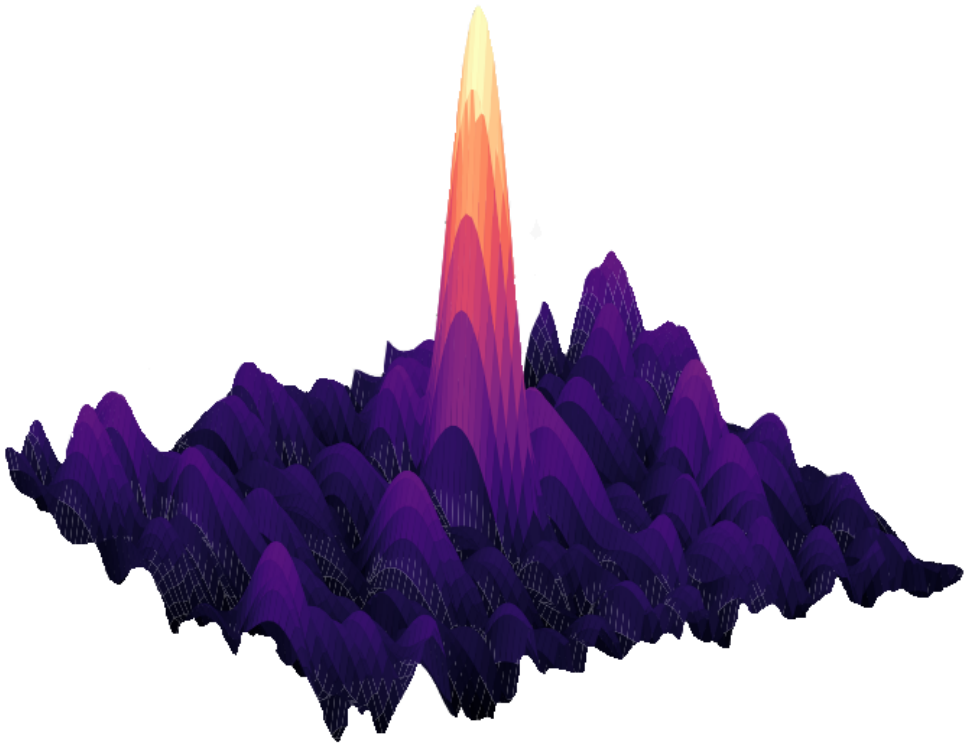


Detection of Ongoing GPS Spoofing Using A Convolutional Neural Network

A Dual-Branch Analysis of Global and Local Cross-Ambiguity Function Maps

Master's thesis in Master Data Science & AI

BENJAMIN PETTERSSON

DEPARTMENT OF PHYICS AND ASTRONOMY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Detection of Ongoing GPS Spoofing Using A Convolutional Neural Network

A Dual-Branch Analysis of Global and Local Cross-Ambiguity Function Maps

BENJAMIN PETTERSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics And Astronomy
Division of Onsala Space Observatory
Chalmers University of Technology
Gothenburg, Sweden 2026

Detection of Ongoing GPS Spoofing Using A Convolutional Neural Network
A Dual-Branch Analysis of Global and Local Cross-Ambiguity Function Maps
BENJAMIN PETTERSSON

Supervisor: Gabriel Gitye.
Examiner: Rüdiger Haas, Department of Physics And Astronomy.

Master's Thesis 2026
Department of Physics And Astronomy
Division of Onsala Space Observatory
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A scenario involving a spoofed GPS signal, showing an acquisition search result with only one correlation peak visible. This demonstrates the complexity of spoofed GPS signals, which do not necessarily need to present two peaks.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Detection of Ongoing GPS Spoofing Using A Convolutional Neural Network
A Dual-Branch Analysis of Global and Local Cross-Ambiguity Function Maps

BENJAMIN PETTERSSON

Department of Physics And Astronomy
Chalmers University of Technology

Abstract

Deliberate GPS interference by broadcasting fake signals, so-called spoofing, can compromise critical infrastructure that relies on location and timing services. This work develops a dual-branch convolutional network (CNN) that automatically detects spoofing attacks during the acquisition stage of a GPS receiver. The receiver's 2-D acquisition map (a single-channel image) is processed both locally (region-of-interest, ROI) and globally to capture spoofing cues. The CNN was trained and evaluated on MATLAB-simulated recordings as well as on two publicly available datasets, the Oak Ridge Spoofing and Interference Test Battery (OAKBAT) and the Finnish Geospatial Research Institute (FGI) repository, which together comprise 11 GPS recordings containing spoofing scenarios. Results show reliable detection when training and testing on data from the same domain, achieving balanced-accuracy scores of 99 % on targeted scenarios. Cross-domain generalization to datasets with different parameters decreases noticeably, with performance dropping to near-random-guessing levels. The study also provides visual illustrations of authentic and spoofed scenarios. The total average inference time of the solution is 22-23 ms per tracked satellite, indicating practical applicability. These findings suggest that a dual-branch CNN operating in the acquisition stage is a viable method for spoofing detection.

Keywords: GPS, spoofing, detection, ongoing, convolutional neural network (CNN), acquisition, Region-Of-Interest (ROI), Cross-Ambiguity Function (CAF).

Acknowledgements

Working with this thesis has been fun and exciting. I have treated it as a long-term project, and thus, many parts have been worked on in parallel, allowing me a balanced work-life environment throughout. I thank my supervisor, Gabriel, for serving as my stakeholder and providing guidance. My examiner, Rüdiger, for meeting me and discussing progress. And my partner, Lovisa, for supporting me.

