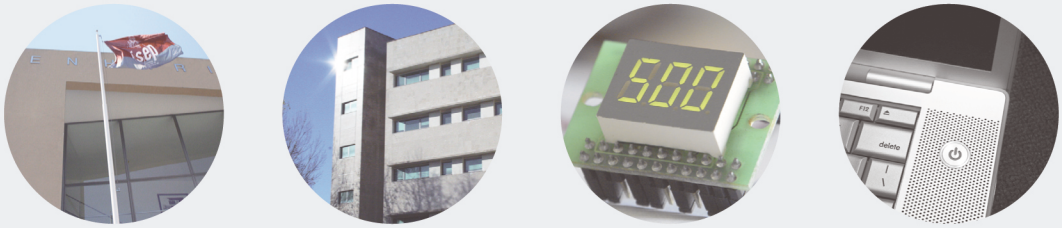
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES



Rumo à Autenticação de Dispositivos IoMT através da Classificação de Sinais RF

STÉPHANE JOAQUIM LOURENÇO MONTEIRO
julho de 2023

POLITÉCNICO
DO PORTO

Politécnico do Porto
Instituto Superior de Engenharia do Porto

Towards Authentication of IoMT Devices via RF Signal Classification

Stéphane Joaquim Lourenço Monteiro

Master in Electrical and Computer Engineering
Specialization Area of Automation and Systems

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

July, 2023

This dissertation partially satisfies the requirements of the Thesis/Dissertation course of the program Master in Electrical and Computer Engineering, Specialization Area of Automation and Systems.

Candidate: Stéphane Joaquim Lourenço Monteiro, No. 1180697, 1180697@isep.ipp.pt

Scientific Guidance: Dr. Ricardo Severino, rar@isep.ipp.pt

Advisor: Emmanuel António Carvalhido Lomba, emmanuel@airlomba.net

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431, 4200–072 Porto

July, 2023

Acknowledgements

To begin, I would like to thank Dr. Ricardo Severino, a thank you will never be enough for all of the guidance he provided in the various research projects, projects that provided me with a wealth of knowledge while also allowing me to develop a professional career, he is a person I greatly admire both professionally and personally.

I want to express my sincere appreciation to Engineer Emmanuel Lomba for all of his help with this project. His assistance was crucial in assisting us to overcome many difficulties and barriers.

I would also like to thank the Instituto Superior de Engenharia do Porto (ISEP) and all of the teachers that taught me all of the essential tools that I use on a daily basis.

Thank you so much to all of my friends I've made in Porto over the last five years; thank you for making this city my home, for making me laugh, and for assisting me in the most difficult times.

To all my ISEP friends, thank you for your company and assistance; I hope I did the same for you when you needed it. Bruno Cunha, Carlos Gonçalves, Francisca Almeida, Tomás Sousa, João Alves, Bruno Mendes, Diogo Lopes, and Hugo Silva deserve special gratitude.

To my childhood friends, who have been with me since I arrived in Portugal, André Guedes, José Soares, Rafael Rodrigues, António Amadeu, Luís Pimenta and José Loureiro, thank you for everything you have done for me.

To my coworker Filipe Gomes, thank you for these three years, for your support, and for being someone I consider a brother.

Finally, I want to thank everyone in my family, especially my parents, sister, brother, and brother-in-law. Thank you for being a source of inspiration and energy for me, for being people who help me, for being my idols, and for all of your teachings.

Abstract

The increasing reliance on the Internet of Medical Things (IoMT) raises great concern in terms of cybersecurity, either at the device's physical level or at the communication and transmission level. This is particularly important as these systems process very sensitive and private data, including personal health data from multiple patients such as real-time body measurements. Due to these concerns, cybersecurity mechanisms and strategies must be in place to protect these medical systems, defending them from compromising cyberattacks. Authentication is an essential cybersecurity technique for trustworthy IoMT communications. However, current authentication methods rely on upper-layer identity verification or key-based cryptography which can be inadequate to the heterogeneous Internet of Things (IoT) environments.

This thesis proposes the development of a Machine Learning (ML) method that serves as a foundation for Radio Frequency Fingerprinting (RFF) in the authentication of IoMT devices in medical applications to improve the flexibility of such mechanisms. This technique allows the authentication of medical devices by their physical layer characteristics, i.e. of their emitted signal. The development of ML models serves as the foundation for RFF, allowing it to evaluate and categorise the released signal and enable RFF authentication. Multiple feature take part of the proposed decision making process of classifying the device, which then is implemented in a medical gateway, resulting in a novel IoMT technology.

Keywords: RFF, ML, CNN, IoMT.

Resumo

A confiança crescente na IoMT suscita grande preocupação em termos de ciber-segurança, quer ao nível físico do dispositivo quer ao nível da comunicação e ao nível de transmissão. Isto é particularmente importante, uma vez que estes sistemas processam dados muito sensíveis e dados, incluindo dados pessoais de saúde de diversos pacientes, tais como dados em tempo real de medidas do corpo. Devido a estas preocupações, os mecanismos e estratégias de ciber-segurança devem estar em vigor para proteger estes sistemas médicos, defendendo-os de ciberataques comprometedores. A autenticação é uma técnica essencial de ciber-segurança para garantir as comunicações em sistemas IoMT de confiança. No entanto, os métodos de autenticação atuais focam-se na verificação de identidade na camada superior ou criptografia baseada em chaves que podem ser inadequadas para a ambientes IoMT heterogéneos.

Esta tese propõe o desenvolvimento de um método de ML que serve como base para o RFF na autenticação de dispositivos IoMT para melhorar a flexibilidade de tais mecanismos. Isto permite a autenticação dos dispositivos médicos pelas suas características de camada física, ou seja, a partir do seu sinal emitido. O desenvolvimento de modelos de ML serve de base para o RFF, permitindo-lhe avaliar e categorizar o sinal libertado e permitir a autenticação do RFF. Múltiplas features fazem parte do processo de tomada de decisão proposto para classificar o dispositivo, que é implementada num gateway médico, resultando numa nova tecnologia IoMT.

Palavras-Chave: RFF, ML, CNN, IoMT.

Contents

List of Figures **ix**

List of Tables **xi**

List of Acronyms **xiii**

1 Introduction **1**
1.1 Overview 1
1.2 Objectives 5
1.3 Research Context 6
1.3.1 ForPharmacy Project 6
1.3.2 iCare4NextG Project 6
1.4 Thesis Structure 7

2 Research Background **9**
2.1 Overview of Healthcare Communication Security 9
2.1.1 IoMT Requirements 9
2.1.2 IoMT Architecture Layers 11
2.1.3 Security Breaches in IoMT 12
2.1.4 Security Solutions 14
Communication Layer Security Solutions 14
Network Routing Security Solutions 15
Transport and Application Layer Security Solutions 16
2.1.5 Towards New Methods of Authentication 17
2.2 Radio Frequency Fingerprint 18
2.2.1 Signal Processing 19
2.2.2 Feature Extraction 20
2.2.3 Machine Learning Approaches for Identification and Classification 21
Differential Contour Stellar-Based RFF 21
Imaging Time Series for Internet of Things Radio Frequency Fingerprinting 22
RF Fingerprint Reconition Based On Spectrum Waterfall Diagram 23

2.3 Artificial Intelligence . . . 23
Unsupervised Learning . . . 24
Supervised Learning . . . 24
Reinforcement Learning . . . 25
2.3.1 Artificial Neural Network . . . 25
2.3.2 Deep Learning . . . 26
2.3.3 Deep Neural Networks . . . 27
Convolutional Neural Network . . . 27
Convolutional Layers . . . 28
Pooling . . . 30
Clustering Layers . . . 30
Activation Function . . . 31
Batch Normalization . . . 32
Dropout . . . 32
Fully connected layer . . . 33
Data augmentation . . . 33
2.3.4 Transfer Learning . . . 34

3 Software and Technologies 35
3.1 Frameworks Used in Machine Learning . . . 35
3.1.1 PyTorch . . . 36
3.1.2 Anaconda . . . 37
3.1.3 Keras . . . 37
3.1.4 Tensorflow . . . 37
3.1.5 PyTorch And Tensorflow Comparasion . . . 38
3.2 Popular CNN image classification models . . . 40
3.2.1 LeNet . . . 40
3.2.2 AlexNet . . . 40
3.2.3 VGG . . . 41
3.3 IoMT Gateway . . . 42
3.3.1 RFF Communication Protocol . . . 46
3.3.2 Signal Acquisition and Feature Extraction . . . 46
3.3.3 Devices . . . 47

4 RFF Machine Learning Framework 49
4.1 IoMT Security RFF Gateway Architecture . . . 49
4.2 Setup . . . 52
4.2.1 Computer Setup . . . 52
4.2.2 Virtual Environment . . . 53
4.3 Dataset . . . 54
4.4 Features . . . 54

Constellation 54
Amplitude 55
Power Spectral Density 56
Differential Constellation 57
Spectrograms 58
4.4.1 Data augmentation 59
4.4.2 Data processing 60
4.4.3 Division of data 61
4.4.4 Model 62
4.4.5 Training 63
Hyperopt 63
Epochs 64
Model Compilation and Training 64
4.4.6 Saving Model 65

5 ML Model Assessment and Classification Results 67
5.1 Class Labels 67
5.2 Evaluation Metrics 68
5.3 Assessment of the Experiences 70
5.3.1 Models performance with the custom block for ER400TRS devices 70
Constellation 70
Power Spectrum Density 71
Spectrogram 72
Amplitude 73
Differential Constellation 74
Performance Improvement in ER400TRS Devices 75
5.3.2 Models performance with the custom block for WiFi devices 76
Constellation 76
Power Spectrum Density 77
Spectrogram 78
Amplitude 79
Differential Constellation 80
Performance Improvement in WiFi Devices 81
5.3.3 Balancing Feature Complexity with Accuracy in Models . . . 82
5.4 Decision Making 83
5.4.1 Classification 83
5.4.2 Execution Time Evaluation 85
5.4.3 Final Classification With Weighted Majority 87

6 Conclusions **89**
6.1 Discussion 89
6.2 Future Work 90

References **91**

List of Figures

1.1 Internet of Things (IoT) Devices Statistics by 2025 [3] 2
1.2 Device Level IoT Security Vulnerabilities [5] 3
1.3 Number of active connections in medical IoT in the European Union [8] 4

2.1 IoMT Architecture [13]. 11
2.2 Typical RFF identification process. 18
2.3 IQ complex number representation. 19
2.4 Feature Extraction. 20
2.5 Creation of differential Contour Stellar Image [26]. 22
2.6 Scheme for Artificial Intelligence Concepts. 23
2.7 Basic Artificial Neural Network Architecture [36]. 26
2.8 Machine Learning (ML) and Deep Learning (DL) comparison [39]. . 27
2.9 Architecture Convolutional Neural Network (CNN) [42]. 28
2.10 Pixel matrix [45]. 29
2.11 Convolution of the image with a filter [46]. 29
2.12 CNN grouping [49]. 30
2.13 Activation functions [50]. 31
2.14 Dropout [53]. 33

3.1 Most popular programming language for Artificial Intelligence (AI) [57]. 36
3.2 Tensorboard [68]. 39
3.3 LeNet-5 architecture [69]. 40
3.4 AlexNet architecture [70]. 41
3.5 The Internet of Medical Things (IoMT) landscape [72]. 42
3.6 Medical Gateway. 43
3.7 Edge Gateway. 45
3.8 ESP32. 47
3.9 ER400TRS. 48

4.1 Block diagram of the proposed architecture. 50
4.2 RFF Integration in Medical Gateway. 51
4.3 Constellation of an ESP32 device. 55

4.4 Amplitude of an ESP32 device. 56
4.5 Power Spectral Density (PSD) of an ESP32 device. 57
4.6 Differential Constellation of an ESP32 device. 58
4.7 Spectrogram of an ESP32 device. 59
4.8 Diagram of model used. 62

5.1 Confusion Matrix [81]. 68
5.2 Loss and confusion matrix for constellation model H. 71
5.3 Loss and confusion matrix for PSD model E. 72
5.4 Loss and confusion matrix for spectogram model H. 73
5.5 Loss and confusion matrix for amplitude model G. 74
5.6 Loss and confusion matrix for differential constellation model F. . . 75
5.7 Accuracy Gain in each feature. 75
5.8 Decrease of loss in each feature. 76
5.9 Loss and confusion matrix for constellation model B. 77
5.10 Loss and confusion matrix for PSD model C. 78
5.11 Loss and confusion matrix for spectrogram model B. 79
5.12 Loss and confusion matrix for amplitude model A. 80
5.13 Loss and confusion matrix for differential constellation model H. . . 81
5.14 Accuracy Gain in each feature. 81
5.15 Decrease of loss in each feature. 82
5.16 Execution time in ER400TRS decision making. 86
5.17 Execution time in WiFi decision making. 86
5.18 Final classification with weighted majority for ER400TRS devices. . 87
5.19 Final classification with weighted majority for WiFi devices. 87

List of Tables

2.1 IoMT potential attacks at Perception layer and their influence on the system's security requirements [12] 12
2.2 IoMT potential attacks at Network layer and their influence on the system's security requirements [12] 13
2.3 IoMT potential attacks at Application layer and their influence on the system's security requirements [12] 14
2.4 The last layer's activation functions prioritize the neural network's ultimate goal. 32

3.1 Comparison of Deep Learning Frameworks [59, 60] 36

4.1 Physical Machine Comparison 52
4.2 Experiences model 63

5.1 Constellation with custom block 70
5.2 PSD with custom block 71
5.3 Spectrogram with custom block 72
5.4 Amplitude with custom block 73
5.5 Differential Constellation with custom block 74
5.6 Constellation with custom block 76
5.7 PSD with custom block 77
5.8 Spectogram with custom block 78
5.9 Amplitude with custom block 79
5.10 Differential constellation with custom block 80

List of Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Network
BRISK	Binary Robust Invariant Scalable Keypoints
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CTF	Constellation Trace Figure
cuDNN	CUDA Deep Neural Network
DL	Deep Learning
DNN	Deep Neural Network
DoS	Denial of Service
ESP	Encapsulation Security Payload
FHIR	Fast Healthcare Interoperability Resources
FN	False Negative
FP	False Positive
FPGA	Field-programmable gate array
GPU	Graphics Processing Unit
HL7	Health Level Seven International
HOG	Histogram of Oriented Gradients
IDS	Intrusion Detection System
IMD	Implantable Medical Devices
IoMT	Internet of Medical Things
IoT	Internet of Things

IQ	In-phase/Quadrature
LBP	Local Binary Pattern
ML	Machine Learning
OvO	One-vs-One
OvR	One-vs-Rest
PSD	Power Spectral Density
QPSK	Quadrature phase-shift keying
ReLU	Rectified Linear Unit
RF	Radio Frequency
RFF	Radio Frequency Fingerprinting
RFID	Radio Frequency Identification
SDK	Software Development Kit
SDR	Software-Defined Radio
SURF	Speeded-Up Robust Features
SVM	Support Vector Machine
TN	True Negative
TP	True Positive

Chapter 1

Introduction

1.1 Overview

The IoT is a network of semi-autonomous devices with computational, networking, sensory, and actuation capabilities that connect to the real environment. It combines a wide range of technologies, including as wireless sensor networks, radio-frequency identification, and machine-to-machine communications. The IoT is becoming a pervasive reality, fueling applications in industries such as healthcare [1].

As the IoT expands, context-awareness will be critical in linking the real world and digital computer entities. Environmental sensing, network connection, and data processing approaches are all part of this. These developments pave the way for a wide range of complex IoT applications, such as intelligent healthcare systems, efficient transportation systems, smart energy infrastructures, and intelligent buildings. IoT network topology is harmonised, combining intelligent IoT-based application services with fundamental IoT sensor networks. Wireless networking technologies and the incorporation of developing technologies such as cloud platforms are driving global expansion in the IoT industry. The demand for linked IoT devices and application services is increasing as a result of this trend [2].

While the number of deployed IoT devices continues to grow annually and is expected to reach 75 billion by 2025 [3], (Figure 1.1), the number of interconnected devices per network will also grow exponentially. With an expanding number of IoT devices comes an increased risk of exploitation, particularly in industrial and medical settings.

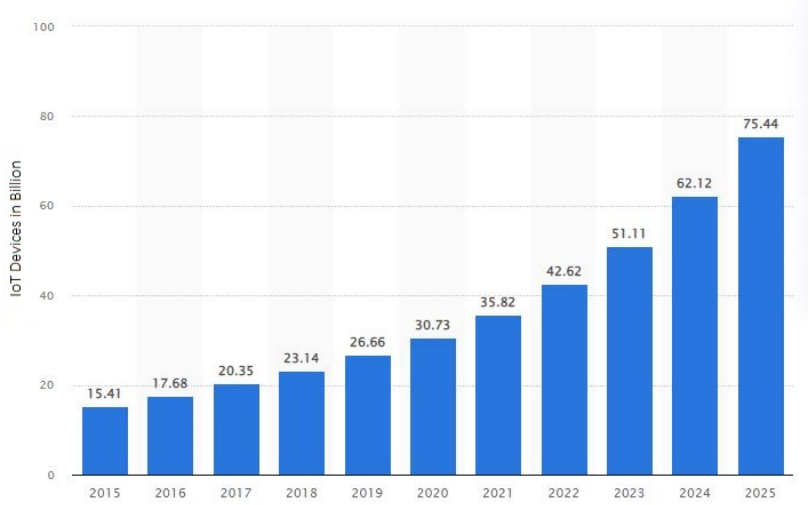


Figure 1.1: IoT Devices Statistics by 2025 [3]

The casual use of devices, the failure to update passwords, and the failure to complete regular device upgrades have increased cybersecurity threats, allowing hostile programmes access to sensitive data within IoT networks. Data breaches and other possible risks are made more likely by such inadequate security practises. Given the lack of security standards and policies in place, the majority of security experts see IoT as a vulnerable target for cyber attacks. Security rules are weakly documented, despite the development of several security techniques to protect IoT devices from cyber threats. As a result, end-users are frequently unable to successfully apply defensive measures to thwart data intrusions [4].

Hackers have been developing various sorts of malware to infiltrate IoT devices since 2008. They've invented a number of phishing techniques to trick employees or individuals into disclosing critical information. As a result of these high-profile hacks, business workstations and personal devices routinely experience privacy breaches. If device manufacturers and security experts appropriately identify these cyber hazards, they will be able to design and implement effective defensive mechanisms to either prevent or mitigate these cyber threats [4].

Unfortunately, most internet-connected IoT devices lack the same level of resistance against infiltration, hacking, and sabotage assaults that other computing devices have evolved over time. They, on the other hand, demonstrate a high level of susceptibility. Surprisingly, almost 70% of IoT devices have major security flaws, such as unsecured network services and low password restrictions, (Figure 1.2 [5]).

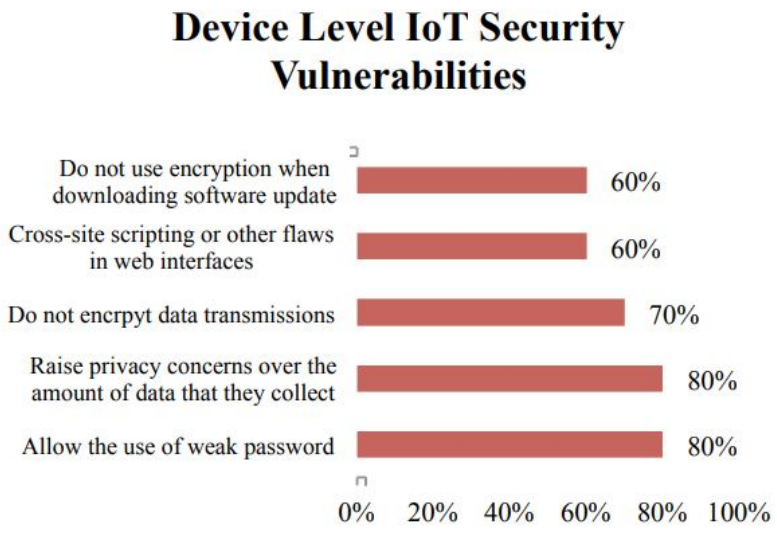


Figure 1.2: Device Level IoT Security Vulnerabilities [5]

The IoMT is a network of internet-connected devices aimed to improve healthcare services. IoMT obtains vital physiological data and monitors patients' pathological details using small wearable or implantable sensors. It has a wide range of uses, from implantable medical devices to wireless body area networks [6].

Furthermore, IoMT has an impact on enhanced drug and equipment control, telemedicine, mobile healthcare, and health data management. It offers real-time monitoring, assuring authenticity, and assisting in medical waste management by utilising technology such as Radio Frequency Identification (RFID) tags. It has improved hospital information administration and plays an important role in health management. IoMT is revolutionising healthcare services by providing remote patient monitoring via implantable sensory nodes [7].

Figure 1.3 depicts the increase in the number of active IoT connections in the healthcare sector in the European Union for 2016, 2019, 2022, and 2025. The graph shows a steady increase over time. It was predicted that from a base of 0.87 million connections in 2016, there will be 10.34 million connections by 2025 [8].

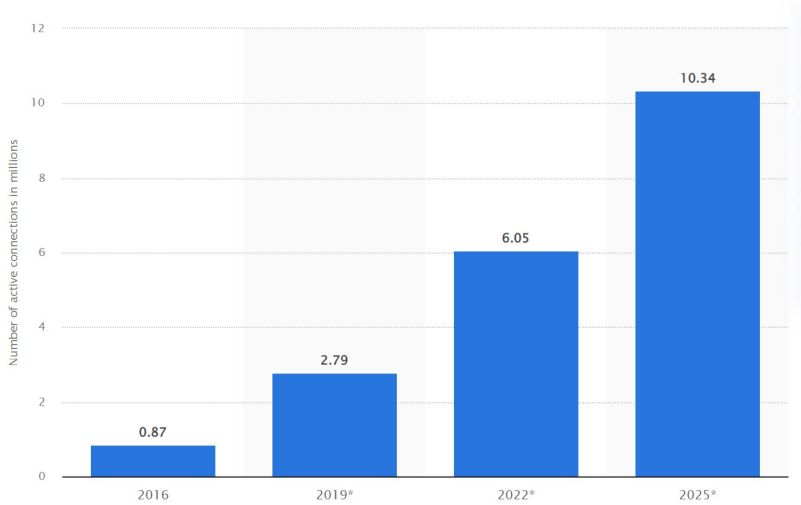


Figure 1.3: Number of active connections in medical IoT in the European Union [8]

With improvements in 5G communications and IPv6 addressing, these IoMT devices have begun to play a key role in extending human lives. The transmission of sensitive sensor data between medical IoT devices raises a number of security concerns. These IoMT devices are vulnerable to a number of security risks, including eavesdropping, hijacking, denial of service, message manipulation, device cloning, parameter configuration alteration, and power denial attacks. Typically IoMT presents restrictions such as limited battery capacity, limited memory, and poor computing power. These restrictions have the greatest impact on availability and security. As a result, implementing a safe and secure IoT in the healthcare business is crucial [9].

These cyberattacks on IoMT equipment have a wide range of serious consequences, from data breaches to life-threatening disinformation. Denial of service assaults overwhelm the system with excessive traffic, resulting in data inaccessibility and tampering with patient information, which can result in erroneous treatments and even patient mortality. Router attacks jeopardise the secure transmission of vital healthcare information. Select Forwarding attacks entail the selective discarding of data packets, which results in incomplete data at the receiver's end and the possibility of misdiagnosis. Sensor assaults can entail attackers replacing sensors with malicious ones, resulting in data manipulation and fake data insertion. In replay assaults, an attacker impersonates legitimate data transmissions, resulting in false recognitions or duplications. Each of these assaults can result in unauthorised access, data manipulation, denial of continuous monitoring, data route changes, and data drops [10].

Hence, the importance of strong authentication techniques cannot be overstated in the field of IoMT, where sensitive data is transferred on an unprecedented scale.

Adoption of Radio Frequency Fingerprinting (RFF), which capitalises on the unique, intrinsic properties of each transmitter during transmission, is a viable route.

These specific characteristics, caused by manufacturing, distinguish each transmitter regardless of brand, model, or serial number [11]. As a result, datasets including these distinct emissions from multiple devices may be compiled, providing a rich foundation for training ML algorithms, that can recognise and categorise devices based on their unique fingerprints, providing strong authentication in a passive fashion.

The necessity for increased security in IoMT, as well as the promise that RFF offers in this context, motivate this thesis. It proposes the development of RFF-based techniques and machine learning models for the identification and authentication of IoMT devices. This work attempts to enhance the security mechanisms of these systems by incorporating these models into the IoMT infrastructure, thereby offering a robust mechanism for device authentication that complements existing protocols. This effort is expected to reduce vulnerabilities in the present IoT ecosystem, creating a safer environment for the critical services provided by IoMT.

1.2 Objectives

One of the main goals of this thesis is to revolutionise the use of ML in Radio Frequency (RF) by converting RF signals into images, increasing applications, and optimising wireless systems. CNN play an important role in this strategy, using their groundbreaking abilities in image recognition. We utilise the potential of CNN to find complicated patterns by translating RF signals into image data. Their ability to understand spatial feature hierarchies makes them a perfect candidate for this method. Using CNN in this way has a lot of potential for enhancing the reliability and effectiveness of RFF in device identification and categorization. The process of converting signals into images is another key aspect. By viewing RF signals in an image-based representation, we can better leverage the inherent structure and correlation in the data, providing a more intuitive way for ML algorithms to learn and make sense of the signals. Furthermore, a new decision-making approach will be implemented to classify devices using multiple features, introducing a novel aspect in this field. Thus, our primary goal is to precisely create features/images from RF signals and develop ML models that enable a reliable classification of devices, which is a crucial prerequisite for the implementation of RFF. The RFF technique is supported by these models. It is anticipated that the successful integration of RFF into a medical gateway will significantly increase its security and offer a strong line of defence against the frequent attacks that IoMT devices frequently experience. We anticipate a major gain in security by adding RFF to the medical gateway, creating a more secure and dependable IoMT environment. This is anticipated to

significantly reduce the likelihood of assaults, protect the integrity of IoMT devices and the sensitive data they manage, and eventually increase public confidence in the developing field of digital healthcare.

1.3 Research Context

This thesis has been developed inline with the ForPharmacy and Icare4NextG projects, which are being carried out at the Porto Research, Technology, and Innovation Center (PORTIC) of the Polytechnic Institute of Porto (IPP) with PORTIC being in charge of designing the medical IoT and cybersecurity technology to be included in these two IoMT-framed projects.

1.3.1 ForPharmacy Project

Due to their frequent interactions with patients, pharmacists play an important role as primary healthcare workers. This connection is increasingly important for addressing present and future health challenges, including the COVID-19 pandemic, as healthcare expenses increase, the ageing population expands, and chronic diseases become more prevalent. By allowing pharmacists to treat patients remotely, telepharmacy, a subspecialty of telemedicine in the pharmacy industry, opens up new opportunities for the delivery of healthcare.

In areas like pharmacovigilance and therapeutic adherence, the ForPharmacy project seeks to investigate and create telepharmacy solutions. The main goal is to improve patient safety and increase access to pharmacy treatment through digitalization and better interprofessional communication.

