chapter. One indicator is the small filter kernel size, set to 2, or a reduced number of GRU units, only 128, required to achieve high accuracy. Figure 5.6 (bottom) shows that even with short sequences, e.g., 500 IQ samples per packet, the models already achieve around 98% accuracy. When the sequence had a length of 3K and 5K IQ samples, the CNN model can outperform the GRU-NN model and achieved an accuracy above 99.86%, as shown in Table 5.8. Finally, we notice that with an input length of 3K and 5K, the CNN model achieved the same value in the prediction quality metrics (with up to 2 decimal points), on average. This result can be expected when a large and well-balanced data set is used to test a classifier that can correctly discriminate each class with almost zero miss-classifications.

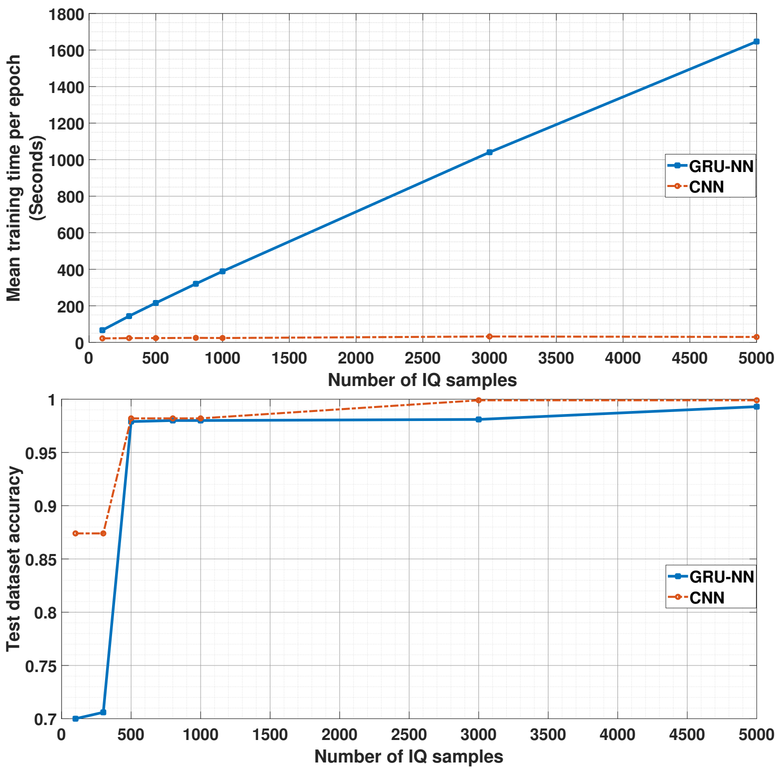


Figure 5.6: Training time per epoch (top) and accuracy on the test data set (bottom) vs. the input size N in task 1.

Figure 5.7 shows the resulting confusion matrices of different input lengths in the CNN. Interestingly, even with an input length of 100 IQ samples, the models can accurately discriminate the L1 packets carrying Management frames from the other types of frames. At the same time, it still has trouble classifying the Control and Data frames. However, this can be explained as Management frames were mainly transmitted using 802.11b (see Table 5.3, in which waveform is generated using the DSSS modulation technique. In contrast, the Control and Data frames are modulated using OFDM.

Table 5.8: Summary of the metrics used to evaluate the quality of the model's predictions with the largest input lengths.

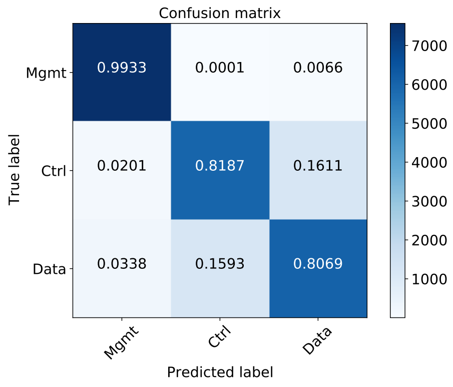
Task	**Input length (IQ samples)**	**Model**	**Quality of prediction metrics (%)**			
			Accuracy	**Precision**	**Recall**	**F1-score**
1	3000	***CNN***	***99.86***	***99.86***	***99.86***	***99.86***
		GRU-NN	98.07	98.18	98.07	98.09
	5000	***CNN***	***99.86***	***99.86***	***99.86***	***99.86***
		GRU-NN	99.28	99.30	99.29	99.29
2	3000	***CNN***	***97.78***	***97.84***	***97.82***	***97.82***
		GRU-NN	76.03	75.96	76.40	75.68
	5000	CNN	97.63	97.67	97.67	97.67
		GRU-NN	78.10	78.33	78.45	78.21
3	3000	***CNN***	***90.44***	***91.16***	***90.45***	***90.60***
		GRU-NN	52.36	53.09	52.58	50.44
	5000	CNN	89.28	90.30	89.25	89.44
		GRU-NN	54.48	56.05	54.77	52.82

Table 5.9: Prediction time per single L1 packet in task 1.

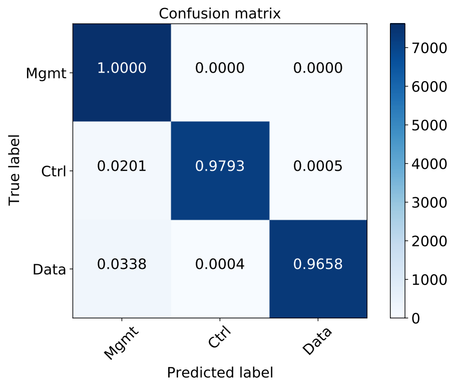
Input length (IQ samples)	**Model**	**Average prediction time (ms)**
3000	CNN	0.092
	GRU-NN	4.71

Moving to the performance in terms of the training time, we can see in Figure 5.6 (top) that the GRU-NN has a training time per epoch that increases linearly to the input length (from 66s with input length 100 to 1646s with input length 5K). On the contrary, the CNN training time per epoch increases sub-linear (from 21s with length 100 to 30s with length 5K). This result implies that GRU-NNs are hard to fine-tune, making them unfeasible solutions when the problem requires long sequences to improve accuracy. Although this was an expected result since RNN-based architectures process their input sequentially while CNN can exploit parallel processing, it is interesting to experimentally validate that CNNs outperform RNNs in sequence-to-label classification tasks using radio spectrum data as it has been found in other research fields like speech recognition and voice generation [143].

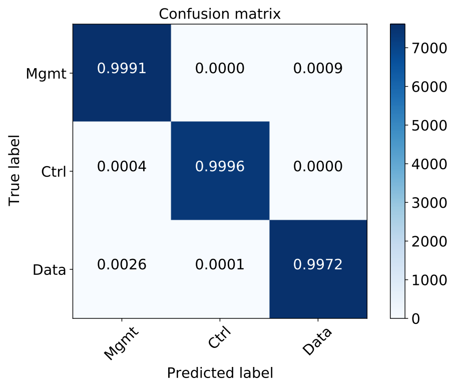
The final metric to evaluate is how long (on average) the model takes to predict over a single L1 packet. Table 5.9 shows that the CNN can classify an L1 packet containing 3K IQ samples in 92µs, 51 times faster than the GRU-NN. The pcap that generates this data set includes 466K packets captured in 1193s. It gives us 390 packets per second on average. Using the CNN model, those 390 packets would be classified in less than 35ms. This prediction time is auspicious for its deployment in real-time traffic analyzers. Some recent works have shown that similar pre-processing steps on 0.5s of spectrum data, i.e., getting spectrum data, framing/packetizing it, and formatting it before sending it to the classifier, can be executed in less than 200ms [48].



(a) N = 100



(b) N = 500



(c) N = 3000

Figure 5.7: Test data set normalized confusion matrices with different input sizes N (number of IQ samples) in task 1.

5.6.2 L7 Application characterization task (Task 2)

This coarse-grained TC task challenges the trained models to classify L1 packets according to the type of application being carried at L7. The three discriminating classes are Video, Audio, and packets that do not carry L7 application data (No App-Type). When we defined this task, we hypothesized it should be more challenging than task 1. Our reasoning comes from two facts: first, all L1 packets carrying Data frames were transmitted using 802.11n, similar to the task 1 data set, and 2) the high standard deviation on both Audio and Video L1 packet lengths (see Table 5.6) would not allow using this feature for discriminating between them. One way to validate our hypothesis is by analyzing the model size, the achieved accuracy, and the impact of the input size on it.

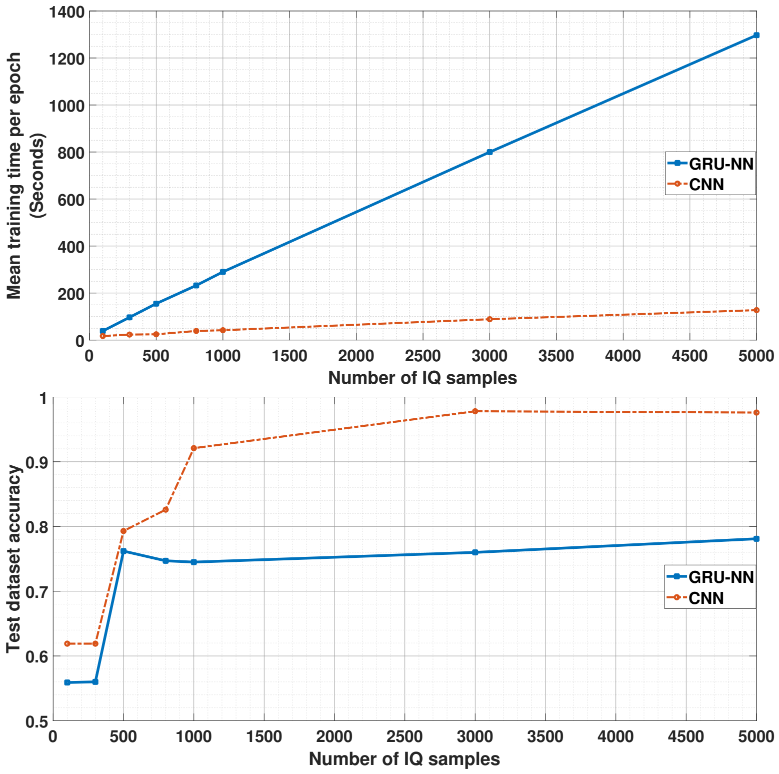


Figure 5.8: Training time per epoch (top) and accuracy on the test data set (bottom) vs. the input size N in task 2.

Let us start by analyzing the model size (see Table 5.11). While task 1 only requires a kernel size of 2, tasks 2 and 3 require a kernel size of 32. The increasing filter kernel size is translated into a larger model, from 106K to 3.7M trainable parameters, and a larger reception field is used to extract the raw signal features. Here, the reception field can be understood as the number of consecutive IQ samples representing the learned features. Similarly, the GRU-NN also increases its learning capabilities by setting the number of GRU to 256, so double the number compared to task 1, and moving from 276K to 1.03M trainable parameters.

In terms of prediction quality, the CNN model outperforms the GRU-NN model by far in

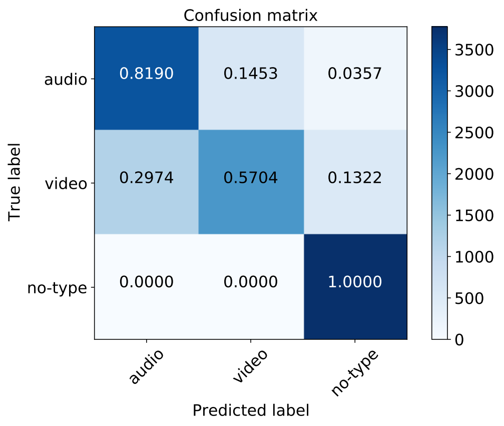
Table 5.10: Prediction time per single L1 packet in task 2.

Input length (IQ samples)	Model	Average prediction time (ms)
3000	CNN	0.15
	GRU-NN	5.13

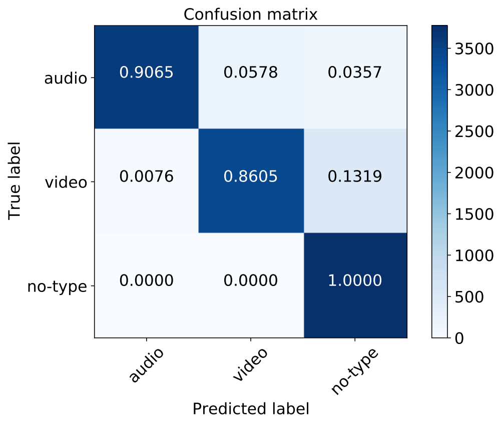
all the metrics, but, in both cases, the achieved values are lower than the ones achieved in task 1, even with larger input size. More in detail, Figure 5.8 (bottom) shows that while the CNN model was able to achieve 97.8% accuracy on the test data set with an input length of 3K IQ samples, the GRU could only achieve 78.1% with an input length of 5K IQ samples. In fact, the GRU-NN dropped its accuracy with an input length larger than 500 IQ samples, which was recovered with an input length of 3K, and it was improved at 5K by only 2%. This behavior has also been found in previous works in this area (see Figure 7 in [119]). It may be possible that the GRU-NN can improve their accuracy with larger sequences, but as we will analyze below, its training cost makes it unfeasible (around 21min per training epoch). Notice that the CNN model achieved 97.6% with an input length of 5K IQ samples, indicating that increasing the input length does not help discriminate among the classes, at least from 3K to 5K. Of course, as the GRU-NN showed, longer sequences may improve it. Similar to task 1, the CNN model achieved similar prediction quality metrics, on average.

Focused on the CNN, which provides the best performance, Figure 5.9 shows that with an input length of 500 IQ samples, the discrimination between L1 packets with Management/Control frames (no application type) is perfect against Data frames (Audio+Video), which is expected based on the results on task 1. However, the CNN model has difficulties separating L1 packets carrying audio and video. With an input length of 1K IQ samples, the CNN model achieved 92.1% accuracy (above 88% if we count only audio and video). With an input length of 3K, this model achieved an accuracy of 97.8% (above 96.7% considering only audio and video classes). This is quite impressive because the input data are L1 packets carrying L2-L7 payloads encrypted using the WPA-2 secure method, transmitted with different MCS values, and include simulated channel effects and noise.

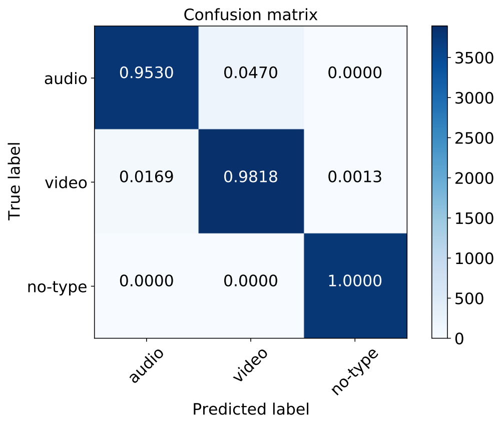
Similar to task 1, Figure 5.8 (top) shows the linear dependency of the GRU-NN training time per epoch concerning the input length, while this relation is sub-linear with the CNN (from 17s with 100 IQ samples to 127s with 5K). However, the slope in the CNN line is higher compared to the results in task 1. This is mainly due to the model's larger size. This result implies that the CNN model is faster to train and fine-tune than the GRU-NN, similar to the results in task 1, but it also shows that the CNN model performs much better than the RNN architecture when using longer sequences of data. The better performance of the CNN is also reflected in the prediction time. Table 5.10 shows that the CNN takes 150µs to classify an L1 packet containing 3K IQ, 34 times faster than the GRU-NN. If we analyze the pcap of the Twitch video app, which is the application with the largest mean packet length (see Table 5.7), it contains 21.7K packets, which were captured in 159s. It gives us 136 packets per second on average. Using the CNN model, those 136 packets would be classified in less than 20ms. It was, again, encouraging results for TC on L1 packets in real-time.



(a) N = 500



(b) N = 1K



(c) N = 3K

Figure 5.9: Test data set normalized confusion matrices with different input sizes N (number of IQ samples) in task 2.

Table 5.11: Summary of the parameters of the CNN and GRU-NN architectures used to perform the different classification tasks using IQ samples as input data.

Classification Task	**Task-dependent parameters**				**Shared training parameters**					
	CNN		**GRU NN**		**optimizer**	**learning rate**	**Maximum number of epochs**	**Batch size**	**Loss function**	**Early Stopping monitored function**
	Model parameters (N=5000)	**Kernel Size (F)**	**Model parameters (N=5000)**	**GRU units (R)**						
Frame characterization	106K	2	276K	128	adam	0.001	200	64	Categorical cross-entropy	Validation loss
Application characterization	3.72M	32	1.03M	256						
Application identification	3.72M	32	1.03M	256						

5.6.3 L7 Application identification Traffic Classification task (Task 3)

Task 2 is a coarse-grain version of this task, where the models must classify each L1 packet according to the application generating it. In addition to the class that identifies the packets that do not carry L7 application data (No App), there are six other classes: 3 classes of applications that generate video-type traffic (Youtube, Netflix, Twitch) and 3 classes of applications that generate audio-type traffic (Spotify, TuneIn, GPodcast). Section 5.5.2.2 and 5.5.2.3 provide more detailed information about the data set used for this task.

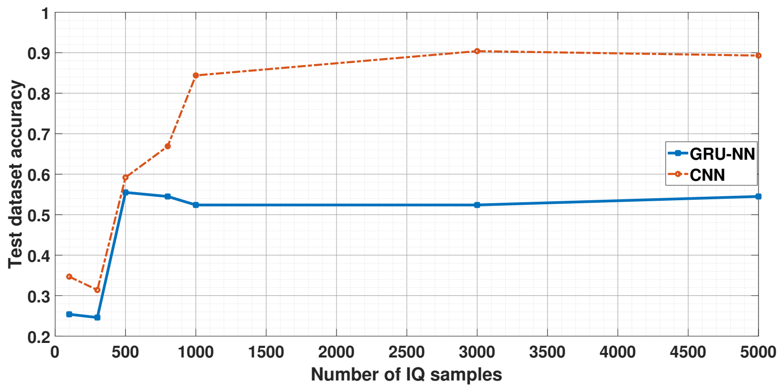
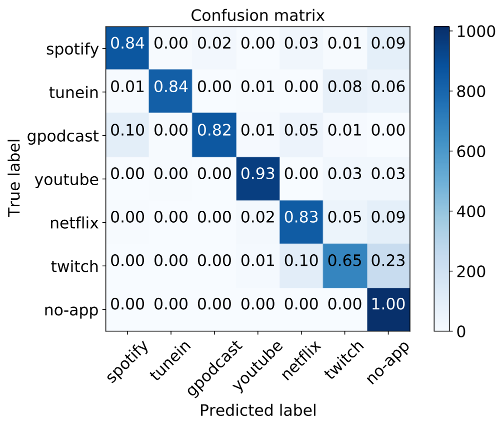


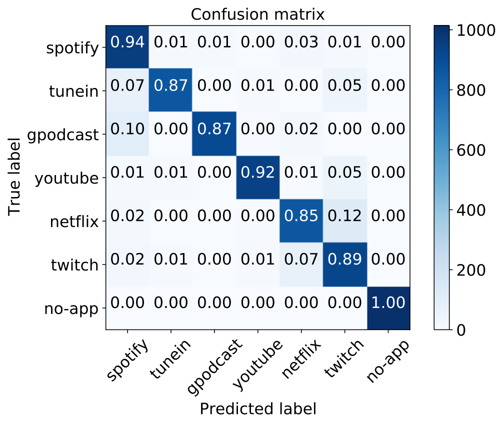
Figure 5.10: Accuracy on the test data set vs. the input size N (number of IQ samples) in task 3.