Benjamin Pettersson, Gothenburg, May 2026

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AWGN	Additive White Gaussian Noise
bps	Bits Per Second
BPSK	Binary Phase Shift Keying
CAF	Cross-Ambiguity Function
CNN	Convolutional Neural Network
CPU	Central Processing Unit
C/A	Coarse/Acquisition
FFT	Fast Fourier Transformation
FGI	Finnish Geospatial Research Institute
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GPU	Graphics Processing Unit
MCC	Matthews Correlation Coefficient
MEO	Medium Earth Orbit
NN	Neural Network
OAKBAT	Oak Ridge Spoofing and Interference Test Battery
PRN	Pseudo-Random Noise
RAM	Random Access Memory
ROI	Region Of Interest

Contents

List of Figures

List of Tables

1

Introduction

Global navigation is available to anyone through electronic devices, with the Global Positioning System (GPS) being the most commonly used. This system is developed and funded by the United States government. GPS is, however, one of several Global Navigation Satellite Systems (GNSS) available, whereas some alternative GNSS include: the European Galileo, the Russian GLONASS, and the Chinese BeiDou [1]. The systems function in distinct technical ways, but all systems support navigation and time synchronization by sending radio wave signals from satellites in medium earth orbit (MEO), an Earth-centred orbit that includes altitudes ranging from 2,000 km above sea level to 35,786 km [2]. Broadcasting of radio waves from MEO can result in weak signals when they reach Earth to be picked up by devices capable of acquiring and tracking the signal. In fact, GPS signals at ground level are found below the thermal noise floor, random electronic noise produced by the agitation of electrons at temperatures of more than 0 Kelvin (the point of absolute zero) [2, 3]. Figure 1.1 illustrates a spectrum plot of a GPS signal's power across frequencies in comparison to thermal noise. The key takeaway from the figure is that the GPS signal is distributed across different frequencies with varying signal strengths (power), and it is weaker than the surrounding random electronic noise.

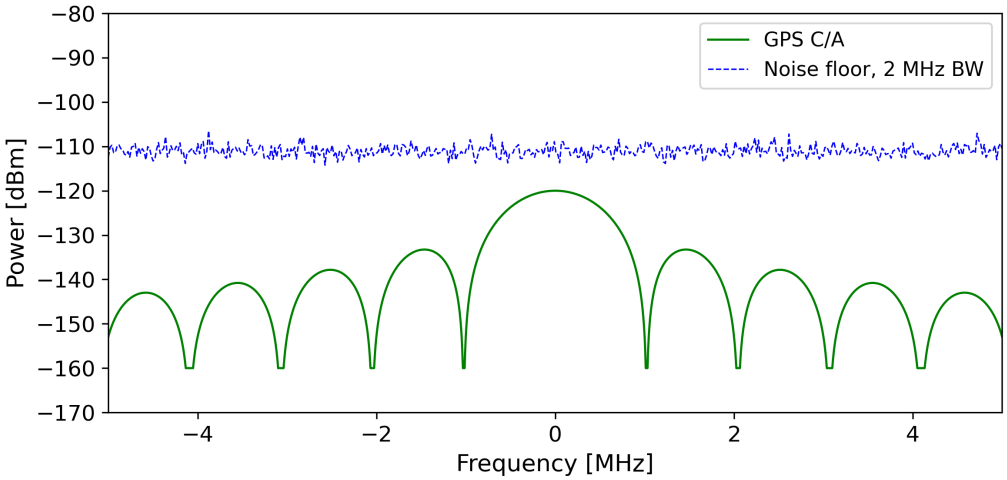


Figure 1.1: Conceptual frequency and power-Domain spectrogram illustration of GPS L1 signals. The illustration, not a measurement of an actual signal, highlights the low signal power and therefore the vulnerability of GPS to interference and spoofing.

The low strength of GPS signals at ground level makes them vulnerable to both

deliberate and accidental interference [4]. Accidental interference can arise from a range of sources, including physical obstructions and vegetation, electronic devices that disturb the GPS signal frequencies, and extreme natural events such as solar activity. Navigation signals can also be reflected off different surfaces, leading to multipath interference, in which a single signal reaches the receiver via several routes and causes the receiver to interpret it as multiple distinct signals [2]. However, not only accidental interference may impair a signal, resulting in loss of global navigation.

Deliberate interference is achieved in two main ways: jamming or spoofing [5]. Jamming of signals consists of overpowering a signal one aims to disrupt, which could be a GPS signal. The method relies on generating a structured or unstructured noise pattern in the direction of a receiver, or generally in a whole environment. The objective of jamming is to prevent any receiver chain mechanism from acquiring or maintaining a lock on the authentic satellite-broadcast signal by saturating the radio-frequency environment with interference [5]. Jamming is intended to disrupt navigation functionality entirely and thus is a blunt tool. A more sophisticated technique is GNSS spoofing, in which the attacker transmits counterfeit signals that are engineered to mimic legitimate ones, thereby tricking the receiver into computing faulty positions, velocity, or timing information [5]. Counterfeit GNSS signals overpower the authentic signals being sent from satellite in orbit by exploiting weaknesses in the GNSS receiver chain. Generally, the power of spoofed signals is stronger than the power of authentic signals, and this causes a GNSS receiver to lock onto fake signals instead of authentic ones, which disrupts navigation.

There are several indications of GPS spoofing affecting everyday life. Reports indicate that ambulance personnel in southern Sweden encounter GPS interference daily [6], where positioning often is unreliable, thereby affecting crucial infrastructure for society. Other reports indicate that boats around the coast of the Swedish island of Gotland have encountered GPS interference that placed the GPS position of the boat outside of Kaliningrad, Russia [7]. This type of interference also indicates a special type of GPS interference called meaconing, where GPS signals are recorded and rebroadcast [8]. There are also extensive reports of real jamming and spoofing occurrences, especially in the Baltic Sea [9].

The objective of this thesis is to investigate an approach for detecting intentional interference of GPS signals and to discuss potential mitigation strategies. Specifically, the work aims to develop and evaluate a methodology for detecting spoofing attacks and to determine an appropriate processing stage within a GPS receiver at which detection is suitable. To allow for this, the thesis first surveys related work on GPS spoofing.

1.1 Related Work

Research on GPS spoofing detection has expanded considerably over the past 10 years. Conventional, non-machine learning methods dominated early work, as il-

lustrated in Figure 1.2. Although machine learning approaches were initially few, their popularity has since passed that of traditional techniques, a trend confirmed by Google Scholar search for "GPS spoofing detection" with and without the keyword "machine learning".