1.3.2 iCare4NextG Project

The iCare4NextG project seeks to develop a service framework for better home care and well-being using data-driven methodologies. To allow flexible user solutions and foster a supportive business environment for emerging digital business models, a multi-track strategy that includes service framework development, business modeling, and use case solutions will be put into place. The framework will put an emphasis on home care, integrated care, personalization, prevention, and involvement. Functional preventive home care is required as the ageing population and incidence of chronic diseases rise. The platform will be created using the integration, standardization, modularization, and flexibility principles to handle this. Home care's significance has been highlighted by the Covid-19 pandemic, and a number of contemporary use cases will spur platform development and highlight its efficiency. The framework includes layers of abstraction for data gathering, administration, and use, enabling the creation of customised solutions using data from various sources.

In this project, we are in charge of creating the medical gateway, connecting medical devices to it, and connecting it to cloud services so that Glintt's data about users' medications can be received, as well as sending medication records and data from medical devices. RFF is being instantiated in these projects, as a new security component for the Medical IoT gateways.

1.4 Thesis Structure

This thesis is composed of an introduction Chapter 1, where a theoretical introduction to the subject is presented, accompanied by the problem in hands, and how this Thesis addresses the mentioned security concerns. In chapter 2, there is a research background of the RFF, covering key topics to RFF. Following that, in Chapter 3, this thesis outlines some of the technologies used to begin the building blocks of ML, and the description of the medical gateway where the RFF is integrated. Chapter 4 describes how the RFF is integrated and how it functions in the medical gateway. The features utilised to train the CNN model to achieve the device categorization required for RFF implementation are also discussed. The results of the CNN models are shown in Chapter 5. Finally, in Chapter 6, numerous broad inferences are derived from the acquired results, and a few recommendations for future research are given.

Chapter 2

Research Background

This chapter provides the fundamental concepts relevant to this thesis and overviews the most recent work on these topics.

2.1 Overview of Healthcare Communication Security

IoT is a collection of technologies that enable a wide range of applications, devices, and objects to connect with one another via network technology. IoMT is a primary application of IoT, and it offers a variety of services to the healthcare business, including Implantable Medical Devices (IMD), RFID Tags, and wearable medical devices. The transmission of critical IoMT data from one device to another raises many security concerns. Side-channel attack, tag cloning, denial of service, tampering devices, rogue access, sniffing, and brute force can target these devices, violating patient data and also raising safety concerns. However it is challenging to address these as IoMT has restrictions such as limited battery capacity, memory, and computational power [9]. We further elaborate on these attacks and prospective solutions in the next sections.

2.1.1 IoMT Requirements

IoMT is a dynamic ecosystem of linked medical tools, services, and procedures that collaborate to improve the provision of healthcare. To protect data and guarantee the system's efficient operation, it is crucial to create and maintain specific requirements as this sophisticated network continues to expand. These specifications are

intended to protect the privacy and confidentiality of patient data, ensure data integrity, ensure system availability, confirm user actions through non-repudiation, verify user identities through authentication, manage user permissions through authorization, and maintain anonymity when necessary. The following are the details of these needs in further detail [12]:

- Confidentiality: In the context of the IoMT, confidentiality refers to the protection of patient health data that is exchanged with medical experts. This information needs to be protected from unauthorised access, suspicious surveillance, and malicious users that might hurt the patient or abuse the data. Despite general recommendations from IoMT standard protocols, it is crucial to implement network access control and data encryption to ensure secrecy.
- Privacy: In the context of IoMT, privacy refers to the safeguarding of confidential patient information against unauthorised disclosure and nefarious use. Currently, many nations have laws in place governing the gathering and archiving of patient health data, such as the Health Insurance Portability and Accountability Act (HIPAA) and the General Data Protection Regulation (GDPR). IoMT systems uphold these privacy laws and give users access to their private information.
- Integrity: Protecting patient data from unauthorised additions, deletions, or revisions is essential for IoMT healthcare systems. By doing this, the data is protected from manipulation both during wireless transmission and when it is at rest. Healthcare organisations are becoming more and more aware of the value of data integrity as medical data is used to depict diagnosis, treatments, and health problems. This integrity must be preserved through safeguards against unauthorised data changes.
- Availability: The readiness of servers and medical equipment to offer users services and data as needed is referred to as availability. This is especially important in healthcare systems because it's required to monitor patients continuously. Systems should have backup plans in place for uncertain data storage or transmission routes, especially against Denial of Service (DoS) or Distributed Denial-of-Service (DDoS) attacks, to ensure availability.
- Non-Repudiation: Non-repudiation describes the system's capacity to make users responsible for their deeds and ensure that they cannot retract their prior system usage. It assesses the system's capacity to affirm or refute the occurrence of an action. Digital signature methods are frequently used to achieve non-repudiation.
- Authentication: A user's identity must be verified when they log into the system. Verifying that a user is really the original source of the delivered data

is the goal of message authentication. The most secure type of authentication is mutual authentication, which calls for client and server authentication before secure data transfer. Due to resource restrictions in some IoMT devices, lightweight authentication techniques are growing in popularity.

- Authorization: It's critical to prevent unauthorised access given the delicate nature of medical data. Only reputable parties that possess the required credentials should be given permission to carry out specified tasks, such giving commands to IoMT devices or updating their software or security patches.
- Anonymity: This criteria makes sure that the patient's or doctor's identity is kept secret when speaking with unauthorised system users. During their conversation, the doctor and patient's names should remain concealed. Attacks that are passive should only expose a user's behaviour and not their identity.

2.1.2 IoMT Architecture Layers

Figure 2.1 is a vital reference point in the understanding of the complicated dynamics of the IoMT. This diagram depicts the IoMT architectural levels, from the fundamental perception layer to the critical application layer. Figure 2.1 depicts the methodical route of raw health data as it is collected, transported, analysed, and finally used for patient healthcare treatments.

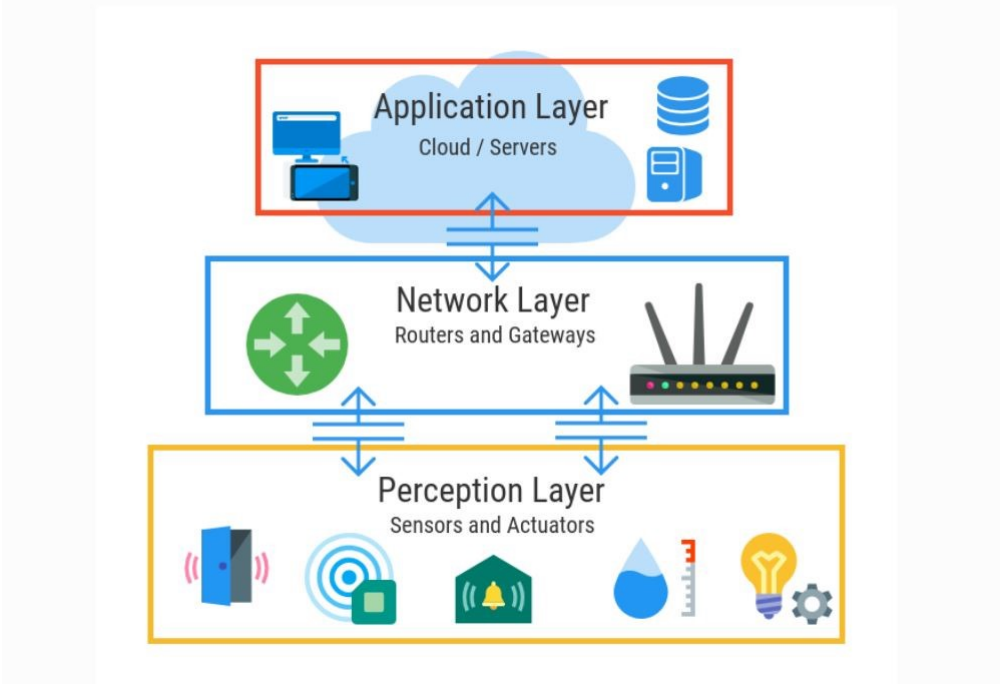


Figure 2.1: IoMT Architecture [13].

The perception layer is the foundation of the IoMT structure. It is in charge of identifying health-related parameters accurately. Sensors of various types, including those that can be incorporated in or attached to the body, such as pacemakers or smartwatches, can collect and record patient data. This data, which is acquired as raw or unprocessed values, is then transferred to the network layer via various communication mechanisms. The data is then relayed to the processing units, which are frequently on the cloud, through the second layer via an IoT gateway. The cloud then does the required analysis. If discrepancies in a patient's health data are discovered, this has serious consequences. These changes are then sent to the application layer and shown to physicians via remote monitoring devices such as smartphones or a dedicated Access Point. This enables prompt medical intervention, such as changing drug doses or changing prescriptions. The information can also be used by the patient for any following steps that are required [12].

2.1.3 Security Breaches in IoMT

There are numerous potential attacks in the IoMT space. These attacks differ not just in their strategy and methodology but also in how they affect the IoMT requirements. We can group these threats according to the attacked IoMT layer. The tables that follow give a brief summary of these attacks, their descriptions, and the areas of the requirements they affect [12]:

Table 2.1: IoMT potential attacks at Perception layer and their influence on the system's security requirements [12]

Attack	**Brief Description**	**Effects**
Side-channel attack	The side channels of the encryption device are used to gather data.	Confidentiality, Integrity
Tampering devices	To change the data (modification in a device utilising RFID or a communication link), the IoMT device is physically accessible.	Confidentiality, Integrity
Tag cloning	An attacker may copy data from a tag that has already been used or utilise data that was gained through a successful side-channel attack. The cloned tag might be used to get access to a restricted location or data, including private medical records.	Confidentiality, Authorization, Integrity
Sensor tracking	This attack violates the privacy of the patients. Through unprotected devices, attackers may acquire patient locations or create phoney GPS data. Sensitive patient information may also be revealed by other sensors, including those for wheelchair management, fall detection, and remote monitoring systems.	Confidentiality, Authorization, Integrity, Privacy

Table 2.2: IoMT potential attacks at Network layer and their influence on the system's security requirements [12]

Attack	**Brief Description**	**Effects**
Eavesdropping	A hacker intercepts and follows the hardware and communications needed to get data. There are numerous uses for the data that was taken illegally.	Confidentiality, Non-repudiation, Privacy
Replay	An attacker can intercept a signed packet and send it many times by using an authentication message that was previously sent between two authorised users.	Authorization
Man-in-the-middle	In order to gain access to their private data, this cyberattack attacks communication between two IoMT devices. Before the intercepted data is transferred to its intended recipient, the attacker can see it or change it.	Confidentiality, Authorization
Rogue access	In this attack, a fake gateway is placed within the wireless network's coverage area to grant access to unauthorised users and intercept traffic.	
DoS/DDoS	A DDoS assault, which differs from DoS attacks from a single node in that it comes from several sources, floods a target with messages or connection requests in order to prevent genuine users from using the service.	Availability
Sinkhole	In this attack, a rogue node draws traffic by promising improved connection quality. Other attacks, including eavesdropping or selective forwarding, can be initiated after a successful execution.	
Sniffing	Data being exchanged between two nodes is passively intercepted during a sniffing attack. This enables the attacker to watch while data is transferred across system layers.	Confidentiality
Selective Forwarding	A rogue node may change, delete, or only forward particular messages to other nodes in the network during an attack. As a result, the recipient only receives partial information.	All

Table 2.3: IoMT potential attacks at Application layer and their influence on the system's security requirements [12]

Attack	**Brief Description**	**Effects**
Brute Force	Automated programmes are frequently used by attackers to create a variety of password combinations until they succeed. IoMT devices are seriously vulnerable to the dictionary attack.	Confidentiality, Integrity
SQL injection	In this attack, a flawed SQL statement is introduced into the application's backend database. Sensitive patient data may be compromised or changed in a successful assault.	All
Account hijacking	Many IoT devices interact across networks in transparent text or with poor encryption. An attacker can steal an account by intercepting the packet during end-user authentication.	Confidentiality, Integrity
Ransomware	Ransomware encrypts important data and requests a ransom to decrypt it. Attackers can encrypt private data, such as health records, and keep the decryption key in exchange for money.	Integrity, Availability

In this thesis we focus on communication cybersecurity, hence we cover by focusing on this three layers.

2.1.4 Security Solutions

A wide variety of attack tactics are pitted against an equally wide variety of defences in the complicated world of IoT device security. These defence mechanisms, created to mitigate or even eliminate such risks, cover four major areas: the communication layer, network routing, transport layer, and application layer, each of which provides particular solutions to guarantee strong IoT security.

Communication Layer Security Solutions

A fundamental part of deploying an IoT network is ensuring secure end-to-end connectivity. One proposed method for enabling such security is to use compressed IP security. Encapsulation Security Payload (ESP) and the Authentication Header (AH) are used in this method to authenticate and encrypt messages between the regular internet and the sensor network. The results show that employing established IPv6 techniques, compressed IPsec can assure message integrity. Furthermore, they developed an ESP for IPsec/6LoWPAN, which they compared to link-layer security for IEEE802.15.4.The authors discovered that while IPsec delivers greater security and has a faster response time than link-layer security, it consumes more energy than link-layer security when tested on an IEEE 802.15.4 testbed [14].

Another protocol for IoT security was proposed, which included a feasible authentication solution based on elliptic curve cryptography and leveraging a secure key function. They also implemented Role-Based Access Control (RBAC) in IoT network applications for access control rules. However, the proposed security analysis was not realistically applied, and IoT sensor nodes had a large connection overhead [14].

Network Routing Security Solutions

The proliferation of IoT devices has increased the demand for security solutions to protect these devices from cyber-attacks. Despite the implementation of encryption and authentication, IoT networks remain vulnerable to a variety of attacks, including sybil, black-hole, sinkhole, fragmentation, selective-forwarding, and man-in-the-middle attacks. Intrusion Detection System (IDS) that can be put at every node of Low-power and Lossy Networks have been proposed to detect these attacks. However, because the nodes have limited resources, IDS must be optimised for each physical object [14].

Some researchers have proposed lightweight intrusion detection system based on harmful pattern detection, which evaluate detection in terms of energy usage and execution time using two schemes: early detection and auxiliary shifting. Another option is INTI, a distributed IDS architecture that combines the "watchdog" and "reputation and trust" approaches to find and mitigate attacks [14].

Others have developed hybrid intrusion detection systems to detect selective forwarding, sinkhole, and wormhole attacks. These IDS are made up of primary IDS modules, a 6LoWPAN mapper, and a mini-firewall at constrained devices and border routers. To identify attacks, the proposed IDS employ a variety of techniques such as monitoring routing traffic, evaluating signal intensity, and identifying aberrant activity [14].

While these IDS have yielded encouraging outcomes, they are not without limits. Some intrusion detection systems, for example, have a high false-positive rate, spend more energy during an assault, or have decreased accuracy as network size increases. As a result, the selection of an appropriate IDS is determined by the unique attack scenario and resource restrictions [14].

Another study proposes an efficient, secure route optimization protocol for the Proxy Mobile IPv6 (PMIPv6), which improves on the existing routing protocol (PMIPv6) when it comes to security, specifically when it comes to authentication, complete forward secrecy, key exchange, and privacy when supporting the protocol mutual. Their method ensures safe transmission while reducing packet loss, latency, and delay [14].

A third paper suggests a novel trust mechanism based on the SecTrust-RPL routing protocol, which was deployed in a test bed experiment. The suggested

technique detects both sybil and rank attacks. The authors, however, did not assess the impact of their approach on performance, energy consumption, and performance overhead [14].

Another paper describes a wormhole attack detection technique for the RPL IoT routing protocol. IDS is installed at both the border router and the host sensor nodes, and they analysed the influence of their IDS on the success rate of detecting the wormhole assault. The results showed that with eight sensor nodes, the true positive rate was high, but fell as the number of sensor nodes increased.

Another study presented a hybrid IDS that targets sinkhole and cloneID threats and analysed its IDS impact in terms of performance and detection. The detection rate, according to the authors, was 100%. However, the statistics show that as the number of sensor nodes increased, the detection rate declined, and the energy and power consumption was higher than the prior IDS they tested [14].

Transport and Application Layer Security Solutions

Some research articles' authors performed a comparison investigation of the security aspects of MQTT and CoAP, focusing on the transport protocols used, specifically MQTT with TLS and CoAP with DTLS. The investigation took into account security mechanisms such Raw Public Key, Certificates, and Pre-Shared Key. They discovered that MQTT provides a diverse security alternative to lightweight and PSK certificates, but certificate-based encryption and authentication remain the most secure [14].

One of the papers examines the use of RSA cryptography on sensing devices through the use of trusted-platform modules. The authors used a DTLS cypher suite TLS-RSA-with-AES-128-CBC-SHA to evaluate the system in terms of latency, energy usage, and memory. Another publication investigated this proposal further in wireless sensor networks. Another proposal for a transport layer authentication security scheme based on the elliptic curve cryptography algorithm [14].

One study developed Lithe, a DTLS/CoAP integrated protocol that uses a compression algorithm to reduce header overhead and is more efficient than conventional CoAP/DTLS. However, it is vulnerable to DoS attacks. Another paper describes E-Lithe, a lightweight DTLS for IoT that customises the DTLS packet to reduce energy consumption and execution time. Another proposed system combines identity-based encryption and a Diffie-Hellman encryption algorithm to provide authentication and integrity security without the use of a digital certificate. Another study tries to improve the CoAP protocol's hash function for integrity security in smart home applications, and a novel architecture suggests the use of cryptography methods rather than DTLS to ensure data confidentiality and integrity in CoAP-based smart home apps [14]. Due to many key obstacles, such as high computational complexity, high communication overheads, high latency, and inadequate security of resource-constrained

mobile IoT devices for healthcare applications, authentication with typical cryptographic approaches does not appear to be always practical. Key management is only appropriate for classical cryptosystems. More importantly which distributed system such as IoMT becomes an even more complex challenge. Traditional cryptography systems' static properties make it impossible to detect unauthorised organisations reusing compromised keys. 5G and beyond networks have enabled rapid progress in the development of IoT devices on diverse platforms. Because of their low cost and small size, such devices are useful for a wide range of applications [15]. Authentication is a critical component of cybersecurity architecture, serving as the first line of defence against unauthorised access to sensitive information systems and data. It is the process of verifying the legitimacy of users, devices, or systems in order to prevent potential security concerns from occurring. Passive authentication technologies such as RFF emerge as effective solutions as cyber threats increase. We may strengthen our cybersecurity posture by augmenting our security methods with a layer of continuous and frictionless authentication, ensuring that only authorised people and devices access vital resources and information.

2.1.5 Towards New Methods of Authentication

Wireless communication networks have grown in popularity and development in recent years. Attacks that leverage information security vulnerabilities such as identity impersonation, replay attacks, and device cloning continue to emerge. Accurately identifying and authenticating the objects is the primary problem.

Traditional authentication mechanisms are implemented at the application layer, using cryptographic algorithms to generate numerical results that are difficult for third parties to counterfeit. Mechanisms like this have risks of protocol security loopholes and key leakage. Key leakage and security issues discovered at the application layer can be mitigated by physical layer authentication.

The use of ML authentication approaches on the physical layer has piqued the interest of academia and industry. This approach smartly and efficiently secures 5G-enabled portable devices. For authenticating entities, ML-enabled approaches such as supervised and unsupervised learning and DL are useful. Because ML approaches take into account unique multidimensional qualities, they can provide more secure, better authorised, fairly accessible, highly dependable, dynamic, and self-driven device verification for an unknown network state. Poor security and lengthy delays can have an impact on the functioning of vital applications like healthcare. Such challenges can be addressed with a quick cross-authentication solution that makes use of SDN-aware joint cryptographic and non-cryptographic features [15].

All of the previously mentioned concepts might be realised by implementing RFF authentication, which would strengthen the safety measures of those systems without limiting the performance of the IoMT devices [15].

2.2 Radio Frequency Fingerprint

Physical layer authentication technology offers more solutions to wireless communication security issues. Physical layer security authentication technology research is still in its early phases, and its core theory has not kept pace with other wireless communication technologies' progress. The method of recognising the hardware characteristics and specific features or signatures inherent in radio frequency waves broadcast via a wireless channel is known as RFF. The manufacturing process for chip components produces hardware flaws that make each emitter unique, regardless of brand, model, or serial number [16]. These flaws are necessary to make the cloning process challenging. Using RFF to discriminate between illicit and legitimate devices is a new physical layer strategy for protecting communication system security. RFF can be utilised for physical identification and access authentication of wireless equipment, just like everyone has their own unique biometric fingerprint characteristics. Figure 2.2 depicts a typical RFF identification.

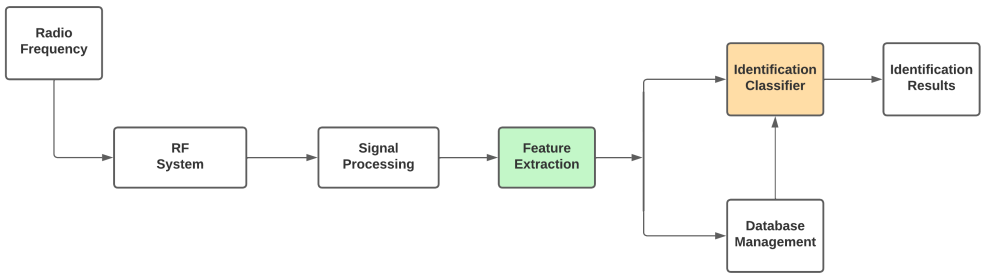


Figure 2.2: Typical RFF identification process.

The procedure pulls features from a given radiation source from the received signal time series for classification and recognition, which is fundamentally a pattern recognition problem, as shown in Figure 2.2. Looking at the diagram again, we may resume the process as follows [17]:

1. The RF signal segment utilised to extract the radio RFF is intercepted and preprocessed.

2. Following the acquisition of the signal to be classified, the RFF can be obtained using various transform domain approaches such as frequency domain analysis, time-frequency analysis, fractal, high order spectrum, constellations, or RFF can also be recovered in the modulation domain.

3. The DL method can also be used to process the signal directly. As it can perform some feature selection or feature dimensionality reduction, as well as feature extraction and classifier creation.

2.2.1 Signal Processing

The RF signal is intercepted and preprocessed, as indicated in Section 2.2. In recent years, Software-Defined Radio (SDR) have been employed in this step. A SDR is a programmable transceiver that may operate numerous wireless communication protocols without the requirement for hardware changes or updates. SDR have two benefits over lab-grade receivers for RFF. The first is cost; lab-grade receivers can cost up to twice as much as SDR. The second is general-purpose accessibility. SDR is made to offer a base and modular capacity that can be improved by building processing chains out of functional blocks to quickly add or integrate additional features [18].

The In-phase/Quadrature (IQ) are samples of the obtained representation of the preprocessed and recorded RF signal, comparing the "in-phase" component to the "quadrature" component, a phase shift of 90 degrees has occurred. Each sample can be expressed as a complex number, where the real part increases the in-phase component and the imaginary part increases the quadrature component.

The IQ convention is an alternate approach to describe magnitude and phase of a RF signal, which leads to the use of complex numbers and the ability to represent them on a complex plane. A complex number also has a magnitude and phase, which makes more sense when viewed as a vector rather than a point. This representation is better explained with an example. Consider the Figure 2.3 and imagine that the point $0.7-0.4j$ is to be transmitted. The following equation can be used to represent the resulting signal to be transmitted.

$$x(t) = I\cos(2\pi ft) + Q\sin(2\pi ft) = 0.7\cos(2\pi ft) + -0.4\sin(2\pi ft).$$

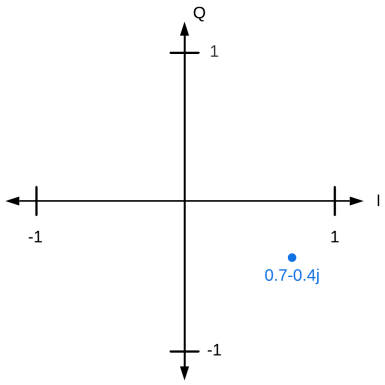


Figure 2.3: IQ complex number representation.

A trigonometry identity, $a\cos(x) + b\sin(x) = A\cos(x - \phi)$, can be employed, where A is our magnitude discovered with $\sqrt{I^2 + Q^2}$ and ϕ is the phase, equal to

$\tan^{-1}(Q/I)$, resulting in the equation:

$$x(t) = 0.806\cos(2\pi ft + 0.519).$$

Despite the fact that it begins with a complex number, the real component is conveyed, which is required because the imaginary component is not transmitted with electromagnetic waves. The complex numbers are useful for obtaining and plotting the features required to proceed in the fingerprint process [19]. The SDR collects the RF signals, but these captures must be passed to a computing device. There are numerous ways to do this step, ranging from writing code to capture and process these data to using software that already includes a major percentage of the necessary functionality, such as GNU Radio [20].

2.2.2 Feature Extraction

Following the acquisition of the signals, the next phase involves the extraction of unique features from various portions of the signal. There are various techniques for extracting multiple elements from an RF signal. According to [21], these features, as shown in Figure 2.4 can be classified into three types: transient-based, steady-state-based, and other approaches based on various signal portions employed for feature extraction.

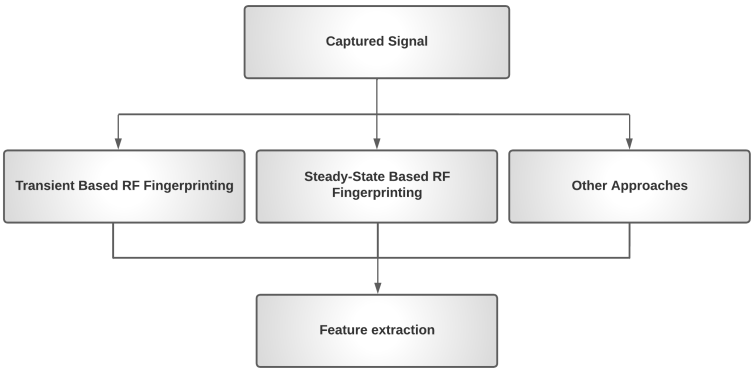


Figure 2.4: Feature Extraction.

Transient-based RF fingerprinting techniques make use of the transition between turning on and off a transmitter prior to the actual transfer of signal data. This method necessitates precise transient extraction (start point and duration). Transient extraction methods are heavily influenced by signal noise, hardware, and the distance between the SDR and the device.

Only when the transient is precisely retrieved does transient-based analysis provide high reliabilty regarding the fingerprint procedure. The lack of transient analysis can make distinguishing devices from the same manufacturer more difficult (same

model). For good transient extraction, very high sampling rates are required, necessitating expensive receiver structures, and isolating the transient signal and locating the start point in channel noise are extremely difficult due to non-stationary properties [22].

The unique features retrieved from the modulated component of the signal are the focus of steady state-based techniques. For physical layer identification, mostly five unique aspects of the modulated signal are used: frequency error, synchronised correlation, IQ origin offset, and amplitude and phase faults [23]. The following considerations must be made: The steady-state signal is not shared by all emitters. The transient signal is constantly there throughout transmission. However, as previously indicated, a greater sample rate is needed to retrieve the transient signal. Also, in comparison to the other methods, they usually require unique wireless technology and/or extract different signal and logical layer features, two of which will be detailed and explained:

- In [24], a Constellation Trace Figure (CTF) feature is shown. In practise, the non-linear behaviour of the amplifier at the transmitter will significantly distort the RF signal, resulting in non-linear signal offset at the receiver, including IQ offset and phase offset. These characteristics are often taken from the traditional constellation map. However, there is a flaw in the latter; it only represents the samples at the decision point. As a result, the CTF method was introduced, in which the base point of the IQ axis on CTF is established as a central point and the CTF is equally split by a fixed angle from the central point.

- A technique was developed in [25] to give further protection against spoofing attacks. It focuses on identifying authorised objects and using their fingerprints to detect prospective intruders in the wireless environment by analysing the PSD of physical signals.

