NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

MARCELO TEIXEIRA COLAÇO

Degree in Electrical and Computer Engineering Sciences

MACHINE LEARNING TECHNIQUES IN MODERN CHIPLESS RFID ENVIRONMENTS

CHIPLESS RFID TAG DISTINCTION AND IDENTIFICATION THROUGH MACHINE LEARNING ALGORITHMS.

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
Setembro, 2022

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL
AND COMPUTER ENGINEERING

MACHINE LEARNING TECHNIQUES IN MODERN CHIPLESS RFID ENVIRONMENTS

CHIPLESS RFID TAG DISTINCTION AND IDENTIFICATION THROUGH MACHINE LEARNING ALGORITHMS.

MARCELO TEIXEIRA COLAÇO

Degree in Electrical and Computer Engineering Sciences

Adviser: João Goes
Full Professor, NOVA University Lisbon

Co-advisers: Tommy Sonne Alstrøm
Associate Professor, Danish Technical University

Hiba Nassar
Assistant Professor, Danish Technical University

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
Setembro, 2022

Machine Learning techniques in modern chipless RFID environments

To my Mother, Father, Brother and all my closest friends.

Acknowledgements

I would like to thank SST-NOVA, the Department of Electronics and more importantly, Professor João Goes, for allowing me to travel abroad and granting me the opportunity to learn in new environments, providing some sort of scientific advancement in a topic I have become so deeply entrenched in. This would have not been even in the slightest bit possible without my coordinators at DTU, Tommy Alstrøm and Hiba Nassar, who I cannot thank enough for granting me their time and space with great comfort and personable warmth. This made the experience not only easier but genuinely worthwhile, since their knowledge on this matter showed to be quite vast. An incredible and special thanks is also due to Bernardo Lopes, who not only showed to be immensely helpful, made the entire anti-collision portion of this document possible.

I would like to extend a warm and huge thank you to my close friends Miguel and Marco, who soldiered on with my several weak moments in college and encouraged me through quite possibly the roughest 5 years of my life. I would also be remiss if I did not thank my family, who supported this entire journey and my endless sea of stressful fits, without them none of this would be possible.

"Learning is an active process. We learn by doing... Only knowledge that is used sticks in your mind." (Dale Carnegie)

Abstract

Radio Frequency (RF) tags currently find themselves on the technological vanguard of object identification (Radio Frequency Identification (RFID)) and protection (Electronic Article Surveillance (EAS)), not only being manufactured by several companies, each with their own standard, but being highly on demand in several industries, such as medical [1], logistics [2] and even agriculture [3].

As the world moves further technology-wise, the substitution of old identification systems like optical barcodes, who require line-of-sight, by RFID is becoming ever so popular. The main challenge being the difference in cost between these, and the biggest contributor being the Application Specific Integrated Circuit (ASIC) chip included in RFID tags used in the market today. Since this chip contains the identification data and anti-collision protocols, this removal has posed an open research problem, for a "Chipless RFID" concept [4]. Since anti-collision protocols and strong identification signals can prove to be a problem in this emerging technology [5], the work described in this dissertation proposes to implement a Machine Learning (ML) or Deep Learning (DL) solution on the reader side to successfully fingerprint several chipless RF tags regardless of multiple unique environmental conditions.

Keywords: RFID, Neural Network (NN), Chipless RFID, Artificial Intelligence (AI), Deep Learning (DL), Fingerprinting

Resumo

Etiquetas de radiofrequência (RF) encontram-se na vanguarda da identificação (RFID) e proteção de objetos (EAS), sendo produzidas por variados fornecedores e procuradas por várias indústrias, como a médica [1], a logística [2] e até agroindústrias [3].

Enquanto o mundo progride, a substituição de sistemas de identificação antigos como o código de barras, que requerem campo de visão para a leitura, por tecnologia RFID está a tornar-se cada vez mais popular. O desafio principal encontra-se na diferença de custo destas duas tecnologias, a qual o maior responsável é o chip ASIC que se encontra na maioria das etiquetas RFID do mercado contemporâneo. Dado que este chip é responsável pela identificação única da etiqueta e os protocolos de anti colisão, a remoção completa deste chip da etiqueta tornou-se um tópico de investigação popular no mundo de RFID ("*Chipless* RFID"). Este trabalho vai focar-se exatamente no reenforço das capacidades desta tecnologia, dado que esta ainda apresenta problemas de forte identificação e colisão (em ambientes de múltiplos objetos) [5], através de uma solução de ML ou DL de forma a identificar e classificar com sucesso várias etiquetas de radiofrequência *chipless* independente das condições vizinhas do seu ambiente.

Palavras-chave: RFID, Redes Neuronais, Chipless RFID, Inteligência Artificial, Deep Learning, Fingerprinting

Contents

List of Figures

List of Tables

List of Listings

Glossary

Artificial Intelligence (AI) AI is any form of intelligence either provided to or demonstrated by machines in order to make inferent decisions on different situations and contexts. Examples include speech recognition, object identification and recommendation systems.

Chipless RFID Chipless RFID is the concept of removing the ASIC chip embedded in traditional RFID technology in order to dramatically reduce its price. It was initially sprung by the need of replacement of older but cheaper identification systems like the optical barcode.

Deep Learning (DL) DL is a branch of machine learning algorithms (which itself is a part of AI) based on the neural network architecture. It provides a deep learning capability through the implementation of several hidden layers located in the neural network used. This grants it the ability to extract higher-order features from data without a great amount of labelling and user interference.

Fingerprinting Fingerprinting, or in this specific context, radio fingerprinting is the study of identification of devices only through their physical properties. That is, the identification of a device through either the transient-based or frequency-based features of its emitted signal.

Neural Network (NN) A NN is an algorithm architecture inspired on brain neuron activity. That is, a system comprised of nodes and weighted connections that possess the ability to learn and change it's weights depending on the information fed to it. The result is a network that, depending on the information fed to it, can classify (logistic regression), or infer on data (linear regression).

Acronyms

ADC Analog-to-Digital Converter
AI Artificial Intelligence
ANN Artificial Neural Network
ASIC Application Specific Integrated Circuit

CNN Convolutional Neural Network
CSI Channel State Information

DCTF Differential Constellation Trace Figure
DL Deep Learning
DWFP Dynamic Wavelet Fingerprint

EAS Electronic Article Surveillance
EPC Electronic Product Code

FC Fully-connected
FDR Frequency Domain Reflectometry
FR4 Glass-reinforced epoxy laminate material
FSS Frequency Selective Surface

GNB Gaussian Naive Bayes

HHT Hilbert–Huang transform

IC Integrated Circuit
IDT Interdigital Transducer
IoT Internet Of Things

KNN K-Nearest Neighbours

LDA Linear Discriminant Analysis
LFM Linear Frequency Modulated
LoRa Long Range

MAC Medium Access Control
ML Machine Learning
MLP Multi-layer Perceptron

NN Neural Network

OFTC Organic Thin Film Transistor

PCA Principal Component Analysis
PET Polyethylene Terephthalate
PPM Pulse Phase Modulation

RCS Radio Cross Section
RdF Random Forests
ReLU Rectified Linear Unit
RF Radio Frequency
RFID Radio Frequency Identification

SAW Surface Acoustic Wave
SIB Sequential Information Bottleneck
SNR Signal-to-Noise Ratio
SVM Support Vector Machine

TDR Time-Domain Reflectometry
TFTC Thin Film Transistor

UHF Ultra High Frequency
UWB Ultra Wide Band

VCO Voltage-Controlled Oscillator

WPD Wavelet Packet Decomposition

Symbols

a	radius of the coil
B	intensity of a magnetic field
c	round inductance geometry constant
C	capacitance
d	average diameter
K_1, K_2	Wheeler geometry constants
I	current
L	inductance
R_s	load resistance
N	Number of turns in the coil
p	inductance of squared geometry constant
μ_o	permeability of free space = $4\pi * 10^{-7}$
Q	quality factor
r	perpendicular distance from the reader coil to the tag's coil
f_0	resonant frequency
V	voltage

w	angular frequency
$Z(jw)$	Impedance in fuction of angular frequency

1 Introduction

RF identification or RFID was first described in a patent as "the primary station comprises a source of oscillations and an emitting circuit; the secondary has a receiving circuit with no local energy source, and in which is inserted a manipulating or modulating device (telephonic or telegraphic apparatus)" [6]. This early concept can still easily be used to describe the communication between an RFID reader (primary station) and a passive/semi-passive/active RF Tag (secondary station), even though this patent was first registered in 1924. It was only in the 60s that the concept of RFID would be commercially available as EAS Tags, tiny round plastic apparatuses that would be attached or contained in store products and eventually supervised by big reader gates located at the front of the store. Its use was fairly simple but highly effective.

The Tag would contain a simple LC tank circuit, whose resonant frequency would match the one emitted by the supervising reader gates found at the entrance/exit of the store, pictured in Figure 1.1. These gates would have a sweep generator, that would emit

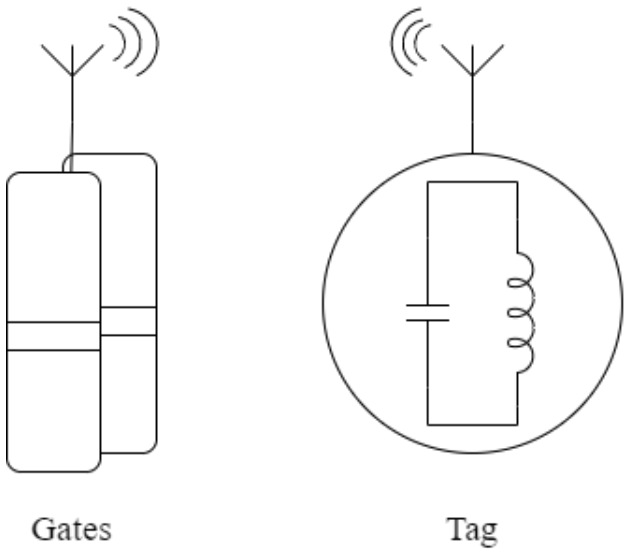


Figure 1.1: Illustration of the EAS Tag process.

a small range of frequencies close to the Tag's resonance. If the sweep reached the Tag at a close range (about 1-3 meters), the Tag would respond with the energy absorbed from the emitting signal, also known as a back-scattering signal. When the back-scattered signal reached the gates, this would set off an alarm warning the shopkeepers that the item that was being taken hadn't been purchased. If it was, the shopkeepers would deactivate the Tag at purchase, disabling the capacitor found in the Tag's circuit by sending a large amplitude signal at the resonant frequency into the Tag. While the world of RFID has improved significantly after this simple concept, its functioning principles still stand. These Tags came to be known as RF or binary Tags, since the only information that was determined with this system would be the presence of the Tag itself. Since then the world of RFID evolved greatly with the advances of integrated circuit technology, making it possible to embed data and multiple object anti-collision protocols (an aspect that was foreign in RF Tags since the mere presence of a signal would indicate something was being stolen, in theory) with a micron-sized ASIC embedded chip. However, the inclusion of this new technology, while impressive, became an obstacle in its worldwide standardization in universal commerce. In other words, cheaper technologies that didn't need a chip to identify objects like optical bar-codes were more popular but lacked the properties that made RFID so valuable, such as a lack of line-of-sight reading and "simultaneous" multiple object detection. This led to the open research investigation to remove the most expensive component of the common RFID Tag, the ASIC chip, for a now named "Chipless RFID" concept [7], spawning several topologies which will be further discussed in this work. Nonetheless, the "Chipless RFID" concept tackled several problems, such as:

- Lack of anti-collision or handshaking protocols with the reader.
- No data present to uniquely identify objects.
- Accessibility and price compared to other technologies.

Relating to the identification and detection of these chipless Tags, radiofrequency fingerprinting can be a strong aspect to look into since this process involves the identification of devices without the need of actual encoded data in chips [8]. Instead, this technique involves the physical properties of the Tag either through a transient-based signature or a frequency-based signature. This can be an especially interesting case when you consider that this technique is quite compatible with machine learning algorithms, which will be discussed further in this documentation.

Taking these aspects into consideration, the following work will be dedicated to the development and training of a machine learning algorithm using simulation data from an RFID system, to specifically address these problems, more specifically the lack of anti-collision and data present to uniquely identify objects (fingerprinting). The algorithm will be able to distinguish (anti-collision) and classify multiple and unique passive (meaning they do not contain an active power source or battery inside) chipless RFID Tags

at different distances from the reader, independent of environmental conditions. This dissertation is organized as follows:

In Chapter 2, a state of the art will be carried out, exploring the most recent and popular examples of modern chipless RFID Tags, as well as their fabrication. Radio fingerprinting techniques will also be researched, in order to compare and figure out the best candidate for the implementation of the project. In Chapter 3, an overview of the implemented system will be reviewed, as will all the operating principles and algorithms used. In Chapter 4, the actual implementation and simulation of the project will be discussed along with the results from the simulations. In Chapter 5, a final conclusion and review of the future work that can be done to improve the project will be displayed.

2

State-of-The-Art

This chapter will primarily focus on the most recent examples and solutions relevant to the discussed theme, solution and technologies used in this following work.

2.1 RF Tags

RF Tags, or rather the technology they operate on, have found themselves to be useful in more applications than their original surveillance purpose, from providing information monitoring capabilities [9] to tuning radio transmitters and receivers. This technology while simple is incredibly effective in these fields while being a very low-cost solution.

2.1.1 Operating and Design Principles

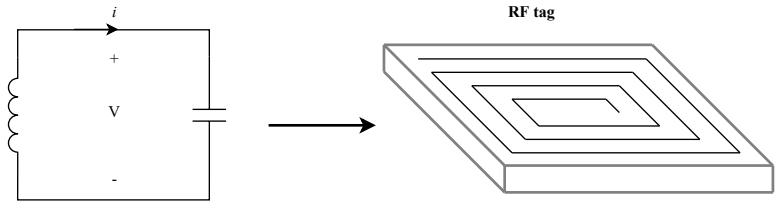


Figure 2.1: LC Circuit and its Tag counterpart.

RF Tags are commonly designed as a simple spiral inductor (seen in 2.1), acting as an antenna. The thickness and material of the Tag are heavily influential in the Tag's behaviour and resonant frequency [10]. The fundamental operating and design principle of the Tag is based on the electrical resonant frequency created by the inductive coupling of the magnetic field of the reader [11]. This magnetic field can be described using the following formula:

$$B = \frac{\mu_o I N a^2}{2r^3} \tag{2.1}$$

where:

- I = current that passes through the inductor coil;
- N = number of windings in that coil;
- a = radius of the coil;
- μ_o = permeability of free space ($4\pi * 10^{-7}$);
- r = perpendicular distance from the reader coil to the Tag's coil.

As we can see, the magnetic field's intensity is inversely proportional to r^3, which is why this type of coupling has been characterized as near-field coupling. If we consider Ohm's Law to define the current passing through the pattern:

$$I = \frac{V}{wL} \tag{2.2}$$

We can then describe the resonant frequency, f_0, as the point at which the current provided to the antenna is at its maximum, given its inductance and capacitance:

$$f_0 = \frac{1}{2\pi\sqrt{LC}} \tag{2.3}$$

Here the inductance L of the circuit is heavily dependent on the geometry of the antenna, meaning if it's square, octagonal or even round (commonly when the number of sides surpasses 64). This inductance can be described through modified versions of the Wheeler formula [12] [13].

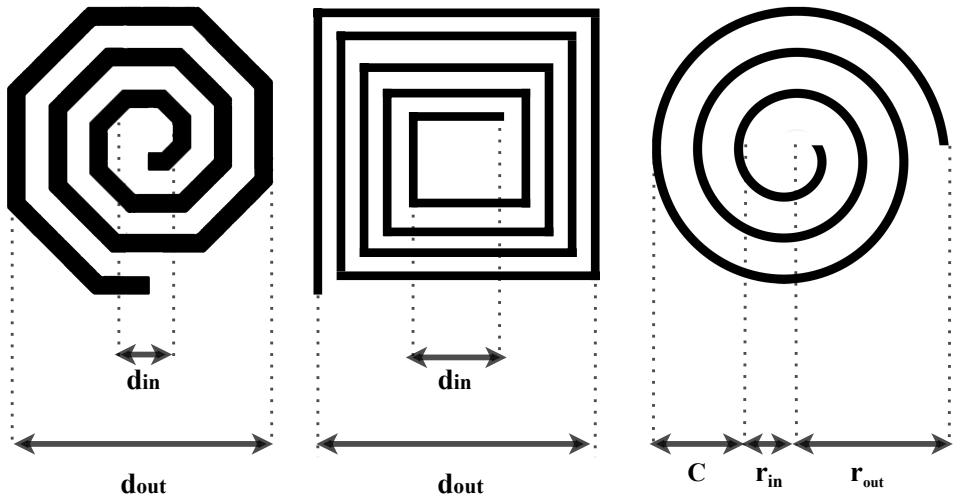


Figure 2.2: Common spiral geometries of RFID antennas.

The inductance of squared antennas such as octagonal (left side of Figure 2.2), squared (centre of Figure 2.2) or hexagonal can be determined by the following equation:

$$L = K_1 \mu_o N^2 \frac{d}{1 + K_2 p} \tag{2.4}$$

where:

- d = average diameter ($\frac{d_{out}+d_{in}}{2}$);
- $p = (\frac{d_{out}-d_{in}}{d_{out}+d_{in}})$;
- K_1, K_2 = geometry constants.

Additionally, this formula has two constants (K_1, K_2) that are varied upon the number of sides in the square antenna, as seen in Table 2.1.

Table 2.1: K constants of the Wheeler Formula for squared antennas.

Geometry	K_1	K_2
Square	2.34	2.75
Octagonal	2.25	3.55
Hexagonal	2.33	3.82

For round spirals (right side of Figure 2.2), we can use the following equation:

$$L_{round} = 31.33 \mu_o N^2 \frac{a^2}{8a + 11c} \tag{2.5}$$

where:

- a = average radius ($\frac{r_{out}+r_{in}}{2}$);
- $c = (r_{out} - r_{in})$.

The squared antennas tend to be more popular in this design process since the fabrication usually involves etching or printing of the conductive material and round spirals are harder to print dependent on the printing machine's resolution capabilities. Now that the inductance of the antenna has been determined, the computation of the capacitance usually follows. This can be done either using a tuning capacitor attached to the terminals of the antenna or using the area of the spiral itself in combination with the substrate's properties.

$$Z(jw) = R + j(X_L - X_C) \tag{2.6}$$

When the circuit is in resonance, it becomes purely resistive, meaning $X_L = X_C$, thus maximizing the current passing through the circuit, acting as a switch. The quality factor of the circuit is also important in its design, since it describes how well the circuit absorbs

the received power from the reader signal and is mainly determined by the energy loss of the Tag's antenna.

$$Q_{Tag} = \frac{wL}{R_s} \tag{2.7}$$

This factor is commonly around 20 to 40 for LC resonant circuits, depending on the core's material (air or ferrite, for example). Lastly, to optimise the Radio Frequency (RF) Tag to its maximum efficiency three general requirements must be met [11]:

- The quality factor (Q) of the antenna must be maximized.
- The size of the antenna must be maximal in the context of the Tag area.
- The LC tank must be tuned to the carrier frequency of the reader (13.56 MHz, 8.2 MHz,...)

Although RF Tags are a starting point in terms of the technology we will be discussing, these fundamentals should be considered as they describe the basis of the design and function of the further systems.

2.1.2 Materials and Fabrication Methods

As previously mentioned, the thickness and material of the Tag is quite influential in the Tag's behaviour and resonant frequency [10] [14]. This can be crucial, especially when you consider that the second-biggest contributor to the cost of a Tag is its material and fabrication method. The fabrication of Tags is usually composed of three or four layers (as seen below on Figure 2.3):

1. A dielectric substrate;
2. A conductive layer;
3. An adhesive and/or protective layer.

Firstly, let's address the substrate: this layer is responsible for keeping the circuit structured and can vary in material depending on the application. The dimensions and material of the substrate can alter not only its dynamic use but the efficiency, read range [14] and even the resonant frequency. The most common form of substrate in RFID is a flexible kind, be it paper, film or plastic, since this allows it to be more dynamic in use and fabrication. Nowadays, the most popular substrate materials are:

1. Polyethylene Terephthalate (PET);
2. Glass-reinforced epoxy laminate material (FR4);
3. Paper.

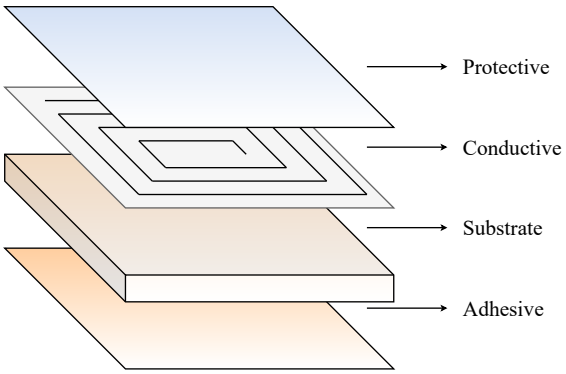


Figure 2.3: Illustration of the Tag composition.

The first two substrate materials and their effects were analysed by Cho et al. They established through the analysis of a meander Tag type on the two materials that PET shows a more robust resistance to the variation of substrate thickness than FR4 in terms of efficiency: considering a thickness of 5 mm, PET achieved 50 % while FR4 a mere 28 % [14]. Additionally, FR4 doesn't present the same amount of flexibility as PET, being only a semi-flexible material. The third material, paper, although higher in conductivity, is becoming ever so popular thanks to its universal availability, price and dielectric properties [15] [16] [17] [18]. Additionally, the development of global awareness to find more sustainable materials for electronics has pushed paper to the technological vanguard for its organic properties [19].

Table 2.2: Various substrate materials and their properties.

Material	**Relative Permittivity ε_r**	**Type**
[20] FR4	2.4	*Semi-Flexible, Non-organic*
[4] PET	2.8	*Flexible, Non-organic*
[4] Paper	3.2	*Flexible, Organic*

By gathering these parameters and taking them into consideration, assessing them through Table 2.2, we can see that while the relative permittivity of paper is higher than PET's (the higher the permittivity the lower the bandwidth achieved by the Tag [4]), it can be argued that paper shows more promise as the future of RFID for its organic and sustainable properties.

Secondarily, the conductive layer is what induces the electronic functions into the Tag, in the case of an RF Tag, it is responsible for the composition of the antenna. There are multiple methods of deposition of this layer on the substrate. However, the most popular are copper-etching, electro-plating and inkjet-printing, in which, the latter is more generally employed due to its ease of use, availability and, again, price. Various conductive materials using this technique have been taken into consideration over the past decade, including silver, and other branded mixtures in order to obtain the best

mix in conductivity and thickness of the metal-line. Nikitin, Lam, and Rao [21] assessed the difference between silver ink and copper (by etching) in read range performance by fabricating two antenna types, a straight dipole and a meandered dipole. While the straight dipole didn't show much difference between the materials, the meandered dipole showed a difference of about 2 meters in the read range (from about 12 meters to 14 meters), the highest distance being the copper material.

Rishani et al. [20] compared two popular combinations of the above-mentioned layers and studied their effects on the antenna's properties. These combinations were copper etching on FR4 substrate and silver ink printing on PET. While there were similar results, it was found that copper, while being a closer match to theoretical calculations, is rigid and complicated to print on flexible material. The hardships of the implementation of copper were also corroborated by Beedasy and Smith [22], mentioning that: while silver is much more expensive, silver is a better conductor and copper as a material is not as dynamic to several environments due to the formation of copper oxide. Taking these sources into account, we can appraise silver as the more dynamic and stable, although expensive, material.

Lastly, the protective and adhesive materials are normally thin non-conductive materials with the sole purpose of adding either extra protection and branding/visual information, in the case of the protective layer, and to provide the ability to stick the Tag onto products or other objects, in the case of the adhesive layer.

2.2 Chipless RFIDs

One of the main challenges related to the implementation of RFID as a commercially viable alternative to bar-codes is the price. While RFID technology clearly delivers more benefits, such as no need for line-of-sight-reading and multiple object identification, these benefits rarely outweigh the overwhelming difference in cost. The component that contributes the most to the price of an RFID Tag is the on-chip memory or data chip contained inside, which is not only responsible for the unique identification of the Tag, but also the anti-collision logic needed for multiple object scenarios. This has led to the rising popularity of what is being called "Chipless RFID" [23] [24], this is, the concept of uniquely identifying passive radiofrequency Tags (something RF Tags lack) without the need for embedded silicon integrated circuits, bringing the expected price down to a fifth of a bar-code's [7]. Thus, most of the modern development of radio-frequency Tags has mostly deviated from chipped and RF Tags, in order to find a solution with the best of both technologies. This concept has been implemented through several technologies, which we will now further discuss.