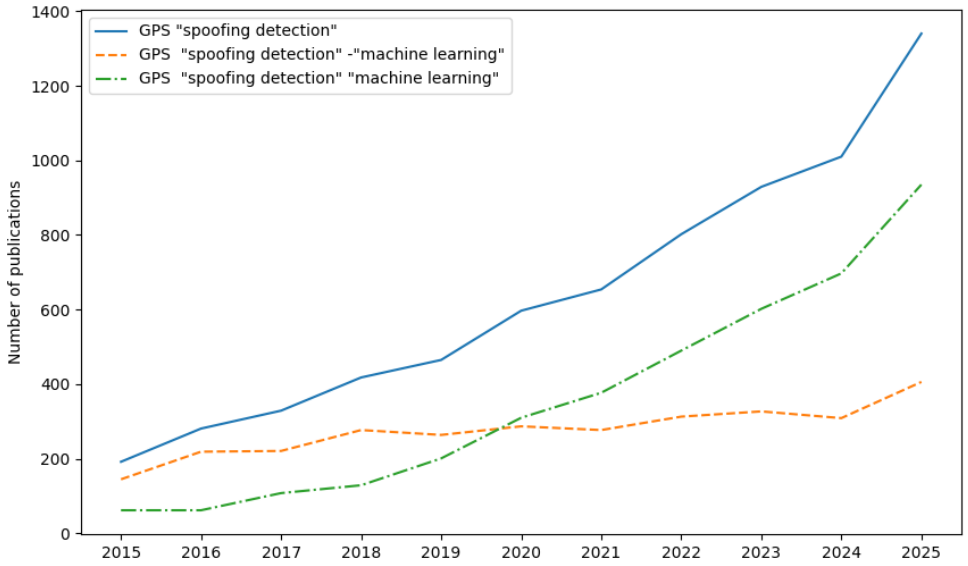


Figure 1.2: A decade trend of GPS spoofing detection publications. Number of publications (search results on Google Scholar) per year for each given search term shown in the legend. Note that quotation marks in the search term imply that the exact word phrasing order is used in the search. The box "include citation" was unchecked.

Machine learning approaches to GPS spoofing detection are not homogeneous since a wide range of techniques has been applied to various stages of the GPS receiver chain [10]. Recent literature reports hundreds of new publications each year using methods such as Support Vector Machines, Neural Networks, Generative Adversarial Networks, Long-Short-Term memory, XGBoost, and others [10, 11, 12]. Authors often quote high accuracies, indicating that their models correctly distinguish authentic from spoofed signals. However, these results are typically obtained on the authors' own simulated or hardware-recorded datasets [13, 14, 15, 16], which makes reproducibility harder and does not show the method's generalization capability. Only a few studies evaluate their approaches on publicly available datasets [17, 18], partly cause the review by [10] identified merely two major public datasets at the time. Since then, additional repositories such as [19] have become accessible, providing a broader base for assessment of the generalization capabilities of methods.

As mentioned, the literature contains a multitude of machine learning methodologies applied to diverse datasets. As highlighted by the aforementioned review [10],

state-of-the-art accuracies are frequently reported, yet reliance on a single metric (accuracy) can be misleading, especially in the presence of class imbalances as seen in [12, 18].

One line of research has investigated the application of Convolutional Neural Networks (CNNs) in the initial signal processing stages of GPS receivers. CNNs excel with grid-structured data, such as images, but GPS receiver data are not naturally image-like. Nonetheless, a few studies have explored CNNs for spoofing detection [13, 16, 20, 21]. Only two of these focus on the early acquisition stage of a GPS receiver [16, 21]. Direct application of a CNN to acquisition-stage data, therefore, remains a relatively novel problem, where unique solutions can be explored.

1.2 Purpose

The purpose of this work is to explore an approach to GPS spoofing detection during the acquisition stage of a GPS receiver. Feeding a CNN with 2-D correlation search space images at the acquisition stage could enable early and continuous spoofing detection. Detecting GPS spoofing could allow for mitigation of the spoofing, thereby improving resilience for everyday users of GPS devices.

A secondary objective is to assess the generalizability of the proposed method to previously unseen data, examining cross-domain performance when the model is initially trained on simulated data and subsequently evaluated against publicly available real-world GPS recordings. Additionally, the practical feasibility of the approach is considered, with attention to inference time.

1.3 Scope

This work focuses exclusively on GPS signals. The GPS receiver is assumed to be stationary at ground level. The methodology considers continuous monitoring of GPS signals and assumes a cold start scenario in which the receiver has no prior knowledge of the GPS constellation, position, or internal almanacs of satellite vehicles. Furthermore, the receiver is assumed to operate on a single frequency band, L1. The problem of multipath interference is not considered. Finally, the scope focuses on detecting spoofing attacks in which the attacker aims to mimic the signals of GPS satellites currently visible, in other words, the current satellite constellation.

1.4 Structure

The thesis begins by laying a theoretical background for the concepts to be discussed throughout the thesis, including the GPS signal structure, the functioning of a GPS receiver, the non-homogeneity of GNSS spoofing, and the foundations of CNNs. The methodology is then presented, describing the simulation of GPS signals in MATLAB, the publicly available repositories of GPS signal data, the CNN architecture,

and the evaluation metrics considered. Thereafter, results are presented, illustrating several acquisition search results, the generalization capabilities of the method, and inference times. Finally, aspects such as the limitations of generalization, what the CNN looks at, and the potential for spoofing mitigation are discussed, followed by conclusions regarding the feasibility of the proposed approach.

2
A Theoretical Background

This theoretical background is intended to establish a foundation for understanding GPS fundamentals, the concept of spoofing, and to offer an introductory overview of machine learning (ML), in particular, CNNs.

2.1 GPS

The GPS satellite navigation system consists of 24 nominal satellite navigation vehicles in MEO necessary for full global coverage, but there are also additional satellite vehicles intended for redundancy, and other alternative constellations [2]. All satellite vehicles generate and broadcast GPS signals, used for acquisition and tracking in receivers.