2.2.3 Machine Learning Approaches for Identification and Classification

Depending on the ML method and features used during training, different results will be produced. As a result, modifications to feature extraction, signal processing, or the machine learning method itself may be required. Below are three examples that demonstrate this process:

Differential Contour Stellar-Based RFF

The fingerprinting of devices using the WiFi communication protocol is covered in [26]. They employ images created from signal capture to train a machine learning

system to determine which device the image belongs to. These images are created by a differential process of in-phase and quadrature signals. Following the differentiation of the signal, the production of the designated differential contour stellar, as shown in Figure 2.5, was carried out.

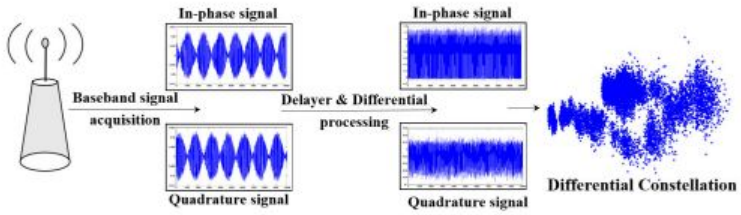


Figure 2.5: Creation of differential Contour Stellar Image [26].

Following the generation of the images, 400 images were picked at random from each class, 320 were used to train the Deep Neural Network (DNN), and 80 were utilised for validation. For the 20 WiFi devices utilised in this testing, this resulted in a recognition success rate of 90.4%.

Imaging Time Series for Internet of Things Radio Frequency Fingerprinting

The RFF process in IoT communication devices is highlighted in [27], with 9 nRF-24LU1+ devices used to construct IoT sensor networks. The way authors approach to fingerprinting is centred on transforming time series into images. The distance between two locations in the time series, from the start of the burst to a different amount of samples, is iterative. Three complex number distances were used: order 2 Minkowski distance (Euclidean distance), order 3 Minkowski distance, and the Chebyshev distance. The generated images are then processed to extract basic image processing features such as Histogram of Oriented Gradients (HOG), Binary Robust Invariant Scalable Keypoints (BRISK), Speeded-Up Robust Features (SURF), and Local Binary Pattern (LBP).

Different accuracies were obtained as a result of the three distances employed in the complex numbers, as well as the amount of samples used for each class. The precision for Minkowski order 3 (size 500 samples) was about 98.36%, for Minkowski order 2 (size 800 samples) was approximately 94.83%, and for Chebyschev (size 200) was approximately 92.02%. A Support Vector Machine (SVM) was employed as the machine learning algorithm for these results. Following these experiments, a comparison was done in which the traditional statistical features used for RFF were employed instead of the images. They discovered that models trained with images had higher accuracy, and it is important to note that the disadvantage of image conversion is related to conversion time.

RF Fingerprint Reconition Based On Spectrum Waterfall Diagram

According to [28], they proceeded to translate the radio signals into two-dimensional images based on earlier work. MATLAB simulations were utilised to produce the dataset used for image conversion. The signal dataset used Quadrature phase-shift keying (QPSK) modulation, had a sample frequency of 20Mbps, a sampling interval of 0.05µs, a signal-to-noise ratio of 10dB, and 52 subcarriers. There were four pilot signals and the rest are data signals, as 20 separate devices were constructed. These signals were then translated into 300x300 spectrum waterfall diagrams, which were put into the artificial intelligence programme that they created.

It should be noted that a dataset of 100,000 images was generated for the 20 devices, 80000 of which were used for training and 20,000 for validation. With this model they were able to obtain an accuracy of 89.01%. To compare their results with other approaches, they decided to use a different CNN model in which they fed IQ data in raw format to the model, obtaining an accuracy of only 58.80% and increasing the training time by about three times.

The essential components required to grasp the notion of AI will be explained in the following parts. Explaining the fundamental concepts and elements that underpin the field. From ML algorithms to neural networks, a greater grasp of the topic is provided, as well as an explanation of CNN, which is critical to the work performed.

2.3 Artificial Intelligence

As illustrated in the diagram below, Figure 2.6, ML, is a subset of AI. AI encompasses all of the techniques that enable machines to "replicate" human intelligence to some extent [29].

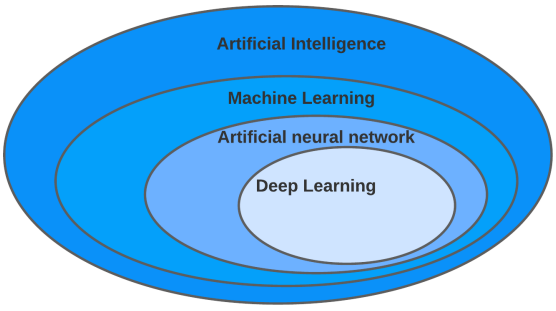


Figure 2.6: Scheme for Artificial Intelligence Concepts.

DL, in turn, employs other techniques that enable machines to benefit from experience in order to improve their tasks. This AI technique is used when adaptability is required and the problem is complex at the human level. In this type of learning,

training data must be entered and model training must be performed. The more training data there is, the more experience there is, and the better the result [30]. Learning can be carried out in a variety of ways and classified into various types. DL is no exception, and it can be classified into different types based on various parameters.

Unsupervised Learning

There is no supervision of the learning process in unsupervised learning, and the data is provided without any labelled training data. Instead, through exploration and analysis, the goal is to discover patterns and relationships in the data. The algorithms are mostly used for association and clustering problems, where the dataset is subdivided into sets with similar characteristics, due to the nature of this learning. This type of clustering can then be used to detect anomalies, with the anomaly identified by being outside of a pattern established by similar characteristics [31, 32].

Supervised Learning

Unlike unsupervised learning, this sort of learning includes extra instructions on how to catalogue the training data along with it. The purpose of this learning is to train the algorithm to recognise the relationship between the labels and the data presented, so that it can catalogue test data without the help information linked. This type of learning is typically employed for cataloguing problems, which are classified into two types: regression problems and classification problems. The regression problems are those that seek to quantify data by attributes such as weight, price, and quantity. In supervised learning for classification, we can find various sorts of algorithms, each with its own set of strengths, limitations, and utility distributions [33].

Linear classifiers make classification decisions based on the linear combination value of all characteristics. Although the decision tree approach is one of the fastest classification algorithms, linear classifiers are also employed to handle issues requiring fast classification, particularly when features are considered scattered. These are most effective when the number of feature dimensions is large, as in document categorization (number of words per document). The logistic regression technique, as well as algorithms such as the SVM, are examples of linear classifiers [33].

The decision tree and, by extension, the random forest algorithm are both supervised learning algorithms. This type of algorithm is similar to a real tree in that the root represents the input, the internal nodes the questions/decisions to be made, and the branches the rules to reach a final category/decision (leaf). Neural networks are typically utilised for enormous amounts of data. Other algorithms used in this

learning include boosting algorithms, quadratic classifiers, Bayesian networks, and probabilistic classifiers such as Naive Bayes classifiers [33].

Reinforcement Learning

Reinforcement learning is a type of learning in which an agent is trained to make decisions in an environment by performing actions and receiving rewards. The agent's goal is to learn a policy that maximises the cumulative reward over time, which is a mapping from states to actions. The agent explores the environment, gathers information about the state-action-reward transitions, and updates its policy based on the observed rewards [34].

2.3.1 Artificial Neural Network

ML is a popular field of study because it provides powerful tools for discovering patterns in large amounts of complex and unstructured data. Unlike traditional statistical techniques, machine learning does not rely on assumptions about the underlying structure of the data. This breakthrough is accomplished by shifting from a deductive problem of finding a rule to an inductive problem in which the data is used to inform the best rule characterising it.

Artificial Neural Network (ANN) is a soft computing tool used in ML that mimics the human mind's ability to reason and recognise patterns. ANN learn by analysing the relationships between input and output data provided by training, and they can generalise their output, making them suitable for non-linear problems requiring experience and knowledge of the environment [35].

As shown in Figure 2.7, ANN typically has three layers: an input layer, one or more hidden layers, and an output layer. The number of hidden layers and neurons in each hidden layer can be changed. Before being used, the network is trained to achieve a very low error rate. The network is then tested with previously unseen data to determine the accuracy of the developed model [35].

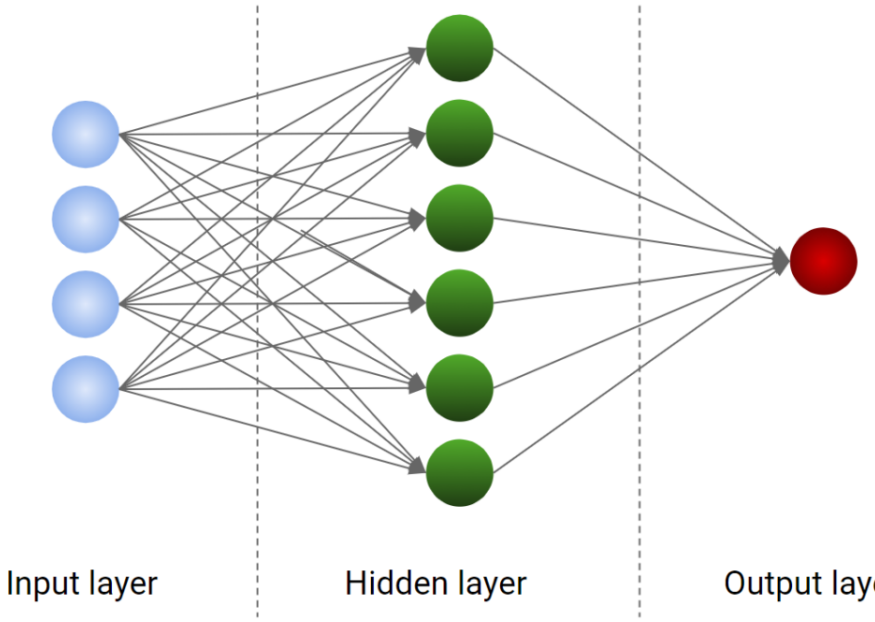


Figure 2.7: Basic Artificial Neural Network Architecture [36].

ANN are modelled after the human nervous system and can be trained to perform tasks such as data classification and pattern recognition. One of the most significant advantages of ANN is their ability to learn from large amounts of complex data [37].

2.3.2 Deep Learning

DL is a subset of ML that is concerned with algorithms that are inspired by the structure and function of the human brain. DL algorithms can process massive amounts of structured and unstructured data. The core concept of DL is ANN, which allow machines to make decisions.

The way data is presented to the machine is the key distinction between DL and ML. DL networks work on multiple layers of ANN, whereas ML algorithms typically require structured data, as can be seen in Figure 2.8 [38].

In the case of the DL architecture, however, the extraction and processing procedure is completed automatically and without much human effort, which is the inverse of ML. This architecture is made up of numerous layers in order to accomplish the entire extraction and classification procedure.

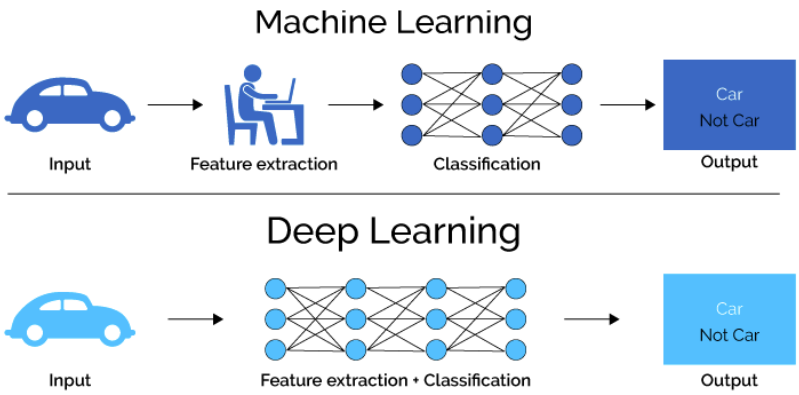


Figure 2.8: ML and DL comparison [39].

Because of these qualities, this form of neural network architecture is appropriate for issues involving vast amounts of potentially low-quality and unstructured input, such as complicated computer vision and natural language analysis tasks. The performance of these networks, however, is dependent on technology (the usage of Graphics Processing Unit (GPU) or Field-programmable gate array (FPGA)), a lengthy training process, and big training data sets. Because large networks require more training time, they are more prone to overfitting, rendering the model useless for data outside the training set [40]. Overfitting occurs when a model trains on the same training data for an excessively lengthy period of time and/or when it is overly complicated. This causes it to begin memorising noise in the training data and overfitting itself to it. This issue can be mitigated by increasing the number of training data, diversifying them, for example, through modifications such as image rotation, and/or employing regularizer algorithms [41].

2.3.3 Deep Neural Networks

As previously stated, DNN feature several inputs and outputs, as well as multiple hidden layers between them. Each layer can transform data into meaningful information for the following layer, allowing for continuous learning alongside data processing.

Convolutional Neural Network

A CNN is made up of numerous layers that allow for the extraction of features and then classification, as seen in Figure 2.9.

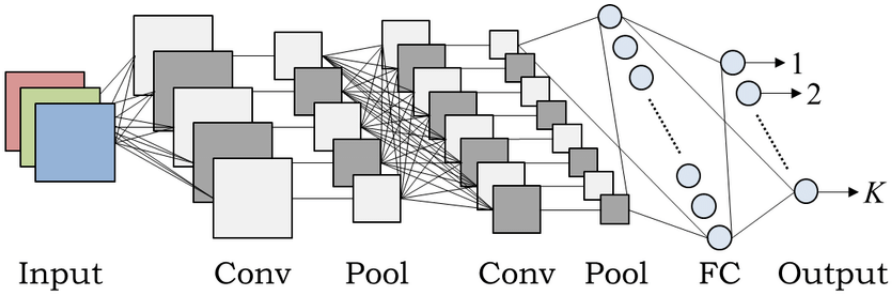


Figure 2.9: Architecture CNN [42].

CNN does not need to manually extract features. CNN architecture is influenced by visual perception. CNN kernels represent diverse receptors that can respond to different features; activation functions imitate the function that only neural electric signals over a specific threshold can be sent to the next neuron. People devised loss functions and optimizers to train the entire CNN system to learn what we expect. CNN has various advantages over fully connected systems [43]:

1. Local connections—Each neuron is no longer connected to all neurons from the preceding layer, but only to a small number of neurons, which helps to reduce parameters and accelerate convergence.

2. Weight sharing—when a collection of connections share the same weights, parameters are reduced even further.

3. Dimension reduction through downsampling—a pooling layer uses the notion of image local correlation to downsample an image, which can minimise the quantity of data while maintaining important information. It can also decrease the amount of parameters by deleting unnecessary features.

To be more detailed, four components are normally required to develop a CNN model. Convolution is a critical phase in the feature extraction process. Convolutional outputs are known as feature maps. We will lose information on the boundary if we choose a convolution kernel of a given size. As a result, padding is created to increase the input with a zero value, which can indirectly change the size. Furthermore, the stride is used to modulate the density of convolving. The smaller the density, the longer the stride. Following convolution, feature maps include a large number of features, which is prone to overfitting. As a result, pooling (also known as downsampling) is proposed to eliminate redundancy, including maximum and average pooling [43].

Convolutional Layers

It is necessary to comprehend the features of the image in order to process them. Humans have this ability because they can view an image and understand the entity

and its form that is depicted in the image. This technique is carried out by neural networks through the pixels of a picture.

As shown in Figure 2.10, a picture can be transformed into a pixel matrix. Each value of those pixels, as well as the values of his neighbours, helps the network understand what to look for in the photos [44].

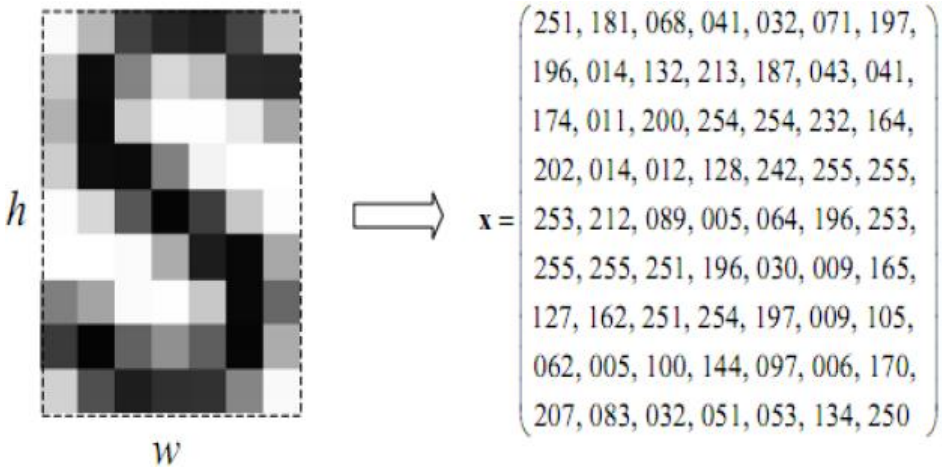


Figure 2.10: Pixel matrix [45].

A higher quality image, has a greater number of pixels. The intensity level can be represented by pixels ranging from 0 to 255 [44]. The convolution layers use convolution masks/filters that divide the image into small sections to read out these pixels and compare them to neighbouring pixels, as seen in Figure 2.11. Other layers use these small regions known as receptive fields.

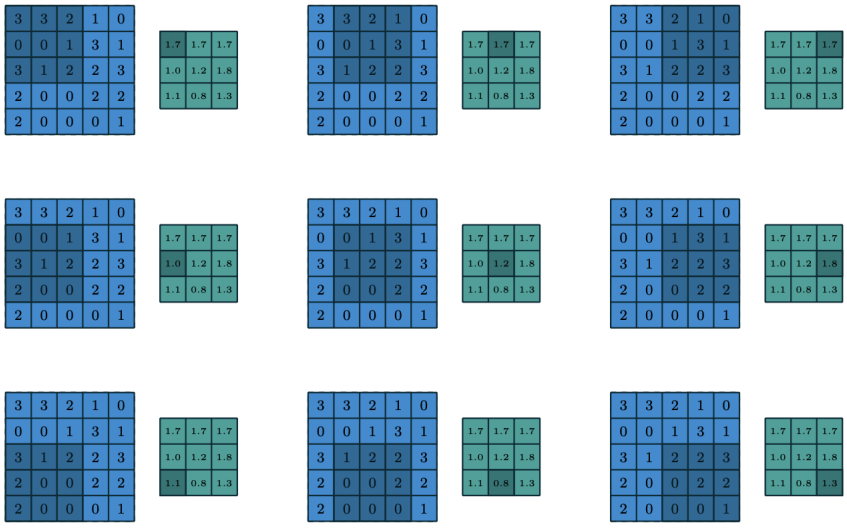


Figure 2.11: Convolution of the image with a filter [46].

The Convolution Layer is the simplest basic but also the most crucial layer in a CNN. It essentially convolves or multiplies the pixel matrix generated for the provided picture or object in order to build an activation map for the given image [47]. Convolution layers are classified as local operations and can be classified according to the type of convolution operation, filter size, and other factors [48].

Pooling

Pooling is a key step in further lowering the dimensions of the activation map, retaining only the important elements and reducing spacial invariance. As a result, the model's learnable features are reduced. This contributes to the resolution of the overfitting issue. Pooling enables CNN to absorb all of the distinct dimensions of an image, allowing it to effectively recognise the provided object even if its shape is distorted or at a different angle. Pooling can be classified into several categories, including maximum pooling, average pooling, stochastic pooling, and spatial pyramid pooling. The most prominent of them is max pooling [47].

Clustering Layers

Clustering layers allow you to group similar inputs together, which is important for a variety of data processing applications. They are also an effective feature learning technique, as they can learn representations of the input data that are useful for clustering and, in many cases, other tasks. This minimises the number of parameters as well as the complexity and size of the map, which reduces overfitting and improves network performance [44].

Clustering by maximum value, average, L2 norm, overlapping, and spatial pyramid clustering are some of the clustering methods that can be used on these layers. Clustering by maximum value is the most commonly used method, and it filters matrices by the maximum values in each quadrant, as shown in Figure 2.12.

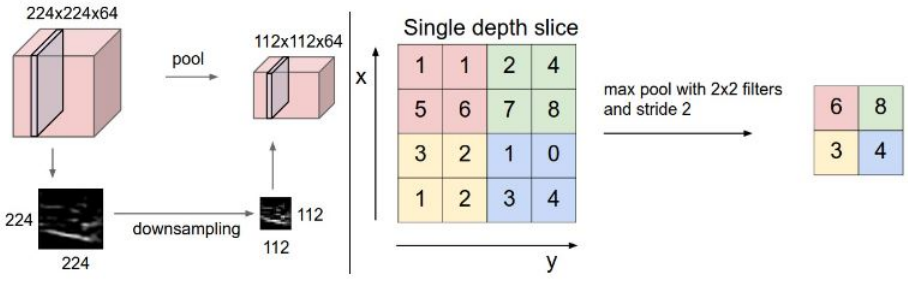


Figure 2.12: CNN grouping [49].

Regardless of the type of operation, clustering can be expressed by the following equation, where Z_i^k denotes the clustering map of layer i of feature map k, F_i^k. The

clustering type is represented by $g_p()$ [48]:

$$Z_i^k = g_p(F_i^k)$$

Activation Function

The activation function is a nonlinear function that aids in learning and decision making, particularly in complex patterns. This function can be represented mathematically as follows [44, 48]:

$$T_i^k = g_a(F_i^k)$$

Where F_i^k denotes the feature map (receptive field), $g_a()$ denotes the use of a nonlinear function-activation, and T_i^k the due converted output [48]. The graphs in Figure 13 demonstrate various activation methods, such as Rectified Linear Unit (ReLU), sigmoid, and hyperbolic tangent [48].

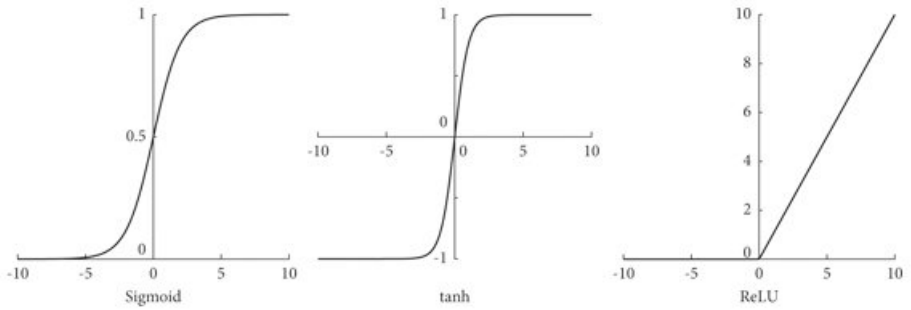


Figure 2.13: Activation functions [50].

However, the former is preferable due to its simpler computation and lack of convergence issues or abrupt variation of the derivative as seen in others [48]. Using the following function [44], the ReLU function sets the maximum threshold to 0, implying that negative values become zero without modifying the volume:

$$f(x) = max(0, x)$$

This function is typically used after a convolution (linear transformation) layer, but it is also useful in the final layer. Another sort of activation function is chosen for this based on the algorithm's demand and goal. Table 2.4 shows the types of functions most commonly used in the last layer based on the purpose [44].

Table 2.4: The last layer's activation functions prioritize the neural network's ultimate goal.

Objective	Activation function
Binary classification	Sigmoid
OvR multiclass classification	SoftMax
OvO multiclass classification	Sigmoid
Continuous value regression	Identity

There are two types of multiclass classification: One-vs-Rest (OvR), which separates multiclass classification into binary classifications per class, and One-vs-One (OvO), which performs binary classifications for each pair of classes [51].

Batch Normalization

Batch normalisation is a regularisation function that is used to overcome problems caused by changes in the distribution of hidden unit values, which can pose convergence constraints. The batch normalisation of a changed feature map, F_i^k, can be described by the following formula [48]:

$$N_i^k = \frac{F_i^k - \mu_B}{\sqrt{+\varepsilon}}$$

Where N_i^k denotes the normalised map, μ_B the batch map mean, σ_B^2 the variance, and is utilised to avoid division by zero [48].

Dropout

Dropout is a popular method for regularising neural networks in which specific units and linkages are randomly eliminated with a given frequency. As seen in the initial schematic of Figure 2.14, this type of network frequently includes a large number of connections, which can lead to overfitting. This strategy is critical because it reduces the amount of connections, which improves network performance and combats overfitting [52].

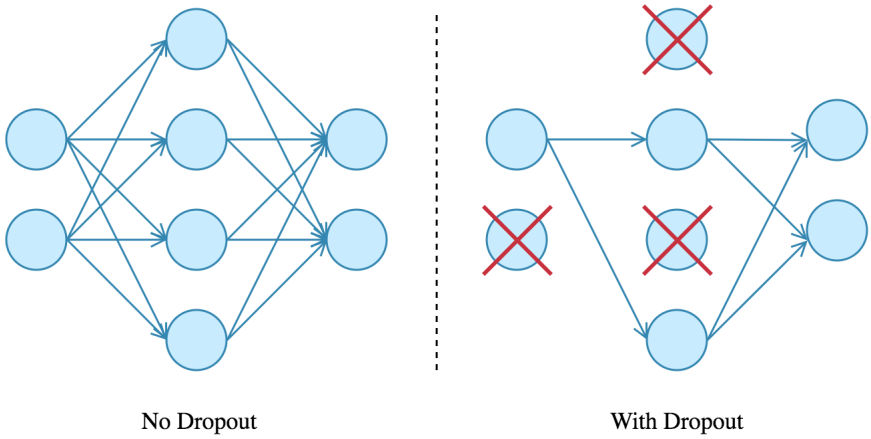


Figure 2.14: Dropout [53].

Fully connected layer

Unlike the previous layers, the completely connected layer, which is usually the last layer of the network, is a global process (convolution and clustering layers). This layer's primary job is classification, for which it does a global analysis of the feature maps collected by the previous layers [48].

Data augmentation

Data augmentation is a technique used in DNN to artificially increase the size of the training dataset by performing modifications to existing data. This prevents overfitting and enhances model generalisation. Three fundamental components of data augmentation in DNN are as follows [54]:

1. A common type of augmentation is flipping, rotating, scaling, cropping, and adding noise to the raw data. These augmentations seek to imitate data variations that may arise in real-world circumstances. For example, turning the image horizontally aids the model in learning the image's invariances [54].

2. Use of augmentations: Data augmentation is typically used during the model's training phase. In addition to the original data, the augmented data is utilised to train the model, allowing the model to learn from a bigger and more diversified dataset. The parameters of augmentations are usually decided through testing and fine-tuning [54].

3. Augmentation restrictions: While data augmentation can be a beneficial tool, it does have some limitations. For example, it may not be suited for all sorts of data, such as audio or text data, and the augmentations used must be carefully chosen to ensure that they are relevant to the problem at hand. Furthermore,

augmentations can be computationally expensive, especially for large datasets, and may necessitate a significant amount of computing power [54].

2.3.4 Transfer Learning

Transfer learning is a machine learning technique in which a model generated for one project is utilised as the foundation for a model on a different task. The concept is that the features and knowledge obtained by the model on the first job can be used as a starting point for the second assignment, rather than training a model from scratch. This may result in shorter training times, improved performance, and the capacity to use less data [55].
There are numerous types of transfer learning, including instance-based, feature-based, and fine-tuning. Instance-based transfer learning occurs when the model's parameters and architecture are reused but the model is trained on new data. This is effective when the new work has a substantial amount of labelled data and is similar to the old task [55].