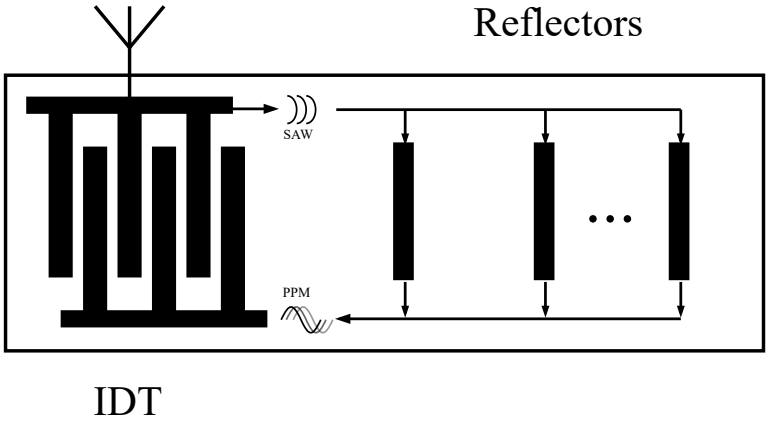


Figure 2.4: Illustration of the Global SAW Tag functioning principles.

2.2.1 TDR-based Tags

Time-Domain Reflectometry (TDR) based radiofrequency Tags, use modulation in time to communicate between the reader and the Tag. This is achieved through a series of pulses or any form of energy that can be read as a function of time, around a single resonant frequency [25]. TDR based Tags are usually read by an Ultra Wide Band (UWB) impulse radio transceiver. This is done by emitting a very short pulse as the interrogation signal, which is then sampled back by the reader's Analog-to-Digital Converter (ADC) time-sampling abilities. One of the more robust implementations of this type of chipless hypothesis was through Surface Acoustic Wave (SAW) technology to create the first commercially available chipless RFID Tag [26]. The aptly named "Global SAW Tag" functions by receiving a 2.45 GHz signal through an antenna, which is then processed by an Interdigital Transducer (IDT), in order to propagate it through the piezoelectric substrate it is printed on. The result is a surface acoustic wave that reverberates throughout the chip, which is then transmitted back into pulses through strategically placed reflectors, that use position to create offsets in the phase of the wave received. The signal outputted by these are then transmitted back into the IDT for the modulated backscattered signal as phased pulses of energy.

This encoding method is called Pulse Phase Modulation (PPM) and is achieved by the placement and number of reflectors present on the chip. Hartmann, Brown, and Bellamy [26] were able to create through this technology a theoretically infinite amount of data that could be transmitted through these chipless Tags. These represent a major step in the right direction to replace bar-codes, since these Tags do not need an Integrated Circuit (IC) and are cheaply made through one process of photolithography. They also tend to need less than one-hundredth of the power that IC containing Tags need to operate [27], largely increasing their read distance. However, while there are modern implementations of this early 2000s technology [28], these Tags present a more complicated demodulation

scheme for readers since it is time-domain based, versus the use of S parameters and spectral signatures for fingerprinting in conventional RFID communication. They also tend to be more complicated in dynamic use since they are usually non-printable and have become more expensive than its frequency-signature counter-part, making it under-used in modern environments. However, these Tags are still used in more challenging environments such as the automobile automation industry [7]. Another TDR-based Tag is called a Thin Film Transistor (TFTC) Tag. These tend to be smaller and consume less power than SAW Tags, but are scarce due to their complex design depending on thin-film transistor printing technology. However, they also present an interesting direction in sustainability with recent research development of organic thin-film transistor technology [29].

2.2.2 Frequency Domain Tags

Frequency domain or frequency signature-based communication in radiofrequency Tags is a type of communication solely focused on the frequency characteristics of the back-scattered signal of the Tag in order to identify it. In the frequency domain, we can distinctly identify two types of radiofrequency Tags: Retransmission-based and Backscattering-based Tags.

2.2.3 Retransmission-based Tags

Retransmission-based Tags usually function by intercepting the energy field of the reader with a vertically polarized antenna (Rx), processing it, and transmitting it back through a separate antenna (Tx). The identification by the reader is usually done by looking at either the S-parameters or the Radio Cross Section (RCS) of the Tag, as we will further discuss in Section 2.2.4. A popular implementation of the chipless concept in this domain is the multi-resonator Tag, first proposed by Preradovic et al. in 2008. This passive Tag takes a signature-based approach and is composed of an Rx antenna, Tx antenna (both UWB monopole antennas that are cross-polarized to minimize the interference between the read and transmitted signal) and several cascaded spiral resonators where each spiral is equivalent to 1-bit capacity [30] [24] [31]. The multiple spiral resonators embedded in the Tag are tuned to different frequencies so that when the interrogation signal from the reader is received, a spectrum analyser can detect several dips in the frequency range (typically 2.4-2.5 GHz with a band of 500 MHz [24]), which are then utilized to decode a binary message. Every resonant frequency is a logic '1', while its absence is a logic '0': thus the Tag can be interpreted as a multi-stop band filter, in terms of spectrum.

This exact technique was expanded upon in 2021 by Shukoor, Mukeshbhai, and Dey, implementing a 12-bit multi-resonator Tag with a resonator for each bit (12 spiral resonators total).

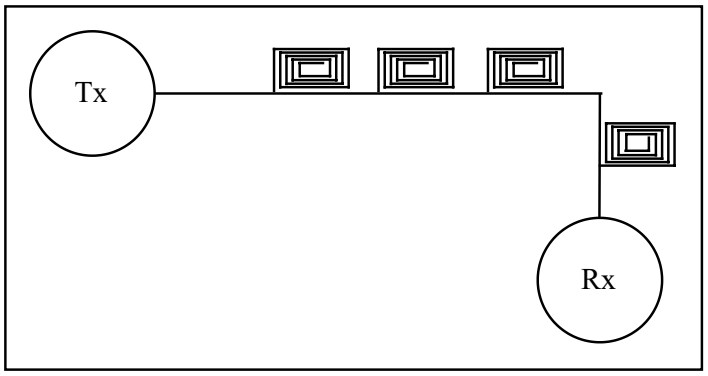


Figure 2.5: Multi Resonator Tag design.

2.2.4 Backscattering-based Tags

Similar to retransmission-based Tags, the Tags are also based on multi-resonator technology, however, there are no separate communication antennas. The resonators tend to act like the encoders and the reflectors by the interception of the electric field of the reader antenna and the following backscattering of the signal. The design is commonly known by following a Frequency Selective Surface (FSS) design methodology [33]. The identification and detection of these Tags are mostly based on the RCS signature the resonators leave on the frequency spectrum. Research shows that modelling the resonators and the antenna performance around this metric supplies better performance compared to chipped RFID [5]. The RCS of a Tag is a metric that describes the Tag's ability to reflect electromagnetic waves to the reader antenna, meaning, the back-scattered energy of the Tag. This metric is dependent on:

1. Operating frequency;
2. Tag's electrical properties and physical shape;
3. Polarization of antenna;
4. Angle of incidence;
5. Observation angle.

And can be defined by [34]:

$$\sigma_{Tag} = \frac{\lambda^2 R_a G_{Tag}^2}{\Pi |2Z_a|} = \left[\frac{S_{21}^{Tag} - S_{21}^{room}}{S_{21}^{ref} - S_{21}^{room}} \right])^2 \sigma_{ref} \tag{2.8}$$

where:

- λ = operating wavelength;

- G_{Tag} = gain of the Tag antenna;
- Z_a = input impedance of the Tag antenna;
- R_a = real component of Z_a;
- S_{21}^{Tag} = S-parameters of the Tag;
- S_{21}^{room} = S-parameters of the empty room;
- S_{21}^{ref} = S-parameters of a reference, i.e metallic plate or other objects.

Furthermore, this metric allows for the easy definition of the Tag's reading range with it being computed by:

$$R_{range} = \sqrt[4]{\frac{\lambda^2 P_t G_{Tx} G_{Rx} \sigma}{(4\pi)^3 P_{min}}} \tag{2.9}$$

where:

- G_{Tx} = gain of the reader antenna;
- G_{Rx} = gain of the Tag antenna;
- P_{min} = sensitivity of the reader;
- P_t = transmitted power.

RCS for chipless Tags can be divided into two main modes:

1. Structural Mode RCS : dependent on the physical properties of the Tag like the reflecting surface and the frame, is established when the Tag is in conjugate match (impedance match of Tag and reader). It is, typically, of greater amplitude, reaching the reader in an easier fashion.
2. Antenna Mode RCS : defined by the design characteristics of the Tag. This mode is what normally contains the actual frequency signature and information, although it tends to be of lesser amplitude and can be at times hard to recover [5].

Now that we have defined an important identification feature for this type of Tag, let us review some recent examples of the implementation of this category of Tag.

Betancourt et al. [35] employed several octagonal shaped-features to create this exact effect. The Tags work in a UWB frequency range (3.1 GHz to 10 GHz) and provide a 5-bit data capability thanks to its numerous resonators. Furthermore, the Tags supplied the capability of late-stage coding, meaning, the ability to alter and write any 5-bit code onto the Tags with a conductive ink pen or printer. This identification was done by analysing

the Tag's RCS. In the same fashion as the previous circuit, the Tag uses loop antennas, however, by combining 6 different octagonal antennas in an onion like cross-section and copying it 4 to 6 times in the Tag space, the 5 total notches are etched deeper in the RCS, increasing the probability of detection [36]. This Tag design can be implemented easily and in a low-cost fashion, thanks to its inkjet printing compatible design. Additionally, the design was also compared in paper and PET substrates with satisfying results [35].

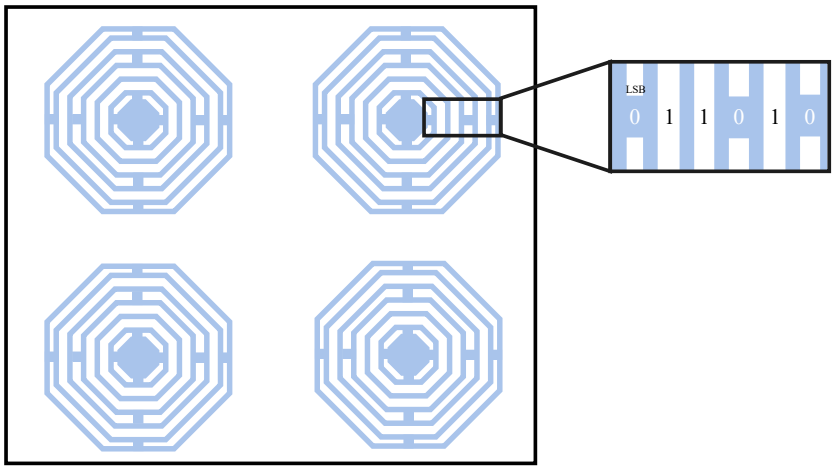


Figure 2.6: Octagon Tag design and illustration of coding principles.

The previously mentioned late-stage coding ability is implemented through the short-circuiting of the loops by bridging the gaps between the octagons symmetrically (as seen in Figure 2.6). This can be done with a conductive ink pen or a printer, where the gap between the inner octagon and the first smallest loop, represents the least significant bit. The codification then follows the same previously mentioned logic as Preradovic et al. [24]'s circuit in the RCS spectrum. In terms of results, the Tag achieved a maximum read range of about 1.5 m if the full codification (5 notches across 3.1 GHz to 10 GHz) is considered by the reader.

Furthermore, there is a variation of the multi-resonant topology that fits into the back-scattered category, using several polarised dipoles instead of spiral resonators, fittingly named multi-resonant dipole based Tags. Havlicek et al. [37] applied several capacitively-loaded dipole scatterers to create a unique frequency signature presented in the RCS, similar to the last Tag mentioned. The Tag operates in the UWB range (2.0 - 3.6 GHz) with a size of 16.7 x 67.8 mm, and consists of an array of the previously mentioned, strategically placed capacitively-loaded dipoles (each one resonating at a different frequency) to produce a 20-bit code (seen in Figure 2.7), in which each dipole represents 1-bit.

Whereas back-scattering Tags are worse in read sensitivity in comparison to re-transmission Tags, the separate antennas needed for the latter tend to make them more expensive [5].

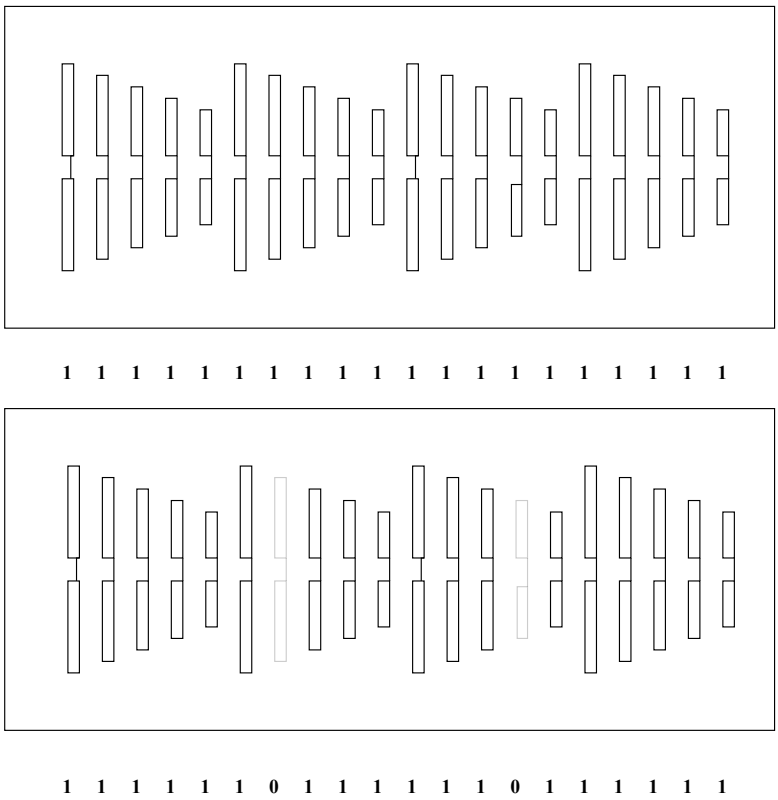


Figure 2.7: 20 bit dipole Tag design and illustration of their respective codes.

The Frequency Domain Reflectometry (FDR) reading process commonly uses continuous wave signals in a range of frequencies (controlled by a Voltage-Controlled Oscillator (VCO)) [25]. The seemingly simple design of these frequency-domain Tags have made them quite popular in the field of chipless RFID [38] [32] [23], although the realistic operating principles for this technology is still in its infancy, since anti-collision protocols and RFID reader development in general for chipless frequency-domain encoding Tags such as these, is quite recent and low in numbers [39] [40] [4] [41].

2.2.5 Image-based Tags

As we have previously established time domain based Tags tend to be more expensive, and, while frequency domain Tags are cheaper they tend to occupy a large amount of frequencies, operating commonly in the UWB range. Image-based or mm-Wave Tags, look to expand on this problem and add solutions by using the imaging of the backscattered electromagnetic field in a small bandwidth capacity [42]. The Tags are commonly designed using microstrip based technology of conductive and dielectric materials to encode barcode-like strips into a small Tag: these produce a specific amount of reflected energy, producing a SAR-based signature electromagnetic image across a small range of frequencies. While this Tag is heavily in its infancy, there are some relevant works

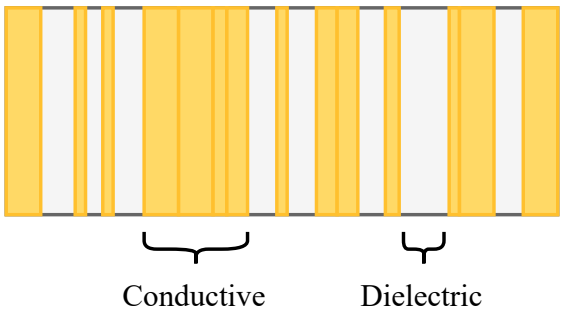


Figure 2.8: Illustration of the barcode-like structure of an image-based Tag.

that illuminate it's interesting purpose [25]. It has also showed some potential for Deep Learning (DL) applications, since image-based algorithms such as Convolutional Neural Network (CNN) thrive in its signature properties [43]. Now that we have explored chipless radio frequency Tags and their most recent examples, let us delve into how to detect them and the modern ethos of "fingerprinting" techniques.

2.3 Radio Fingerprinting

While various techniques involve the detection of RF circuitry, there is a case to be made that radio fingerprinting might be the most promising one given its compatibility with artificial intelligence, more specifically CNN [44] which will be discussed later on. Radiofrequency fingerprinting is, at its basis, the acquisition and computation of the received signal's physical properties, or features, in order to identify the device that sourced it [45], meaning the act of taking a signal and by its transient shape, frequency or even power spectral density [8] identifying what created it. This can be done using a transient-based approach, which utilises properties from the waveform of the received signal in a time domain, or a steady-state approach, which utilises several transforms and techniques to obtain a consistent signature of the signal received, sometimes in different domains. The most practical approach is typically the latter, since it tends to obtain more unique and stable features. Fingerprinting is also a very commonly used technique to reinforce security at the physical Medium Access Control (MAC) layer in communication protocols [46].

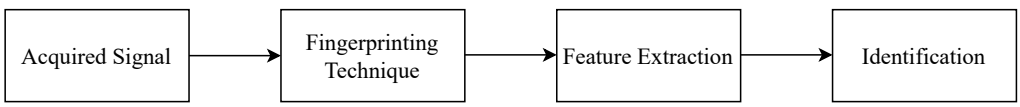


Figure 2.9: Flowchart of the fingerprinting process.

Pertaining to RFID, its first realistic example (usable in higher Signal-to-Noise Ratio (SNR) environments) was performed by Danev, Heydt-Benjamin, and Capkun [47], who used features like modulation shape and spectral features of the back scattered signals

to classify and identify 50 RFID smart cards with an error rate of 2.43 % on a single run, with the purpose of preventing RFID cloning efforts. Fundamentally, the features would be compared to a pre-existing database of "fingerprints" of different RF transponders, in order to either identify a Tag distinctly (identification) or verify the identity of a Tag (classification) and extracted in a three-step process:

1. A one-dimensional Fourier transform is applied to each sample, turning a signal f(t,l), where f is the amplitude of signal l at time t, into $F(\omega,l)$.

$$F(\omega,l) = \frac{1}{\sqrt{M}} \sum_{m=0}^{M-1} f(t,l) e^{-2\pi i \frac{t\omega}{M}} \tag{2.10}$$

   where M is the length of the sample. The spectrum obtained is then filtered, removing the DC component and unnecessary frequencies, and named $\vec{sl}$.

2. A projected vector $\vec{gl}$, also known as a spectral feature, is extracted from the Fourier spectrum using a Principal Component Analysis (PCA) matrix, W_{PCA}^{t}.

$$\vec{gl} = W_{PCA}^{t} \vec{sl} \tag{2.11}$$

   Each of these projected vectors is then stored in an array $G = W_{PCA}^{t} S$.

3. The final fingerprint or feature template is then computed:

$$h = \{\hat{G}, \Sigma_G\} \tag{2.12}$$

   where $\hat{G}$ is the mean vector of G and Σ_G is the covariance matrix of G. The difference between the reference feature templates and the test ones that were computed, is calculated through the Mahalanobis distance, providing a matching score.

This marks an important aspect in RFID fingerprinting since the operations provided were low-costly in computation and memory, providing the possibility to be integrated into lightweight readers. The processing pipeline presented also supplies a remarkable resemblance to traditional machine learning algorithms (with the use of PCA matrixes), which will be discussed further in Section 2.3.1. The process of feature extraction in this application of fingerprinting is important as it shows to be one of the key differences in most articles of this problem, being very influential in the accuracy of detection and identification [8]. This is where Artificial Intelligence (AI) shines in capabilities as well.

2.3.1 Machine Learning and RF Fingerprinting

Machine Learning (ML) can be defined as a subset of AI and the study of computational algorithms with the purpose of training and providing intelligent and inferring capabilities on several situations to machines. It is also very recurrent in the use of data and

pattern recognition, being widely popular in the world of data science and computational statistics. Specifically, in the ethos of data and pattern recognition, a section of ML stands out, DL. DL is a specific class of ML algorithms comprised of multiple hidden layers to break down and extract higher-order patterns and features from raw data. These algorithms are normally based on Neural Network (NN) architectures, neuron-like nets that mimic the human brain's ability to learn through repeated feeding of impulses or, in this case, data.

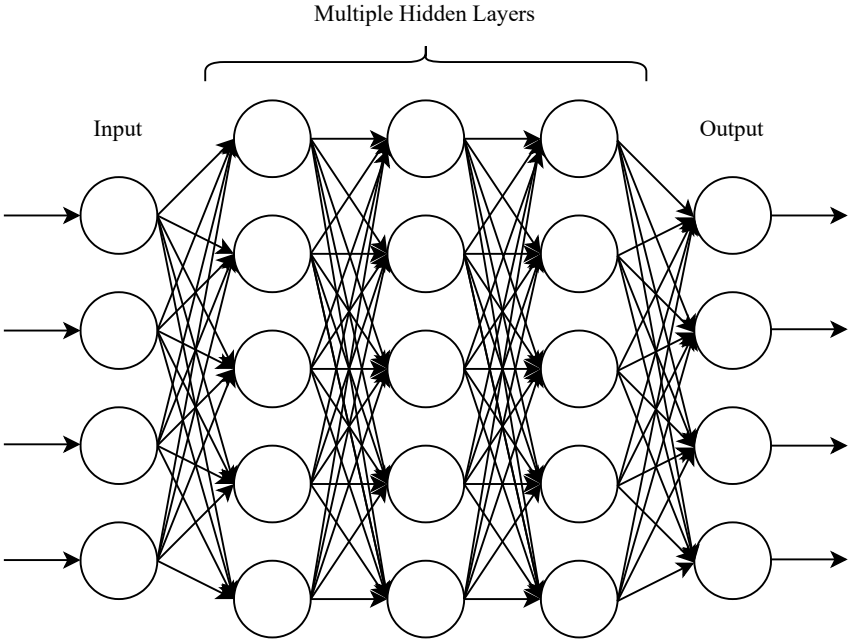


Figure 2.10: Illustration of a typical DL neural network.

As mentioned previously, machine learning has the capability to identify and classify information when fed raw data, while also extracting higher-order features from large data pools efficiently. Since feature extraction, classification and identification are such vital components of RF fingerprinting, a case can be made that such algorithms fit the needs of the technology quite well [48] [49] [50] [51] [52] [53] [54] [55].

Patel, Temple, and Baldwin [53] compared several machine learning algorithms reaching an accuracy of 90 % with a SNR of up to 18 dB employing an algorithm known as a Random Forests (RdF) classifier to correctly identify ZigBee devices, to prevent rogue hacker attacks and hijacking of Internet Of Things (IoT) devices since these are becoming ever so popular with consumer and military technology. Although 90 % presents itself as a confident number, the low level of noise resistance (42 % accuracy in 0 dB SNR environments [53]) and lack of low dimensionality and scalability in these algorithms make it complicated to implement in today's quotidian environments.

Other ML algorithms have also been experimented with by Robyns et al. [54], who

used supervised learning and zero-shot image classification to exclusively identify another type of IoT device, known as Long Range (LoRa) devices, with an accuracy of 59 % to 99 % in devices of the same chipset. This is quite significant, taking into account that devices of the same manufacturer tend to have similar physical properties or fingerprints, making it hard to distinguish them. The classification was done by using image pixels or time-domain waveform samples of the devices instead of hand-picked features. It was found that supervised classification in this field might be complicated to implement since this requires a large database of previously labelled data in order to properly train the algorithm, a process that can take a long time and a lot of effort. In a more realistic scenario, the reader would not have a large database on unknown transmitters: this scenario calls for classifiers such as zero-shot classification and semi-supervised anomaly detection. These function as an entry point to determine if the emitter acquired can be found in the database or not by distributing the previous labels in a Gaussian distribution and using the hypothesis function previously obtained by the supervised classifier, it decides if the probability outputted fits in the distribution by a threshold.