2.1.1 The GPS Signal

GPS satellites broadcast both civilian and military radio signals in the L-band frequency, ranging between 1-2 GHz [2]. The main signal broadcast for civilian use is the GPS L1 signal that consists of the carrier wave, Coarse/Acquisition (C/A) code, navigation messages, and also a precision code P(Y) with military use. An overview of the signal components is illustrated in Figure 2.1.

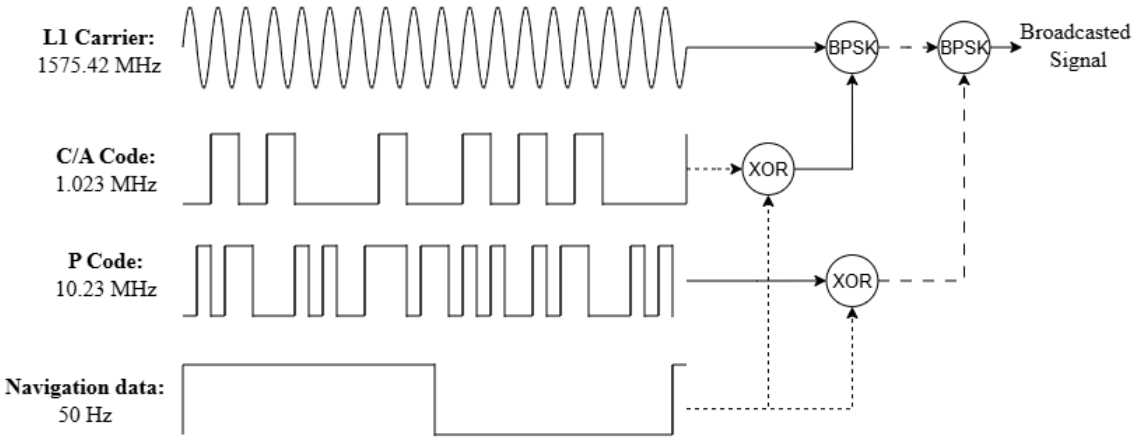


Figure 2.1: GPS L1 signal components. Illustrating how navigation data, C/A code, and P(Y) code is modulated onto the carrier wave.

The C/A code is a bit sequence of 1023 bits that repeats every 1 ms, corresponding to a rate of 1.023 Mbit per second (bps). The code is a pseudo-random noise (PRN) sequence, which resembles random noise but follows a deterministic pattern that can be generated. Each satellite vehicle transmits its own unique PRN, and because these PRNs are publicly available, they can be reproduced locally within GPS receivers [2]. A feature of the PRN for each satellite ID is that cross-correlation between code-IDs is small to non-existent. The purpose of the code is to *acquire* satellite vehicles that are broadcasting signals in view of the receiver, and also for enabling tracking. However, the C/A code alone is not sufficient for tracking since a GPS receiver must also know the positions of GPS satellite vehicles.

The navigation message data is a 50 bps stream that is likewise transmitted by satellite vehicles. This comparatively low data rate includes information required for positioning, such as the ephemeris, which describes the precise satellite orbits, the almanac, which provides coarse orbital parameters, as well as satellite health information [22]. Without delving into the detailed format of the navigation message, it provides the information required to determine the absolute position between satellite vehicles and receivers.

The precision code, known as P(Y), is a signal transmitted at 10.23 Mbit per second, designed specifically for military use and therefore encrypted. Although it is not meant for civilian applications, it is still broadcast on the L1 frequency at a rate ten times faster than the C/A code.

The carrier wave for GPS L1 signals has a frequency of 1575.42 MHz and a constant amplitude, generated by a transmitter in a satellite vehicle [2]. Essentially, the carrier wave can be considered blank, utilized to propagate data such as navigation messages and C/A codes from the satellite vehicle to receivers. Thus, the carrier wave must be able to vary, where the variation is the data encoded onto the carrier wave. The varying of the carrier wave to represent data is known as modulation. In the overview of the GPS L1 signal shown in Figure 2.1, two modulation methods are used: XOR and binary phase shift keying (BPSK). XOR can serve as a form of digital modulation, and this is also illustrated in Figure 2.1, where the C/A code is XORed with the navigation data and the P(Y) code is likewise XORed with the navigation data. In BPSK modulation, information is imposed on the carrier by applying a 180-degree phase shift to the L1 carrier whenever the data is applied. Thus, the digital data is first modulated, and this modulated digital signal is then imposed onto the L1 carrier. The signal is then broadcast from satellite vehicles in MEO.

A broadcast signal encounters a variety of interferences before it reaches a receiver. Thereby, the quality of a GNSS signal is measured using a carrier-to-noise-density ratio (C/N_0) and models the received power C relative to the noise density (N_0) [2]. C/N_0 is expressed in hertz or decibel-hertz (dB-Hz) and is defined by

$$\frac{C}{N_0} = \frac{C}{N}\frac{1}{B} = \frac{\frac{C}{N}}{B}, \tag{2.1}$$

where C/N is the conventional carrier-to-noise ratio and B is the receiver's bandwidth (Hz). In practice, a GNSS receiver can estimate (C/N_0) by measuring the correlation peak P_{peak} and the average noise floor P_{noise} in the receiver chain. Normal C/N_0 values for GPS signals range between 35-55 dB-Hz, depending on the surrounding environment [23].

2.1.2 The GPS Receiver

A GPS receiver is a device that captures GPS signals and typically consists of an antenna, analogue circuitry, and digital processing [2]. An overview of a normal GPS receiver is shown in Figure 2.2. Note that individual receiver designs may vary, and the figure is meant to provide a general outline tied to the problem at hand. As illustrated, the process begins when GPS signals reach the antenna, are amplified by a preamplifier, then passed to the downconverter in the analogue section, forwarded to the Analogue-to-Digital Converter (ADC), and processed in a digital processing channel. The following section aims to provide an overview of a typical GPS receiver and use the aforementioned figure as a visual aid.