Feature-based transfer learning occurs when the model's learnt features are used as input to a new model rather than the raw input data. This is useful when the new task differs from the original task but the model's features are applicable to the new assignment [55].

Fine-tuning is the process of modifying the model's parameters and retraining it on new data. This is useful when the new work is comparable to the original task but the model needs to be modified to accommodate the new task [55].

The emphasis in the next chapters will be on the application and framing of ML for RFF. A portion of the thesis will be devoted to integrating the RFF process into the medical gateway, thereby broadening the practical application of this technology in the healthcare industry. Furthermore, it will be explained the CNN model's design, emphasising how this specific model was important in achieving device categorization. Finally, a thorough assessment of the features employed in training these models will be presented, followed by a detailed interpretation of the results. The order of these chapters will eventually provide an in-depth grasp of the RFF application potential inside a machine learning framework.

Chapter 3

Software and Technologies

3.1 Frameworks Used in Machine Learning

Python is a popular choice for ML, computer vision, and DNN due to its ease of use, versatility, and big developer community. For starters, Python features an easy-to-learn and read syntax, making it suitable for both novice and expert developers. This allows academics to swiftly prototype their ideas without becoming caught down in the complexity of low-level languages like as C++ or Java [56].

Python also has a robust ecosystem of libraries and tools for ML, computer vision, and DL. As seen in Figure 3.1, Python leads searches for programming languages used in AI. These libraries include pre-built, tested, and optimised functions and models that can be quickly incorporated into the development process. Popular frameworks such as TensorFlow, PyTorch, and Keras, for example, have made it easy to design and train sophisticated deep neural networks with only a few lines of code. This allows researchers to concentrate on the high-level aspects of their models rather than having to build the low-level functions themselves [56].

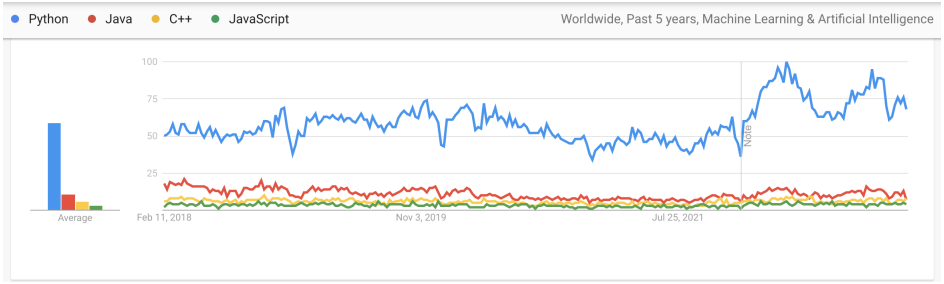


Figure 3.1: Most popular programming language for AI [57].

There are numerous DL frameworks available today, including Caffe from UC Berkeley, TensorFlow from Google, PyTorch from Facebook, and CNTK from Microsoft. TensorFlow and PyTorch are now the most popular frameworks. TensorFlow is interesting due to its active community and excellent visualisation, whereas PyTorch is popular due to its ease of programming and debugging [58]. A resumed comparison between these frameworks can be seen in table 3.1.

Table 3.1: Comparison of Deep Learning Frameworks [59, 60]

Categories	**TensorFlow**	**CNTK**	**PyTorch**	**Caffe**
Developed By	Google	Microsoft	Facebook	Berkeley AI Research
Computation Graphs	Static and Dynamic	Symbolic	Dynamic	Static
Primary Language	Python, C++	Python, C++	Python, C++	C++, Python
Mobile Support	Yes	Limited	Yes	Limited
Special Features	TensorBoard, TPUs	Easy cloud deployment	Dynamic graphs	Image focused

Before discussing the two most well-known frameworks, it is necessary to note that both Tensorflow and PyTorch support the use of GPU. In general, GPU are chosen over Central Processing Unit (CPU) for CNN because of the performance benefit they can provide. GPU acceleration was provided through the CUDA Deep Neural Network (cuDNN) package, which was released in 2014 and provided highly optimised access to typical CNN algorithms. It is vital for development because shorter testing times result from shorter training sessions.

3.1.1 PyTorch

PyTorch, which was released in 2016, is a library that achieves a balance between usability and speed by following to four key principles. The first is familiarity with Python's method of designing simple interfaces for the modules it provides. The same simplicity approach is evident in the way the complexity of ML is disguised, which serves as another principle and aids in readily identifying and applying neural network concepts. Sacrificing speed for a simpler design is another core fundamental that is allowed if the impact on performance is not significant. This allows the library to remain competitive while maintaining a simpler, more user-friendly approach. The same idea of keeping the API simple also allows for the easy addition

or modification of functionalities, which is PyTorch's last premise, and with it, the library can stay up with new breakthroughs in the field of AI. This general approach to development assures that any network design, unique or not, may be simply developed since layers or even losses are easily represented in a class structure [61].

3.1.2 Anaconda

Anaconda is a prominent open-source Python and R distribution for scientific computing and data science. Its fundamental significance in ML arises from capabilities like environment management, package management, cross-platform compatibility, integration with IDEs and notebooks, and project repeatability support. Anaconda's environment management aids in dependency management by allowing developers to build various environments with distinct package versions. Conda, its package manager, makes it easier to install multiple libraries and tools. ML models can be deployed smoothly across different operating systems thanks to the platform's cross-compatibility [62, 63].

3.1.3 Keras

Keras is a Python-based high-level neural networks API that allows for rapid experimentation and simplifies the creation and training of DL models. Its main advantages in ML are its simplicity, modularity, versatility, and interaction with TensorFlow. Keras user-friendly interface and Pythonic idioms make it easy to build sophisticated models. The API is designed in a modular manner, allowing developers to build models with pluggable and configurable modules such as layers, optimizers, and functions. Despite being a high-level tool, Keras provides surprising flexibility by allowing customisation of layers, loss functions, and optimizers, allowing complex architectures and learning approaches to be implemented [64, 65].

3.1.4 Tensorflow

TensorFlow version 1.0 was launched in 2015, with the goal of allowing any ML algorithm to be executed on a number of platforms. This method resulted in a lot of development flexibility because a number of tools that could result in the same output were made available to the user. A neural network model, for example, might be developed using Contrib, Keras, or by specifying them manually via layers. However, for a new user, the fact that there were so many alternatives to choose from caused a lot of uncertainty when they first started developing on the library. Another disadvantage of this approach was that in order to train the network, a separate function that generated a session had to be constructed. Debugging became more complex as errors had to be searched for both during definition and execution, resulting in a less user-friendly library.

When TensorFlow 2.0 was released in 2019 all of the issues listed above were addressed. The latest edition included significant changes aimed at bringing the library in line with other solutions on the market. First, many libraries were eliminated from the framework with the goal of reducing the number of alternatives a user had when constructing a model and, as a result, removing the confusion that had been created. The two-step approach of constructing and executing utilising sessions was then revised, with the execution phase now being called automatically, resulting in a much more traceable one-step building process. In addition, the library provided a more "Pythonic" manner of building neural networks through classes, similar to PyTorch, making TensorFlow more user-friendly. Another novel aspect was the model's abstraction, which enabled the same described system to be executed on a local PC or a multi-GPU server environment. Finally, by establishing a simplified pipeline, the way input data is processed has been considerably improved. Originally, dummy variables had to be constructed to feed data into the network, which would later be filled with real data during session execution. This caused confusion once more, therefore it was replaced with the TensorFlow Datasets module, which gave a better way to use and even add data.

With all of the new enhancements, TensorFlow now provides a comparable experience to PyTorch, namely a simple, beginner-friendly interface that is also efficient in terms of speed [61].

3.1.5 PyTorch And Tensorflow Comparasion

The two libraries are compared in article [61], in which the same neural network architecture was constructed, the same type of data was fed into them, and the same hardware specifications were used. Based on a pair of stereo images of the same size, one taken from a right perspective and the other from a left perspective, and both describing the same scenario, the neural model developed is intended to be used in deep estimation applications. The network produces a matrix corresponding to the size of the images describing the depth of the objects in the image.

The images used to train the model were taken from the KITTI 2015 dataset. The system was also designed in such a way that the libraries could be swapped out, allowing it to run on the same hardware without the need for another project. The following hardware specifications were provided [61]:

- CPU: AMD Ryzen 7 1700X, 3.80 GHz
- GPU: ASUS GeForce GTX 1070 Ti A8G, 8GB GDDR5
- RAM: 16 GB 3200 MHz
- STORAGE: 500GB SSD

PyTorch features a more user-friendly API, making components like models, loss functions, optimizers, and schedulers easier to comprehend and construct. TensorFlow, on the other hand, has a more complicated API but more advanced capabilities and flexibility. Both libraries included several answers to problems caused by user error, as well as detailed explanations of components and usage.

The study indicates a difference in accuracy between Tensorflow and PyTorch, with Tensorflow guaranteeing 1.16% greater accuracy [61].

However, as stated in this article [66] , accuracy and training times might vary based on the neural network models and datasets employed. Depending on the dataset, the accuracies in Tensorflow were higher than in Pytorch in some instances, while the inverse occurred in others.

Both systems are currently recognised as excellent ML frameworks. TensorFlow is a more comprehensive framework in terms of capabilities, libraries, and community, whereas PyTorch is a more simplistic tool in ongoing development. Tensorflow is a more appropriate choice for this project, however it is not mentioned that PyTorch will not be a good choice. Tensorflow is picked because it appears to have more community support and is more mature than PyTorch [67].

The existence of tensorboard was the final factor in selecting TensorFlow. TensorBoard is a visualisation tool that aids in the understanding and debugging of TensorFlow scripts. TensorBoard allows users to readily visualise a model's computation graph, training metrics, and parameter values. As illustrated in Figure 3.2, in which is illustrated the loss of the model deployed [68].

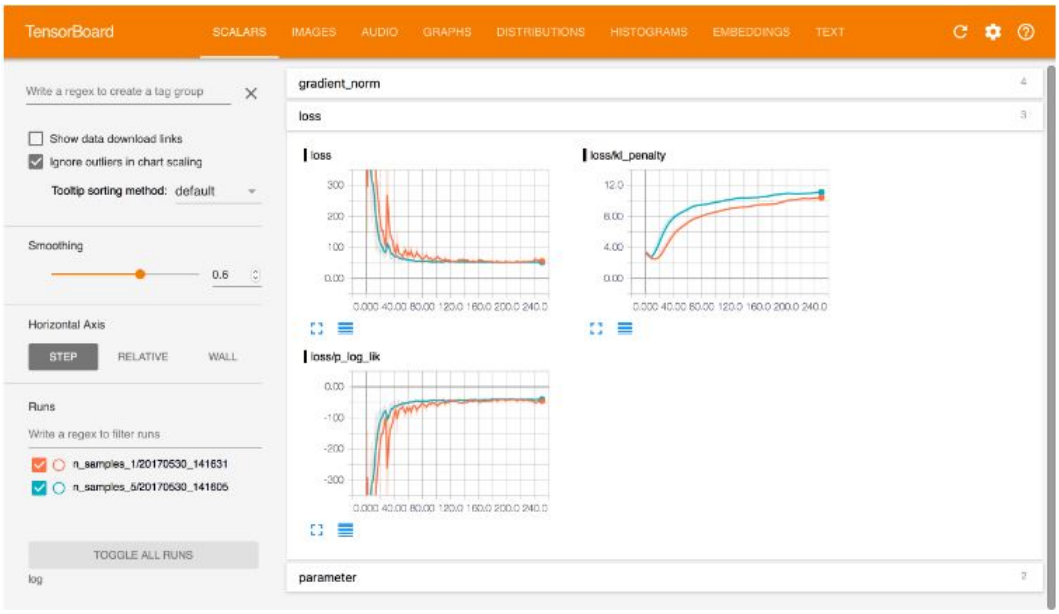


Figure 3.2: Tensorboard [68].

3.2 Popular CNN image classification models

Three CNN models are described, which resulted in major alterations to the constructions of CNN architectures as well as the obtained outcomes.

3.2.1 LeNet

LeNet, the first CNN that achieved state-of-the-art performance on hand digit identification tasks, was proposed by LeCun. LeNet's capacity to categorise digits without being influenced by minor distortions, rotation, and position and scale variations made it famous for its historical significance. The network is a feed-forward neural network with five alternating layers of convolutional and pooling, followed by two fully connected layers, as can be seen in Figure 3.3. Traditional multilayered fully connected neural networks considered each pixel as a separate input, which was computationally expensive at the time. LeNet addressed this issue by taking use of the association between neighbouring pixels and dispersing feature patterns across the entire image, using convolution with learnable parameters to extract similar features with few parameters. The use of sharable parameters by LeNet revolutionised the usual approach of training, and it was the first CNN architecture to automatically learn features from raw pixels, lowering the amount of parameters necessary [48].

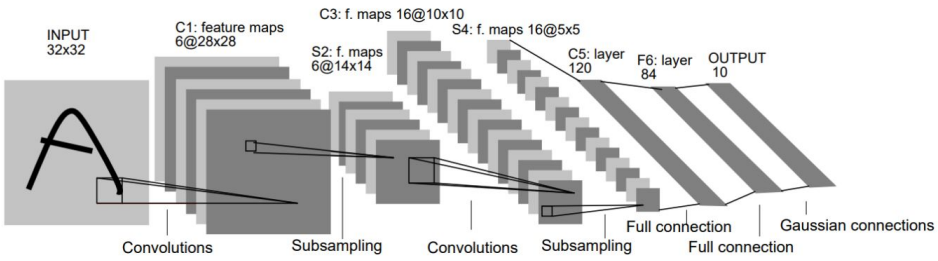


Figure 3.3: LeNet-5 architecture [69].

3.2.2 AlexNet

Although the first CNN, LeNet, performed well on hand digit recognition tests, it had limitations in classifying other sorts of images. However, AlexNet is widely regarded as the first deep CNN architecture to reach breakthrough results in image classification and recognition applications. To improve its learning capacity, AlexNet extended the depth of the CNN from 5 to 8 layers and used different parameter optimization procedures. It was trained in parallel on two NVIDIA GTX 580 GPU to overcome hardware limitations. Despite the benefits of depth in enhancing generalisation, overfitting was addressed by randomly skipping transformational units during training.

In addition, to increase convergence rate, ReLU was utilised as a non-saturating activation function, while overlapping subsampling and local response normalisation were used to prevent overfitting. Other changes included the use of larger-sized filters (11x11 and 5x5) in the initial layers. AlexNet's efficient learning approach has played a crucial role in the advancement of CNN architectures, ushering in a new era of study in this subject. Figure 3.4 depicts AlexNet's core architectural design [48].

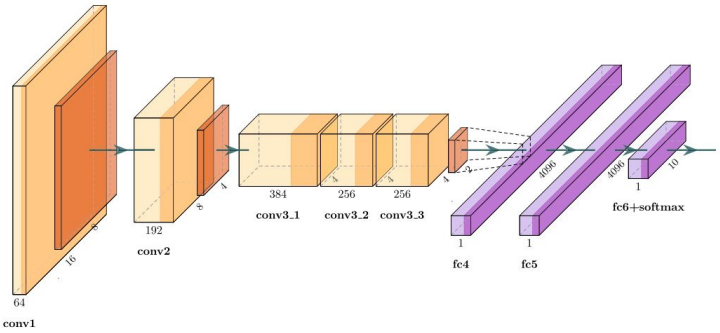


Figure 3.4: AlexNet architecture [70].

3.2.3 VGG

Simonyan et al. presented the VGG design approach for CNN architectures, which was modular in layers pattern. In comparison to AlexNet, VGG was created 19 layers deep to investigate the relationship between depth and the network's representational capabilities. VGG proved that contemporaneous placement of small filters can generate the effect of larger filters by replacing larger filters with a stack of smaller 3x3 filters. This trend of utilising smaller filters in CNN not only benefits from reduced computational cost but also minimises the number of parameters. VGG additionally controls network complexity by inserting 1x1 convolutions between convolutional layers that learn a linear mixture of feature-maps. To tune the network, max-pooling is applied after the convolutional layer, while padding maintains spatial resolution. VGG has demonstrated good results for both picture classification and localization problems; nevertheless, its fundamental disadvantage is the use of 138 million parameters, which makes it computationally expensive and difficult to deploy on low-resource platforms [48].

The next section focuses on the implemented medical gateway architecture, in which will be employed the RFF authentication for increasing the security in IoMT systems. The general design for most IoMT systems is outlined in section 3.3.

3.3 IoMT Gateway

The Open Web Application Security Project (OWASP) defines the typical components of IoMT solutions as depicted in Figure 3.5 [71]:

1. Objectives: According to the FDA, connected medical devices (also known as IoMT endpoints) are medical devices that are linked to hospital networks, the Internet, or other medical devices. For completeness, the current work examines non medical devices that can be employed in IoMT situations, such as environmental sensors.

2. Gateways: These are networking devices that help connect weak endpoints to the backend by acting as a bridge network.

3. Back-end: Current IoT systems rely on back-end servers to run the IoMT solution, as well as process and store data.

4. Mobile devices/applications: Mobile devices/applications are often used in IoT systems to provide remote control of endpoints and back-end management as well as rapid notifications.

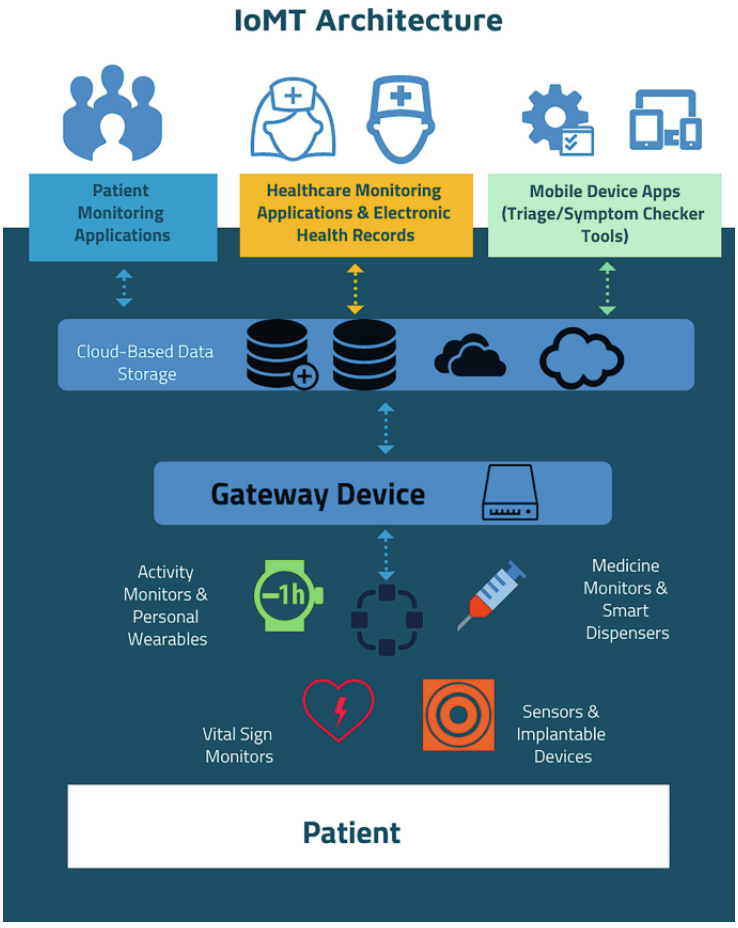


Figure 3.5: The IoMT landscape [72].

The components required in an IoMT solution vary based on the solution. As illustrated in Figure 3.5, IoMT solutions are categorised as follows [71].

- Wearable (e.g., heart monitors), implantable (e.g., embedded cardiac function monitors), ambient (e.g., door sensors), or fixed devices (e.g., computerised tomography scanners). Most current endpoints include networking capabilities, but some do not and, as a result, require a gateway.
- Platforms, which are typically cloud-based platforms designed to make smart devices and apps more accessible. IoMT solutions provide administrators with centralised back-end management features such as ecosystem administration, backup of reports and analytics, and online interfaces.
- Edge computing is used to evaluate acquired data (e.g., medical analytics) and interface with other systems via services (i.e., mobile or web applications) (e.g., accounting and insurance). IoMT solutions are typically a combination of these sorts.

The development of a medical gateway is critical in the context of IoMT because it is the primary component in achieving the establishment of IoMT in the first place. To better understand the medical gateway, consider Figure 3.6. Medical devices are responsible for collecting medical measurements from users, and the edge IoT gateway is responsible for connecting to these devices and storing the acquired data. The data is then structured in order to be delivered to the appropriate cloud services or API, where the sensitive data is debited for further study by the medical team. The medical team, in turn, visualises the data and offers information or alerts about the data it has processed; these alerts are transmitted from the cloud/API to the various gateways, allowing users or healthcare assistants to take the right action.

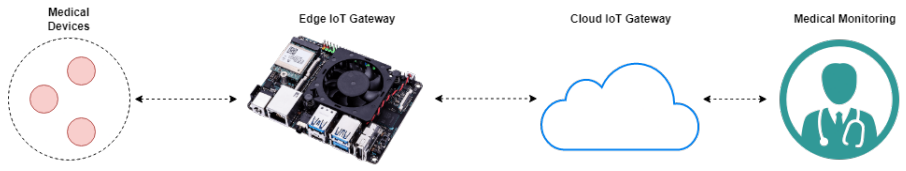


Figure 3.6: Medical Gateway.

Medical devices can now communicate via Bluetooth. Software Development Kit (SDK) that enable the integration of the device into the appropriate gateway, allowing data to be received and then delivered to cloud services, are necessary to establish communication between a medical gateway and the medical devices. To accomplish this, the gateway must support an operating system capable of running these SDK. In the case of our built gateway, the ASUS ThinkerBoard Edge R, it supports the Android operating system, which is essential for integration because

the medical devices we utilise include the Android SDK. When measurements are done, the SDK offered allow to terminate the corresponding connections. Allowing these devices to connect to the medical gateway only when measurements from the user are required.

After receiving data from the appropriate devices, these are stored in the medical gateway for two reasons: first, to allow the user to view the most recent measurements, and second, in the event that it is not possible to send medical data to cloud/API services, to be saved for when the network conditions are reunified. It will also be possible to get alerts or messages from medical equipment originating from cloud or API services, informing the user of procedures or medication dosages.

All gateway data will be received here, in relation to the cloud service. Following receipt of these data, they are formatted for Fast Healthcare Interoperability Resources (FHIR) format for interoperability purposes, allowing them to be sent to the appropriate medical equipment and API. If it is necessary to install data analysis modules, such as certain values that do not fall within the established norms for a specific user, notifications will be sent to the user or directly to medical teams in the event that an immediate response is required. FHIR is a standard designed by Health Level Seven International (HL7) for the exchange, integration, sharing, and retrieval of electronic health information. These standards aid clinical practise as well as the management, delivery, and evaluation of health care services. One of the most appealing aspects of FHIR is its interoperability with RESTful APIs, which allows it to exploit modern web-based communication mechanisms. This combination enables FHIR to ease the sharing of healthcare information over popular internet technologies such as HTTP and JSON. This means that healthcare systems that support FHIR can communicate with other health and non-health systems more readily [73].

After receiving the data sent by the appropriate cloud services, medical teams can perform the necessary analyses by sending feedback, or even the names of the medications and dosages that patients must take, allowing these teams to work from a distance, reducing the burden placed on hospital staff.

A medical gateway is divided into two sections: the edge and the cloud. The edge component, which I created, whereas the other cloud component was created by another project participant. The usage of SDK is crucial in the creation of the Edge Gateway to enable extensive interaction with the Android application. This integration was critical in obtaining and processing data from numerous medical devices, considerably improving the functioning of the programme.

These devices connected with the application via Bluetooth, ensuring a dependable, real-time data interchange. The data from the medical equipment was saved in the programme for two purposes. First, it provided users with access to their most recent health indicators, allowing them to track their health state over time.

Second, the programme was meant to retain data locally if an internet connection was absent. When the link was restored, the stored data was transmitted to the cloud, ensuring that no essential health information was lost.

OAuth2 procedures, a standard authorisation framework, were deployed as part of data security precautions. This system ensured that both user information and medical data were safely sent to the cloud. It also managed individual data access permissions, ensuring user privacy and data integrity.

A user-friendly software interface and a graphic user interface was also created. This feature allowed users to easily access previously collected data from medical equipment, presenting it in a clear manner and eliminating the possibility of data misunderstanding.

Finally, the gateway construction included a notification system to keep consumers and their healthcare professionals in continual communication. This system allowed medical teams to send users important health-related notifications such as prescription reminders and updates. This feature encouraged user participation and proactivity in health management.

Figure 3.7 depicts all of the components of the Edge Gateway development. This diagram provides a detailed overview of the system, allowing for a more in-depth understanding of its structure and functions.

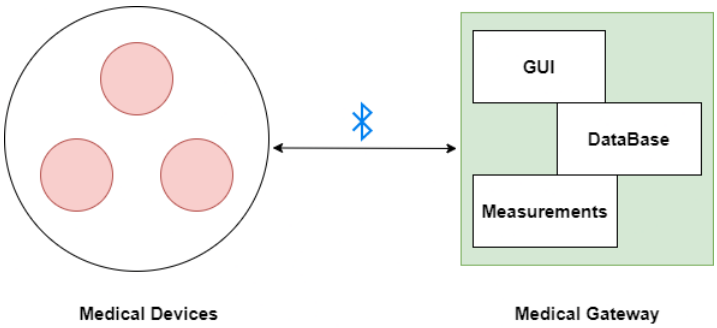


Figure 3.7: Edge Gateway.

The cloud gateway component created by another colleague matches to the full cloud environment of the established medical gateway. The cloud gateway collects health data from the Edge Gateway and performs all communication operations with external systems into which the medical gateway can be linked. The cloud environment is built on Microsoft Azure services, and communication with the edge gateway is done securely through the Azure IoT Hub platform.

After establishing a secure HTTPS connection using the POST method, the collected data can be transferred to external medical monitoring systems. Data is sent via the FHIR format, which is designed for health data exchange. The FHIR data is subsequently transferred to the external system, which should have an FHIR database, as a JSON object.

In conclusion, the creation of the Edge Gateway was a big step towards a more user-centric and interactive healthcare system. The incorporation of modern technologies served to improve healthcare service delivery while assuring secure and efficient health data administration.

3.3.1 RFF Communication Protocol

The IoMT Gateways sensitive data is protected by a strong communication protocol that uses RFF for authentication. To ensure reliability, this protocol provides numerous messages for device authentication, ongoing authentication process communication, and final conclusion communication, as well as acknowledgement responses. There are provisions in place to combat threats such as spoofing, to ensure information transfers are secure and dependable, and to protect the integrity of medical measures. Furthermore, packet monitoring, often known as "packet sniffing," is crucial in detecting unauthorised network activities. It detects unknown devices, detects duplicate MAC addresses suggestive of spoofing efforts, and analyses real-time traffic to prevent security vulnerabilities.

In practise, successful authentication comprises the secure transmission of patient health data from a medical device to a medical gateway. In the meantime, the RFF platform begins device authentication, and after successful validation, the acquired medical data is delivered to a cloud service. A failed authentication scenario, on the other hand, involves a rogue device, detected by RFF procedures and MAC Level Authentication, delivering modified data. If this device fails the authentication process, its data is ignored and is not transmitted to the cloud service. The system leverages 433MHz transceivers via the ER400TRS devices as a last line of defence. If a device fails to authenticate after three signal readings (WiFi devices), a selected ER400TRS in the gateway delivers a signal that is assessed via the RFF process; if it passes, the data is delivered to the cloud; otherwise, the system suspects an attack and prevents measurements at the medical gateway.

