Machine Learning for Packet Detection in Satellite Communications

Master Thesis
submitted to the Faculty of the
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
by

Pol Simon Campreciós

In partial fulfillment
of the requirements for the master in
MASTER IN TELECOMMUNICATIONS ENGINEERING

Supervisors

Estefanía Recayte (DLR German Aerospace Center)

Andrea Munari (DLR German Aerospace Center)

Giuseppe Cocco (UPC Universitat Politècnica de Catalunya)

Munich and Barcelona, February 2024

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

AUC	Area Under the Curve
AWGN	Additive White Gaussian Noise
BPSK	Binary Phase Shift Key
CFO	Carrier Frequency Offset
CNN	Convolutional Neural Network
CSI	Channel State Information
DL	Deep Learning
FC	Fully Connected
FSPL	Free Space Path Loss
IoT	Internet of Things
ISI	Intersymbol Interference
LEO	Low Earth Orbit
LoS	Line of Sight
ML	Machine Learning
mMTC	massive Machine Type Communications
PDF	Probability Density Function
ReLU	Rectified Linear Unit
ROC	Receiver Operating Characteristic
SINR	Signal-to-Interference-plus-Noise Ratio
SNR	Signal-to-Noise Ratio
SVM	Support Vector Machines

Abstract

Satellite-based IoT networks demand efficient and robust short-packet detection techniques, particularly in Low Earth Orbit (LEO) scenarios where devices operate with low power and transmit sporadically. This thesis explores and compares two approaches to address these challenges under realistic channel conditions. The first approach employs a traditional correlation-based detection method, widely regarded as optimal in noise-limited environments but subject to performance degradation under heavier traffic loads, collisions, and channel impairments. The second approach uses a supervised learning scheme based on convolutional neural networks (CNNs), designed to handle low signal-to-noise ratio (SNR) and diverse channel impairments.

Initially, both methods are evaluated under ideal, noise-limited conditions, revealing similar detection rates. However, when multiple users transmit simultaneously and random phase shifts or Doppler effects arise, the CNN consistently outperforms correlation, demonstrating greater resilience. Correlation remains attractive due to its simplicity and lower computational overhead; it also offers an inherent Doppler estimation capability when implemented as a bank of correlators. By contrast, the CNN adapts more effectively to varying channel loads and unknown scenarios, maintaining good performance in general even under severe impairments.

These results underscore the potential of machine learning for next-generation packet detection in satellite networks. Future work involves extending the CNN to estimate Doppler shifts, integrating detection and frequency estimation in a single neural framework, and further exploring hybrid solutions that combine neural networks and traditional methods for improved performance.

Acknowledgements

I would like to express special gratitude to the supervisors of the project at DLR, Estefanía Recayte and Andrea Munari, and UPC Professor Giuseppe Cocco for their support, help and advice during the project study.

I extend my thankfulness to the German Aerospace Center (DLR), which has allowed me to experience a mobility stay that has enriched me both personally and academically.

Finally, I want to thank my family and friends for their unconditional support.

Revision history and approval record

Revision	Date	Purpose
0	13/11/2024	Document creation
1	17/12/2024	Document revision
2	12/01/2025	Document revision
3	20/01/2025	Document approval

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Pol Simon Campreciós	pol.simon.camprecios@estudiantat.upc.edu
Giuseppe Cocco	giuseppe.cocco@upc.edu
Estefanía Recayte	Estefania.Recayte@dlr.de
Andrea Munari	Andrea.Munari@dlr.de

Written by:		**Reviewed and approved by:**	
Date	12/01/2025	Date	20/01/2025
Name	Pol Simon Campreciós	Name	Giuseppe Cocco
Position	Project Author	Position	Project Supervisor

1 Introduction

1.1 Problem Description

The rapid growth of satellite-based systems and Internet of Things (IoT) networks has underscored the need for efficient and reliable packet detection. Many devices operate under low-power constraints and transmit data sporadically, posing significant challenges to conventional approaches. These challenges are particularly relevant in the context of massive machine-type communications (mMTC), one of the three core 5G service areas, where a vast number of devices send small data packets simultaneously. mMTC supports various IoT applications by collecting data from sensors that help reduce energy consumption, increase operational efficiency, or enhance overall quality of life. For instance, in a logistics setting, each shipping container might sporadically send tiny status and location updates, while in agriculture or environmental monitoring, sensors in remote fields or wildlife areas periodically send soil moisture or wildlife-tracking data, consisting each transmission of just a few bytes. Given the unpredictable and uncoordinated nature of this scenario, enabling the receiver to locate and identify the packets within the received signal—i.e., packet detection—becomes a critical aspect.

This thesis addresses the previous challenges by investigating machine learning (ML)-based detection techniques, and evaluating their performance under realistic satellite channel conditions. Specifically, the thesis aims to:

1. Analyze the limitations of traditional correlation-based detection method in low Earth orbit (LEO) satellite scenarios.
2. Design and implement a machine learning-driven packet detection scheme capable of handling low signal-to-noise (SNR) ratios and channel impairments.
3. Evaluate the proposed ML-based solution in comparison to existing methods by conducting simulations under various channel impairments and traffic conditions.

In order to accomplish the thesis objectives, several key requirements and specifications have been established:

1. Scalability: The proposed detection algorithm must accommodate a large number of devices transmitting intermittently, common in IoT scenarios.
2. Low-Power Operation: Since many IoT devices have severe power constraints, the proposed detection scheme must minimize energy consumption and computational overhead on the device side.
3. Robustness to Channel Impairments: The algorithm should be resistant to low-SNR environments and impairments typical of LEO satellite links.

This thesis has been carried out at Deutsches Zentrum für Luft- und Raumfahrt (German Aerospace Center), in the Institute for Communications and Navigation, Communications division. It was inspired by the previous work done in [1]. The results were replicated to become familiar with the code and challenges related to packet detection in an impairment-free scenario. Then, further explorations were conducted to gain deeper insights into the performance of the detection algorithms, improving the initial ML approach by making it effective in harsh scenarios.

This thesis is organized as follows. Section 2 outlines current approaches to packet detection and introduces machine learning techniques for signal processing. Section 3 describes the system model, while Section 4 presents the initial setup and baseline results. Section 5 discusses the results obtained under various impairments, and Section 6 concludes with the final remarks.

1.2 Gantt Diagram

Figure 1 illustrates the project timeline followed throughout the thesis. In the first month, research on the topic was conducted, and the results from prior work were successfully replicated. Once this milestone was reached, along with a deeper understanding of the baseline results, various channel impairments were introduced to evaluate their impact on detection performance. Simultaneously, the network architecture and parameters were optimized to maximize the results. The documentation phase also constituted a significant portion of the project's duration.

Thanks to prior experience in machine learning and programming, the planned timeline fulfilled without any significant delays.

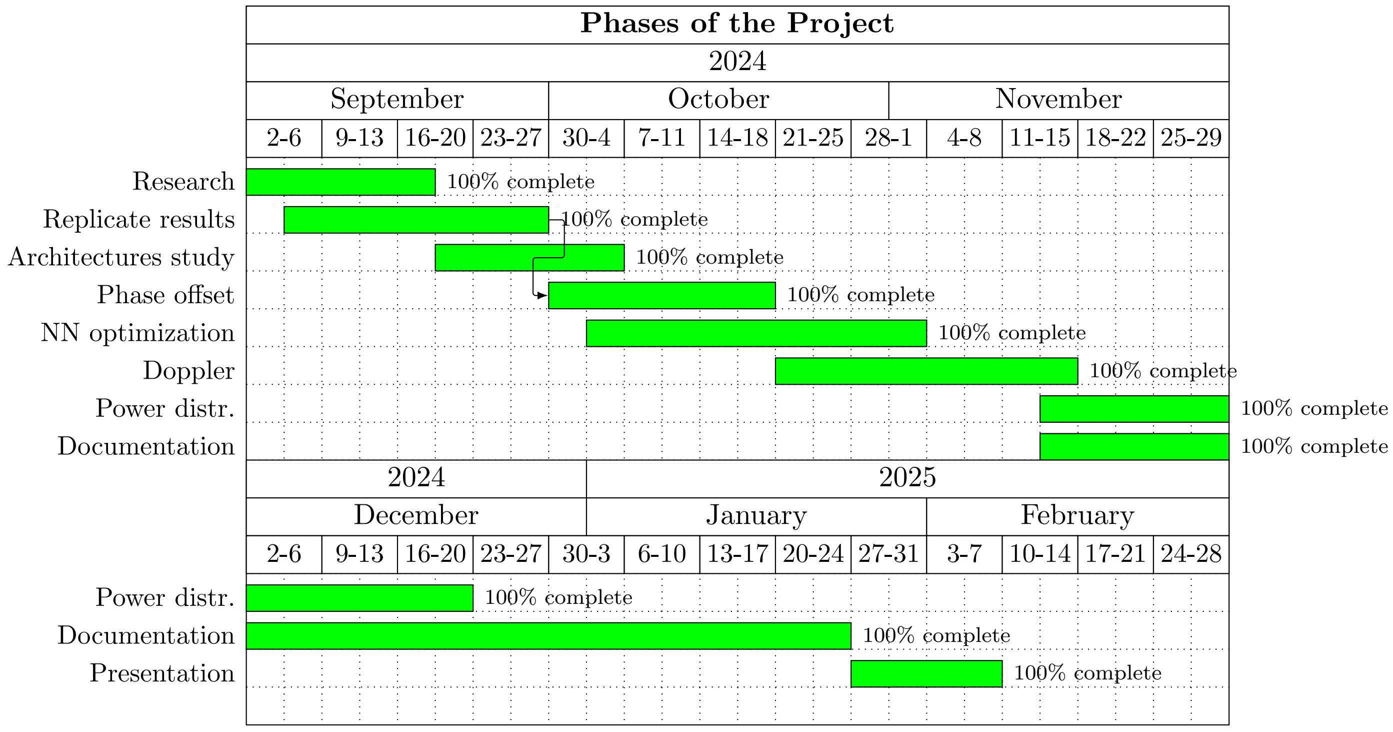


Figure 1: Gantt diagram of the project

2 State of the Art

This section provides an overview of the state-of-the-art techniques and advancements relevant to packet detection and machine learning, with a focus on their application in satellite communications.

2.1 Packet Detection

Packet detection is a fundamental process in digital communication systems, where the goal is to identify the presence of data packets in a received signal. It is a critical step in ensuring reliable communication, particularly in wireless and satellite communication systems, where signals are susceptible to noise, interference, and other impairments.

A packet is a formatted unit of data that is transmitted across a communication channel. It typically consists of a preamble, which is a known sequence of bits or symbols used for synchronization and detection, followed by a payload, which actually contains the information. The preamble acts as a unique identifier, helping the receiver detect the beginning of a packet and synchronize its processing to the incoming data.

Packet detection involves determining whether a received signal contains a packet and, if so, identifying the start of the packet. This task is complicated by several factors:

- **Noise:** The presence of random noise in the communication channel can obscure the packet signal.
- **Interference:** Signals from other users or sources can overlap with the packet, making detection challenging.
- **Channel impairments:** Fading, Doppler shifts, and other effects can alter the signal's characteristics, complicating its detection.

2.1.1 Packet Detection in Satellite Communications

In IoT satellite communications, packet detection becomes a key task as it ensures reliable and efficient identification of short, sporadic data transmissions from a vast number of devices, even in challenging conditions such as low signal-to-noise ratios, high mobility, and interference, which are inherent to satellite networks. In these scenarios, low-complexity devices transmit in an uncoordinated manner, often following grant-free random access protocols like ALOHA [2]. While this approach reduces complexity at the transmitter side allowing low-power devices to "wake up" intermittently and send data, the burden of accurately detecting bursts in real time shifts to the receiver [1].

A significant difficulty in LEO-based systems is the presence of considerable Doppler shifts due to high satellite speeds, which can severely degrade detection performance if not properly mitigated. One possible strategy is to pre-compensate for the expected Doppler at the transmitter, but this often requires precise orbital and timing information not always available to power-limited IoT devices [3], [4]. As a result, satellite receivers commonly rely on burst detection techniques that simultaneously handle timing and frequency uncertainties.

A widespread strategy for burst detection uses short preambles inserted at the beginning of each transmitted packet [5]. The receiver correlates the incoming signal with the known preamble, then applies a threshold to decide whether a packet is present. This "one shot" correlation-based approach simplifies implementation and can be very effective in moderate signal-to-noise ratio conditions [6], [7]. However, in the low-SNR regimes typical of satellite IoT applications, threshold selection becomes challenging: a low threshold leads to high false alarm rates, whereas a high threshold risks missing legitimate packets. Additionally, when multiple users collide or when transmissions suffer from intersymbol interference (ISI) due to imperfect filtering, correlation-based detection may lose accuracy [8].

An alternative line of research employs hypothesis-testing detectors derived or approximated from optimal likelihood methods [9], [10], [11], [12]. These detectors compare a test statistic against a threshold to declare the presence of a preamble. While some of these methods offer near-optimal performance at moderate SNR, recent studies emphasize that the very low SNR conditions in satellite IoT can significantly degrade their performance [3]. Furthermore, satellite links often suffer from channel impairments such as fading and large attenuation due to the long travel distances, adding another layer of complexity to synchronization and detection.

Despite the challenges, correlation-based and likelihood-based detection strategies remain the cornerstone of burst-mode satellite receivers. They strike a balance between practical implementation requirements and acceptable performance. Indeed, standardization efforts and industrial implementations continue to refine these methods, focusing on adaptive thresholding, more robust Doppler correction, and improved frame synchronization procedures. By systematically addressing the unique constraints of satellite channels (long propagation delays, high Doppler shifts, bursty traffic, and low SNR), ongoing research and development aim to ensure that packet detection in LEO constellations can keep pace with the growing demands of global IoT services.

2.2 Machine Learning

Machine learning (ML) is a field of artificial intelligence that enables computational systems to learn from data, identify patterns, and make decisions with minimal human intervention. Traditional ML algorithms include methods such as linear regression, decision trees, and support vector machines (SVM), which can be used in various tasks ranging from classification and regression to clustering and outlier detection [13], [14]. Over the past decade, the explosion of available data and advances in computing hardware have paved the way for deep learning (DL), a subfield of ML centered on artificial neural networks with many layers. These deep architectures shine at automatically extracting complex features from raw data, making them exceptionally powerful for a range of applications [15].

The branch of ML more suitable is often determined by the nature of the problem tackled, as different approaches work better for different goals. ML tasks are commonly divided into supervised learning, unsupervised learning, and reinforcement learning.

- **Supervised Learning:** In supervised learning, each training sample comes with a corresponding label or target value. The algorithm's goal is to learn a function that maps inputs to outputs accurately. Typical supervised tasks are classification (where the model predicts discrete labels, such as whether a packet is present in a burst) and regression (where the model outputs continuous values, such as predicting channel quality indicators). The performance of supervised methods usually depends on the size and representativeness of the labeled dataset.
- **Unsupervised Learning:** Unsupervised learning deals with unlabeled data and seeks to uncover underlying structure or patterns within it. Clustering is a prime example in signal processing, where the objective might be to group similar signal characteristics without any prior labels. Techniques such as dimensionality reduction can also be applied to high-dimensional waveform data, simplifying subsequent processing steps or highlighting notable features.
- **Reinforcement Learning:** Reinforcement learning differs significantly from the above paradigms. Here, an agent interacts with an environment by performing actions, and it learns an optimal policy by maximizing a reward function. In communication networks, reinforcement learning can be used for dynamic resource allocation or adaptive beamforming, where the system continually adjusts its parameters in response to feedback about network performance [14].

2.2.1 Neural networks

Deep learning involves neural networks composed of multiple layers capable of learning intricate representations of data. Each layer typically applies a linear transformation to its inputs, followed by a nonlinear activation function (e.g., ReLU, sigmoid). By stacking many such layers, these networks can capture highly complex, hierarchical patterns and have thus become state-of-the-art in numerous domains [15]. Figure 2 depicts a basic architecture of a neural network.

Deep learning models can include a wide range of layer types, each designed to handle different aspects of the learning process. Although the appropriated choice can vary depending on the task (computer vision, natural language processing, etc.), the following layer types are among the most commonly used in modern deep learning architectures:

- **Fully connected layers** connect each neuron in one layer to every neuron in the next layer, offering a straightforward way to learn mappings but quickly becoming computationally expensive for large input spaces.
- **Convolutional layers** (used in convolutional neural networks, CNNs) apply local filters to extract localized features, making them well-suited for images, spectrograms, or short-time segments of signals [16].
- **Recurrent layers** such as Long Short-Term Memory (LSTM) cells maintain hidden states to process sequential or streaming data, enabling the network to capture temporal dependencies [17].
- **Transformer layers** have recently gained attention for their attention-based mech-

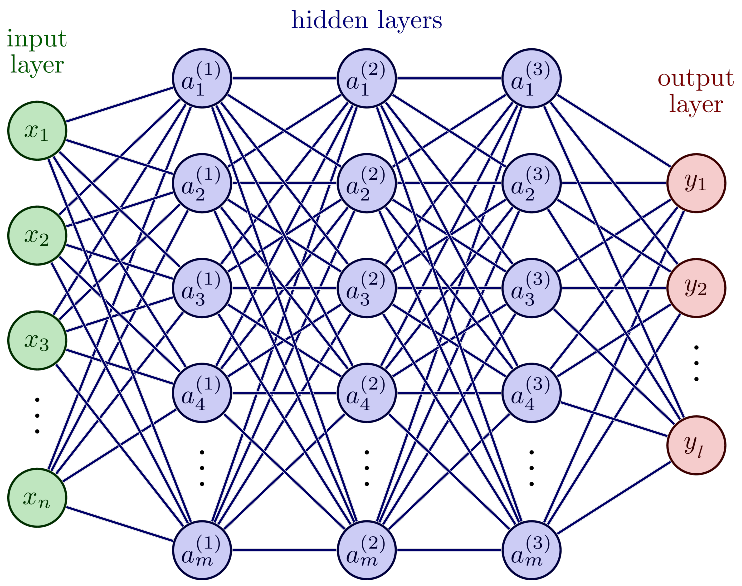


Figure 2: Example of a neural network architecture with n input features, 3 hidden layers of m elements, and l outputs.

anism, which models relationships across long input sequences efficiently and is increasingly investigated for signal processing tasks involving time-series or spectral data [18].

Deep networks typically undergo a training phase during which labeled data are passed through the network, producing predictions compared against ground truth labels via a loss function. The parameters are iteratively updated through backpropagation and optimizers like stochastic gradient descent or Adam. Once trained, the network moves to an inference phase, where weights remain fixed, and the system processes new input samples to generate predictions as classification scores, regression values, or any other learned output. This ability to learn features directly from data has proven especially beneficial in complex, high-dimensional domains, ranging from computer vision to communications.

2.2.2 Machine Learning in Signal Processing

Machine learning techniques are increasingly adopted in signal processing and satellite communications for tasks like signal detection, classification, interference mitigation, and channel estimation. The general application of ML in signal processing can be attributed to its ability to model complex, non-linear patterns and adaptively learn from data, which is particularly valuable in the dynamic and noisy environments encountered in satellite communications. Some of these applications are:

- **Channel Estimation:** In satellite communications, channel conditions can vary significantly due to Doppler effects, atmospheric disturbances, and multipath fading. Deep neural networks such as LSTM networks can adaptively estimate the channel state information (CSI) by learning from historical and real-time data. These techniques often outperform traditional estimation approaches, especially in highly dynamic or unknown channels [19], [20].

- **Resource Allocation:** ML algorithms, particularly reinforcement learning, have been adopted to optimize resource allocation, such as power and bandwidth management, in satellite networks [21]. By learning from past allocation strategies and network states, these algorithms can help maximize spectral efficiency and minimize interference across multiple satellite beams.
- **Interference Detection:** Detecting interference in satellite communication systems is traditionally performed using rule-based algorithms and spectrum analysis tools. These methods often result in high false detection rates due to dynamic environments and overlapping signals. ML models analyze spectrum usage and signal anomalies to identify and classify interference patterns. Techniques like CNNs and SVMs are typically used. [22] showed that ML-based interference detector decreases the false detection probability by 44%, highlighting its superior performance in differentiating between genuine interference and benign signal variations.
- **Flexible Payload Configuration:** Configuring satellite payloads is often done statically, based on anticipated demand. This can lead to unmet capacity in high-demand areas or underutilized resources in low-demand areas. ML enables dynamic optimization of payload resources using real-time traffic data and historical trends. Reinforcement Learning models, for example, can adaptively allocate resources to meet changing demands [22], [23].
- **Congestion Prediction:** Predicting network congestion in satellite systems is complex due to dynamic traffic patterns, weather impacts, and the mobility of satellite beams. Traditional statistical models struggle to account for these variations. ML models, particularly time-series approaches like LSTM networks, predict congestion more accurately by analyzing historical and real-time traffic data [22].