If it fits, it can be classified by the most probable label. Otherwise, it is fed into a different algorithm [56] in order to be clustered and used in classification for future encounters. While the accuracy of this technology is impressive, an SNR metric was never analysed, making it difficult to envision in realistic or high SNR environments. Relative to RFID, Bertoncini et al. [49] used wavelet fingerprinting, a subtype of radio fingerprinting that utilizes the Discrete Wavelet Transform to extract features to identify from the Tag, more specifically Dynamic Wavelet Fingerprint (DWFP) and Wavelet Packet Decomposition (WPD), with an accuracy of up to 99 %, with the purpose of anti-counterfeit of RFID. While the feature extraction is mostly done by the DWFP and WPD, computing properties such as Shannon entropy, Skewness, Kurtosis and the mean and maximum cross-correlation of the Electronic Product Code (EPC) code of the Tag, Bertoncini et al. went on to then use machine learning classifiers to make the final decision of identification and classification when fed the computed ”fingerprints”, such as Support Vector Machine (SVM) and K-Nearest Neighbours (KNN). These last two algorithms were also used by Jeong et al. [57] in the ethos of RFID. However, a 2-bit retransmission-based chipless Tag was studied extensively instead, achieving a detection accuracy of 98.5 %. This was done by recording the S parameters of the Tag through multiple parameter sweeps, such as:

- Varying the read distance from 2 cm to 50 cm;
- Attaching the Tag to multiple objects;
- Placing different objects in front of the Tag.

The process yielded about 612 different measurements of the S-parameters (magnitude, phase, real and imaginary information), which were fed into different dimensionality machine learning algorithms such as SVM, Linear Discriminant Analysis (LDA), KNN

and Decision Trees. SVM with a quadratic function yielded the best results, with a 98.5 % accuracy, as previously mentioned. However, similar to Patel, Temple, and Baldwin's work, the paper does not approach SNR as a realistic metric to take into account, making it complicated to see it implemented in today's environment.

Riyaz et al. [48] used a type of Deep Learning algorithm, known as CNN, to sift through $20 * 10^6$ raw I/Q samples to identify devices with an accuracy of 98 %. The authors also discussed the impact of distance on radio fingerprinting and classification accuracy, specifically how path-loss and multipath fading are resisted by the algorithm to a certain point, maintaining an accuracy greater than 95 % up to 34 ft. (or 10.3632 meters) of distance from the reader. This is an important aspect when we consider the low power used and backscattered by passive radio frequency Tags, where distance and reading capabilities are one of the biggest challenges. Peng et al. [51] used this algorithm to uniquely identify 53 IoT (ZigBee) devices with an accuracy of 99.1 % and 93.8 % with SNR levels as high as 30 dB and 15 dB, respectively. This was done by feeding the algorithm a novel data type, a 2-D representation of a signal time series known as a Differential Constellation Trace Figure (Differential Constellation Trace Figure (DCTF)), a figure that features the device characteristics without the need for synchronization. Finally, this method was also compared extensively (Hilbert–Huang transform (HHT) transform and Sequential Information Bottleneck (SIB), to extract features) and used by Ding et al. [55] in order to obtain accuracy as high as 99 % and 61 % in SNR environments of about 30 dB and 0 dB, respectively. Using a local integral bi-spectrum technique with Fisher discriminants as a high-dimensionality fingerprint, the author used an adapted CNN algorithm to further reduce and parse through redundant information in order to obtain the high-level features hidden in the data.

2.4 Technology Summary

We will now review all the previously discussed results in a summarized format, in order to infer what are the best and most recent solutions to be tested in the implementation of the following project. Let's start by reviewing the chipless RF Tags.

Table 2.3: Chipless RF Tags summary.

Tag	Signature	Read Distance (m)	Frequency Range	bits	Printable
[26] Global SAW Tag	*TDR, PPM*	—	UWB	Size-Dependent	No
[4] TFTC/Organic Thin Film Transistor (OFTC)	*TDR, PPM*	—	UWB	Size-Dependent	Yes
[24] Multi Resonant	*FDR, S-parameters*	—	UWB	Spiral-Dependent	Yes
[32] Multi Resonant	*FDR, Channel State Information (CSI)*	—	UWB	12	Yes
[35] Octagon	*FDR, RCS*	1.5-3.5	UWB	Spiral-Dependent	Yes
[37] Dipole-Based	*FDR, RCS*	—	UWB	20	Yes
[42] Image-Based	*FDR, EM*	—	mm-Wave	Size-Dependent	Yes

As previously mentioned, TDR Tags, in general, while having greater read distance, are mostly non-printable, being that the ones who are printable are either very complex or still in the infancy stage of development (TFTC and OFTC [29]). With the addition that these tend to require greater reader protocol complexity, they are mostly used in

specific industrial environments. Frequency signature-based Tags favour a more universal approach when it comes to usability and ease of design (tend to support printable capabilities, as seen on Table 2.3). They mostly use multi-resonant spiral or dipole circuits, which are easily projected and printed. Furthermore, they tend to only be limited in encoding capabilities by the size of the Tag and printing resolution. However, actual implementation of these Tags in commercial settings is scarce since anti-collision methods for chipless Tags, in general, are a huge challenge and are mostly still in development [4] [39] [40]. Additionally, we can infer by the research done in Section 2.1.2, that the best combination of the substrate and conductive layer for this technology is either PET or paper, if we consider the sustainable aspect valuable in this case, and conductive ink such as silver.

Table 2.4: Machine Learning with Fingerprinting summary.

Algorithm	**Accuracy (%)**	**SNR (dB)**	**Feature type**
[53] Random Forests	99 - 42	30 - 0	*Transient*
[54] Supervised Learning	99	—	*Steady-state*
[57] Decision Trees	86.9	—	*Steady-state*
[57] KNN	97.2	—	*Steady-state*
[57] LDA	98.2	—	*Steady-state*
[57] SVM	98.5	—	*Steady-state*
[48] CNN	82	30	*Steady-state*
[51] DCTF-CNN	99 - 25	30 - 0	*Steady-state*
[55] CNN	99 - 61	30 - 0	*Steady-state*

Lastly, by gathering all the previous algorithms mentioned, as seen in Table 2.4, we can easily assess that the most robust algorithm and fingerprinting approach is a neural network based solution like a CNN trained through steady-state features, presenting a high level of accuracy even in high-noise environments in the case of an SNR of 0 dB. It can also be said that CNN has shown promise in RF Tag capabilities thanks to its strong response in terms of identification accuracy and large distances [48].

By combining all of these technologies, we hope to facilitate or even solve the problem of anti-collision in multiple object reading scenarios along with a better readability of state-of-the-art radiofrequency Tags through a machine learning solution such as a CNN fed by RCS or other steady-state features.

3

Project Overview and Operation Concepts

This chapter has the objective of providing an overview of the project's key operating principles and theoretical concepts.

3.1 Dissertation Project Overview

The following work aims at the development and training of a ML algorithm using simulation data from a chipless RFID Tag, that can distinguish and classify multiple and unique chipless RFID Tags at different distances from the RFID Reader. The chipless Tag in focus is Betancourt et al.'s aforementioned (see Chapter 2) back-scattered Tag [35]. This tag features octagon shaped resonators that encode it's unique ID in the frequency domain. Various algorithms will be trained, and compared, through the RCS signature of the Tag which will be extensively analysed and simulated on a diverse range of situations, be it environmental or manufacturing based. In terms of anti-collision, the system will depend on the work of Lopes and Matos's "Simulation of a Chipless RFID System using discreet FrFT to recover individual tags IDs" [40], which proposes the use of Fractional Fourier Transforms to window signals in a fractional domain avoiding time and frequency overlapping. This entire system can be broken down into two main blocks: the Tag and the Reader (which is comprised by the machine learning classifier and the anti-collision logic). In order to fully implement and understand this system, this chapter will be dedicated to explaining the several theoretical aspects that these "blocks" entail, in a more in depth fashion.

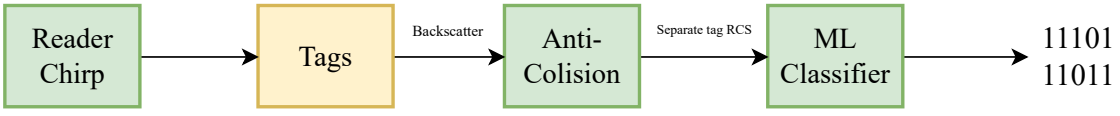


Figure 3.1: Flowchart of the overall system structure.

3.2 Backscattered Chipless Tag

The Tag is vital in this system as it changes not only the typical fabrication method but the traditional reading protocol for RFID Tags, given its chipless nature. The backscattered chipless Tag is typically defined as a frequency-based signature Tag, meaning its unique identifier is encoded through a frequency-based spectrum. The 'backscattered' property defines how the Tag does this: through several resonant structures that follow the FSS methodology [33]. This method employs several formulas to carefully design resonant geometric surfaces who through their dimensions create "gaps" in the electric field surrounding them at specific frequencies. This allows a simple paper substrate to contain these easily ink printable structures and encode a sequence of 'notches' in the electric field spectrum that can then be read as a code or unique identifier of a Tag when hit by a sweep signal. Let us look at the Tag we will be implementing, based on the work of Betancourt et al. [35], in order to better understand this effect. The phenomenon starts by the impulse the Reader sends out, also known as the interrogation signal or chirp, this induces current in the structures, given that these are octagons and can work as singular coils. Each one of these coils has their resonant frequency, that is, the value of which the electric field reflected by them is at its maximum. This frequency is dependent on the size and geometry of the resonant structure. The currents induced in each ring are similar in intensity at their resonant frequency but are phased at a level of π, making it so that every couple of octagons, the electric fields cancel each other out in between the respective resonant frequencies of the couple, thus creating the notch effect visible in the frequency spectrum.

This 'notch' classically represents the presence of a bit or a '1', and the absence of the notch represents a '0'. Here, this absence is obtained through what the author called the 'late-stage coding ability', the action of bridging the gap between two resonant structures to set the opposite current direction into a similar one. This nulls the cancelling effect of the conflicting electric fields of the structures. The 'late-stage' aspect is implemented through the fact that the gap bridging could be done with either an ink printing machine or even a conductive ink pen in order to change the code of the Tag, making it easily adaptable through a number of situations. The signature chosen for the Tag identification is the RCS spectrum of the Tag, as we have seen in Chapter 2, it tends to supply the most stable signature in the face of noise and spatial variation. The implemented Tag will allow a 5-bit coding capacity, through 6 octagon-shaped resonators. With the initial block defined, the Tag, let us review the fundamental point of this project, the classifier, or more specifically the machine learning algorithms that will be employed and compared.

3.3 Chipless RFID Reader

The Chipless RFID Reader, is, as previously mentioned, vital to this process as it needs to adapt to the type of Tag it needs to read, since this one presents such a different encoding

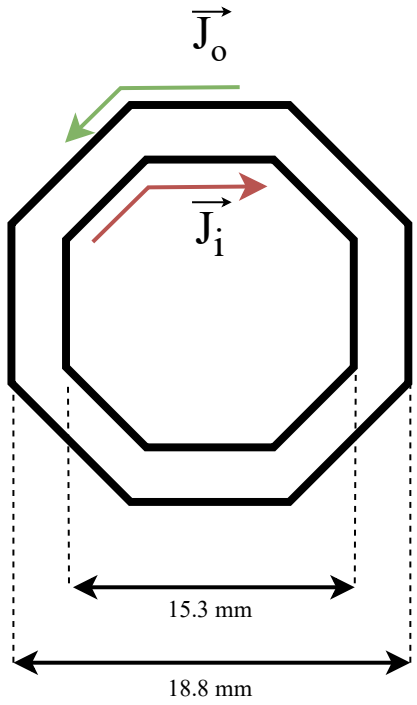


Figure 3.2: Surface current direction of two octagon structures.

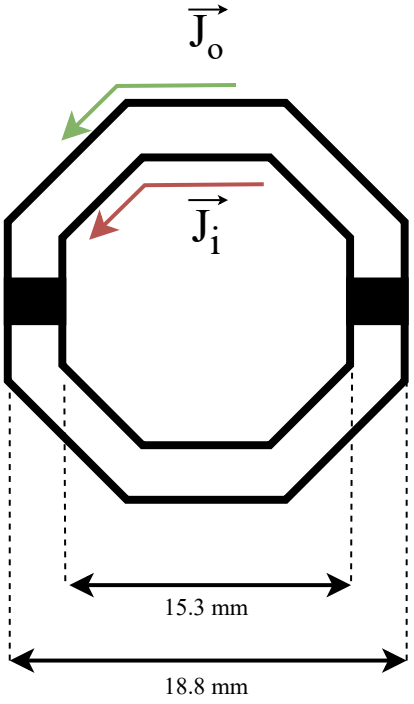


Figure 3.3: Surface current direction of two 'bridged' octagon structures.

method than conventional and modern chipped RFID Tags. The main difference being the information with which it classifies each Tag, that being, a steady-state frequency signature rather than a modulated series of pulses in the chipped case, and the lack of anti-collision protocols present in chipped RFID Tags, leaving the work of separating and avoiding collision fully up to the Reader. This ultimately, as mentioned in Chapter 2, heavily simplifies the manufacturing process of the Tags and thus, reduces the price of fabrication. However, it tends to make the Reader more complex in operating principles and concepts, which we will be now discussing. Let us start by reviewing the classifier needed to identify the Tags. Since the information we are working with in terms of the classifier input is quite novel in the RFID world, we must build an equally novel classifier. To do this, we will use a variety of algorithms to classify the Tags based on their RCS signature and compare their performance in order to find the best classifier possible for this specific fingerprint. To compare these algorithms and to properly treat the data in order to train them, we must first review the tools needed to analyse the validity and variance of the dataset. To do this, we used PCA.

3.3.1 Principal Component Analysis

PCA is an unsupervised learning algorithm whose main uses tend to be as a data analysis and dimensionality reduction tool. It's used as such, as the 'unsupervised learning'

component of it allows the automatic clustering of data points without explicit labelling. Meaning, as a data analysis tool, we can look at how similar or varied each example is based on the code it belongs to, which can illustrate how balanced and varied each label is, which in turn facilitates the separation and classification of each label for each classifying algorithm. In terms of dimensionality reduction, when we consider that in an entire dataset there are parts or components of an example that might not differ a lot or at all from each other in the same label, we can easily see that these points are just consuming time in the training process as the algorithms aren't learning anything truly new with these similar examples. Since PCA can find the direction of the highest variability in the data, we can build a dataset that holds a more varied representation of each example, successfully reducing the number of dimensions and truly representing the most important components of each example of a dataset.

Now that we have described the main analysis tool used for the classifier construction, let us now briefly review all of the classifying algorithms implemented and compared.

3.3.2 Logistic Regression

Logistic Regression can be purely defined as a statistical model that allows the classification of some input data as a discrete label that is known (making it a supervised learning algorithm), through the attribution of probability. Meaning, it typically approaches data and assigns the probability of it either being the true label or not. It can focus on the existence of one label (Binary Logistic Regression) or the existence of multiple (Multinomial Logistic Regression), which is the case with our problem. The attribution of probability to each example, x, is achieved through the use of what is called the logistic function:

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \tag{3.1}$$

β_0 is known as the vertical intercept and β_1 as the rate parameter; these are the parameters that fit to the data the logistic regression model receives, in order to present the best classification accuracy. This action of fitting is mostly universal to all algorithms we will be discussing, so let's dive deeper. A common measure for the approximation of a function that can accurately classify the data it receives, also known as the fitting process, is the loss function. This attributes a value of error to the predictions made using an algorithm. By outputting a probability, we can measure the difference between what was guessed and the correct answer, setting the overall goal to minimize this difference through the variation of our parameters (β_0 and β_1, in the case of logistic regression) so they can be a good fit for our data. The loss function of logistic regression for every n example can be defined by using a likelihood function such as [58]:

$$\prod_{n=1}^{N} p_n^{y_n} (1 - p_n)^{1 - y_n} \tag{3.2}$$

where y_n is the labelled value and p_n is the probability predicted by our logistic function. We can then define the error or loss function by taking the inverse logarithm of this previous equation, resulting in:

$$l = \sum_{n=1}^{N} (y_n \ln(p_n) + (1 - y_n) \ln(1 - p_n)) \tag{3.3}$$

We can then derive our parameters from the loss function, equating them to zero, in order to find the local minimum of loss pertaining to each of them. In the case of logistic regression:

$$0 = \frac{dl}{d\beta_0} = \sum_{n=1}^{N} (y_n - p_n) \tag{3.4}$$

$$0 = \frac{dl}{d\beta_1} = \sum_{n=1}^{N} (y_n - p_n) x_n \tag{3.5}$$

where x_n represents our input data. This concludes the fitting process, successfully implementing a generalised classifier of the logistic regression model.

3.3.3 Gaussian Naive Bayes

The Gaussian Naive Bayes (GNB) classifier is grounded on the Bayes Theorem of conditional probability, where it is assumed that the values associated with every label are distributed through an independent Gaussian distribution (no covariance, thus the naive label). with n While it is a simple algorithm, it is known to be efficient in the training or fitting process, needing little data to converge. The model is simply fitted by finding the variance and mean deviation of every class and proceeding to calculate the Z-score distance, found in equation 3.6, of the input data to each class y, attributing the prediction to the closest class.

$$Z_{score} = \frac{x - \mu_y}{\sigma_y} \tag{3.6}$$

$$\mu_y = \int_{-\infty}^{\infty} x_y f(x_y) dx \tag{3.7}$$

$$\sigma_y = \sqrt{\int_{-\infty}^{\infty} (x_y - \mu_y)^2 f(x_y) dx} \tag{3.8}$$

3.3.4 Linear Discriminant Analysis

LDA functions as a classifier through a set of assumptions similar to GNB, by using the mean and covariance of every class to determine the most probable one, however what

varies here is the classification rule used, rather than using the Z-score, something known as the discriminant function is applied instead (3.9).

$$\delta_y(X) = X^T \Sigma^{-1} \mu_y - \frac{1}{2} \mu_y^T \Sigma^{-1} \mu_y + \log \pi_k \tag{3.9}$$

where X is a vector of all the x input data. The ”linear” attribute is in reference to the boundaries between each class being of linear relation (the discriminant function itself is linear in relation to X), which, while fast, supplies a rudimentary fit of the data.

3.3.5 K-Nearest Neighbors

KNN, similar to all the classifier algorithms we'll be using, is a supervised learning algorithm commonly used for classification, which uses proximity as the determining factor for prediction making. Similar to LDA and GNB, the decision is based on a distance metric, however this is based on the actual values rather than a mean or covariance. The most commonly used metric is the Euclidean distance formula (3.10):

$$d(x, y) = \sqrt{\sum_{z=1}^{Z} (y_{1,z} - x_{1,z})^2} \tag{3.10}$$

where Z is the dimensionality of the data.

3.3.6 Random Forests

The RdF classifier is based on the use of multiple random decision trees, known as an ensemble, in order to create a sum of predictions, predicting the class that has the biggest number of votes. The decision trees are able to break down the features that best describe the subgroups or clusters found in the data, while having the potential of being uncorrelated to each other. In basis, a decision tree is formed through a series of nodes that connect downward through branches in order to specify the data to then make decisions, seen in Figure 3.4. The tree is formed first by the root node, which is the starting point and a representation of the whole dataset of examples. This is then split by two decision nodes and those two can branch off in another split (which would constitute a branch or sub-tree) or end in two terminal nodes.

This structure can be uniquely generated various times with the same dataset, hence the use of several decision trees in the final prediction in order to level out any error margin.

3.3.7 Support Vector Machine

SVM is an algorithm that attempts to best draw one or multiple straight lines that hold the maximum distance between data points of multiple classes, thus separating them in order to assign them to input data. The algorithm can support a large number of features to separate, through the use of a multidimensional plane, also known as a hyperplane.

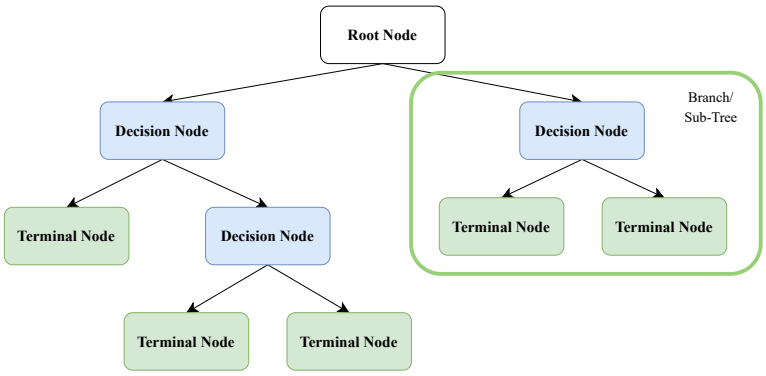


Figure 3.4: Structure of a typical decision tree.

The main hyperparameter of this algorithm is defined as the margin between the data points and this hyperplane, the goal being the maximization of it. This is done through the loss function known as hinge loss (Equation 3.11).

$$l(x, y, f(x)) = (1 - y \cdot f(x)) \tag{3.11}$$

where x is our input, y our true label and $f(x)$ the prediction the algorithm made. While this loss function could work, we must add a regularization parameter in order to balance the loss and margin maximization.

$$l(x, y, f(x)) = min_w \lambda \|w\|^2 + \sum_{i=1}^{n} (1 - y_i \cdot f(x_i)) \tag{3.12}$$

where λ is a regularization parameter and w represents the weights that will be used to fit the function. Now that the loss function is defined, we can look to the gradients in order to update these weights.

$$\frac{d}{dw_k} \lambda \|w\|^2 = 2\lambda w_k \tag{3.13}$$

$$\frac{d}{dw_k}(1 - y \cdot f(x_i)) = \begin{cases} 0 & \text{if } (1 - y_i \cdot f(x_i)) \geq 0 \\ -y_i x_i & \text{else} \end{cases} \tag{3.14}$$

we can then take these and according to the prediction being true or false, we can update the value of w, as so:

$$w = \begin{cases} w - \alpha \cdot (2\lambda w), & \text{if correct} \\ w + \alpha \cdot (y_i \cdot x_i - 2\lambda w), & \text{if wrong} \end{cases} \tag{3.15}$$

When the training set has been then fully fed, this defines the training process of the SVM algorithm, and it is ready to predict on the new data.

3.3.8 Multi-layer Perceptron and Neural Networks (NN)

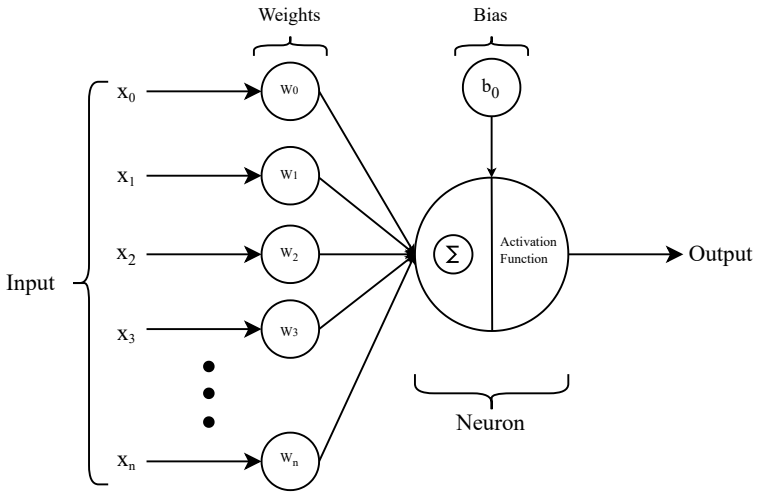


Figure 3.5: Structure of a neuron in a neural network.

A Multi-layer Perceptron (MLP) classifier is, at it's core, a fully connected feedforward Artificial Neural Network (ANN), so in a lot of ways, when one mentions an MLP classifier they are typically talking about the classical definition of a neural network, that is a neuron-like algorithm comprised of at least three layers: an input layer, a hidden layer and an output layer. The training process is done through a method called back-propagation. This is the evaluation of the predicted class in comparison to the known correct prediction, in order to alter the various weights that comprise a neural network. Each neuron has a weighted input, that is, a numerical coefficient that is multiplied at the entry of the neuron, followed by an activation function inside the neuron, seen in Figure 3.5. The most common activation function being the Rectified Linear Unit (ReLU) function, defined by:

$$y(z_i) = max(0, z_i) = \begin{cases} 0 & \text{if } z_i < 0 \\ z_i & \text{if } z_i \geq 0 \end{cases} \tag{3.16}$$

These weights are updated on each run of the training process through backpropagation, working as a hyperparameter of the algorithm. Another hyperparameter, is the bias term that is also added into the activation function.