3.3.2 Signal Acquisition and Feature Extraction

A project colleague created an Out-Of-Tree block to satisfy the project's unique requirements including WiFi signal bursts reception from an ADALM-Pluto SDR. Beyond the capabilities of pre-existing GNU Radio blocks, the block optimises data storage by processing signal bursts and removing non-transmission intervals. Notably, the block uses a configurable cutoff technique to maintain crucial transient phenomena in signal bursts. This ensures that extensive signal attributes are captured, which is critical for creating an adequate dataset. The colleague's dataset attempted to collect RF, with a concentration on WiFi and 433 MHz signals. The

procedure involved gathering data in an office setting to simulate real-world situations such as background noise and signal interference. For signal receiving, an ADALM-Pluto SDR and a 2.4 GHz antenna were utilised, while signal emitters included ESP32 2.4GHz Dual-Mode WiFi and Bluetooth Development Boards, a Microsoft Surface Book 2 computer, and ER400TRS transmitters. GNU Radio was used to receive and handle data. For WiFi data collection, a private WLAN was established up, and devices pinged an access point while the SDR collected the signals at varied distances. The ER400TRS transceivers emitted signals captured by the SDR using a similar distance variation approach for 433 MHz transmissions. Each approach produced 500-sample datasets per device.

3.3.3 Devices

Espressif Systems created the ESP32 that can be seen in Figure 3.8, a very powerful and adaptable microcontroller. It has WiFi and Bluetooth capabilities, making it excellent for IoT applications. The chip has a dual-core CPU, a plethora of GPIO pins, and support for a number of peripheral interfaces, which increases its flexibility [74].

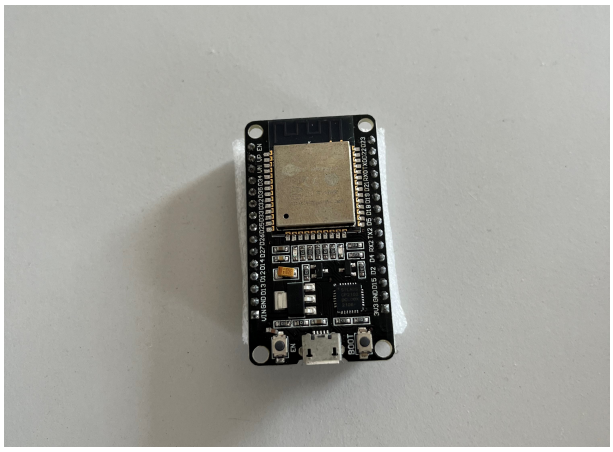


Figure 3.8: ESP32.

The easyRadio ER400TRS transceiver Figure 3.9, which incorporates 'easyRadio' technology, provides high-performance data transmission up to 250m. It boasts minimal voltage and power consumption, as well as a crystal controlled synthesiser for frequency stability. Users can configure the frequency, data rate, output power, RS232 BAUD rate, and encryption, which is supported by a 16-bit CRC [75].

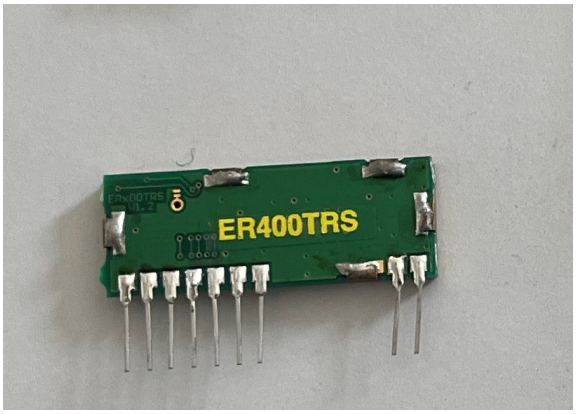


Figure 3.9: ER400TRS.

In RFF authentication operations developed in this thesis, both the ER400TRS transceiver and the ESP32 microcontroller play critical roles. Because of their programmability and powerful data transfer capabilities, they are suitable for developing secure, configurable authentication systems. These devices provide the flexibility and efficiency required to keep wireless communication networks secure and dependable.

Chapters 4 and 5 provide a thorough overview of the CNN model's development, highlighting the characteristics used for training the models, the resulting model performance, and the combining of some aspects in the device categorization for authentication reasons.

Chapter 4

RFF Machine Learning Framework

This part describes the project's implementation, beginning with a description of security RFF gateway architecture, the installation and configuration of the essential tools, moving on to the dataset, pre-processing of the dataset data, and the building and training of the models.

4.1 IoMT Security RFF Gateway Architecture

Consider the following scenario: a genuine user is operating an authorised medical device that is linked to the Edge Gateway. Simultaneously, an unauthorised entity attempts to exploit this active session by connecting a rogue device to the Edge Gateway in order to inject fake data into the system.

The authorised medical device connects to the Edge Gateway and begins communication. The Edge Gateway sends a request for authentication of this device to the PC in accordance with the defined communication protocol. In response, the PC authenticates the device, notifies the Edge Gateway that the authentication process is in progress, and eventually communicates the successful authentication result. The unauthorised entity attempts to connect its rogue device to the Edge Gateway during this ongoing authorised session. The goal is to bypass authentication and introduce unauthorised data directly. The security gateway must include se-

curity mechanisms to counter such threats, notifying the IoMT gateway that the communication was not validated.

The architecture we propose consists of a security gateway that employs RFF technology and works in conjunction with an IoMT gateway. The IoMT gateway is in charge of receiving data from medical devices such as oximeters and thermometers, as well as further analysing this data. The gateway can be divided into edge and cloud stages, with the cloud performing more demanding computation. The security RFF gateway is designed to deal with IoMT system security threats making use of RFF and ML techniques. It must capture and recognise the signals that the devices send to the IoMT Gateway. If a rogue device attempts to communicate with the gateway, the security gateway must notify the IoMT gateway that the communication was not validated. This process should take place in real time to detect security threats as soon as possible. The system should also have an active mode that can demand the device to communicate based on specific parameters for better identity detection.

The proposed RFF Security Gateway block diagram is depicted in Figure 4.1. It begins with signal capture using an SDR (ADALM-PLUTO), followed by feature extraction using SDR software (GNU Radio in this case). The first loopback, signal acquisition controller, refers to the previously described active mode concept. Following the feature extraction stage, the system proceeds to the generation of fingerprints, where the device identity should be present. This fingerprint is classified based on the device emitting the signal using ML. Finally, a second loopback is in charge of controlling the fingerprint generator for better classification results. Seeing Figure 4.1 again, there are three green blocks on which this thesis is centred, which are detailed in the rest of this chapter and chapter 5, providing for a better knowledge of how they work to generate the final output. The loopbacks depicted in the diagram have not yet been built and are regarded as future development.

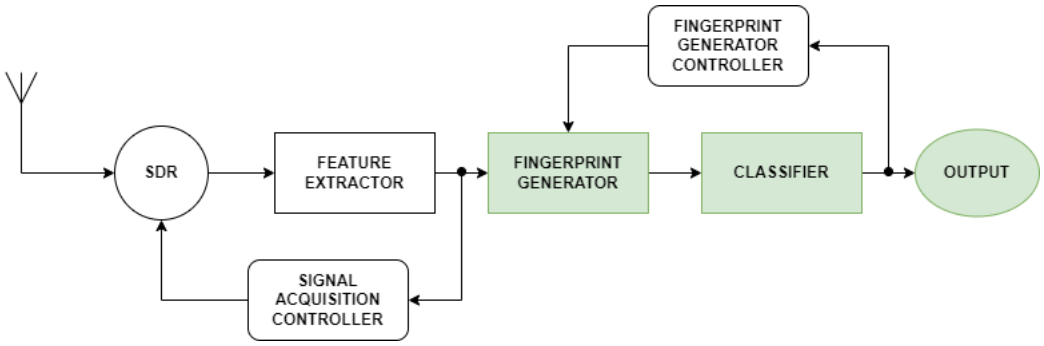


Figure 4.1: Block diagram of the proposed architecture.

Figure 4.2 depicts the process of incorporating RFF into the medical gateway. In our example, the SDR, represented by the ADALM-PLUTO, takes the lead in recording signals sent by the device attempting to authenticate. These signals are then promptly received and processed.

The computer manages the features of these signals, which are subsequently categorised using CNN models. The results of the CNN models are compared to the device specified by the medical gateway. This process provides secure exchanges and the implementation of additional security procedures, allowing for reliable authentication.

As previously described, the RFF integration procedure into the medical gateway is extremely relevant to a wide range of systems that integrate similar architectural aspects, particularly those that use a gateway for communication. The use of an SDR for signal capture, along with computer-based signal management, guarantees a dependable and dynamic solution.

Furthermore, the development of a different communication protocol strengthens the system. This protocol not only provides for easy network interactions, but it also supports secure exchanges, which is critical for system integrity.

As a result, this integrated model provides a customizable template that can be changed to strengthen security and increase efficiency in a variety of systems centred on a gateway architecture. It emphasises the potential for scalability and adaptability in similar or related sectors, hence accelerating the further implementation of such technology integrations.

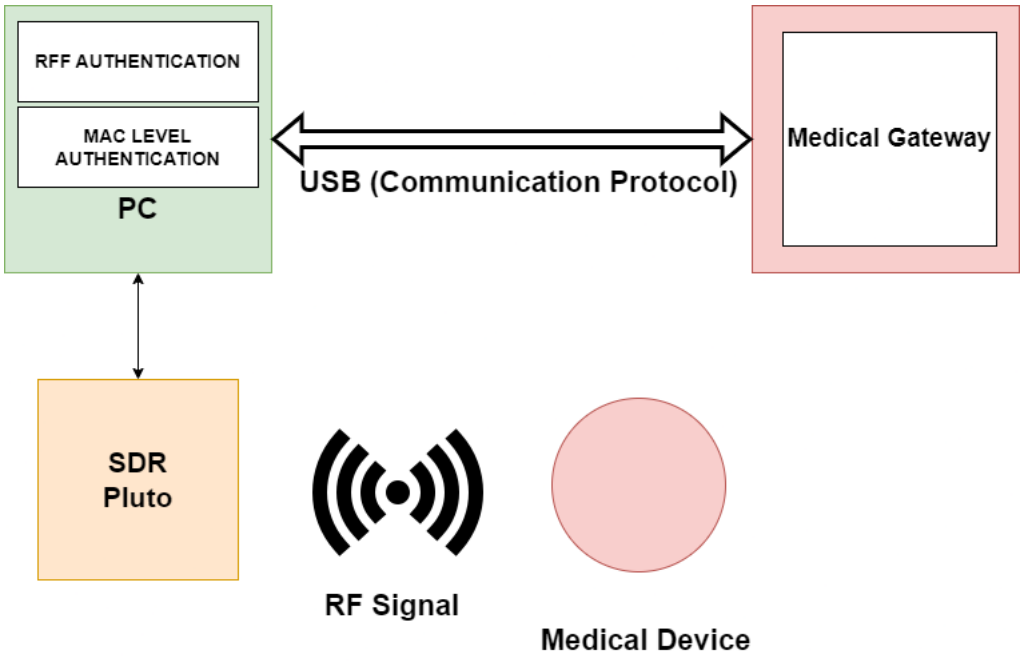


Figure 4.2: RFF Integration in Medical Gateway.

Relative to the RFF authentication module shown in Figure 4.2, it is responsible for transforming RF signals into features, allowing them to be used by CNN models for classification, in conjunction with a decision-making mechanism that combines the results of various CNN models, as it is explained in the rest of this Chapter and

Chapter 5. Looking again at Figure 4.2 the MAC level authentication module is in charge of determining whether any unauthorised or "rogue" devices are attempting to connect to the network. This is especially critical when devices try to mimic MAC addresses. When a rogue access is detected via the MAC address, the data supplied to the medical gateway is not validated.

4.2 Setup

This section overviews the many stages required to work with the ML models, such as the tools and frameworks to be installed, and essential configurations. In our work, we relied on the TensorFlow framework setup in a virtual environment.

4.2.1 Computer Setup

We had both physical computer systems and virtual machines available to deploy the ML framework. The physical machines that are available for this project are shown in the table 4.1.

In the case of the free virtual machine industry, we have Google collab and Kaggle Kernel, both from Google. The issue here is twofold: the time limit for use in each session, as well as its performance when compared to, say, the personal desktop [76] in table 4.1. Of course, the virtual machines provide significantly higher performance when compared to the laptop in the same table, but this is not the case when compared to the personal desktop.

Taking these criteria into account, it is clear that the personal desktop is the better option, as it has better graphics, a faster CPU, more RAM, and more memory, the latter of which is significant because RFF databases take up a lot of space. The final reason is that the personal desktop does not have the time usage constraint that virtual machines do.

Table 4.1: Physical Machine Comparison

	Personal Desktop Computer	**Personal Laptop**
Processor	AMD Ryzen 1600 Hexa-Core 3.2 GHz	i5-7300U dual-core processor, 3.5GHz Max Turbo
RAM	16.0 GB	8.0 GB
Disk Memory	SSD 512 GB HDD 1024 GB	SSD 256 GB
Graphics Card	NVIDIA GeForce GTX 1060 (6.0 GB dedicated)	Intel® HD Graphics 620 integrated GPU

4.2.2 Virtual Environment

Anaconda is used to eliminate various problems that may come from the PC's working environment. It is possible to establish a virtual environment/profile where the necessary packages and dependencies are allocated.

```
conda create -n tf_gpu pyhton==3.8
conda activate tf_gpu
conda install cudatoolkit=11.0 cudnn=8.0 -c=conda-forge
pip install --upgrade tensorflow-gpu==2.4.1
```

Listing 4.1: Anaconda Virtual Environment

The implementation of the virtual environment begins after installing Anaconda and accessing one of the terminals. As shown in Listing 4.1, the first line allows the development of the desired profile/virtual environment with the Python version. The second line activates this profile in any instance to activate this virtual environment. Further installations for the setup of the virtual environment, such as cuDNN, are required to use the GPU. This library is in charge of accelerating the GPU during the training, compilation, and evaluation of deep neural networks. The third command line is used for that, and the fourth command line is required to install the more reliable tensorflow version with the cuDNN version utilised [77].

The first line of listing 4.2 lists all of the physical devices available, specifically the GPU. The if statement determines whether any physical devices are available. If there are, the code loops through each physical device and uses the set virtual device configuration function to set the memory limit for the virtual device configuration to 4 GB. This code is useful because it activates the GPU to be used for the code; it was configured with 4GB of memory because it was the configuration that permitted higher system stability for all model training.

```
physical_devices = tf.config.list_physical_devices('GPU')
if physical_devices:
    for device in physical_devices:
        tf.config.experimental.set_virtual_device_configuration(
    device, [tf.config.experimental.VirtualDeviceConfiguration(
    memory_limit=4000*1024*1024)])
```

Listing 4.2: Set GPU memory limit

4.3 Dataset

Since the availability of datasets for this technology is limited, the dataset developed by another project colleague was used, mostly for the ESP32 devices used in the WiFi communication protocol and the ER400TRS devices that interact in the 433 MHz region. In this case, his work focuses on minimizing the captures of RF signals as much as possible so that they are only, or as reduced as feasible to present the signal sample that concerns only actual messages provided by the device. The dataset consists of eight devices that have been categorised according to their inherent characteristics. Four devices are categorised as belonging to the WiFi protocol category, and the remaining four are categorised as belonging to the 433 MHz frequency band. Three ESP32 devices in the WiFi category are identical, and the fourth device is a Surface Book 2. The four ER400TRS devices in the 433 MHz category are also all the same model. An average of 500 files per class are present on each device in the dataset.

4.4 Features

The RFF approach provided in this thesis focuses on translating the electromagnetic waves transmitted by the devices into images that can then be employed in CNN models, a quick description of how these image transformations are achieved follows.

Constellation

The constellation images can aid in the analysis of minor variations in signal behaviour that can be used as RFF features. The constellation plots can indicate unique patterns or anomalies in the signal that can be ascribed to a specific transmitter by viewing the real (I) and imaginary (Q) components of the complex-valued data points. These distinct patterns can be utilised as input features for ML algorithms, allowing devices to be identified and classified based on their RFF.

Listing 4.3 code reads complex-valued IQ data files and generates a constellation plot to view the data. A constellation plot is a scatter plot of the complex-valued data points' real (I) and imaginary (Q) components.

```
1 data = np.fromfile(filename, np.complex64)
2 plt.rcParams["figure.figsize"] = (100,100)
3 plt.plot(np.real(data), np.imag(data), '.')
4 plt.grid(True, which='both'))
```

Listing 4.3: Constellation Plot

Figure 4.3 shows an example created by the code, in which the real and imaginary components of the IQ file is plotted, giving origin to an image capable of being used for ML.

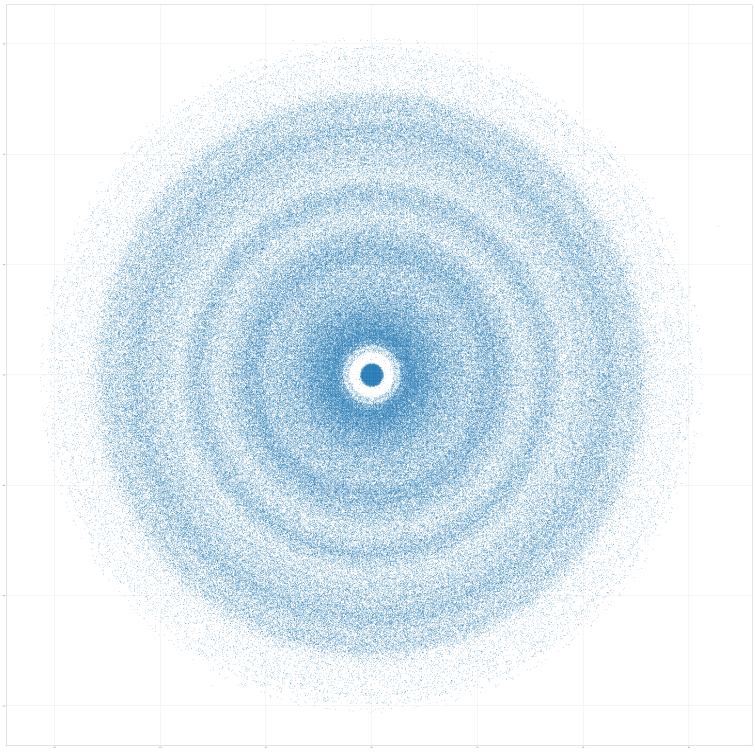


Figure 4.3: Constellation of an ESP32 device.

Amplitude

The amplitude-time images can aid in the analysis of subtle variations in the amplitude of the signal over time, which can be used as features for RFF. These plots show the amplitude fluctuations of the signal, displaying unique patterns or anomalies that can be linked to certain devices or transmitters.

Listing 4.4 contains code that loads complex-valued IQ data, calculates signal amplitude, and graphs amplitude over time; an example of this plot is shown in figure 4.4.

```
1 data = np.fromfile(f, np.complex64)
2 signal = np.abs(dat.real + 1j * dat.imag)
3 plt.rcParams["figure.figsize"] = (100,100)
4 plt.plot(signal)
```

Listing 4.4: Amplitude Plot

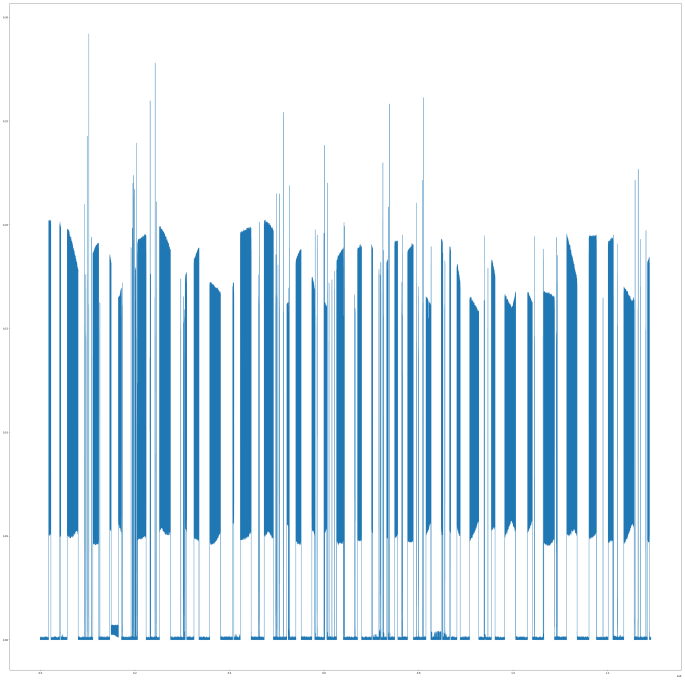


Figure 4.4: Amplitude of an ESP32 device.

Power Spectral Density

The PSD is an element for RFF since it reveals the unique properties of each wireless devices by exposing the distribution of the signal's strength across a range of frequencies.

Listing 4.5 provides an effective approach for computing and displaying the PSD of RF signals, which is an important aspect for RFF.

```
data = np.fromfile(filename, np.complex64)
PSD = np.abs(np.fft.fft(data))**2 / (N*Fs)
PSD_log = 10.0*np.log10(PSD)
PSD_shifted = np.fft.fftshift(PSD_log)
f += center_freq
plt.rcParams["figure.figsize"] = (100, 100)
plt.plot(f, PSD_shifted)
```

Listing 4.5: Power Spectral Density Plot

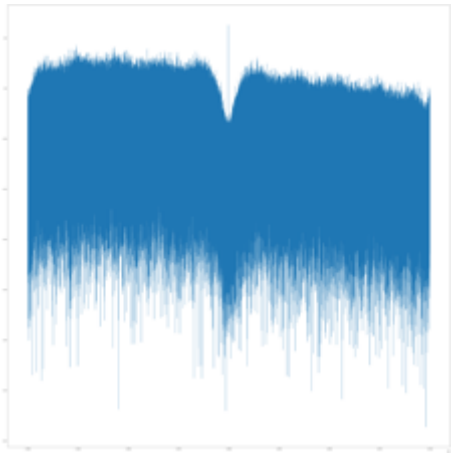


Figure 4.5: PSD of an ESP32 device.

Differential Constellation

Differential constellations are also used in CNN models [78]. In Listing 4.6, the complex data is loaded via the IQ file, then the differential signal is calculated by multiplying each sample with the complex conjugate of the next sample, followed by a scatter plot of the real and imaginary parts of the differential signal, resulting in the image shown in Figure 4.6.

```
I_data = data.real
Q_data = data.imag
differential_signal = []
for i in range(len(I_data) - n):
    X_t = I_data[i] + 1j * Q_data[i]
    X_tn = I_data[i+n] + 1j * Q_data[i+n]
    D_t = X_t * np.conj(X_tn)
    differential_signal.append(D_t)
plt.rcParams["figure.figsize"] = (100,100)
plt.scatter(np.real(differential_signal), np.imag(
    differential_signal))
```

Listing 4.6: Differential Constellation Plot

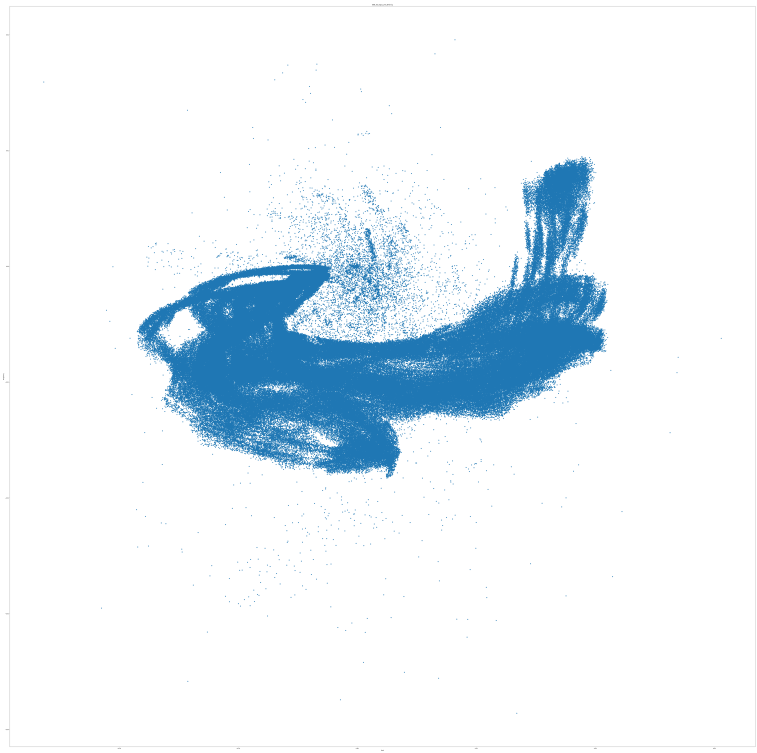


Figure 4.6: Differential Constellation of an ESP32 device.

Spectrograms

Spectrograms can provide useful insights into a wireless signal's frequency content and modulation properties, which can aid in the process of creating and evaluating RFF. The code in Listing 4.7 reads the complex number and then plots the spectrogram, resulting in the image shown in Figure 4.7.

```
1 data = np.fromfile(f, np.complex64)
2 plt.specgram(data, NFFT=1024, Fs=2000000)
```

Listing 4.7: Spectrograms Plot

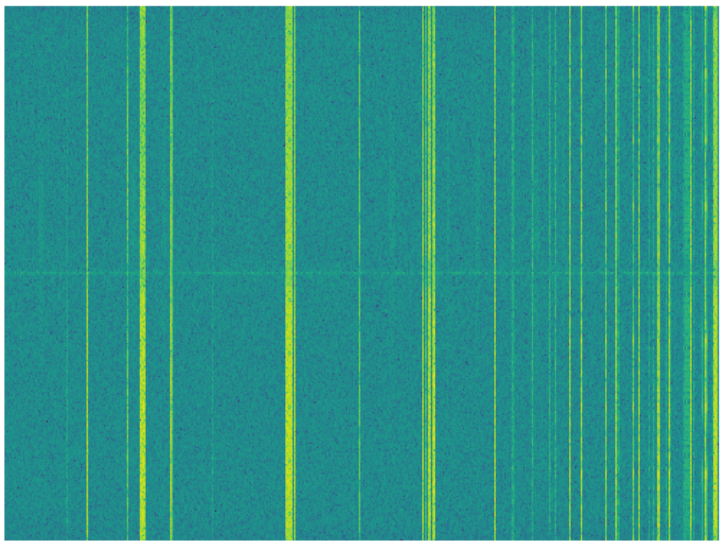


Figure 4.7: Spectrogram of an ESP32 device.

To determine whether the above listed features are appropriate for accurate characterization of the real devices within each protocol, their effectiveness and performance will be assessed. In order to distinguish between various devices, this evaluation aims to determine how reliable and sturdy the features are. The objective is to determine which aspects for each protocol perform best by examining how well they perform. Additionally, the research will show the potential of the ADALM-PLUTO device for integration in RFF systems and evaluate its capacity to handle these kinds of signals.

4.4.1 Data augmentation

This section describes the data augmentation method used to enhance the CNN model's performance. In the realm of DL, data augmentation is a widely used technique, especially for image-based tasks because it helps improve the diversity and size of the training dataset. By applying various changes to the input images, it is possible to generate new training examples that aid in the learning of more generalized features by the CNN model. This approach also helps mitigate the risk of overfitting [79].
The technique for implementing data augmentation is described below, and it is also shown in listing 4.8:

1. Using the Python Imaging Library (PIL) function "Image.open," the input image file is first opened.

2. Use the 'resize()' function to resize the image to the appropriate dimensions (224 x 224). The CNN model will always get images with the same input shape thanks to this standardisation.