Machine Learning in Packet Detection

Among the various ML applications in the field of signal processing, we are going to focus on packet detection. Limitations of traditional methods have driven research into ML-based solutions, which offer robustness and adaptability through data-driven approaches.

Neural networks, particularly CNNs, have been a focal point in this field. By learning patterns from raw signal data, neural networks have demonstrated improved detection accuracy compared to traditional methods, especially in scenarios involving interference and fading. For instance, in asynchronous grant-free random access systems, NN-based methods significantly outperform correlators by adapting to variable arrival times and channel conditions [24].

Another significant innovation is the use of neural networks for blind coherent combining. This technique aggregates signals from multiple antennas, maximizing SNR without requiring channel state information—a critical advantage in initial access phases of satellite communications [25]. Additionally, hybrid approaches that combine ML with classical methods, such as preprocessing with complex power delay profiles, have enhanced detection reliability under noisy conditions [26].

Applications of ML-based packet detection span satellite communications and massive machine-type communications. These systems benefit from the robustness and low computational complexity of ML solutions, which handle sporadic and uncoordinated transmissions with high accuracy [24], [27].

While promising, challenges persist. Effective training requires large datasets, and ensuring consistent performance across diverse conditions and hardware platforms remains an area for further research. Nonetheless, ML's ability to surpass traditional methods positions it as a cornerstone for future advancements in packet detection.

3 Scenario and System Model

This section provides a detailed description of the scenarios addressed in this thesis, along with the methodology used to generate the various datasets. Additionally, it outlines the system model, its architecture, and the training parameters employed.

3.1 Scenario Description

In the context of small data networks as a solution for IoT communications via LEO satellite constellations, data are generated by a vast population of terminals that transmit in a fully asynchronous and uncoordinated manner, sporadically and unpredictably. Typically, satellite beams cover a large number of users, but only a small fraction of them transmit at any given time, sending short packets with little information. Because each transmission is short, the network must be optimized for low overhead, robust coverage, and the ability to support a massive number of endpoints, rather than maximizing throughput or handling large file transfers.

With these considerations in mind, let us focus on a scenario where medium access is governed by an ALOHA policy. The packet structure employed in this thesis is illustrated in Figure 3.

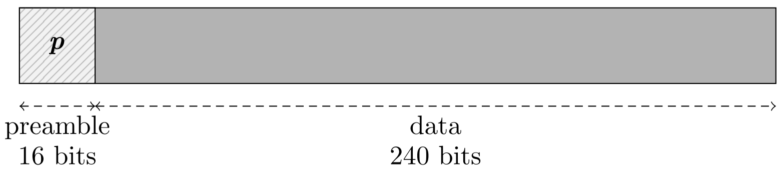


Figure 3: Packet structure considered.

Packets in our scheme are considered to be $N = 256$ bits long, with the first $L = 16$ bits reserved to a preamble (syncword) with the form

$$\boldsymbol{p} = [1110\ 1011\ 1001\ 0000]. \tag{1}$$

The sequence in (1) was originally proposed by the Consultative Committee for Space Data Systems (CCSDS), and specifically designed to perform well in the detection of short packets thanks to good auto-correlation properties [29].

These packets are then binary phase shift key (BPSK) modulated and sent through a channel with additive white Gaussian noise (AWGN). For simplicity, users are assumed to be symbol-synchronous, i.e., no timing offsets are considered [1]. Two pertinent scenarios are examined to evaluate the capabilities of detection algorithms:

- **Interference-free scenario**: On the one hand, the scenario depicted in Figure 4a is considered, in which one single node sends information and there is no overlapping packets. Thus, in this case the receiver has to deal with noise and channel impairments, but not with interference. This setup serves as a benchmark for assessing achievable performance and provides valuable insights into the behavior of various detection methods. Due to the lack of interference, this scenario will also be referred as *single user*.

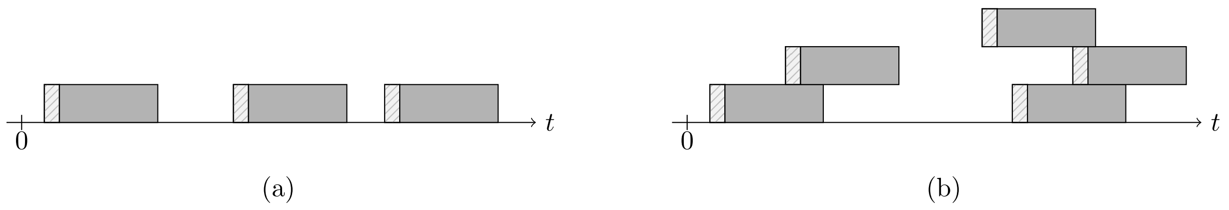


Figure 4: Scenarios contemplated: (a) interference-free scenario with non-overlapping packets and (b) interference scenario with overlapping packets.

- **Interference scenario**: On the other hand, the scenario illustrated in Figure 4b is also considered. In this case, simultaneous transmissions can occur, leading to a received signal that is affected by both noise and interference, making the detection process significantly more challenging. Different channel loads will be studied as well to examine various interference intensities. This scenario will be referred to as *multi-user*.

3.1.1 Channel Impairments

Let us define the received baseband signal as follows:

$$z_n = h \cdot x_n \cdot e^{j\phi_n} + w_n$$

where $z_n \in \mathbb{C}$ is the complex sample seen at the receiver, $x_n = A_n$ is the transmitted BPSK signal with $A_n \in \{-1, 1\}$, h and ϕ_n are, respectively, any amplitude and phase variations introduced by the channel, and $w_n \sim \mathcal{CN}(0, \sigma^2)$ denotes the additive Gaussian noise.

In the absence of any other distortion apart from the AWGN noise, assuming ideal timing, phase, and frequency synchronization, the amplitude variation would be $h = 1$ and the phase shift $\phi_n = 0$, and the received baseband signal could be described as

$$z_n = x_n + w_n.$$

However, this simplified representation deviates significantly from real-world scenarios. In practical communication systems, signals are subject to a variety of impairments that introduce additional complexity.

A. Random phase offset

A phase offset can result from various reasons such as carrier frequency offset (CFO), initial phase mismatches due to imperfect synchronization, polarization mismatches, or hardware imperfections. Static phase offsets do not degrade the correlation peak in correlation-based detection, as they only introduce a phase shift without reducing the peak magnitude. However, such offsets may affect machine learning-based detection algorithms differently, depending on how the model processes the phase information or relies on phase-sensitive features.

In this work, we will consider a phase offset that is constant through the whole packet ($\phi_n = \theta$), leading to a received signal in the form

$$z_n = x_n \cdot e^{j\theta} + w_n,$$

with $\theta \sim \text{Uniform}(-\pi, \pi)$. Notice that this added phase offset has no effect at all on the amplitude, and h remains unitary. A graphical representation of the effect of the considered random phase offset can be appreciated in Figure 5. The assumption of a constant phase offset is meaningful in this setting because it simplifies the signal model while still capturing key practical impairments. In many communication scenarios, particularly with slow-varying channels or relatively short packet durations, the phase offset introduced by the channel or hardware remains approximately constant over the observation window. This assumption allows the system to model and address phase distortions effectively without introducing unnecessary complexity.

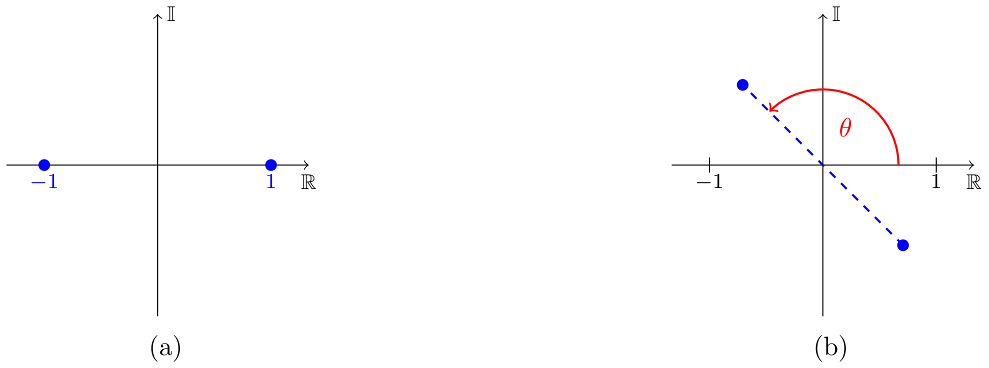


Figure 5: Comparison of a signal (a) without phase offset and (b) with an added random phase offset θ.

B. *Doppler shift*

The Doppler effect is a phenomenon observed when there is relative radial motion between a wave source and an observer, leading to a perceived change in the wave's frequency. When the source and observer move closer to each other, the frequency appears higher, and when they move apart, the frequency appears lower. In telecommunications and satellite communications, the Doppler effect significantly impacts signal reception, as the relative motion between satellites and ground users can cause frequency shifts, affecting synchronization and system performance. This effect is particularly pronounced in LEO satellite systems, where satellites move at high speeds relative to the Earth's surface. A more detailed discussion on the impact of the Doppler effect can be found in Section 5.2.

This Doppler frequency shift can be calculated as follows:

$$f_d = \frac{\Delta v}{c} f_c$$

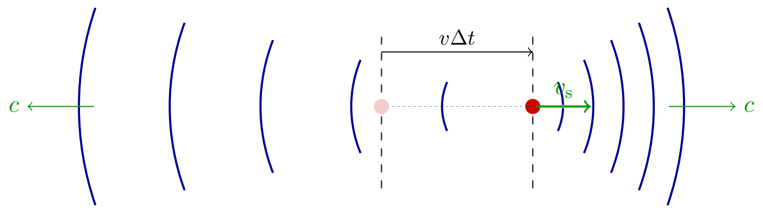


Figure 6: Representation of the Doppler effect.

being c the speed of light, f_c the carrier frequency, and $\Delta v = -(v_r - v_s)$ the opposite of the relative radial speed of the receiver with respect to the source; it is positive when the source and the receiver are moving towards each other. This effect is shown in Figure 6.

When the received signal at the satellite is downconverted to baseband, this offset introduces a phase rotation that is cumulative across symbols:

$$z_n = x_n \cdot e^{j2\pi \frac{f_d}{B} n} + w_n \text{ [6]},$$

being B the bandwidth of the signal. As a result, each symbol experiences a slightly different phase shift relative to the previous one, which means the phase of each symbol is incrementally rotated over the packet's duration. Although the Doppler shift causes this gradual phase rotation, the Doppler frequency is assumed to remain constant throughout the packet duration.

C. *Free Space Path Loss*

In satellite communication, free space path loss (FSPL) is a fundamental concept that describes the reduction in power density of an electromagnetic wave as it propagates through free space. It is influenced by two primary factors: the distance between the transmitter and the receiver, and the signal's frequency. Transmitted signals may come from different users spread over the satellite beam's coverage area, leading to varying distances from the satellite. As a result, FSPL is not uniform across the beam; users closer to the satellite experience lower losses, while those at the edges of the beam or farther away experience significantly higher losses. The formula to calculate this loss is given by

$$FSPL = \frac{P_T}{P_R} = \left(\frac{4\pi \cdot d}{\lambda}\right)^2 = \left(\frac{4\pi \cdot d \cdot f}{c}\right)^2$$

where P_T and P_R are the transmitted and received power, respectively, λ is the signal wavelength, d the distance between transmitter and receiver, f the frequency, and c the speed of light. Therefore, with this loss, the received signal becomes

$$z_n = \frac{\lambda}{4\pi \cdot d} \cdot x_n + w_n$$

This non-uniform FSPL has a direct impact on the received signal strength, affecting the quality and reliability of communication. For instance, users at the edge of the beam may require higher transmit power or more sensitive receivers to maintain a reliable link.

3.1.2 Link Budget

In this section, the link budget for the application scenario is analyzed. A link budget is a fundamental tool in telecommunications that calculates the balance of power in a communication link, accounting for all gains and losses from the transmitter to the receiver. It provides a detailed understanding of whether a system can achieve reliable communication under given conditions, ensuring the received signal is strong enough for proper detection and decoding.

As is common in many satellite communications studies, we focus exclusively on the line of sight (LoS) component and do not consider fading, which simplifies the analysis while still providing meaningful insights into system performance. The received power at the receiver for each link is computed as follows:

$$P_R = P_T \frac{G_T G_R}{L} \left(\frac{\lambda}{4\pi \cdot d} \right)^2 ,$$

where G_T and G_R are the gains of the transmitter and receiver antennas respectively, L accounts for other losses discussed shortly, λ is the wavelength, and d is the distance of the link. In Table 1 the values of the different parameters are specified, which represent a typical link budget configuration for a LEO satellite designed to provide connectivity to IoT devices. These devices typically require low data rates and operate under constraints of low power and cost, where the focus is on energy-efficient communication rather than high throughput.

The antenna radiation patterns are assumed to be pointing upwards and downwards, for the users and the satellite respectively. Best case scenario refers to a user at the center at the beam, i.e., satellite is right on top of the user and the distance separating that

Table 1: Parameters for the link budget study.

Parameter	Description	Value
P_T	Transmitters power	0 dBW
$G_{T_{max}}$	Best case transmitter antenna directivity	5.2 dB
$G_{T_{min}}$	Worst case transmitter antenna directivity	-0.6 dB
$L_{T_{rad}}$	Transmitter antenna radiation loss	0.5 dB
$G_{R_{max}}$	Best case receiver antenna directivity	12 dB
$G_{R_{min}}$	Worst case receiver antenna directivity	9 dB
$L_{R_{rad}}$	Receiver antenna radiation loss	0.5 dB
$L_{R_{pol}}$	Receiver polarization loss	1.5 dB
$L_{R_{feeder}}$	Receiver feeder loss	0.35 dB
T	Receiver noise temperature	438.93 K
L_{other}	Additional losses (implementation, atmospheric...)	0.2 dB
f_c	Carrier frequency	2400 MHz
d_{min}	Minimum range, i.e. satellite height	600 km
d_{max}	Maximum slant range	1000 km
B	Signal bandwidth	100 kHz
v_s	Satellite speed	7.5 km/s

user and the satellite is directly the satellite height. On the other hand, the worst case scenario refers to users allocated at the border of the beam. For the rest of radial distances along the satellite beam's coverage area, a linear interpolation between the best and worst values has been carried out to determine the antenna gains at each specific position.

The losses represent typical real-world factors affecting the performance of satellite communication systems. Transmitter and receiver antenna radiation losses account for inefficiencies in the antennas, such as imperfections in material or design, which result in some energy not being effectively radiated or captured. Additionally, receiver polarization loss reflects the mismatch between the polarization of the received signal and the receiver's antenna, which can lead to partial rejection of the signal.

Further, receiver feeder loss arises from energy dissipation in the cables and connectors that transmit the signal from the antenna to the receiver, while additional losses provide a general estimate for other factors like atmospheric attenuation, minor implementation inefficiencies, or other signal degradation mechanisms.

Once all these parameters are considered, the received power can be calculated with respect to the distance among the satellite and the users. This is shown in Figure 7.

After obtaining the received power, the noise power can be also computed, yielding the signal to noise ratio (SNR):

$$SNR = \frac{P_R}{\sigma^2} = \frac{P_R}{K \cdot T \cdot B}$$

where K is the Boltzmann's constant, T is the receiver noise temperature, and B the signal bandwidth. With all these parameters, the SNR values range from -2.5 dB for users at the edge of the beam's coverage area to 10 dB for users at the center of the beam.

For simplicity, the satellite's coverage range has been calculated using a flat Earth assumption. Specifically, the Pythagorean theorem was applied with a 600 km satellite altitude

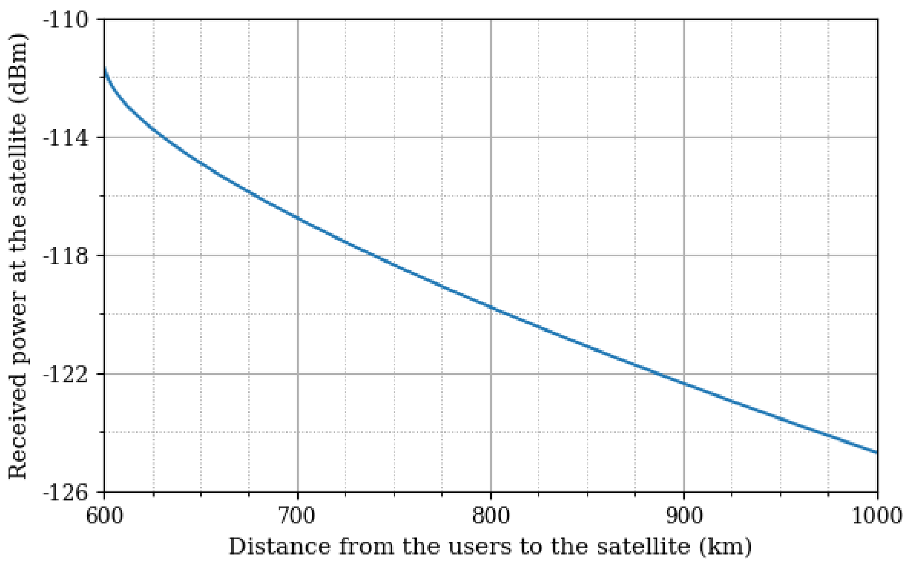
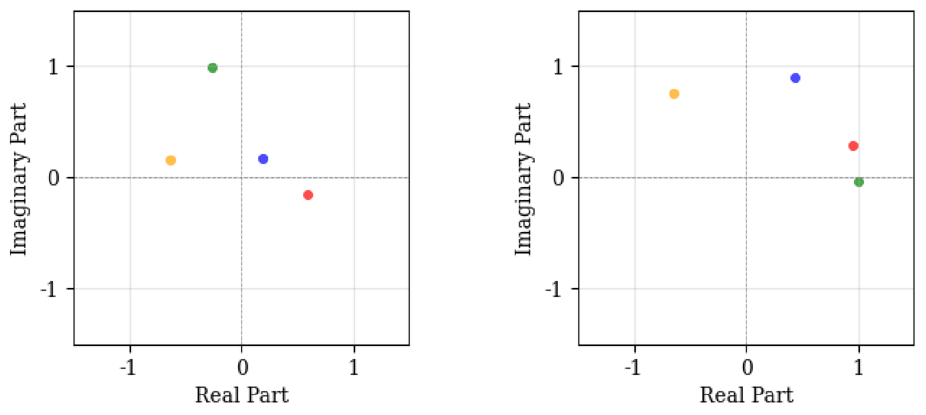


Figure 7: Link budget analysis of the scenario.



(a) SNR = -1.84 dB, $\theta = 3.01$ rad, $f_d = -3.2$ kHz

(b) SNR = 6.32 dB, $\theta = -1.20$ rad, $f_d = 18.44$ kHz

Figure 8: First four symbols of two preambles affected by different channel impairments. Green, blue, orange and red dots indicate the first, second, third and fourth symbols, respectively.

and a 1000 km maximum slant range, providing a radius of 800 km. Calculations according to [30] considering Earth's curvature can be found in the Appendix A, but it should be remarked that such an assumption does not impact significantly the results.

With this link budget and the impairments considered all together, the complete received signal is defined as:

$$z_n = \frac{\lambda}{4\pi \cdot d} \cdot x_n \cdot e^{j\left(2\pi \frac{f_d}{B} n \,+\, \theta\right)} + w_n$$

Taking the satellite velocity v_s from Table 1 into account, the maximum Doppler experienced by any user in the covered area is ±48 kHz (calculations in Section 5.2), nearly spanning the entire signal bandwidth. This substantial Doppler shift, relative to the bandwidth, greatly impacts the transmitted packet symbols.

To see how all these variations can differently affect the BPSK symbols, Figure 8 depicts the first four BPSK symbols of the preamble sequences (1, 1, 1, -1) of two different packets, seen on the received complex plane. Specifically, 8a illustrates a case with lower SNR and moderate Doppler, while 8b highlights a higher SNR but also with a higher Doppler offset. As a result, the symbol points in the complex plane deviate significantly from their nominal positions, illustrating that even a high SNR cannot preserve symbol coherence when accompanied by a large Doppler shift as in the second plot.