This task is helpful to determine how much more challenging or easier the TC is if the traffic classes are fine-grained. We hypothesized that this task should be more problematic than its coarse-grained version as the fine-grained applications may have similar statistical properties, and the L7 protocols may also be similar. The coarse-grained classification exploits this but can cause trouble for the fine-grained one. This idea comes from analyzing Table 5.7, where we can see that while Netflix and Twitch L1 packets have similar statistical length values very close, they are far from the length statistics of audio applications. On the other hand, audio applications have statistical length values similar to each other, which may increase the difficulty of discriminating among them. Notice that this intuition contradicts the results found in [104], where the traffic characterization task results are better than the application classification.

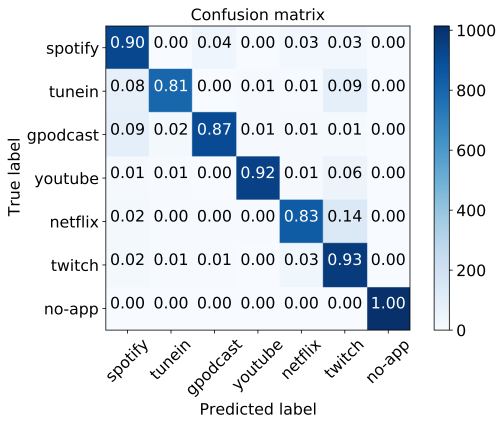
Figure 5.10 shows that this task is more challenging than the previous two. Although the results are very similar to the ones in Figure 5.8 (right), there is a drop in accuracy of 7.4% in the CNN (90.4% vs. 97.8% with 3K input length) and 23% in the GRU-NN (54.4% vs. 78.1% with 5K input length) compared to task 2. Focusing on the CNN model results, Figure 5.11 shows that the sources of misclassification are located in Netflix vs. Twitch (adding around 19% misclassifications) and Spotify vs. TuneIn $\cap$ Gpodcast (adding about 17% misclassifications). Finally, Table 5.8 shows that the precision of the CNN model is higher than its recall for this task, on average. In other words, the CNN model tends to have fewer false positives (mostly concentrated in Spotify, Netflix, and Twitch) than false negatives (mostly concentrated in TuneIn, Gpodcast, and Netflix). We omit the training and prediction time results as they are congruent to the results of the previous task since the same data set was used but with fine-grained labels.



(a) N = 1K



(b) N = 3K



(c) N = 5K

Figure 5.11: Test data set normalized confusion matrices with different input sizes N (number of IQ samples) in task 3.

5.6.4 Comparison against Deep Learning and statistical Machine Learning on bytes

To compare the quality of the spectrum-based TC algorithm trained and evaluated in the previous sub-sections, Table 5.12 compares its accuracy against two state-of-the-art DL models that receive a byte representation of the packet at L2, and two ML classifiers, trained with the Gradient Boost (GB) ensemble method. We use the packet's input representation length as a unique input feature for the statistical ML-based models. Note that we can only use this feature for all three tasks as the 802.11n L2 packet's payload was encrypted.

The two DL models for the byte/protocol representation use the same architectures and hyper-parameter configuration as the spectrum-based models presented in Section 5.4 and use a fixed-size input of 1546 bytes, the largest L2 packet captured in the datasets (see Table 5.6), with post-zero-padding/truncation. The GB models were also fine-tuned via hyper-parameter search, and the best results were achieved using a learning rate of 0.1, maximum depth of the individual regression estimators set to 5, 500 number of boosting stages, and 95% of the samples are used for fitting the individual base learners for both models.

As expected, the DL models working on the byte representation of the packet at L2 provide the highest accuracy on all three evaluated tasks. The well-defined framing can explain this at L2 and the shorter length of the input data compared to the dynamic nature of the L1 representation of the input data, where the L1 header has a well-defined structure. However, the number of IQ samples representing the payload depends on the channel conditions and the MCS values used while transmitting. This is also reflected in the high and similar accuracy obtained by both the CNN and GRU architectures used in the byte-based approach.

To the author's best knowledge, this is the first work that also performs TC on raw bytes from encrypted 802.11n L2 packets. Although no pre-processing was done on the raw packets except padding/truncation, the obtained results are aligned with other approaches using DL on raw bytes with more pre-processing steps given the protocol structure's visibility on higher layers [104]. However, the spectrum-based approach still provides a very competitive accuracy with a similar performance on task 1, a drop of 1.38% in task 2, and 4.37% in task 3, which is outstanding given the added complexity of using spectral data as input, i.e., multi-dimensional, longer sequences, in comparison to an input using raw bytes. In the case of specialized hardware that can demodulate and decode the L1 packets of a given technology, it is clear that the packet-based approach, even on encrypted packets, will be the natural choice to guarantee performance in both accuracy and execution time. However, realizing the proposed framework removes the need for specialized hardware as L1 packets of any technology can be captured and classified without the need to demodulate and decode them with a limited negative impact on performance. Of course, the proposed framework opens new challenges in reducing complexity and improving the classifier's performance and the whole chain.

Table 5.12: Comparison of the 3 different approaches for TC on the three evaluated tasks: DL models using raw spectrum and byte representation and Gradient Boosting using the input length as feature.

Packet representation	**Machine Learning Method**	**Algorithm**	**Features**	**Input Length (Number of Features)**	**Accuracy**		
					L2 Frame characterization	**L7 App characterization**	**L7 App Identification**
Spectrum	NN	CNN	Raw IQ samples with zero-padding	3000	**99.86%**	**97.78%**	**90.44%**
		GRU		5000	99.28%	78.10%	54.48%
	Ensemble	Gradient Boosting	Number of IQ samples (L1 packet length)	1	99.06%	80.83%	59.66%
Bytes	NN	CNN	Raw Bytes with zero-padding	1546	**99.99%**	**99.16%**	**94.81%**
		GRU			99.99%	99.12%	94.46%
	Ensemble	Gradient Boosting	Number of bytes (L2 packet length)	1	99.86%	76.35%	54.84%

Finally, the GB models show the worst performance, but their results indicate that the input data's length was enough to solve task 1. This result was expected given the high correlation between L2 frame type and the packet length in the data set (see Table 5.3). However, this feature is insufficient to differentiate the L7 application in tasks 2 and 3. On the contrary, the DL models can automatically extract hidden features from the raw data, probably higher-level protocol structures relevant to solving the other two TC tasks with minimal pre-processing.

5.7 Conclusion

This chapter introduces a general framework to achieve TC at any layer in the radio stack. With this framework, a ImRAT GW would be able to perform TC on any packet that is being transmitted over the air in the surroundings of the GW, independent of the technology and the wireless domain which packets belong. We also proposed a novel procedure to perform TC on raw spectrum data on top of this framework. This procedure first combines spectrum sensing with a state-of-the-art approach for recognizing radio technologies to build a PHY representation of the packets. Then, a DL architecture is used to perform TC on the L1 packet. As a result, a unique representation of a single L1 packet is needed to perform the TC at any layer as it already carries all the information describing the different protocols at L2 and above without the added complexity needed to perform demodulation, decoding, and decryption.

We focused on building and evaluating two DL architectures to classify the L1 packets to perform the proposed procedure. For this, we created and performed a statistical analysis on two different datasets that were used to solve three TC tasks: one coarse-grained at L2 (task 1: frame identification in 802.11b/g/n), one coarse-grained at L7 (task2: Audio/Video/No-App type), and one fine-grained at L7 (task3: 3 Audio apps, 3 Video apps, No-App). The datasets were generated by combining packet traces from real transmissions with a standard-compliant waveform generator for 802.11 radio technologies.

Performance evaluations showed that the DL model based on CNN could achieve the best performance on the three proposed tasks, achieving above 99.9% in task accuracy discriminating among classes in task 1, 97.8% in task 2, and 92% in task 3. These results are very promising if we compare them to byte-based DL models, where spectrum-based achieved similar accuracy on task 1, a drop of 1.38% in task 2, and 4.37% in task 3. Finally, the proposed DL architecture could predict a given class in the order of microseconds, compelling prediction times for integrating into spectrum-based real-time traffic analyzers.

Chapter 6

Parallel Reinforcement Learning with Minimal Communication Overhead

The content of this chapter has been partially published in:

- M. Camelo, M. Claeys, and S. Latré, "Parallel Reinforcement Learning With Minimal Communication Overhead for IoT Environments," in IEEE Internet of Things Journal, vol. 7, no. 2, pp. 1387-1400, Feb. 2020, doi: 10.1109/JIOT.2019.2955035.
- M. Camelo, M. Claeys, and S. Latré, European patent application for "EXPLORING AN UNEXPLORED DOMAIN BY PARALLEL REINFORCEMENT" filed at the European Patent Office (EPO) on October 12, 2018, and received the application number EP 18200069.5.

In previous chapters, we introduced AI-based solutions to enhance the performance of Intelligent Radios (IRs) by making them spectrum-aware. However, simply improving IRs' spectrum-sharing capabilities may not suffice to handle the increasing data generated by the next generation of agent-based smart applications, which continuously interact with other applications and their environment. In this chapter[1], we propose algorithms that enable agent-based smart applications using Parallel Reinforcement Learning (PRL) to share knowledge efficiently among learning agents over a network, minimizing communication overhead and alleviating resource demands, such as spectrum, in wireless networks. Consequently, PRL-empowered applications can seamlessly operate across the network as scalable distributed intelligent programs.

6.1 Introduction

Addressing spectrum scarcity extends beyond the development of Intelligent Radios (IRs) and encompasses a broader ecosystem, particularly the proliferation of intelligent

[1]Chronologically, this chapter was my first contribution to this thesis but is presented as the last chapter for consistency in the narrative. This work was an initial part of our research in developing efficient and distributed ML algorithms. While exploring various use cases, including IoT, we eventually decided to apply these techniques to wireless networks, as they held significant potential, as was later demonstrated in the preceding chapters, both during and after the DARPA SC2 project.

applications interconnected within the Internet of Things (IoT). IoT is giving rise to a new wave of distributed applications where computational decisions are made across diverse endpoints, from IoT devices to gateways and cloud servers [144]. Many of these IoT applications operate within the domain of Reinforcement Learning (RL) [34], a method that enables autonomous learning by interacting with the environment. Examples of these applications are autonomous robot navigation in industrial sites [145, 146], Smart Traffic Signal Control [147] in cities, and the control and management of the network infrastructures connecting IoT devices [148].

The rapid proliferation of these IoT applications and their continuous interaction with their environments places considerable strain on the resources shared by the IoT devices such as the spectrum[149, 150, 151]. While the algorithms and frameworks discussed in the preceding chapters aim to enhance spectrum efficiency through the use of IRs, they may not be sufficient to comprehensively address the challenge of these applications generating data while learning and acting. To address this effectively, it is imperative to augment these IoT-based smart applications with communication-awareness as an integral component of their RL-driven learning algorithms. By doing so, these applications can strategically reduce their communication overhead by optimizing their learning objectives while minimizing bandwidth and network resource consumption. Communication-awareness will make their applications more spectrum-efficient since they can minimize redundant or non-critical communication related to their learning. However, achieving communication-aware smart applications poses a challenge due to the nature of how RL algorithms learn.

In RL, the task is modeled as a Markov Decision Process (MDP). An MDP models the problem as a set of states and actions, called the state-action space, transitions between states, and rewards [50]. A RL agent aims to improve its policy, i.e., its memory, until it learns the best action to take in each environment state. However, RL algorithms suffer from two main problems from a learning perspective: 1) they require a long learning time, in terms of learning steps, to converge to an optimal policy, and 2) the curse of dimensionality of large state-action spaces [50], which is a result of interacting with complex environments with an exponential combination of state-action pairs. While the first problem is mainly addressed by Multi-Agent Reinforcement Learning (MARL) [37], the second one is tackled by either 1) using an efficient representation of the space-action space for table-based RL algorithms [152], or 2) using function approximation algorithms [153].

Parallel Reinforcement Learning (PRL), a type of MARL, is a framework that allows reducing the learning time by leveraging parallelization. MARL achieves this by sharing knowledge among multiple RL agents that run in parallel and use a unique shared policy without coordination [38]. In general, PRL can reduce the learning time at a rate proportional to the number of agents [39]. However, the execution time, i.e., actual time, is reduced at a slower rate due to the communication overhead between agents and the shared Q-Table (QT), the wasted learning of using overlapping search strategies, and processing and storage constraints of the infrastructure [154].

State-of-the-art PRL algorithms address the poor decrease in the execution time by either 1) assuming agents with unlimited capacity for processing, storage, and data link (infinite bandwidth and no delay), or 2) partitioning the shared QT using domain knowledge [155, 156]. Domain knowledge allows agents to learn about specific parts of the problem and share/merge only information about the overlapping regions among partitions, minimizing communication costs. However, this approach is not practical since assuming domain knowledge is hard to meet in real, usually unknown environments, and it breaks one of the main advantages of model-free RL algorithms.

In this chapter, we present a novel general-purpose partitioning algorithm that enhances PRL algorithms to support the execution of RL-based applications in distributed environments with improved execution time due to minimal communication overhead while sharing information. In other words, it provides communication-awareness to the agents by design. Performance evaluations of a PRL system solving the robot navigation problem in an intelligent factory, i.e., an IoT-like environment where communication is expected to be wireless, show that the proposed algorithm incurs almost no communication cost in a converged state and is scalable in the number of RL agents solving the problem. An essential property of the proposed algorithms is that it is application-agnostic since its design is linked to the training aspect of RL agents and not to the application itself so that it can be applied to other distributed applications/problems beyond smart IoT applications such as MARL version of spectrum sharing algorithms [157, 158, 159, 160, 161], which can also target optimization of communication resources.

The rest of this chapter is structured as follows. Recent works on RL-based distributed applications and PRL are discussed in Section 6.2. The proposed algorithm is described in Section 6.3, and the results of the performance evaluations are provided in Section 6.4. Conclusions are presented in Section 6.5.

6.2 Related work

This section reviews recent works that use RL algorithms to solve problems related to different IoT domains and applications. Then, we discuss some state-of-the-art PRL algorithms and analyze the main issues of extending RL-based IoT applications with PRL.

6.2.1 Reinforcement Learning-based IoT applications

Mohammadi et al. implement a Deep Reinforcement Learning (DRL) agent to solve the indoor localization problem using Bluetooth devices' signal strength [162]. The agent was trained following a Semi-Supervised Deep Reinforcement Learning (SSDRL) approach. As a result, the proposed algorithm can improve up to 23% the estimation of the distance to the target locations compared to a Supervised Deep Reinforcement Learning (SDRL) model, which does not take into account experiences with no reward. Min et al. propose a RL-based algorithm that protects the location and usage pattern of healthcare IoT devices when their information is offloaded to the edge [163]. The RL agent learns to choose the offloading rate and the local computing rate with an increase of 36.6% of the privacy level, and a decrease of 9.63% and 68.79% of the energy used by the healthcare IoT devices and the computation latency at the edge, respectively.

Salahuddin et al. propose a heuristic to select and reconfigure services on the Internet of Vehicles (IoV) [164]. By modeling the service configurations and the cost of migrating

Virtual Machines (VMs) between the set of available data centers as an MDP, they were able to select a setting that reduces the VM migration overhead over the long term. Wang et al. develop a RL-based algorithm for the IoV that selects a routing protocol that maximizes the packet delivery ratio and minimizes the end-to-end delay [165]. As a result, the proposed algorithm can dynamically change the routing protocol and maximize network reliability and communication quality. El-Tantawy et al. propose a MARL framework to learn and control the signal switching sequence at a junction to minimize the intersection delay [166]. The results show a reduction in the average intersection delay of up to 39%. Mannion et al. extend this work by using PRL, where multiple agents share their experiences while learning in parallel on separate instances of a problem [155].

RL algorithms are also being used to optimize aspects such as power consumption and channel interference in IoT wireless access technologies. Khan et al. propose an algorithm that uses MARL to adapt the power of transmission to minimize the interference between IoT devices [167]. Simulation results show that the average throughput of the system was increased. Zhu et al. use a RL approach to design a new scheduling mechanism for Cognitive IoT [168]. Simulation experiments show that the algorithm achieves an appropriate policy that maximizes the system throughput while reducing power consumption and packet loss. In the same line, an algorithm that solves the adaptive sampling interval problem for Wireless Sensor Network (WSN) via RL is proposed in [169]. This algorithm avoided nearly 73% of transmissions while keeping a similar quality of the recollected data compared to when the smallest sampling interval was adopted. Debizet et al. propose a Q-Learning (QL)-based Adaptive Power Management (APM) hardware module to optimize the energy consumption of IoT System-on-Chip (SoC) [170].

6.2.2 Parallel Reinforcement Learning

While Single-Agent Reinforcement Learning (SARL) [56] and general MARL [37] have been an object of extensive research, PRL has been less studied. The effectiveness of using a cooperative mechanism among RL agents has been proved in [39]. Using observations of other agents as an auxiliary source of experience, called the Learning By Watching (LBW) strategy, the learning time decreases at a rate of $\Omega(1/n)$, where n is the number of agents.

Kretchmar evaluates two algorithms where multiple agents do not interact with each other directly but by sharing Q-values [38]. The first algorithm, the Constant-Share Reinforcement Learning (CS-RL), assumes a shared QT that an agent can access and update at any moment. The second algorithm, the Max-Shared RL (MS-RL), assumes that the agents communicate much less often. The results show that CS-RL and MS-RL, each with ten agents and considering no communication overhead, are 7.6x and 1.2x faster than a single agent, respectively. The sub-linear speed-up in CS-RL was due to increased wasted learning in small problems with overlapping search strategies.

A parallel version of Q-learning via a communication scheme with local cache (PQL-C) was presented in [171]. Based on a master-slave model, slave agents are assigned to different parts of the QT and learn over a subgroup of states. When an agent requires a Q-value from a remote state, it contacts the master agent that keeps a shared copy of the QT. The communication among agents is reduced by keeping a copy of the most recent values in a local cache at the slave nodes. This value is only updated when the number of requests is more significant than a given threshold.

Mannion et al. propose the State Action Space Partition (SASP) algorithm that extends PQL-C by adding a static partition strategy for the QT based on domain knowledge [155]. The Prioritized Field Learning Method (PFLM) improves the performance of the PQL-C in the early stages by assigning priorities, which change dynamically, to partitions that are being updated the most [156]. Quan Liu et al. propose the divide-and-conquer strategy based on a scalable parallel reinforcement learning (DCS-SPRL) algorithm [172]. This algorithm decomposes the original problem into smaller ones by considering the capacity of available resources. Then, a scheduling algorithm assigns problems to agents. The scheduling algorithm dynamically selects which subproblems must be solved and which are put idle.

Finally, a Co-allocation of Storage and Processing (PCSP) PRL algorithm is proposed in [154]. This algorithm performs dynamic allocation of agents to partitions. Instead of requesting data from remote partitions, this algorithm deploys temporal agents on those partitions until they either reach a goal state or visit a state in their origin partition. The main idea behind this approach is to minimize the communication cost by performing only local operations on remote partitions. Results show that the number of episodes decreases proportional to $\Omega(1/n)$, while it is up to 24 times faster than CS-RL with centralized QT in terms of execution time.