3. Three new augmented images are produced for each image by rotating the existing one 90 degrees three times. This is accomplished by executing the 'rotate()' function with an angle of '90 * (i+1)' degrees while iterating through a range of three (i=0,1,2). As a result, the original image is produced in three rotated variants at 90, 180, and 270 degrees.

4. Using the 'save()' function, the rotated photos are then saved with a new file name that contains the augmentation index.

```
1 im = Image.open(file_name)
2 im = im.resize((224, 224))
3
4 for i in range(3):
5     rotated_im = im.rotate(90 * (i+1))
6     save_path = os.path.splitext(file_name)[0] + f'_augmented_{i}.
      png'
7     rotated_im.save(save_path)
```

Listing 4.8: Data augmentation

This augmentation caused each feature to have 2000 images per device, which is four times more than it did previously. This kind of rotation is employed to make sure that there is no cut in the images generated.

4.4.2 Data processing

Before feeding the images into the CNN model, the images must be pre-processed to verify that they are all formatted in the same way. This phase should be conducted to both the training and validation data to ensure that they are equivalent.

Several standards have been maintained for the various types of images in order to aid implementation:

- The various sorts of images linked with a feature type are segregated into different directories, resulting in improved data organisation. The images are kept in a separate directory with the names of the devices they refer to in each features directory.

- It is assured that the images are loaded with the same height and length (224 by 224).

In listing 4.9, it is possible to verify that in this case is performed for features of WiFi devices, is iterated by the different class labels of the devices, and loaded the images from each class subdirectory within the "basedata", the only thing to change if it was for the ER400TRS devices would be the name of the labels. These are

then scaled to 224x224 pixels and transformed to numpy arrays. Divide the pixel values by 255 to normalise them, and save each image with its matching label in a list called data.

```
for label in ["esp1", "esp2", "espap", "surf"]:
    for image_path in os.listdir(f"basedata/{label}/"):
        img = load_img(f"basedata/{label}/{image_path}",
    target_size=(224,224))
        img = img_to_array(img) / 255
        data.append([img, label])
```

Listing 4.9: Loading and preprocessing images

4.4.3 Division of data

The dataset in Listing 4.10 is first treated to random shuffling to guarantee an equal distribution of data points between the training and validation sets. The split index is computed using 80% of the training set of the dataset. The data is then split into train data and validation data using slicing. The input features (X) and target labels (Y) for each of these subsets are extracted and saved individually in the files X train, y train, X val, and y val.

To map string labels to their associated integer values, a label map dictionary is built. The string labels on the y train and y val lists are then changed to numbers using the label map. The integer labels in y train and y val are then one-hot encoded using TensorFlow's tf.keras.utils.tocategorical function. ML models frequently employ one-hot encoding to represent categorical variables because it makes it simpler for the algorithm to distinguish between distinct categories.

```
np.random.shuffle(data)

split_index = int(len(data) * 0.80)
train_data = data[:split_index]
validation_data = data[split_index:]

X_train = np.array([i[0] for i in train_data])
y_train = [i[1] for i in train_data]
X_val = np.array([i[0] for i in validation_data])
y_val = [i[1] for i in validation_data]

label_map = {"esp1": 0, "esp2": 1, "espap": 2, "surf":3}
y_train = [label_map[i] for i in y_train]
y_val = [label_map[i] for i in y_val]

y_train = tf.keras.utils.to_categorical(y_train)
```

```
y_val = tf.keras.utils.to_categorical(y_val)
```

Listing 4.10: Division of data

4.4.4 Model

As shown in Figure 4.8, the CNN used for image classification has five convolutional, three max-pooling, and three fully connected layers. The need to successfully acquire complex features and patterns from input images while managing computational constraints influenced the choice to use this architecture. The colour channels are represented by pictures with dimensions of 224x224x3. (RGB).

The model is made up of five convolutional layers, the first, second, and fifth of which are followed by a max-pooling layer, which reduces dimensionality and improves robustness. Following feature extraction, the model utilises fully connected layers to learn non-linear feature combinations and construct a more abstract representation. An output layer with the softmax activation function produces probability scores for each class to complete the classification.

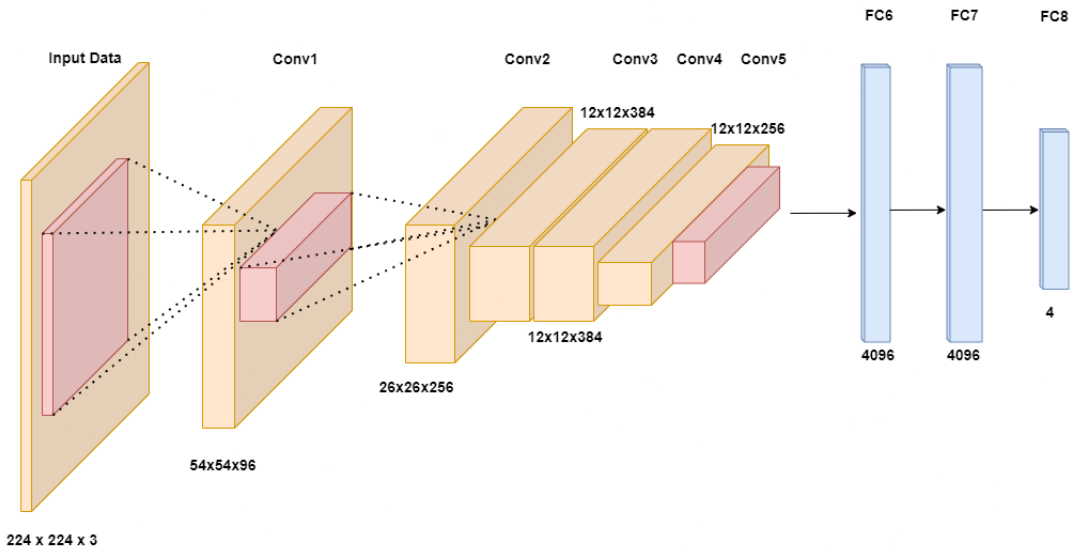


Figure 4.8: Diagram of model used.

This model was built using the design described in article [26]. This model is built on AlexNet, with the following distinctions:

- In this model, the image input size is 224x224x3, whereas in the AlexNet model, the image input size is 227x227x3.
- This algorithm has four output layers compared to Alexnet's thousand. The model only has four devices because only four devices were categorised in each of the WiFi and ER400TRS devices.

- In the end, 8 experiments were performed in which the filters of the convolutional layers were changed, and the impact of the dropout (fixed at the rate of 20%) was evaluated for each of these changes; the differences in filters and dropout are listed in table 4.2, with the goal of determining if this is an improvement and using the better experiment.

Table 4.2: Experiences model

Experiment	Filters (Conv Layers)	Dropout
A	96-256-384-384-256	No
B	96-256-384-384-256	Yes
C	48-128-192-192-128	No
D	48-128-192-192-128	Yes
E	144-384-576-576-384	No
F	144-384-576-576-384	Yes
G	192-512-768-768-512	No
H	192-512-768-768-512	Yes

4.4.5 Training

The features stated earlier in the chapter are trained on the model using the changes listed in table 4.2.

Hyperopt

The code in Listing 4.11 makes use of the python library hyperopt [80], which is used to optimise hyperparameters, in this case to find the optimal learning rate. A learning rate search space is specified. After determining the optimal learning rate, the model is recompiled and trained again using the training data. The optimal learning rate is used during the training process to improve the model's performance on validation data, eventually improving the overall model quality.

```
1 from hyperopt import fmin, tpe, hp, STATUS_OK
2
3 def objective(learning_rate):
4
```

```
    model.compile(optimizer=Adam(lr=learning_rate), loss='
    categorical_crossentropy', metrics=['accuracy'])

    model_fit = model.fit(X_train, y_train, epochs=45, batch_size
    =64, validation_data=(X_val, y_val))
    val_loss = model_fit.history['val_loss'][-1]
    val_acc = model_fit.history['val_accuracy'][-1]

    return {'loss': val_loss, 'accuracy': val_acc, 'status':
    STATUS_OK}

search_space = hp.uniform("learning_rate", 0.00001,0.00005)

best = fmin(objective, search_space, algo=tpe.suggest, max_evals
    =30, rstate=np.random.default_rng(123))
```

Listing 4.11: Hyperopt

Epochs

Based on preliminary testing, which demonstrated that exceeding a particular number of epoches resulted in overfitting, a fixed epoch limit was specified to prevent overfitting in CNN models and their corresponding features. The epoch limit for Constellation is 45, 30 for Diferential Constellation, 10 for Amplitude, 15 for PSD, and 20 for Spectogram. Models are effectively taught to spot patterns using this technique without becoming unduly specialised to the training set, enhancing generalizability and performance.

Model Compilation and Training

Before training the model, it was necessary to configure the metrics, optimizer, and loss function through the model compilation process. Due to the use of the softmax activation function in the last layer of the model, the categorical_crossentropy loss function was chosen. The Adam optimizer and accuracy metric were used, as can be seen in Listing 4.12.

```
model.compile(optimizer=Adam(lr=best['learning_rate']), loss='
    categorical_crossentropy', metrics=['accuracy'])
```

Listing 4.12: Model Compile

In Listing 4.13 the model was trained using the fit() function, incorporating the optimal learning rate found using Hyperopt, as well as other hyperparameters like the number of epochs and batch size. The validation data (validation set and corresponding labels), and the training data (training set with associated labels)

were also specified during the training process. This approach ensures a robust and well-trained model that can effectively generalize to new data.

```
model_fit = model.fit(X_train, y_train, epochs=10, batch_size=64,
    validation_data=(X_val, y_val))
```

Listing 4.13: Model Fit

4.4.6 Saving Model

Each training was saved in a file that specified which experiment it was after training and validation. The Keras load_model() function was used for later use of these models, particularly considering their performance. Additionally, measures such as the confusion matrix, accuracy, loss, precision, recall, and F-score were used to justify which of the experiments produced better results to be chosen as a model for the RFF process.

Chapter 5

ML Model Assessment and Classification Results

This chapter carefully examines the performance of several RFF features. Finding the effects of custom blocks on each RFF feature and selecting the most successful experiment for each situation are the main goals of this investigation. The advantages provided by the custom block are then investigated, indicating the optimal experiment for each RFF feature.

5.1 Class Labels

Categorization is an important element of ML, as it involves predicting class tags from supplied data. Classifiers effectively assign class tags to new samples by studying and digesting the data, allowing for intelligent decision making. We consider the following, as shown in Figure 5.1.

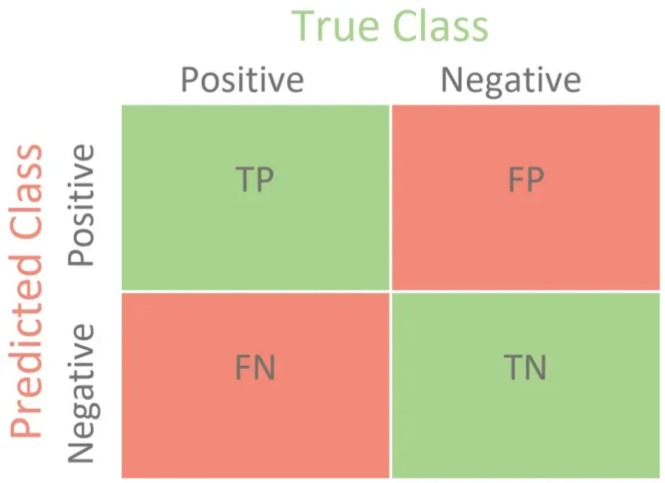


Figure 5.1: Confusion Matrix [81].

Some key concepts can be extracted from the confusion matrix, as shown in Figure 5.1 [82]:

- True Positive (TP): the number of instances that were correctly accepted, predicted, or classed as positive.
- True Negative (TN): the number of cases that were accurately rejected or forecasted as negative.
- False Positive (FP): the number of positive events that were wrongly accepted or predicted/classified.
- False Negative (FN): the number of cases that were rejected or predicted/classified as negative wrongly.

5.2 Evaluation Metrics

There are numerous performance indicators available for comparing and evaluating CNN models. However, which metric is best for the problem must be considered. Given that there are numerous measures for measuring categorization performance, here are several examples, according to [83]:

- Average Accuracy:
$$\frac{\sum_{i=1}^{l} \frac{TP_i+TN_i}{TP_i+FP_i+FN_i+TN_i}}{l}$$
- Precision μ(Micro-average Precision). The micro-average precision, which is derived from the sums of images decisions, assesses the average agreement of the data class labels with those of a classifier. The total number of true

positives (TPi) divided by the total number of false negatives (FNi) across all classes:

$$\frac{\sum_{i=1}^{l} TP_i}{\sum_{i=1}^{l}(TP_i + FN_i)}$$

- Recall µ(micro-average), which is determined from the sums of individual image judgements, assesses the average ability of a classifier to identify class labels. The total number of true positives (TPi) divided by the total number of false positives (FPi) across all classes.

$$\frac{\sum_{i=1}^{l} TP_i}{\sum_{i=1}^{l}(TP_i + FP_i)}$$

- The link between the positive labels assigned by a classifier based on the sums of per-image judgements and those given by the data is measured by the micro-average F-score. With a weighting factor of 2, the F-score is a weighted harmonic mean between Precision µand Recall µ.

$$\frac{(\beta^2 + 1) \times Precision_\mu \times Recall_\mu}{(\beta^2 \times Precision_\mu + Recall_\mu)}$$

- Precision M (Macro-average Precision) assesses the average per-class agreement between the data class labels and the labels assigned by a classifier. It divides the true positives (TPi) by the total of true positives (TPi) and false negatives (FNi) for each class Ci. After that, the total number of classes (l) is divided by the sum of all classes.

$$\frac{\sum_{i=1}^{l} \frac{TP_i}{TP_i + FN_i}}{l}$$

- Recall M (Macro-average Recall) assesses how well a classifier performs on average for each class in identifying class labels. It divides the true positives (TPi) by the total of true positives (TPi) and false positives (FPi) for each class Ci. After that, the total number of classes (l) is divided by the sum of all classes.

$$\frac{\sum_{i=1}^{l} \frac{TP_i}{TP_i + FP_i}}{l}$$

- Fscore M (Macro-average F-score): The macro-average F-score evaluates the correspondence between the positive labels provided by the data and the classifier's per-class averaged labels. With a weighting factor of 2, the F-score is a weighted harmonic mean between Precision M and Recall M.

$$\frac{(\beta^2 + 1) \times Precision_M \times Recall_M}{(\beta^2 \times Precision_M + Recall_M)}$$

5.3 Assessment of the Experiences

Before delving into the model results, there are a few things to consider. First, the hyperparameters used to obtain the results for each feature. Following that, the metrics for the values of the experiments for each feature.

Validation accuracy, validation loss, micro-average of precision, recall, and F1-score, as well as macro-average of precision, recall, and F1-score, are all included in the tables. It should be noted that in a multiclass task, the micro-average values and the validation accuracy value produce the same results [84]. The best model for each table will be represented in bold, to make it easier to visualize.

For each feature, the confusion matrix of the best model, as well their as graphs of training and validation loss, will be presented. The custom block, carefully created by a project team member, represents a significant advancement in the field of radio frequency signal processing. Its primary goal is to significantly minimise the noise associated with these signals, therefore improving the purity of the collected data. The custom block ensures a higher level of signal capture accuracy by successfully isolating and recognising bursts or signals legitimately originating from the device.

5.3.1 Models performance with the custom block for ER400TRS devices

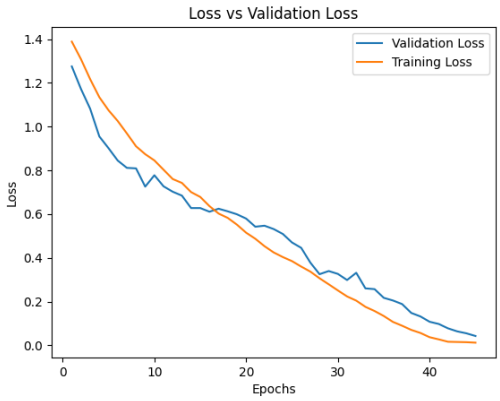
Constellation

In terms of hyperparameters, a batch size of 64 was employed, as well as a learning rate of 0.0003, acquired by using the hyperopt library. A detailed review of table 5.1 reveals that model H, which is equipped with a greater number of filters, is the most effective model for providing good performance for the constellation feature in the ER400TRS devices.

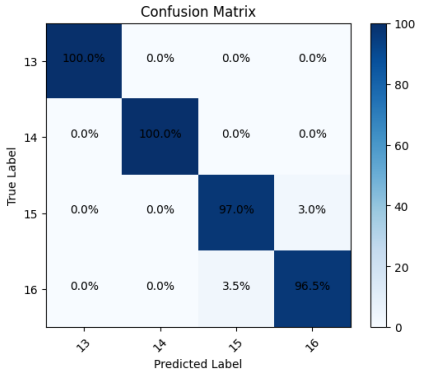
Table 5.1: Constellation with custom block

Experience	Accuracy	Loss	Precision µ	Recall µ	F-Score µ	Precision M	Recall M	F-Score M
A	96,25%	0,0812	96,25%	96,25%	96,25%	95,00%	94,66%	96,65%
B	97,15%	0,0956	97,15%	97,15%	97,15%	97,56%	98,27%	97,55%
C	94,77%	0,1358	94,77%	94,77%	94,77%	94,90%	94,76%	94,84%
D	96,92%	0,0904	96,92%	96,92%	96,92%	97,01%	96,92%	96,97%
E	95,70%	0,1520	95,70%	95,70%	95,70%	96,31%	95,70%	96,02%
F	95,07%	0,0777	95,07%	95,07%	95,07%	95,39%	95,07%	95,23%
G	95,45%	0,1356	95,45%	95,45%	95,45%	95,64%	95,45%	95,55%
H	**98,37%**	**0,0430**	**98,37%**	**98,37%**	**98,37%**	**98,38%**	**98,37%**	**98,37%**

Figure 5.2 depicts the confusion matrix, which demonstrates that the model cannot distinguish between devices 15 and 16 as well as it does between devices 13 and 14 from the ER400TRS devices, a trend that is maintained in the other features. The training and validation losses are decreasing and have values that are close to one another, indicating that the model is learning well.



(a) Loss in constellation model H.



(b) Confusion matrix for constellation model H.

Figure 5.2: Loss and confusion matrix for constellation model H.

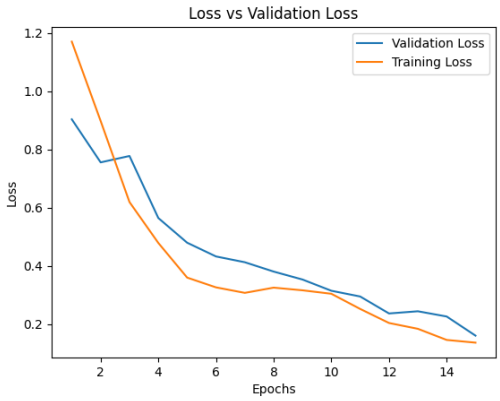
Power Spectrum Density

In terms of hyperparameters, a batch size of 16 was employed, as well as a learning rate of 0.0002, acquired by using the hyperopt library. Following a thorough examination, it is obvious that model E exhibits beneficial proficiency, particularly in regard to the PSD feature, as shown in table 5.2.

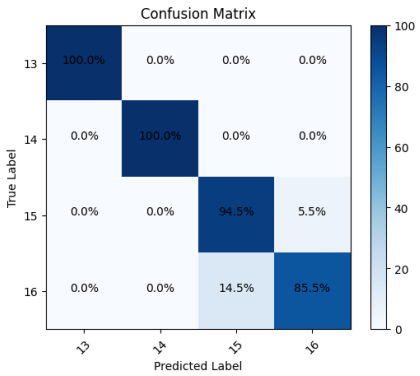
Table 5.2: PSD with custom block

Experience	Accuracy	Loss	Precision μ	Recall μ	F-Score μ	Precision M	Recall M	F-Score M
A	88,25%	0,2260	88,25%	88,25%	88,25%	88,26%	88,25%	88,25%
B	88,81%	0,3283	88,81%	88,81%	88,81%	89,14%	88,87%	89,01%
C	90,08%	0,2436	90,08%	90,08%	90,08%	90,12%	90,07%	90,06%
D	90,73%	0,1735	90,73%	90,73%	90,73%	90,74%	90,72%	90,73%
E	**95,00%**	**0,1608**	**95,00%**	**95,00%**	**95,00%**	**95,16%**	**95,00%**	**94,99%**
F	88,40%	0,3642	88,40%	88,40%	88,40%	92,08%	88,40%	87,74%
G	87,55%	0,3386	87,55%	87,55%	87,55%	89,80%	87,55%	88,66%
H	91,12%	0,2990	91,12%	91,12%	91,12%	91,14%	91,12%	91,13%

As shown in figure 5.3, the PSD feature has more difficulty distinguishing between devices 15 and 16. The model is learning as the training and validation losses decrease.



(a) Loss in PSD model E.



(b) Confusion matrix for PSD model E.

Figure 5.3: Loss and confusion matrix for PSD model E.

Spectrogram

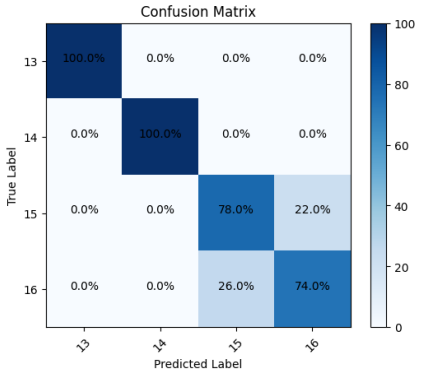
In terms of hyperparameters, a batch size of 16 was employed, as well as a learning rate of 0.0004, acquired by using the hyperopt library. As demonstrated in table 5.3, the spectrogram feature benefits from a greater number of filters because model H produced better results and has a smaller loss when compared to the other models.

Table 5.3: Spectrogram with custom block

Experience	Accuracy	Loss	Precision µ	Recall µ	F-Score µ	Precision M	Recall M	F-Score M
A	83,22%	0,2843	83,22%	83,22%	83,22%	84,44%	83,22%	83,82%
B	83,60%	0,3150	83,60%	83,60%	83,60%	84,26%	83,60%	83,93%
C	81,35%	0,3046	81,35%	81,35%	81,35%	81,53%	81,35%	81,44%
D	86,37%	0,3034	86,37%	86,37%	86,37%	89,34%	86,37%	87,83%
E	82,90%	0,2705	82,90%	82,90%	82,90%	82,94%	82,90%	82,92%
F	81,38%	0,2966	81,38%	81,38%	81,38%	82,44%	81,29%	81,36%
G	83,57%	0,3082	83,57%	83,57%	83,57%	86,64%	83,30%	84,94%
H	**88,00%**	**0,2480**	**88,00%**	**88,00%**	**88,00%**	**88,02%**	**88,00%**	**88,01%**

Figure 5.4 confusion matrix show that the CNN model has even more difficulty distinguishing between devices 15 and 16 of the ER400TRS devices in this feature than it does in the constellation and PSD features.

(a) Loss in spectrogram model H.



(b) Confusion matrix for spectogram model H.

Figure 5.4: Loss and confusion matrix for spectogram model H.

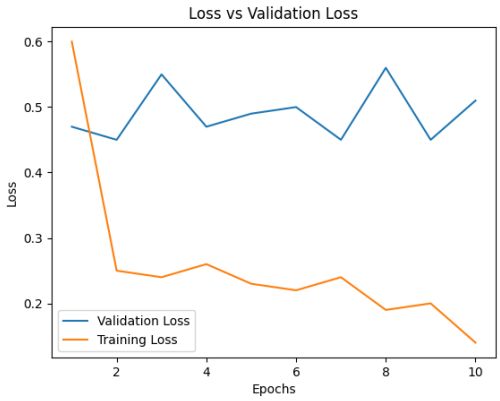
Amplitude

In terms of hyperparameters, a batch size of 8 was employed, as well as a learning rate of 0.0001, acquired by using the hyperopt library. Table 5.4 demonstrates that, despite the fact that the model performed better in G, which has a bigger number of filters, the loss values remain relatively high across the models, showing that the amplitude feature may not be as well matched as the other three previously stated features.

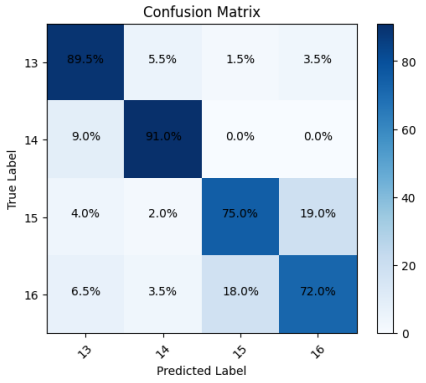
Table 5.4: Amplitude with custom block

Experience	Accuracy	Loss	Precision μ	Recall μ	F-Score μ	Precision M	Recall M	F-Score M
A	83,00%	0,5359	83,00%	83,00%	83,00%	83,54%	83,00%	83,26%
B	78,95%	0,5764	78,95%	78,95%	78,95%	79,35%	78,95%	79,15%
C	69,24%	0,6724	69,24%	69,24%	69,24%	70,73%	69,24%	69,33%
D	73,13%	0,5647	73,13%	73,13%	73,13%	74,42%	73,14%	73,77%
E	76,23%	0,6780	76,23%	76,23%	76,23%	76,37%	76,20%	76,29%
F	78,98%	0,9549	78,98%	78,98%	78,98%	79,63%	78,97%	79,30%
G	**80,62%**	**0,5137**	**80,62%**	**80,62%**	**80,62%**	**80,44%**	**80,62%**	**80,63%**
H	80,22%	0,7328	80,22%	80,22%	80,22%	80,12%	80,22%	80,17%

As illustrated in Figure 5.5, the model now begins to have greater difficulty distinguishing between devices 13 and 14, which did not occur in the previous features. The training and validation loss demonstrates the CNN model's inability to learn well this feature.



(a) Loss in amplitude model G.



(b) Confusion matrix for amplitude model G.

Figure 5.5: Loss and confusion matrix for amplitude model G.

Differential Constellation

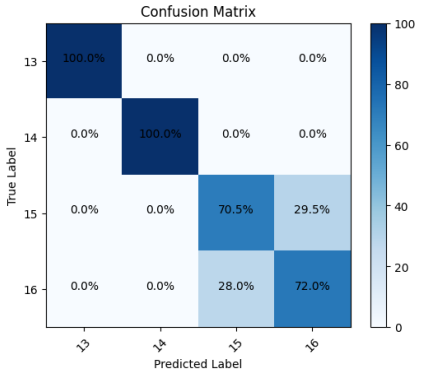
In terms of hyperparameters, a batch size of 64 was employed, as well as a learning rate of 0.0005, acquired by using the hyperopt library. According to table 5.5, model F performs the best for the differential constellation in the ER400TRS devices.