3.3.9 CNN

While there are various architectures, a basic CNN (based on the original LeNet-5 architecture by Lecun et al. [59]) can be broken up into a 3 layer pipeline:

1. Convolutional Layer: this layer defines the fundamental function of the network, since it's where most of the computation occurs. This layer receives the raw data, which will be received by a feature detector, who detects patterns in the data

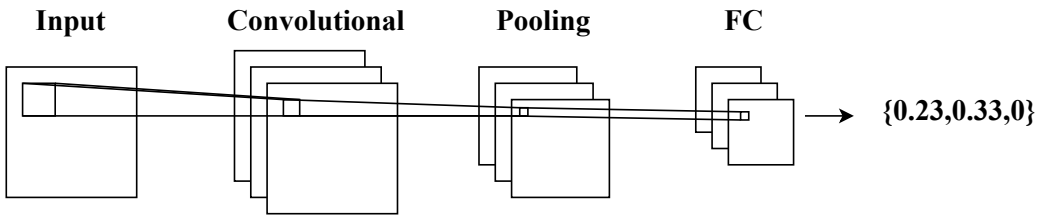


Figure 3.6: Illustration of the LeNet-5 CNN architecture.

through a process known as convolution. This is typically a 3x3 filter, whose weights are eventually to be corrected through back-propagation, that is applied to a section of the data followed by an internal product between the filter's inputs and the original data. The filter is then moved and reapplied until the entire data has been covered. This outputs a feature map, that will be transformed through a ReLU, or rectified linear unit activation function.

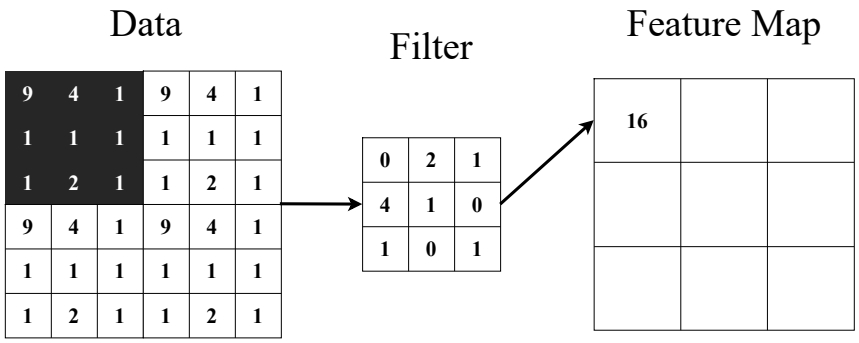


Figure 3.7: Illustration of the convolution process.

This layer needs to be defined initially and then adjusted through the following parameters:

- Number of filters: This is dependent on the dimensionality of the data since there needs to be a filter for each layer of data, for example, an RGB colour image has 3 layers thus might use 3 individual filters;
- Stride: The distance the filter moves on each iteration of convolution;
- Padding: Gives the ability for the filter to adapt to mismatched dimensions in the data.

2. Pooling layer: The pooling or sub-sampling layer is also a highlight to the CNN algorithm, since it provides the ability to reduce the obtained features to the most abstract and vital. This layer also applies a filter, however, this filter doesn't contain dynamic weights, instead, it applies a function to the feature map that selects the

maximum value covered by the mask and outputs it, followed by another application of the ReLU function. Alternatively, the function can also compute the average of the covered elements.

3. Fully-connected (FC) layer: the fully-connected layer or FC layer, aims to classify the given features supplied by the previous layers of pooling and convolution, meaning to attribute a value of 1 to 0 (probability) through a softmax function, instead of a ReLU . This is also where the neural network is found and trained by the extracted data.

While there are 3 types of layers, the pooling and convolutional layers can be repeated in order to extract even more abstract features. CNN's success in various areas has motivated several articles to use this technology, making one of the most popular AI technologies at the moment. This is due to the reduced dimensionality it provides in the output data, while still maintaining high levels of scalability, evidenced by previous examples taking on 6 orders of magnitude worth of data [48].

3.3.10 Anti-collision

The anti-collision algorithm and process we will be implementing is based on the work of Lopes and Matos [40], where they describe a method based on the Fractional Fourier Transform which allows a windowing of the signal that does not belong to a domain of either frequency or time, preventing any overlap in any of these realms. This is vital since the few works on anti-collision protocols for chipless RFID relied heavily on beam forming technology, that, while becoming ever so accessible with the rise of 5G [60], can not accurately prevent the reading of two Tags hit by the same beam. This work expands on this problem, enabling the separation of the individual ID's found in the same beam. For us to understand this, let us examine a fundamental block of RFID Reader operation, the Linear Frequency Modulated (LFM) signal.

3.3.11 Linear Frequency Modulated signal

An LFM signal is, simply put, a signal that varies its frequency over a span of time. These signals are commonly used in the radiofrequency ethos as interrogation signals, or chirps, as they sweep a specified range of frequencies that the device they are contacting or locating responds to, in the case of Betancourt et al.'s Tag [35], from 1 GHz to 20 GHz. This pulse is then, typically, reflected by the device with its information convolved inside it, enabling the retrieval of its ID. A signal analyser can be used to obtain, generate and read this signal.

3.3.12 Fractional Fourier Transform

The Fractional Fourier Transform, or FrFT, can be defined as a linear Fourier transform applied to the nth power. Here it is used as a rotation operation of the time-frequency

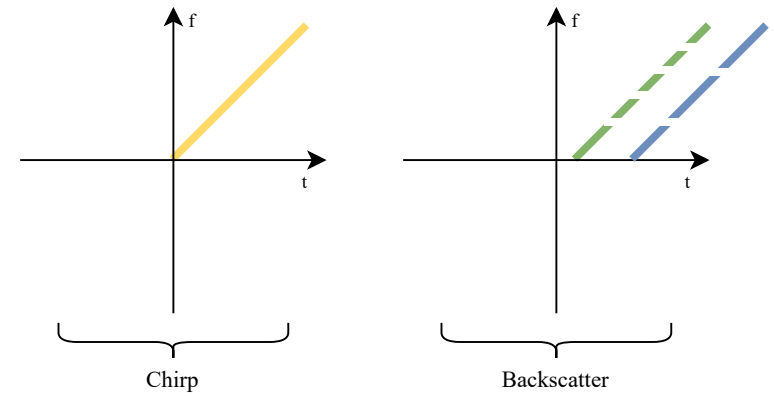


Figure 3.8: Illustration of chirp signal and it's respective backscatter (a response of Tags with code 1111 and 1010).

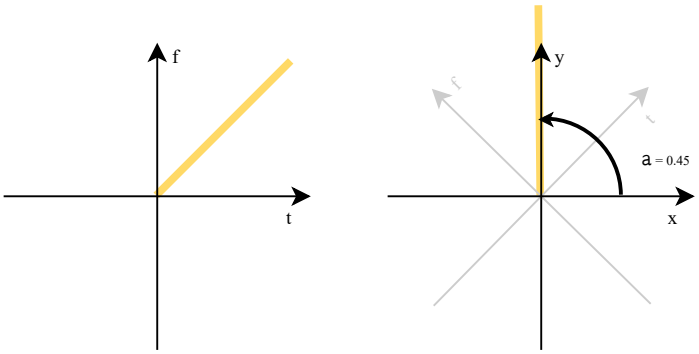


Figure 3.9: Illustration of the Fractional Fourier Transform with a rotation of alpha 0.45.

plane in the function of an alpha value, that serves as the angle of rotation, as shown in Figure 3.9. The definition can be seen in equation 3.17.

$$F_a(u) = \sqrt{1 - icot(\alpha)}e^{i\pi cot(\alpha)\mu^2} \int_{-\infty}^{\infty} e^{-2i\pi(cst(\alpha)ux - \frac{cos(a)}{2}x^2)} f(x)dx \tag{3.17}$$

By using FrFT we can properly compact an otherwise "angled" signal into a single direction, making windowing as simple as choosing a value of x or y and preserving about 30 samples on each side. The operation is then reversed by applying the same transform with a negative alpha, and the windowed signal is transformed back into frequency vs. time.

Having defined the main functioning blocks, let us give a final overview of the anti-collision system. The Reader sends out an LFM signal in a broad direction, waiting for a backscattered signal from the Tags. The Tags, as they receive the signal, each convolve their frequency signature onto the sweep and reflect it back at phased times since they

are at different physical distances from the Reader. A signal analyser is used to view these and applies a Fractional Fourier Transform in order to compact the reflected signals into a single direction, in order to facilitate the windowing of each signature. When the windowing is completed, the FrFT is reversed and each individual signature is recovered and, thus, successfully isolated, ready to be fed into a classifier and identified.

4

Implementation and Simulation

This chapter will primarily focus on the implementation, simulation, and discussion of results of the previously discussed system.

4.1 Dataset Generation and Preparation

In the ethos of ML and AI as a whole, one can argue that the most fundamental part to the successful implementation of an algorithm is the data we feed it and how it is processed and labelled [61]. The data chosen here is the RCS signature of the octagonal tag, based on Betancourt et al.'s work. The simulation and modelling environment chosen, for the tag implementation, was CST Studio Suite, as seen in Figure 4.1 and 4.2. It was chosen for it's several electromagnetic simulation options and extensive graphing and computing abilities (use of Maxwell Equations). The program provides several parametric sweep options, that make the generation of the data for the test set of the algorithm easy and fast, while still providing an extensive analysis of far-field and near-field properties of the tag.

This was followed by the simulation of the tag's RCS while changing the following parameters, as seen in Figure 4.3:

- Reader distance from the tag (0 – 100 millimetres);
- Obstructed by paper;
- Multiple incident angles;
- Ink thickness tolerance (5 %);
- Substrate thickness tolerance (5 %).

Most of these parameters are indifferent to the inductance of the tag, and heavily depend on the situational context of the application, meaning, the variation of these parameters focuses more on the position of the reader and tag itself rather than the variation of parameters due to fabrication errors and such. The reason for this is that fabrication

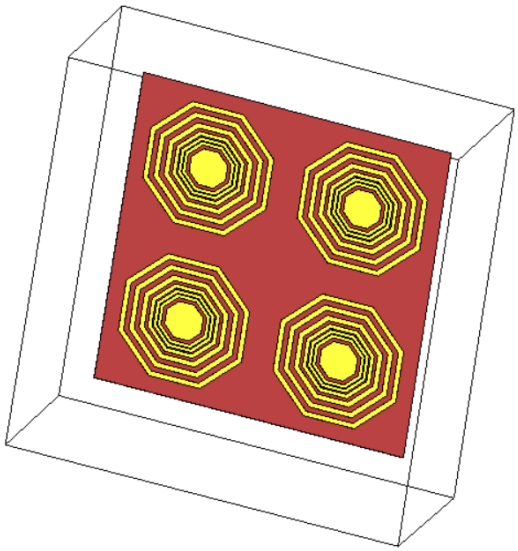
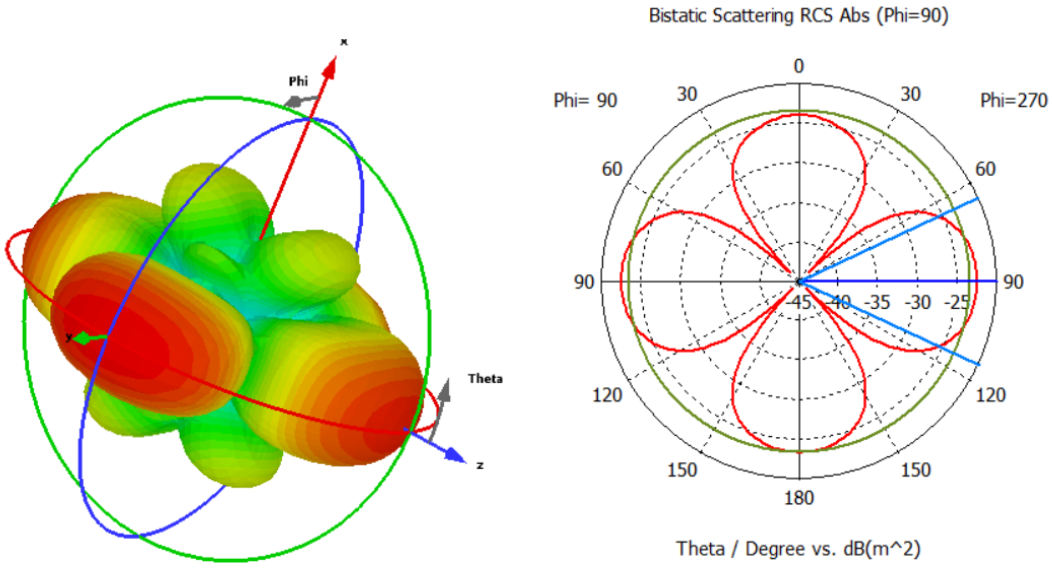


Figure 4.1: Simulated model of the octagonal chipless RF tag.



(a) Farfield radiation of tag

(b) Bi-scattering RCS of the tag

Figure 4.2: RCS characteristics of the Octagonal Chipless RF tag.

errors tend to be less and less with the introduction of new fabrication technology of the tags, and most of the error going forward will be more dependent on human and contextual error of RFID applications (i.e: poor placement of the tag, obstructed tag, etc.). However, the fabrication process will also be considered with a variation of the ink and substrate thickness which, as previously mentioned in Section 2.1.2, has influence on the resonant frequency of the "dips", and can produce further results to better train the algorithm [61]. This process was then recreated for 6 differently coded tags (11111, 11110, 11101, 11011, 10111, 01111) to finally obtain a balanced dataset of about 4560 unique examples. Each signature provided is limited in frequency from 1 GHz to 20 GHz, and has an RCS quantified in dBm^2, an absolute metric used to measure power in relation to 1 mW.

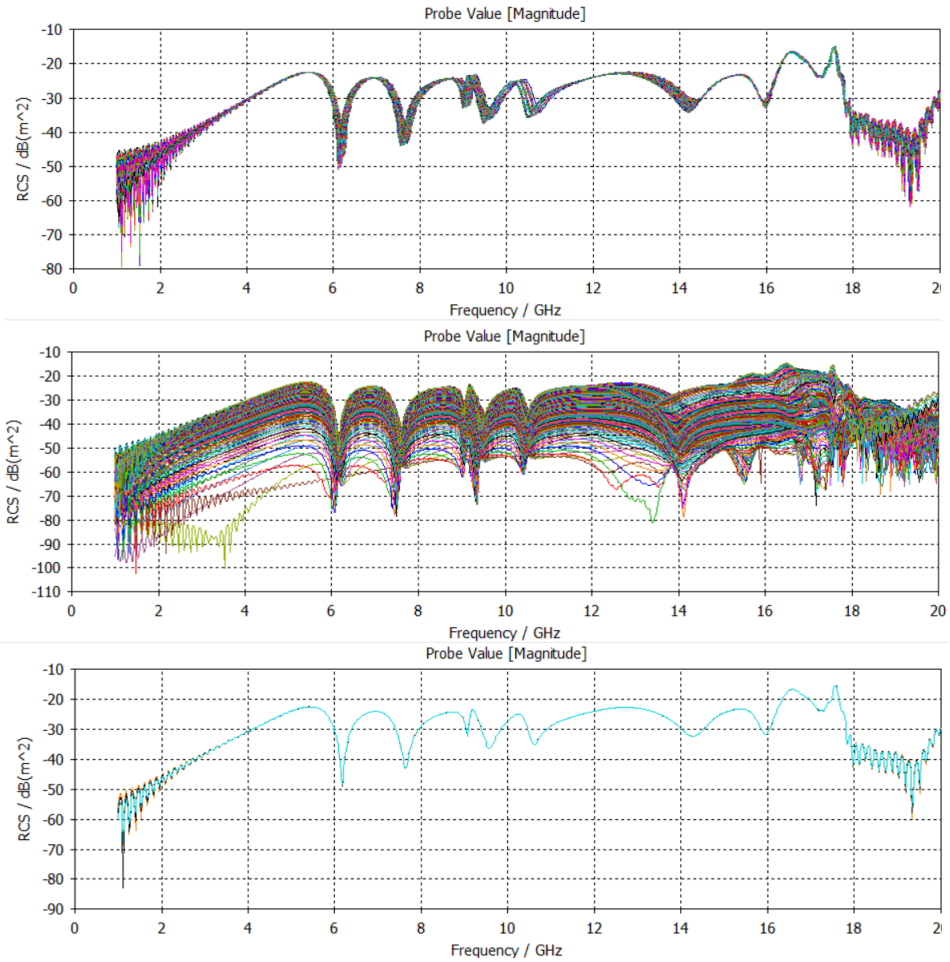
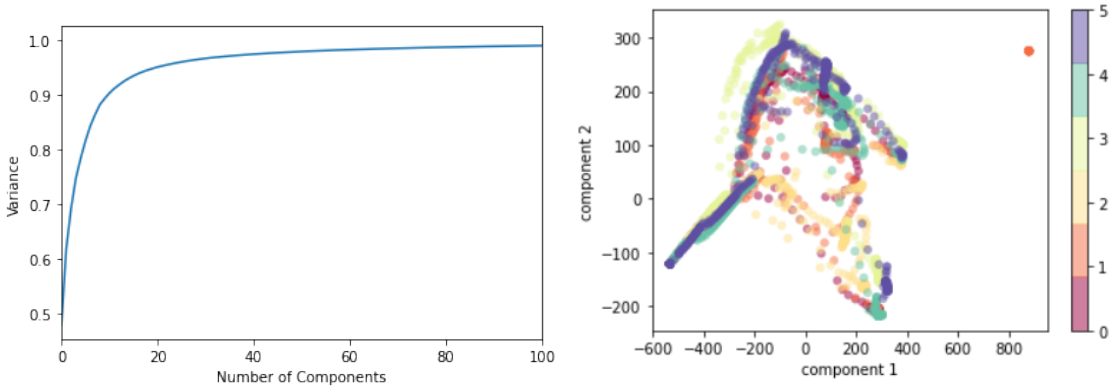


Figure 4.3: Parameter sweep of the substrate thickness, angle of incidence and silver thickness (code "01111").

The environment chosen to handle and manipulate this dataset was Python, and it's many data oriented libraries such as NumPy [62] and Matplotlib [63]. This choice was also motivated by the existence of it's many easy-to-use and highly efficient machine learning focused libraries [64], of which Sci-Kit Learn [65] and Keras [66] were chosen to

implement the algorithms. The data was originally processed by reading the ASCII data exported from the CST Studio Suite, line by line, and archiving it into a NumPy array, where each signature would have its corresponding label attached to it. Ensuing was a PCA analysis of the dataset: first to analyse the number of components necessary to reach 98 % total variance of the dataset, followed by a two-dimensional visualization of the data's variance by label.



(a) Total variance vs. Number of Components

(b) 2 Dimensional PCA Analysis

Figure 4.4: Principal Component Analysis of the initial dataset.

As it can be easily assessed, the PCA total variance tells us that about 98 % of the dataset representation can be fit into less than 100 components, effectively reducing not only the training time but the prediction time for algorithms such as KNN and SVM that use the correlation of each example, which can be very time-consuming down the line. By looking into the 2 dimensional analysis, we see that there is a good amount of data overlapping on the left-hand side, this could be caused by the unnecessary data found on the extremities of the actual "bit" encoding portion of the signature (3 to 12 GHz). Additionally, it is possible to see in the upper right corner a small concentration of outlier data, which when plotted showed to be the very rare examples of when the CST presented little to no amplitude data, resulting in a highly different signal data-wise. This PCA analysis became quite recurrent during the simulation process since as it can be seen it informs on how the dataset will be processed and seen by the machine learning algorithms. This prompted the erasure of the outlier data, resulting in a total number of 4534 examples, a reduction of the frequency range to only be from 3 GHz to 12 GHz and data normalization in order to better separate each example. While this only represents a tenth of the final dataset used (read distance and SNR simulation creates a dataset for each value set), it's important to understand the use of these tools.

4.1.1 Read Distance and SNR Simulation

In Chapter 2, in the definition of RCS we found that there is a read distance attached to each value of RCS, meaning each value has a minimum read distance value associated

with it, in order to reach the reader (see Equation 2.9). In order to compute these values for each signature, the variables were defined as:

- $G_{Tx} = 1$;
- $G_{Rx} = 1$;
- P_{min} = -70 dBm;
- P_t = 3 dBm.

This resulted in the plot shown in Figure 4.5, in which, most of the signal is conserved if we consider that the reader is about 1 meter away from the tag.

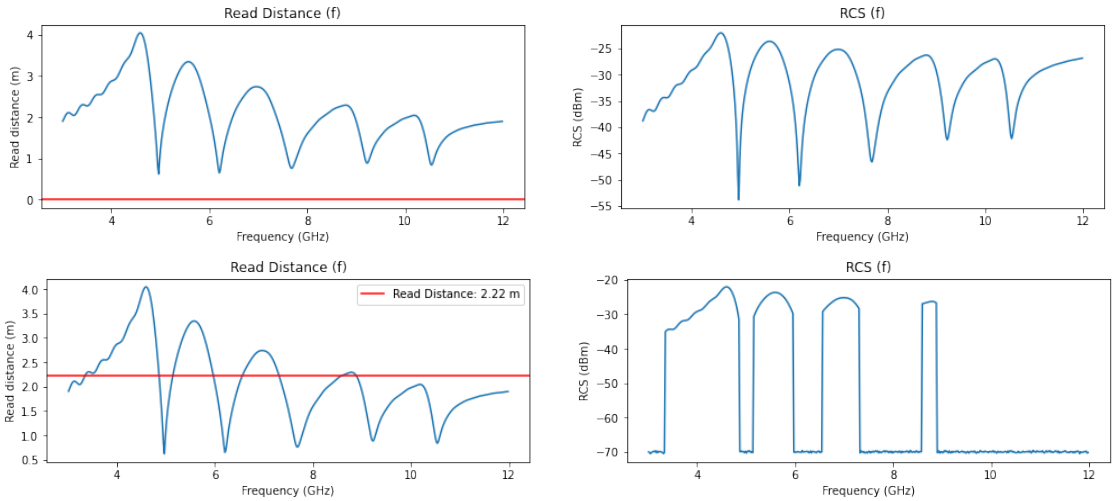
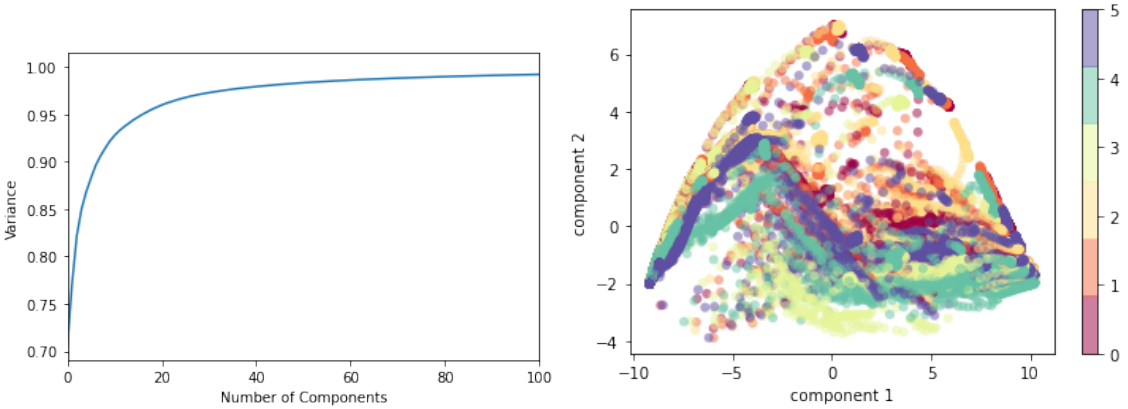


Figure 4.5: Read distance limits and their corresponding RCS signature.

However, if we were to consider the tag further away, 2.22 meters for example, only the ”hills” would be obtained by the reader. So the further away the reader is, the less information is ”seen” by the reader. So in order to train the algorithms and evaluate them, 10 datasets were created from 0 to 4 meters, in which each dataset corresponded to a different read distance.

This resulted in the PCA analysis of this new combined dataset, seen in Figure 4.6. While the total variance by number of components hasn't changed significantly, we can see a large difference in the 2D PCA given the size of the new dataset. The simulation of SNR, was done in the reading process of the data adding Gaussian white noise, given a specified SNR level. The ratios chosen were from 0 to 30 dB, creating 10 datasets similar to the read distance variation dataset, as shown in Figure 4.7.