3.2 Detection Algorithms

This section introduces two distinct algorithms for performance evaluation: a correlation-based method and a deep learning-based approach, utilizing a convolutional neural network (CNN) to perform preamble detection directly from raw received samples at the satellite.

3.2.1 Correlation

By comparing the received signal with the known preamble sequence, correlation identifies the presence of the preamble through a distinctive peak in the correlation output, indicating alignment. This process is computationally efficient, making it suitable for real-time applications in resource-constrained environments. Moreover, correlation has been demonstrated to be optimal for preamble detection in scenarios where false alarms are dominated by noise [31].

The receiver operates via a sliding window of length L, equal to the preamble length. At each position, the correlation output is calculated by taking the dot product of the windowed segment and the preamble sequence. Mathematically, the correlation at position n can be expressed as

$$c_n = \left| \sum_{i=0}^{L-1} y_{i+n}^* \cdot p_i \right| \tag{2}$$

where c_n is the correlation output, y_j^* is the conjugated received j-th sample and p_i is the i-th symbol of the preamble sequence defined in (1). These values are then compared against a predetermined threshold μ as visualized in Figure 9; if c_n exceeds the threshold, it indicates the likely presence of the preamble at that position. The choice of this threshold dictates the trade-off between detection probability and false alarm rate.

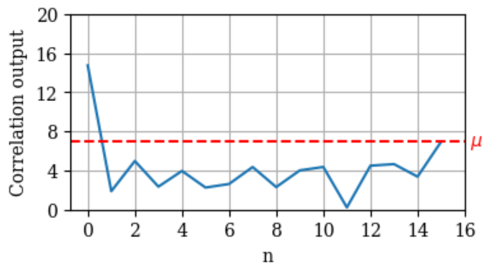


Figure 9: Correlation output of a received signal with a packet at the first position, exceeding the threshold μ.

3.2.2 Convolutional Neural Network

The detection of preambles can be effectively approached as a classification problem from a machine learning perspective. After experimenting with several neural network architectures, the final model chosen for this task is a CNN. The architecture, originally inspired by [1], consists of two 1D-convolutional layers with 48 filters of dimension 7 each, which are responsible for extracting features from the real and imaginary parts separately [24], [28]. These layers are followed by two fully connected (FC) layers of 325 and 320 neurons with dropout regularization, which ensure robust learning while mitigating overfitting. All layers employ the ReLU activation function, which introduces non-linearity and accelerates convergence during training. The described architecture is depicted in Figure 10.

Although the final layer contains four neurons corresponding to four output classes, the predictions are post-processed in a binary way: the probabilities of neurons 2, 3, and 4 are

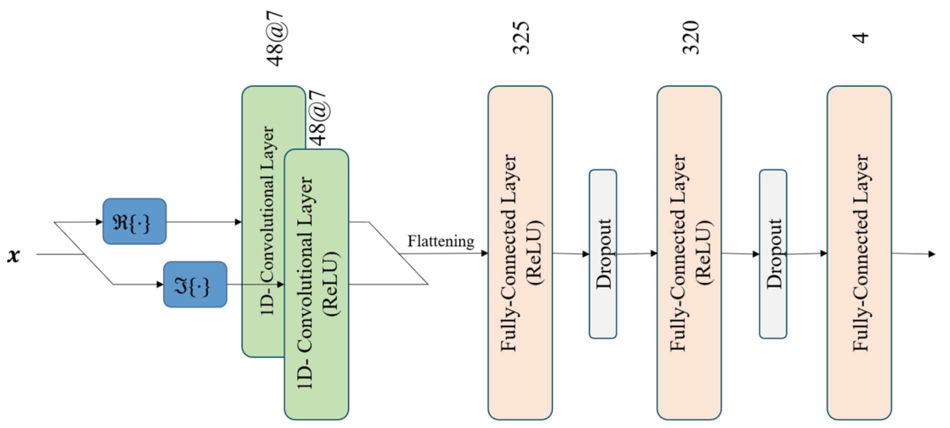


Figure 10: A schematic representation of the architecture of the proposed CNN, where the size of each layer is specified.

summed together and compared against the probability of neuron 1 (see Section 3.3). This procedure allows to mirror the threshold-based approach used in the correlation method by applying a threshold to the confidence score to determine how certain the network must be to classify an output as 0 (no preamble) or 1 (preamble).

The neural network models were trained with varying sizes of training datasets, specifically $2 \cdot 10^5$, 10^6, and $5 \cdot 10^6$ samples, and tested also on specific test datasets of $5 \cdot 10^5$ samples that matched the conditions of the training. Further results on Section 4.1 show that while increasing the training dataset size beyond 10^6 provided marginal improvements in detection performance, it significantly increased training time, making it less efficient without notable benefits.

A grid search was conducted to optimize the hyperparameters of the CNN model, ensuring the best performance for preamble detection. The search evaluated various combinations of parameters, ultimately identifying a learning rate of 0.001 and a weight decay of 0.001 for the Adam optimizer, a batch size of 512, 100 training epochs, and a dropout probability of 0.1 as the optimal configuration. The models were trained using Python 3.12 and PyTorch 2.4. All experiments were conducted on a system equipped with one Intel Core i9-13950HX processor, an NVIDIA RTX 3500 Ada Generation Laptop GPU, and 64 GB of memory.

3.2.3 Performance metrics

To evaluate and compare the performance of the correlation-based and CNN-based detection algorithms, two key metrics will be used: detection probability (P_d) and false alarm probability (P_{fa}). To define these metrics the indicator function $\mathbb{1}$ will be used. This indicator function $\mathbb{1}(condition)$ equals 1 if the condition is true and 0 otherwise, and it is used to count only the samples that satisfy certain requirements.

$$\mathbb{1}(condition) := \begin{cases} 1, & \text{if } condition \text{ is satisfied,} \\ 0, & \text{if } condition \text{ is not satisfied.} \end{cases}$$

Detection probability measures the algorithm's ability to correctly identify the presence of a preamble, reflecting its sensitivity and effectiveness in detecting valid signals. It is defined as the ratio of correctly detected preambles (true positives) to the total number of transmitted preambles (true positives + false negatives). Taking $k_j = 1$ as the true label of sample x_j indicating that it contains a preamble and y_j the output of the detection algorithm, the detection probability can be estimated over a set of s samples as

$$P_d = \frac{\sum_{j=1}^{s} \mathbb{1}\{y_j = 1, k_j = 1\}}{\sum_{j=1}^{s} \mathbb{1}\{k_j = 1\}}.$$

False alarm probability, on the other hand, quantifies the rate at which the algorithm incorrectly identifies a preamble when none exists, representing the likelihood of false positives. This is calculated as the ratio of false detections (false positives) to the total number of observations that do not represent the start of a preamble (true negatives + false positives). Mathematically, it can be expressed as

$$P_{fa} = \frac{\sum_{j=1}^{s} \mathbb{1}\{y_j = 1, k_j \neq 1\}}{\sum_{j=1}^{s} \mathbb{1}\{k_j \neq 1\}}.$$

To visualize and analyze the trade-offs between these metrics, receiver operating characteristic (ROC) curves will be used. The ROC curve plots P_d against P_{fa} for varying thresholds, providing a comprehensive view of the detection algorithm's performance. By comparing the ROC curves for both approaches, one can assess their relative strengths and weaknesses across different operating conditions.

An effective way to quantitatively compare the performance of different ROC curves is through the area under the curve (AUC) metric. The AUC provides a single scalar value that summarizes the overall performance of a detection algorithm, capturing its ability to balance detection and false alarm probabilities across all thresholds. A higher AUC indicates a better-performing algorithm, as it reflects a ROC curve that stays closer to the top-left corner, where detection is maximized, and false alarm is minimized. The ideal algorithm with ideals false alarm and detection rates would reach an AUC of 1 as depicted in Figure 11.

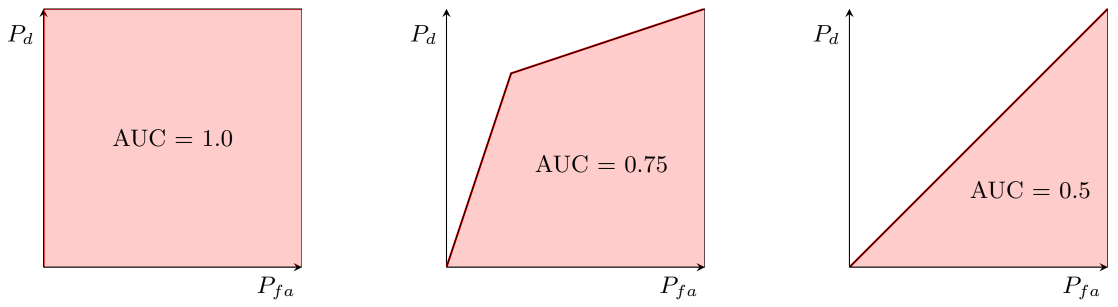


Figure 11: Representation of the AUC metric for (from left to right) perfect, sub-optimal, and random algorithms.

By leveraging the AUC, one can objectively rank and compare detection approaches while retaining the insights provided by the full ROC curve. The AUC is typically calculated using the trapezoidal rule, which provides an estimation of the integral under the ROC curve:

$$AUC \approx \sum_{i=1}^{P} \frac{P_{d,\ i} + P_{d,\ i-1}}{2} \left(P_{fa,\ i} - P_{fa,\ i-1}\right) [32] \tag{3}$$

where P is the total number of points of the ROC curve, each of them with their corresponding detection and false alarm rates.

3.3 Dataset Generation

A key objective of this thesis is to analyze and evaluate the impact of various impairments on the algorithms' ability to detect preambles. To achieve this, datasets are generated to provide insights into the effects of individual impairments as well as their combined influence. To assess detection performance across diverse scenarios, distinct training and testing datasets are created, tailored to the specific conditions of each scenario. This approach enables the development of ad-hoc ML models optimized for each scenario. Below, the process, key algorithms, and considerations involved are outlined.

To generate the dataset, the process described in Algorithm 1 was followed. All the parameters for the dataset generation are listed in Table 2.

The procedure consists of a *while* loop that does not stop until reaching the desired number of samples. In each iteration of the loop, the *create_scenario* function is responsible for generating the simulated received signal and metadata for a specific scenario. Its operation involves the following steps:

- **Initialization:** A vector is initialized, representing the simulation timeline. The

Algorithm 1 Dataset Generation for Multi/Single User Scenarios

```
Require: Parameters: $total\_samples$, $scenario\_slots$, $num\_pkts$, $sample\_length$, $packet\_length$
 1: Initialize:
 2: $samples \leftarrow []$
 3: $labels \leftarrow []$
 4: while len($samples$) < $total\_samples$ do
 5:     ($scenario$, $packet\_positions$) ← create_scenario($num\_pkts$, $snr\_level$)
 6:     $no\_packet\_positions$ ← random.choice($scenario \sqcup packet\_positions$)    ▷ Randomly
    sample positions that do not contain the start of a packet
 7:     $chosen\_positions \leftarrow packet\_positions \cup no\_packet\_positions$
 8:     for all $pos \in chosen\_positions$ do
 9:         ($input\_vector$, $label$) ← get_vector_and_label($pos$, $packet\_positions$, $scenario$)
10:         Append $input\_vector$ to $samples$
11:         Append $label$ to $labels$
```

Table 2: Parameters for the datasets generation.

Parameter	Description	Value
packet_length	Length of the packets	256 bits
scenario_slots	Length of the simulated sequence timeline measured in slots	60
num_pkts	Number of packets allocated in the sequence	{18, 30, 45, 54, 72}*
sample_length	Number of signal samples for each generated input vector	16
snr_level	Tested SNR values previous to apply Rx power distribution	{-3, 0, 3, 8} dB

*By varying the number of packets, different channel occupancies were tested.

vector is initially filled with zeros and its length depends on the *scenario_slots* parameter, having each slot a length equivalent to a packet.

- **Packet Generation and Allocation:** Packets are generated by concatenating the preamble sequence with $N - L$ random bits. The configuration of the target dataset dictates the specific impairments applied to the packets during their generation. When generating a new packet, a user position is randomly chosen according to a uniform distribution of the users within the beam coverage area. This uniform distribution is achieved by selecting a random angle and radius as follows:
  - $user\_angle = \text{Uniform}(0,\, 2\pi)$
  - $user\_radius = beam\_radius \cdot \sqrt{\text{Uniform}(0, 1)}$.

  This user position is used for calculating the Doppler shift and/or received power distribution of the specific packet if these impairments are included in the dataset. Then, each packet is randomly allocated into the vector until reaching the desired channel load specified with the parameter *num_pkts*.
- **Overlap Control:** Depending on the scenario's configuration, the function determines whether packet overlaps are allowed. If allowed, packets may occupy the same or overlapping positions, simulating interference.
- **Noise Addition:** After allocating all packets with their corresponding impairments, AWGN with appropriate noise power (according to the parameter *snr_level*) is added to the sequence.
- **Output:** The function returns the sequence containing the generated signal and a list with the packet starting indexes.

Continuing with the loop iteration, once the scenario is generated, the list of starting indexes where packets have been placed is extended with additional randomly selected positions that do not correspond to the start of a preamble. This ensures that the dataset includes samples from different classes, as required for training the classification model. The number of these non-preamble positions can be adjusted to achieve the desired class proportions.

Finally, the function *get_vector_and_label* is called for each chosen position. This function simply generates the input sample that will be fed to the network together with its corresponding label. The vector is extracted from the noisy scenario starting at the specific position and extending up to the defined *sample_length* parameter, which has been set to 16, matching the length of the preamble sequence. The real and imaginary parts of the vector are extracted and processed separately. Each part is extended by appending the sum of squares of its values, which provides additional feature information about the signal's power. This decision is motivated by the fact that portions of the signal containing packets are more likely to exhibit a higher mean power compared to segments dominated by noise. By including this feature, the model is given a simple yet effective metric to help distinguish between signal and noise, improving its ability to identify patterns relevant for classification. The processed real and imaginary components are stacked along a new dimension, creating a dual-channel vector format and thus leading to 17x2-component input vectors for the ML algorithms

$$\boldsymbol{x}_i = \begin{bmatrix} r_i, & r_{i+1}, & \dots, & r_{i+L-1}, & \sum_{k=0}^{L-1} r_{i+k}^2 \\ j_i, & j_{i+1}, & \dots, & j_{i+L-1}, & \sum_{k=0}^{L-1} j_{i+k}^2 \end{bmatrix}.$$

Since the neural network aim is to detect the initial point of the preambles, a packet is considered to have interference if and only if the interference is present during its preamble sequence. With this in mind, the labels for the classification system are the followings:

- **Label 0**: The position does not contain the start of any preamble.
- **Label 1**: The position contains the start of a preamble, and no other packets overlap or interfere with this preamble.
- **Label 2**: The position contains the start of a complete preamble, but exactly one other packet overlaps with it, causing interference.
- **Label 3**: The position contains the start of a complete preamble, but two or more other packets overlap with it, causing significant interference.

With this labeling system, it is worth noticing that in the interference-free scenario all the labels will be either 0 or 1. On the other hand, in the interference scenario there will also be samples labeled with classes 2 and 3. Moreover, it has been tested that the trained models in the interference scenario performed better with this labeling system rather than with a binary labeling system that did not account for interference (0 no preamble, 1 preamble). However, as explained in the previous section, the predictions of the multi-classification neural networks are indeed post-processed in a binary manner, accounting for classes 1, 2, and 3 all as one (i.e., preamble) versus the class 0 (no preamble).

Apart from generating the datasets for training and testing, the *create_scenario* function was also used to generate the ROC plots in a fair and consistent way. After generating a large number of scenarios, the sliding window approach was applied over the received signal to compute correlation outputs for the correlator. Simultaneously, these windowed segments of the signal were stored and formatted to generate the input features required for the neural network. These inputs were then fed into the trained neural network models to obtain predictions. By simulating distinct scenarios and repeating this process for both algorithms, the ROC curves were constructed with the exact same cases, enabling a direct comparison of their performance across various thresholds and operating conditions.

4 Initial Setup and Validation

In this section, the tuning process of the model is detailed as well as the baseline performance of the packet detection system under ideal conditions. The primary objective is to understand the fundamental limits of detection performance for both single and multi-user cases under varying SNR and channel utilization conditions.

In the initial validation, system performance is examined under different channel loads and SNR levels without including any channel impairments. However, during model tuning, these impairments were incorporated to ensure that the solution remained robust under more realistic conditions.

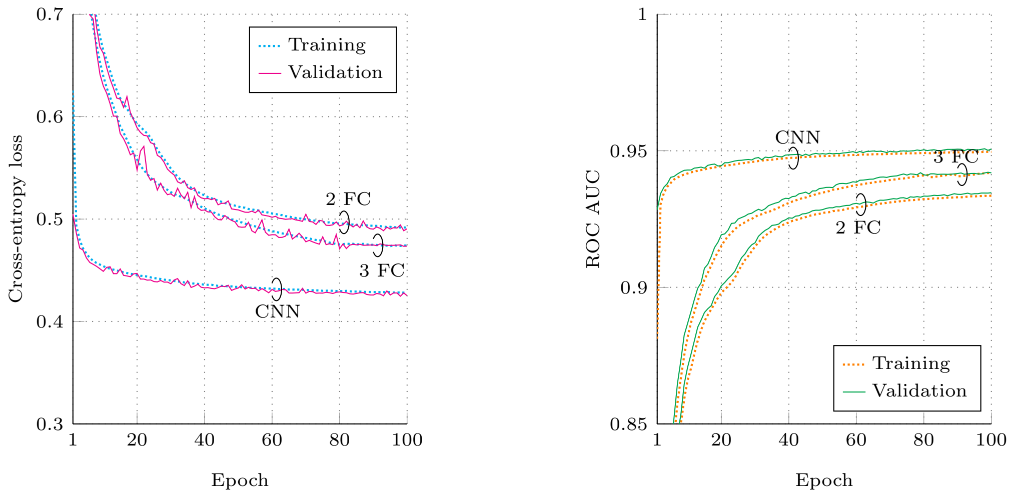
4.1 Model tuning

During model optimization, various training configurations were tested to refine the architecture. As previously mentioned, this tuning process includes Doppler shift and phase offset to ensure resilience in impaired scenarios. This approach prevents a situation where the model is optimized in an impairment-free environment only to become suboptimal once realistic conditions are considered.

Besides, the tuning has been performed using a single set of parameters: multi-user scenario with an SNR of 3 dB and a channel load of 1.2.

4.1.1 Network architecture

The initial architecture of the network was based on the design used in [1]. This foundational model (referred as “2 FC”) consisted on two fully connected layers of 325 and 320 neurons respectively and served as a starting point for further experimentation and



(a) (b)

Figure 12: Learning curves with (a) the training and validation losses (cross-entropy) and (b) the AUC of the ROC curves versus the epochs of the algorithms varying their architectures.

optimization. To explore alternative architectures, another network with one additional fully connected layer of 300 neurons was incorporated to the study (referred as “3 FC”). Inspired by other designs of the literature [24], [28], [27], two 1D-convolutional layers were also introduced to the original model to enhance feature extraction capabilities (referred as “CNN”).

Figure 12 presents the metrics recorded during the training process for the various architectures. Rather than displaying the commonly used accuracy metric, the AUC of the ROC curves is calculated with (3) and plotted, as it provides greater relevance in the context of evaluating probabilities of detection and false alarm. Cross-entropy loss was used during the training phase, as it is particularly useful in multi-classification tasks because it directly measures how well the predicted probability distribution matches the true distribution across all classes. By penalizing low probabilities assigned to the correct class, it drives the model to learn more accurate class probabilities.

These two metrics are displayed for both training and validation sets. The validation set serves as an independent checkpoint for assessing a model’s performance on data not used for training. It helps detect overfitting, guides hyperparameter tuning, and ensures that improvements seen during training translate into real predictive power rather than merely memorizing the training data. The model may actually perform slightly better on the validation set because dropout is only active during training, and is disabled during validation.