Table 5.5: Differential Constellation with custom block

Experience	Accuracy	Loss	Precision μ	Recall μ	F-Score μ	Precision M	Recall M	F-Score M
A	79,27%	0,3805	79,27%	79,27%	79,27%	79,27%	79,28%	79,28%
B	70,62%	0,5079	70,62%	70,62%	70,62%	71,64%	71,62%	71,12%
C	76,05%	0,4603	76,05%	76,05%	76,05%	76,14%	76,05%	76,09%
D	75,90%	0,4272	75,90%	75,90%	75,90%	74,24%	75,90%	75,06%
E	76,02%	0,4477	76,02%	76,02%	76,02%	76,11%	76,02%	76,06%
F	**85,58%**	**0,3986**	**85,58%**	**85,58%**	**85,58%**	**85,58%**	**85,57%**	**87,57%**
G	79,44%	0,4877	79,44%	79,44%	79,44%	78,62%	79,43%	79,03%
H	75,65%	0,4218	75,65%	75,65%	75,65%	76,21%	75,65%	75,93%

Figure 5.6 depicts how the CNN model struggles to distinguish between ER400TRS devices 15 and 16. The model is learning as the training and validation losses decrease.

(a) Loss in differential constellation model F.



(b) Confusion matrix for differential constellation model F.

Figure 5.6: Loss and confusion matrix for differential constellation model F.

Performance Improvement in ER400TRS Devices

The block has helped to improve the performance of different system functionalities by efficiently reducing noise interference. Figure 5.7 indicates that significant accuracy gains in Constellation (1.61%), PSD (1.25%), Spectrograms (2.5%), Amplitude (1.62%), and, most importantly, Differential Constellation (7.11%) have been recorded.

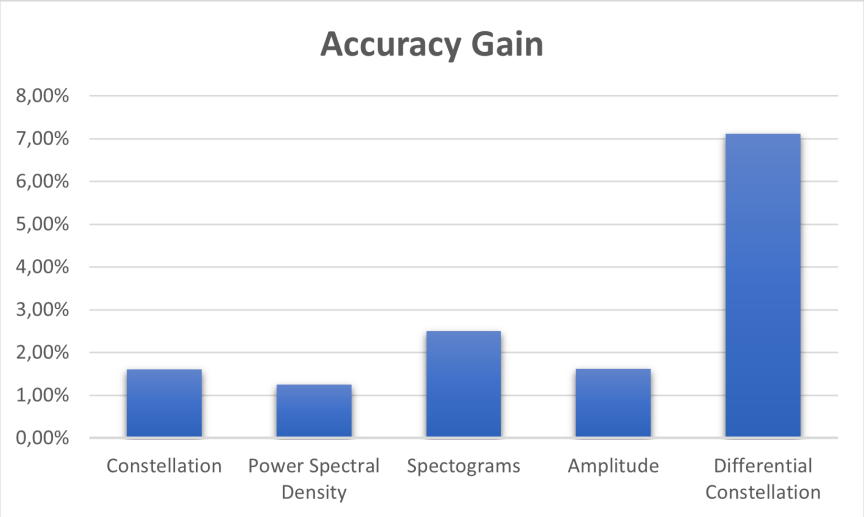


Figure 5.7: Accuracy Gain in each feature.

After reviewing Figure 5.8 the block has considerably contributed to the drop in loss values for most features, highlighting its significance in optimizing the system's performance. Furthermore, the Amplitude feature's performance emphasises the need of selecting appropriate features for RFF systems to ensure optimal performance.

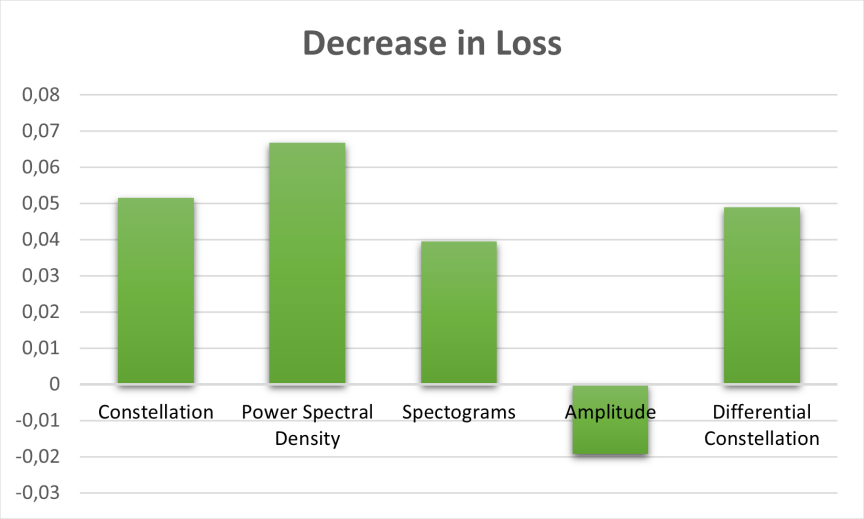


Figure 5.8: Decrease of loss in each feature.

5.3.2 Models performance with the custom block for WiFi devices

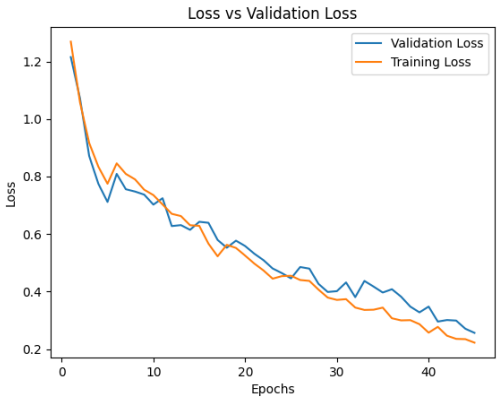
Constellation

In terms of hyperparameters, a batch size of 64 was employed, as well as a learning rate of 0.0003, acquired by using the hyperopt library. As can be seen in Table 5.6, all models have comparable performance characteristics; nonetheless, there are clear differences. Notably, the models with higher filter specifications, notably models D and E, show an increased rate of losses, implying a decrease in efficiency. Model B appears as the most capable of all the models examined.

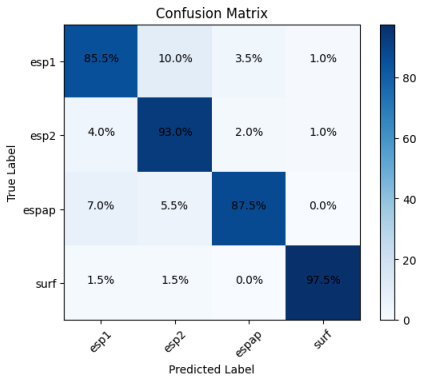
Table 5.6: Constellation with custom block

Experience	Accuracy	Loss	Precision μ	Recall μ	F-Score μ	Precision M	Recall M	F-Score M
A	89,67 %	0,2947	89,67%	89,67%	89,67%	89,67%	89,62%	89,65%
B	**90,87%**	**0,2566**	**90,87%**	**90,87%**	**90,87%**	**91,08%**	**90,88%**	**90,89%**
C	86,10%	0,2785	86,10%	86,10%	86,10%	87,86%	86,10%	86,15%
D	85,92%	0,3654	85,92%	85,92%	85,92%	86,63%	85,92%	86,27%
E	88,87%	0,3511	88,87%	88,87%	88,87%	88,98%	88,98%	88,93%
F	90,22%	0,2654	90,22%	90,22%	90,22%	90,46%	90,22%	90,34%
G	90,18%	0,2980	90,18%	90,18%	90,18%	90,15%	90,18%	90,14%
H	90,32%	0,3004	90,32%	90,32%	90,32%	90,54%	90,32%	90,43%

The confusion matrix in Figure 5.9 reveals that the surface device has greater rates than the ESP32 devices, which makes reasonable given that they are different devices. The training and validation losses indicate that the model is learning well because the values are lowering and close to one another.



(a) Loss in constellation model B.



(b) Confusion matrix for constellation model B.

Figure 5.9: Loss and confusion matrix for constellation model B.

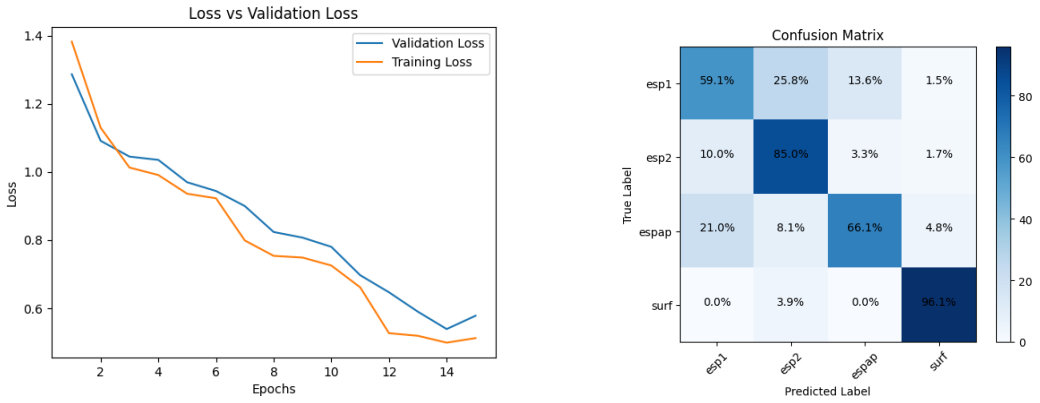
Power Spectrum Density

In terms of hyperparameters, a batch size of 64 was employed, as well as a learning rate of 0.0005, acquired by using the hyperopt library. A careful analysis of Table 5.7 reveals that model C is the most proficient of its counterparts. Surprisingly, a common feature shared by all of the examined models is a high rate of loss. This high frequency of losses highlights the difficulties that the CNN model faces when learning this specific feature of the WiFi spectrum.

Table 5.7: PSD with custom block

Experience	Accuracy	Loss	Precision μ	Recall μ	F-Score μ	Precision M	Recall M	F-Score M
A	71,71%	0,6908	71,71%	71,71%	71,71%	72,81%	71,71%	72,25%
B	71,61%	0,6844	71,61%	71,61%	71,61%	72,87%	71,61%	72,70%
C	**76,58%**	**0,5781**	**76,58%**	**76,58%**	**76,58%**	**76,69%**	**76,57%**	**76,62%**
D	67,90%	0,7239	67,90%	67,90%	67,90%	69,40%	67,90%	68,64%
E	72,65%	0,5918	72,65%	72,65%	72,65%	72,85%	72,65%	72,75%
F	71,21%	0,7383	71,21%	71,21%	71,21%	73,56%	71,20%	72,36%
G	64,65%	0,8832	64,65%	64,65%	64,65%	66,77%	64,65%	64,98%
H	70,85%	0,7951	70,85%	70,85%	70,85%	75,57%	70,85%	73,13%

Figure 5.10 shows that it is difficult to identify between ESP32 devices in this feature, showing that the feature has significant restrictions in this protocol. Despite the difficulties, the model appears to be learning, as the training and validation losses are decreasing.



(a) Loss in PSD model C.

(b) Confusion matrix for PSD model C.

Figure 5.10: Loss and confusion matrix for PSD model C.

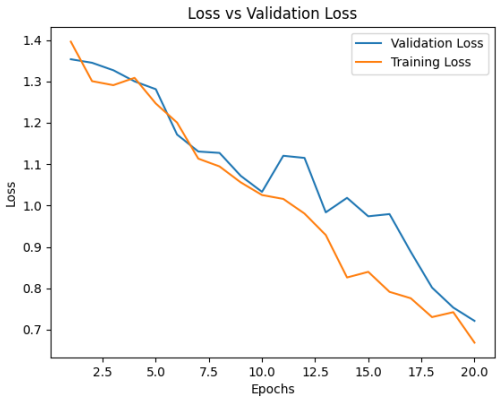
Spectrogram

In terms of hyperparameters, a batch size of 16 was employed, as well as a learning rate of 0.0004, acquired by using the hyperopt library. Table 5.8 shows a clear visual illustration of the prevalence of significant losses across all models. Even the best-performing model, model B, is susceptible to this trend, with a somewhat high degree of loss. This consistency highlights a common challenge that all models appear to confront during the optimisation process.

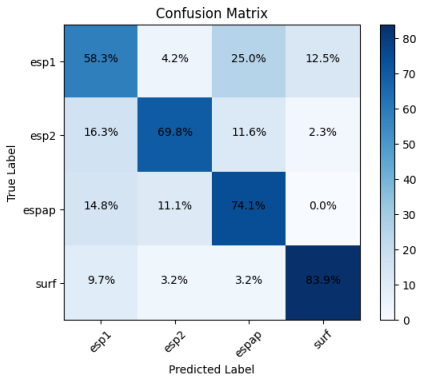
Table 5.8: Spectogram with custom block

Experience	Accuracy	Loss	Precision µ	Recall µ	F-Score µ	Precision M	Recall M	F-Score M
A	66,07%	0,8139	66,07%	66,07%	66,07%	70,25%	66,07%	68,10%
B	**71,53%**	**0,7213**	**71,53%**	**71,53%**	**71,53%**	**71,99%**	**71,52%**	**71,66%**
C	66,17%	0,7107	66,17%	66,17%	66,17%	64,16%	66,17%	65,15%
D	61,66%	0,7722	61,66%	61,66%	61,66%	61,37%	61,66%	61,51%
E	68,14%	0,8184	68,14%	68,14%	68,14%	68,63%	68,14%	68,38%
F	66,74%	0,8308	66,74%	66,74%	66,74%	67,28%	66,74%	67,01%
G	66,54%	0,8536	66,54%	66,54%	66,54%	68,98%	66,53%	67,73%
H	64,96%	0,8079	64,96%	64,96%	64,96%	63,44%	64,96%	64,19%

A deeper examination of the confusion matrix in Figure 5.11 reveals that, the feature causes difficulty in categorising identical devices. The training and validation loss, especially the validation loss shows some signs of overfit.



(a) Loss in spectrogram model B.



(b) Confusion matrix for spectrogram model B.

Figure 5.11: Loss and confusion matrix for spectrogram model B.

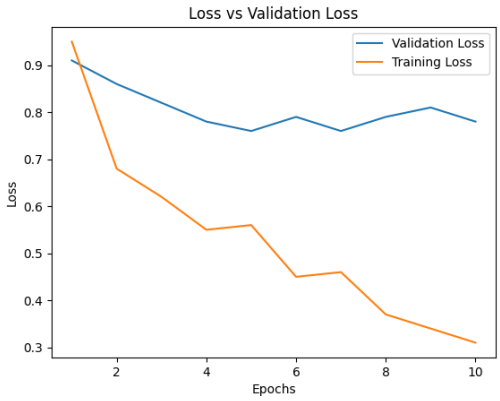
Amplitude

In terms of hyperparameters, a batch size of 8 was employed, as well as a learning rate of 0.0001, acquired by using the hyperopt library. Table 5.9 reveals a repeating pattern in which all models connected with the amplitude feature register large losses. This pattern shows that the amplitude feature may be ineffective for device categorization during the RFF procedure.

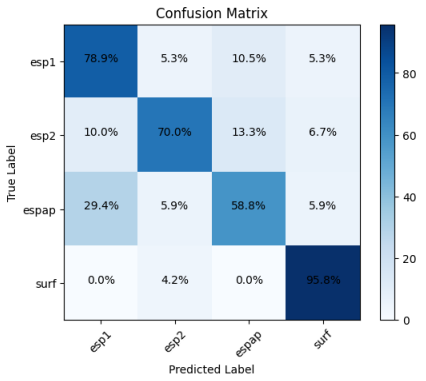
Table 5.9: Amplitude with custom block

Experience	Accuracy	Loss	Precision μ	Recall μ	F-Score μ	Precision M	Recall M	F-Score M
A	**74,90%**	**0,7790**	**74,90%**	**74,90%**	**74,90%**	**76,03%**	**75,88%**	**75,46%**
B	72,50%	0,7901	72,50%	72,50%	72,50%	75,16%	72,50%	73,80%
C	70,82%	0,8065	70,82%	70,82%	70,82%	68,59%	70,82%	69,69%
D	67,44%	0,8411	67,44%	67,44%	67,44%	70,13%	67,44%	68,76%
E	71,21%	0,6648	71,21%	71,21%	71,21%	71,57%	71,21%	71,39%
F	70,15%	0,7530	70,15%	70,15%	70,15%	73,49%	70,15%	71,78%
G	67,83%	0,7751	67,83%	67,83%	67,83%	69,15%	67,83%	68,48%
H	72,52%	0,7535	72,52%	72,52%	72,52%	73,24%	72,57%	72,90%

Visualising Figure 5.12 demonstrates that the system has difficulty distinguishing the categorization between the various devices, suggesting that this feature is not the best for classifying the devices.



(a) Loss in amplitude model A.



(b) Confusion matrix for amplitude model A.

Figure 5.12: Loss and confusion matrix for amplitude model A.

Differential Constellation

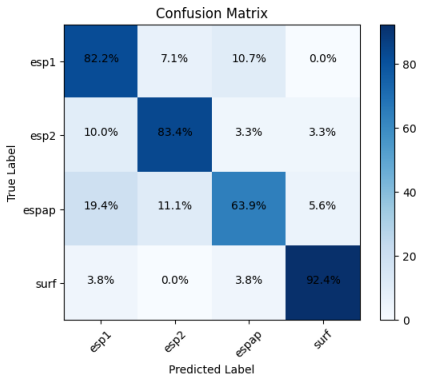
In terms of hyperparameters, a batch size of 64 was employed, as well as a learning rate of 0.0005, acquired by using the hyperopt library. Table 5.10 shows a clear pattern in which models with lower filters display more losses than models with higher filters. This clearly demonstrates the importance of filter size in the performance of these models. Model H is the best performer among all of the models tested.

Table 5.10: Differential constellation with custom block

Experience	Accuracy	Loss	Precision μ	Recall μ	F-Score μ	Precision M	Recall M	F-Score M
A	73,96%	0,5951	73,96%	73,96%	73,96%	74,12%	73,96%	74,04%
B	76,56%	0,5367	76,56%	76,56%	76,56%	75,86%	76,55%	76,21%
C	70,77%	0,7126	70,77%	70,77%	70,77%	70,86%	71,56%	71,22%
D	70,35%	0,7961	70,35%	70,35%	70,35%	69,73%	69,51%	69,62%
E	79,53%	0,5952	79,53%	79,53%	79,53%	80,26%	79,53%	79,89%
F	79,38%	0,6959	79,38%	79,38%	79,38%	79,61%	79,38%	79,50%
G	71,70%	0,5832	71,70%	71,70%	71,70%	73,14%	71,70%	72,41%
H	**80,47%**	**0,5127**	**80,47%**	**80,47%**	**80,47%**	**80,69%**	**80,47%**	**80,30%**

In Figure 5.13 it is possible to visualise that the differential constellation is capable of identifying devices better than the PSD, amplitude and spectrogram feature.

(a) Loss in differential constellation model H.



(b) Confusion matrix for differential constellation model H.

Figure 5.13: Loss and confusion matrix for differential constellation model H.

Performance Improvement in WiFi Devices

In Figure 5.14, the block has effectively contributed to large accuracy gains in many system aspects, emphasising its importance in performance optimisation. Notably, accuracy increases in Constellation (1.87%), PSD (4.74%), Spectrograms (8.58%), and Differential Constellation (6.05%) have been recorded. However, the drop in accuracy for the Amplitude characteristic (-1.42%) indicates that it may not be the best option for RFF.

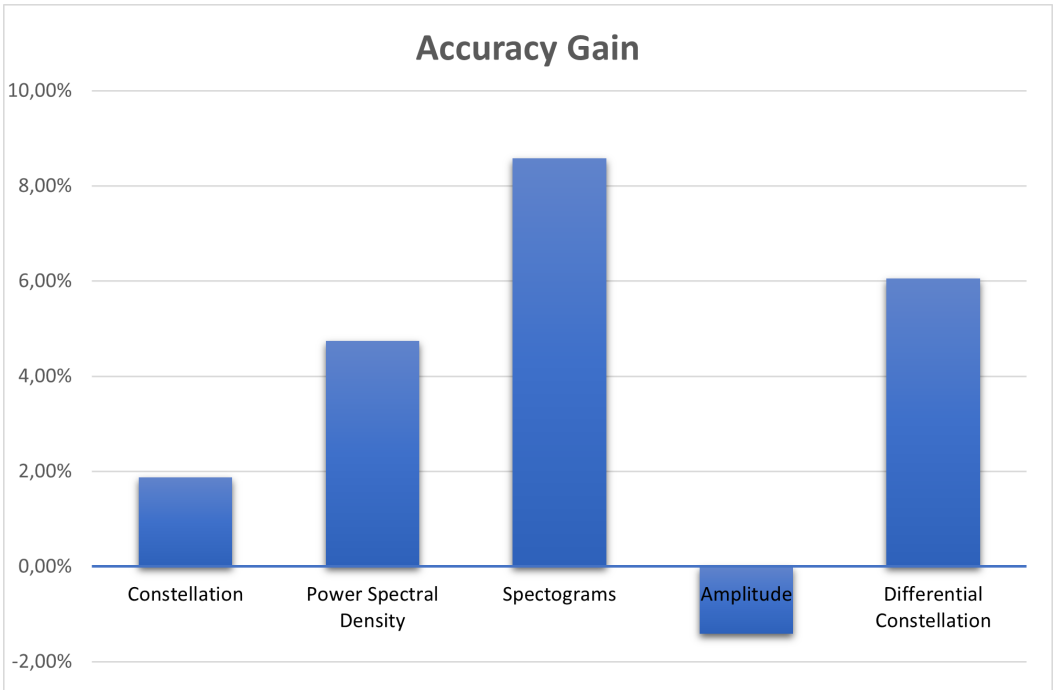


Figure 5.14: Accuracy Gain in each feature.

Figure 5.15 shows that the loss has decreased for Constellation (0.0801), PSD (0.1653), Spectrograms (0.1712), Amplitude (0.197), and Differential Constellation (0.1484).

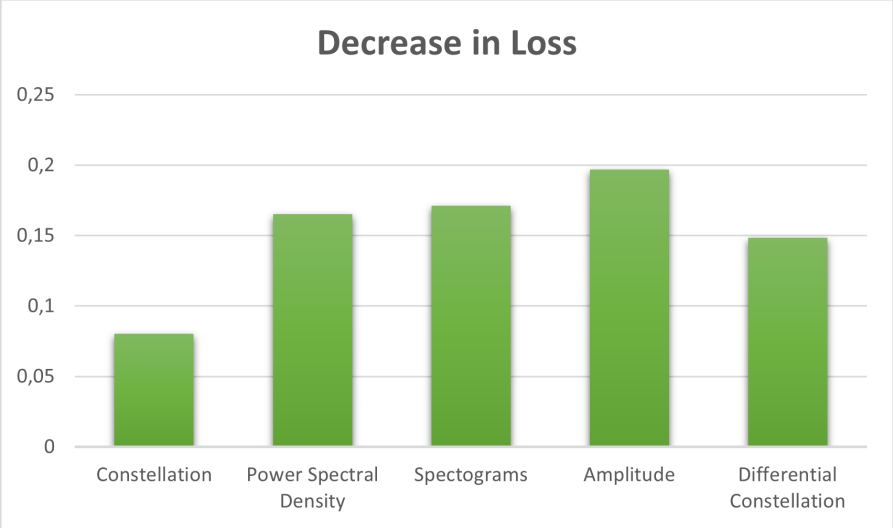


Figure 5.15: Decrease of loss in each feature.

5.3.3 Balancing Feature Complexity with Accuracy in Models

A comparison of the complexity of feature generation - in terms of time - and the accompanying validation accuracy findings was performed to determine any potential association between the two. Several tests were performed throughout the development of these functionalities. Notably, the average time for constellation was 6.38 seconds, whereas the average time for amplitude was roughly 27.82 seconds. The spectrogram took around 12.12 seconds, the differential constellation took about 11.87 seconds, and the PSD took about 9.12 seconds. Visual examination of the ER400TRS results revealed that the constellation feature produced the best accurate results while using the least amount of time for image production. PSD came in second place in terms of accuracy while also retaining a reasonable image generating time. Spectrogram and the differential constellation were then used, with comparable results and mean feature creation durations. Despite requiring the longest mean time for feature development, amplitude produced the least stunning outcomes. Specifically for WiFi devices, the constellation delivered improved outcomes despite being the shortest in terms of feature building time. The differential constellation followed quickly behind. These findings imply that there is an inverse relationship between feature complexity and the time required to obtain superior outcomes on the CNN. This suggests that signal quality may deteriorate as the feature's complexity increases.

5.4 Decision Making

It is important to highlight a procedure that plays a role in the decision-making process for the final architecture and communication protocol with regard to the device to authenticate. Multiple features in both ER400TRS and WiFi devices are investigated, and their effectiveness is assessed for the accurate identification of the device. Given that several features perform well, there is interest in employing these features in combination for device authentication. It is vital to set up a method that permits the features with superior performance to have a more substantial weight in the final selection because each of the features has a varied proportion of validations. In order to authenticate the device, Weighted Majority Voting is used as the final decision-making method.

The weighted majority vote is a technique used in ensemble classification to produce a final conclusion by combining different classifiers. The classifiers in the ensemble are given varying weights based on their individual accuracies in this method, giving more competent classifiers more influence in the final choice. The weighted majority vote concept can be mathematically stated as follows [85]:

$$d_{i,j} = \begin{cases} 1, & \text{if } D_i \text{ labels x in } \omega_j, \\ 0, & \text{otherwise.} \end{cases} \tag{5.1}$$

The discriminant function for class ω_j obtained through weighted voting is given by [85]:

$$g_j(x) = \sum_{i=1}^{L} b_i d_{i,j} \tag{5.2}$$

Where b_i is a classifier D_i coefficient. The discriminant function's value (3.2) is the sum of the coefficients for ensemble members whose output for x is ω_j.

5.4.1 Classification

The features that will be employed are decided upon after selecting the joining process. In order to do this, CNN models with validation accuracy higher than 80% are picked. This decision was made in an effort to reduce the likelihood that the final system will contain errors, as higher validation tends to be associated with better results. The CNN model produces extremely unstable results with the amplitude feature, so it is not regarded as a feature capable of being used for authentication by RFF. The amplitude feature will not be placed in both the Weighted Majority Voting of ER400TRS and WiFi devices.

The following Pseudo Code 1 shows the logic to implement the weighted majority in the case of the WiFi classification:

Algorithm 1 Weighted Classification

constellation ← *Y*
differential_constellation ← *X*
predictions ← [*y_pred_constellation*, *y_pred_differential_constellation*]
weights ← [*constellation*, *differential_constellation*]
Normalize weights: *weights* ← *weights*/*sum(weights)*
Identify unique class labels in predictions array: *unique_classes* ← *unique(predictions)*
Initialize weighted votes: *weighted_votes* ← *zeros(len(unique_classes))*
Construct a mapping from class labels to indices: *class_to_index* ← {*class_label: index* | *index, class_label* ∈ *enumerate(unique_classes)*}
for each *prediction* in *predictions* **do**
    weighted_votes[class_to_index[prediction]] ← *weights[i]*
end for
Identify class with highest vote in weighted votes: *y_pred_weighted* ← *unique_classes[argmax(weighted_votes)]*
Print out *y_pred_weighted*

For ER400TRS devices Weighted Majority Voting the models chosen are the ones trained after the noise reduction block, as can be seen in subsection 5.3.1 of chapter 5, where in the constellation the H model is used, for the PSD the E model, for the spectogram the H model is used and for the differential constellation the F model is used.