Here the dataset needs quite a lot more components to reach 98 %, about 204 components, more than double what would be needed in the previous analyses (see Figure 4.8). However, the 2D analysis shows a similar shape to the singular dataset, with an obvious slight variation in each example due to the noise added.



(a) Total variance vs. Number of Components (b) 2D PCA

Figure 4.6: Principal Component Analysis of Read Distance dataset.

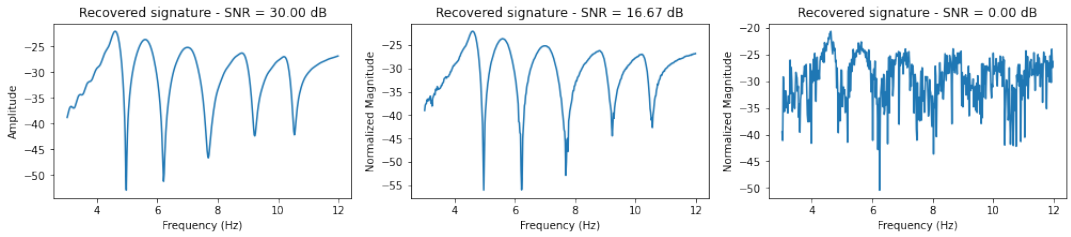
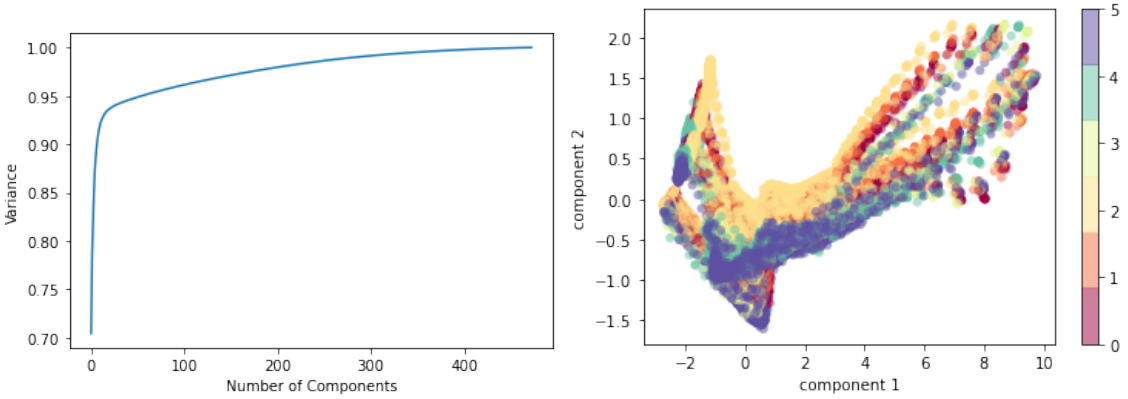


Figure 4.7: SNR generation for the RCS classifier.



(a) Total variance vs. Number of Components (b) 2D PCA

Figure 4.8: Principal Component Analysis of SNR dataset.

4.2 RCS Classifier

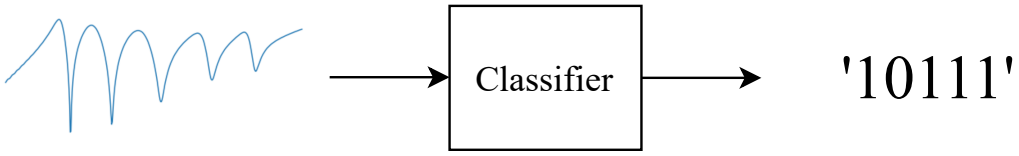


Figure 4.9: Illustration of the RCS classifier process.

The RCS classifier presents itself as the core and main subject of this investigation, providing the ability to classify tags based on their RCS signature, normally obtained by the reader. This classifier presents a logistic regression problem, more specifically, a multi-class classification problem. In order to implement this classifier, several algorithms were employed and compared in terms of detection accuracy facing a range of read distances and SNR values. These algorithms also range in terms of what is called, in the machine learning ethos, as the Big O notation. This concept describes the growing complexity and resource use of an algorithm depending on the size of it's input. This was done to better visualize and to justify the use of more complex algorithms further in the project. The following algorithms are used:

- Logistic Regression;
- GNB;
- LDA;
- KNN;
- RdF;
- SVM;
- MLP;
- ANN (deep NN);
- CNN.

The implementation and parameter definition of these algorithms was mostly streamlined since, as previously mentioned, Python has many efficient ML oriented libraries, of which Sci-kit Learn and Keras were used. As soon as these algorithms were chosen, a "pipeline" was implemented for each algorithm that contained an initial PCA process that would use a simpler version of the data corresponding to 98 % total variance of the dataset, making the training process not only easier on the algorithms, but faster as well. The training process was carried out by using a technique known as K-Fold Cross Validation, which divides the dataset into equally sized K groups of data. This technique then

proceeds to pick one of these groups for prediction and the rest for training, repeating this process and iterating through the entirety of the K groups, as seen in Figure 4.10. This allows for a fuller undertaking of the dataset, being a very popular technique when dealing with smaller datasets.

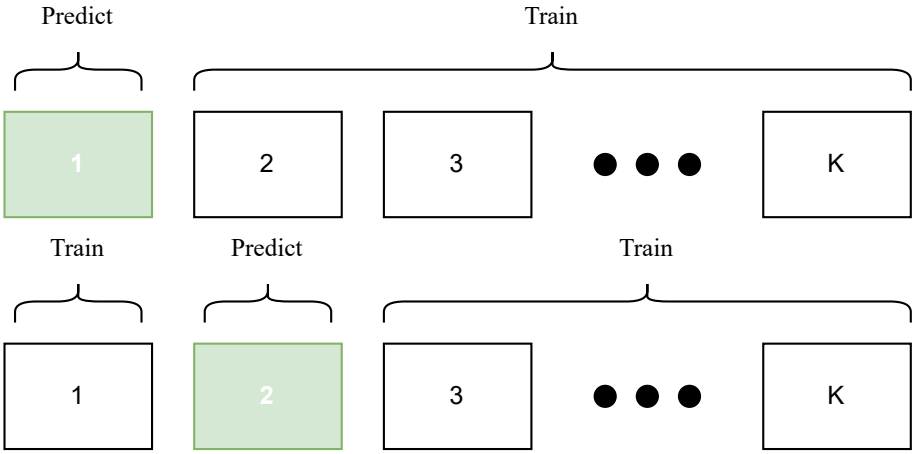


Figure 4.10: Illustration of the K-Fold Cross Validation process.

For this project, this process was applied with K=5. Following the training procedure, a process known as Hyperparameter Tuning is typically involved. Hyperparameters tend to be unique for each algorithm and can dramatically alter its overall performance. These would be parameters like the number of hidden units in a neural network layer, or the number of estimators in a Random Forest algorithm. This process can be very time-consuming, especially when attempting to compare the best of 9 different algorithms. However, this process is, again, streamlined by a Sci-kit Learn function known as GridSearchCV. GridSearchCV takes a dictionary of parameters set by the user for a machine learning classifier, and proceeds to iterate through this dictionary changing the settings of the classifier, followed by K-Fold cross validation. The function then evaluates each combination of parameters given, and outputs the best one based on its accuracy score. While there are 3 datasets in total to evaluate and train the algorithms (Read Distance, SNR and SNR in the anti-collision context), a set of parameters were ultimately defined for each algorithm after running them through multiple times and seeing no significant changes.

```
pipe_svc = Pipeline([('pca', PCA(0.98)),
            ('svc', SVC(kernel='rbf', random_state=42, C=50, gamma=0.005))])


pipe_MLP = Pipeline([('pca', PCA(0.98)),
            ('MLP', MLPClassifier(random_state=42, activation='tanh', max_iter=300))
                ↪ ])
```

```
pipe_rf = Pipeline([('pca', PCA(0.98)),
            ('rf', RandomForestClassifier(random_state=42,n_estimators=250,
                ↪ max_depth=20))])

pipe_knn = Pipeline([('pca', PCA(0.98)),
             ('knn', KNeighborsClassifier(n_neighbors=25, leaf_size=1, p=1))])

pipe_gnb = Pipeline([('pca', PCA(0.98)),
             ('GNB', GaussianNB())])

pipe_LDA = Pipeline([('pca', PCA(0.98)),
             ('LDA', LinearDiscriminantAnalysis())])

pipe_LogReg = Pipeline([('pca', PCA(0.98)),
                ('LogReg', LogisticRegression(random_state=42, max_iter=2000, C=10.0)
                    ↪ )])
```

Listing 4.1: Code for the Sci-kit pipelines for each algorithm.

The parameters can be seen in Listing 4.1, where a class of object known as a Pipeline, from Sci-kit, was implemented. The Pipeline is, in summation, a literal pipeline in which the user can input several data processing techniques or classifiers in order to create a customized process for classification. In this case, an initial PCA process was implemented with 98% variance of the dataset, in order to speed up the training operation for algorithms such as KNN and SVM. This was done for every algorithm with the exception of CNN and ANN, since they were implemented through Keras.

```
## ANN implementation
modelNN = Sequential()

n_cols = data_norm.shape[1]

DPlabels = to_categorical(labels)

modelNN.add(Dense(250, activation='relu', input_shape=(n_cols,)))
modelNN.add(Dense(250, activation='relu'))
modelNN.add(Dense(250, activation='relu'))
modelNN.add(Dense(6, activation='softmax'))

modelNN.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['
    ↪ accuracy'])

early_stopping_monitor = EarlyStopping(patience=2)
modelNN.fit(data_norm, DPlabels, validation_split=0.2, epochs=30, callbacks=[
    ↪ early_stopping_monitor])
```

```
## CNN implementation
modelCNN = Sequential()

CNNdata = data_norm.reshape(-1, data_norm.shape[1], 1)

modelCNN.add(Conv1D(2, kernel_size=11, activation='relu', input_shape=(n_cols,1)))
modelCNN.add(MaxPooling1D(pool_size=2))

modelCNN.add(Conv1D(4, kernel_size=7, activation='relu'))
modelCNN.add(MaxPooling1D(pool_size=2,padding="same"))

modelCNN.add(Conv1D(8, kernel_size=11, activation='relu'))

modelCNN.add(Flatten())

modelCNN.add(Dense(250, activation='relu'))
modelCNN.add(Dense(250, activation='relu'))
modelCNN.add(Dense(250, activation='relu'))
modelCNN.add(Dense(6, activation='softmax'))

modelCNN.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['
    ↪ accuracy'])

early_stopping_monitor = EarlyStopping(patience=2)
modelCNN.fit(CNNdata, DPlabels, validation_split=0.2, epochs=30, callbacks=[
    ↪ early_stopping_monitor])
```

Listing 4.2: Code for the Keras models for ANN and CNN.

Here, in Listing 4.2, Keras uses a similar concept through the Sequential model. This object provides the ability to organize a series of processing layers, in order to create a final custom classifying procedure. It provides Dense, Convolutional, Pooling layers and more, in an easy to comprehend fashion, making the process of creating a custom neural network architecture very easy. The ANN implemented here is made of three hidden layers of 250 nodes, followed by an output layer of 6, in order to present probabilities for each of the 6 labels. The CNN however, is formed by several convolution and pooling layers of varying sizes, finishing off with the FC layer being the previously implemented ANN. This allowed for a better comparison between CNN and ANN, since we can solely examine the performance of the components that define the CNN, such as the convolutional and pooling layers. These parameters and methods can be seen implemented in full in Appendix A at the end of this document. Now that we have discussed how the classifier was implemented, let us review how it performs.

4.2.1 Read Distance performance

As previously mentioned, 10 datasets were created from 0 to 4 meters by calculating the read distance value attached to every RCS value. To further implement a more realistic notion into the data, an SNR of 30 dB was also considered in the generation of the dataset. The results shown were obtained after several rounds of hyperparameter tuning, and as we can see NN based algorithms tend to fare better, in general, when a lack of data is presented. Even so, we can observe that some Supervised Learning algorithms, like KNN, RdF and SVM seem to perform slightly better in the last stages.

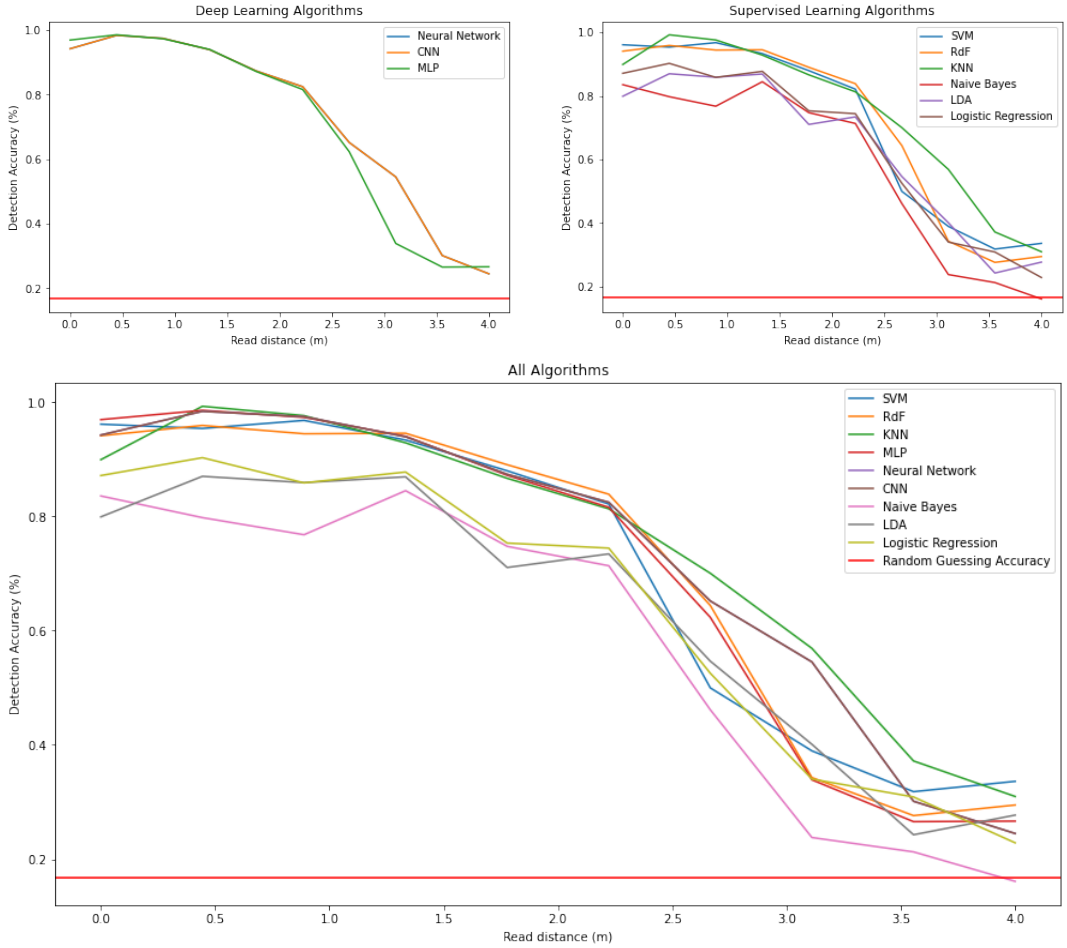
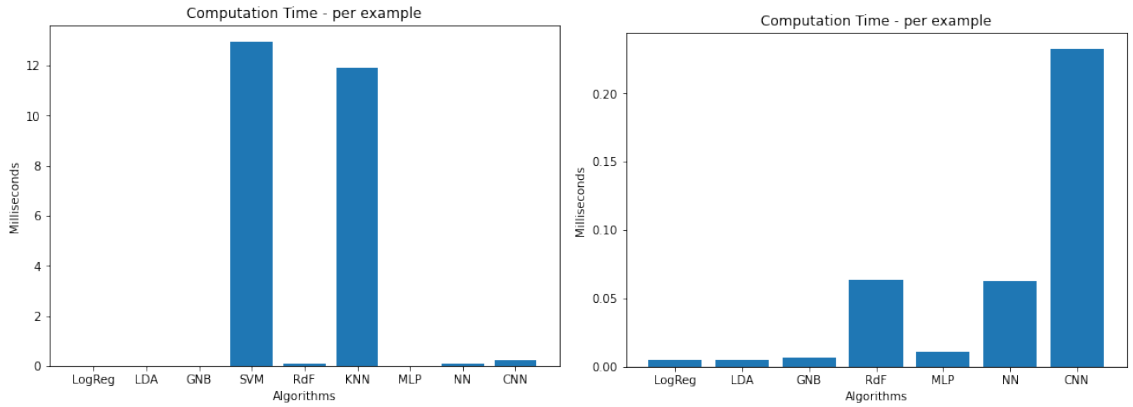


Figure 4.11: Detection Accuracy (%) vs. Read Distance (m).

While it can be tempting to choose KNN as the more effective algorithm, especially when we consider its simplicity regarding implementation, when compared to algorithms such as the Neural Network algorithms, there is another parameter to take into account when comparing these algorithms and that is the computation time (see Figure 4.12). Susan Flake, Motorola's senior manager of retail RFID business development, was recently quoted when asked about RFID Ultra High Frequency (UHF) passive tags reading times in a modern commercial environment, replying "the minimum time needed for a passive

UHF EPC tag is 397.75 microseconds, and the maximum is 45,437.5 microseconds" [67]. This means that our classifier shouldn't come close to the maximum time duration of around 45 milliseconds, since it doesn't even account for the detection and windowing process the reader is responsible for.



(a) All algorithms

(b) Without KNN and SVM

Figure 4.12: Computation time per example - Read Distance.

By plotting the median computation times of each algorithm by example, which can be seen in Figure 4.12, we can immediately observe that SVM and KNN while performing very well in accuracy have massive computation costs in comparison to the rest of the algorithms. This is something that can be justified by the Big O notation concept, KNN and SVM are algorithms that are simple in concept but become ever more resource costly in relation to the dataset presented to them (lack of scalability). By removing KNN and SVM from the plot, we can observe a very low computation time in general, presenting tenths of milliseconds instead of values close to 12 milliseconds. By taking this into consideration, we can infer that the deep neural network (labelled as Neural Network) is the most effective algorithm for read distance variation of the RCS classifier, with a maximum detection accuracy of 99 % and a minimum of 23 %, at 4 meters.

4.2.2 SNR performance

For the SNR performance evaluation of the algorithm, in similar fashion to the read distance dataset creation, 10 datasets were created ranging the SNR from 0 to 30 dB, that is, from a realistically "ideal" signal to a ratio where there is as much noise as there is signal. Here a fixed read distance of 15 cm from the reader was chosen. As seen in Figure 4.13, most of the algorithms remained quite consistent throughout the SNR variations, never going below 80 % with the exception of GNB.

For these results it can also be hard to discern a clear option, however by applying the same logic, mentioned in the previous section, and reviewing the computation times of the algorithms, seen in Figure 4.14, we can, once again, exclude SVM and KNN since

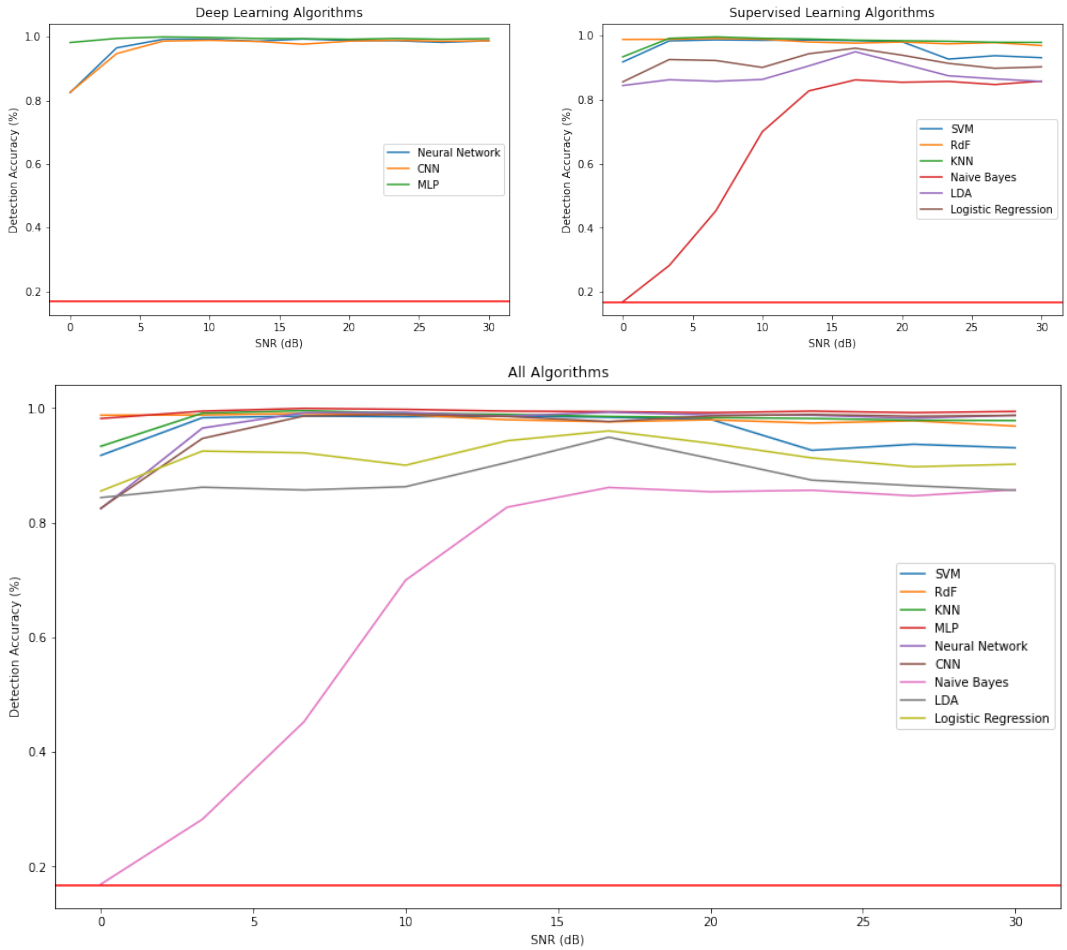
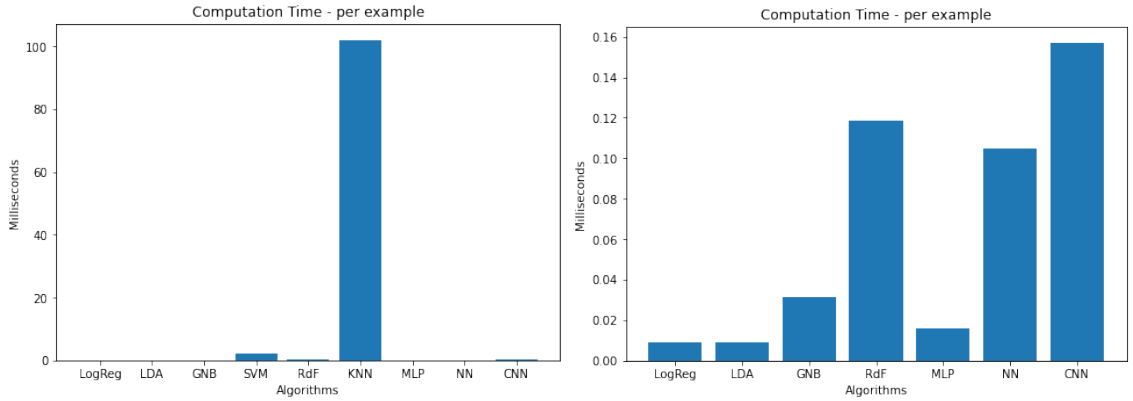


Figure 4.13: Detection Accuracy (%) vs. SNR (dB).

they tend to occupy a large amount of computation time. This leaves us with MLP, CNN, Neural Network and RdF. Here, MLP seems to have the upper hand in accuracy, with a maximum of 99.6 % and minimum of 99.4 %. While a pure RCS classifier will not be used in the final system implementation, since the data will have artefacts from the passing of the RCS signature to the FIR filter and the application of the Fractional Fourier Transform, I found this research not only interesting but vital if the anti-collision itself is implemented differently. Let us now review the anti-collision portion of the system.

4.3 Anti-collision algorithm

The anti-collision algorithm, as formerly mentioned in the previous chapter, was based on the work of Lopes, Varum, and Matos [68], which describes a process that enables the distinction of frequency-coded tags in a fractional domain that is neither time nor frequency. The algorithm takes the spectrogram of the backscattered signal of the tags and applies a Fractional Fourier Transform to it, making the individual signals compact in this domain and easier to window and extract. These windowed signals are then run



(a) All algorithms

(b) Without KNN and SVM

Figure 4.14: Computation time per example - SNR.