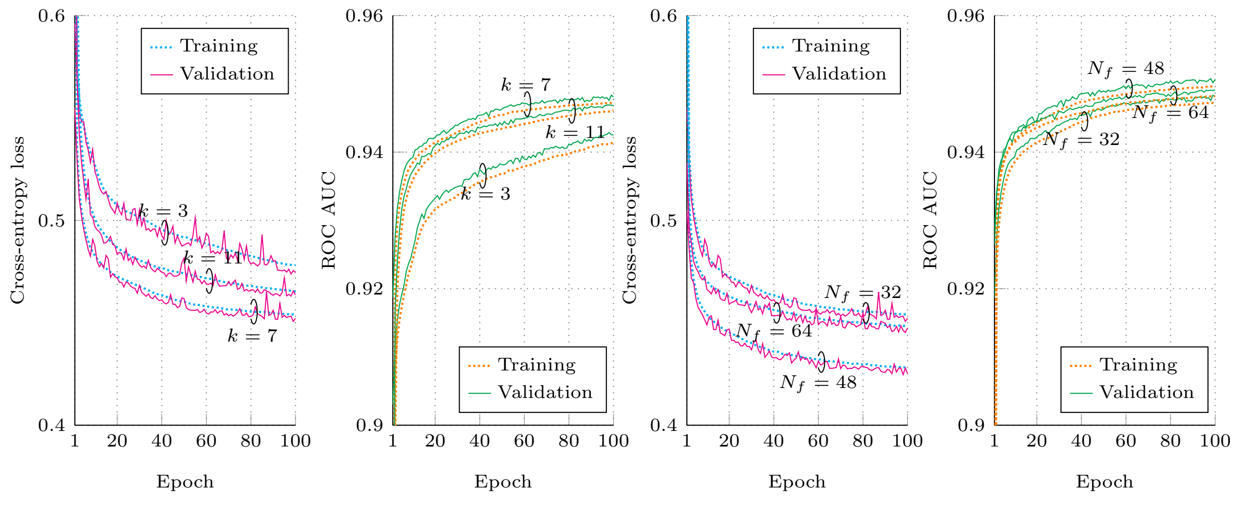
It can be observed that both metrics stabilize around 80-100 epochs, with minor improvements. Although the three tested architectures offer nice results, incorporating convolutional layers enhances the AUC further (reaching up to 0.95) with compared to the 0.93 or 0.94 AUCs offered by the networks composed only of fully connected layers, though with a slight increase in model size.

4.1.2 Convolutional layer

The effectiveness of convolutional layers is largely influenced by two key parameters: the kernel size (k) and the number of feature maps (N_f). The kernel size determines the spatial or temporal extent of the receptive field, dictating how much local context the layer can capture. Meanwhile, the number of feature maps controls the diversity of features that the layer can learn, directly affecting the model’s representational capacity.

To determine the optimal parameters for the convolutional layers, various kernel sizes and numbers of feature maps have been tested. First, Figures 13a and 13b were used to determine the optimal size of the kernel filters, setting $N_f = 32$ as a starting reference. Later, Figures 13c and 13d were obtained fixing the optimal kernel size obtained previously and varying the number of feature maps. These tests show that the optimal configuration of the convolutional layers is 48 feature maps with of dimension 7.

Pooling layers are commonly applied after convolutional layers to reduce spatial dimensions and enhance computational efficiency. This approach was evaluated, but the results shown a slight decline in performance, obtaining a test AUC of 0.9382 with the pooling layer versus 0.9456 without it. Given that the network is relatively small and model size



(a) Loss varying k with $N_f = 32$

(b) AUC varying k with $N_f = 32$

(c) Loss varying N_f with $k = 7$

(d) AUC varying N_f with $k = 7$

Figure 13: Learning curves with the training and validation losses and the AUC of the ROC curves versus the epochs of the algorithms varying the convolutional layer parameters.

is not a limiting factor, it was decided not to include a pooling layer in the architecture.

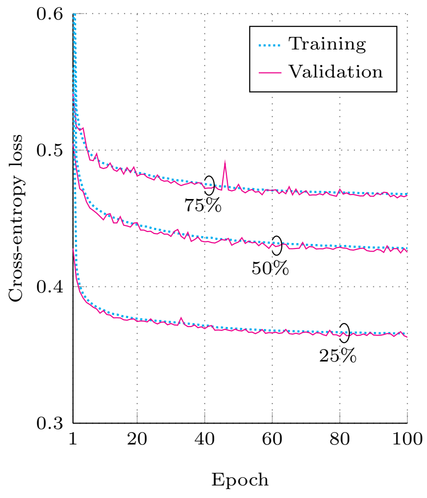
Additionally, the impact of doubling the sizes of the fully connected layers following the feature extraction by the optimized convolutional layers was explored. While this adjustment offered an AUC of 0.9464 and did not degrade the model's performance, it did not yield a noticeable improvement either. Consequently, the original fully connected layer sizes of 325 and 320 were maintained for simplicity and efficiency.

4.1.3 Dataset proportions

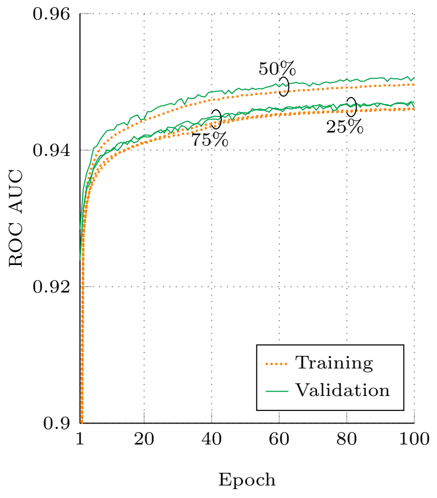
In signal processing real-world scenarios, the occurrence of non-preamble starting points far outweighs the positions containing the starting points of preambles. This imbalance reflects the natural distribution of signal data, where preamble segments represent only a small fraction of the total data. To better understand how this imbalance affects model performance, tests were conducted using varying proportions of the dataset, adjusting the ratio of preamble samples to non-preamble samples.

Figure 14 show the results obtained trying different proportions of preamble and non-preamble samples. When it comes to the loss, training with only 25% may seem the best option. Nonetheless, when looking at the AUC plots it can be appreciated that training with equally proportioned classes still offers a higher result than the other imbalanced options. This indicates that the model may prioritize minimizing the loss by confidently predicting the majority class, thereby reducing the overall loss. However, this approach can compromise its ability to accurately identify preamble samples, leading to lower performance in terms of AUC, which reflects the trade-off.

Another effect of training with differently balanced datasets was the significant bias in



(a)



(b)

Figure 14: Learning curves with (a) the training and validation losses and (b) the AUC of the ROC curves versus the epochs of the algorithms varying training classes proportions. Percentages of preamble class are indicated.

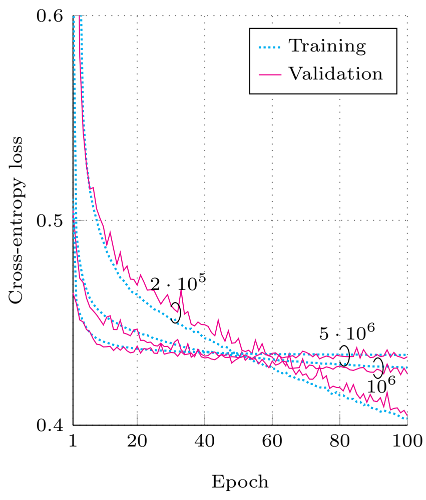
threshold values, despite producing similar ROC curves and AUC values. For example, when training with balanced classes, a threshold of 0.5 resulted in a detection of 0.86 and a false alarm of 0.13. In contrast, training with 25% of preambles led to a detection of 0.69 and a false alarm of 0.05 for the same threshold.

4.1.4 Dataset sizes

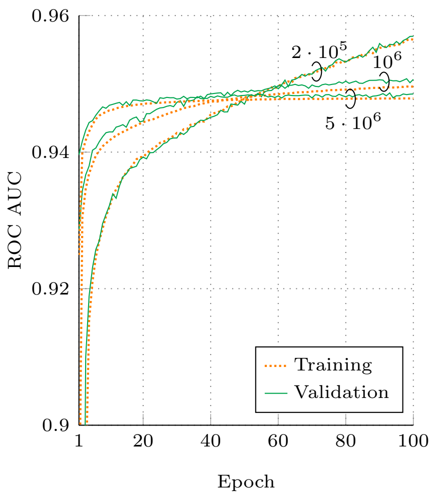
In this analysis, three different dataset sizes were evaluated: $2 \cdot 10^5$, 10^6, and $5 \cdot 10^6$. The results, as illustrated in Figure 15, show that the smaller dataset appears to achieve a competitive performance margin, particularly during the earlier epochs. However, further testing for 600 epochs revealed that while the training metrics continued to improve, the test AUC remained constant around 93–93.5%. This indicates that the smaller dataset size limits the model's ability to generalize effectively.

On the other hand, for the larger datasets the test AUC consistently reached around 94.5%, closely aligning with the training metrics. This suggests that these larger datasets provided sufficient diversity and examples to improve the model's generalization to unseen data.

An interesting observation is that, in the smallest dataset size case, the validation results closely followed the training trends, even though the validation dataset is distinct and not used during training. This behavior is unexpected because the validation dataset should serve as an indicator of generalization, similar to the test set. A possible explanation could be that the smaller dataset size led to overfitting or specific patterns being learned that align closely with both the training and validation datasets.



(a)



(b)

Figure 15: Learning curves with (a) the training and validation losses and (b) the AUC of the ROC curves versus the epochs of the algorithms varying training dataset sizes, specified along with the curves.

4.2 AWGN scenario

After showing the optimization process of the model, let us now proceed to analyze the baseline results under no impairments at all. Firstly, the results of the interference-free case are discussed, i.e., when AWGN noise is the only thing disrupting the signal. As there is also no phase offset, this baseline setup employs a simple real-valued BPSK modulation and the transmitted symbols are directly mapped to $+1$ and -1, resulting in a straightforward signal structure.

The noise affecting the signal has a total power of σ^2 calculated according to each SNR value. Since the noise is complex, it consists of two components (i.e. real and imaginary) each with a noise power of $\sigma^2/2$. This ensures that the noise power is evenly distributed across both dimensions of the signal constellation. Figure 16 shows the I-Q planes with received packet symbols for the single user case.

Because this scenario involves only one user and no interference, the concept of channel load — which typically describes the number of simultaneous users or overall traffic — isn't applicable. Instead, we focus on the generation rate λ_r [pkt/pkt duration], representing how quickly a single user produces and transmits packets, with $\lambda_r \leq 1$.

Figures 17a and 17b show the ROC curves in this setting for generation rates of $\lambda_r = 0.3$ and $\lambda_r = 0.5$ [pkt/pkt duration], respectively. Each point of the correlation curves was obtained by varying the threshold value used to declare the presence of a preamble. This threshold was compared with the output of the correlation operation, highlighting the trade-off between detection and false alarms. Similarly, the points of the neural network curves were obtained by varying the confidence threshold for accepting the presence of a preamble. A lower confidence threshold increases the detection probability but also raises

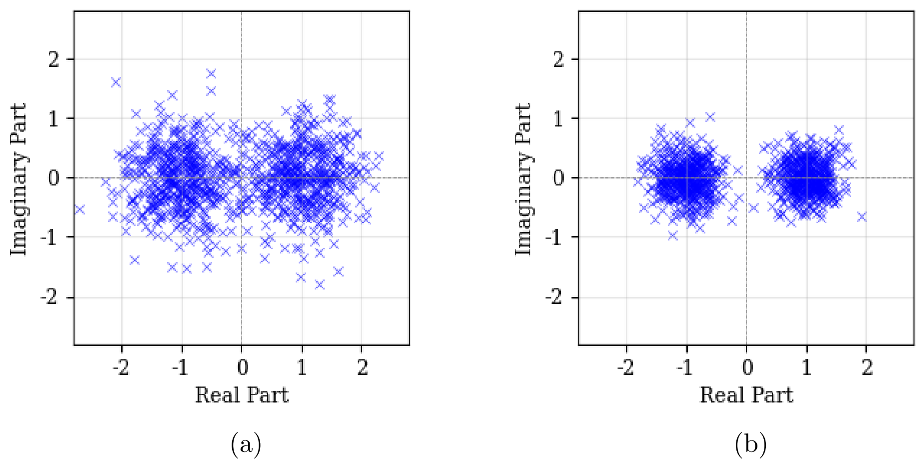
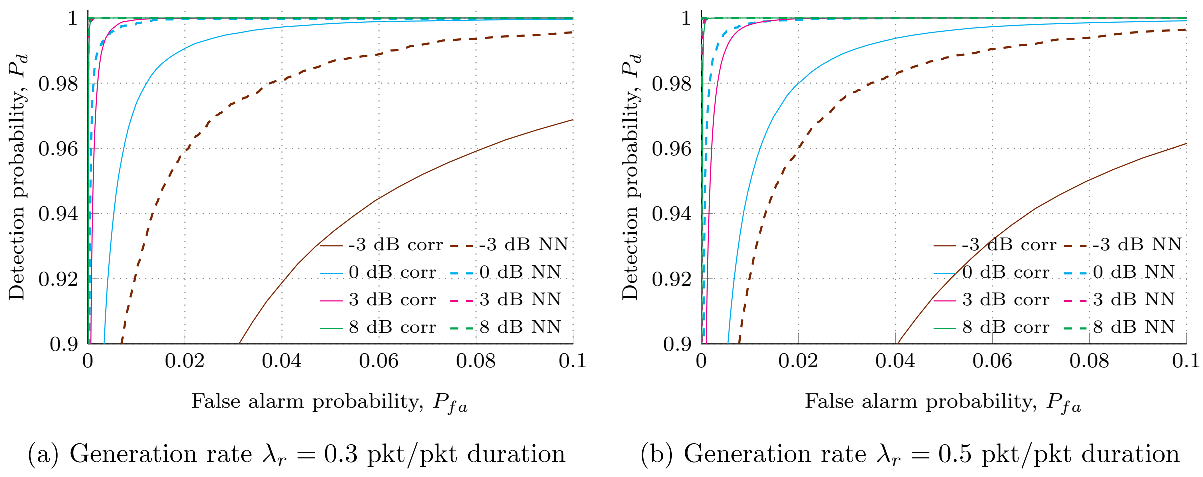


Figure 16: Visualization of received signal symbols in the I-Q plane for the single user case and SNR values of (a) 3 dB and (b) 8 dB, respectively.

the likelihood of false alarms.

Starting first with a generation rate of $\lambda_r = 0.3$, the performance of correlation-based detection and neural networks shows distinct differences between varying SNR levels. Notice that only an expanded region of the ROC curve is shown, with detection probabilities above 0.9 and false alarm probabilities below 0.1, focusing on practically relevant values. The most significant difference between both algorithms is observed at -3 dB SNR, where the correlation approach is the most affected and struggles to achieve a detection probability above 0.9 without tolerating higher false alarm probabilities of around 0.03 or 0.04, while the CNN is able to provide detection probabilities above 0.9 with false alarm rates below 0.01. At higher SNR values, the CNN achieves nearly perfect detection while main-



(a) Generation rate $\lambda_r = 0.3$ pkt/pkt duration (b) Generation rate $\lambda_r = 0.5$ pkt/pkt duration

Figure 17: Detection probability as a function of the false alarm considering AWGN scenario and no impairments for SNR $= -3, 0, 3, 8$ dB. Dashed and dotted curves represent the results obtained by correlation and neural network, respectively.

taining minimal false alarm rates, showcasing their superiority in both low and high-SNR conditions. Correlation performs robustly, achieving near-perfect performance with just 8 dB SNR. Recall that correlation is considered optimal for detecting preambles when false alarms are primarily caused by noise, which occurs under low channel occupancy (i.e., few packets). However, as the packet count increases (adding also more payloads to the signal), false alarms are no longer noise-dominated but rather triggered by these payloads. Even so, the correlator's performance degradation remains moderate, and its simplicity counterbalances this drawback.

In the generation rate $\lambda_r = 0.5$ graphic, a deterioration in the performance of the correlation is observed. In contrast, the CNN maintains the same results as with lower generation rate. This behavior could be explained by the increased number of packets, which may result in sequences within the packet payloads resembling the preamble more closely, leading to more frequent peaks in the correlation output. It is important to note that the correlation method is limited to detecting the specific structure of the signal and is not designed to generalize or learn patterns. Conversely, the consistency of the CNN's results as the generation rate increases suggests that the network is robust in scenarios where noise is the only issue (without multi-user interference) and its performance is robust with respect to the generation rate. Table 3 offers a comparison of the performance for this specific generation rate.

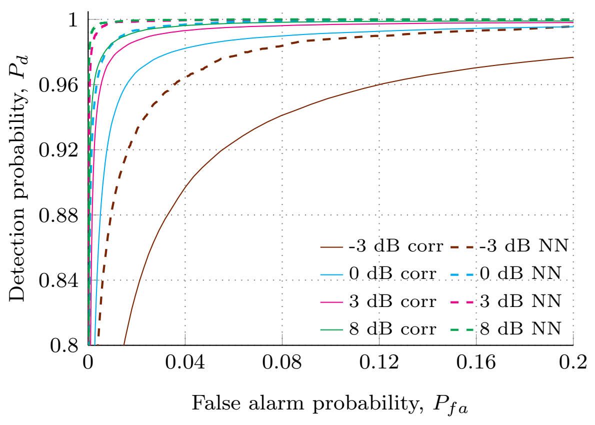
Table 3: Detection probabilities for varying false alarm probabilities for the single user scenario and a generation rate of $\lambda_r = 0.5$ pkt/pkt duration.

	Correlator			**CNN**		
SNR (dB)	$P_{fa} = 0.005$	$P_{fa} = 0.01$	$P_{fa} = 0.02$	$P_{fa} = 0.005$	$P_{fa} = 0.01$	$P_{fa} = 0.02$
-3	0.6125	0.7234	0.8270	0.86	0.9196	0.9601
0	0.8902	0.9518	0.9789	0.9955	0.9983	0.9994
3	0.9904	0.9980	0.9997	1.0	1.0	1.0
8	1.0	1.0	1.0	1.0	1.0	1.0

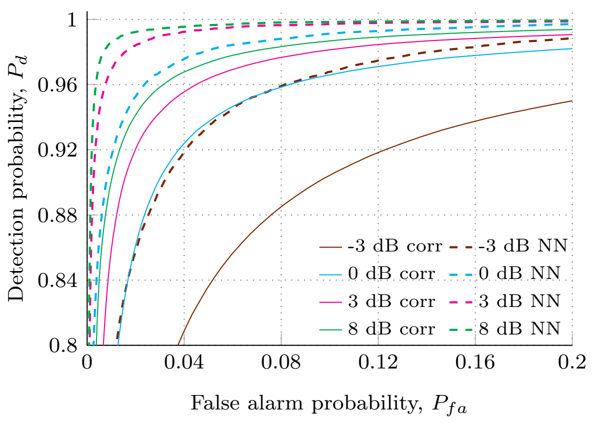
4.3 Interference scenario

We now turn our attention to a scenario of practical significance for small data packet transmission in mMTC applications, where multiple users simultaneously contend for the channel. In this case, the uncoordinated nature of the access protocol policy leads to potential collisions, making the detection of packets affected by interference notably more challenging. This scenario is depicted in Figure 18, showing the performance of the detection algorithms under channel loads of 0.3, 0.75, 0.9, and 1.2 [pkt/pkt duration]. Notice that now the plots comprise the detection rates above 0.8 and false alarm rates below 0.2.

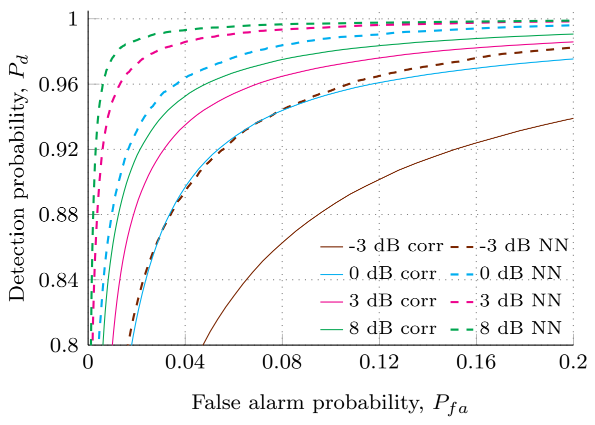
As anticipated, all methods experience a decline in performance due to the lower signal-to-interference-plus-noise ratio (SINR) at the receiver. For instance, examining the -3 dB curves under a channel occupancy of 0.3 (where multi-user interference is relatively low) shows that even this moderate interference is enough to reduce the detection rate of both methods by 0.02 points at a false alarm level of 0.04. This deterioration becomes more noticeable as the channel load increases, since more collisions take place and interference



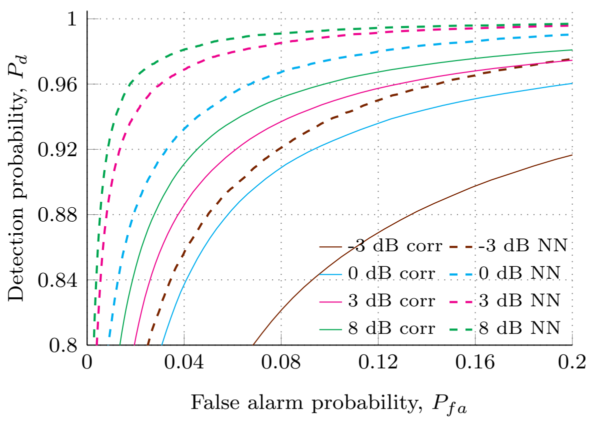
(a) Channel load $G = 0.3$ pkt/pkt duration



(b) Channel load $G = 0.75$ pkt/pkt duration



(c) Channel load $G = 0.9$ pkt/pkt duration



(d) Channel load $G = 1.2$ pkt/pkt duration

Figure 18: Detection probability as a function of the false alarm considering interference scenario for SNR $= -3, 0, 3, 8$ dB. Dashed and dotted curves represent the results obtained by correlation and neural network, respectively, for different channel loads.

increases. This effect is visualized in Figure 19, which shows the preambles class distribution for the different channel loads. Higher channel loads lead to an increase in preambles with one or multiple interferers. For instance, with a channel load of $G = 0.3$, almost 80% of the preambles have no interference, as the overlapping likelihood is low. On the other hand, for a channel load of $G = 1.2$ more than 70% of the preambles have at least 1 interferer.