The code shown in Listing 5.1 demonstrates the way weighted majority voting is used in ensemble classification. 4 models are present in the case of ER400TRS devices, as was already mentioned. These classifiers (models) are given a weight based on each one's accuracy. This illustrates the notion that, as stated in 5.4, classifiers with higher levels of expertise ought to be given more consideration when making a choice. After that, the weights are normalised to guarantee that the sum of all weights equals 1, suggesting that each weight now indicates the relative importance of each classifier. The voting process is then carried out, in which each classifier casts a vote for the anticipated class, with the weight of the vote being determined by the significance of the classifier.

```
constellation = 0.9837
psd = 0.95
spectogram = 0.88
differential_constellation = 0.8858

predictions = np.array([y_pred_constellation, y_pred_psd,
    y_pred_spectogram, y_pred_differential_constellation])
weights = np.array([constellation, psd, spectogram,
    differential_constellation])
weights = weights / weights.sum()
```

```
9
10 unique_classes = np.unique(predictions)
11 weighted_votes = np.zeros(len(unique_classes), dtype=np.float64)
12
13 class_to_index = {class_label: index for index, class_label in
       enumerate(unique_classes)}
14
15 for i, pred in enumerate(predictions):
16     weighted_votes[class_to_index[pred]] += weights[i]
17
18 y_pred_weighted = unique_classes[np.argmax(weighted_votes)]
19 print(f"Weighted classification: {y_pred_weighted}")
```

Listing 5.1: Weighted Majority Voting 433

Using the same methodology for WiFi devices, only models that score above 80% are picked, and since there are fewer features in this case, model B is selected for the constellation and model H for the differential constellation, resulting in the code shown in Listing 5.2.

```
1 constellation = 0.9087
2 differential_constellation = 0.8047
3
4 predictions = np.array([y_pred_constellation,
      y_pred_differential_constellation])
5 weights = np.array([constellation, differential_constellation])
6 weights = weights / weights.sum()
7
8 unique_classes = np.unique(predictions)
9 weighted_votes = np.zeros(len(unique_classes), dtype=np.float64)
10
11 class_to_index = {class_label: index for index, class_label in
       enumerate(unique_classes)}
12
13 for i, pred in enumerate(predictions):
14     weighted_votes[class_to_index[pred]] += weights[i]
15
16 y_pred_weighted = unique_classes[np.argmax(weighted_votes)]
17 print(f"Weighted classification: {y_pred_weighted}")
```

Listing 5.2: Weighted Majority Voting WiFi

5.4.2 Execution Time Evaluation

A CNN model's success is strongly dependent on its execution time, specifically how rapidly it can load and classify input. Minimising the time it takes a CNN to load and analyse information can make a big impact in real-time decision making. This test

analyses the execution time in two classification cases: WiFi devices and ER400TRS devices, each with over 40 separate tests. The times for each execution refer to the process between loading the model and the respective feature, classification, and then using the weighted majority to obtain the final result.

As shown in Figure 5.16, the decision making in ER400TRS devices included four CNN models, each of which was related to a feature. We discovered a mean execution time of 7.21 seconds after analysing over 40 experiments.

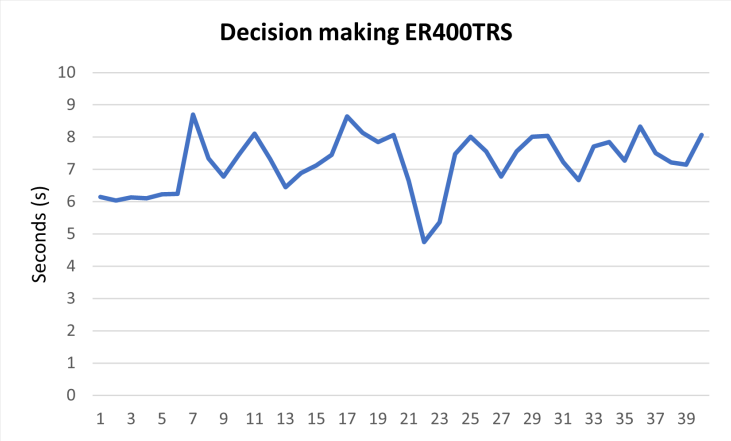


Figure 5.16: Execution time in ER400TRS decision making.

In WiFi devices Figure 5.17, as opposed to ER400TRS devices, decision making involved two unique CNN models and the same amount of experiments. These tests offered useful information on execution times in a different arrangement than the first example. Notably, the average execution time across all tests was 3.82 seconds, which was much shorter than in the first case. This reduction was ascribed to the adoption of fewer CNN models, which presumably reduced overall complexity and thus shortened execution time.

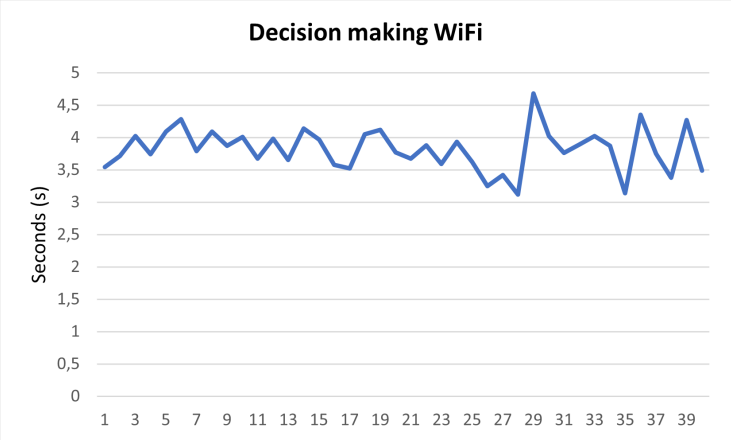


Figure 5.17: Execution time in WiFi decision making.

5.4.3 Final Classification With Weighted Majority

To evaluate the weighted majority's performance, the following test was performed: 40 authentications were performed on each device to determine the final result that the weighted majority provided. As can be seen in Figure 5.18 devices 13 and 14 outperformed the other ER400TRS devices, scoring the same number of true positives and false positives. These two devices identified positive instances with more precision and generated fewer errors (false positives) than the other two devices. Devices 15 and 16 had the most false positives, with device 15 having the most. However, device 16 properly identified a higher proportion of true positives than device 15.

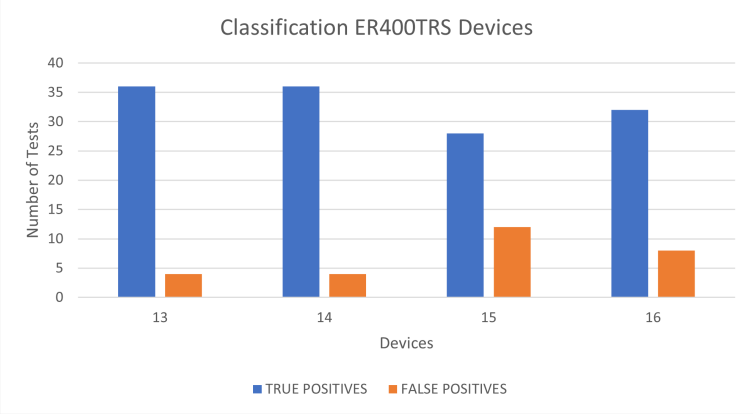


Figure 5.18: Final classification with weighted majority for ER400TRS devices.

Figure 5.19 depicts the weighted majority utilised in the WiFi device results. Because it is a completely different device, the device surface had the highest number of true positives, which means it correctly detected the positive cases the most frequently. The ESPAP had the most false positives.

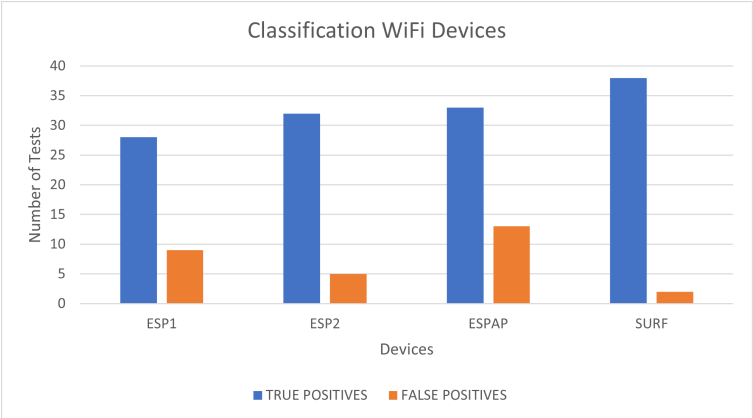


Figure 5.19: Final classification with weighted majority for WiFi devices.

Chapter 6

Conclusions

6.1 Discussion

In conclusion, this thesis has demonstrated that the transformation of RF signal features into images is a valid approach for training CNN models capable of classifying various devices, as evidenced by some of the models achieving accuracies in the feature constellation of 98,37% in the ER400TRS devices and 90,82% for WiFi devices. We've shown that features like the constellation and differential constellation can produce promising results in a variety of spectrum regions. Despite not resulting in optimal results for WiFi the spectogram and PSD, wielded good results for the ER400TRS devices, with 88,00% for the spectogram feature and 95,00% for the PSD feature. The ADALM-PLUTO demonstrated acceptable validation rates in WiFi context, lowering expenses associated with RFF authentication system adoption.

Furthermore, this thesis supports the major advantages of using custom blocks in signal processing for SDR and RFF. This technique, created by a project colleague, has shown promising results in improving noise reduction in the collected signal as well as more precise collection of bursts/signals from the corresponding devices. The use of the custom block technique resulted in significant increases in signal quality, which improved the performance of the CNN models in our investigation. As a result, optimising signal capture and processing through the use of custom blocks is a beneficial method for increasing the effectiveness of CNN models and overall RF signal analysis. It should be noted, however, that the amplitude feature proved

unstable and should not be used for device authentication. It would have been incredibly difficult to effectively realise the notion of RFF without the application of ML techniques.

Based on the findings of this thesis, an important element has been discovered that exerts a significant influence the performance of CNN models in RFF. The findings show that the amount of processing and calculation needed to generate the feature/image in RFF is inversely proportional to the success of the CNN models. The lower the complexity of the transformation and extraction procedure, the better the performance of the CNN models, as evidenced by greater accuracy scores. Features or images that needed more intense processing and calculation, on the other hand, resulted in inferior model performance. As features like constellation and differential constellation yielded better results and where the features that required less time to generate from the obtained singals. This implies that the simplicity and efficiency of RFF feature extraction and transformation can indeed improve the efficacy of CNN models. It emphasises the significance of optimising the processing stage, with a focus on attaining the desired outputs with the least amount of computation. This minimum calculation is additionally aided by custom blocks that minimise noise and obtain bursts from the device's signal to be authenticated.

Another notable achievement of this research is the successful integration of this RFF mechanism into the medical gateway. This result not only represents a significant milestone in our project, but it also paves the way for the application of RFF mechanisms in the larger realm of medical technology. RFF authentication can be used to other systems similar to the medical gateway, particularly systems with edge gateways, demonstrating that RFF techniques can be applied to a wide range of systems.

6.2 Future Work

Regarding CNN models, there is a constraint inherent in their design: if the model was developed to categorise four devices, any device that is not one of those four devices will be classified as one of those four devices. As a result, if there is a need in the future of adding a new device, there will be a need to capture enough samples to train the model again by adding classes equal to the number of new devices that are supposed to be within the CNN model's knowledge. To put it another way, as new devices are introduced, it should be created a process to add datasets and train them to add the correct class to the device, allowing the system to become really adaptive to new devices.

References

[1] F. Al-Turjman, M. H. Nawaz, and U. D. Ulusar, "Intelligence in the internet of medical things era: A systematic review of current and future trends," *Computer Communications*, vol. 150, pp. 644–660, 2020. [Cited on page 1]

[2] R. Krishnamurthi, A. Kumar, D. Gopinathan, A. Nayyar, and B. Qureshi, "An overview of iot sensor data processing, fusion, and analysis techniques," *Sensors*, vol. 20, no. 21, 2020. [Cited on page 1]

[3] Statista1, "Internet of things - number of connected devices worldwide 2015-2025." Available at `https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/`, 2023. (Last accessed in 06/05/2023). [Cited on pages ix, 1, and 2]

[4] L. Tawalbeh, F. Muheidat, M. Tawalbeh, and M. Quwaider, "Iot privacy and security: Challenges and solutions," *Applied Sciences*, vol. 10, no. 12, 2020. [Cited on page 2]

[5] A. M. Gamundani, A. Phillips, and H. N. Muyingi, "An overview of potential authentication threats and attacks on internet of things(iot): A focus on smart home applications," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 50–57, 2018. [Cited on pages ix, 2, and 3]

[6] W. Sun, Z. Cai, Y. Li, F. Liu, S. Fang, and G. Wang, "Security and privacy in the medical internet of things: a review," *Security and Communication Networks*, vol. 2018, pp. 1–9, 2018. [Cited on page 3]

[7] S. S. Mishra and A. Rasool, "Iot health care monitoring and tracking: A survey," in *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, pp. 1052–1057, 2019. [Cited on page 3]

[8] Statista, "Number of internet of things (iot) active connections in healthcare in the european union (eu) in 2016, 2019, 2022 and 2025." Available at `https://www.statista.com/statistics/691848/iot-active-connections-in-healthcare-in-the-eu/`, 2023. (Last accessed in 07/05/2023). [Cited on pages ix, 3, and 4]

[9] R. Somasundaram and M. Thirugnanam, "Review of security challenges in healthcare internet of things," *Wireless Networks*, vol. 27, pp. 5503–5509, 2021. [Cited on pages 4 and 9]

[10] S. A. Butt, J. L. Diaz-Martinez, T. Jamal, A. Ali, E. De-La-Hoz-Franco, and M. Shoaib, "Iot smart health security threats," in *2019 19th International Conference on Computational Science and Its Applications (ICCSA)*, pp. 26–31, 2019. [Cited on page 4]

[11] E. Lomba, R. Severino, and A. F. Vilas, "Work in progress: Towards adaptive rf fingerprint-based authentication of iiot devices," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–4, IEEE, 2022. [Cited on page 5]

[12] R. Hireche, H. Mansouri, and A.-S. K. Pathan, "Security and privacy management in internet of medical things (iomt): A synthesis," *Journal of Cybersecurity and Privacy*, vol. 2, no. 3, pp. 640–661, 2022. [Cited on pages xi, 10, 12, 13, and 14]

[13] mPython, "Internet of things." Available at `https://mpython.readthedocs.io/en/master/tutorials/advance/iot/index.html`, 2022. (Last accessed in 25/05/2023). [Cited on pages ix and 11]

[14] P. K. Sadhu, V. P. Yanambaka, and A. Abdelgawad, "Internet of things: Security and solutions survey," *Sensors*, vol. 22, no. 19, 2022. [Cited on pages 14, 15, and 16]

[15] A. H. Sodhro, A. I. Awad, J. van de Beek, and G. Nikolakopoulos, "Intelligent authentication of 5g healthcare devices: A survey," *Internet of Things*, vol. 20, p. 100610, 2022. [Cited on page 17]

[16] J. A. Gutierrez del Arroyo, B. J. Borghetti, and M. A. Temple, "Considerations for radio frequency fingerprinting across multiple frequency channels," *Sensors*, vol. 22, no. 6, 2022. [Cited on page 18]

[17] S. Wang, H. Jiang, X. Fang, Y. Ying, J. Li, and B. Zhang, "Radio frequency fingerprint identification based on deep complex residual network," *IEEE Access*, vol. 8, pp. 204417–204424, 2020. [Cited on page 18]

[18] T. R. Smith, "Comparing rf fingerprinting performance of hobbyist and commercial-grade sdrs," Master's thesis, Wright State University, Ohio, 2020. [Cited on page 19]

[19] M. Lichtman, "Iq sampling." Available at `https://pysdr.org/content/sampling.html`, 2022. (Last accessed in 11/12/2022). [Cited on page 20]

[20] S. Katz and J. Flynn, "Using software defined radio (sdr) to demonstrate concepts in communications and signal processing courses," in *2009 39th IEEE Frontiers in Education Conference*, (San Antonio, TX, USA), pp. 1–6, October 2009. [Cited on page 20]

[21] N. Soltanieh, Y. Norouzi, Y. Yang, and N. C. Karmakar, "A review of radio frequency fingerprinting techniques," *IEEE Journal of Radio Frequency Identification*, vol. 4, no. 3, pp. 222–233, 2020. [Cited on page 20]

[22] C. Zhao, T.-Y. Chi, L. Huang, Y. Yao, and S.-Y. Kuo, "Wireless local area network cards identification based on transient fingerprinting," *Wireless Communications and Mobile Computing*, vol. 13, no. 7, pp. 711–718, 2013. [Cited on page 21]

[23] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*, MobiCom '08, (New York, NY, USA), p. 116–127, Association for Computing Machinery, 2008. [Cited on page 21]

[24] L. Peng, A. Hu, J. Zhang, Y. Jiang, J. Yu, and Y. Yan, "Design of a hybrid rf fingerprint extraction and device classification scheme," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 349–360, 2019. [Cited on page 21]

[25] F. Galtier, R. Cayre, G. Auriol, M. Kâaniche, and V. Nicomette, "A psd-based fingerprinting approach to detect iot device spoofing," in *2020 IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC)*, pp. 40–49, 2020. [Cited on page 21]

[26] J. Li, Y. Ying, C. Ji, and B. Zhang, "Differential contour stellar-based radio frequency fingerprint identification for internet of things," *IEEE Access*, vol. 9, pp. 53745–53753, 2021. [Cited on pages ix, 21, 22, and 62]

[27] G. Baldini, G. Steri, R. Giuliani, and C. Gentile, "Imaging time series for internet of things radio frequency fingerprinting," in *2017 International Carnahan Conference on Security Technology (ICCST)*, pp. 1–6, IEEE, 2017. [Cited on page 22]

[28] D. Liu, M. Wang, and H. Wang, "Rf fingerprint recognition based on spectrum waterfall diagram," in *2021 18th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pp. 613–616, IEEE, 2021. [Cited on page 23]

[29] Microsoft, "Deep learning vs. machine learning in azure machine learning." Available at `https://learn.microsoft.com/en-US/azure/`

machine-learning/concept-deep-learning-vs-machine-learning, 2022. (Last accessed in 28/12/2022). [Cited on page 23]

[30] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014. [Cited on page 24]

[31] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, pp. 381–386, 2020. [Cited on page 24]

[32] J. Brownlee, "Supervised and unsupervised machine learning algorithms." Available at https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/, 2020. (Last accessed in 2/1/2022). [Cited on page 24]

[33] R. Saravanan and P. Sujatha, "A state of art techniques on machine learning algorithms: a perspective of supervised learning approaches in data classification," in *2018 Second international conference on intelligent computing and control systems (ICICCS)*, pp. 945–949, IEEE, 2018. [Cited on pages 24 and 25]

[34] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018. [Cited on page 25]

[35] P. Kulkarni, S. Londhe, and M. Deo, "Artificial neural networks for construction management: a review," *Journal of Soft Computing in Civil Engineering*, vol. 1, no. 2, pp. 70–88, 2017. [Cited on page 25]

[36] G. MALATO, "How many neurons for a neural network." Available at https://www.yourdatateacher.com/2021/05/10/how-many-neurons-for-a-neural-network/, 2021. (Last accessed in 3/01/2023). [Cited on pages ix and 26]

[37] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, 2018. [Cited on page 26]

[38] C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning," *Electronic Markets*, vol. 31, no. 3, pp. 685–695, 2021. [Cited on page 26]

[39] S. T. Help, "Data mining vs machine learning vs artificial intelligence vs deep learning." Available at https://www.softwaretestinghelp.com/data-mining-vs-machine-learning-vs-ai/, 2023. (Last accessed in 5/01/2023). [Cited on pages ix and 27]

[40] A. Rosebrock, *Deep Learning for Computer Vision with Python: Starter Bundle*. PyImageSearch, 2017. [Cited on page 27]

[41] X. Ying, "An overview of overfitting and its solutions," *Journal of Physics: Conference Series*, vol. 1168, p. 022022, feb 2019. [Cited on page 27]

[42] A. Hidaka and T. Kurita, "Consecutive dimensionality reduction by canonical correlation analysis for visualization of convolutional neural networks," vol. 2017, pp. 160–167, 12 2017. [Cited on pages ix and 28]

[43] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999–7019, 2022. [Cited on page 28]

[44] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into imaging*, vol. 9, no. 4, pp. 611–629, 2018. [Cited on pages 29, 30, and 31]

[45] M. B. A. Miah, S. Akter, and C. Bonik, "Automatic bangladeshi vehicle number plate recognition system using neural network," *Am. Int. J. Res. Sci. Technol. Eng. Math*, pp. 62–66, 2015. [Cited on pages ix and 29]

[46] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016. [Cited on pages ix and 29]

[47] A. Ajit, K. Acharya, and A. Samanta, "A review of convolutional neural networks," in *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, pp. 1–5, 2020. [Cited on page 30]

[48] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artificial intelligence review*, vol. 53, no. 8, pp. 5455–5516, 2020. [Cited on pages 30, 31, 32, 33, 40, and 41]

[49] S. University, "Cs231n convolutional neural networks for visual recognition)." Available at `https://cs231n.github.io/convolutional-networks/conv`, 2022. (Last accessed in 07/01/2023). [Cited on pages ix and 30]

[50] R. Jin and Q. Niu, "Automatic fabric defect detection based on an improved yolov5," *Mathematical Problems in Engineering*, vol. 2021, 2021. [Cited on pages ix and 31]

[51] E. Alpaydin, *Introduction to machine learning*. MIT press, 2020. [Cited on page 32]

[52] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014. [Cited on page 32]

[53] C. Reddy, "Convolutional neural networks." Available at Source, 2020. (Last accessed on 8/01/2023). [Cited on pages ix and 33]

[54] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019. [Cited on pages 33 and 34]

[55] 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. [Cited on page 34]

[56] A. Reustle, T. Rabbani, and F. Huang, "Fast gpu convolution for cp-decomposed tensorial neural networks," in *Intelligent Systems and Applications* (K. Arai, S. Kapoor, and R. Bhatia, eds.), (Cham), pp. 468–487, Springer International Publishing, 2021. [Cited on page 35]

[57] Google, "Trends." Available at `https://trends.google.pt/trends/?geo=PT`, 2023. (Last accessed in 11/02/2023). [Cited on pages ix and 36]

[58] H. Dai, X. Peng, X. Shi, L. He, Q. Xiong, and H. Jin, "Reveal training performance mystery between tensorflow and pytorch in the single gpu environment," *Science China Information Sciences*, vol. 65, pp. 1–17, 2022. [Cited on page 36]

[59] M. Shivanandhan, "Deep learning frameworks compared: Mxnet vs tensorflow vs dl4j vs pytorch." Available at Source, 2023. (Last accessed on 08/05/2023). [Cited on pages xi and 36]

[60] Imaginghub, "Comparing deep learning frameworks: Tensorflow, cntk, mxnet, and caffe." Available at Source, 2023. (Last accessed on 08/05/2023). [Cited on pages xi and 36]

[61] M. C. Chirodea, O. C. Novac, C. M. Novac, N. Bizon, M. Oproescu, and C. E. Gordan, "Comparison of tensorflow and pytorch in convolutional neural network - based applications," in *2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pp. 1–6, 2021. [Cited on pages 37, 38, and 39]

[62] DOMINO, "What is anaconda?." Available at `https://www.dominodatalab.com/data-science-dictionary/anaconda`, 2023. (Last accessed in 11/06/2023). [Cited on page 37]

[63] Anaconda, "Anaconda documentation." Available at `https://docs.anaconda.com/`, 2023. (Last accessed in 12/06/2023). [Cited on page 37]

[64] Keras, "Keras." Available at `https://keras.io/`, 2023. (Last accessed in 13/06/2023). [Cited on page 37]

[65] C. Francois, "Deep learning with python, manning publications," 2017. [Cited on page 37]

[66] R. Elshawi, A. Wahab, A. Barnawi, and S. Sakr, "Dlbench: a comprehensive experimental evaluation of deep learning frameworks," *Cluster Computing*, vol. 24, pp. 2017–2038, 2021. [Cited on page 39]

[67] B. Pang, E. Nijkamp, and Y. N. Wu, "Deep learning with tensorflow: A review," *Journal of Educational and Behavioral Statistics*, vol. 45, no. 2, pp. 227–248, 2020. [Cited on page 39]

[68] Edward, "Tensorboard." Available at `http://edwardlib.org/tutorials/tensorboard/`, 2020. (Last accessed in 10/02/2023). [Cited on pages ix and 39]

[69] N. Gómez, L. López, A. Albert, and J. Llamas, "Image classification with convolutional neural networks using gulf of maine humpback whale catalog," *Electronics*, vol. 9, p. 731, 04 2020. [Cited on pages ix and 40]

[70] N. Strisciuglio, M. Lopez Antequera, and N. Petkov, "Enhanced robustness of convolutional networks with a push–pull inhibition layer," *Neural Computing and Applications*, vol. 32, pp. 1–15, 12 2020. [Cited on pages ix and 41]

[71] F. Alsubaei, A. Abuhussein, V. Shandilya, and S. Shiva, "Iomt-saf: Internet of medical things security assessment framework," *Internet of Things*, vol. 8, p. 100123, 2019. [Cited on pages 42 and 43]

[72] Z. B. Caldwell, "The case for a security metric framework to rate cyber security effectiveness for internet of medical things (iomt)," in *Women Securing the Future with TIPPSS for Connected Healthcare: Trust, Identity, Privacy, Protection, Safety, Security*, pp. 63–81, Springer, 2022. [Cited on pages ix and 42]

[73] D. Bender and K. Sartipi, "Hl7 fhir: An agile and restful approach to healthcare information exchange," in *Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems*, pp. 326–331, 2013. [Cited on page 44]

[74] E. Systems, "Esp32." Available at `https://www.espressif.com/en/products/socs/esp32`, 2023. (Last accessed in 13/06/2023). [Cited on page 47]

[75] DatasheetsPDF, "Er400trs transceiver datasheet pdf." Available at `https://datasheetspdf.com/datasheet/ER400TRS.html`, 2014. (Last accessed in 13/06/2023). [Cited on page 47]

[76] O. Kröger, "Google colab vs own gtx 1060." Available at `https://opensourc.es/blog/colab-vs-gtx/`. (Last accessed in 26/2/2023). [Cited on page 52]

[77] Google, "Installation." Available at `https://www.tensorflow.org/install/source_windowshl=pt-br`. (Last accessed in 26/2/2023). [Cited on page 53]

[78] L. Peng, J. Zhang, M. Liu, and A. Hu, "Deep learning based rf fingerprint identification using differential constellation trace figure," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 1091–1095, 2020. [Cited on page 57]

[79] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017. [Cited on page 59]

[80] B. Komer, J. Bergstra, and C. Eliasmith, "Hyperopt-sklearn," *Automated Machine Learning: Methods, Systems, Challenges*, pp. 97–111, 2019. [Cited on page 63]

[81] C. Vaddepally, "Confusing with confusion matrix." Available at `https://charan316-cv.medium.com/confusing-with-confusion-matrix-dbc69d0cd57b`, 2021. (Last accessed in 15/01/2023). [Cited on pages x and 68]

[82] M. Heydarian, T. E. Doyle, and R. Samavi, "Mlcm: Multi-label confusion matrix," *IEEE Access*, vol. 10, pp. 19083–19095, 2022. [Cited on page 68]

[83] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information processing & management*, vol. 45, no. 4, pp. 427–437, 2009. [Cited on page 68]

[84] scikit learn, "Metrics and scoring: quantifying the quality of predictions." Available at `https://scikit-learn.org/stable/modules/model_evaluation.html#multilabel-ranking-metrics`, 2023. (Last accessed in 02/05/2023). [Cited on page 70]

[85] L. I. Kuncheva, *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2014. [Cited on page 83]