6.2.3 Speeding-up RL-based IoT applications with PRL

IoT applications are becoming more intelligent, and RL algorithms enhance them by adding learning capabilities and making decisions autonomously by interacting with the environment. To deploy table-based RL algorithms and avoid the curse of dimensionality of large state-action spaces, most works use an efficient representation of the states and actions that reduce the requirements for storage in constrained IoT devices. Regarding where the decisions are taken, some works assume a centralized computational platform, where observations of multiple agents are recollected via sensory data from the environment, and actions are taken in a centralized way. Other applications are fully distributed, and decisions are taken at the nodes. In both cases, RL-based algorithms found optimal solutions without a priori information about the environment. However, the cost of collaborating and sharing information among agents, in terms of execution time, is not evaluated or ignored merely by assuming resources with unlimited capacity in storage, processing, and data link (i.e., unlimited bandwidth and no delay).

Similarly, the design and deployment of PRL algorithms have been focused on reducing the required training period. However, the communication costs associated with PRL in distributed environments have yet to be thoroughly researched. In fact, most algorithms have assumed either 1) domain knowledge to partition the problem into loosely coupled subproblems, reducing communication among agents by only sharing knowledge between subproblems with overlapping states, or 2) specialized parallel hardware, which is very expensive, or distributed environments with unlimited storage, processing, and bandwidth capabilities. As a result, most algorithms are not scalable in terms of the number of deployed agents and perform poorly in distributed environments, limiting their deployment on IoT infrastructures with constrained devices.

It is important to recognize that although there has been a paradigm shift from multi-agent table-based RL towards DRL-based agents in recent years [173, 174], which use Neural Network (NN)s as function approximators of the QT, the contributions presented in this chapter remain state of the art for table-based PRL multi-agent systems. Moreover, the ideas presented here can be extended to MARL with DRL to improve their scalability, which is a current research topic in these areas [175, 176].

6.3 Dynamic partitioning for Parallel Reinforcement Learning for IoT applications

New Smart-X IoT applications include domains such as robotics, telecommunications, vehicular traffic, resource management, and distributed control [144]. The large-scale, distributed, and resource-constrained nature of the IoT requires algorithms that solve the problems behind these applications under dynamic and uncertain environments, with large state-action spaces and collaborating with other entities in a short time. Here is where PRL [51, 37] is increasingly being used since these algorithms match perfectly to these new requirements. In this section, we will introduce the motivation of the problem and our dynamic partitioning algorithm that allows deploying PRL algorithms in constrained IoT environments. Table 6.1 summarizes the most relevant parameters used in the rest of the chapter.

6.3.1 The need for a dynamic state-action space partitioning

PRL-based IoT applications face two main challenges in constrained IoT environments: 1) how to optimize the resource usage in terms of spectrum, bandwidth, processing power, storage capacity, and energy consumption, and 2) how to increase the performance in terms of execution and learning time. Both challenges are highly linked to partitioning and assigning the problem to RL agents. In order to see this relationship, we will introduce our use case test scenario, which is based on a simplified version of the item-fetch use case for Autonomous Vehicles in a Smart-Factory [177] with wireless communication capabilities.

In this use case, multiple wireless autonomous mobile robots are distributed around a factory. The task of the robots is to transport the inventory from the *replenishment stations* (RS), where the shelving units are filled, to the *picking stations* (PS), where they are incrementally emptied. There is one robot per RS and one Access Point (AP) located at each RS, which has a logical location in the factory. The main objective of robots is to learn the path from their RS to the location of the PS, where they leave the transported inventory while reducing the energy consumption due to a) wireless transmissions between agents and b) distance traveled. Notice that the energy constraint reduces this problem to find the shortest path between the RSs and the PS. Figure 6.1 (a)[2] shows the simplified version of an item-fetch problem in a grid of 3x5 with two RS, one PS, and one shortest path for each agent, called g_0 and agent g_1. Notice that, without loss of generality, the association of an AP to the RS will help us to highlight the link between learning algorithms that are

[2]This figure contains some assets from Freepik.com

Table 6.1: Parameters/variables used in the algorithms.

Parameter/Variable	**Description**
S	Set of states
A	Set of actions
N	Set of agents
E	Set of episodes
E_{max}	Maximum number of episodes
S	Set of partitions
α	Learning rate
γ	Discount factor
ϵ	Epsilon for ϵ-greedy policy
η	Wildcard threshold
κ	Local policy Affinity
τ	Local policy tolerance
β	Episodes before calling the partitioner
Δ	Communication cost
ρ	Number of remote updates
μ	Number of iterations
λ	Number of exchanged states
RS	Replenishment station
PS	Picking station

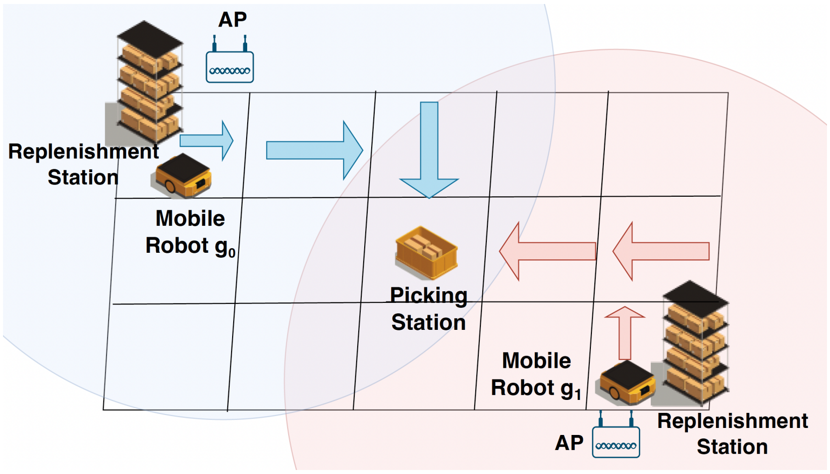
communication-aware and the objectives of the task that the algorithm is trying to solve, as we are going to see below.

As we want to solve this problem using a RL approach, we model it as an MDP as follows. Each environment state is related to a unique location identified by a single point in a 2D-Euclidean space, i.e., a state is a pair $(x, y) \in \mathbb{Z}^2$. Every agent has four possible actions (UP, DOWN, LEFT, RIGHT). If the robot arrives at a location on the grid sides, any action that moves it out of the grid will result in no movement. This decision allows the agents to avoid collisions when they reach a limiting wall or stop exploring locations out of the defined environment in real environments. The reward for visiting any state is -1, except for reaching the goal state, where the reward is +100. This reward configuration motivates the selection of the shortest path toward the goal. Figure 6.1 (b) shows the assigned state to a given location in the grid. The location of the RSs is mapped to states s_0 and s_1, which are the initial states for the agents g_0 and g_1, respectively, and the location of the PS is mapped to state s_7, the goal state for both agents.

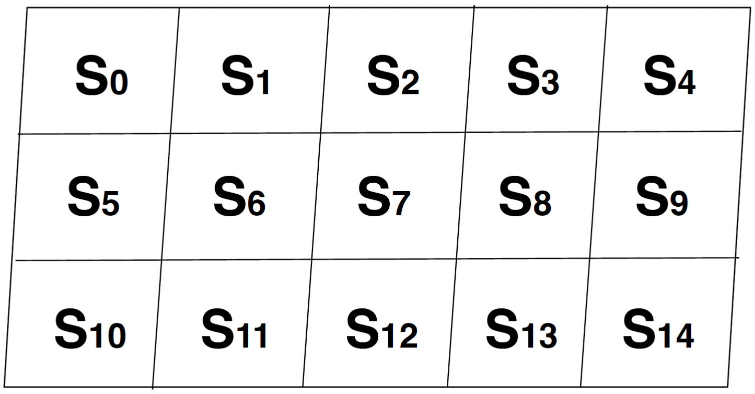
Once the MDP is defined, the agents must meet the following conditions to parallelize the problem (See Section 2.2.3):

Be Fully collaborative: This implies that agents receive the same reward after taking the same action in the same state, which is required to maximize the joint reward of the agents [178, 57]. This condition only depends on the reward function in simulated and real environments. Therefore, it must be guaranteed during the design of the MDP model.

Update a unique and shared policy: This allows minimizing the wasted learning of



(a)



(b)

Figure 6.1: (a) The item-fetch problem in a grid of 3x5, 2 RS, 2 AP, and 1 PS. (b) States associated with each location (b).

using overlapping search strategies [38, 39]. All agents must update the same QT in simulated and real environments.

Take actions that have local effects only: This assumption reduces the problem to an MDP, whose action space is the joint action space of the Stochastic Game (SG). Assuming independent agents is also a practical decision since the observability of joint actions is hard to meet in real environments [57]. The locality of the effects of taking actions can be ensured by either including a collision avoidance mechanism in real environments or interacting with an isolated copy of the environment in simulated ones.

Note that in small-sized problems, a SARL or a centralized PRL algorithm with a few agents may be fast enough to solve it, and then the resulting table can be fully deployed to each agent. However, if the problem becomes complex and the state-action space is very large, these approaches can not scale in space or time due to the cost of sharing information among agents. Moreover, although solving the learning problem will also result in a reduction of the energy consumption due to wireless transmissions since power received at a given distance is inversely proportional to the square of the distance from the transmitter, this reduction will be negligent compared to the communication overhead of the learning procedure as demonstrated in Section 6.4. Therefore, we need to tackle this problem from the perspective of the operation of the learning algorithm itself. For this, the state-action space and the policy, represented by the QT, must be divided and assigned to the distributed agents in such a way that a) agents try to keep only states associated with their (shortest) paths, and b) reduce communication among agents querying state-action values of states that are exploring. In this context, a QT partitioning is equivalent to problem decomposition. In this work, we assume that q-values associated with a given state are always in the same partition; therefore, the word state and q-values related to that state are interchangeable.

Most state-of-the-art PRL algorithms assume an a priori knowledge of the environment. It allows optimal state-action space partitioning by decomposing the main problem into small, loosely coupled ones and assigning them to the agents. Table 6.2 shows two possible ways of partitioning the QT of the example based on the q-values assigned to a given state. Note that s_7 was not included in the partitions as it can be assigned to any partition. In a distributed scenario, a random partition of the QT will entail communication between agents g_0 and g_1 each time g_0 needs a q-value associated with s_1, and it would be the case if g_0 follows its shortest path to the RS as in Figure 6.1 (right).

Table 6.2: Possible partitions of the QT based on states.

Type of partitioning	Agent	Assigned states with their q-values
Random	g_0	$s_0, s_2, s_4, s_6, s_8, s_{10}, s_{12}$
	g_1	$s_1, s_3, s_5, s_9, s_{11}, s_{13}, s_{13}$
Optimal	g_0	$s_0, s_1, s_2, s_3, s_4, s_5, s_6$
	g_1	$s_8, s_9, s_{10}, s_{11}, s_{12}, s_{13}, s_{14}$

A PRL algorithm can avoid communication among agents by using domain knowledge and assigning the q-values following the optimal partition as in Table 6.2. In this case, no communication is required between agents once the agents follow the optimal paths. Note that this partitioning allows assigning agents to partitions with the main objectives of 1) focusing on a specific part of the problem and 2) limiting communication among agents. However, this approach is not practical since assuming domain knowledge is hard to meet in real, usually unknown environments, and it breaks one of the main advantages of model-free RL algorithms. In general, the proposed algorithm, PARTITIONER, works as an add-on for any PRL algorithm that wants to reduce its execution time in distributed environments. How PARTITIONER creates the initial and subsequent partitions is described below.

6.3.2 Dynamic co-allocation of processing and storage

As IoT environments are large, dynamic, and mostly unknown, the assumption of partitioning and distributing the state-action space using an a priori domain knowledge is far from realistic. We aim to design an algorithm that dynamically creates loosely coupled partitions of the QT, which represents the problem space, and assign each partition to the agent exploiting it the most. If the partitioning algorithms can provide an optimal co-allocation of storage and processing, i.e., the learning agents update mostly the states in the partition that it is assigned, and the communication cost is minimized. Algorithm 2 shows a generic PRL algorithm that includes a state-action space partitioning procedure (PARTITIONER).

Algorithm 2 PRL algorithm with dynamic partitioning.

```
Require: Total states $|S|$, actions $|A|$, Max number episodes $E_{max}$
Require: Number of episodes before calling PARTITIONER $\beta$
Require: Set of agents $N$ with $|N| = n$ agents
 1: if $n > 0$ then
 2:     Call PARTITIONER and create the initial $n$ partitions
 3:     Assign agents to partitions
 4:     Distribute and initialize the QT
 5:     for $e \in E$ do
 6:         Execute RL episode $e \in E$ on $g, \forall g \in N$
 7:         if if episode number $\%\beta$==0 then
 8:             $\forall g \in N$ stop learning
 9:             Call PARTITIONER and create $n$ partitions
10:     Obtain statistics from $g, \forall g \in N$
11: return $QT$ of size $|S||A|$
```

6.3.2.1 Initial partitioning

Before a PRL algorithm is deployed on the IoT infrastructure, an initial partitioning of the state-action space is executed. In this initial step, the q-values, pairs of state-action values, are mapped to partitions (Algorithm 2 line 2), and then the agents are mapped to partitions (Algorithm 2 line 3). The PARTITIONER algorithm takes as inputs the number of states $|S|$ and the number of actions $|A|$, which are required to determine the QT size and to verify if there are enough available resources to deploy the distributed QT, and the number of agents $|N|$=n, to determine the number of partitions to create. Although we are assuming that only one agent is assigned to each partition, MARL architectures that support the separation of the learning from storage will allow taking into account the hardware capabilities of the available IoT devices [154].

We assume no previous domain knowledge, so we propose two different procedures to create the initial mapping from q-values to partitions. The first procedure creates the partitions via a random assignment of q-values to partitions. Once the partition is created, agents are randomly assigned to these partitions. The second procedure assigns q-values to partitions during the learning episodes (Algorithm 2 line 6) on the fly. Initially, there are no q-values assigned to partitions, but the (empty) partitions are already assigned to the agents. Then, suppose an agent visits a state that no agent has visited before. In that case, it creates the q-values entries for that state and all the available actions in its partition and initializes them, then updates the value related to the taken action. In both procedures, the maximum number of new entries per partition will be limited to the local storage size.

In the case of the PRL algorithm having access to domain knowledge, it can use such information to create the initial partitioning and dynamically update it under environmental changes that may increase the communication cost.

6.3.2.2 Dynamic Partitioning

After the PRL algorithm has assigned agents to partitions, the agents start learning until it reaches a maximum number of episodes E_{max}. While the algorithm runs, the episodes before the partition parameter, or β, defines when the agents must stop learning and call the PARTITIONER algorithm. Our PARTITIONER algorithm comprises two steps: a local optimization procedure that splits each partition into smaller ones, which uses statistics about the state updates, and a global and distributed optimization procedure that re-distributes states among agents, which is based on state-trading heuristics.

Local optimization (performed by each agent locally): As we want to keep states that a local agent in its partition mainly visits, we need to determine which states have been visited the most by each one of the n agents. To do it, during the learning phase, each agent keeps track of the number of updates that the agent $g_i \in N$ has performed on state s since the last call to PARTITIONER. Let $UPDATES(s, g_i)$ be the data structure inside each state s to store that information. Each partition is divided into $n + 1$ internal partitions when PARTITIONER is called. The states are re-assigned to the internal partitions as follows: a state $s \in S$ in the local partition $p_l \in P$ is assigned to the internal partition $p_{l,k}$, where $k \in [1, n + 1]$ and $l \in [1, n]$, if the agent g_k has updated s the most in the previous learning period. As a result, partition $p_{l,k}$ will contain the states that agent g_k has visited the most.

To avoid local optima in the global optimization step, the special partition $p_{l,n+1}$ will store the states that were visited below the given threshold parameter $\eta \in [0, 1]$. The procedure is as follows. Before the algorithm assigns states to the internal partitions, it normalizes the total number of updates per state between 0 and 1. Then, all states with normalized total updates value $\leq \eta$ are assigned to partition $p_{l,n+1}$. In other words, $p_{l,n+1}$ contains the states that were updated just a few times, compared to the total number of updates performed during the previous learning period in the partition, and thus are not strongly bound to a specific agent. These states are used as wildcards in exchanging other states that an agent has most remotely updated during the global optimization step. Algorithm 3 shows the pseudo-code of this optimization step.

Algorithm 3 LOCAL OPTIMIZER heuristic

```
Require: Access to local partition $p_l$
Require: Access to local UPDATES map
Require: Threshold parameter $\eta \in [0, 1]$
1: for each $g_l \in N$ do
2:     Create $n + 1$ internal partitions on $p_l$
3:     for $s \in S$ in local partition $p_l$ do
4:         $norm_{updates} \leftarrow$ normalize total updates on state $s$
5:         if $norm_{updates} > \eta$ then
6:             $k \leftarrow$ Determine id of agent that updated $s$ the most
7:             move $s$ to internal partition $p_{l,k}$
8:         else
9:             move $s$ to internal partition $p_{l,k+1}$
```

To illustrate how this algorithm works, we will use the example of Figure 6.1 and an initial random partitioning as described in Table 6.2. Let us assume that β episodes have run, and then each agent stops the RL algorithm and starts running the Algorithm 3. Each agent creates $n + 1 = 3$ new partitions to re-order the states in their partition $p_0 \in P$ (Algorithm 3 line 2). Note that after running several iterations, it is expected that the states $\{s_0, s_1, s_2, s_5, s_6\}$ have been mainly visited by agent g_0, and $\{s_8, s_9, s_{12}, s_{13}, s_{14}\}$ by agent g_1.

Back to agent g_0, after normalizing the number of updates received by its local states in its partition $p_0 \in P$ (Algorithm 3 line 4), we may expect that the states s_0, s_2, and s_6 were mainly visited by agent g_0 and s_4, s_8, and s_{12} by agent g_1, with normalization values $> \eta$. In this case, then states $\{s_0, s_2, s_6\}$, and their related q-values, are moved to the new internal partition $p_{0,0}$ and $\{s_4, s_8, s_{12}\}$ to $p_{0,1}$ (Algorithm 3 line 6-7). As state s_{10} is far from any shortest path of g_0 and g_1, then it is expected that its number of updates by any agent, after normalization, is $< \eta$, and therefore it can be moved to $p_{0,2}$ (Algorithm 3 line 9). A similar process is executed in g_1. Now, we are ready for the global optimization step.

Global optimization (performed among agents): Reducing agent communication can be achieved by putting states near the agent that visits it the most. In order to do it, a global trading heuristic to exchange states among partitions is performed after the LOCAL OPTIMIZER has created the $n + 1$ divisions inside each partition. Let g_1 and g_2 be two agents in the system. If g_1 has states in the internal partition $p_{1,2}$, then it offers them to g_2. Let $|p_{1,2}|$ be the number of states in partition $p_{1,2}$. When g_2 receives the offer from g_1, it checks how many states it contains in its internal partition $p_{2,1}$ and shares it with g_1. Without loss of generality, let us assume that $|p_{1,2}| \leq |p_{2,1}|$. Then g_1 sends its states in $p_{1,2}$ to g_2 together with at most $z = |p_{2,1}|$-$|p_{1,2}|$ states from $p_{1,n+1}$. g_2 sends $|p_{2,1}| - z$ states from its partition $p_{2,1}$ back to g_1. This procedure is repeated among all agents following a random ordering when selecting the agent to exchange states. In order to support stochastic environments, the map UPDATES is reset after this step. It allows faster convergence when the transition between states changes over time. Algorithm 4 shows the pseudo-code of the global optimization step.