Going back to the ROC plots, the more pronounced the curves are (resembling a knee shape), the easier it becomes to identify the optimal point (the one closest to the upper left corner). Nonetheless, as the curves become smoother, determining an optimal point becomes more challenging. For this reason, to compare the algorithms, a maximum tolerable false alarm probability has been defined, and the detection performance achievable without exceeding this false alarm level has been evaluated. In this scenario, a maximum tolerable false alarm has been determined of $P_{fa} = 0.05$. Table 4 presents the detection probabilities for different channel loads and SNR values, comparing the performance of

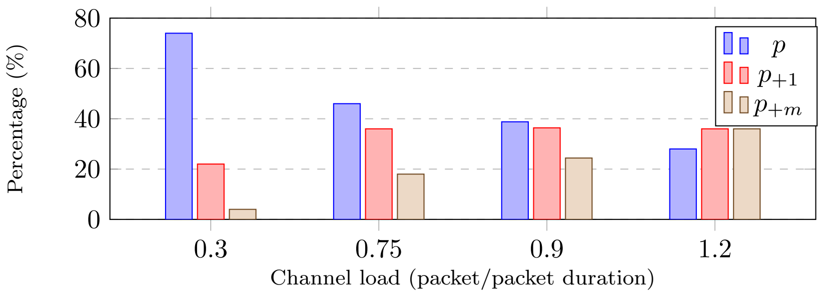


Figure 19: Impact of channel load on interference. Labels indicate: (p) preamble, (p_{+1}) preamble with one interferer, (p_{+m}) preamble with multi-interferers.

both algorithms under the fixed false alarm rate.

For the correlator, detection probability decreases consistently as the channel load increases, with the most significant degradation occurring at -3 dB. For instance, at $G = 0.3$, the detection probability is 0.9131, whereas at $G = 1.2$, it drops to 0.7534, highlighting the correlator's sensitivity to interference and increasing load. This trend persists across all SNR levels, though the impact is less pronounced at higher SNRs. At 8 dB, the correlator achieves nearly perfect detection at low channel loads, but the performance declines modestly at higher loads.

The CNN, in contrast, demonstrates superior robustness to increasing channel load. While a slight reduction in detection probability is observed at higher loads, the degradation is far less pronounced compared to the correlator. At -3 dB, the CNN maintains a detection probability of 0.9718 at $G = 0.3$, which only reduces to 0.8805 at $G = 1.2$. Similarly, at higher SNR values, the CNN consistently outperforms the correlator, with detection probabilities remaining above 0.9444 even at the highest channel load.

These results underscore the resilience of the CNN to interference and its ability to adapt to challenging multi-user scenarios, particularly at low SNR levels. In contrast, the correlator's performance is more significantly impacted by increasing channel loads, highlighting its limitations in environments with higher levels of interference.

These results set a strong foundation for exploring more realistic scenarios, where additional impairments are introduced. The following chapter delves into these scenarios, evaluating the impact of these impairments on detection performance and further comparing the effectiveness of the proposed algorithms.

Table 4: Detection probabilities for varying channel loads and a false alarm rate of $P_{fa} = 0.05$ for the multi user scenario.

	Correlator				**CNN**			
SNR (dB)	$G = 0.3$	$G = 0.75$	$G = 0.9$	$G = 1.2$	$G = 0.3$	$G = 0.75$	$G = 0.9$	$G = 1.2$
-3	0.9131	0.8361	0.8074	0.7534	0.9718	0.9345	0.9125	0.8805
0	0.9853	0.9372	0.9150	0.8641	0.9976	0.9811	0.9704	0.9447
3	0.9944	0.9637	0.9461	0.9048	0.9996	0.9940	0.9888	0.9757
8	0.9970	0.9736	0.9611	0.9265	0.9998	0.9965	0.9945	0.9851

5 Results with Channel Impairments

Building upon the baseline evaluations and model refinements described in the previous chapter, this chapter focuses on assessing the performance of the detection algorithms under progressively more realistic and challenging channel conditions. Until now, we have explored idealized scenarios, starting from simple AWGN environments without interference and then introducing multi-user overlapping signals. However, as real-world communication systems rarely operate under such controlled conditions, it is essential to extend the analysis to more adverse scenarios that better reflect practical deployments.

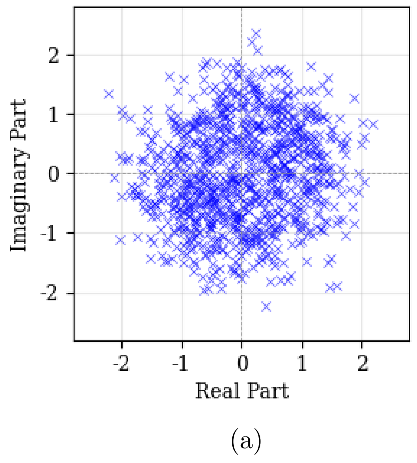
In this chapter, a comprehensive set of results obtained from simulations that incorporate various channel impairments is presented. Each of these impairments poses unique challenges to both classical and machine learning-based detection approaches.

5.1 Phase offset

We now introduce a random constant phase offset into the system. In this scenario, the transmitted symbols, still employing a simple real-valued BPSK modulation scheme, are directly mapped to $+1$ and -1 at the transmitter. However, upon reception, each packet is affected by a constant, yet randomly selected phase offset. The assumption of a constant phase offset for each received packet is justified by the nature of short packets and their limited time on air ($t_{air} = N \cdot T_{sym} = N \cdot \frac{1}{B} = 2.56$ ms).

The noise model remains unchanged from the baseline case, with complex noise of total power σ^2 applied to the signal at a given SNR. As before, this noise is split equally between the real and imaginary components, ensuring a uniform distribution across both dimensions of the signal constellation. Due to the imposed phase offset, the received symbols will now appear rotated in the I-Q plane, as depicted in Figure 20.

Comparing these figures to the previous ones without phase offset from Figure 16 high-



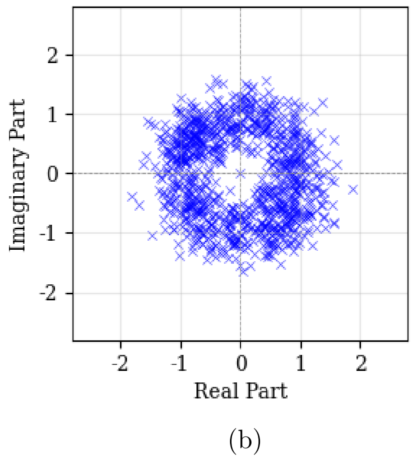


Figure 20: Visualization of received signal symbols in the I-Q plane for the single user case, added random phase offset and SNR values of (a) 3 dB and (b) 8 dB, respectively.

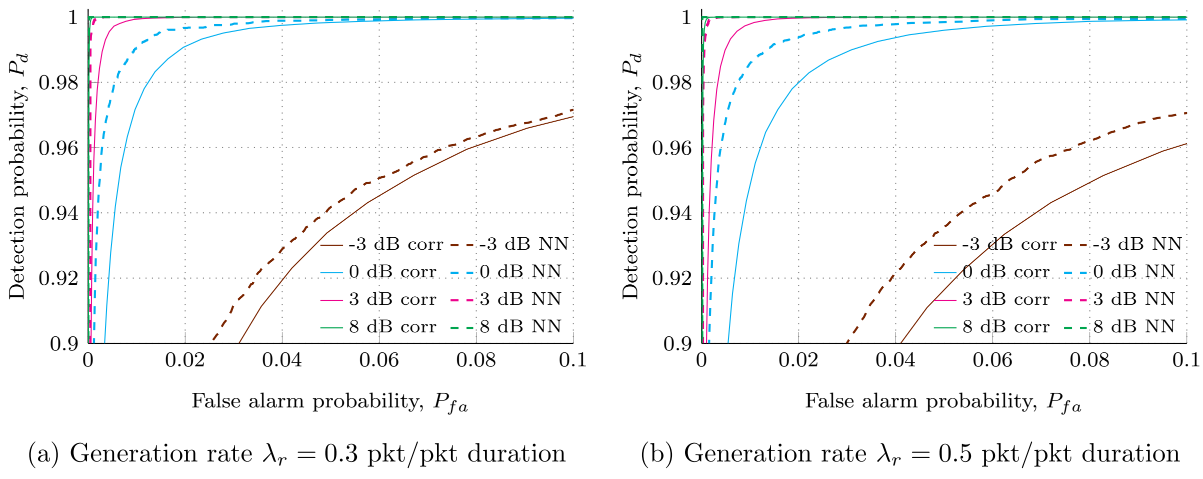
lights the significant impact of introducing a random constant phase offset. In the non-offset case, the BPSK symbols are primarily spread along the real axis, forming two distinct clusters centered around +1 and –1, with the noise scattering them slightly into the imaginary dimension. As a result, the received constellations without phase offset visibly resemble two fairly well-defined clouds.

In contrast, once a random constant phase is introduced, these two clusters become effectively rotated in the I-Q plane. Since the phase offset is constant for each packet but varies randomly from one packet to another, the aggregate effect when visualizing multiple preambles is that the points are no longer aligned along the real axis. Instead, they become distributed around the entire I-Q plane, forming more circular or ring-shaped patterns, whose diameter and thickness depend on the noise level.

A. Interference-free scenario

Following the previous procedure, first the single-user case will be studied, where no interference from other transmitters occurs. In this setting, the introduction of a random phase offset has distinct consequences for the two detection methods, reflected in Figure 21. In general, both algorithms maintain excellent performance, achieving detection rates above 0.98 and false alarm rates below 0.02 at all but the lowest SNR.

Despite adding the random rotation, the correlation-based detector remains largely unaffected because, by its definition in (2), it computes the cross-correlation between the known preamble and the received signal, taking the magnitude of the correlation output. As a result, the correlation peak is invariant to constant phase rotations, causing the method's performance to remain virtually unchanged compared to the no-offset scenario. Even with rotated preambles, the measured detection probability and false alarm probability show minimal deviation, reinforcing the robustness of correlation in single-user scenarios dominated by noise.



(a) Generation rate $\lambda_r = 0.3$ pkt/pkt duration (b) Generation rate $\lambda_r = 0.5$ pkt/pkt duration

Figure 21: Detection probability as a function of the false alarm considering single-user scenario and random phase offset for SNR $= -3, 0, 3, 8$ dB and generation rates λ_r of (a) 0.3 and (b) 0.5.

Table 5: Detection probabilities for varying false alarm probabilities for the single user scenario, random phase offset and a generation rate of $\lambda_r = 0.5$ pkt/pkt duration.

	Correlator			CNN		
SNR (dB)	$P_{fa} = 0.005$	$P_{fa} = 0.01$	$P_{fa} = 0.02$	$P_{fa} = 0.005$	$P_{fa} = 0.01$	$P_{fa} = 0.02$
-3	0.6161	0.7262	0.8223	0.6964	0.7856	0.8618
0	0.8912	0.9486	0.9799	0.9686	0.9859	0.9938
3	0.9904	0.9979	0.9997	0.9999	1.0	1.0
8	1.0	1.0	1.0	1.0	1.0	1.0

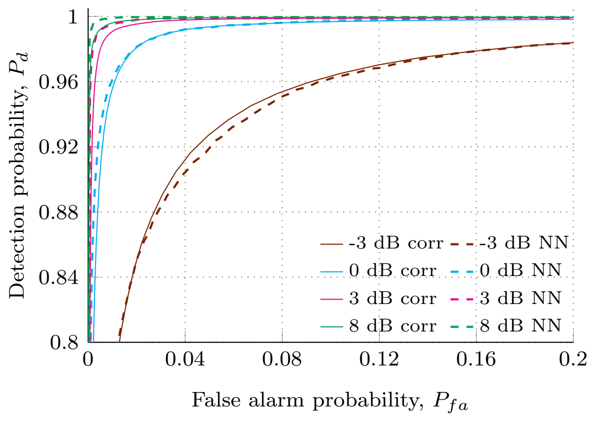
On the other hand, the CNN-based detector, although being trained including randomly rotated preambles, shows a slight degradation compared to the scenario without phase offset. This performance gap can be attributed to the added complexity of learning a broader distribution of signal orientations in the I-Q plane, whereas the no-offset model dealt with a more uniform pattern. Nevertheless, the CNN continues to outperform the correlation method at all SNR levels, demonstrating that even with random phase offsets the network maintains a strong detection capability.

At higher generation rates, the correlation detector experiences some deterioration similar to the no-phase-offset case. This effect can be attributed to the increase in the total number of packets, since the payloads are generated randomly, and sending more packets raises the chance that parts of the payload will resemble the preamble. As a result, there are more peaks in the correlation output, which inflates the false alarm rate and degrades overall performance. However, the additional random phase does not magnify this issue with respect the no-phase-offset case, reflecting correlation insensitivity to constant phase offsets.

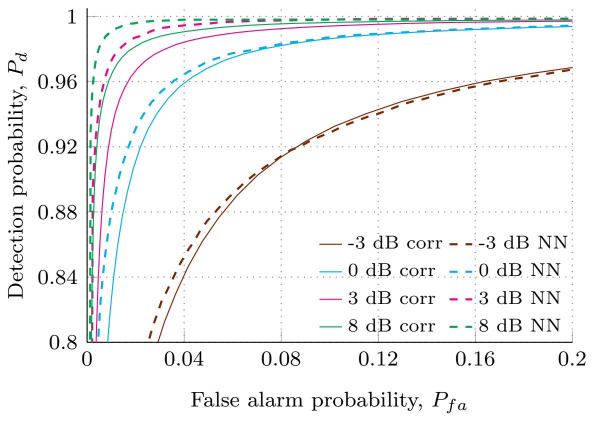
Despite being the CNN the one more affected by the random phase offset, this algorithm still demonstrates a more consistent performance across different generation rates. While there is a modest drop due to phase rotation, it does not worsen as much as the correlation when the generation rate increases, as can be appreciated by the increased gap between the curves of both algorithms when changing from $\lambda_r = 0.3$ to $\lambda_r = 0.5$. Specific detection rates with generation rate $\lambda_r = 0.5$ of both algorithms are displayed in Table 5. Correlation results mirror the non-offset ones from Table 3, whereas the CNN outcomes reveal the phase offset impact predominantly at lower SNRs.

B. Interference scenario

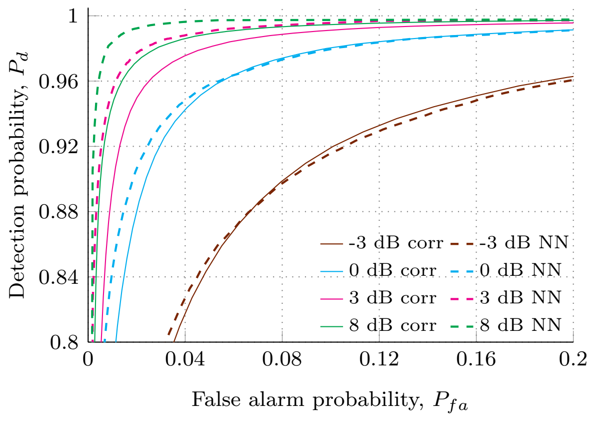
Let us now study the interference scenario. Again, results for the different channel loads are shown in Figure 22, which captures the increasing effect of the interferences as the channel load augments. At first sight, one can see that at the lower SNR values of –3 and 0 dB, both detectors produce ROC curves that are nearly overlapping, reflecting similar detection performance under these harsher conditions. In particular, for lower false alarm rates, the neural network curves sit slightly above the correlation ones, suggesting the NN retains a small advantage in rejecting false detections at these thresholds. Beyond that lower false alarm region, their performance becomes more aligned, indicating that once the false alarm tolerance is relaxed, both methods converge to a comparable detection capability.



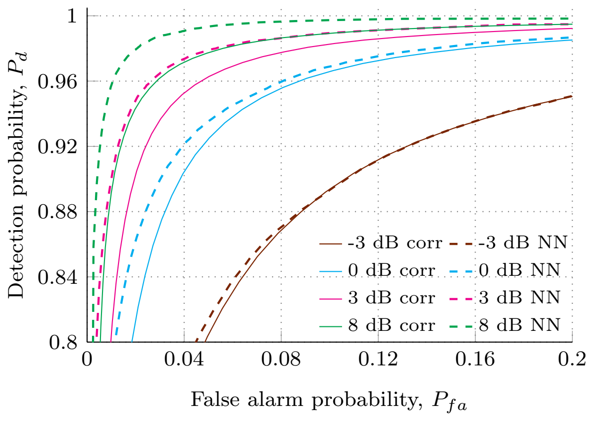
(a) Channel load $G = 0.3$ pkt/pkt duration



(b) Channel load $G = 0.75$ pkt/pkt duration



(c) Channel load $G = 0.9$ pkt/pkt duration



(d) Channel load $G = 1.2$ pkt/pkt duration

Figure 22: Detection probability as a function of the false alarm considering interference scenario and random phase offset for SNR $= -3, 0, 3, 8$ dB and different channel loads.

For ease of comparison between both algorithms and with the non-offset case, the maximum tolerable false alarm has been set to $P_{fa} = 0.05$ and specific detection probabilities for this false alarm rate are presented in Table 6. Across all SNRs and channel loads, the correlation consistently benefits from introducing random phase offsets, because the phase diversity breaks the coherence of overlapping packets and reduces interference power. Compared to non-offset results from Table 4, an improvement appears for every combination of SNR and channel load. This improvement becomes more noticeable as the channel load increases, as more overlapping situations happen. For instance, at -3 dB and channel load of 0.3, the correlation passes from a detection rate of 0.9131 in the non-offset case to 0.9276 when random rotation is present, but for the same SNR and a channel load of 1.2, the improvement is more appreciable going from 0.7534 to 0.8045.

By contrast, the CNN exhibits a more differentiated behavior. At low SNR values (-3 and 0 dB), adding random phase rotations poses an additional challenge for the network, so its performance declines relative to the no-offset scenario. Taking a look at the -3 dB and $G = 0.3$, the CNN detection changes from 0.9718 in the non-offset case to 0.9192 when

Table 6: Detection probabilities for varying channel loads and a false alarm rate of $P_{fa} = 0.05$ for the multi user scenario with random phase offset.

	Correlator				CNN			
SNR (dB)	$G = 0.3$	$G = 0.75$	$G = 0.9$	$G = 1.2$	$G = 0.3$	$G = 0.75$	$G = 0.9$	$G = 1.2$
-3	0.9276	0.8673	0.8467	0.8045	0.9192	0.8753	0.8520	0.8139
0	0.9936	0.9685	0.9552	0.9245	0.9932	0.9725	0.9577	0.9353
3	0.9982	0.9875	0.9808	0.9626	0.9988	0.9956	0.9905	0.9790
8	0.9992	0.9925	0.9890	0.9774	0.9996	0.9980	0.9971	0.9934

phase offset is introduced. Here, the noise component still dominates, and any diversity advantage from randomly rotated interference does not translate into a meaningful gain. However, at higher SNRs (3 and 8 dB), once the desired signal becomes clearer and the channel load increases (leading to more overlapping packets), the same random phase diversity actually assists the CNN in distinguishing desired preambles from interference. Consequently, for higher SNR and heavier interference, the CNN slightly outperforms its no-offset counterpart, underscoring that the helpfulness of phase diversity grows as noise becomes less of a factor and multi-user collisions become more prevalent.