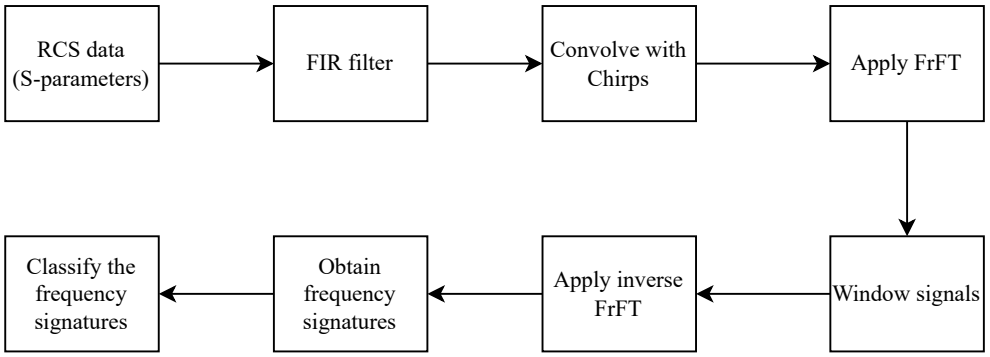


Figure 4.15: Flowchart of the anti collision algorithm process.

through an exact inverse of that Fractional Fourier Transform, recovering the individual backscattered signals, and hence the codes of each tag. Similar to the classifier, this was also implemented in Python, using the SciPy library, a popular signal processing library. The implemented algorithm process starts by simulating the convolution process the tags imprint on the chirp signal when it reaches the tags. While the fingerprint the tags leave on the chirp signal is typically the pure S-parameters, the RCS signature was chosen instead since it presents similar properties and definition [35] [34], with the exception of it typically being more stable. To be able to convolve this signature, we can look to Lopes, Varum, and Matos's work once again and see that it is possible to use a Finite Impulse Response, or FIR, filter, to represent the tag's S parameters or, in this case, the RCS.

To do this, an algorithm was developed that detects the biggest peaks in the RCS signature of the tag and proceeds to match them in intensity to a FIR bandstop filter. This enables us to use our previously obtained RCS dataset to train a classifier able to identify tags in this algorithm further in the system process, as seen in Figure 4.16. The FIR filter is then convolved with the chirp or interrogation signal the tags receive, that is, a frequency sweeping signal over a period of time.

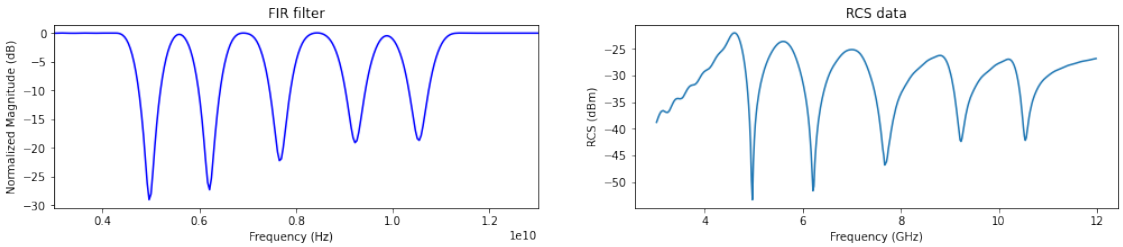
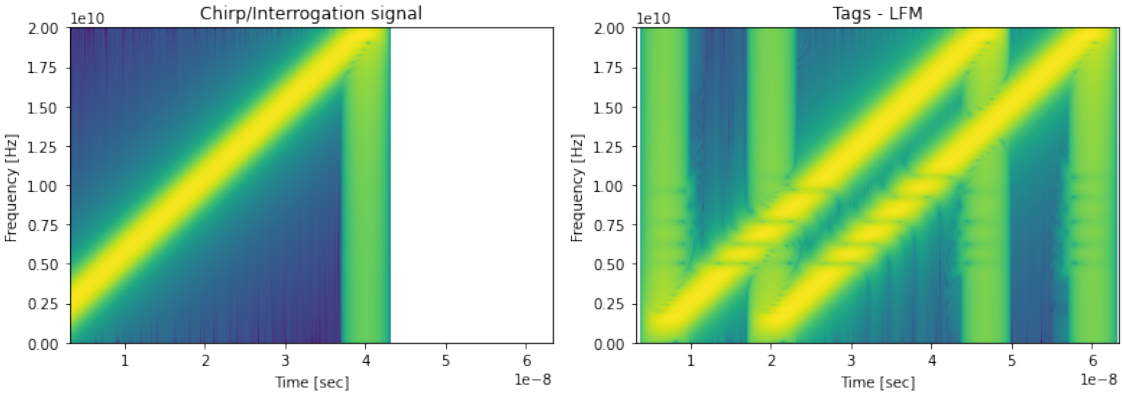


Figure 4.16: FIR filter from RCS signature.



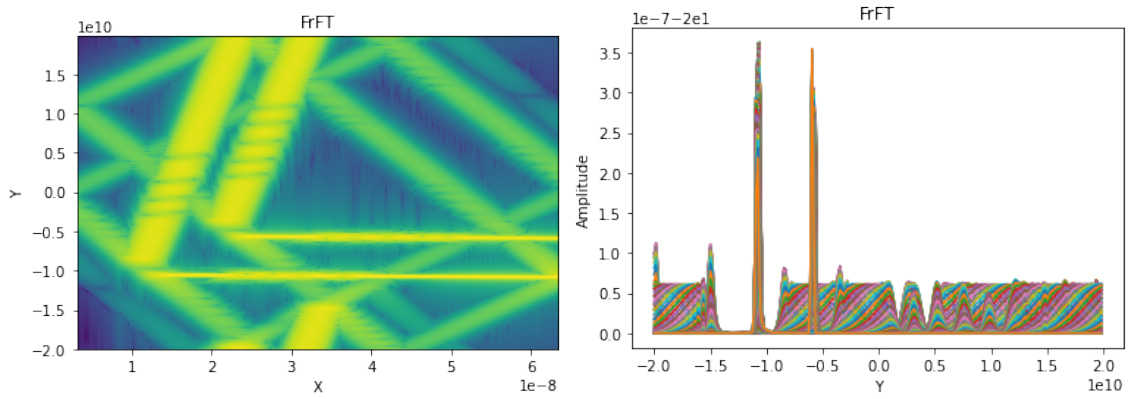
(a) Spectogram of the interrogation signal

(b) Spectogram of the backscattered signal

Figure 4.17: Spectrogram of the sent and received signals.

As can be seen in Figure 4.17, the interrogation signal chosen is defined as taking 40 nanoseconds and sweeping from 1 GHz to 20 GHz. This is then followed by a sum of the two tag backscattered signals phased by a certain amount of time, representative of the physical distance between them, also seen in Figure 4.17. It's important to mention here that the distance tolerated in this simulation to be able to separate the tags in time is rated at about 40 centimetres minimum. This is where the aforementioned Fractional Fourier Transform comes in, allowing a "rotation" of the signal in order to further compact the separated backscattered signals resulting in the spectrogram seen in Figure 4.18. The transform was done with an alpha value of -0.42, this value depends on the slope of the chirp signal, that is, the bandwidth swept divided by the time it takes to finish the sweep, and can be set once during reader calibration since the slope of the signal sent is always the same. If looked at from the Y side after applying this Fractional Fourier Transform, we can distinctly identify two peaks. These peaks represent our tag signatures, standing out from the noise floor created by the transformation. By windowing and isolating these peaks, we can obtain most of the information found in the original backscattered signal for each tag.

The windowing process was done through the SciPy function, "find_peaks", conserving about 30 samples on the sides of each peak, the result of which can be seen in Figure



(a) Spectogram of the FrFT result

(b) Y-sided display of the FrFT result

Figure 4.18: Fractional Fourier Transform result.

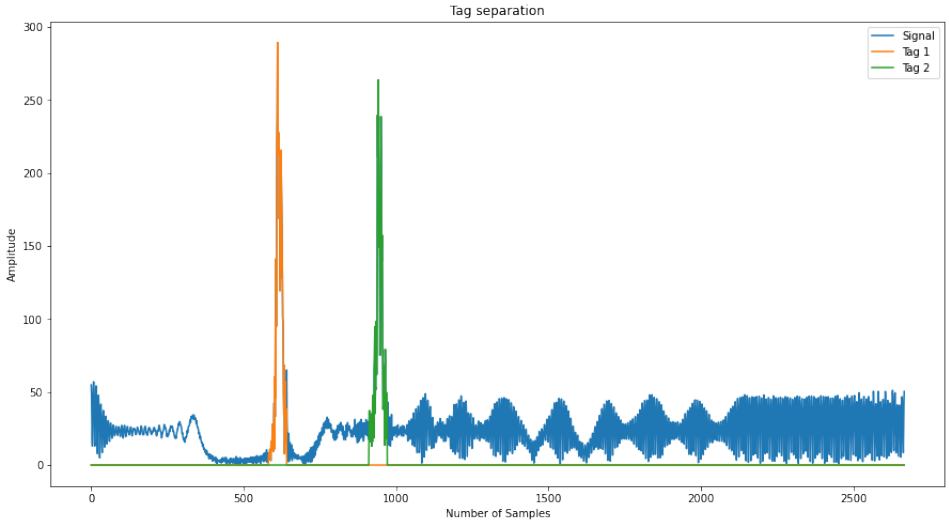


Figure 4.19: Result of the windowing process.

4.19. This leaves us with a significantly clean signal, with some small artefacts introduced by the FrFT, as can be seen in Figure 4.20 and 4.21. In these Figures, we can also see a comparison of the convolved chirp signal with the tag's signature before and after the FrFT and windowing process. Here we can see where the algorithm truly shines, almost completely recovering the initial signal, with a small loss in overall power, maintaining the signature's coding information. We have now implemented a fully functional anti-collision algorithm with a tolerance for distance between tags of about 40 cm, let us now review the overall system simulation and how it performs under SNR variation.

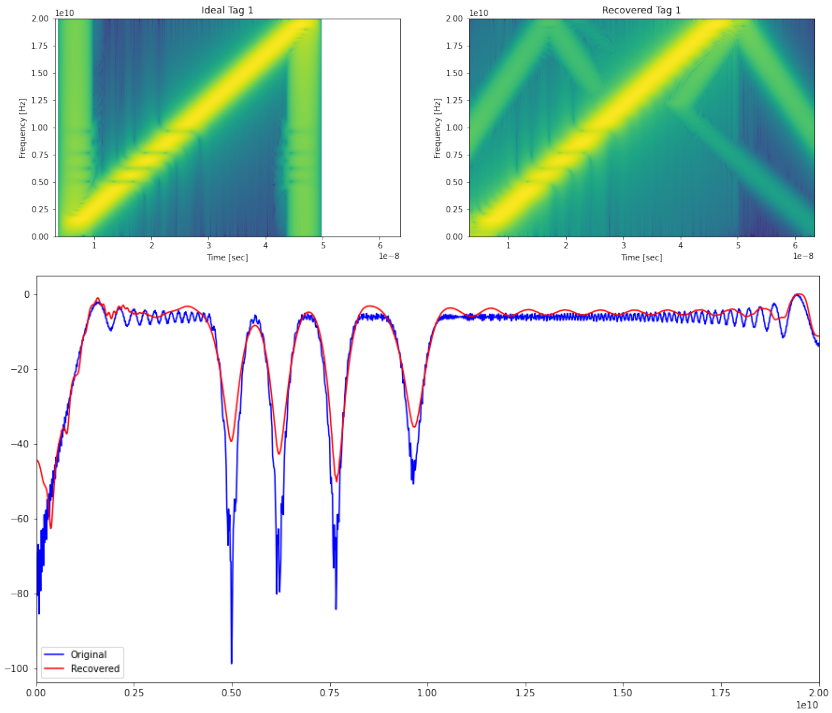


Figure 4.20: Spectrogram and signature comparison of Tag 1 before and after decoding process.

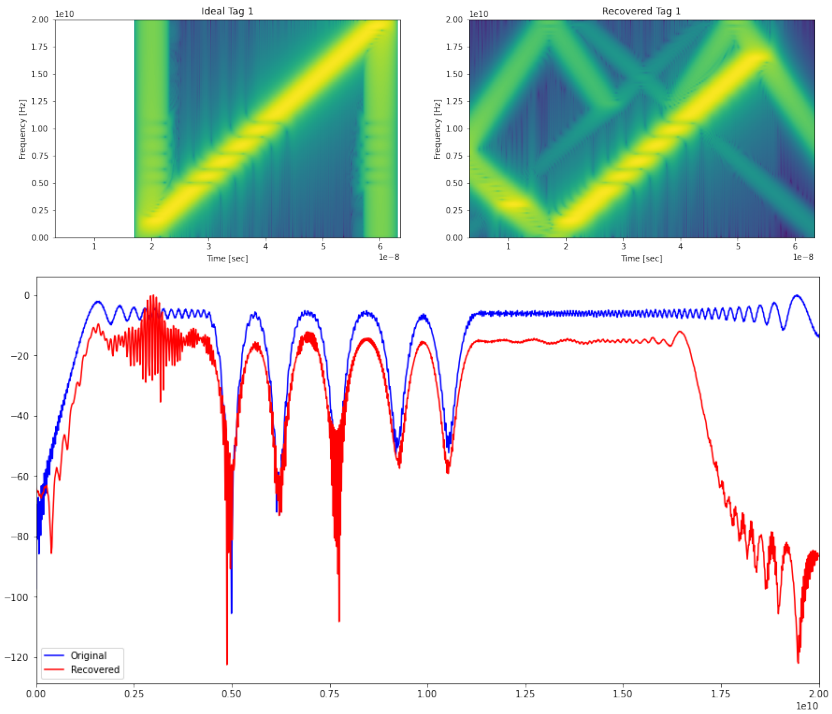


Figure 4.21: Spectrogram and signature comparison of Tag 2 before and after decoding process.

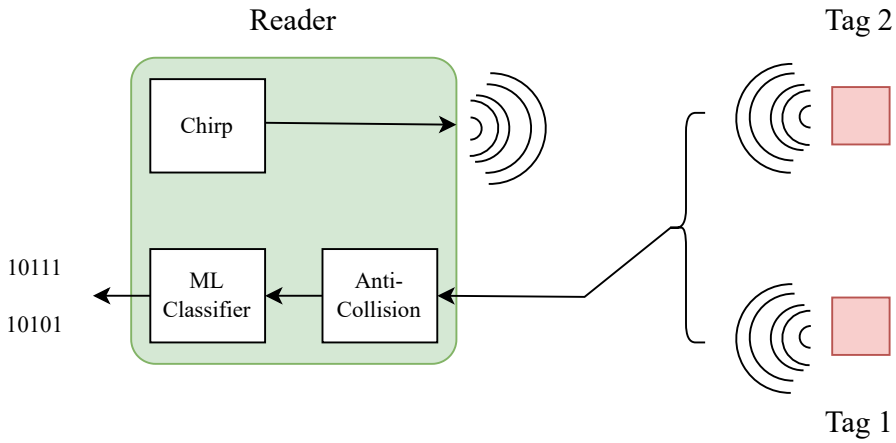


Figure 4.22: Illustration of the full reader process.

4.4 Reader simulation and performance

We have fully implemented all the system components needed to fully simulate and build a functional reader for frequency-domain backscattered chipless RFID tags, we can now proceed to evaluate this system. We will evaluate it on its capability to resist SNR variation. In order to generate the data needed to train the classifier for this evaluation, a data set was created with all the recovery results of the anti-collision algorithm for each RCS signature provided through the process mentioned in Section 4.2, that being, the generation of FIR filters as RCS signatures. This is vital since the anti-collision process introduces even more artefacts to the signatures, as can be seen on Figure 4.20, and the classifier that receives these should be trained accordingly. There is quite a visual difference in the data being fed, but fundamentally, the notches themselves that encrypt the code are still intensely present as seen in Figure 4.23.

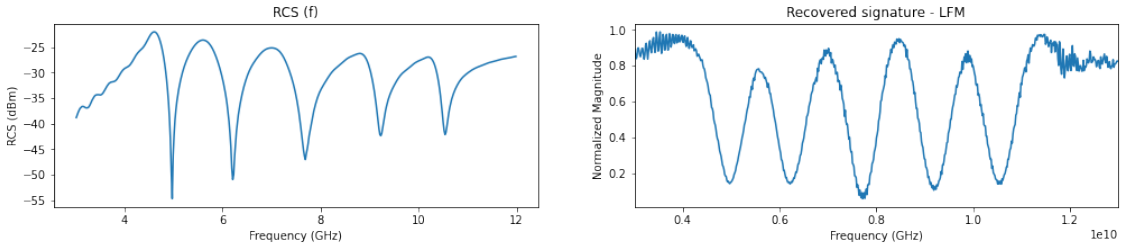
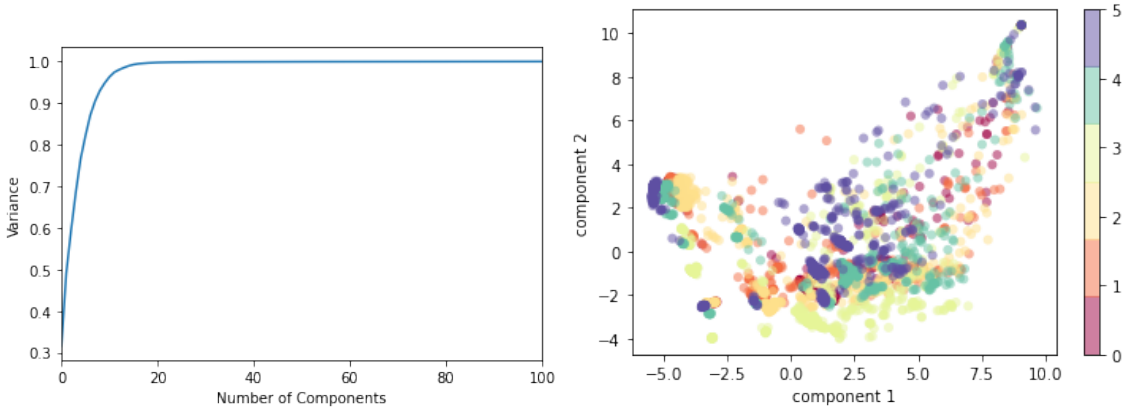


Figure 4.23: Illustration of the dataset generation process.

Here the data has already been limited to the notches themselves found in the ranges of 3 GHz and 12 GHz, as Section 4.2 has proven it to be beneficial for the classifier's performance. The data set presents quite a balanced distribution in the 2 dimensional PCA analysis, providing some small clusters of labels, particularly class 3, 4 and 5. This distribution proves to be quite helpful in the representation of the data, demonstrated by

the 1 dimensional PCA, needing a total of 14 out of 549 components to represent 98 % of the total variance.



(a) Total variance vs. Number of Components (b) 2D PCA

Figure 4.24: PCA analysis of the anti-collision dataset.

With the dataset generated for the training of the classifier, we will now evaluate the system's performance.

4.4.1 SNR performance

With the SNR evaluation, noise was also added in 10 instances, varying from 30 to 0 dB, similar to the RCS classifier process, as shown in Figure 4.25. However, here the noise is added after the tag is "recovered". This generated the PCA analysis as shown

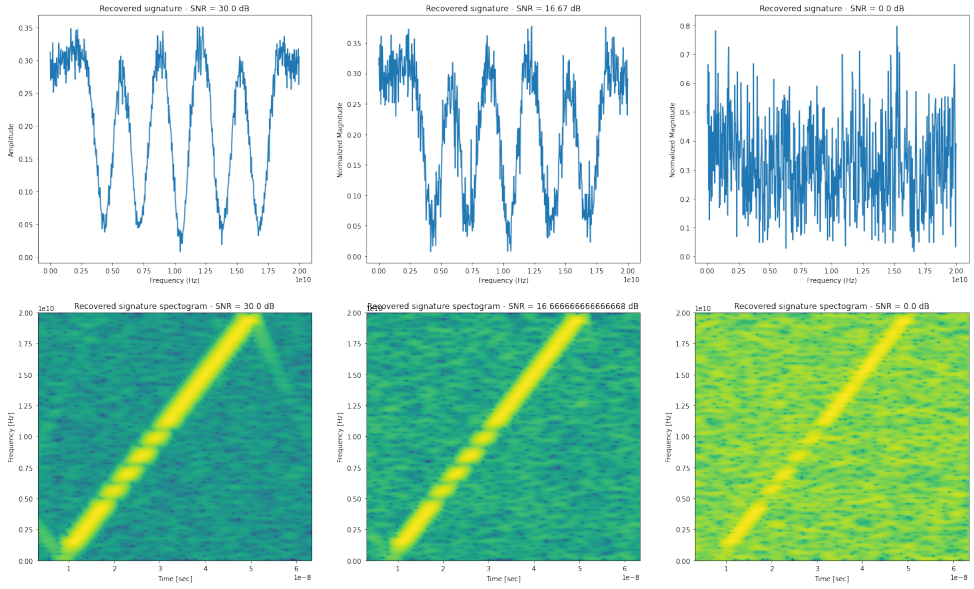
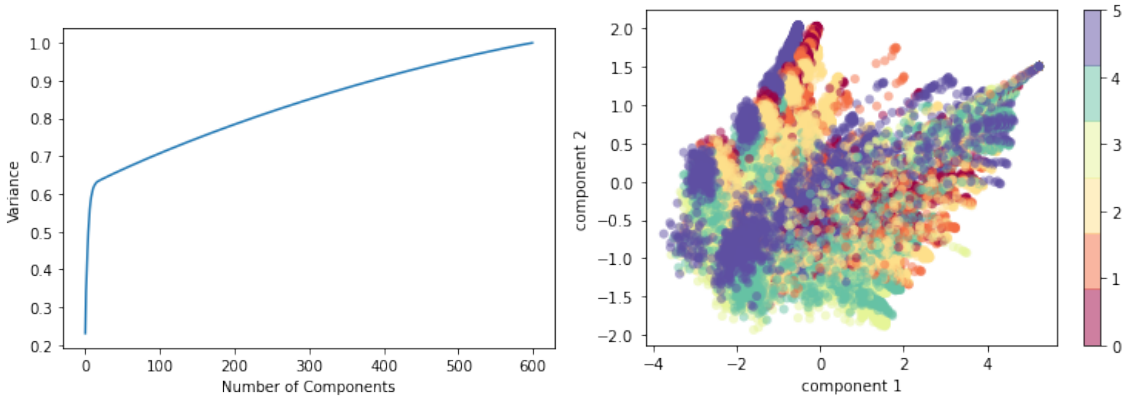


Figure 4.25: SNR generation for the RFID reader system.

on Figure 4.26, in which we can assess some damage to the total variance vs. number of components, increasing the number of components from 14 to 549 components needed to fully represent 98 % variance of the dataset. The dataset presents a similar effect on the 2-dimensional PCA analysis from the noise generation as the aforementioned RCS classifier, showing a "smeared" version of the, previously generated, ideal dataset, thanks to the addition of the Gaussian white noise needed to simulate the SNR.



(a) Total variance vs. Number of Components

(b) 2D PCA

Figure 4.26: PCA analysis of the anti-collision SNR dataset.

For this last evaluation, let us look at the confusion matrix of each algorithm's predictions, seen in Figure 4.27. A confusion matrix is a common tool utilized in ML and AI research to compare an algorithm's prediction performance across multiple labels. This allows us to view what specific classes might be causing confusion in the algorithm's classification process. The table is composed of two axes, one to represent the true label and another for the actual predictions. Each cell on the grid displays the number of occurrences of this phenomenon, that being, for example, the Logistic Regression algorithm guessing correctly 1518 times when the code was '11111' (True Label = '11111', Predicted Label = '11111'), or guessing the code '11110' 341 times when it was supposed to be '11111' (True Label = '11111', Predicted Label = '11110').

Looking at Figure 4.27 we can find that most algorithms struggle with the distinction between labels '11111', '11110' and '11101'. This distinction is complicated for most algorithms since they represent the higher frequency areas of the coding band where the peaks tend to be weaker in depth, thus making it hard for the algorithm to detect either the absence or presence of these later peaks. To compare these algorithms based on the confusion matrix, let us look at the distinction between more naive algorithms like KNN and GNB and stronger dataset-dependent algorithms like CNN and LDA. The naive algorithms cannot seem to grasp the overall difference between labels as well as the more dataset-dependent algorithms, specially with codes like '11011'. This justifies the use of more complex algorithms, like CNN and ANN, rather than the more simple KNN and LDA.