Algorithm 4 GLOBAL OPTIMIZER heuristic

```
Require: Access to Distributed QT with $|S||A|$ values
1: for each pair $g_l, g_k \in N$ do
2:     for each internal partition $p_{l,k}$ do
3:         if $|p_{l,k}| \geq 0$ then
4:             $g_l$ offers $|p_{l,k}|$ states to $g_k$
5:             if $|p_{k,l}| \geq 0$ then
6:                 if $|p_{k,l}| \geq |p_{l,k}|$ then
7:                     $g_l$ sends min $(|p_{l,k}| + |p_{l,n+1}|, |p_{k,l}|)$ to $g_k$
8:                 else
9:                     $g_k$ sends min $(|p_{k,l}| + |p_{k,n+1}|, |p_{l,k}|)$ to $g_l$
```

Let us return to the example to illustrate how the Algorithm 4 works. Based on what g_0 and g_1 are expecting to learn, a possible output from Algorithm 3 can be the following: agent g_0 may have $\{s_0, s_2, s_6\} \in p_{0,0}$, $\{s_4, s_8, s_{12}\} \in p_{0,1}$, and $s_{10} \in p_{0,2}$; agent g_1 may have $\{s_1, s_5\} \in p_{1,0}$, $\{s_{14}, s_{13}, s_9\} \in p_{1,1}$, and $\{s_3, s_{11}\} \in p_{1,2}$. Let us assume that agent g_0 triggers the communication with agent g_1 (Algorithm 4 line 2-4). Agent g_0 offers the states in its internal partition $p_{0,1}$ to Agent g_1. As $|p_{0,1}| = 3 > 2 = |p_{1,0}|$, then g_1 prepares the set of states $p_{1,0} \cup s_3 \vee s_{11}$ to exchange them with g_0 (Algorithm 4 line 9). Note that we used one of the states in the wildcard partition $p_{1,2}$ to get the maximum number of states from g_0.

Finally, g_1 performs the same procedure to get the states it updated the most in p_0 before partitioning. After both agents have exchanged their states, the new partitioning will be the following: $\{s_0, s_1, s_2, s_5, s_6, s_{10}, s_{11}\} \in p_0$, and $\{s_3, s_4, s_8, s_9, s_{12}, s_{13}, s_{14}\} \in p_1$. Note that the main difference of this partitioning, which is optimal after convergence and following a greedy policy, with the optimal partition presented in Table 6.2 are the states that are not part of the shortest paths of the agents, i.e., $\{s_3, s_4, s_{10}, s_{11}\}$, and which final partition would mainly depend on the number of updates that they may receive during agents' exploration phase.

6.3.3 Local-affinity policy

In general, action selection policies for SARL algorithms do not exploit information related to the locality of a state to select actions. However, in distributed environments, this information is fundamental in improving the initial partitioning. For this reason, we introduce a second policy, called the local-affinity policy, which is executed once the agent selects an action following its behavior policy (e.g., $\epsilon-$greedy).

Note that during the learning period, each agent keeps track of the updates performed by remote agents on its local partition. As a way of exploring and exploiting states that are stored locally and may be good states for a given agent's actions, the local-affinity policy tries to select an action that, in the past, moved the agent to another local state and whose q-value is close to the q-value of the action chosen by the agent's policy. Identifying if an action a taken in state s moved the agent to a remote/local state s' in the past is performed as follows: each time we select an action, we tag it as local or remote according to the location of the next state s'.

Algorithm 5 shows a QL agent that includes the local-affinity policy. Given a state s, the agent selects an action a using the behavior policy. Then, this action is evaluated by the local-affinity policy. This policy selects an alternative action a', with a probability of $\kappa \in [0, 1]$, which moved the agent to a local state last time it was taken, and for which the difference between $Q(s, a')$ and $Q(s, a)$, which is normalized in the range $|a_{max} - a_{min}|$, where $a_{max} = \max_{a'} Q(s', a') - Q(s, a)$ and $a_{min} = \min_{a'} Q(s', a') - Q(s, a)$, is minimal and less than a maximum tolerance $\tau \in [0, 1]$. In other words, this policy acts as a ϵ-greedy policy, where $\epsilon = \kappa$ allows local states exploitation. Note that if no available actions move the agent to local states, it follows the behavior policy. Like the exploitation-exploration trade-off in ϵ-greedy policies, high values of κ may make the agent stuck in local states. In contrast, low values may reduce the possibility of exploring states in their local partition. Algorithm 6 shows the pseudo-code of the proposed policy.

Compared with a non-modified version of Q-learning, Algorithm 5 differs in the new lines 5, 7-11. Note that lines 7-11 do not modify the behavior of the agent and are only used to store information about the times a state is updated by a given agent (Algorithm 5 line 7) and tag states are local or not (Algorithm 5 line 8- 11). On the other hand, calling Algorithm 6 will force the agent to find local states via exploration and controlled by

Algorithm 5 Q-Learning with local-affinity

```
Require: Access to Distributed QT
Require: agent identifier $l$ ($g_l$)
 1: repeat(for each $e \in E$)
 2:     Initialize $s$
 3:     repeat(for each iteration)
 4:         Use behavior policy to choose $a$
 5:         $a \leftarrow LocalAffinity(a, \kappa, \tau)$
 6:         Take action $a$, observe $r, s'$
 7:         $UPDATES(s, g_l) = +1$
 8:         if $s'$ is local state then
 9:             $(s, a).moveToLocal = True$
10:         else
11:             $(s, a).moveToLocal = False$
12:         $s \leftarrow s'$
13:         $TDerror = \max_{a'} Q(s', a') - Q(s, a)$
14:         $Q(s, a) \leftarrow Q(s, a) + \alpha \cdot (r + \gamma \cdot TDerror)$
15:         $s \leftarrow s'$
16:     until s is goal state
17: until e == $E_{max}$ or convergence = true
```

Algorithm 6 Local-Affinity Policy

```
Require: action $a$, affinity $\kappa$ and tolerance $\tau$
1: if random.nextDouble() $\geq \kappa$ then
2:     for $a_2 \in A$ do
3:         if $(s, a_2)$.moveToLocal then
4:             if $\frac{|Q(s,a2)-Q(s,a)|}{|Q(s,a_{max})-Q(s,a_{min})|} \leq \tau$ then
5:                 new action $\leftarrow a_2$
6: return new action
```

the parameter κ. Retaking the example, if agent 0 would have s_1 in its local partition and s_5 would be in a remote one, and assuming the Q-values of their actions towards destination via the shortest path are similar at s_0, then Algorithm 6 will try to privilege the path via s_1 as it is a local state (Algorithm 6 line 4-5).

6.3.4 Algorithm complexity analysis

The proposed algorithm aims to reduce the communication overhead of PRL algorithms when they run in distributed environments. However, this is achieved by adding some extra cost, in time or space, to the learning process itself. Following, we present the complexity analysis of PARTITIONER among all the agents [3].

Time: Algorithm 3 runs in $O(|S|)$ and Algorithm 4 runs in $O(|N|^3)$. Therefore, PARTITIONER has a time complexity of $O(|S| + |N|^3)$.

Space: Algorithm 3 requires $O(|S||A|)$ to store the QT. Algorithm 5 requires $O(|S||A||N|)$

[3]We have excluded the Algorithm 6 from the complexity analysis since it does not change the complexity of the RL algorithm

to store the data structure *UPDATES* and the label variable *moveToLocal* per q-value. Therefore, PARTITIONER has a time complexity of $O(|S||A||N|)$. Clearly, the data structure *UPDATES* is wasting the space obtained by distributing the QT. However, this can be improved if PARTITIONER only stores the number of local or remote updates per state, regardless of the agent that performed the update. It reduces the space complexity of the data structure *UPDATES* to $O(|S||A|)$. Of course, the agents will run Algorithm 4 in a more stochastic fashion since the exchange of states is random and uses states that are not important to them. In the long term, the algorithm will converge if the states are exchanged enough times to guarantee that states can reach the partitions where an agent will exploit it the most.

Communication: Measured in terms of the number of messages exchanged between agents during learning and partitioning. By assuming one message can contain the q-values associated with a given state, Algorithm 4 will require $O(|S|)$ messages during partitioning, i.e., in the worst case, all the agents exchange all their states, and Algorithm 5 will exchange up to $O(\mu)$ messages while the agents are learning. Then PARTITIONER has a total communication complexity of $O(|S| + \mu)$ Note that although the number of messages exchanged between agents in Algorithm 5 does depend on n, its effects are canceled since we expect that μ is reduced proportionally to $O(1/n)$.

From a practical point of view, our solution can be deployed in two main setups: 1) Simulated agents deployed in the cloud/edge that are learning by interacting with an isolated simulated environment and control resource-constrained Robots in real environments, 2) Real agents interacting in a real environment. In the first case, offloading the learning task from the constrained IoT devices to cloud/edge will improve the execution time in distributed environments such as cloud/edge by minimizing the communication cost and removing the (highly probable) latency of taking actions and changing states in the real environment. Of course, the quality of the policy will depend on the accuracy of the environment model used by the simulated agents during training. In the second case, our solution will reduce the energy consumption of the IoT devices since the energy required to transmit messages over the network, e.g., in wireless devices, is minimized. Note that if the IoT-RL application is controlling moving robots, the assumption of using a collision avoidance mechanism is inevitable since RL algorithms involve a trial-and-error component [179, 180].

6.4 Performance Evaluations

In this section, we present the results of evaluating the performance of our algorithm in terms of communication cost and the impact of changing the number of agents, the partitioner execution frequency, and the problem size.

6.4.1 Scenario and PRL algorithm

To evaluate our algorithm, we simulate the item-fetch use case for Autonomous Vehicles in a Smart Factory scenario, which is described in Section 6.3.1 using the Brown-UMBC Reinforcement Learning and Planning (BURLAP) library 3.0 [181]. RL agents are modified version of QL BURLAP agents implementing Algorithm 5. We use the in-memory computing framework Apache Ignite 2.3 [182] to implement the multi-agent framework proposed in [154], where each RL agent is both a processing agent, i.e., it runs the RL algorithm, and storage agent, i.e., it stores part of the shared QT. The multi-agent system was deployed in a computer cluster composed of 8 PC Dell Optiplex 780 running Ubuntu 16.04, where each PC runs a single agent at most.

To run the algorithm in different environment setups, we model the evaluation scenario as a BURLAP environment, and each agent interacts with a copy of it in isolation. This removes the need to implement collision avoidance algorithms required when agents share the same environment. As a proof of concept, we also use the resulting framework to deploy two RL agents that control the same number of Sparki robots [183] in an emulated environment of size 7x7. Some media showing a demo of our proposal in the emulated scenario can be found in [184].

The proposed algorithm is implemented inside the PRL algorithm CS-RL[38]. We selected the CS-RL algorithm for two main reasons. Firstly, this algorithm is an implementation of the LBW collaboration strategy [39], and therefore, it decreases to the minimum required learning time at a rate of $\Omega(1/n)$. Secondly, if we assume distributed agents updating a shared QT, CS-RL has the highest communication overhead and will allow us to verify if the proposed algorithm will also work in this worst-case scenario. Finally, as our approach assumes a table representation of the Q-function, then the algorithm convergence is guaranteed.

6.4.2 Algorithm parameters and hyper-parameters selection

Each agent runs a QL algorithm with the following configuration: ϵ-greedy policy with $\epsilon = 0.1$, $\alpha = 0.1$, $\gamma = 0.9$, and $E = 5000$ episodes. These parameters worked well for a single agent solving the problem with one RS in one corner and one PS in the center. To obtain the threshold η, affinity κ, and tolerance τ hyper-parameters for the proposed partitioner, we first run a set of experiments in a grid of size 31×31. The values of n, τ, and both κ and η were incremented in steps of 1, 0.01 and 0.1, respectively.

To compute the communication cost Δ and ensure that our evaluations are hardware-independent, we use definition 2.2.4, resulting in the following formula:

$$\Delta = \frac{\lambda + \rho}{\mu} \tag{6.1}$$

Where λ is the cumulative number of the states exchanged after partitioning, ρ is the cumulative remote updates performed by the agents, and μ is the total learning time (number of iterations). In our implementation, and without loss of generality based on the complexity analysis of our algorithm, the number of remote updates represents only the update operation of a Q-value in a remote partition, and the number of states exchanged after partitioning is used as a good estimate of the number of messages required to move states between agents after partitioning. For each result, we computed the cumulative values of the μ and Δ. Then, μ and the Δ were normalized between 0 and 1 using the maximum and minimum values for μ and Δ among all the results. After normalization, the result with the best tuple (η, κ, τ) was selected following a multi-objective formulation of the problem since reducing the number of iterations may increase the communication cost. The globally optimized tuple is chosen as:

$$\min_{\eta,\kappa,\tau} 0.5\mu + 0.5\Delta \tag{6.2}$$

By solving the equation 6.2 using a hyper-parameter sweeping, we obtained $\eta = 0.3$, $\kappa = 1.0$, $\tau = 0.1$. This result is congruent with the features exposed by this scenario. For example, a combination of a low κ, which results in applying the local-affinity policy most of the time, and a high τ, which allows exploiting local states but (possibly) not optimal, was expected due to most of the states have more than one action that minimizes the length of the path towards the destination. Thus, when agents are traveling paths that increase the communication cost, i.e., two or more agents find the same shortest path, the local-affinity policy allows agents to explore alternative and optimal paths. Of course, other types of problems may require different values for τ to guarantee the convergence of the PRL algorithm.

The η seems to be a good trade-off between exploiting the intra-partition found by the local optimization step and having a large temporal local partition with states available for the global optimization step. A low η may limit the capability of leaving local optima in scenarios where the agents explore partitions of different sizes. For example, if the PE is located in one of the grid corners, then the agent at the opposite corner on the diagonal will have to explore the most. With a low η, at some point in time, it will have a large number of states in its partition mainly visited by itself, but it will not have many states to get those located in remote partitions, which may be interesting for itself. In the same way, a high η will limit the exchange of states between partitions. Although increasing the number of states in the local temporal partition will allow more opportunities to exchange states, it will not happen since the intra-partitions will contain only a few states to trade with other agents.

6.4.3 Performance using optimal hyperparameters

To evaluate and compare the performance of the implemented CS-RL algorithm with our dynamic partitioner (DP) with local-affinity (LA) policy and globally optimized, CS-RL DP+LA Globally Optimized, we also implemented four variants of the CS-RL algorithm. The first is the fully distributed CS-RL with random partitioning. The second one is the CS-RL algorithm using our partitioner but no local-affinity policy, CS-RL DP-only with $\eta = 0$. The third one is the CS-RL algorithm with our partitioning strategy with the tuple (η, κ, τ) that is locally optimized for a given β and n, CS-RL DP+LA Locally Optimized. Finally, a CS-RL using an optimal partitioning was implemented to have a reference to a lower bound communication cost.

Note that a comparison against the one with only dynamic partitioning gives us a good view of the positive impact of our local-affinity policy, while comparing against the DP+LA that is locally optimized provides a baseline to measure how good the global optimized parameters, i.e., obtained among all the results, generalize to different variations of the problem (n, β, problem size).

Figure 6.2 shows the communication cost and the iterations per episode (training curve) during the evaluated algorithms' training period with eight agents, $\beta = 100$, and 5000 episodes. The CS-RL with random and static partitioning has the highest communication cost, which is over 0.85. This value is expected since the probability of having two consecutive states in the same partition is around $1/n$. CS-RL with only dynamic partitioning obtains a higher reduction in the communication cost, near 50% compared to using a random partitioning. However, as it does not have preferences for exploring local states, only the $\epsilon-$greedy policy provides some exploration, and it gets stuck in a local optimum.

Table 6.3: Numerical range of the algorithm's variables.

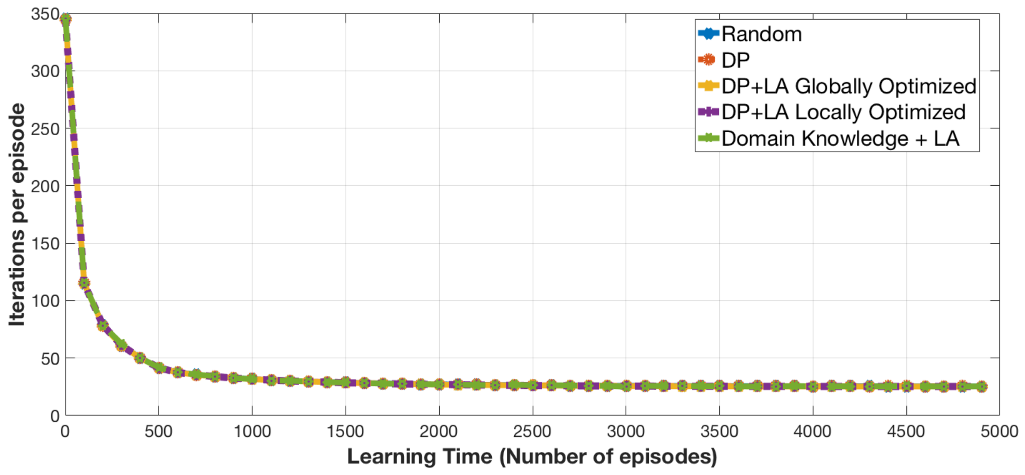
Parameter/Variable	**Range**
α	$[0, 1]$
γ	$[0, 1)$
ϵ	$[0, 1]$
η	$[0, 1]$
κ	$[0, 1]$
τ	$[0, 0.01]$
β	$[10, 50, 100]$
$\lvert N \rvert = \text{n}$	$[2, 8]$
Square grid side	$[31, 51, 71, 101, 301]$

CS-RL with dynamic partitioning and local-affinity policy obtain the best results, and the communication cost is reduced to less than 0.02 in both cases (globally and locally optimized). In other words, less than 2% of the total changes in the QT are performed by remote agents. Since the locally optimized version uses the best tuple (η, κ, τ) for the specific grid size, n and β, its communication cost is reduced at the highest rate. However, the globally optimized gets comparable results. Regarding iterations per episode, the partitioner did not affect the convergence of the PRL algorithm.

Finally, it is noticeable that after converging the RL agents, they still minimally communicate. This results from the problem definition and is not a limitation of the proposed algorithm. In our evaluation scenario, the picking station can be accessed in four directions. It means that when there are more than four agents, some agents need to share information to exploit the access states surrounding the picking station. In more practical scenarios, the picking station may be only accessible via one single cell. Therefore, the lower bound for the number of messages that need to be exchanged is $\Omega(n + |S|)$.