Overall, random phase offsets benefit the correlation-based detection as it has a negligible effect in the single user case and improves the multi-user case. Contrarily, this phase offset has a double impact on the CNN, deteriorating its performance in the single-user scenario and low SNR multi-user scenarios, but assisting in high SNR multi-user scenarios. It could be stated that even random phase offset equalizes the performance of both algorithms, the neural network continues to maintain a little advantage in terms of detection and false alarm trade-offs.

5.2 Doppler Shift

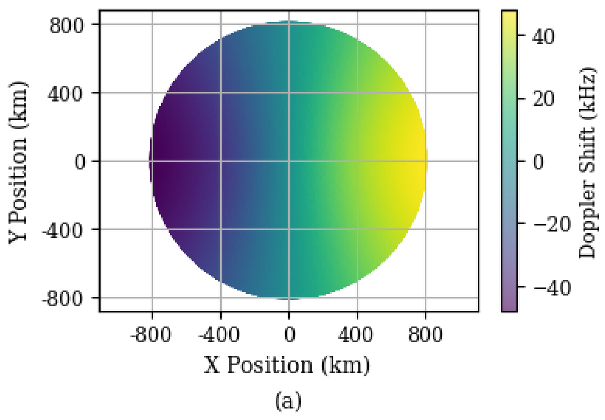
Let us now consider a time-varying mismatch scenario caused by Doppler shift. Given a 2D coordinate (x, y) of a user on the ground plane within the satellite beam's coverage area, the procedure to calculate the Doppler is the following:

- Calculate the ground distance from the beam center via the Euclidean norm, $r = \sqrt{x^2 + y^2}$.
- Calculate the elevation angle $\epsilon = \arctan\left(\frac{satellite\_height}{r}\right)$
- Calculate the azimuth angle $\phi = \arctan\left(\frac{y}{x}\right)$, giving the horizontal orientation of the user relative to the beam center.
- Calculate the effective velocity along the line of sight by projecting the satellite's velocity vectors (v_{s_x}, v_{s_y}) onto the user's line of sight:

$$v_{eff} = (v_{s_x} \cos(\phi) + v_{s_y} \sin(\phi)) \cos(\epsilon)$$

- Compute the Doppler shift $f_d = \frac{v_{eff}}{c} \cdot f_c$

Figure 23a illustrates how Doppler frequencies can vary across the satellite beam's coverage area, with values spanning approximately ± 48 kHz depending on the user's position.



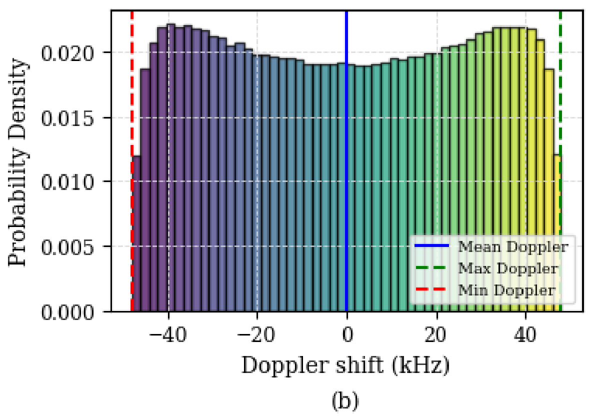


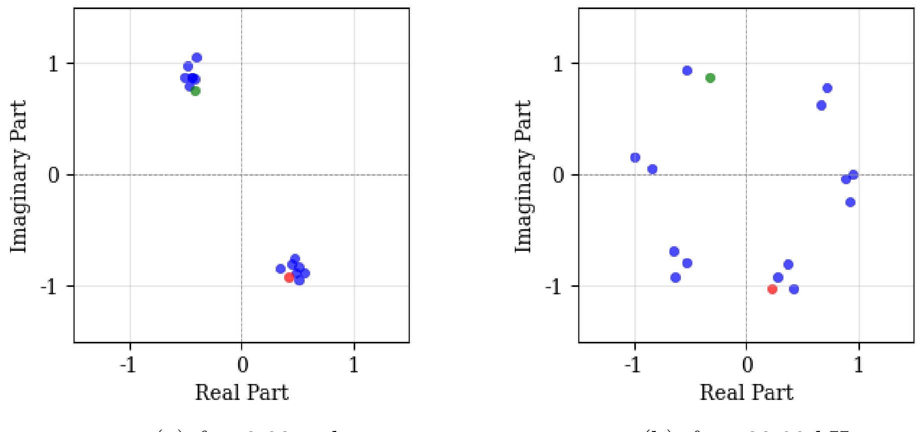
Figure 23: (a) Doppler shift along the satellite beam's coverage area and (b) probability density function of the Doppler shift.

These maximum value correspond to users at the edge of the beam and at the satellite direction. In this depicted case, the satellite is assumed to be moving towards the positive x-axis. However, varying the direction of the satellite would not have any impact due to the symmetry of the distribution of the Doppler shift given uniformly allocated users. Considering the channel bandwidth is only 100 kHz, the Doppler effect spans nearly the entire spectrum, creating significant challenges for accurate detection and synchronization. Meanwhile, Figure 23b presents a probability density function (PDF) of the observed Doppler shifts for a user spread uniformly within the coverage beam. Although the distribution is symmetric with 0 mean, it has slightly heavier tails toward the extremes, indicating that users are likely to experience more pronounced Doppler variations. This arises primarily from geometry: users near the coverage boundary experience more motion relative to the satellite's velocity vector, leading to higher Doppler shifts. Consequently, while the distribution remains centered at zero, the denser outer regions of the beam provoke these heavier tails.

Plotting various received preambles in the I-Q plane with the Doppler effect would generate graphics with similar shapes to Figure 20. Therefore, in this case it is more interesting to visualize one individual preamble for each case, as depicted in Figure 24. SNR level has been set to 20 dB to minimize the noise effect and appreciate better the differences.

When only a random phase offset is present (Figure 24a), all symbols in the same preamble are consistently aligned to the randomly chosen phase offset, with the primary distortion arising from noise. This consistency within the preamble explains why correlation-based detection remains unaffected in the single-user case, as the correlation inherently handles such constant rotations. However, the CNN struggles slightly, as it must learn to generalize across a broader range of inter-packet phase rotations.

In contrast, the Doppler shift introduces a more complex distortion (Figure 24b). Here, symbols within the same preamble can experience different orientations due to the continuous symbol rotation caused by the frequency offset. This intra-packet symbol phase inconsistency makes detection more challenging as not only does the inter-packet variability remain, but symbol-to-symbol misalignment within a single packet becomes a



(a) $\theta = 2.03$ rad (b) $f_d = 33.08$ kHz

Figure 24: Comparison between (a) random phase offset and (b) Doppler shift on a single preamble. Green and red dots indicate the first and last symbol of the preamble, respectively.

significant factor as well. Such intra-packet phase rotation can reduce or even negate correlation peaks among the preamble symbols, making a single correlator at the central frequency less effective. The CNN, on the other hand, must adapt to this added layer of complexity, attempting to learn both the broader inter-packet and intra-packet patterns, which substantially increases the learning difficulty.

5.2.1 Bank of Correlators

Since a single correlator may be inadequate for proper detection when large frequency offsets are present within the same packet (as demonstrated later in Figure 26), we are going to use a bank of correlators to address this issue and fairly compare both detection algorithms.

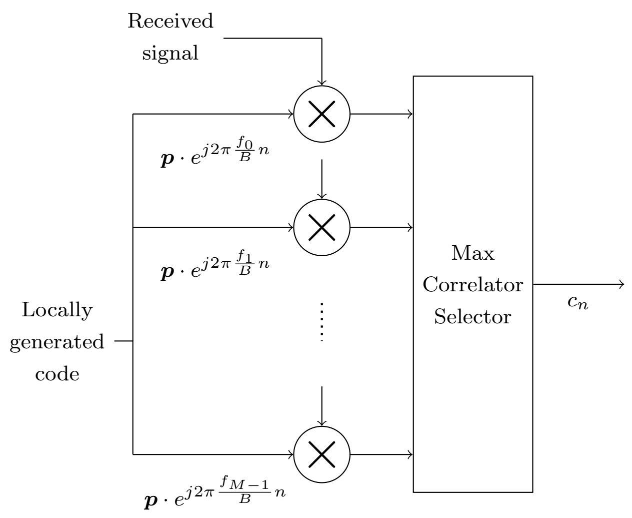

Received signal
$\boldsymbol{p} \cdot e^{j2\pi \frac{f_0}{B} n}$
$\boldsymbol{p} \cdot e^{j2\pi \frac{f_1}{B} n}$
$\boldsymbol{p} \cdot e^{j2\pi \frac{f_{M-1}}{B} n}$
Locally generated code
Max Correlator Selector
c_n

Figure 25: Block diagram of the bank of correlators used

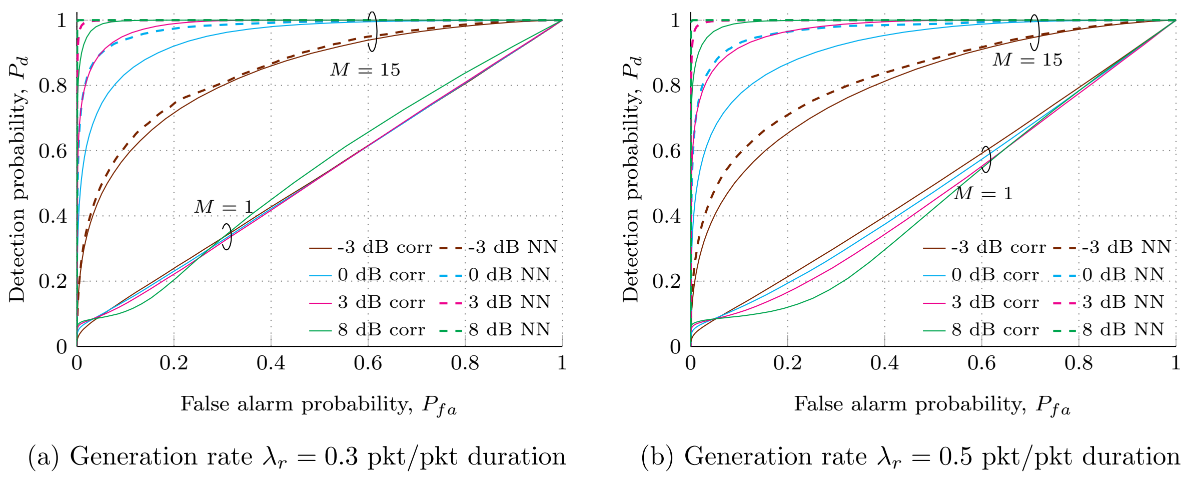
A bank of correlators, as visualized in Figure 25, is a set of multiple correlators, each tuned to detect the preamble at a different frequency offset $f_i = -f_{d,max} + i\frac{2f_{d,max}}{M-1}$, with i going from 0 to $M-1$ and being M the total number of correlators. By processing the received signal in parallel across all correlators and selecting the one with the strongest correlation peak, the system can detect the preamble with higher accuracy, and at the same time provide an estimate of the undergone Doppler shift at the expense of an additional computational complexity.

To make this concept intuitive, one can think of the correlator bank as a "tuner" scanning multiple radio frequencies. Just as a tuner must adjust to the correct frequency to receive a desired station, the bank of correlators simultaneously "listens" to all possible frequencies and selects the one that best matches the desired signal. This parallel processing ensures robust detection even in the presence of challenging Doppler effects.

A. Interference-free scenario

Once more, the single-user case will be studied first. Figure 26 shows the results of this scenario, where the faded solid curves represent the correlation performance when using just $M = 1$ correlator at reception, confirming the poor behavior of this algorithm in presence of the Doppler effect. In addition, the colorful solid lines display the performance of the correlator bank with $M = 15$ correlators. Further experiments in Section 5.4 will show the impact of varying this parameter M.

In order to obtain more complete results, random phase offset was also added to the Doppler shift. The effects of the random phase offset discussed in Section 5.1 are mostly mitigated when combined with Doppler shift, as the latter naturally introduces a phase rotation that accumulates over time. This means that even without an extra random phase offset, the phases of the symbols are already randomized by the Doppler-induced rotation. Because the phase is already non-coherent due to Doppler, adding a separate



(a) Generation rate $\lambda_r = 0.3$ pkt/pkt duration (b) Generation rate $\lambda_r = 0.5$ pkt/pkt duration

Figure 26: Detection probability as a function of the false alarm considering single-user scenario with both Doppler and random phase offset for SNR $= -3, 0, 3, 8$ dB and generation rates λ_r of (a) 0.3 and (b) 0.5.

random phase offset does not significantly further decorrelate the interfering packets. In fact, when Doppler is present, the extra random phase offset has no impact at all on the correlation results, and the CNN's performance experiences only a minor decrease.

These updated ROC curves indicate a clear degradation in both algorithms' performance for the single-user scenario. Although detection results are lower across all SNR values, 3 and 8 dB still achieve detection probabilities above 0.9 for false alarm rates under 0.1. However, at 0 dB and especially at -3 dB detection rates decrease sharply, making reliable detection increasingly challenging. For instance, maintaining a detection rate above 0.8 at -3 dB requires tolerating a false alarm rate of about 0.4 (this is an extreme operating point, given that the Doppler shift is approximately half the bandwidth and the signal level is well below the noise floor).

Table 7 reinforces these findings by comparing both algorithms at multiple false alarm thresholds. Previously, with only random phase offsets, detection rates above 0.8 were feasible at -3 dB with a false alarm probability below 0.02. Yet, once Doppler is introduced, at -3 dB and a 0.1 false alarm rate, the correlator and CNN only reach about 0.5 and 0.6 detection probability, respectively.

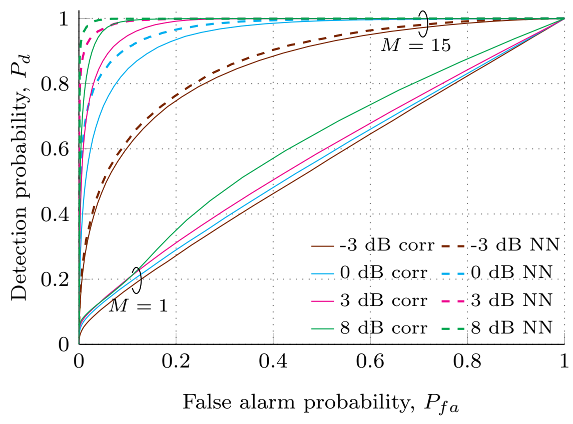
Table 7: Detection probabilities for varying false alarm probabilities for the single user scenario, Doppler, random phase offset and a generation rate of $\lambda_r = 0.5$ pkt/pkt duration.

	Correlator			**CNN**		
SNR (dB)	$P_{fa} = 0.02$	$P_{fa} = 0.05$	$P_{fa} = 0.1$	$P_{fa} = 0.02$	$P_{fa} = 0.05$	$P_{fa} = 0.1$
-3	0.2827	0.4036	0.5174	0.3546	0.4813	0.5904
0	0.5452	0.6721	0.7706	0.7672	0.8620	0.9197
3	0.7606	0.8527	0.9157	0.9895	0.9974	0.9995
8	0.9126	0.9673	0.9906	1.0	1.0	1.0

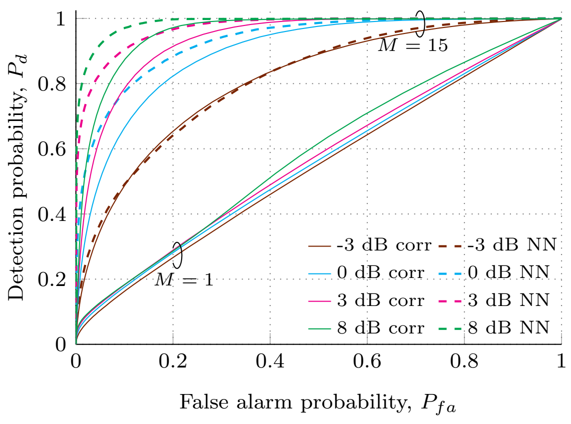
B. Interference scenario

Changing now to the multi-user scenario shown in Figure 27, it can be sensed that as the channel load rises the interference from multiple users becomes more pronounced, which visibly pushes the detection curves downward compared to the single-user case. At a channel load of $G = 0.3$, the interference is limited and both the correlator and CNN exhibit relatively high detection probabilities even at modest false-alarm rates. However, these curves degrade significantly as more users join the channel. The SNR remains a key factor in the performance. For instance, the CNN at higher SNR values like 3 or 8 dB yield detection probabilities approaching or even exceeding 0.8 when the false-alarm rate is below 0.1, despite the additional multi-user interference. At lower SNR, especially -3 dB, these ROCs bend sharply downward, requiring a much higher false-alarm probability to reach the same detection thresholds.

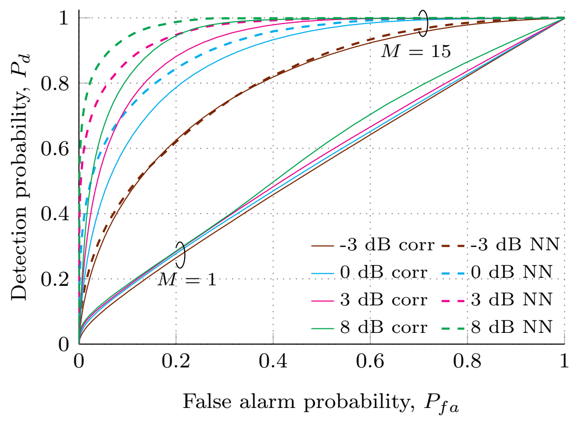
Table 8 provides discrete data points at $P_{fa} = 0.1$, confirming the trends visible in the plots. For an SNR of -3 dB and $G = 0.3$, the correlator achieves a detection probability of about 0.6039, while the CNN slightly outperforms it at 0.6263. As the channel load moves to 1.2, both algorithms see their detection rates drop by roughly 20 percentage points or more. At 0 dB, the correlator's detection probability slides from about 0.8438 at $G = 0.3$ down to 0.5298 at $G = 1.2$, whereas the CNN declines from approximately



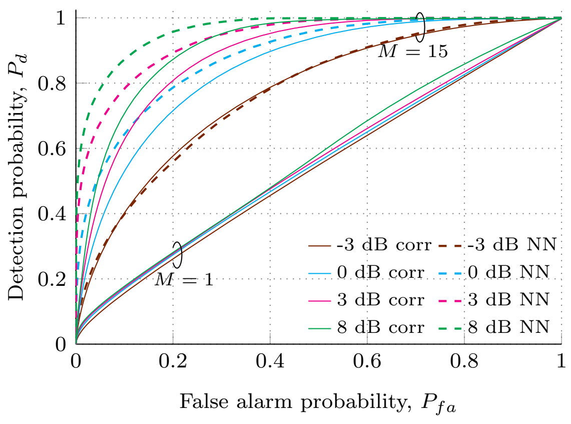
(a) Channel load $G = 0.3$ pkt/pkt duration



(b) Channel load $G = 0.75$ pkt/pkt duration



(c) Channel load $G = 0.9$ pkt/pkt duration



(d) Channel load $G = 1.2$ pkt/pkt duration

Figure 27: Detection probability as a function of the false alarm considering interference scenario with both Doppler and random phase offset for SNR $= -3, 0, 3, 8$ dB and different channel loads.

0.9106 to 0.6435 for the same change in load. Meanwhile, for 3 and 8 dB, both methods sustain stronger detection performance across all channel loads, although the correlator's detection probability dips more severely than the CNN's.

Summarizing, these results extend the single-user findings into the multi-user domain, showing that Doppler randomizes the phase sufficiently to dominate any additional static random phase offset. The CNN consistently outperforms or matches the correlator method even when multiple correlators ($M = 15$) are used in parallel. Both algorithms see a sharp drop in detection probability due to the severe interference. Therefore, in the presence of Doppler and random phase, channel load and SNR become the dominant factors influencing detection performance in the multi-user setting, and the CNN approach continues to prove more resilient to deal with interference than the correlator.

Table 8: Detection probabilities for varying channel loads and a false alarm rate of $P_{fa} = 0.1$ for the multi user scenario with Doppler and random phase offset.