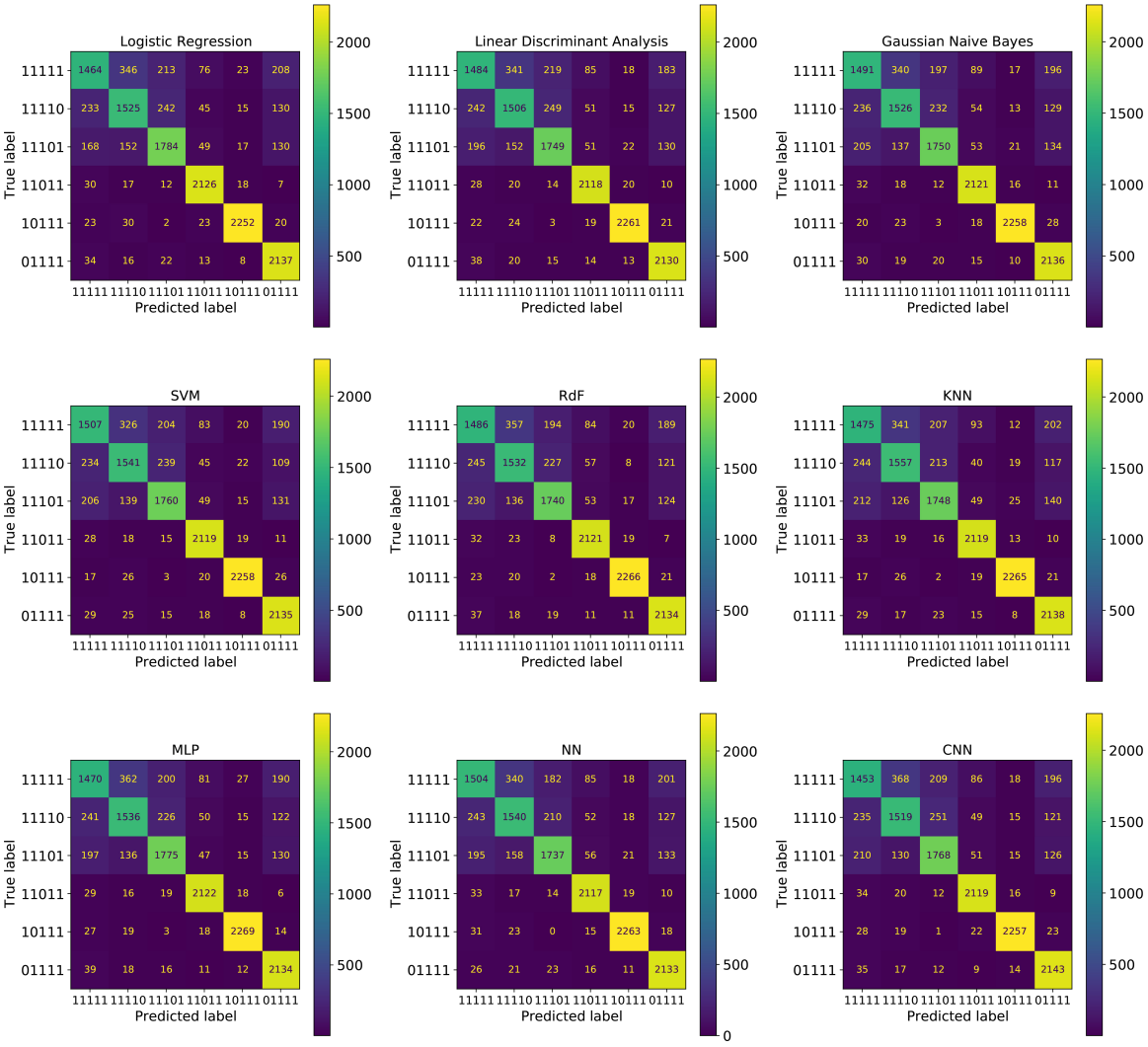


Figure 4.27: Confusion matrixes of the entire SNR variation for the anti-collision dataset.

In terms of the algorithms' performance, seen in Figure 4.28, most of the algorithms behave similarly to the RCS classifier, however, CNN underperforms its previous examples. In this context, it seems that a deep neural network (ANN), that is, in this case, a neural network with three layers consisting of 250 nodes each, outperforms most algorithms, maintaining an accuracy of around 95 % during most of the SNR variation, dipping to 74.5 % when it reaches 0 dB.

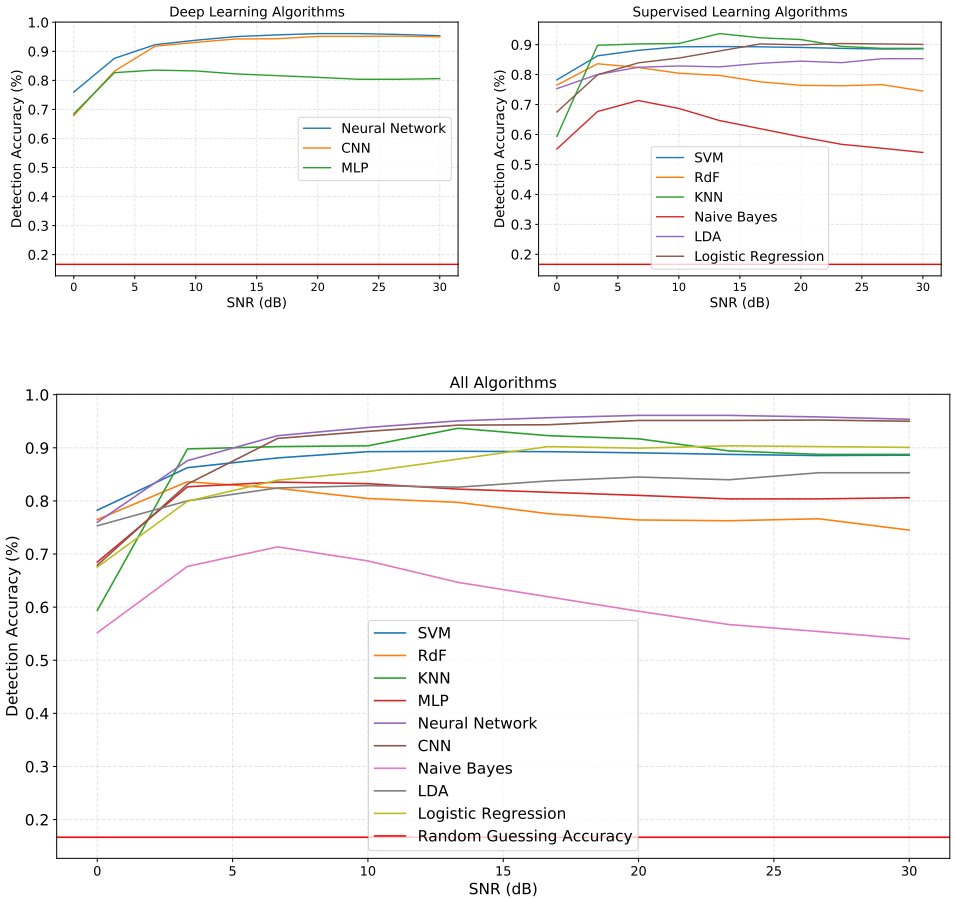


Figure 4.28: Detection Accuracy (%) vs. SNR (dB).

5

Conclusions and Future Work

5.1 Conclusions

This documentation sought out to determine a better solution for the classical problems rooted in modern chipless RFID classification and anti-collision, through the implementation of ML and DL techniques. This has proven itself useful, granting a highly accurate prediction rate in the RCS classifier implementation, where MLP appears to be the most robust, resisting noise variation with a maximum prediction accuracy of 99.6 % (SNR 30 dB) and minimum of 99.4 % (SNR 0 dB), and ANN with distance variation (lack of information) with a maximum of 99 % and a minimum of 23 %, at 4 meters, providing a more robust algorithm than most compared in the fingerprinting examples studied in Chapter 2. In the anti-collision portion, these algorithms were also put to the test by receiving and training on the anti-collision algorithm retrieval data in order to classify the signatures when separated. This brought out MLP as the most stable algorithm when it comes to SNR variation, setting the maximum accuracy at 95 % and the minimum at 74.5 % with an ANN. This can be seen also be seen as a justification and confirmation for the use of more complex algorithms in terms of performance vs. computation costs (see Figure 4.12 and 4.14) and label distinction (see Figure 4.27), in relation to more simple classification algorithms like KNN and SVM, showing us that the evolution of RFID detection lies in the progress of ML and AI algorithms. This also shows us how neural network-based algorithms are key in the process of improving detection accuracy in modern Chipless RFID environments, specifically of the DL type.

5.2 Proposals for Future Work

The future of this technology can be focused in various directions. Nevertheless, the main takeaways from this project should be: the improvement of the distance tolerated by the anti-collision, since this shows to be a maximum of 40 centimetres between tags. However, this can be improved upon by the calibration of the sampling frequency of the signal analyser used for the reading process. Another important point is the overall improvement

that can be done on the algorithms, that is, the expansion of the dataset used in order to obtain better results in the initial stages. This can be done through processes like data-augmentation or the retrieval of more experimental data. Proving that this technology can be done experimentally can also open doors when we consider that this technology can be fully biodegradable and very low-cost. Warehouses can use this simple technology to categorize their items at about 2 meters distance while still being cost-effective and environmentally friendly. Farms and sustainable environments can use this technology to manage live-stock and not worry about harming the environment or creating waste, since these paper-based tags with organic conductive ink can create fully biodegradable tags. Additionally, an IEEE article will be carried out summarizing the contents and results of the simulation contained in this document, which will be later submitted to publication to IEEE International Symposium on Circuits and Systems, ISCAS'23 (to be hosted in Monterey, California, USA, in June 2023), on the 24th of October.

Bibliography

[1] Y. Bendavid, H. Boeck, and R. Philippe. "Redesigning the replenishment process of medical supplies in hospitals with RFID". In: *Business Process Management Journal* (2010) (cit. on pp. xiii, xv).

[2] H. K. Chow et al. "Design of a RFID case-based resource management system for warehouse operations". In: *Expert systems with applications* 30.4 (2006), pp. 561–576 (cit. on pp. xiii, xv).

[3] A. S. Voulodimos et al. "A complete farm management system based on animal identification using RFID technology". In: *Computers and electronics in agriculture* 70.2 (2010), pp. 380–388 (cit. on pp. xiii, xv).

[4] A. Subrahmannian and S. K. Behera. "Chipless RFID: A Unique Technology for Mankind". In: *IEEE Journal of Radio Frequency Identification* 6 (2022), pp. 151–163. DOI: 10.1109/JRFID.2022.3146902 (cit. on pp. xiii, 9, 16, 21, 22).

[5] N. C. Karmaker. "Tag, You're It Radar Cross Section of Chipless RFID Tags". In: *IEEE Microwave Magazine* 17.7 (2016), pp. 64–74. DOI: 10.1109/MMM.2016.2549160 (cit. on pp. xiii, xv, 13–15).

[6] B. Evenor. *Process for radiotelegraphic or radiotelephonic communication* (cit. on p. 1).

[7] S. Preradovic and N. C. Karmakar. "Chipless RFID: Bar Code of the Future". In: *IEEE Microwave Magazine* 11.7 (2010), pp. 87–97. DOI: 10.1109/MMM.2010.938571 (cit. on pp. 2, 10, 12).

[8] N. Soltanieh et al. "A Review of Radio Frequency Fingerprinting Techniques". In: *IEEE Journal of Radio Frequency Identification* 4.3 (2020), pp. 222–233. DOI: 10.1109/JRFID.2020.2968369 (cit. on pp. 2, 17, 18).

[9] R. Rodriguez and Y. Jia. "A wireless inductive-capacitive (LC) sensor for rotating component temperature monitoring". In: *INTERNATIONAL JOURNAL ON SMART SENSING AND INTELLIGENT SYSTEMS VOL* 4 (June 2011). DOI: 10.21307/ijssis-2017-442 (cit. on p. 5).

[10] B. E. Li et al. "Design of Ink-Printed RFID Tags for Electronic Article Surveillance Systems". In: *2017 Mediterranean Microwave Symposium (MMS)*. 2017, pp. 1–4. DOI: 10.1109/MMS.2017.8497077 (cit. on pp. 5, 8).

[11] H. Lehpamer. "RFID Design Principles". In: (Jan. 2008) (cit. on pp. 5, 8).

[12] H. Wheeler. "Simple Inductance Formulas for Radio Coils". In: *Proceedings of the Institute of Radio Engineers* 16.10 (1928), pp. 1398–1400. DOI: 10.1109/JRPROC.1928.221309 (cit. on p. 6).

[13] *How to design a 13.56 mhz customized antenna for ST25 NFC*. Jan. 2008. URL: https://www.st.com/content/ccc/resource/technical/document/application_note/d9/29/ad/cc/04/7c/4c/1e/CD00221490.pdf/files/CD00221490.pdf/jcr:content/translations/en.CD00221490.pdf (cit. on p. 6).

[14] C. Cho et al. "Effect of the substrate, metal-line and surface material on the performance of RFID tag antenna". In: *2007 IEEE Antennas and Propagation Society International Symposium*. 2007, pp. 1761–1764. DOI: 10.1109/APS.2007.4395856 (cit. on pp. 8, 9).

[15] C. Herrojo et al. "Near-Field Chipless-RFID System With Erasable/Programmable 40-bit Tags Inkjet Printed on Paper Substrates". In: *IEEE Microwave and Wireless Components Letters* 28.3 (2018), pp. 272–274. DOI: 10.1109/LMWC.2018.2802718 (cit. on p. 9).

[16] M. P. Jayakrishnan et al. "Electronically Re-Writable Chipless RFID Tag Using Solid State Metal-Insulator-Metal Switches on Paper Substrate". In: *2019 IEEE MTT-S International Microwave Symposium (IMS)*. 2019, pp. 400–403. DOI: 10.1109/MWSYM.2019.8701004 (cit. on p. 9).

[17] A. Iswarya and B. Priyalakshmi. "Design and fabrication of rfid antenna tag using paper substrate for low-cost RFID applications". In: *2016 Online International Conference on Green Engineering and Technologies (IC-GET)*. 2016, pp. 1–6. DOI: 10.1109/GET.2016.7916716 (cit. on p. 9).

[18] X. J. Huang et al. "Design of an UHF RFID Tag Antenna with a Paper Substrate". In: *2018 IEEE International Symposium on Antennas and Propagation USNC/URSI National Radio Science Meeting*. 2018, pp. 1363–1364. DOI: 10.1109/APUSNCURSINRSM.2018.8609257 (cit. on p. 9).

[19] R. Martins et al. "Papertronics: Multigate paper transistor for multifunction applications". In: *Applied Materials Today* 12 (2018), pp. 402–414. ISSN: 2352-9407. DOI: https://doi.org/10.1016/j.apmt.2018.07.002. URL: https://www.sciencedirect.com/science/article/pii/S2352940718302749 (cit. on p. 9).

[20] N. R. Rishani et al. "Comparison of UHF RFID loop matching antennas based on various substrate-metal material combinations". In: *2018 IEEE Middle East and North Africa Communications Conference (MENACOMM)*. 2018, pp. 1–5. DOI: 10.1109/MENACOMM.2018.8371046 (cit. on pp. 9, 10).

[21] P. Nikitin, S. Lam, and K. Rao. "Low cost silver ink RFID tag antennas". In: *2005 IEEE Antennas and Propagation Society International Symposium*. Vol. 2B. 2005, 353–356 vol. 2B. DOI: 10.1109/APS.2005.1552015 (cit. on p. 10).

[22] V. Beedasy and P. J. Smith. "Printed Electronics as Prepared by Inkjet Printing". In: *Materials* 13.3 (2020). ISSN: 1996-1944. DOI: 10.3390/ma13030704. URL: https://www.mdpi.com/1996-1944/13/3/704 (cit. on p. 10).

[23] O. Necibi, S. Beldi, and A. Gharsallah. "Design of a chipless RFID tag using cascaded and parallel spiral resonators at 30 GHz". In: *2015 2nd World Symposium on Web Applications and Networking (WSWAN)*. 2015, pp. 1–5. DOI: 10.1109/WSWAN.2015.7210327 (cit. on pp. 10, 16).

[24] S. Preradovic et al. "A Novel Chipless RFID System Based on Planar Multiresonators for Barcode Replacement". In: *2008 IEEE International Conference on RFID*. 2008, pp. 289–296. DOI: 10.1109/RFID.2008.4519383 (cit. on pp. 10, 12, 15, 21).

[25] V. Sharma and M. Hashmi. "Advances in the Design Techniques and Applications of Chipless RFIDs". In: *IEEE Access* 9 (2021), pp. 79264–79277. DOI: 10.1109/ACCESS.2021.3084056 (cit. on pp. 11, 16, 17).

[26] C. Hartmann, P. Brown, and J. Bellamy. "Design of Global SAW RFID Tag Devices". In: Mar. 2004, pp. 15–20 (cit. on pp. 11, 21).

[27] V. P. Plessky. "Review on SAW RFID tags". In: *2009 IEEE International Frequency Control Symposium Joint with the 22nd European Frequency and Time forum*. 2009, pp. 14–23. DOI: 10.1109/FREQ.2009.5168134 (cit. on p. 11).

[28] H. H. Su et al. "A Chipless RFID Tag Antenna Based on Surface Acoustic Wave". In: *2020 IEEE International Symposium on Antennas and Propagation and North American Radio Science Meeting*. 2020, pp. 1607–1608. DOI: 10.1109/IEEECONF35879.2020.9330186 (cit. on p. 11).

[29] T. N. Tina Ng. "Printed organic TFT sensor tags". In: *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2017, pp. 1–3. DOI: 10.1109/ISCAS.2017.8050680 (cit. on pp. 12, 21).

[30] M. A. Shukoor, S. S. Mukeshbhai, and S. Dey. "12-Bit Multiresonator Based Chipless RFID System for Low-Cost Item Tracking". In: *2021 IEEE International Conference on RFID Technology and Applications (RFID-TA)*. 2021, pp. 136–138. DOI: 10.1109/RFID-TA53372.2021.9617361 (cit. on p. 12).

[31] S. Preradovic et al. "Multiresonator-Based Chipless RFID System for Low-Cost Item Tracking". In: *IEEE Transactions on Microwave Theory and Techniques* 57.5 (2009), pp. 1411–1419. DOI: 10.1109/TMTT.2009.2017323 (cit. on p. 12).

[32] M. A. Shukoor, S. S. Mukeshbhai, and S. Dey. "12-Bit Multiresonator Based Chipless RFID System for Low-Cost Item Tracking". In: *2021 IEEE International Conference on RFID Technology and Applications (RFID-TA)*. 2021, pp. 136–138. DOI: 10.1109/RFID-TA53372.2021.9617361 (cit. on pp. 12, 16, 21).

[33] B. A. Munk. "Frequency Selective Surfaces: Theory and Design". In: 2000 (cit. on pp. 13, 24).

[34] A. Vena, E. Perret, and S. Tedjini. "High-Capacity Chipless RFID Tag Insensitive to the Polarization". In: *IEEE Transactions on Antennas and Propagation* 60.10 (2012), pp. 4509–4515. DOI: 10.1109/TAP.2012.2207347 (cit. on pp. 13, 48).

[35] D. Betancourt et al. "Bending and Folding Effect Study of Flexible Fully Printed and Late-Stage Codified Octagonal Chipless RFID Tags". In: *IEEE Transactions on Antennas and Propagation* 64.7 (2016), pp. 2815–2823. DOI: 10.1109/TAP.2016.2559522 (cit. on pp. 14, 15, 21, 23, 24, 32, 35, 48).

[36] M. Borgese et al. "Radar Cross Section of Chipless RFID Tags and BER Performance". In: *IEEE Transactions on Antennas and Propagation* 69.5 (2021), pp. 2877–2886. DOI: 10.1109/TAP.2020.3037800 (cit. on p. 15).

[37] J. Havlicek et al. "Chipless RFID Tag Based on Electrically Small Spiral Capacitively Loaded Dipole". In: *IEEE Antennas and Wireless Propagation Letters* 16 (2017), pp. 3051–3054. DOI: 10.1109/LAWP.2017.2760059 (cit. on pp. 15, 21).

[38] O. Necibi, S. Beldi, and A. Gharsallah. "Design of a chipless RFID tag using cascaded and parallel spiral resonators at 30 GHz". In: *2015 2nd World Symposium on Web Applications and Networking (WSWAN)*. 2015, pp. 1–5. DOI: 10.1109/WSWAN.2015.7210327 (cit. on p. 16).

[39] C. Su et al. "A MIMO Radar Signal Processing Algorithm for Identifying Chipless RFID Tags". In: *Sensors* 21.24 (2021). ISSN: 1424-8220. DOI: 10.3390/s21248314. URL: https://www.mdpi.com/1424-8220/21/24/8314 (cit. on pp. 16, 22).

[40] B. Lopes and J. N. Matos. "Simulation of a Chipless RFID System using discreet FrFT to recover individual tags IDs". In: *2019 SBMO/IEEE MTT-S International Microwave and Optoelectronics Conference (IMOC)*. 2019, pp. 1–3. DOI: 10.1109/IMOC43827.2019.9317688 (cit. on pp. 16, 22, 23, 32).

[41] F. Paredes et al. "High Data Density Near-Field Chipless-RFID Tags With Synchronous Reading". In: *IEEE Journal of Radio Frequency Identification* 4.4 (2020), pp. 517–524. DOI: 10.1109/JRFID.2020.2996586 (cit. on p. 16).

[42] M. Zomorrodi, N. C. Karmakar, and S. G. Bansal. "Introduction of electromagnetic image-based chipless RFID system". In: *2013 IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*. 2013, pp. 443–448. DOI: 10.1109/ISSNIP.2013.6529831 (cit. on pp. 16, 21).

[43] L. M. Arjomandi, G. Khadka, and N. C. Karmakar. "mm-Wave Chipless RFID Decoding: Introducing Image-based Deep Learning Techniques". In: *IEEE Transactions on Antennas and Propagation* (2021), pp. 1–1. DOI: 10.1109/TAP.2021.3137197 (cit. on p. 17).

[44] S. Riyaz et al. "Deep Learning Convolutional Neural Networks for Radio Identification". In: *IEEE Communications Magazine* 56.9 (2018), pp. 146–152. DOI: 10.1109/MCOM.2018.1800153 (cit. on p. 17).

[45] S. U. Rehman et al. "Radio frequency fingerprinting and its challenges". In: *2014 IEEE Conference on Communications and Network Security*. 2014, pp. 496–497. DOI: 10.1109/CNS.2014.6997522 (cit. on p. 17).

[46] O. Ureten and N. Serinken. "Wireless security through RF fingerprinting". In: *Canadian Journal of Electrical and Computer Engineering* 32.1 (2007), pp. 27–33. DOI: 10.1109/CJECE.2007.364330 (cit. on p. 17).

[47] B. Danev, T. S. Heydt-Benjamin, and S. Capkun. "Physical-layer Identification of RFID Devices." In: *USENIX security symposium*. 2009, pp. 199–214 (cit. on p. 17).

[48] S. Riyaz et al. "Deep Learning Convolutional Neural Networks for Radio Identification". In: *IEEE Communications Magazine* 56.9 (2018), pp. 146–152. DOI: 10.1109/MCOM.2018.1800153 (cit. on pp. 19, 21, 22, 32).

[49] C. Bertoncini et al. "Wavelet Fingerprinting of Radio-Frequency Identification (RFID) Tags". In: *IEEE Transactions on Industrial Electronics* 59.12 (2012), pp. 4843–4850. DOI: 10.1109/TIE.2011.2179276 (cit. on pp. 19, 20).

[50] S. Wang et al. "A Convolutional Neural Network-Based RF Fingerprinting Identification Scheme for Mobile Phones". In: *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 2020, pp. 115–120. DOI: 10.1109/INFOCOMWKSHPS50562.2020.9163058 (cit. on p. 19).

[51] L. Peng et al. "Deep Learning Based RF Fingerprint Identification Using Differential Constellation Trace Figure". In: *IEEE Transactions on Vehicular Technology* 69.1 (2020), pp. 1091–1095. DOI: 10.1109/TVT.2019.2950670 (cit. on pp. 19, 21, 22).

[52] G. Shen et al. *Radio Frequency Fingerprint Identification for LoRa Using Spectrogram and CNN*. 2020. arXiv: 2101.01668 [eess.SP] (cit. on p. 19).