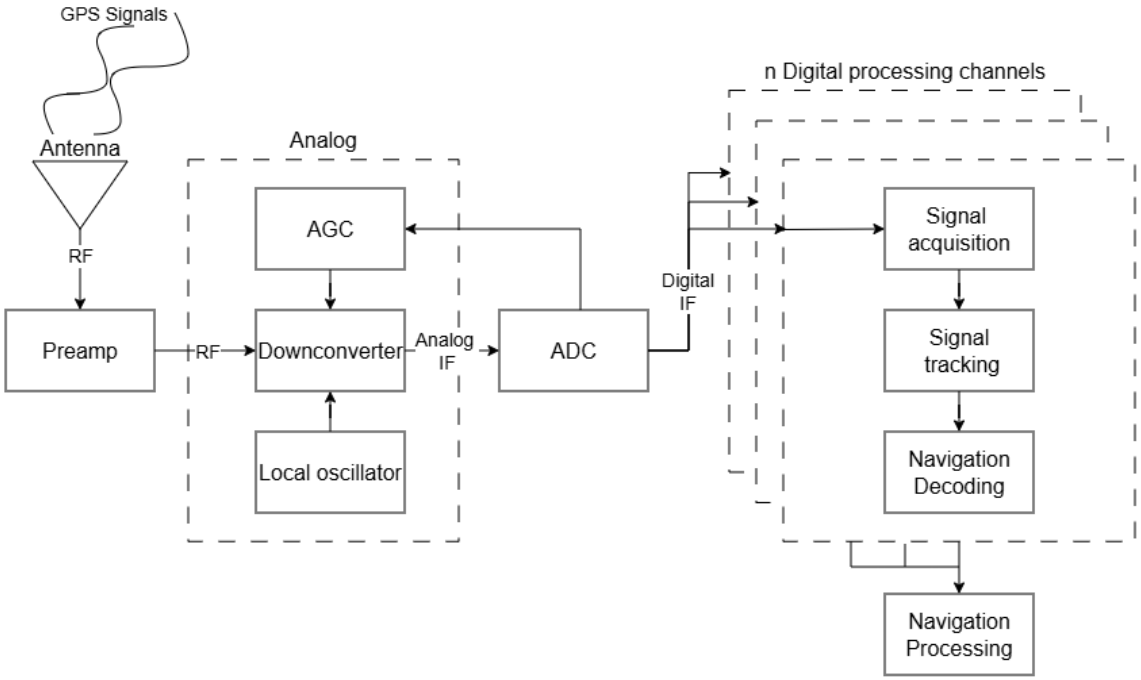


Figure 2.2: A block diagram illustrating typical components of a GPS receiver. The path for a GPS signal in space, to analogue and digital processing, and a navigation result is also showcased.

When arriving at Earth's surface, GPS signals are extremely weak and typically lie below the thermal noise floor. To enable detection, the GPS receiver utilizes an electronic oscillator whose frequency is closely matched to that of the incoming GPS carrier wave [2]. This local oscillation enables processing of the received signal, thereby allowing the receiver to acquire signals whose power is significantly below

the ambient noise level. Nevertheless, the radio-frequency (RF) signal remains very weak at the point of reception by the GPS antenna. To mitigate the risk of the signal being overwhelmed by subsequent circuit noise, it is commonly amplified using a low-noise preamplifier positioned immediately after the antenna output [2]. Once amplified, the signal is downconverted to an intermediate frequency (IF) since the original L1 carrier frequency of 1575.42 MHz contains more data than a receiver channel can process directly at once. Downconversion is achieved by generating a local RF signal at a slightly lower frequency with a local oscillator. This locally generated signal is used as a wipe-off wave to effectively subtract the original L1 frequency component, leaving a signal at the IF. The analog IF signal can then be passed to an ADC, where it is converted into a digital binary. The ADC sampling rate in a civilian GPS receiver is typically chosen in order to be able to capture the C/A code received at 1.023 Mbps, and thus the sampling rate must be at least twice as much as the PRN code [2].

The digitized signal at IF is passed to n parallel digital processing channels, where each channel is responsible for tracking the carrier and code of an individual satellite. In typical GPS receivers, the number of channels ranges from about 12 to 100 in general GNSS receivers [2]. Each channel performs acquisition and tracking independently of the others. Satellite signal acquisition involves generating a local replica of the PRN code for each satellite ID and correlating it with the incoming signal. The acquisition process makes it possible to determine the code delay, that is, the time offset between signal transmission and reception. The acquisition stage also estimates the Doppler shift due to the satellite's motion, and thus, the acquisition stage of Figure 2.2 illustrating the GPS receiver block diagram, provides both Doppler and time offset necessary for tracking. Note that acquisition of satellites in the receiver chain is a central part of this thesis, and the essential takeaway is that performing acquisition results in a 2D search grid, a cross-ambiguity function (CAF).

The result of the two-dimensional search performed during the acquisition stage is thereafter utilized in tracking, where the detected peak in Doppler frequency and code delay are used as an initial estimate for the tracking process. Various types of tracking loops can be implemented: Phase-Locked Loops (PLLs), Frequency-Locked Loops (FLLs), and Delay-Locked Loops (DLLs) [24]. Fundamentally, these loops are control systems within the receiver that provide feedback based on measurements used for positioning. The detailed internal operation of each loop type is not essential in this context, but briefly, a PLL employs a voltage-controlled oscillator that is continuously adjusted to follow the incoming signal's phase and frequency, which is the mechanism by which a receiver acquires and maintains lock on a signal and continuously tracks it [2]. For a GPS receiver to provide a navigation solution (a GPS position), the receiver must also decode the navigation message.

The navigation message is essential for a GPS receiver because it provides the information required to solve the system of positioning equations, the equations resulting in a GPS coordinate. The receiver uses the navigation message to retrieve the broadcast ephemerides (satellite orbital parameters) and timing information, specifically

the satellite position and time of transmission [22]. This information must be available for at least four satellites for the GPS receiver to solve the equation system of its own three-dimensional position and local time.