	Correlator				CNN			
SNR (dB)	$G = 0.3$	$G = 0.75$	$G = 0.9$	$G = 1.2$	$G = 0.3$	$G = 0.75$	$G = 0.9$	$G = 1.2$
-3	0.6039	0.4867	0.4534	0.4020	0.6263	0.5003	0.4640	0.4012
0	0.8438	0.6644	0.6153	0.5298	0.9106	0.7725	0.7208	0.6435
3	0.9469	0.7852	0.7307	0.6270	0.9906	0.9034	0.8653	0.7860
8	0.9926	0.8726	0.8140	0.7067	0.9987	0.9710	0.9412	0.8667

5.3 Received Power Distribution

When we introduce varying received power based on user positioning, we effectively model a scenario closer to reality, where each user's signal experiences a different path loss and antenna gain within the satellite beam coverage. This variation does not introduce any additional distortion beyond basic attenuation, but it does lead to a non-uniform distribution of SNR among users. As shown in Figure 28, the probability density function of SNR now skews toward lower values, reflecting how path loss increases near the beam edges while the antennas' gains decrease due to lower elevation angles. The vertical blue line indicates the mean SNR, which is around 2 dB. Additionally, these results incorporate the same random phase offset and Doppler shift discussed earlier, so that the overall setup reflects a comprehensive and realistic channel condition.

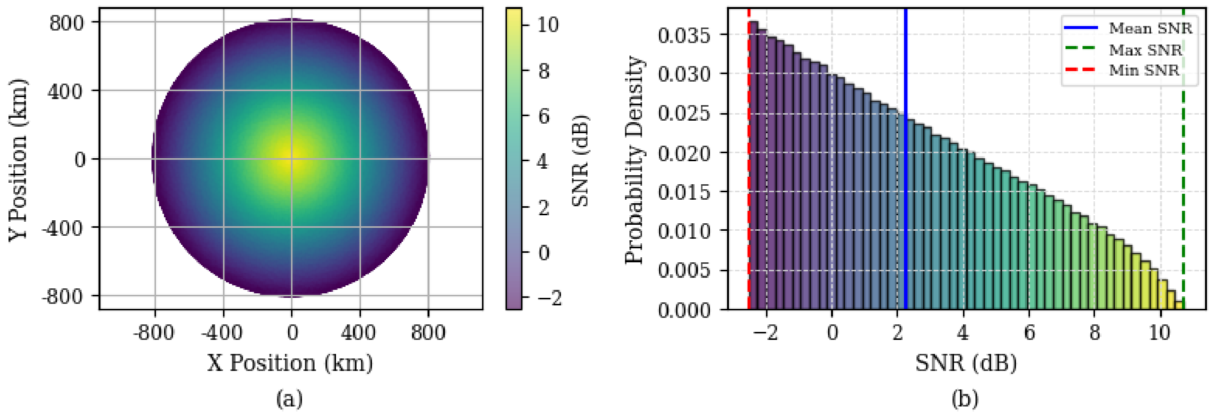


Figure 28: (a) SNR along the satellite beam's coverage area and (b) probability density function of the SNR.

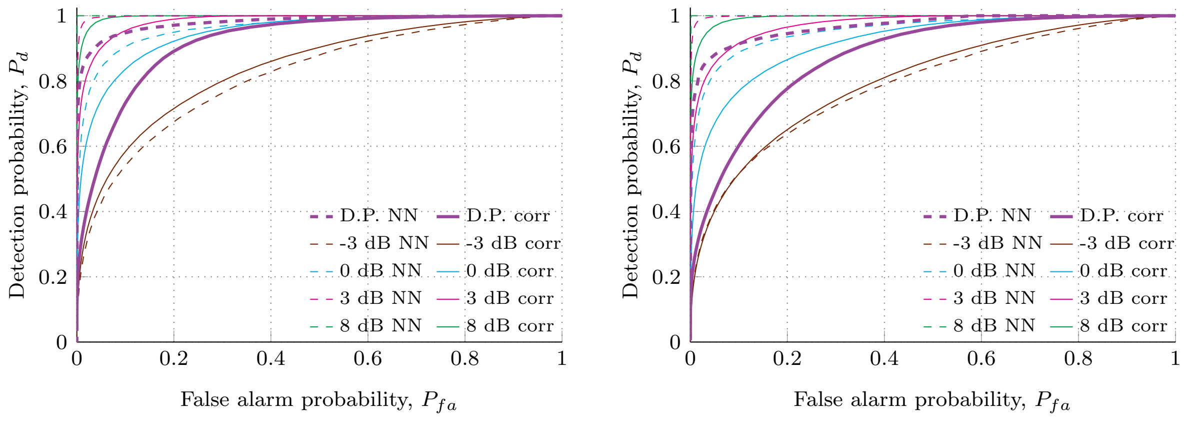
With this setup, we train new NN models to cover the full range of SNR values rather than focusing on specific SNR targets. This choice reflects real-world conditions, where the satellite's detection algorithm cannot predict whether an incoming signal comes from a high- or low-SNR user, and the final NN model must therefore handle any SNR scenario. The same procedure as for the Doppler case is followed: each newly generated packet is assigned a random position (uniformly distributed within the coverage beam), which determines the Doppler shift and corresponding SNR for that user. However, the channel occupancy remains trained ad hoc, meaning each load or generation rate still has its own dedicated model.

A. *Interference-free scenario*

Because the average SNR is about 2 dB, one could imagine that the resulting detection performance would be contained between the curves for 0 and 3 dB obtained in earlier experiments. Figure 29 compares these distributed-power results (thick purple lines) with those from the single-SNR values (thin, faded lines).

Looking first at the CNN curves, notice that at -3 dB the CNN's performance now falls below that of the correlator, unlike in the single-SNR models presented earlier. This drop arises because the new model must accommodate the entire SNR range simultaneously, rather than being trained strictly for -3 dB. Consequently, its performance at any individual SNR may not match that of an ad-hoc model specialized for a single SNR level. Still, at both generation rates, the CNN trained under received power distribution remains bounded by the 0 and 3 dB curves (cyan and magenta dashed lines), indicating it adapts across the distribution although with some trade-off in performance at specific SNR values.

By contrast, the correlator bank with $M = 15$ correlators does not behave as accurately. Its performance under path-loss variation falls between -3 and 0 dB results, instead of between 0 and 3 dB. A likely reason is that the correlator's average detection probability becomes dominated by the lower-SNR tail of the distribution rather than the mean. Each packet is processed independently, and without an adaptive mechanism to exploit higher-SNR signals, weaker users drag down the correlator's overall performance. Meanwhile, although the CNN's performance declines at -3 dB, its stronger detection rates at higher SNR levels (particularly in the low false alarm region) compensate for this drop and keep the overall distribution bounded by the 0 dB and 3 dB curves.



(a) Generation rate $\lambda_r = 0.3$ pkt/pkt duration

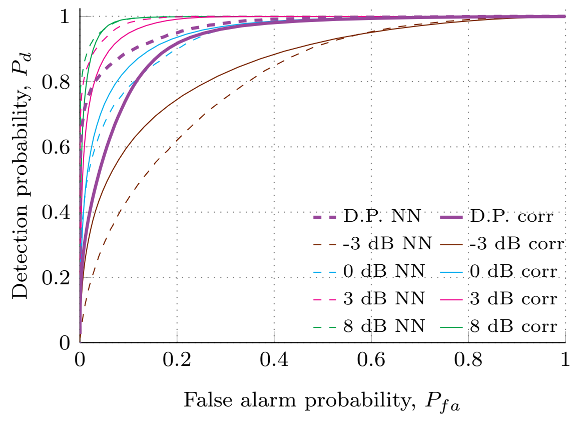
(b) Generation rate $\lambda_r = 0.5$ pkt/pkt duration

Figure 29: Detection probability as a function of the false alarm considering single-user scenario with both Doppler and random phase offset with received power distribution for and generation rates λ_r of (a) 0.3 and (b) 0.5.

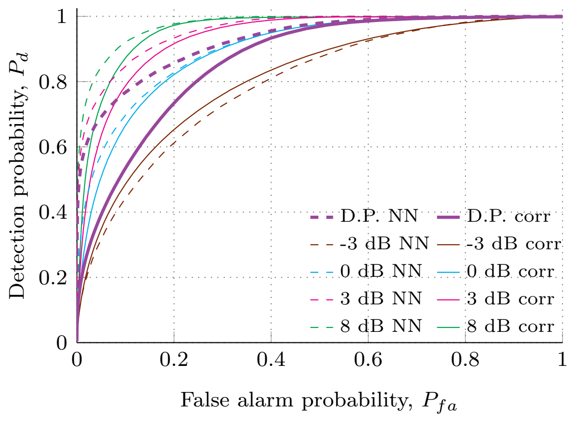
B. Interference scenario

In the multi-user scenario, the same pattern emerges as in the single-user case. Figure 30 shows that overall detection performance declines with increasing channel load under interference, as expected. The dashed purple curve of the CNN still falls between the 0 and 3 dB references, whereas the solid purple curve of the correlation remains bounded by the -3 and 0 dB lines. This aligns with the idea that the correlator's detection does not scale linearly with SNR, and packets at lower SNR drag down the mean detection probability with more strength than the smaller number of higher-SNR packets can compensate.

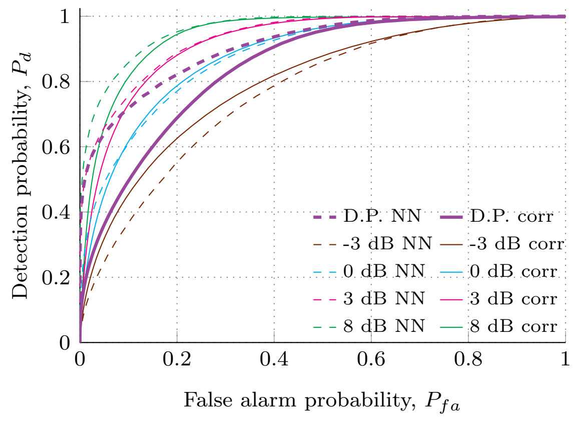
Another notable point is that the correlator experiences added difficulty in the lower-false-alarm region. Past a false alarm probability of about 0.4, both methods converge to comparable detection rates. However, at a more practical threshold of $P_{fa} = 0.1$, the CNN is able to achieve detection probabilities close to 0.7 at the highest channel load, whereas the correlator struggles to exceed 0.5 whenever the interference becomes significant (that



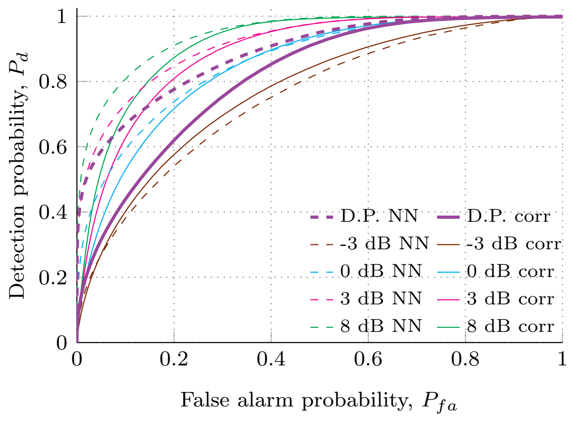
(a) Channel load $G = 0.3$ pkt/pkt duration



(b) Channel load $G = 0.75$ pkt/pkt duration



(c) Channel load $G = 0.9$ pkt/pkt duration



(d) Channel load $G = 1.2$ pkt/pkt duration

Figure 30: Detection probability as a function of the false alarm considering interference scenario with both Doppler and random phase offset with received power distribution for different channel loads.

is, for channel loads of around 0.75 or more). This difference arises because the correlator relies entirely on detecting peaks in the correlation output, which are heavily distorted under high-interference conditions due to overlapping signals and noise. With its fixed approach, the correlator finds it difficult to distinguish preambles from interference at demanding false-alarm values. In contrast, the CNN learns complex patterns from the data, enabling it to recognize partial or noisy preamble structures and leverage stronger SNR signals, even in challenging interference scenarios. This adaptive capability allows the CNN to maintain higher detection rates at $P_{fa} = 0.1$, despite the increased channel load. Beyond $P_{fa} = 0.4$, the thresholds for both methods become more permissive, reducing the performance gap.

5.4 Final Results

In addition to building models that can handle a range of SNR values, it is essential to develop models that can adapt to varying channel occupancies. In a practical system the channel load can fluctuate significantly over time as the satellite passes over regions with differing numbers of active users. For instance, densely populated areas may experience higher channel loads, leading to increased interference, while sparsely populated areas may have lighter loads with minimal interference. To ensure robust performance across these scenarios, the NN model must be trained to handle a broad spectrum of channel conditions, from low to high occupancy, rather than being optimized for a fixed load.

Although the previous section already introduced models trained with a random received power distribution, each was still related to a specific channel load. The goal here is to examine the performance of a single, more general model, representing the kind of detection algorithm a satellite might use in practice without prior knowledge of channel usage.

This generalized model uses the dataset described in Figure 31, which aggregates samples from all the generation rates and channel loads covered in the thesis, totaling about one million samples. The expectation is that this wider training scope will yield a slightly lower performance than the specialized models, which were each tailored to a single channel occupancy scenario.

A. Interference-free scenario

All the CNN results from now on are obtained with the same single generic model, trained to handle all SNR and channel occupancy scenarios. Focusing first on the interference-free scenario, Figure 32 presents the results for generation rates $\lambda_r = 0.3$ and $\lambda_r = 0.5$. The CNN's overall performance (thick, dashed, purple line) remains bounded by the 0

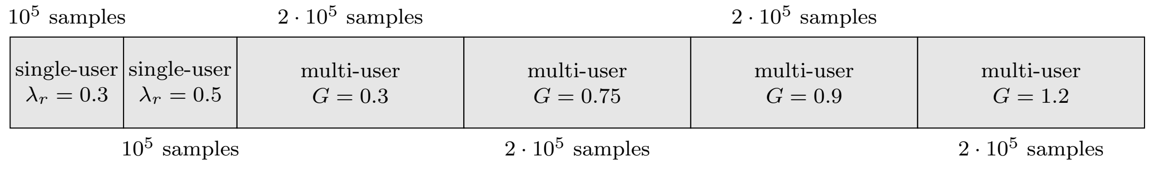
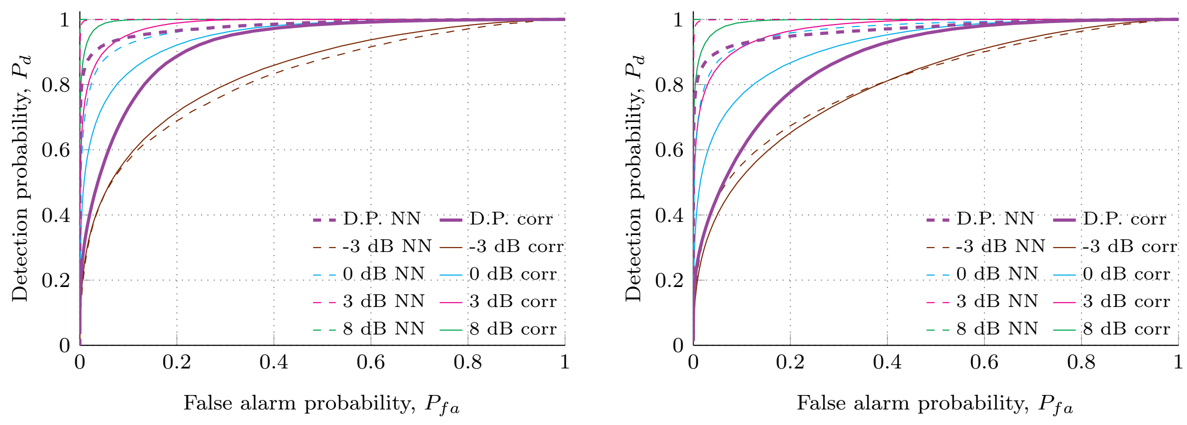


Figure 31: Dataset used to train a generic CNN able to perform under any condition.



(a) Generation rate $\lambda_r = 0.3$ pkt/pkt duration (b) Generation rate $\lambda_r = 0.5$ pkt/pkt duration

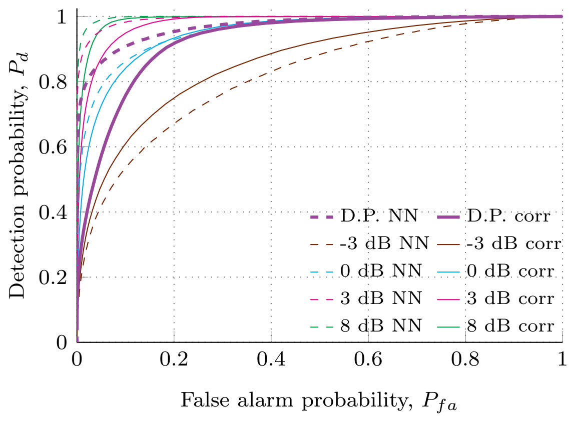
Figure 32: Detection probability as a function of the false alarm of the final model considering single-user scenario with both Doppler and random phase offset with received power distribution for and generation rates λ_r of (a) 0.3 and (b) 0.5.

dB and 3 dB curves, though it now leans slightly closer to the 0 dB cyan curve. In contrast, the correlation method's performance remains unchanged, staying below the 0 dB curve. Table 9 provides detailed values for the $\lambda_r = 0.5$ case, comparing the correlation approach and two CNN models. The "Ad-hoc CNN" columns refer to the previous neural network trained specifically for the single-user case with received power distribution and a generation rate of 0.5, with its results shown in Figure 29b. Meanwhile, the "Final CNN" represents the newly obtained model trained across various single- and multi-user scenarios.

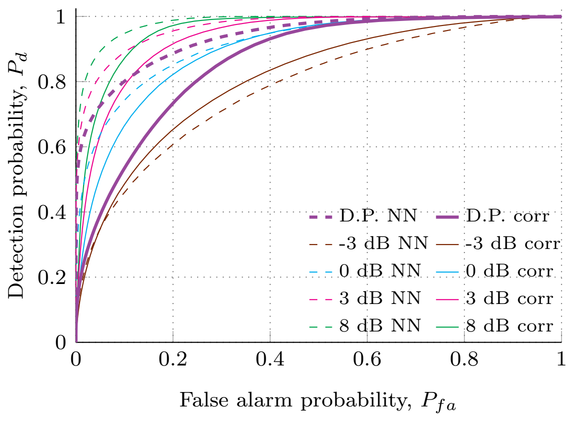
As observed, the new model even exceeds the ad-hoc model in nearly all cases, though the gap is not huge. This little improvement likely comes from the final model's exposure to additional scenarios; even though it only includes 10^5 samples specifically for the single-user, $\lambda_r = 0.5$ setup, it also benefits from training data covering various other conditions. Consequently, it can adapt more effectively and learn more robust, generalized decision boundaries. Focusing on the distributed SNR row underscores the correlator's weakness at lower false-alarm rates: for a false-alarm probability of 0.1, the correlator reaches only 0.6021 detection, whereas the final model achieves 0.9278.

Table 9: Detection probabilities for varying false alarm probabilities for the single user scenario and the final model with received power distribution, Doppler shift, random phase offset and a generation rate of $\lambda_r = 0.5$.

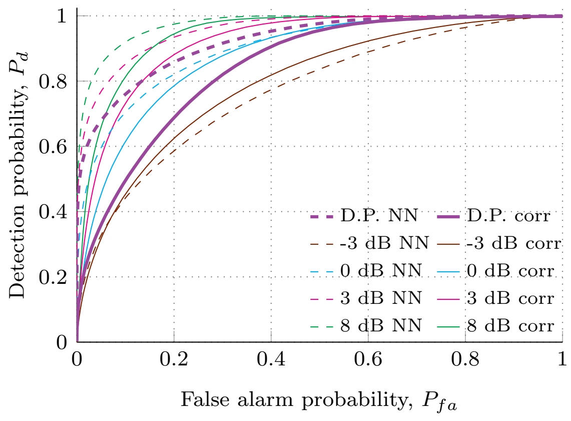
	Correlator		**Ad-hoc CNN**		**Final CNN**	
SNR (dB)	$P_{fa} = \mathbf{0.05}$	$P_{fa} = \mathbf{0.1}$	$P_{fa} = \mathbf{0.05}$	$P_{fa} = \mathbf{0.1}$	$P_{fa} = \mathbf{0.05}$	$P_{fa} = \mathbf{0.1}$
-3	0.4022	0.5155	0.4069	0.5165	0.4024	0.5569
0	0.6694	0.7691	0.8244	0.8877	0.8682	0.9220
3	0.8530	0.9155	0.9967	0.9990	0.9994	0.9999
8	0.9681	0.9910	1.0	1.0	1.0	1.0
Distributed	**0.4591**	**0.6021**	**0.8800**	**0.9136**	**0.9045**	**0.9278**



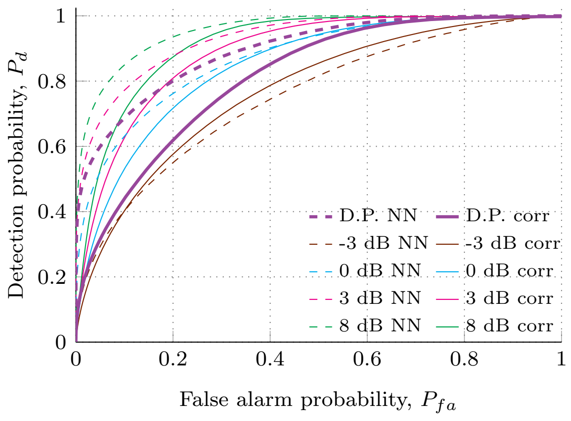
(a) Channel load $G = 0.3$ pkt/pkt duration



(b) Channel load $G = 0.75$ pkt/pkt duration



(c) Channel load $G = 0.9$ pkt/pkt duration



(d) Channel load $G = 1.2$ pkt/pkt duration

Figure 33: Detection probability as a function of the false alarm of the final model considering interference scenario with both Doppler and random phase offset with received power distribution for different channel loads.