6.4.4 Number of agents

Figure 6.3 shows the cumulative communication cost against the increase in the number of agents in a 31x31 grid and $\beta = 100$. Since more agents are solving the problem concurrently, the probability of updating remote states and communication costs increases. CS-RL with random partitioning performs worst and obtains a communication cost proportional to $\frac{n-1}{n}$. CS-RL with only dynamic partitioning significantly improves and maintains a cumulative communication cost lower than 0.55. The proposed algorithm using the local and global optimized hyperparameters always keeps the communication cost lower than 0.42 and 0.38, respectively. These results show the positive impact of



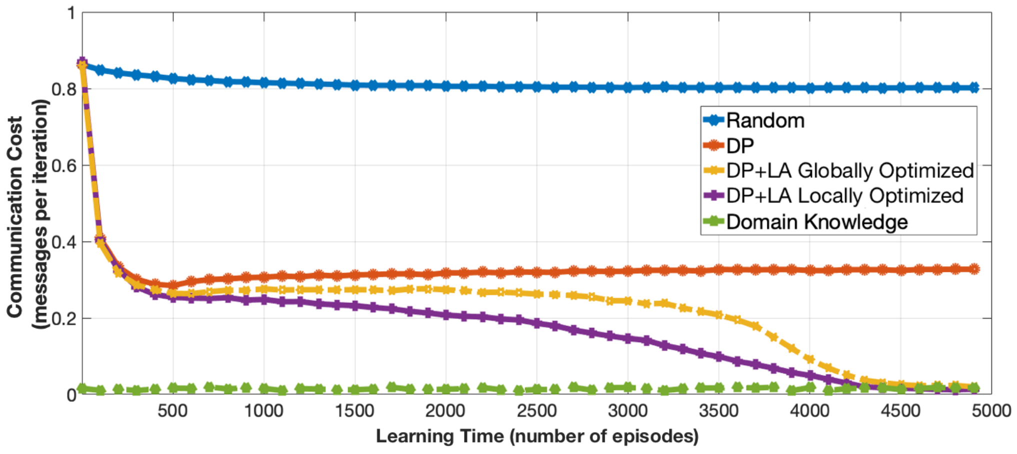


Figure 6.2: Learning curve (left) and communication cost (right) of the evaluated algorithms. Grid size 31x31, 8 agents and $\beta = 100$.

using the local-affinity policy to explore more states in the local partition. It is essential to show that even if the number of agents increases, the relative improvement among agents remains almost constant.

Note also that the cumulative Δ increase concerning the n does not represent a scalability issue of the proposed algorithm but an increase of the ρ and/or λ due to concurrency. In fact, Figure 6.2 shows how our algorithm could also converge with eight agents. Finally, although the proposed algorithm with locally optimized hyperparameters obtains the best performance, the tuple (η, κ, τ) used in the globally optimized one gets close results. These results are independent of the number of agents n, grid size, and β, as shown in the following subsections.

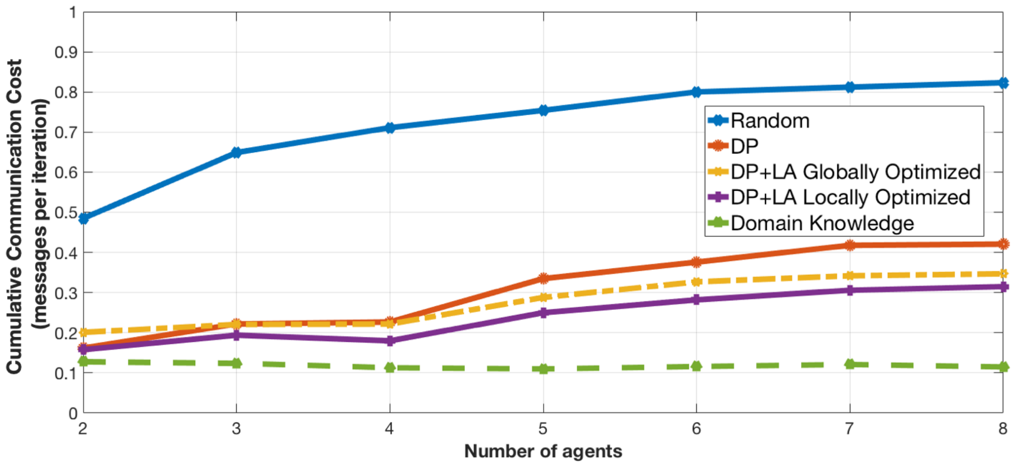


Figure 6.3: Impact of the number of agents in the communication cost with grid size 31x31 and $\beta = 100$.

6.4.5 Problem size

We have shown that the globally optimized tuple (η, κ, τ) provides good results for a different number of agents and β in a small grid with a fixed size of 31x31. However, it is important to verify that our algorithm is generic enough to reuse the hyperparameters found in small problems on larger ones without losing performance and, simultaneously, avoid extra training time by performing a hyperparameter sweep again. Figure 6.4 shows how the tuple $(\eta = 0.3, \kappa = 1.0, \tau = 0.1)$, which was found in Section 6.4.2 for a 31x31 grid with a state-action space of size 3844 q-values (31×31 states $\times$ 4 actions), also performs well in larger grids. Note that since we only have optimal values for (η, κ, τ) with different n and $beta$ on a fixed problem size (Section 6.4.2), then the curve for DP+LA Locally Optimized is not presented in Figure 6.4.

While the communication cost of CS-RL with random partition remains above 0.8, CS-RL with dynamic partitioning only and DP+LA using the globally optimized tuple remain below 0.42 and 0.32 with minimum values of 0.28 and 0.08, respectively, for all the evaluated grids. It means that the performance of our proposal is independent of the problem size, and It can be applied to problems with large state-action spaces by using the optimized tuple (η, κ, τ) found in a smaller problem. A more detailed statistical analysis of the results shows that when the problem size increases, the minimum values for the communication cost tend to be a little higher. For example, when the algorithm converged, the minimum communication cost was 0.08 in a grid of 301x301 but 0.014 in a grid of 51x51. However, it is still below the expected 0.1% of the $\epsilon-$greedy policy with $\epsilon = 0.1$.

6.4.6 Episodes Before Partitioning

The changes in the β parameter directly affect the number of remote updates ρ. If we execute the partitioner less often, i.e., we increase β, then both the ρ and μ will

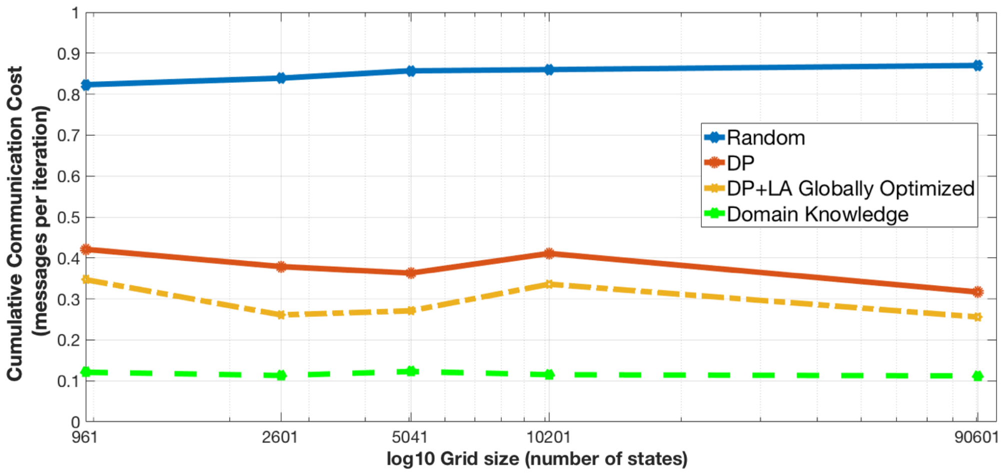


Figure 6.4: Impact of varying the size of the problem. The number of agents was fixed to 8.

increase. At the same time, λ will decrease because the RL algorithm will provide more information about the q-values that are more interesting for the agent. On the contrary, when PARTITIONER is called too often, then more states will be exchanged, i.e., λ will increase and may become comparable to ρ and, therefore, the overall communication overhead increases.

For this specific problem, Table 6.4 shows that the ρ value adds the highest contribution to the cumulative communication cost. In this evaluation scenario, this result is expected since the RL agents converge fast to an optimal solution. This allows agents to have just a few calls to PARTITIONER before the states are distributed to the correct partition. Finally, λ may also increase independently of the β. For example, in dynamic environments, where the transition probability function is stochastic, the same (s, a) pair may take the system to different states. Then, the locality of a state, which is exploited by the local-affinity policy, may also change each time.

6.4.7 Energy consumption and total communication overhead

Our previous results demonstrated the benefits of using PRL with built-in communication awareness. As presented in Section 6.3.1, the main objective of robots was to learn the path from their RS to the location of the PS, where they leave the transported inventory while minimizing energy consumption related to wireless communication and travel distance. By solving this problem through MDP, the robots successfully learned the optimal routes, aligning with the energy-saving objectives, i.e., the distance traveled is the shortest and also the number of visited states.

Let us compare the two extreme cases in the largest grid of size 301x301, 8 agents, and β 100 using the values presented in Table 6.5. On the one hand, we have random partitioning with a cumulative and minimal communication cost Δ of 0.87. This is translated into performing transmissions due to remote updates 87% of the time. On the other hand,

Table 6.4: Impact of Remote Updates and Exchange States in the Communication Cost.

Algorithm	**Cumulative**	β					
		10		50		100	
		Total	% Δ	Total	% Δ	Total	% Δ
Random partitioning	Remote updates (ρ)	154743	100.0	157062	100.0	156389	100.0
	Exchanged states (λ)	0	0.0	0	0.0	0	0.0
	Communication Cost (Δ)	0.82	100	0.83	100	0.82	100
Dynamic Partitioning (DP) only	Remote updates (ρ)	76304	83.6	74936	99.4	79812	99.8
	Exchanged states ((λ))	14964	16.4	431	0.6	200	0.2
	Communication Cost (Δ)	0.48	100	0	100	0.42	100
DP+LA globally optimized	Remote updates (ρ)	54837	98.0	61997	99.7	65577	99.8
	Exchanged states (λ)	1113	2.0	164	0.3	117	0.2
	Communication Cost (Δ)	0.30	100	0.33	100	0.35	100
DP+LA locally optimized	Remote updates (ρ)	54837	98.0	56010	99.4	59392	99.7
	Exchanged states (λ)	1113	2.0	340	0.6	191	0.3
	Communication Cost (Δ)	0.30	100	0.30	100	0.32	100

DP+LA locally optimized has a cumulative Δ of 0.32 and minimal Δ at convergence of 0.08. The minimal Δ is due to remaining transmissions needed to share information to exploit the access states surrounding the PS derived from the problem definition, as explained in Section 6.4.3.

Interestingly, despite our goal of minimizing energy consumption and finding the shortest path, random partitioning led to 2.7 times more cumulative transmissions than DP+LA optimization and ten times more transmissions even after convergence. As an example, assuming 802.11ac APs with transmitting power of 14dBm, 20MHz channel, and 64 Byte payloads as presented in [185], the mean energy consumption will be 260.65 Joules/packet (J/packet) in the worst case, e.g., where agents have no local states in their partitions, 226.7 J/packet with random partitions, 83.4 J/packet with DP+LA locally optimized, and 20.8 J/packet with DP+LA locally optimized after convergence.

Table 6.5: Comparison of transmitted packets between random and DP+LA locally optimised partitioning.

Partitioning	**Cumulative Δ (until convergence)**	**Minimal Δ (after convergence)**
Random	0,87	0,87
DP+LA locally optimized	0,32	0,08
Reduction transmitted packets	**63,22**	**90,80**

6.5 Conclusion

In this chapter, we proposed a general-purpose partitioning algorithm to enhance PRL algorithms and improve the execution time by minimizing the communication overhead of RL-based applications in distributed environments. To the authors' best knowledge, this is the first work focused on solving the communication overhead of distributing PRL algorithms without requiring any a priori information about the environment. The proposed algorithm combines a dynamic partitioning strategy, which iteratively splits the main problem into multiple small and loosely coupled ones without requiring any a priori domain knowledge, with an efficient co-allocation of processing and storage that minimizes agent communication. Performance evaluations showed that the proposed algorithm incurs no communication cost when the PRL algorithm converges and is scalable when more agents are added. Moreover, it removes the assumption of having domain knowledge or unlimited resource capabilities to perform an optimal partitioning of the problem, which is hard to meet in real, and usually unknown, environments.

Finally, further evaluations have demonstrated the versatility of our algorithm, as factors such as problem size and the number of agents have a minimal impact on its performance. Our proposed approach enables RL-based distributed applications to expedite their learning process by leveraging multiple agents in parallel while maintaining low communication overhead. This makes it possible to deploy this multi-agent system even on resource-constrained devices. Additionally, we have shown that PRL agents, equipped with built-in communication awareness, offer a mechanism to optimize resource utilization (e.g., spectrum, bandwidth, and energy). This optimization is achieved indirectly by strategically minimizing redundant or non-critical communication during learning. In contrast, RL agents with optimization objectives solely focused on resource usage, e.g., as a task in a Cognitive Radio (CR), may not achieve the same level of communication overhead reduction, given the learning procedure's inherent nature.

Chapter 7
Conclusions

In this chapter, we summarize the main contributions of this dissertation that were mainly directed to address the problem statements presented in Section 1.2. These contributions are the results of investigating the research questions formulated in Section 1.3, which will be answered by validating the hypotheses presented in 1.4. We finalize this chapter by analyzing the open challenges after this research's different contributions and future prospects.

7.1 Main research contributions

This dissertation aims to provide a novel framework for spectrum sharing based on recent advances in Deep Learning (DL) to tackle research problems P_1 to P_5 and design them such that we can overcome some of the overlooked limitations when using or when deploying them in wireless networks to tackle the research problems P_4 to P_6, respectively. In this context, let us recall the two main hypotheses that we stated in Section 1.4 for this research:

[H_1] **Data-driven Technology Recognition (TR) is a practical approach to enable spectrum sharing in Collaborative Intelligent Radio Networks (CIRNs). Even with a small number of labeled measurements, acceptable classification performance can be achieved.**

[H_2] **To efficiently employ data-driven methods in CIRNs, it is crucial to leverage collaborative learning algorithms capable of operating over distributed infrastructures with minimal communication overhead, thereby ensuring practical implementation and scalability.**

In order to validate these hypotheses, we have defined four research questions that were presented in Section 1.3. Through this dissertation, we have provided a set of contributions to answer such questions. Specifically, four main research contributions were identified in this dissertation:

[C_1] **A label-efficient TR as enabling technology for next generation spectrum sensing techniques for Dynamic Spectrum Access (DSA)**. [Chapter 3, research problem P_4 and enabler technology to solve research problems P_1, P_2, and P_3]

TR will play an essential role in how new wireless technologies make decisions to use the available spectrum efficiently and coexist with any new, legacy, and even unknown technologies. In Chapter 3, we proposed a novel Semi-supervised Learning (SSL) approach for TR that minimizes the need for labeling large data sets of spectrum data. More precisely, the main findings in this contribution are the following:

- We introduced an SSL approach for wireless TR that can work on raw In-phase and Quadrature (IQ) samples and does not require the whole data set to be labeled, which is a time-consuming and challenging task.
- The proposed scheme was implemented using Deep Autoencoders (DAEs), which requires an unlabeled data set and only a few labeled examples. The scheme's performance was evaluated against a Deep Neural Network (DNN) architecture that requires the whole data set to be labeled. We showed that the proposed scheme outperforms the DNN while requiring a limited labeled data set.
- We validated our approach in Colosseum [19, 18], the world's largest RF channel emulator built for the DARPA Spectrum Collaboration Challenge (SC2) competition. The results demonstrated that the designed algorithms successfully recognized sixteen unknown wireless technologies with an accuracy above 97% using the entire data set and > 70% using only 10%. This translates to 4.6x times better accuracy than the DNN model using the same amount of labeled data. Using the Defense Advanced Research Projects Agency (DARPA) Colosseum, we provided strong evidence about the robustness of the proposed approach.

These three findings provide a positive answer to the research question Q_2: **Can we formulate the TR problem for DL-based algorithms such that they can use labeled and unlabeled data and design robust systems that can deal with different amounts of them?**.

[C_2] **A novel architecture for next-generation spectrum sharing frameworks**. [Chapter 4, research problems P_1 and P_2]

In Chapter 4, we presented the architectural design and the experimental validation of the next generation of a spectrum-sharing framework with the capabilities to protect the incumbent. Specifically, the main findings in this contribution are the following. Specifically, the main findings in this contribution are the following:

- We designed a two-tier architecture that enables efficient spectrum sharing, built on top of the concept of CIRN and with the guarantee of incumbent protection. Compared to new approaches like Citizens Broadband Radio Service (CBRS) and Licensed Shared Access (LSA), our system requires no central infrastructure to control and grant access to the shared spectrum. It only requires that the incumbents collaborate with other networks by sharing information such as location, interference measurements, and frequency operation parameters. This approach takes spectrum-sharing frameworks to an entirely new era compared to the state-of-the-art spectrum-sharing systems, which are mainly database-assisted [186].

- We designed and implemented a two-step Artificial Intelligence (AI)-based algorithm that, combined with collaborative information, can recognize, learn, and proactively forecast incumbent's transmissions in near real-time and with an accuracy above 95%. Compared to recent works in AI for spectrum management [187], this is the first work that proposes a system to protect the incumbent's transmissions in a collaborative environment to the author's best knowledge.
- We validated our approach in Colosseum [19], the world's largest RF channel emulator built for the SC2 competition, using up to two incumbents with different transmission patterns simultaneously and sharing spectrum with up to 5 additional networks, each one composed of up to 10 nodes.

The results of this contribution positively answer the research question Q_1: **Can we design an architecture for a spectrum sharing system that does not require any central infrastructure to control and grant access to a shared spectrum?**.

[C_3] **A novel framework to achieve Traffic Classification (TC) at any layer on the radio network stack**. [Chapter 5, research problems P_3 and P_5]

In Chapter 5, we introduced a novel framework to achieve TC at any layer on the radio network stack. Building on top of it, a procedure based on DL to perform TC on spectrum samples is proposed. This procedure enables the management algorithms running at the Gateway (GW) nodes (or beyond) to perform better by having a broader view of the traffic flowing in the shared spectrum. The main findings of this chapter are summarized as follows:

- We presented a general framework that enables the development of TC algorithms optimized for wireless networks. To the best of our knowledge, this is the first framework that allows the development of Radio Access Technologies (RAT)-agnostic spectrum-based TC algorithms.
- We proposed a spectrum-based TC procedure that exploits the proposed framework's functional blocks and works on Physical Layer (L1) packets. Compared to similar works like [119], the proposed procedure includes the traffic classifier design and the complete chain to achieve spectrum-based TC. Moreover, the proposed approach removes the need for specialized algorithms to separate/aggregate users' traffic flows (e.g., a radio identification procedure [188]), as the one required in recent approaches like [117], as it uses as classification object single L1 packets (as a sequence of raw IQ samples).
- We designed and evaluated a DL architecture based on Convolutional Neural Networks (CNNs) to solve the task of classifying packets directly on spectrum data. To the best of the authors' knowledge, this is the first work that uses this type of DNN to solve such a task. Moreover, we demonstrate that the proposed architecture outperforms a Recurrent Neural Network (RNN) architecture, also used in [119], and that is traditionally used to solve classification problems with time series as input data.
- We presented the first detailed analysis of the performance achieved by different spectrum-based classifiers using DL architectures in terms of 1) classification accuracy on different classification tasks at different radio stack layers (L2

and L7), including a comparison against encrypted L2 byte-based packet classifiers, and 2) execution time in training and inference using 802.11 standard-compliant L1 packets and with input sequences of more than 3K spectrum samples. These evaluations complement and extend previous results where the performance of DL approaches solving the TC problem has been evaluated and compared using byte [189] and spectrum representation of the packets [119, 117]. Moreover, these results provide initial insights into the feasibility of this approach for real-time classification.