[53] H. J. Patel, M. A. Temple, and R. O. Baldwin. "Improving ZigBee Device Network Authentication Using Ensemble Decision Tree Classifiers With Radio Frequency Distinct Native Attribute Fingerprinting". In: *IEEE Transactions on Reliability* 64.1 (2015), pp. 221–233. DOI: 10.1109/TR.2014.2372432 (cit. on pp. 19, 21, 22).

[54] P. Robyns et al. "Physical-layer fingerprinting of LoRa devices using supervised and zero-shot learning". In: July 2017, pp. 58–63. DOI: 10.1145/3098243.3098267 (cit. on pp. 19, 22).

[55] L. Ding et al. "Specific Emitter Identification via Convolutional Neural Networks". In: *IEEE Communications Letters* 22.12 (2018), pp. 2591–2594. DOI: 10.1109/LCOMM.2018.2871465 (cit. on pp. 19, 21, 22).

[56] M. Ester et al. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231 (cit. on p. 20).

[57] S. Jeong et al. "A Machine Learning Approach-based Chipless RFID System for Robust Detection in Real-world Implementations". In: *2021 IEEE MTT-S International Microwave Symposium (IMS)*. 2021, pp. 661–664. DOI: 10.1109/IMS19712.2021.9574849 (cit. on pp. 20, 22).

[58] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, Jan. 2006. URL: https://www.microsoft.com/en-us/research/publication/pattern-recognition-machine-learning/ (cit. on p. 26).

[59] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791 (cit. on p. 30).

[60] A. Nordrum, K. Clark, and I. Spectrum. *5G bytes: Beamforming explained*. Sept. 2021. URL: https://spectrum.ieee.org/5g-bytes-beamforming-explained (cit. on p. 32).

[61] J. Brownlee. *Impact of dataset size on Deep Learning Model Skill and performance estimates*. Aug. 2020. URL: https://machinelearningmastery.com/impact-of-dataset-size-on-deep-learning-model-skill-and-performance-estimates/ (cit. on pp. 35, 37).

[62] N. Py. 2022. URL: https://numpy.org/ (cit. on p. 37).

[63] M. lib. 2022. URL: https://matplotlib.org/ (cit. on p. 37).

[64] S. Raschka, J. Patterson, and C. Nolet. "Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence". In: *Information* 11.4 (2020). ISSN: 2078-2489. DOI: 10.3390/info11040193. URL: https://www.mdpi.com/2078-2489/11/4/193 (cit. on p. 37).

[65] S. Kit. *SciKit*. 2022. URL: https://scikit-learn.org/stable/ (cit. on p. 37).

[66] K. Team. *Simple. flexible. powerful.* 2022. URL: https://keras.io/ (cit. on p. 37).

[67] R. Journal. *How long does it take to read passive UHF tags?* 2009. URL: https://www.rfidjournal.com/question/how-long-does-it-take-to-read-passive-uhf-tags (cit. on p. 46).

[68] B. Lopes, T. Varum, and J. N. Matos. "Use of FrFT in an indoor scenario for chipless RFID tags ID recovery". In: *IET Microwaves, Antennas & Propagation* 14.12 (2020), pp. 1316–1322. DOI: https://doi.org/10.1049/iet-map.2020.0212. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/iet-map.2020.0212. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-map.2020.0212 (cit. on pp. 47, 48).

A

Python Project

The following section presents a clean and summarized version of the project code used to implement the dataset preparation, algorithm generation and anti-collision simulation.

```
#!/usr/bin/env python
# coding: utf−8

# # Algorithm comparison and classification − RFID
#
# To recover the last session's data, use 'dill.load_session('ThesisFullData.db')'
# Created and implemented by Marcelo ColaÃğo

# In[1]:


get_ipython().run_line_magic('config', 'IPCompleter.greedy=True')
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
import pandas as pd
import os
import io
import cv2
import math
import time as tt
import dill
import keras
import frft
import frft2
import frft3

from matplotlib import cm
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from scipy import signal, constants
```

```
from scipy.fft import fftshift
from tensorflow.keras.utils import to_categorical
from keras.callbacks import EarlyStopping
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Flatten
from keras.layers.convolutional import Conv1D, MaxPooling1D

from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn import preprocessing
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn import preprocessing
from sklearn.decomposition import PCA
from sklearn.neighbors import KNeighborsClassifier
from sklearn import datasets, linear_model
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.decomposition import PCA as RandomizedPCA
from sklearn.pipeline import make_pipeline
from sklearn.ensemble import RandomForestClassifier


# ### 1. Data Cleanup and Processing
#
# This stage will be used to pre-process the data for plotting and eventually training from the
    ↪ text files generated in CST Studio Suite. Additionally, it will be used to compute the
    ↪ read distance from the frequency and RCS data obtained, using the following
    ↪ formula:
#
#
# <center>$R_{range}= \sqrt[4]{\frac{\lambda^2P_{t}G_{Tx}G_{Rx}\sigma}{(4\pi)^3P_{
    ↪ min}}}$</center>
#
# Where:
# <ul>
# <li> $G_{Tx}$ = gain of the reader antenna </li>
# <li> $G_{Rx}$ = gain of the tag antenna </li>
# <li> $P_{min}$ = sensitivity of the reader </li>
# <li> $P_{t}$ = transmitted power </li>
# </ul>
#
# ### Constants:
```

```
# In[4]:


def convert_dbm_to_W(value):
    dbm = value
    w = (1*10**(dbm/10))/(10**3)
    return w


# In[5]:


def convert_W_to_dbm(value):
    W = value
    dbm = 10*np.log10(1000*abs(W))
    return dbm


# In[6]:


P_t = convert_dbm_to_W(3) ##Trasmitted Power in dBm
P_min = -70 ##Power Sensitivity in dBm
P_minW = convert_dbm_to_W(P_min) ##Power Sensitivity in W
G_tx = 1 ##Gain of the tag antenna in dB
G_rx = 1 ##Gain of the reader antenna in dB

low_freq = 3 ##GHz
high_freq = 12 ##GHz
read_distance_set = 2 ##in meters

read_distances = np.linspace(0,4.0,10)
files_to_remove = [[1238, 1239, 1240, 1241, 1242, 1243, 1244, 1245, 1246, 1247, 1248,
    ↪ 1249, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, 1258, 1259, 1260, 1261,
    ↪ 1262]]
SNR_values = np.flip(np.linspace(0,30,10))


# In[7]:


def RDfromRCS(frequency, RCS):
    RCS = convert_dbm_to_W(RCS)
    return ((((299792458/(frequency*10**6))**2)*P_t*G_tx*G_rx*RCS)/((4*math.pi)
        ↪ **3*P_minW))**(1/4)
```

```
# In[8]:


def readFiles(read_distance, SNR):
    files=['\\Paper\\RCS_SilverTh_0.05_0.15_All.txt','\\Paper\\RCS_Theta_0_180_All
        ↪ .txt','\\Paper\\RCS_SubTh_0.05_0.15_All.txt',
         '\\RCS_SilverTh_0.05_0.15_All.txt','\\RCS_Theta_0_180_All.txt','\\
            ↪ RCS_SubTh_0.05_0.15_All.txt']

    codes=['11111','11110','11101','11011','10111','01111']
    data=[]
    imgdata=[]

    labels=[]

    for index, code in enumerate(codes):
        print(str(index) +'␣−−>␣'+ code)
        nrImages = 0
        for file in files:
            f = open('C:\\Users\\marce\\Documents\\CST_Data\\ChiplessOctagon\\'
                ↪ +code+file, 'r')
            label=[]
            frequency=[]
            rcs=[]
            RD=[]
            a = 0
            for line in f:
                line = line.strip()
                if line[0].isnumeric() and low_freq <= float(line.split()[0]) <=
                    ↪ high_freq:

                    freq = float(line.split()[0])
                    rcs_dB = float(line.split()[1])
                    read_d = RDfromRCS(freq,rcs_dB)

                    frequency.append(freq)
                    RD.append(read_d)
                    label.append(index)
                    if(read_d>=read_distance):
                        rcs.append(rcs_dB)
                    else:
                        rcs.append(P_min)

                    if(freq == 11.982):
```

```
                        data.append(np.column_stack((frequency,rcs,label,RD)))
                        labels.append(index)
                        frequency=[]
                        rcs=[]
                        label=[]
                        RD=[]
                        a+=1
                        nrImages+=1
    data = np.array(data)
    data = np.delete(data,files_to_remove,axis=0)
    labels = np.array(labels)
    labels = np.delete(labels,files_to_remove,axis=0)
    return data, labels


# In[11]:


def addNoisebySNR(time_sig,SNR):
    # Set a target SNR
    target_snr_db = SNR
    # Calculate signal power and convert to dB
    sig_avg_watts = np.mean(time_sig**2)
    sig_avg_db = 10 * np.log10(sig_avg_watts)
    # Calculate noise according to [2] then convert to watts
    noise_avg_db = sig_avg_db - target_snr_db
    noise_avg_watts = 10 ** (noise_avg_db / 10)
    # Generate an sample of white noise
    mean_noise = 0
    noise_volts = np.random.normal(mean_noise, np.sqrt(noise_avg_watts), len(
        ↪ time_sig))
    # Noise up the original signal
    Rtag1 = time_sig + noise_volts
    return Rtag1


# In[12]:


fs = 40e9
def obtain_chirp_signature(filter_RCS, SNR):
    fs = 40e9

    c= constants.c
    range_max = 4
    Tchirp = 40e-9
```

```
distance_tag1 = 0
distance_tag2 = 0

tchirp = np.arange(0,Tchirp,1/fs)
Nchirp=len(tchirp)


B=10e9

T= (2*range_max/c) + Tchirp

t = np.arange(0,T,1/fs)
N = len(t)
f = np.linspace(0,fs,N)
a = B/Tchirp

## CHIRP GENERATION

chirp = signal.chirp(tchirp,1e9,Tchirp,20e9);


fill = np.arange(Tchirp,T,1/fs)*0
chirp = np.concatenate([chirp,fill[1:]])

tag1_s=chirp
tag1_s = signal.convolve(chirp,filter_RCS);
tag1_s = tag1_s[0:N-1];

tmp = np.where(tag1_s > 0.2);
if not tmp:
    idx = tmp[0][0];
    tag1_s = tag1_s[idx:idx+Nchirp];
    tmp1 = len(tag1_s[Nchirp:]);
    tag1_s[Nchirp:] = np.zeros(tmp1);

delay_tag1 = round(2* distance_tag1/c * fs) ;
delay_tag2 = round(2* distance_tag2/c * fs) ;


r_signal = np.zeros(len(chirp));
r_signal[delay_tag1:delay_tag1+len(tag1_s)] += tag1_s


alpha_opt = -0.42
```

```
    chirps = frft2.frft(r_signal,alpha_opt)
    X1 = np.fft.fftshift(np.fft.fft(chirps))

    peaks = signal.find_peaks(20*np.log(abs(X1)/max(abs(X1))),height = -10, distance
        ↪ =50)
    n_samples = 60

    tag1_z = np.zeros(len(X1))
    l = int(peaks[0][0]-n_samples/2)
    r = int(peaks[0][0]+n_samples/2)
    tag1_z[l:r] = 1
    tag1=X1*tag1_z

    Rtag1 = np.real(frft2.frft(np.fft.ifft(np.fft.ifftshift(tag1)),-alpha_opt))
    Rtag1 = addNoisebySNR(Rtag1,SNR)

    return Rtag1


# In[13]:


def RCS_to_RCSChirp(data, index, SNR):
    fill_index = np.linspace(1e9,20e9,1000)
    lower_lim = 50
    upper_lim = 500
    bw = 0.6
    fs = 40e9
    frequency = data[index,lower_lim:upper_lim,0]
    RCS = data[index,lower_lim:upper_lim,1]

    pks,mag = signal.find_peaks(RCS*-1, height = 30)
    filters=[]
    for bit in pks:
        numtaps = int(-3.5*RCS[bit])
        if (numtaps % 2) == 0:
                numtaps += 1
        filters.append(signal.firwin(numtaps,cutoff = (np.array([frequency[bit]-bw/2,
            ↪ frequency[bit]+bw/2])*1e9)/(fs/2),pass_zero='bandstop'))

    for f in range(len(filters)-1):
        if(f>0):
            filter_signature = signal.convolve(filter_signature,filters[f+1])
        else:
            filter_signature = signal.convolve(filters[f],filters[f+1])
```

```
    w, h = signal.freqz(filter_signature);
    r_tag = obtain_chirp_signature(filter_signature, SNR)
    rf_tag = np.array(abs(np.fft.fft(r_tag)))
    rf_tag = rf_tag[0:int(rf_tag.shape[0]/2)]
    return rf_tag, r_tag


# In[14]:


fullRD_Data=[]
fullRD_Labels=[]
chirp_RCS=[]
full_chirp_RCS=[]
R_tag=[]
fullR_tag=[]

for i, read_distance in enumerate(SNR_values):
    chirp_RCS = []
    print('Run '+str(i+1)+'/'+str(len(read_distances)))
    data,labels = readFiles(0.15,30)
    print(read_distance)
    for i in range(data.shape[0]):
        if(i % 1000 == 0):
            print('Simulating recovery: '+str(i)+'/'+str(data.shape[0]))
        (rf_tag, r_tag) = RCS_to_RCSChirp(data, i, read_distance)
        chirp_RCS.append(rf_tag[200:800])
        R_tag.append(r_tag)
    full_chirp_RCS.append(chirp_RCS)
    fullR_tag.append(R_tag)
    fullRD_Data.append(data)
    fullRD_Labels.append(labels)


# In[15]:


fullRD_Data = np.array(fullRD_Data)
fullRD_Labels = np.array(fullRD_Labels)
full_chirp_RCS = np.array(full_chirp_RCS)
fullR_tag = np.array(fullR_tag)


# In[16]:
```

```
V_fullRD_Data = fullRD_Data.reshape(-1, *fullRD_Data.shape[-2:])
V_fullRD_Labels = fullRD_Labels.flatten()
V_full_Chirp = full_chirp_RCS.reshape(-1, *full_chirp_RCS.shape[-1:])
V_full_Rtag = fullR_tag.reshape(-1, *fullR_tag.shape[-1:])

scaler = preprocessing.MinMaxScaler()
data_norm = scaler.fit_transform(V_full_Chirp)
labels = V_fullRD_Labels


# ## Normal Neural Network

# In[17]:


modelNN = Sequential()

n_cols = data_norm.shape[1]

DPlabels = to_categorical(labels)

modelNN.add(Dense(250, activation='relu', input_shape=(n_cols,)))
modelNN.add(Dense(250, activation='relu'))
modelNN.add(Dense(250, activation='relu'))
modelNN.add(Dense(6, activation='softmax'))
modelNN.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['
    ↪ accuracy'])

early_stopping_monitor = EarlyStopping(patience=2)
modelNN.fit(data_norm, DPlabels, validation_split=0.2, epochs=30, callbacks=[
    ↪ early_stopping_monitor])


# ## CNN

# In[18]:


modelCNN = Sequential()

CNNdata = data_norm.reshape(-1, data_norm.shape[1], 1)

modelCNN.add(Conv1D(2, kernel_size=11, activation='relu', input_shape=(n_cols,1)))
modelCNN.add(MaxPooling1D(pool_size=2))
```

```
modelCNN.add(Conv1D(4, kernel_size=7, activation='relu'))
modelCNN.add(MaxPooling1D(pool_size=2,padding="same"))

modelCNN.add(Conv1D(8, kernel_size=11, activation='relu'))

modelCNN.add(Flatten())

modelCNN.add(Dense(250, activation='relu'))
modelCNN.add(Dense(250, activation='relu'))
modelCNN.add(Dense(250, activation='relu'))
modelCNN.add(Dense(6, activation='softmax'))

modelCNN.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['
    ↪ accuracy'])

early_stopping_monitor = EarlyStopping(patience=5)
modelCNN.fit(CNNdata, DPlabels, validation_split=0.2, epochs=30, callbacks=[
    ↪ early_stopping_monitor])


# ### 2. Cross Validation — Multiple Models
#
# Various models will be cross validated (5–fold cross validation) across the entire dataset:
# <ul>
# <li>Support Vector Machine</li>
# <li>Random Forest Classifier</li>
# <li>K Neighbors Classifier</li>
# <li>Gaussian Naive Bayes Classifier</li>
# <li>Linear Discriminant Analysis Classifier</li>
# <li>Logistic Regression Classifier</li>
#
# </ul>

# In[19]:


pipe_svc = Pipeline([('pca', PCA(0.98)),
                     ('svc', SVC(kernel='rbf', random_state=42, C=50, gamma=0.005))])


pipe_MLP = Pipeline([('pca', PCA(0.98)),
                     ('MLP', MLPClassifier(random_state=42, activation='tanh',
                         ↪ max_iter=300))])

pipe_rf = Pipeline([('pca', PCA(0.98)),
```

```
                    ('rf', RandomForestClassifier(random_state=42,n_estimators=250,
                        ↪ max_depth=20))])

pipe_knn = Pipeline([('pca', PCA(0.98)),
                    ('knn', KNeighborsClassifier(n_neighbors=25, leaf_size=1, p=1))])

pipe_gnb = Pipeline([#('pca', PCA(0.98)),
                    ('GNB', GaussianNB())])

pipe_LDA = Pipeline([('pca', PCA(0.98)),
                    ('LDA', LinearDiscriminantAnalysis())])

pipe_LogReg = Pipeline([('pca', PCA(0.98)),
                    ('LogReg', LogisticRegression(random_state=42, max_iter=2000, C
                        ↪ =10.0))])


# In[36]:


param_svc = [{'svc__C': [0.01,0.1,1, 5, 10, 50],
      'svc__gamma': [0.0001, 0.0005, 0.001, 0.005]}]

param_rf = [{'rf__n_estimators': [6, 10, 50, 100, 250],
      'rf__max_depth': [5, 10, 20]}]

param_knn = [{'knn__p':[1,2],
            'knn__leaf_size':[1,2,5,10,20,30,40,50]}]

param_gnb={}

param_LDA={}

param_MLP = [{'MLP__hidden_layer_sizes': [(50,50,50), (50,100,50), (100,)],
    'MLP__activation': ['tanh', 'relu'],
    'MLP__solver': ['sgd', 'adam'],
    'MLP__alpha': [0.0001, 0.05],
    'MLP__learning_rate': ['constant','adaptive']}]

param_LogReg= [{"LogReg__C":np.logspace(−3,3,7), "LogReg__penalty":["l1","l2"]}]


# In[25]:


# Train the grid search model
```

```
logreg_cv = GridSearchCV(pipe_LogReg, {}, scoring='accuracy', cv=5, n_jobs=4,verbose
    ↪ =5).fit(data_norm.astype(np.float64), labels)


# In[ ]:


# Train the grid search model
mlp_cv = GridSearchCV(pipe_MLP, {}, scoring='accuracy', cv=5, n_jobs=4,verbose=5).
    ↪ fit(data_norm.astype(np.float64), labels)


# In[39]:


lda_cv = GridSearchCV(pipe_LDA, {}, scoring='accuracy', cv=5, n_jobs=4,verbose=5).
    ↪ fit(data_norm.astype(np.float64), labels)


# In[39]:


gnb_cv = GridSearchCV(pipe_gnb, {}, scoring='accuracy', cv=5, n_jobs=4,verbose=5).fit
    ↪ (data_norm.astype(np.float64), labels)


# In[41]:


svc_cv = GridSearchCV(pipe_svc, {}, scoring='accuracy', cv=5, n_jobs=4,verbose=5).fit(
    ↪ data_norm.astype(np.float64), labels)


# In[42]:


rf_cv = GridSearchCV(pipe_rf, {}, scoring='accuracy', cv=5, n_jobs=4,verbose=5).fit(
    ↪ data_norm.astype(np.float64), labels)


# In[43]:


knn_cv = GridSearchCV(pipe_knn, {}, scoring='accuracy', cv=5, n_jobs=4,verbose=5).
    ↪ fit(data_norm.astype(np.float64), labels)
```

```
# In[26]:


DPmodels = [modelNN, modelCNN]
models = [logreg_cv, lda_cv, gnb_cv, svc_cv, rf_cv, knn_cv, mlp_cv]
for model in models:
        print(model.best_params_)
algo_dict2={0:"LogReg",1:"LDA",2:"GNB",3:"SVM",4:"RdF",5:"KNN",6:"MLP",7:"NN",8:"
    ↪ CNN"}
algo_labels = list(algo_dict2.values())



# In[27]:


scores=[]
plotScores=[]
plotPC=[]
plotTC=[]
plotPred=[]
plotTest=[]
plotTime=[]
plotTimes=[]

for i, read_distance in enumerate(SNR_values):
    X_train, X_test, y_train, y_test = train_test_split(scaler.fit_transform(
        ↪ full_chirp_RCS[i,:,:]), fullRD_Labels[i,:], test_size=0.3, random_state=1)
    plt.plot(X_test[700,:])
    plt.show()
    #print(X_test.shape)
    for a, model in enumerate(models):
        print(algo_dict2[a])
        tic= tt.perf_counter()
        y_pred = model.predict(X_test)
        toc= tt.perf_counter()
        scores.append(accuracy_score(y_test, y_pred))
        plotTime.append(toc-tic)
        plotPred.append(y_pred)
        plotTest.append(y_test)

    for c, modelNN in enumerate(DPmodels):
        print(algo_dict2[7+c])
        X2_test = X_test
        pred =[]
```

```
            tic= tt.perf_counter()
            y_pred = modelNN.predict(X2_test)
            toc= tt.perf_counter()
            for result in y_pred:
                pred.append(np.argmax(result))
            y_pred = pred
            scores.append(accuracy_score(y_test, y_pred))
            plotTime.append(toc-tic)
            plotPred.append(y_pred)
            plotTest.append(y_test)
        scores.append(read_distance)
        plotTimes.append(plotTime)
        plotPC.append(plotPred)
        plotTC.append(plotTest)
        plotScores.append(scores)
        plotTime=[]
        print(scores)
        plotPred=[]
        plotTest=[]
        scores=[]


# In[48]:


plotScores = np.array(plotScores)

f2, (ax1, ax2) = plt.subplots(1,2,figsize=(17,5));

ax1.plot(plotScores[:,9],plotScores[:,7],label="Neural Network")
ax1.plot(plotScores[:,9],plotScores[:,8],label="CNN")
ax1.plot(plotScores[:,9],plotScores[:,6],label="MLP")
ax1.set(xlabel='SNR (dB)', ylabel='Detection Accuracy (%)');
ax1.set_title('Deep Learning Algorithms')
ax1.axhline(y=1/6, color='r', linestyle='-')
ax1.legend()


ax2.set_title('Supervised Learning Algorithms')
ax2.plot(plotScores[:,9],plotScores[:,3],label="SVM")
ax2.plot(plotScores[:,9],plotScores[:,4],label="RdF")
ax2.plot(plotScores[:,9],plotScores[:,5],label="KNN")
ax2.plot(plotScores[:,9],plotScores[:,2],label="Naive Bayes")
ax2.plot(plotScores[:,9],plotScores[:,1],label="LDA")
ax2.plot(plotScores[:,9],plotScores[:,0],label="Logistic Regression")
ax2.legend()
```

```
ax2.set(xlabel='SNR (dB)', ylabel='Detection Accuracy (%)');
ax2.axhline(label='Random Guessing Accuracy',y=1/6, color='r', linestyle='—')

f = plt.figure()
f.set_figwidth(15)
f.set_figheight(8)
plt.plot(plotScores[:,9],plotScores[:,3],label="SVM")
plt.plot(plotScores[:,9],plotScores[:,4],label="RdF")
plt.plot(plotScores[:,9],plotScores[:,5],label="KNN")
plt.plot(plotScores[:,9],plotScores[:,6],label="MLP")
plt.plot(plotScores[:,9],plotScores[:,7],label="Neural Network")
plt.plot(plotScores[:,9],plotScores[:,8],label="CNN")
plt.plot(plotScores[:,9],plotScores[:,2],label="Naive Bayes")
plt.plot(plotScores[:,9],plotScores[:,1],label="LDA")
plt.plot(plotScores[:,9],plotScores[:,0],label="Logistic Regression")
plt.axhline(label='Random Guessing Accuracy',y=1/6, color='r', linestyle='—')
plt.legend()
plt.title('All Algorithms')
plt.xlabel('SNR (dB)')
plt.ylabel('Detection Accuracy (%)')
plt.show()
```

Listing A.1: Python Project