B. *Interference scenario*

For the last time, let us study again the interference case. ROC curves for the different channel loads are plotted in Figure 33. The curves show that the CNN consistently outperforms the correlator across all channel loads. At low channel loads, the CNN achieves rather good performance, with steep curves indicating high detection probabilities above 0.8 even at low false alarm rates below 0.1 (it is important to recall the fully uncoordinated access protocol we are working with). As the channel load increases, interference impacts all methods, but the CNN maintains a clear advantage, with its curves remaining higher and closer to ideal performance compared to the flatter correlator curves.

Table 10 highlights that the Final CNN achieves superior detection probabilities across practically all SNR levels and channel loads. For a low channel load of $G = 0.3$, at -3 dB, the Final CNN reaches 0.5352, slightly below the correlator (0.6039) but above the Ad-hoc CNN (0.4421). As the SNR increases, the Final CNN shows significant improvement, reaching 0.8620 at 0 dB, 0.9825 at 3 dB, and an almost perfect 0.9989 at 8 dB,

Table 10: Detection probabilities for channel loads of 0.3 and 1.2 and a false alarm rate of $P_{fa} = 0.1$ for the multi user scenario and the final model with received power distribution, Doppler shift, random phase offset.

	Correlator		**Ad-hoc CNN**		**Final CNN**	
SNR (dB)	$G = 0.3$	$G = 1.2$	$G = 0.3$	$G = 1.2$	$G = 0.3$	$G = 1.2$
-3	0.6039	0.4020	0.4421	0.3811	0.5352	0.4022
0	0.8438	0.5298	0.7828	0.5929	0.8620	0.6311
3	0.9469	0.6270	0.9798	0.7299	0.9825	0.7719
8	0.9926	0.7067	0.9916	0.8073	0.9989	0.8494
Distributed	**0.7595**	**0.4392**	**0.8906**	**0.6684**	**0.9079**	**0.6866**

outperforming both methods.

For the highest channel load, all methods show a performance drop due to increased interference. However, at 3 and 8 dB, the Final CNN remains dominant, achieving 0.7719 and 0.8494, respectively, compared to the correlator's lower values of 0.6270 and 0.7067.

The distributed detection probabilities reinforce this trend, as the Final CNN achieves 0.9079 for $G = 0.3$ and 0.6866 for $G = 1.2$, outperforming the correlator and the Ad-hoc CNN . This demonstrates the Final CNN's robustness and consistent performance, particularly at higher SNR levels and under heavy interference conditions.

As a final step, the algorithms were evaluated with a dataset encompassing all studied combinations of SNR and channel loads, as well as single and multi-user cases, to obtain a global mean performance. This testing dataset was half the size of the training dataset described in Figure 31, but maintaining the proportions of each scenario. All the packets of the dataset were generated with the same procedure of picking a user in a random position uniformly spread in the beam, and then determining its path loss and Doppler shift, as well as adding the random phase offset. The results of this comprehensive setup are shown in Figure 34.

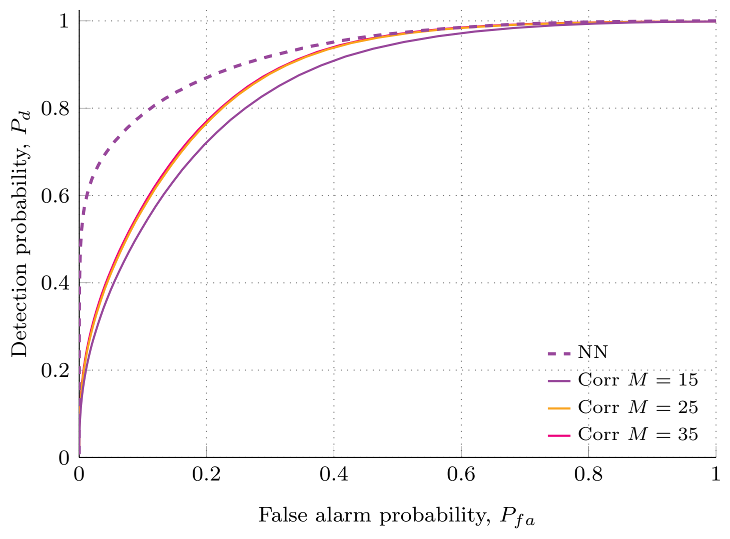
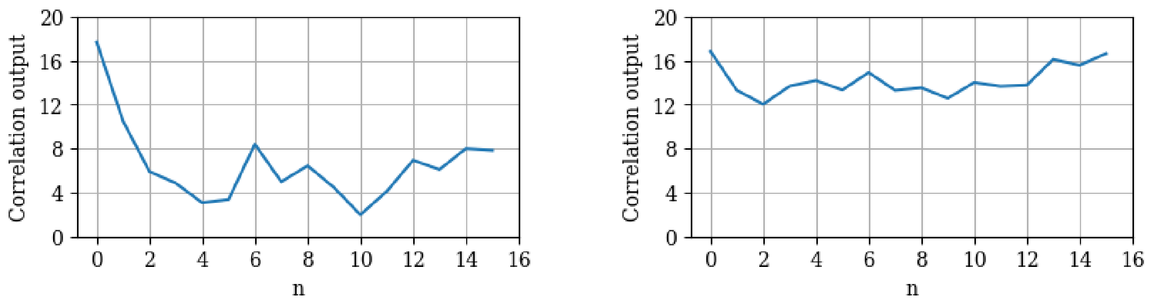


Figure 34: Performance of the detection algorithms in front of an unknown varying scenario.

Table 11: Detection probabilities and AUC metrics for the different setups of the bank of correlators.

M	P_d	AUC
15	0.5278	0.8513
25	0.5695	0.8726
35	0.576	0.8749

In this scenario, three configurations of the correlator bank were tested: $M = 15, 25, 35$. While it might seem intuitive that increasing the number of correlators would improve performance towards an optimal level, the plots indicate that even with a higher number of correlators, the performance gains are limited. In fact, Table 11 confirms this minimal gain in terms of detection rate and AUC when going from 25 to 35 correlators. Further analysis revealed that increasing the number of correlators while taking the maximum value across them at each position causes the correlator output's mean to increase. This higher mean is more likely to trigger false detections, leading to a degradation in the overall ROC curve, as shown in Figure 35. To examine this effect, a single packet with a maximum Doppler shift was placed at position 0 with an SNR of -3 dB. Detection was performed using a single perfectly matched correlator at that Doppler shift (Figure 35a) and with a bank of $M = 35$ correlators (Figure 35b). As illustrated in the figures, the final correlation output from the bank shows higher values, making it challenging to set an appropriate decision threshold. Consequently, the trade-off between detection and false alarms deteriorates notably.



(a) (b)

Figure 35: Correlation comparison between (a) a perfectly Doppler-matched single correlator and (b) a bank of $M = 35$ correlators and selecting the maximum correlator at each position.

The results presented in this chapter highlight the challenges and trade-offs in designing robust detection algorithms for satellite communication scenarios. While increasing the number of correlators in the detection bank allows for improved Doppler resolution, our analysis reveals that this approach has a performance ceiling. Beyond a certain point, adding more correlators does not continuously enhance detection performance, as it cannot completely mitigate the effects of Doppler shift.

On the other hand, the proposed generic CNN model demonstrates significant adaptability and robustness across a wide range of SNR levels, channel loads, and interference scenarios. By training on a diverse dataset covering both single-user and multi-user cases, the model achieves superior detection probabilities compared to traditional correlation-based methods. Notably, the final CNN excels in scenarios with higher interference and varying conditions, achieving a more balanced trade-off between detection and false alarms.

6 Conclusions and future development

In this thesis we demonstrated the effectiveness of supervised learning techniques for short packet detection in satellite communications. Beginning by reproducing previous results in the literature [1], we saw that in an impairment-free channel affected only by AWGN noise, both the traditional correlation-based method and the CNN achieve high detection rates, exceeding 0.9 for false alarms below 0.05 in the single-user case. However, the multi-user case (where packet collisions occur) shows a performance decline due to reduced SINR. Despite this, CNN results consistently outperforme traditional correlation. Additionally, the correlation algorithm exhibits higher sensitivity to channel load, while the CNN maintains robust performance as channel load increases.

The analysis of channel impairments began with the impact of a random constant phase shift. In the single-user scenario, correlation remains unaffected due to its reliance on the magnitude of the correlation output, whereas the CNN experiences slight degradation due to the broader distribution of signal orientations in the I-Q plane. In the multi-user case, correlation benefits from the random phase shift as it introduced phase diversity and non-coherent interference, with the improvement more pronounced at higher channel loads. While the CNN struggles under low SNRs in this scenario, it leverages random phase diversity at higher SNRs, maintaining a slight performance advantage overall.

The introduction of Doppler shift neutralizes the effects of the random phase shift, as Doppler inherently randomizes symbol phases. Significant Doppler shifts causes intra-packet symbol misalignment, making a single correlator ineffective and requiring the use of a bank of correlators. Comparing the CNN with a 15-correlator bank, similar performance is observed at -3 dB SNR in both single- and multi-user scenarios. However, at higher SNRs, the CNN significantly outperformes the correlation approach, especially in the low-SNR region. Under the maximum channel load ($G = 1.2$) and tolerating a false alarm rate of 0.1, both algorithms obtain a detection rate of 0.4 at -3 dB, while at 8 dB detection rates of 0.7067 and 0.8667 are observed for correlation and CNN, respectively.

Adding received power distribution among users introduces attenuation effects, leading to a non-uniform SNR distribution. This causes the correlation method to perform worse in low-SNR regions with many low-SNR users. Conversely, the CNN effectively compensates by leveraging high-SNR frames, maintaining its advantage in both single- and multi-user cases. The most significant differences between the two methods are observed again in the low-SNR regions of the ROC plots.

When considering a completely unknown scenario with varying channel loads and using a definitive CNN trained on diverse channel conditions, the CNN demonstrates superior adaptability, achieving a detection rate of 0.8 compared to 0.5 for correlation when accepting a false alarm rate of 0.1. Increasing the number of correlators in the bank does not effectively mitigate the Doppler effect, as the selection of maximum correlation outputs introduces larger correlation side lobes, degrading performance in low-SNR regions and distorting the ROC curve. However, the bank of correlators offers the advantage of providing an estimate of the Doppler shift for detected packets.

Future work could focus on designing neural networks capable of not only detecting packets but also estimating the Doppler frequency of detected frames, potentially mimicking the mentioned advantage of the correlator bank. This approach could integrate Doppler estimation into the detection process, improving efficiency and reducing computational overhead. Moreover, extending the dataset to include more diverse scenarios, such as additional channel impairments or non-symbol-synchronous scenarios, would enhance the generalizability of the proposed methods. Finally, exploring hybrid approaches that combine neural networks with traditional methods could further improve detection performance. Relevant future development also involves a comparison with reference benchmarks that go beyond the correlation-based approach.

References

[1] E. Recayte, A. Munari, and F. Clazzer. Grant-free access: Machine learning for detection of short packets. *Proc. 10th Adv. Satell. Multimedia Syst. Conf, pp. 1–7*, Graz, Austria, Oct. 2020.

[2] N. Abramson. The aloha system: another alternative for computer communications. *Fall Joint Computer Conference, vol. 37, pp. 281–285*, 01 1977.

[3] M. Liess, F. Lazaro, and A. Munari. Frame synchronization algorithms for satellite internet of things scenarios. *11th Advanced Satellite Multimedia Systems Conference and the 17th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, Graz, Austria, Sep. 2022.

[4] 3GPP. Study on narrow-band internet of things (NB-IoT) /enhanced machine type communication (eMTC) support for non-terrestrial networks (NTN). *3rd Generation Partnership Project (3GPP), Technical Report (TR) 36.763, version 17.0.0. [Online]*, 06 2021. Available: `https://www.3gpp.org/ftp/Specs/archive/36_series/36.763/`.

[5] T. Pratt, C. Bostian, and J. Allnutt. *Satellite Communications.* 2nd ed. John Wiley & Sons, Inc., 2003.

[6] J. G. Proakis and M. Salehi. *Digital Communications.* 5th ed. John Wiley & Sons, Inc., 2007.

[7] H. Meyr, M. Moeneclaey, and S. Fechtel. *Digital Communication Receivers: Synchronization, Channel Estimation, and Signal Processing.* John Wiley & Sons, Inc., 1997.

[8] R. De Gaudenzi, O. del Rio Herrero, C. R. da Silva, and G. Gallinaro. Random access schemes for satellite networks, from VSAT to M2M: A survey. *International Journal of Satellite Communications and Networking, vol. 36, no. 1, pp. 66–107*, 2018.

[9] J. Gansman, M. Fitz, and J. Krogmeier. Optimum and suboptimum frame synchronization for pilot-symbol-assisted modulation. *IEEE Transactions on Communications, vol. 45, no. 10, pp. 1327–1337*, 1997.

[10] Z. Y. Choi and Y. Lee. Frame synchronization in the presence of frequency offset. *IEEE Transactions on Communications, vol. 50, no. 7, pp. 1062–1065*, 2002.

[11] R. Wuerll, J. Robert, G. Kilian, , and G. Heuberger. Optimal one-shot detection of preambles with frequency offset. *IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), pp. 1-4*, 2018.

[12] M. Chiani. Noncoherent frame synchronization. *IEEE Transactions on Communications, vol. 58, no. 5, pp. 1536–1545*, 2010.

[13] T. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[14] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press, 2014.

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

[16] T. J. O'Shea, J. Corgan, and C. Clancy. Convolutional radio modulation recognition networks. *International Conference on Engineering Applications of Neural Networks, pp. 213–226*, 2016.

[17] S. Haykin. *Neural Networks and Learning Machines.* 3rd ed., Pearson, 2008.

[18] A. Vaswani, N. Shazeer, J. Uszkoreit N. Parmar, L. Jones, A. N. Gomez, and I. Polosukhin L. Kaiser. Attention is all you need. *Advances in Neural Information Processing Systems, vol. 30*, 2017.

[19] M. Soltani, V. Pourahmadi, A. Mirzaei, , and H. Sheikhzadeh. Deep learning-based channel estimation. *IEEE Commun. Lett., vol. 23, no. 4, pp. 652–655*, Apr. 2019.

[20] K. Mei, J. Liu, X. Zhang, N. Rajatheva, and J. Wei. Performance analysis on machine learning-based channel estimation. *IEEE Transactions on Communications, vol. 69, no. 8, pp. 5183-5193*, Aug. 2021.

[21] D. Zhou, M. Sheng, Y. Wang, J. Li, , and Z. Han. Machine learning-based resource allocation in satellite networks supporting internet of remote things. *IEEE Trans. Wireless Commun., vol. 20, no. 10, pp. 6606–6621*, Oct. 2021.

[22] M. A. Vazquez, P. Henarejos, I. Pappalardo, E. Grechi, and R. M. Lancellotti. Machine learning for satellite communications operations. *IEEE Communications Magazine, vol. 59, no. 2, pp. 22–27*, Feb. 2021.

[23] G. Cocco, T. de Cola, M. Angelone, Z. Katona, and S. Erl. Radio resource management optimization of flexible satellite payloads for dvb-s2 systems. *IEEE Transactions on Broadcasting, vol. 64, no. 2, pp. 266–280*, June 2018.

[24] M. U. Khan, E. Testi, E. Paolini, , and M. Chiani. Preamble detection in asynchronous random access using deep learning. *IEEE Wireless Commun. Lett., vol. 13, no. 2, pp. 279–283*, Feb. 2024.

[25] J. Mohammadi, G. Schreiber, T. Wild, and Y. Chen. Blind coherent preamble detection via neural networks. *25th International ITG Workshop on Smart Antennas*, 2021.

[26] D. H. Yang and S. W. Choi. On the structured design for efficient machine learning based prach preamble detection. *14th International Conference on Information and Communication Technology Convergence (ICTC), pp. 1017-1021*, 2023.

[27] H. Sun, A. O. Kaya, M. Macdonald, H. Viswanathan, and M. Hong. Deep learning based preamble detection and toa estimation. *Proc. of 2019 IEEE Global Communications Conference (GLOBECOM), pp 1-6*, Waikoloa, HI, USA, Dec. 2019.

[28] J. H. I. De Souza and T. Abrao. Deep learning-based activity detection for grant-free random access. *IEEE Syst. J., vol. 17, no. 1, pp. 940–951*, 2023.

[29] CCSDS. *TC synchronization and channel coding - summary of concept and rationale.* NASA Headquarters, Washington, DC 20546-0001, USA, issue 2 ed., 2012.

[30] G.D. Gordon and W.L. Morgan. *Principles of communication satellites.* John Wiley & Sons, Inc., 1993.

[31] Harry L. Van Trees. *Detection, Estmation and Modulation Theory.* John Wiley & Sons, Inc., 2001.

[32] Kendall E. Atkinson. *An Introduction to Numerical Analysis.* John Wiley & Sons, Inc., New York, 2 ed., 1989.

Appendices

A Satellite geometry calculations

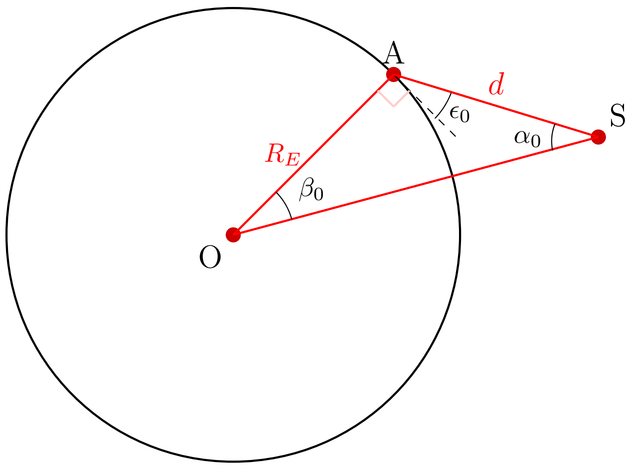


Figure 36: Satellite geometry.

The basic geometry between a satellite and ground station or user is depicted in Figure 36. Two sides of the red triangle are usually known (the distance from the user to the Earth's center, $R_E = 6378$ km, and the distance from the satellite to the Earth's center, taking into account the satellite height). There are four variables in this triangle: the elevation angle ϵ_0, the nadir angle α_0, the central angle β_0 and the slant range d. As soon as two quantities are known, the others can be found with the following equations:

$$\epsilon_0 + \alpha_0 + \beta_0 = 90$$

$$d \cos \epsilon_0 = r \sin \beta_0$$

$$d \sin \alpha_0 = R_E \sin \beta_0$$

Applying cosines law for the triangle in the figure, we can reach an expression of the slant range as a function of elevation angle ϵ_0.

$$d(\epsilon_0) = R_E \left[\sqrt{\left(\frac{h + R_E}{R_E} \right)^2 - \cos^2 \epsilon_0} - \sin \epsilon_0 \right],$$

where h is the satellite height above Earth's surface. Since the maximum slant range in our scenario is known, which is 1000 km, the elevation angle can be calculated as well as all the other parameters. In Table 12 a comparison of the values obtained with and without Earth's curvature is provided.

Table 12: Comparison flat Earth model vs curved Earth model.

Parameter	**Curved Earth**	**Flat Earth**
ϵ_0	33.35°	36.87°
α_0	49.77°	53.13°
Coverage area's radius	766 km	800 km