- We create and provide an open-source dataset that contains 802.11 standard-compliant L1 waveforms. The waveforms are generated by different 802.11 technologies (b, g, n), which results in different transmission schemes such as Direct-Sequence Spread Spectrum (DSSS) in 802.11b and Orthogonal Frequency-Division Multiplexing (OFDM) in 802.11g/n, different types of L2 frames (management, control, and data), and multiple Modulation and Coding Scheme (MCS). Moreover, the payloads carried by these L1 packets (information at L2 and above) were generated using real traces of the L7 application running on a mobile device and connected to a secured 802.11 Access Point (AP) with Wi-Fi Protected Access (WPA)-2. This is the first open and available dataset for testing traffic classification at the spectrum level. We believe this dataset would foster reproducibility and allow further advances on this topic. The dataset and the code associated with this contribution can be obtained in Zenodo[1] and Github[2], respectively.

These results provide a positive answer to the research question Q_3: **Can we design a general framework that enables the development of TC algorithms optimized for wireless networks?**

[C_4] A novel general-purpose partitioning algorithm that enhances Parallel Reinforcement Learning (PRL) algorithms to support the execution of Reinforcement Learning (RL)-based applications in distributed environments with improved execution time and minimal communication overhead. [Chapter 6, research problem P_6]

Chapter 6 presented a novel general-purpose partitioning algorithm that enhances PRL algorithms to support the execution of RL-based applications in distributed environments with improved execution time by making the agents communication-aware by design. The main findings of this contribution are three-fold:

- We provided a novel approach that allows any table-based PRL algorithm to run RL-based applications with minimal communication overhead in a distributed environment. To the authors' best knowledge, this is the first work that focuses on solving the communication overhead of distributing PRL algorithms without requiring any a priori information about the environment, making the agents communication-aware by design. We achieved this by combining a dynamic partitioning strategy with an efficient heuristic that performs a co-allocation of processing, i.e., the RL agent, and storage, i.e., Q-table, that minimizes agent communication. Our approach exploits the agent's exploration capabilities to build the necessary domain knowledge to divide the state-action space into multiple small and loosely coupled partitions and assign a given partition to the agent that exploits it the most.

[1]https://doi.org/10.5281/zenodo.5208200
[2]https://github.com/miguelhdo/tc_spectrum

- We designed and implemented a local-affinity policy that exploits the locality of state-action values in a given partition without affecting the convergence guarantee of the RL algorithm. This policy, which is executed after the learning policy of the RL agent, includes a second action selection filter to detect which (semi) optimal actions may take the agent to a new state located in the same partition as the actual state.
- We provided performance evaluations of a PRL system solving the robot navigation problem in a smart factory and showed that the proposed algorithm incurs almost no communication cost in a converged state and is scalable in the number of RL agents solving the problem. Moreover, the results showed that the communication-aware learning capability of the agents indirectly drastically reduces the usage of available resources, e.g., energy consumption or spectrum usage due to wireless communication transmissions, which may complement the system's overall efficiency.

The results of our final contribution provide a positive answer to the research question Q_4: **Can we provide communication-awareness capabilities to PRL agents to reduce the communication overhead while deploying them in distributed infrastructure without requiring any a priori information about the deployment environment?**

Figure 7.1 summarizes the contributions presented in this dissertation in the context of Machine Learning (ML)-based algorithms for the next-generation spectrum-sharing frameworks using CIRNs. As mentioned before, part of the success of this research is that the TR module and the pattern prediction algorithm were implemented in the SCATTER radio [15] and evaluated in the Colosseum testbed during the DARPA SC2.

A complete visualization of all the contributions in this dissertation is presented in Figure 7.2. This figure is an extension of the one presented in Figure 4.2 and indicates where the TC block would be placed and how a control- or user-plane application can be deployed to run a PRL-based algorithm. Notice that although the example used in Chapter 6 is the robot navigation problem in a smart factory, it is expected that in the future, the communication system integrated into them will be based on intelligent radios and that Multi-Agent Reinforcement Learning (MARL)-based algorithms at the user (e.g., the presented item-fetch application for autonomous vehicles in a smart-factory) and/or the control plane (e.g., wireless routing) will be running and exchanging information over the air.

Let us now analyze these contributions and how they support our two hypotheses from the perspective of the problem statements presented in Section 1.2. Hypothesis H_1 envisions a next-generation spectrum sharing framework with incumbent protection based on the concept of CIRNs, which directly addresses the problem statements P_1 and P_2 via C_2. Specifically, C_2 provides an architecture for next-generation spectrum sharing frameworks with a guarantee of incumbent protection that does not require any central infrastructure to control and grant access to a shared spectrum, providing the solution to the problem statement P_1 (**Centralized multi-tier spectrum sharing models can not scale**). Simultaneously, this framework is built on top of the concept of CIRNs and contains a two-step ML-based algorithm that, combined with collaborative information, can recognize, learn, and proactively forecast incumbent's transmissions in near real-time. This outcome directly tackles the problem statement P_2 (**Cognitive Radio (CR)**

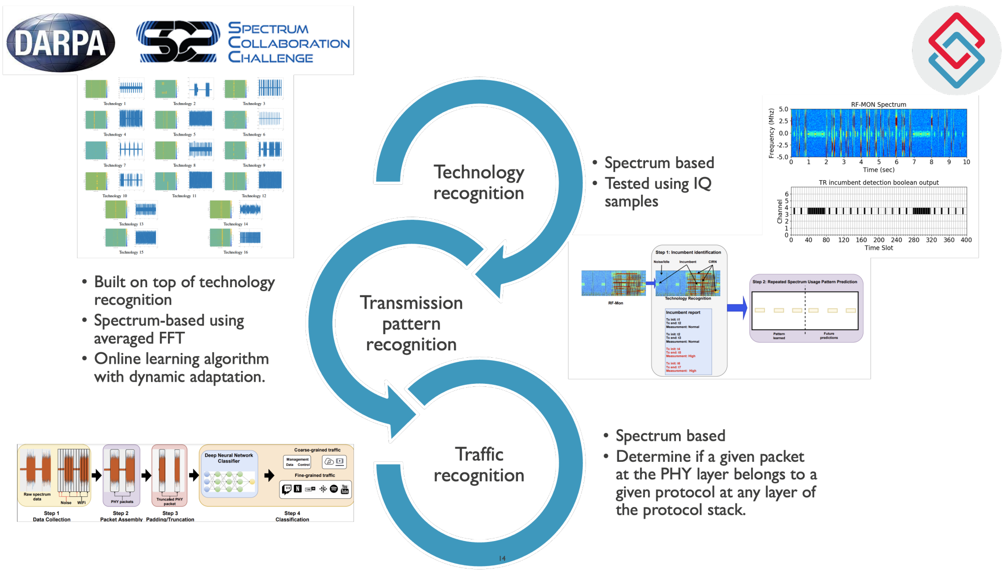

DARPA
SPECTRUM COLLABORATION CHALLENGE
Technology recognition
• Spectrum based
• Tested using IQ samples
RF-MON Spectrum
TR incumbent detection boolean output
Transmission pattern recognition
• Built on top of technology recognition
• Spectrum-based using averaged FFT
• Online learning algorithm with dynamic adaptation.
Step 1: Incumbent identification
Technology Recognition
Step 2: Repeated Spectrum Usage Pattern Prediction
Traffic recognition
• Spectrum based
• Determine if a given packet at the PHY layer belongs to a given protocol at any layer of the protocol stack.
Step 1 Data Collection
Step 2 Packet Assembly
Step 3 Padding/Truncation
Step 4 Classification
Deep Neural Network Classifier
Coarse-grained traffic
Fine-grained traffic

Figure 7.1: Research contributions in the context of CIRN. The two first components have been validated and demonstrated during the DARPA SC2 competition (team SCATTER). The third component has been validated using emulated data, and the data set is publicly available for repeatability of the research.

technologies can not share and reuse spectrum efficiently as they work in isolation and only local information).

Complementing C_2, C_3 provides a general framework to perform TC at any layer of the radio network, which removes the limitation of TC systems that are based on packets at Link Layer (L2) (or above) when using them in wireless environments. This novel capability allows the radio to sense and understand the environment state based on the traffic that flows over the spectrum to adapt the radio parameters dynamically, so the users' requirements are fulfilled while optimizing the shared spectrum usage, which addresses the problem statement (P_3 **TC systems are not designed to support the spectrum management decision-making processes for radio networks in a shared spectrum**).

The use of SSL to design a DL model for TR in C_1 allows the use of all the collected spectrum data to train the DL while only requiring labeling a part of it to achieve a competitive performance compared to a model that requires the whole dataset to be labeled, solving the problem statement P_4 (**Collecting spectrum data to train DL models for spectrum sensing is easy, but labeling is hard**). At the same time, the proposed algorithm is a critical functional block of the contributions C_2 and C_3, supporting hypothesis H_1 too. In this regard, the designed and evaluated a DL architecture based on CNNs to solve the task of classifying packets directly on spectrum data from contribution C_3 provided experimental validation of the feasibility to deploy this approach for real-time operations, addressing the problem statement (P_5 **DL algorithms have outperformed traditional optimization methods in solving several networking problems when considering learning-related performance metrics like accuracy, but ML practitioners tend to neglect equally essential metrics related to the execution of these algorithms like inference time**).

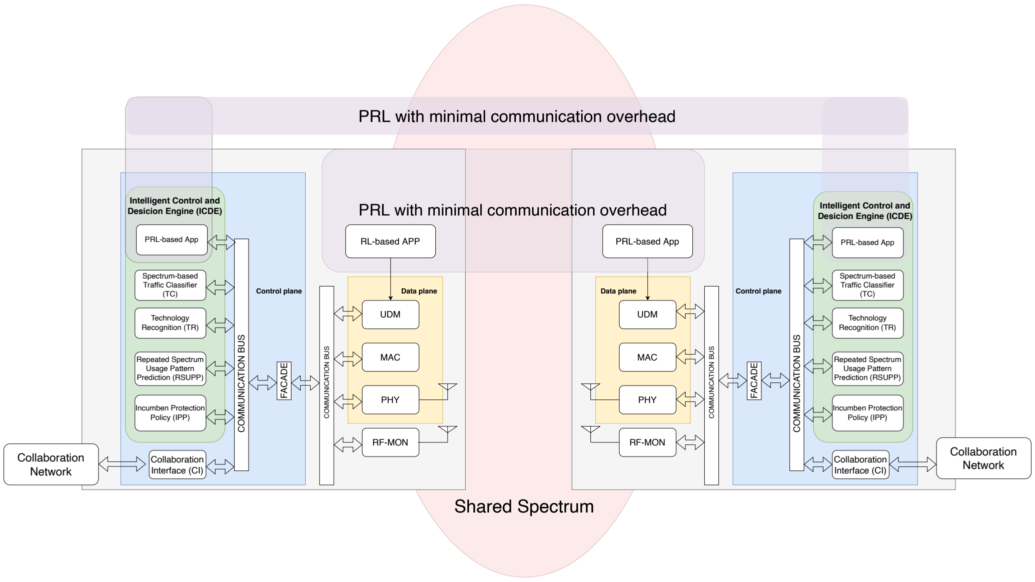

PRL with minimal communication overhead
PRL with minimal communication overhead
Intelligent Control and Desicion Engine (ICDE)
PRL-based App
Spectrum-based Traffic Classifier (TC)
Technology Recognition (TR)
Repeated Spectrum Usage Pattern Prediction (RSUPP)
Incumben Protection Policy (IPP)
COMMUNICATION BUS
Control plane
FACADE
Data plane
RL-based APP
UDM
MAC
PHY
RF-MON
Collaboration Network
Collaboration Interface (CI)
PRL-based App
Shared Spectrum

Figure 7.2: Combining the different contributions of this dissertation into the SCATTER CIRN.

Hypothesis H_2 focuses on leveraging collaborative learning algorithms to efficiently employ data-driven methods in CIRNs, addressing problem statement P_6. For this, Contribution C_4 provides a novel general-purpose partitioning algorithm that enhances PRL algorithms. This enhancement supports the execution of RL-based applications in distributed environments with minimal communication overhead and without requiring any a priori information about the environment, making the agents communication-aware by design, thus addressing problem statement P_6 (Table-based RL algorithms are not designed to run in decentralized networking environments as they are not communication-aware). Compared to the other contributions, which directly support hypothesis H_1 by providing tailored solutions to radio network problems, C_4 offers a more generic solution. It enables any table-based PRL algorithm to run on a distributed network, independently of the specific problem being solved, including CIRNs. In other words, when novel solutions are developed on top of C_4 and tailored to learn and solve problems in CIRNs, they will address the learning scalability issues in table-based RLs in centralized environments (P_1) and the inefficiency of traditional CR technologies when learning in isolation (P_2).

7.2 Open challenges and future prospects of this research

In the previous section, we presented this dissertation's main contributions and how they answered the research questions and validated the stated hypotheses. However, technology is evolving every day, and new challenges emerge at the same peace. Therefore, it is crucial to analyze the open challenges this dissertation has left and the perspective of this research from both the radio spectrum management and ML point of view.

7.2.1 Autonomous spectrum management frameworks

It is essential to recognize that although we demonstrated that CIRNs could collaborate to share spectrum efficiently while protecting the incumbents autonomously, we believe and hope this research brings new research challenges where game theory, AI, and wireless networks converge all together. More specifically:

1. The proposed architecture is based on decentralization and collaboration. The DARPA SC2 competition provided a point score system to incentivize collaboration [96, 16]. However, that scoring system was designed for a competition, not a real use-case scenario. Therefore, further research is required to design and validate practical mechanisms to incentivize the collaboration among CIRNs toward a common objective like fair spectrum sharing and incumbent protection.
2. Today's integration of AI-based functionalities in networking devices is still in the early stages. Most algorithms are designed and trained offline to be later deployed on devices without automation. Unfortunately, this approach is not scalable, error-pruned, and reduces the opportunities for adoption at a large scale. Therefore, further efforts are required towards designing an end-to-end AI-native architecture in 5G and beyond to support advanced technologies like CIRNs.

7.2.2 ML-based functions for autonomous spectrum management

In this dissertation, we presented two state-of-the-art ML-based functionalities to provide novel capacities to CIRNs: TR and TC at any layer. However, during the design of the DNN architectures and its implementation, the following open challenges remain:

7.2.2.1 Technology Recognition

1. The proposed DL model for TR can be implemented to use either IQ samples, as shown in Chapter 3, or average Fast Fourier Transform (FFT) values, as in Chapter 4. There is a trade-off between selecting one of the other, e.g., raw IQ removes the need for any transformation of the spectral data, but the models require a bigger DNN to achieve good performance. However, similar works on modulation recognition, which is closely related to TR, have proposed other input representations of the spectral data, such as spectrograms [190] or wavelet transform [191]. As the DL model proposed for TR is also performing automatic feature extraction on raw IQ samples, further research is required to evaluate and analyze the impact of the input representation on the model complexity and performance, an important aspect to consider when implementing and deploying the algorithm in real hardware.
2. The workflow of the proposed SSL approach for TR provides the building block to gather data from the radios, pre-process the data for exploiting unsupervised learning, and only label a few samples per class. However, it is important to include a mechanism to automatize the whole workflow, and Machine Learning Model Operationalization Management (MLOps) frameworks would be ideal for this. However, how do we integrate MLOps into the system to provide the ML

workflows required for this task? Moreover, it is expected that functionalities such as TR would become virtualized functionalities that can be easily deployed on any radio. Therefore, how can we provide management and orchestration capabilities for intelligence to the radios? These are open questions that recent projects are trying to answer from an architectural point of view [192], and that can be explored and extended to the radio domain.

3. In general, traditional methods such as Likelihood-Based (LB) and expert Feature-Based (FB) engineering combined with pattern recognition have been outperformed by supervised DL methods in the task of TR. Supervised DL methods remove the need for expert knowledge about the environment and the signal features used for classification by using the power of automatic feature abstraction. However, it requires the whole data set to be labeled. Labeling becomes time-consuming and challenging for both the technologies to be recognized and the environment to be entirely unknown. To overcome these limitations, in this chapter, we propose an SSL approach for TR that separates the feature extraction from the classification task in the DL architecture, so the use of unlabeled data is maximized. At the same time, the proposed approach minimizes the use of domain expertise knowledge by requiring only a small portion of the entire data set to be labeled to obtain a good performance, which is not the case with supervised DL models.

4. Although SSL using DAE remains state of the art for achieving label-efficient TR, several techniques have emerged in recent years. Self-Supervised Learning (Self-SL)[72] leverages large amounts of unlabeled data to generate predictive signals. Transfer Learning (TL)[73] allows the adaptation of pre-trained models to new tasks with minimal labeled data. Few-Shot Learning (FSL)[74] and Meta-learning (Meta-L)[75] enable rapid generalization from limited examples, addressing the challenges of scarce data. These recent advances also open new opportunities to develop novel architectures that further reduce the need for labeled data, utilizing either these novel learning paradigms or more advanced DAE architectures [78, 79].

7.2.2.2 Traffic Classification using spectrum data

1. The DL model for TC should be validated on L1 packets affected by real channel conditions as a performance decrease is expected is expected as demonstrated in [117, 118]. Depending on the results, several mechanisms can be used to minimize the negative impact on accuracy. For example, the mechanism we used to create the dataset, i.e., generating synthetic standard-compliant L1 packets that carry real L2 frames, can perform data augmentation to improve generalization.

2. The evaluations were only carried out on the proposed DL models to solve the TC tasks. Although the prediction time was very promising, we only focused on the classification tasks with L1 packets, where the models run on high-end hardware. In addition, we did not benchmark the pre-processing step nor compared the models against state-of-the-art TC systems on L2. Therefore, it is essential to benchmark the complete procedure in several implementation platforms. This includes Software Defined Radio (SDR)-only, SDR+host machine, and RAT on-chip for pre-processing and CPU, low-end GPUs, high-end GPUs for running the models, and their respective comparison against byte-based approaches. Those evaluations will provide an initial base on further improvement on the proposed

models such that they can run on constrained-resource devices and find a good trade-off between model size and accuracy while providing competitive results compared to TC systems on L2 packets.

3. As presented in Chapter 5, we created three different TC models to solve three different but related tasks. This approach does not scale well at the network edge as it involves a parallel execution of the various models (for training and inference) with the inevitable increment of computational and storage requirements. To overcome this problem, Multi-Task Learning (MTL) should be explored to reduce the computing/storage requirements, achieve higher performance, and simplify the training procedure. As recently investigated and demonstrated in [114] and combined with distributed learning approaches, such as Gossip Learning (GL) used in [116], MTL might speed up the learning process and increase the system's scalability. Alternative, or in combination with MTL, techniques like Neural Network (NN) quantization [193] and or pruning [194] to reduce the complexity of each model can be explored. A related problem is a possible bias and damage to the model performance caused by using fixed-length input as required by CNNs in the presence of significant variations in the input data's length. An alternative to address this problem is to explore RNNs again in the context of MTL (e.g., RNN-based Autoencoders (AEs)) to reduce the final complexity of the resulting model. However, a careful performance analysis should be conducted to determine if the reduction in computation complexity obtained using MTL compensates for the inevitable complexity increase of using RNNs to support input of variable lengths instead of CNNs.