Assuming that the signal travels at the speed of light, the distance between each satellite and the GPS receiver can be solved from the measured signal travel time [2]. In essence, the calculation of the GPS receiver's position consists of solving a non-linear system of equations in which the unknowns are the receiver's coordinates and its clock offset relative to GPS system time. The known quantities are the satellite positions, obtained from the navigation messages, and the measured pseudoranges calculated from the signal travel time. Without the navigation message, the satellite positions cannot be determined, resulting in the system of equations being unsolvable. Furthermore, without signals from at least four satellites, the equation system also becomes unsolvable due to there being too many unknown variables. The solution of this equation system yields the navigation solution, expressed in GPS coordinates [22]. In addition to positioning, continuous reception and tracking of GPS signals enable highly accurate time measurement, providing globally consistent, precise time synchronization.

Thus, in summary, at the receiver front end, the incoming GPS signal hits the antenna, is amplified by low-noise amplification stages, and is then downconverted and digitized by an ADC. The resulting digital signal is processed in one or more digital signal processing channels, where code and carrier tracking are performed, satellite Doppler shifts and timing offsets are estimated, and the navigation message is extracted. These processing steps together allow the receiver to extract the necessary variables and navigation data to compute both position and precise time.

2.2 GNSS/GPS Spoofing

Spoofing consists of broadcasting a false signal to trick a GNNS receiver into locking onto it and thus manipulating the receiver's position [8]. However, spoofing is not a homogeneous concept, as it can be achieved in a multitude of ways [19]. The attack can be targeted when the attacker knows where a GNSS receiver is located; it could be as simple as recording and rebroadcasting signals, or as complex as trying to cancel out a real signal. The following is an overview that delves into spoofing varieties, their mechanisms, and the level of sophistication.

2.2.1 Asynchronous and Replay-Based Attacks

Unsynchronized spoofing attacks do not attempt to align transmitted signals with authentic signals present at the same time and location. An example is *meaconing*, a spoofing technique in which authentic signals are recorded and then retransmitted with a time delay [8]. If this delay is short, the replayed signals closely resemble the satellite constellation currently visible, but a longer delay can correspond to an entirely different constellation. Rebroadcasting GNSS signals that no longer reflect the actual satellite constellation at the receiver's location can also be regarded as

an untargeted attack, since the attacker does not know where the victim receiver is situated and blindly transmits fake signals.

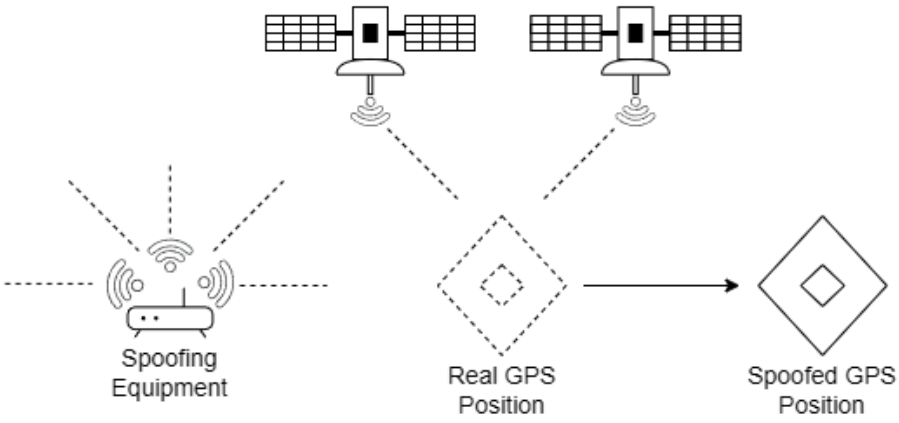


Figure 2.3: An untargeted spoofing scenario not relying on the current constellation. Untargeted spoofing is inherently asynchronous, it cannot achieve synchronization with a receiver when the receiver's position and the currently tracked satellite constellation are unknown.

2.2.2 Overpowering and Takeover Behaviour

A key characteristic of spoofing attacks is that the transmitted signal must have high power to successfully take control of the receiver chain. In practice, the introduction of an overpowering spoofing attack can initially appear as jamming due to the authentic signal being weaker than the spoofed signal, causing a loss of tracking and forcing the receiver to re-acquire [19]. Due to the power advantage, a spoofed signal would appear as the dominant signal, which also gets acquired. Thus, a common methodology for spoofing detection is to observe shifts in carrier-to-noise ratio and Doppler shift, since jumps could indicate a hijacking [8]. Note that this method of detection only works when spoofing is introduced, but does not work when starting up a GNSS receiver when spoofing is already ongoing.

2.2.3 Synchronized (Carry-off) Attacks

Synchronized attacks are possible when the attacker knows the position of a receiver [10]. The aim is to align spoofed GPS signals with authentic signals by observing the current constellation at a given position and time. The purpose is, thereafter, to transmit a fake but synchronized signal towards a designated position. Synchronized in the meaning that the spoofed signal arrives at a receiver simultaneously as authentic signals arrive from GPS satellite vehicles. It is possible to either try to align the position or the time with the attacked receiver, either by initializing the attack with a low-power signal advantage over the authentic signals. In this case, the receiver would not be initially jammed, but instead would introduce an extra copy of a signal being received [10]. Then, the purpose would be to carry off the spoofed signal by slowly increasing the power advantage. When the carry-off attack

has been successful, the spoofer can start by either alternating the time of arrival of the signal and then affecting the positioning, having the receiver fully track the spoofed signal, and keeping the authentic signal suppressed. Thus, a slow and controlled transition could be used in order for spoofing attacks not to be detected.

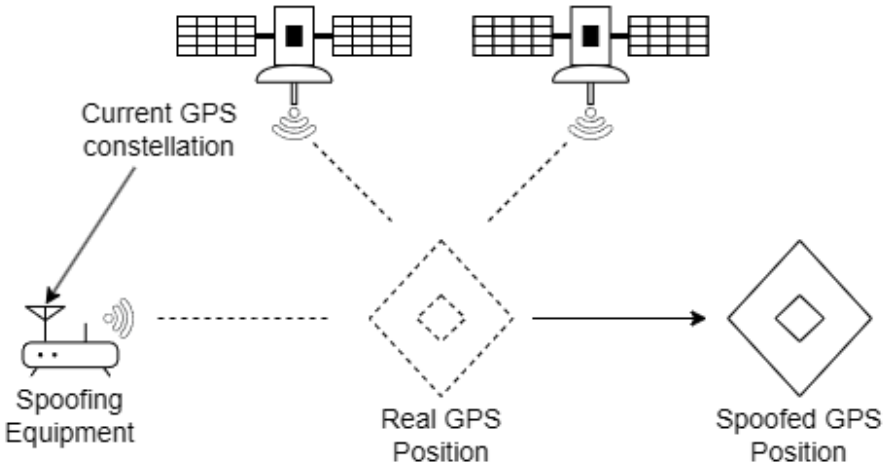


Figure 2.4: A targeted spoofing scenario where the spoofer mimics the current constellation. Targeted spoofing when the attacker is aware of a receiver's position. The attacker can try to synchronize the spoofed signal with authentic signals from satellite vehicles.