4. Transmitting L1 packets over the air will result in receiving packets with a large variety of Signal-to-Noise Ratio (SNR) values compared to the values used in the generated dataset (20 to 30dB). Changes in the SNR values will modify the original signal and negatively impact the DL classifier's performance in the case of low SNR values, as shown in previous work [83, 130]. For this, data augmentation techniques can provide a data set with packets generated with more SNR values (e.g., in the range between -20 to 20 dB), so an SNR sensitivity analysis can be performed. With the resulting dataset, researchers could investigate mechanisms to mitigate the negative impact on the classifier's performance when packets are received with low SNR. For instance, denoising AEs as feature extractors may improve the performance in the presence of high noise levels. Additionally, the augmented dataset could foster the development and implementation of novel algorithms, closing the performance gap when a classifier is trained with a synthetic but standard-compliant dataset and deployed in a real environment.

7.2.3 Multi-Agent Reinforcement Learning running on distributed environments

AI-based closed-loop control will be a key aspect of future networks, and MARL algorithms will naturally empower many of them. Distributed intelligence running across the network, e.g., an AI-based functionality running on the radios, needs to be designed such that the cost of sharing the knowledge among agents over the network is minimal so that it has a minimal impact on the learning process. In this dissertation, we focused on PRL as a particular case of MARL. However, MARL is a framework, and several approaches can be followed to realize it, which results in the need for further research on designing and implementing similar mechanisms tailored to other MARL approaches. More precisely:

1. In the case of PRL, we need to investigate the performance of this algorithm further using other RL algorithms like State–action–reward–state–action (SARSA) and other types of problems from different domains (e.g., routing on wireless networks [35]). Besides, it may be helpful to design and implement a heuristic to automatically tune the tuple (η, κ, τ) to avoid a parameter sweeping to select them.

2. State-of-the-art MARL algorithms are based on Deep Reinforcement Learning (DRL) [195]. The core of a DRL agent is a supervised DL algorithm, and these algorithms suffer from scalability problems when running over a distributed network due to communication overhead [196]. One problem is that our approach for table-based PRL can not be directly applied to DRL agents as the knowledge about state-action space is not represented as a table but is encoded internally in the weights of the DNN. In this case, further research is required to design DRL algorithms that reduce the communication overhead by design.

Bibliography

[1] ETSI, "Autonomous Networks, supporting tomorrow's ICT business," 3rd Generation Partnership Project (3GPP), White Paper 40, October 2020, 1st Edition. [Online]. Available: https://www.3gpp.org/DynaReport/23288.htm

[2] T. Forum, "Autonomous Networks: Empowering digital transformation – from strategy to implementation," TM Forum, Whitepaper IG1305, September 2022.

[3] O-RAN Alliance, "AI/ML Workflow Description and Requirements v01.02.02," O-RAN Alliance, Technical Specification, 2020.

[4] A. Garcia-Saavedra *et al.*, "O-RAN: Disrupting the Virtualized RAN Ecosystem," *IEEE Communications Standards Magazine*, pp. 1–8, 2021.

[5] O-RAN Alliance, "Architecture Description v02.00.00," O-RAN Alliance, Technical Specification, 2020.

[6] Vodafone. An industrial 5g spectrum policy for europe. vodafone public policy paper. (Visited on 06-January-2020). [Online]. Available: https://www.vodafone.com/content/dam/vodcom/files/public-policy/5g-report/an-industrial-5g-spectrum-policy-for-europe.pdf

[7] P. Marshall, *Three-Tier Shared Spectrum, Shared Infrastructure, and a Path to 5G*. Cambridge University Press, 2017.

[8] F. C. Commission, "Spectrum policy task force," FCC, Report ET Docket No. 02-135, November, 2004.

[9] A. Karpathy. Software 2.0. Visited on 10-August-2021. [Online]. Available: https://karpathy.medium.com/software-2-0-a64152b37c35

[10] E. Research. Usrp n310. (Visited on 06-January-2023). [Online]. Available: https://www.ettus.com/all-products/usrp-n310/

[11] N. I. Corp. (2023) Software defined radio: Past, present, and future. Visited on 10-July-2023. [Online]. Available: https://www.ni.com/en/perspectives/software-defined-radio-past-present-future.html

[12] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding o-ran: Architecture, interfaces, algorithms, security, and research challenges," *Commun. Surveys Tuts.*, vol. 25, no. 2, p. 1376–1411, jan 2023. [Online]. Available: https://doi.org/10.1109/COMST.2023.3239220

[13] DARPA. (2016) Spectrum collaboration challenge (sc2). [Online]. Available: https://www.darpa.mil/program/spectrum-collaboration-challenge

[14] ——. Darpa spectrum collaboration challenge. Visited on 10-August-2021. [Online]. Available: https://www.darpa.mil/news-events/spectrum-collaboration-challenge-sc2

[15] S. D. Giannoulis, C. Donato, R. Mennes, F. A. P. de Figueiredo, I. Jabandzic, Y. D. Bock, M. Camelo, J. Struye, P. Maddala, M. Mehari, A. Shahid, D. Stojadinovic, M. Claeys, F. Mahfoudhi, W. Liu, I. Seskar, S. Latré, and I. Moerman, "Dynamic and collaborative spectrum sharing: The scatter approach," in *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, 2019, pp. 1–6.

[16] R. Mennes, J. Struye, C. Donato, M. Camelo, I. Jabandžić, S. Giannoulis, I. Moerman, and S. Latré, "Collaborative flow control in the darpa spectrum collaboration challenge," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2024–2038, 2020.

[17] R. Mennes, M. Camelo, M. Claeys, and S. Latré, "A neural-network-based mf-tdma mac scheduler for collaborative wireless networks," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, 2018, pp. 1–6.

[18] L. Bonati, P. Johari, M. Polese, S. D'Oro, S. Mohanti, M. Tehrani-Moayyed, D. Villa, S. Shrivastava, C. Tassie, K. Yoder, A. Bagga, P. Patel, V. Petkov, M. Seltser, F. Restuccia, A. Gosain, K. R. Chowdhury, S. Basagni, and T. Melodia, "Colosseum: Large-scale wireless experimentation through hardware-in-the-loop network emulation," in *2021 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, 2021, pp. 105–113.

[19] P. Tilghman, "Will rule the airwaves: A darpa grand challenge seeks autonomous radios to manage the wireless spectrum," *IEEE Spectrum*, vol. 56, pp. 28–33, 2019.

[20] N. U. Institute for the Wireless Internet of Things. Colosseum: The world's most powerful wireless network emulator. (Visited on 06-January-2023). [Online]. Available: https://www.northeastern.edu/colosseum/

[21] ITU-T, " Architectural framework for machine learning in future networks including IMT-2020," ITU-T, Recommendation, 2019.

[22] Y. Xiao, S. Shi, W. Lou, C. Wang, X. Li, N. Zhang, Y. T. Hou, and J. H. Reed, "Decentralized spectrum access system: Vision, challenges, and a blockchain solution," *IEEE Wireless Communications*, vol. 29, no. 1, pp. 220–228, 2022.

[23] M. Troglia, J. Melcher, Y. Zheng, D. Anthony, A. Yang, and T. Yang, "Fair: Federated incumbent detection in cbrs band," in *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, 2019, pp. 1–6.

[24] Z. Li, W. Wang, J. Guo, Y. Zhu, L. Han, and Q. Wu, "Blockchain-assisted dynamic spectrum sharing in the cbrs band," in *2021 IEEE/CIC International Conference on Communications in China (ICCC)*, 2021, pp. 864–869.

[25] M. M. Sohul, M. Yao, T. Yang, and J. H. Reed, "Spectrum access system for the citizen broadband radio service," *IEEE Communications Magazine*, vol. 53, no. 7, pp. 18–25, 2015.

[26] D. Stojadinovic and M. Buddhikot, "Design of a secondary market for fractional spectrum sub-leasing in three-tier spectrum sharing," in *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, 2019, pp. 1–8.

[27] T. W. Hazlett, "Assigning property rights to radio spectrum users: Why did fcc license auctions take 67 years?" *The Journal of Law & Economics*, vol. 41, no. S2, pp. 529–576, 1998. [Online]. Available: http://www.jstor.org/stable/10.1086/467402

[28] C. Baquero Barneto, T. Riihonen, M. Turunen, L. Anttila, M. Fleischer, K. Stadius, J. Ryynänen, and M. Valkama, "Full-duplex ofdm radar with lte and 5g nr waveforms: Challenges, solutions, and measurements," *IEEE Transactions on Microwave Theory and Techniques*, vol. 67, no. 10, pp. 4042–4054, 2019.

[29] S. Biswas, A. Bishnu, F. A. Khan, and T. Ratnarajah, "In-band full-duplex dynamic spectrum sharing in beyond 5g networks," *IEEE Communications Magazine*, vol. 59, no. 7, pp. 54–60, 2021.

[30] P. Soto, M. Camelo, J. Fontaine, M. Girmay, A. Shahid, V. Maglogiannis, E. De Poorter, I. Moerman, J. F. Botero, and S. Latré, "Augmented wi-fi: An ai-based wi-fi management framework for wi-fi/lte coexistence," in *2020 16th International Conference on Network and Service Management (CNSM)*, 2020, pp. 1–9.

[31] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *Siam Review*, vol. 60, pp. 223–311, 6 2016. [Online]. Available: http://arxiv.org/abs/1606.04838

[32] A. Halevy, P. Norvig, and F. Pereira, "The unreasonable effectiveness of data," *IEEE Intelligent Systems*, vol. 24, pp. 8–12, 3 2009. [Online]. Available: http://ieeexplore.ieee.org/document/4804817/

[33] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016.

[34] O. B. Sezer, E. Dogdu, and A. M. Ozbayoglu, "Context-aware computing, learning, and big data in internet of things: a survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 1–27, 2017.

[35] T. Hendriks, M. Camelo, and S. Latré, "Q <sup>2</sup> -routing : A qos-aware q-routing algorithm for wireless ad hoc networks," in *International Conference on Wireless and Mobile Computing, Networking and Communications*, vol. 2018-Octob, 2018.

[36] X. Chen, Y. Li, and L. Liu, "A coordinated path planning algorithm for multi-robot in intelligent warehouse," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2019, pp. 2945–2950.

[37] L. Busoniu, R. Babuska, B. D. Schutter, and B. D. Schutter, "A comprehensive survey of multiagent reinforcement learning," *Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 38, pp. 156–172, 2008.

[38] R. M. Kretchmar, "Reinforcement learning algorithms for homogeneous multi-agent systems," *Workshop on Agent and Swarm Programming*, 2003.

[39] S. D. Whitehead, "A Complexity Analysis of Cooperative Mechanisms in Reinforcement Learning," *AAAI-91 Proceedings*, pp. 607–613, 1991.

[40] K.-L. Du and M. N. Swamy, *Wireless communication systems: from RF subsystems to 4G enabling technologies*. Cambridge University Press, 2010.

[41] P. Baudin, *Wireless transceiver architecture: Bridging RF and digital communications*. John Wiley & Sons, 2014.

[42] J. W. Leis, *Digital signal processing using MATLAB for students and researchers*. John Wiley & Sons, 2011.

[43] V. K. Madisetti, *The Digital Signal Processing Handbook-3 Volume Set*. CRC press, 2018.

[44] Z. Zhu and A. K. Nandi, *Automatic Modulation Classification*. John Wiley & Sons, Ltd, 12 2014. [Online]. Available: http://doi.wiley.com/10.1002/9781118906507

[45] S. K. Jayaweera, *Signal processing for cognitive radios*. John Wiley & Sons, 2014.

[46] S. Haykin, "Cognitive radio: brain-empowered wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 2, pp. 201–220, 2005.

[47] J. Mitola and G. Maguire, "Cognitive radio: making software radios more personal," *IEEE Personal Communications*, vol. 6, no. 4, pp. 13–18, 1999.

[48] M. Camelo, R. Mennes, A. Shahid, J. Struye, C. Donato, I. Jabandzic, S. Giannoulis, F. Mahfoudhi, P. Maddala, I. Seskar, I. Moerman, and S. Latre, "An ai-based incumbent protection system for collaborative intelligent radio networks," *IEEE Wireless Communications*, vol. 27, no. 5, pp. 16–23, 2020.

[49] O. Chapelle, B. Schlkopf, and A. Zien, *Semi-Supervised Learning*, 1st ed., O. Chapelle, B. Scholkopf, and A. Zien, Eds. The MIT Press, 9 2006. [Online]. Available: https://direct.mit.edu/books/book/3824

[50] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[51] K. Tuyls and G. Weiss, "Multiagent learning: Basics, challenges, and prospects," *AI Magazine*, vol. 33, p. 41, 2012. [Online]. Available: https://aaai.org/ojs/index.php/aimagazine/article/view/2426

[52] E. Research. Introducing the most advanced sdr – the ni ettus usrp x410. (Visited on 06-January-2023). [Online]. Available: https://www.ettus.com/introducing-the-most-advanced-sdr-the-ni-ettus-usrp-x410/

[53] R. Mennes, F. A. P. D. Figueiredo, and S. Latré, "Multi-agent deep learning for multi-channel access in slotted wireless networks," *IEEE Access*, vol. 8, pp. 95 032–95 045, 2020.

[54] V. Vapnik, *Estimation of dependences based on empirical data*. Springer Science & Business Media, 2006.

[55] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, 1989.

[56] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[57] F. S. Melo and M. Lopes, "Convergence of independent adaptive learners," *Progress in Artificial Intelligence, Proceedings*, vol. 4874, pp. 555–567, 2007.

[58] R. M. Kretchmar, "Parallel reinforcement learning," in *The 6th World Conference on Systemics, Cybernetics, and Informatics*, 2002.

[59] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0893608089900208

[60] I. C. Systems, "Cisco visual networking index: Global mobile data traffic forecast update, 2017-2022 (white paper)," *Cisco Public Information*, pp. 1–35, 2017.

[61] Ericsson, "Mobile data traffic outlook," Ericsson, Ericsson Mobility report data and forecasts, June 2023.

[62] M. Bkassiny, Y. Li, and S. K. Jayaweera, "A survey on machine-learning techniques in cognitive radios," *IEEE Communications Surveys & Tutorials*, vol. 15, pp. 1136–1159, 2013. [Online]. Available: http://ieeexplore.ieee.org/document/6336689/

[63] E. Karami and O. A. Dobre, "Identification of sm-ofdm and al-ofdm signals based on their second-order cyclostationarity," *IEEE Transactions on Vehicular Technology*, vol. 64, pp. 942–953, 3 2015. [Online]. Available: http://ieeexplore.ieee.org/document/6819454/

[64] O. Dobre, "Signal identification for emerging intelligent radios: classical problems and new challenges," *IEEE Instrumentation & Measurement Magazine*, vol. 18, pp. 11–18, 4 2015. [Online]. Available: https://ieeexplore.ieee.org/document/7066677

[65] M. Shi, A. Laufer, Y. Bar-Ness, and W. Su, "Fourth order cumulants in distinguishing single carrier from ofdm signals," in *MILCOM 2008 - 2008 IEEE Military Communications Conference*. IEEE, 11 2008.

[66] A. Al-Habashna, O. A. Dobre, R. Venkatesan, and D. C. Popescu, "Second-order cyclostationarity of mobile wimax and lte ofdm signals and application to spectrum awareness in cognitive radio systems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 6, pp. 26–42, 2 2012. [Online]. Available: http://ieeexplore.ieee.org/document/6111235/

[67] M. Firdaoussi, H. Ghennioui, and M. E. Kamili, "Recognition of ofdm and scld signals based on the generalized mean ambiguity function," in *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*. IEEE, 10 2016, pp. 230–234. [Online]. Available: http://ieeexplore.ieee.org/document/7777219/

[68] A. Bouzegzi, P. Ciblat, and P. Jallon, "Maximum likelihood based methods for ofdm intercarrier spacing characterization," in *2008 IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications*. IEEE, 9 2008, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/document/4699444/

[69] M. Kulin, T. Kazaz, I. Moerman, and E. D. Poorter, "End-to-end learning from spectrum data: A deep learning approach for wireless signal identification in spectrum monitoring applications," *IEEE Access*, vol. 6, pp. 18 484–18 501, 2018. [Online]. Available: http://ieeexplore.ieee.org/document/8325299/

[70] N. Bitar, S. Muhammad, and H. H. Refai, "Wireless technology identification using deep convolutional neural networks," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE, 10 2017, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/8292183/

[71] S. Yi, H. Wang, W. Xue, X. Fan, L. Wang, J. Tian, and R. Matsukura, "Interference source identification for ieee 802.15.4 wireless sensor networks using deep learning," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 9 2018.

[72] J. Gui, T. Chen, J. Zhang, Q. Cao, Z. Sun, H. Luo, and D. Tao, "A survey on self-supervised learning: Algorithms, applications, and future trends," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–20, 2024.

[73] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2021.

[74] G. Huang, I. Laradji, D. Vázquez, S. Lacoste-Julien, and P. Rodríguez, "A survey of self-supervised and few-shot object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 4, pp. 4071–4089, 2023.

[75] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 5149–5169, 2022.

[76] H. Navidan, M. Girmay, M. Seif, H. V. Poor, I. Moerman, and A. Shahid, "Vehicular intelligence at the edge: A decentralized federated learning approach for technology recognition," in *2024 IEEE Vehicular Networking Conference (VNC)*, 2024, pp. 283–289.

[77] M. Girmay, V. Maglogiannis, D. Naudts, M. Aslam, A. Shahid, and I. Moerman, "Technology recognition and traffic characterization for wireless technologies in its band," *Vehicular Communications*, vol. 39, p. 100563, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214209622001103

[78] I. Remadna, L. S. Terrissa, Z. Al Masry, and N. Zerhouni, "Rul prediction using a fusion of attention-based convolutional variational autoencoder and ensemble learning classifier," *IEEE Transactions on Reliability*, vol. 72, no. 1, pp. 106–124, 2023.

[79] D. D. Chakladar, S. Datta, P. P. Roy, and V. A. Prasad, "Cognitive workload estimation using variational autoencoder and attention-based deep model," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 15, no. 2, pp. 581–590, 2023.

[80] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proceedings of the 19th International Conference on Neural Information Processing Systems*. MIT Press, 2006, pp. 153–160. [Online]. Available: http://dl.acm.org/citation.cfm?id=2976456.2976476

[81] G. Alain and Y. Bengio, "What regularized auto-encoders learn from the data-generating distribution," *J. Mach. Learn. Res.*, vol. 15, pp. 3563–3593, 1 2014. [Online]. Available: http://dl.acm.org/citation.cfm?id=2627435.2750359

[82] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time-series," in *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. MIT Press, 1995.

[83] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Convolutional radio modulation recognition networks," in *Engineering Applications of Neural Networks*, C. Jayne and L. Iliadis, Eds. Springer International Publishing, 2016, pp. 213–226.

[84] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

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

[86] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://keras.io

[87] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'16. USA: USENIX Association, 2016, p. 265–283.

[88] F. A. de Figueiredo, D. Stojadinovic, P. Maddala, R. Mennes, I. Jabandzic, X. Jiao, and I. Moerman, "Scatter phy: A physical layer for the darpa spectrum collaboration challenge," in *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN) (IEEE DySPAN 2019)*, 11 2019.

[89] O-RAN Alliance, "O-ran: Towards an open and smart ran," O-RAN Alliance, White Paper, 2018.

[90] A. Sinha and M. P. Wellman, "Incentivizing collaboration in a competition," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 556–564.