The level of sophistication is higher compared to the unsynchronized attacks, where there would be no targeting, and a blunt higher-power advantage is necessary to force the receiver to re-acquire signals [19]. But with synchronized attacks, the principle is to seize control of the receiver's tracking loop without losing any tracking lock. There are detection possibilities for synchronized attacks when the spoofed signals are introduced, since the spoofed signal must take over the authentic signal, and thus, the Doppler shift and carrier-to-noise ratio would also need to be altered.

2.3 Machine Learning Foundations: Focus on CNN

In conventional software development, the programmer defines the decision rules that are applied to data. When a predefined threshold is reached, the corresponding rule is triggered, and an action is executed. By contrast, machine learning shifts the emphasis from hand-crafted rules to automatic pattern discovery where algorithms learn decision rules directly from data without manual intervention [25]. For instance, a model can be trained on pre-processed GPS signals so that it learns to classify each sample as authentic or spoofed, whilst in traditional programming, a developer would have to encode such rules manually. The choice of a decision-making model, therefore, depends on how the problem is represented. When the data are labelled with categories or ground-truth values, supervised learning is the appropriate paradigm. Neural networks constitute one widely used class of supervised models.

2.3.1 Neural Network

Neural networks are machine learning models that are loosely inspired by how biological brains work. The analogy is not exact, but it provides a useful conceptual comparison. In this analogy, biological neurons are abstracted as nodes, and the synaptic connections are represented by weights (trainable parameters) [25]. Figure 2.5 illustrates a simple feed-forward neural network that will be used as a visual aid. It illustrates that a neural network can receive input, possibly visual sensor data, and thereafter output a result, such as interpreting the content of the visual data.

In a standard feed-forward architecture, each input is multiplied by the weights of all connected nodes. These weights are **not** set manually, they are initialized with random values and then updated iteratively during the network training phase [25]. Connections that are found to be important are strengthened, whereas less useful connections are weakened. The weight adjustment process, called backpropagation, is how a network learns, as discussed later.

All the nodes in one part of the network are called a layer, and a network typically consists of a multitude of layers. The schematic in Figure 2.5 shows the flow from the input layer, through one or more hidden layers, to the output layer. Note that "hidden layers" are layers of nodes located between the input and the output layer. A node in any layer becomes active only when its total input exceeds the activation threshold defined by a chosen activation function.

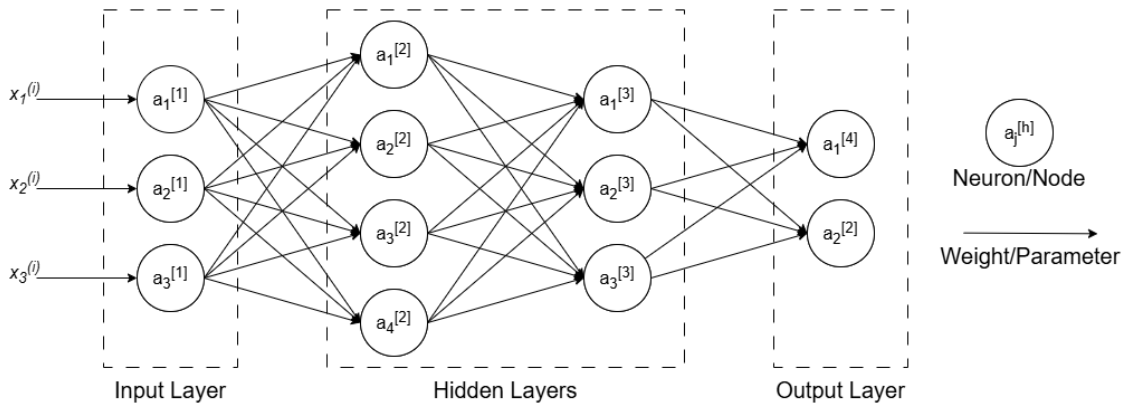


Figure 2.5: An illustrative schematic of a fully connected feed-forward neural network. Comprising three input units, two hidden layers with the first hidden layer containing four neurons and the second containing three neurons and a final output layer with two output units.

The activation of the j^{th} neuron in layer h is denoted by $a_j^{[h]}$. The activation of the neuron $a_j^{[h]}$ depends on the weights that connect the preceding layer to the neuron. Lets denote a weight between two neurons as $w_{a_j^{[h-1]} \to a_j^{[h]}}$. Each neuron's activation is the sum of its inputs plus an offset term called the bias (analogous to the intercept in a linear function). Hence, the activation of neuron $a_1^{[2]}$ consists of the weighted sum of the preceding activations (plus a bias), such as

$$a_1^{[2]} = w_{a_1^{[1]} \to a_1^{[2]}} \cdot a_1^{[1]} + w_{a_2^{[1]} \to a_1^{[2]}} \cdot a_2^{[1]} + w_{a_3^{[1]} \to a_1^{[2]}} \cdot a_3^{[1]} + b_1^{[2]} \tag{2.2}$$