[91] T. Alpcan, H. Boche, M. L. Honig, and H. V. Poor, *Mechanisms and games for dynamic spectrum allocation*. Cambridge University Press, 2013.

[92] R. Begleiter, R. El-Yaniv, and G. Yona, "On prediction using variable order markov models," *J. Artif. Int. Res.*, vol. 22, pp. 385–421, 12 2004.

[93] G. Ding, Y. Jiao, J. Wang, Y. Zou, Q. Wu, Y. Yao, and L. Hanzo, "Spectrum inference in cognitive radio networks: Algorithms and applications," *IEEE Communications Surveys Tutorials*, vol. 20, pp. 150–182, 2018.

[94] D. Roy, T. Mukherjee, M. Chatterjee, and E. Pasiliao, "Primary user activity prediction in dsa networks using recurrent structures," in *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, 2019, pp. 1–10.

[95] DARPA. Phase 3 sc2 cil project. Visited on 10-August-2021. [Online]. Available: https://github.com/Furtad0/CIL

[96] D. Stojadinovic, F. A. P. de Figueiredo, P. Maddala, I. Seskar, and W. Trappe, "Sc2 cil: Evaluating the spectrum voxel announcement benefits," in *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, 2019, pp. 1–6.

[97] M. Hoyhtya, H. Sarvanko, M. Matinmikko, and A. Mammela, "Autocorrelation-based traffic pattern classification for cognitive radios," in *2011 IEEE Vehicular Technology Conference (VTC Fall)*, 2011, pp. 1–5.

[98] M. K. Ehsan, "Performance analysis of the probabilistic models of ism data traffic in cognitive radio enabled radio environments," *IEEE Access*, vol. 8, pp. 140–150, 2020.

[99] G. S. Uyanik and S. Oktug, "Primary user activity classification aided channel assignment in cognitive radio networks," in *2016 IEEE Symposium on Computers and Communication (ISCC)*, 2016, pp. 838–842.

[100] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Toward effective mobile encrypted traffic classification through deep learning," *Neurocomputing*, vol. 409, pp. 306 – 315, 2020.

[101] S. Valenti, D. Rossi, A. Dainotti, A. Pescapè, A. Finamore, and M. Mellia, *Reviewing Traffic Classification*. Springer Berlin Heidelberg, 2013, pp. 123–147.

[102] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, "Towards the deployment of machine learning solutions in network traffic classification: A systematic survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1988–2014, 2019.

[103] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, pp. 2224–2287, 2019.

[104] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, 2020.

[105] P. Wang, X. Chen, F. Ye, and Z. Sun, "A survey of techniques for mobile service encrypted traffic classification using deep learning," *IEEE Access*, vol. 7, pp. 54 024–54 033, 2019.

[106] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys & Tutorials*, vol. 10, pp. 56–76, 2008.

[107] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: An overview," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 76–81, 2019.

[108] Z. Chen, K. He, J. Li, and Y. Geng, "Seq2img: A sequence-to-image based approach towards ip traffic classification using convolutional neural networks," in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 1271–1276.

[109] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2017, pp. 43–48.

[110] S. Rezaei and X. Liu, “How to achieve high classification accuracy with just a few labels: A semi-supervised approach using sampled packets,” *arXiv preprint arXiv:1812.09761*, 2018.

[111] X. Wang, S. Chen, and J. Su, “Real network traffic collection and deep learning for mobile app identification,” *Wireless Communications and Mobile Computing*, vol. 2020, 2020.

[112] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, “Network traffic classifier with convolutional and recurrent neural networks for internet of things,” *IEEE Access*, vol. 5, pp. 18 042–18 050, 2017.

[113] D. Li, Y. Zhu, and W. Lin, “Traffic identification of mobile apps based on variational autoencoder network,” in *2017 13th International Conference on Computational Intelligence and Security (CIS)*. IEEE, 2017, pp. 287–291.

[114] A. Rago, G. Piro, G. Boggia, and P. Dini, “Multi-task learning at the mobile edge: An effective way to combine traffic classification and prediction,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 10 362–10 374, 2020.

[115] M. Miozzo, Z. Ali, L. Giupponi, and P. Dini, “Distributed and multi-task learning at the edge for energy efficient radio access networks,” *IEEE Access*, vol. 9, pp. 12 491–12 505, 2021.

[116] H. D. Trinh, A. Fernandez Gambin, L. Giupponi, M. Rossi, and P. Dini, “Mobile traffic classification through physical control channel fingerprinting: A deep learning approach,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1946–1961, 2021.

[117] M. Camelo, T. De Schepper, P. Soto, J. Marquez-Barja, J. Famaey, and S. Latré, “Detection of traffic patterns in the radio spectrum for cognitive wireless network management,” in *2020 IEEE International Conference on Communications (ICC)*, 2020.

[118] T. De Schepper, M. Camelo, J. Famaey, and S. Latré, “Traffic classification at the radio spectrum level using deep learning models trained with synthetic data,” *International Journal of Network Management*, vol. n/a, no. n/a, p. e2100, 2020.

[119] T. J. O’Shea, S. Hitefield, and J. Corgan, “End-to-end radio traffic sequence recognition with recurrent neural networks,” in *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2016, pp. 277–281.

[120] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[121] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: continual prediction with lstm,” in *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, vol. 2, 1999, pp. 850–855 vol.2.

[122] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks,” in *International conference on machine learning*, 2013, pp. 1310–1318.

[123] Z. C. Lipton, J. Berkowitz, and C. Elkan, “A critical review of recurrent neural networks for sequence learning,” *arXiv preprint arXiv:1506.00019*, 2015.

[124] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[125] S. K. Sharma, T. E. Bogale, S. Chatzinotas, B. Ottersten, L. B. Le, and X. Wang, "Cognitive radio techniques under practical imperfections: A survey," *IEEE communications surveys and tutorials*, 2015.

[126] V. Maglogiannis, A. Shahid, D. Naudts, E. De Poorter, and I. Moerman, "Enhancing the coexistence of lte and wi-fi in unlicensed spectrum through convolutional neural networks," *IEEE Access*, vol. 7, pp. 28 464–28 477, 2019.

[127] A. Shahid, J. Fontaine, M. Camelo, J. Haxhibeqiri, M. Saelens, Z. Khan, I. Moerman, and E. Poorter, "A convolutional neural network approach for classification of lpwan technologies: Sigfox, lora and ieee 802.15.4g," in *Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks workshops*, vol. 2019-June, 2019.

[128] P. K. Taksande, P. Jha, A. Karandikar, and P. Chaporkar, "Open5g: A software-defined networking protocol for 5g multi-rat wireless networks," 2020.

[129] M. Schulz, D. Wegemer, and M. Hollick, "Nexmon: Build your own wi-fi testbeds with low-level mac and phy-access using firmware patches on off-the-shelf mobile devices," in *Proceedings of the 11th Workshop on Wireless Network Testbeds, Experimental Evaluation & CHaracterization*, ser. WiNTECH '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 59–66. [Online]. Available: https://doi.org/10.1145/3131473.3131476

[130] M. Camelo, A. Shahid, J. Fontaine, F. A. P. de Figueiredo, E. De Poorter, I. Moerman, and S. Latre, "A semi-supervised learning approach towards automatic wireless technology recognition," in *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, 2019, pp. 1–10.

[131] M. Boucadair, O. Bonaventure, M. Piraux, Q. D. Coninck, S. Dawkins, M. Kühlewind, M. Amend, A. Kassler, Q. An, N. Keukeleire, and S. Seo, "3GPP Access Traffic Steering Switching and Splitting (ATSSS) - Overview for IETF Participants," Internet Engineering Task Force, Internet-Draft draft-bonaventure-quic-atsss-overview-00, 2020-05-30, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-bonaventure-quic-atsss-overview-00

[132] F. Shaheen, B. Verma, and M. Asafuddoula, "Impact of automatic feature extraction in deep learning architecture," in *2016 International conference on digital image computing: techniques and applications (DICTA)*. IEEE, 2016, pp. 1–8.

[133] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, and Yiqiang Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *2017 International Conference on Information Networking (ICOIN)*, 2017, pp. 712–717.

[134] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[135] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[136] M. Kim, Z. Zhang, D. Kim, and S. Choi, "Deep-learning-based frame format detection for ieee 802.11 wireless local area networks," *Electronics*, vol. 9, no. 7, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/7/1170

[137] V. Ninkovic, D. Vukobratovic, A. Valka, and D. Dumic, "Preamble-based packet detection in wi-fi: A deep learning approach," 2020.

[138] D. Magrin, C. Pielli, C. Stefanovic, and M. Zorzi, "Enabling lte rach collision multiplicity detection via machine learning," in *2019 International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT)*, 2019, pp. 1–8.

[139] M. Dwarampudi and N. Reddy, "Effects of padding on lstms and cnns," *arXiv preprint arXiv:1903.07288*, 2019.

[140] ITU-T, "Architectural framework for machine learning in future networks including imt-2020," ITU-T, Recommendation ITU-T Y.3172, Jun, 2019. [Online]. Available: http://handle.itu.int/11.1002/1000/13894

[141] F. Wilhelmi, S. Barrachina-Munoz, B. Bellalta, C. Cano, A. Jonsson, and V. Ram, "A flexible machine-learning-aware architecture for future wlans," *IEEE Communications Magazine*, vol. 58, no. 3, pp. 25–31, 2020.

[142] V. Erceg, L. Schumacher, P. Kyritsi, A. Molisch, and D. S. Baum, "Tgn channel models," IEEE, Tech Report Version 4, May 2004.

[143] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[144] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[145] H. X. Pham, H. M. La, D. Feil-Seifer, and L. V. Nguyen, "Autonomous UAV navigation using reinforcement learning," *arXiv preprint arXiv:1801.05086*, 2018. [Online]. Available: http://arxiv.org/abs/1801.05086

[146] D. N. Ray, A. Mandal, S. Majumder, and S. Mukhopadhyay, "Human-like gradual multi-agent q-learning using the concept of behavior-based robotics for autonomous exploration," in *2011 IEEE International Conference on Robotics and Biomimetics*, 2011, pp. 2725–2732.

[147] P. Mannion, J. Duggan, and E. Howley, "Parallel reinforcement learning for traffic signal control," in *Procedia Computer Science*, vol. 52, no. 1, 2015, pp. 956–961.

[148] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys & Tutorials*, 2019.

[149] F. Restuccia, T. Melodia, and J. Ashdown, *Spectrum Challenges in the Internet of Things: State of the Art and Next Steps*. John Wiley & Sons, Ltd, 2022, ch. 19, pp. 353–375. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119892199.ch19

[150] I. T. U. ITU. Spectrum management: Key applications and regulatory considerations driving the future use of spectrum. (Visited on 18-July-2023). [Online]. Available: https://digitalregulation.org/spectrum-management-key-applications-and-regulatory-considerations-driving-the-future-use-of

[151] ——. Iot and imt spectrum issues. (Visited on 18-July-2023).

[152] A. W. Schapaugh and A. J. Tyre, "A simple method for dealing with large state spaces," *Methods in Ecology and Evolution*, vol. 3, no. 6, pp. 949–957, 2012.

[153] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Transactions on Automatic Control*, vol. 42, pp. 674–690, 1997.

[154] M. Camelo, J. Famaey, and S. Latré, "A scalable parallel Q-learning algorithm for resource constrained decentralized computing environments," in *Proceedings of MLHPC 2016: Machine Learning in HPC Environments*, 2016, pp. 27–35.

[155] E. H. Patrick Mannion, Jim Duggan, "Parallel Reinforcement Learning with State Action Space Partitioning," in *The 12th European Workshop on Reinforcement Learning*, 2015.

[156] M. Kushida, K. Takahashi, H. Ueda, and T. Miyahara, "A comparative study of parallel reinforcement learning methods with a PC cluster system," in *IEEE/WIC/ACM Conf Intelligent Agent Technology*, 2006, pp. 416–419.

[157] D. Wang, W. Zhang, B. Song, X. Du, and M. Guizani, "Market-based model in cr-iot: A q-probabilistic multi-agent reinforcement learning approach," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 1, pp. 179–188, 2020.

[158] H. Li, "Multi-agent q-learning for competitive spectrum access in cognitive radio systems," in *2010 Fifth IEEE Workshop on Networking Technologies for Software Defined Radio Networks (SDR)*, 2010, pp. 1–6.

[159] B. Xia, M. H. Wahab, Y. Yang, Z. Fan, and M. Sooriyabandara, "Reinforcement learning based spectrum-aware routing in multi-hop cognitive radio networks," in *2009 4th International Conference on Cognitive Radio Oriented Wireless Networks and Communications*, 2009, pp. 1–5.

[160] C. Tarver, M. Tonnemacher, V. Chandrasekhar, H. Chen, B. L. Ng, J. Zhang, J. R. Cavallaro, and J. Camp, "Enabling a "use-or-share" framework for pal–gaa sharing in cbrs networks via reinforcement learning," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 3, pp. 716–729, 2019.

[161] X. Tan, L. Zhou, H. Wang, Y. Sun, H. Zhao, B.-C. Seet, J. Wei, and V. C. M. Leung, "Cooperative multi-agent reinforcement-learning-based distributed dynamic spectrum access in cognitive radio networks," *IEEE Internet of Things Journal*, vol. 9, no. 19, pp. 19 477–19 488, 2022.

[162] M. Mohammadi, A. Al-Fuqaha, M. Guizani, and J.-S. Oh, "Semisupervised deep reinforcement learning in support of iot and smart city services," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 624–635, 2017.

[163] M. Min, X. Wan, L. Xiao, Y. Chen, M. Xia, D. Wu, and H. Dai, "Learning-based privacy-aware offloading for healthcare iot with energy harvesting," *IEEE Internet of Things Journal*, 2019.

[164] M. A. Salahuddin, A. Al-Fuqaha, and M. Guizani, "Software-defined networking for rsu clouds in support of the internet of vehicles," *IEEE Internet of Things Journal*, vol. 2, no. 2, pp. 133–144, 2015.

[165] C. Wang, L. Zhang, Z. Li, and C. Jiang, "Sdcor: Software defined cognitive routing for internet of vehicles," *IEEE Internet of Things Journal*, 2018.

[166] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): Methodology and large-scale application on downtown toronto," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1140–1150, 2013.

[167] M. I. Khan, M. M. Alam, Y. L. Moullec, and E. Yaacoub, "Cooperative reinforcement learning for adaptive power allocation in device-to-device communication," in *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, 2018, pp. 476–481.

[168] J. Zhu, Y. Song, D. Jiang, and H. Song, "A new deep-q-learning-based transmission scheduling mechanism for the cognitive internet of things," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2375–2385, 2017.

[169] G. M. Dias, M. Nurchis, and B. Bellalta, "Adapting sampling interval of sensor networks using on-line reinforcement learning," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, 2016, pp. 460–465.

[170] Y. Debizet, G. Lallement, F. Abouzeid, P. Roche, and J. Autran, "Q-learning-based adaptive power management for iot system-on-chips with embedded power states," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018, pp. 1–5.

[171] A. Printista, M. Errecalde, and C. Montoya, "A parallel implementation of Q-learning based on communication with cache," *Journal of Computer Science & Technology*, 2002.

[172] Q. Liu, X. Yang, L. Jing, J. Li, and J. Li, "A parallel scheduling algorithm for reinforcement learning in large state space," *Frontiers of Computer Science*, vol. 6, pp. 631–646, 2012. [Online]. Available: http://link.springer.com/10.1007/s11704-012-1098-y

[173] R. Bokade, X. Jin, and C. Amato, "Multi-agent reinforcement learning based on representational communication for large-scale traffic signal control," *IEEE Access*, vol. 11, pp. 47 646–47 658, 2023.

[174] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: a survey," *Artif. Intell. Rev.*, vol. 55, no. 2, p. 895–943, feb 2022. [Online]. Available: https://doi.org/10.1007/s10462-021-09996-w

[175] C. Zhu, M. Dastani, and S. Wang, "A survey of multi-agent deep reinforcement learning with communication," in *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '24. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2024, p. 2845–2847.

[176] C. Liu and D. Liu, "Deep reinforcement learning algorithm based on multi-agent parallelism and its application in game environment," *Entertainment Computing*, vol. 50, p. 100670, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1875952124000387

[177] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses," *AI Magazine*, vol. 29, no. 1, p. 9, 2008.

[178] M. Tan, "Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents," in *Machine Learning Proceedings 1993*, 1993, pp. 330–337.

[179] F. Sadeghi and S. Levine, "Cad2rl: Real single-image flight without a single real image," *arXiv preprint arXiv:1611.04201*, 2016.

[180] C. Cai, C. Yang, Q. Zhu, and Y. Liang, "Collision avoidance in multi-robot systems," in *2007 International Conference on Mechatronics and Automation*, Aug 2007, pp. 2795–2800.

[181] J. MacGlashan. Burlap: Brown-umbc reinforcement learning and planning. [Online]. Available: http://burlap.cs.brown.edu/

[182] M. Zheludkov, T. Isachenko *et al.*, *High Performance in-memory computing with Apache Ignite*. Lulu. com, 2017.

[183] ArcBotics. The sparki robot. [Online]. Available: http://arcbotics.com/products/sparki/

[184] imec. Teaching intelligent robots the art of delegation. [Online]. Available: https://vimeo.com/340200165

[185] M. O. Demir, G. K. Kurt, and M. Karaca, "An energy consumption model for 802.11ac access points," in *2014 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2014, pp. 67–71.

[186] M. Hoyhtya, A. Mammela, A. Chiumento, S. Pollin, M. Forsell, and D. Cabric, "Database-assisted spectrum prediction in 5g networks and beyond: A review and future challenges," *IEEE Circuits and Systems Magazine*, vol. 19, pp. 34–45, 2019.

[187] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Artificial neural networks-based machine learning for wireless networks: A tutorial," *IEEE Communications Surveys Tutorials*, vol. 21, pp. 3039–3071, 2019.

[188] S. Riyaz, K. Sankhe, S. Ioannidis, and K. Chowdhury, "Deep learning convolutional neural networks for radio identification," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 146–152, 2018.

[189] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 445–458, 2019.

[190] Y. Zeng, M. Zhang, F. Han, Y. Gong, and J. Zhang, "Spectrum analysis and convolutional neural network for automatic modulation recognition," *IEEE Wireless Communications Letters*, vol. 8, no. 3, pp. 929–932, 2019.

[191] J. Yang and F. Liu, "Modulation recognition using wavelet transform based on alexnet," in *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*, 2019, pp. 339–342.

[192] A. Banchs, M. Fiore, A. Garcia-Saavedra, and M. Gramaglia, "Network intelligence in 6g: Challenges and opportunities," in *Proceedings of the 16th ACM Workshop on Mobility in the Evolving Internet Architecture*, ser. MobiArch '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 7–12. [Online]. Available: https://doi.org/10.1145/3477091.3482761

[193] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," 2021. [Online]. Available: https://arxiv.org/abs/2103.13630

[194] A. Renda, J. Frankle, and M. Carbin, "Comparing rewinding and fine-tuning in neural network pruning," 2020. [Online]. Available: https://arxiv.org/abs/2003.02389

[195] M. R. Samsami and H. Alimadad, "Distributed deep reinforcement learning: An overview," 2020. [Online]. Available: https://arxiv.org/abs/2011.11012

[196] J. Keuper and F.-J. Preundt, "Distributed training of deep neural networks: Theoretical and practical limits of parallel scalability," in *2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC)*, 2016, pp. 19–26.