The bias term provides an additional degree of freedom, allowing the activation to be shifted away from a origo [25]. The activation for neuron $a_1^{[2]}$ is linear, as shown in Equation 2.2. To enable neural networks to model complex non-linear relationships, activation functions are applied [26]. The activation function determines if a neuron/node should activate at all or how much it should activate. Thus, Equation 2.2 is the activity of neuron $a_1^{[2]}$ without an activation function. Let σ denote the activation function, and the full modelling of neuron $a_1^{[2]}$ becomes

$$a_1^{[2]} = \sigma\left(w_{a_1^{[1]} \to a_1^{[2]}} \cdot a_1^{[1]} + w_{a_2^{[1]} \to a_1^{[2]}} \cdot a_2^{[1]} + w_{a_3^{[1]} \to a_1^{[2]}} \cdot a_3^{[1]} + b_1^{[2]}\right) \tag{2.3}$$

which yields the final output of node $a_1^{[2]}$ including the activation function σ. Many activation functions have been proposed. Early neural network work used the sigmoid function, while the most widely used today is the rectified linear unit (ReLU) function [26]. The sigmoid, defined in Equation 2.4, compresses its input to the interval [0,1], thereby limiting extreme activations. ReLU, given in Equation 2.5, outputs zero for negative inputs and the positive value for non-negative inputs, mimicking the "all-or-nothing" firing behaviour observed in biological neurons.

$$\sigma_{\text{Sigmoid}} = \frac{1}{1 + e^{-a_j^{[h]}}} \tag{2.4}$$

$$\sigma_{\text{ReLU}} = \begin{cases} 0 & a_j^{[h]} < 0 \\ a_j^{[h]} & a_j^{[h]} \geq 0 \end{cases} \tag{2.5}$$

Thus, Figure 2.5 exemplifies a feed-forward neural network where the activation of each neuron is obtained by weighting the outputs of the preceding layer and applying the chosen activation function. Input features, placed on the left, are propagated strictly forward through successive layers, and the network output is produced by processing the activations of all preceding layers in sequence.

During the forward pass, the network does not assess the relevance of its weights, instead the ground-truth target for a given input is known, and the discrepancy between the predicted output and the true output (the prediction error) drives the weight update. This error-driven optimization, in which the error is propagated backward through the layers, is called backpropagation and is the learning step of the network [25]. Learning corresponds to minimizing a loss function to find either a local or global optimum. For classification tasks with known labels, the cross-entropy loss is frequently utilized:

$$\text{Cross Entropy} = -\sum_{i=1}^{C} t_i \cdot \log(p_i) \tag{2.6}$$

where $t_i \in 0, 1$ denotes the target label for class i (binary case) and p_i is the predicted probability for that class. An incorrect high-confidence prediction results in a large loss value, thereby providing a quantitative measure of the network's performance. The loss function defines a high-dimensional error surface over the weight

space, where optimization algorithms navigate this surface to locate local or global optima. A widely used optimiser is (stochastic gradient) descent, which iteratively adjusts parameters in the direction of the loss gradient [25]. The network's weights are adjusted iteratively and the goal is to make the prediction error as low as possible.

However, what the network outputs depends on the task at hand. If the goal is regression, the network utilizes a single scalar output. For a K-class classification task, it produces K *logits*, which are real-valued scores, one per class. The class with the maximal logit is taken as the network's prediction. Rather than selecting the maximal logit directly, a *softmax layer* is commonly appended to the output. The softmax layer converts these scores into a categorical probability by re-weighting them with the softmax function.

$$\text{Softmax}(\text{output}_i) = \frac{e^{\text{output}_i}}{\sum_{j=1}^{K} e^{\text{output}_j}} \tag{2.7}$$

Applying the softmax to all logits yields a probability distribution over the K classes. Consequently, a neural network can output raw logits (continuous score) or, after the softmax, class probabilities.

2.3.2 Convolutional Neural Network (CNN)

A multitude of architectures belong to the family of artificial neural networks. CNNs constitute a subclass that is specifically designed to extract features from grid-structured data (possibly images) [27]. They achieve this by applying learnable filters (kernels) over the input, thereby automatically learning and detecting patterns. The previously discussed (fully-connected) neural network can be viewed as the decision-making component, whereas the convolutional backbone acts as a feature extractor. The input of a CNN is fed to one or more fully-connected layers that perform the final classification. This architecture is illustrated in Figure 2.6, where a CNN is shown as a feature extraction module followed by fully-connected layers and a classification probability. There is a wide variety of layer types [28], but we will focus on those that are most relevant to the present work.

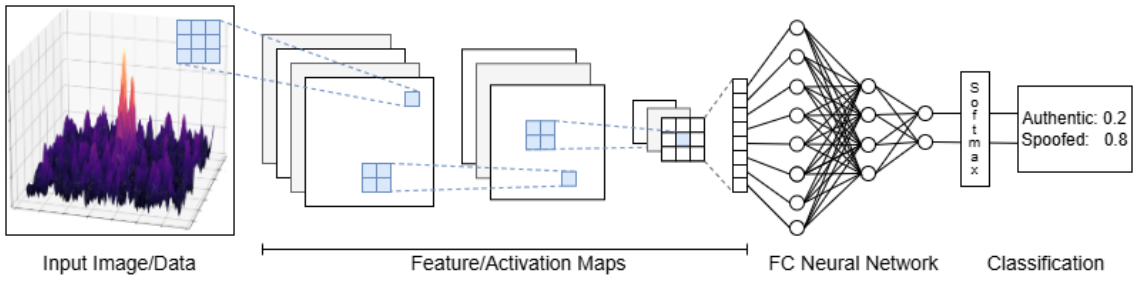


Figure 2.6: An abstract overview of how a CNN works. Showcasing filters being applied to an input image, resulting in activation maps, which are later fed to a fully connected neural network. The classification results are provided by a Softmax layer.

Figure 2.7 shows how a CNN kernel is applied to an input image. A CNN does not employ a single kernel but many of them, where the kernel values are learned during training and are updated via backpropagation [27], just like the weights of a conventional neural network. While a kernel is "dragged" over the image, it performs a convolution product between its current parameters and the underlying pixel values, producing a feature (activation) map. When the kernel's pattern matches a structure present in the image, the corresponding activations are stronger. Kernels need not be square; their spatial dimensions define the filter size, which is a hyperparameter chosen before training [27]. Thus, learned filters extract the most informative aspects of an image for subsequent decision making. Convolutional layers are, however, not the only type of layer used in CNNs.

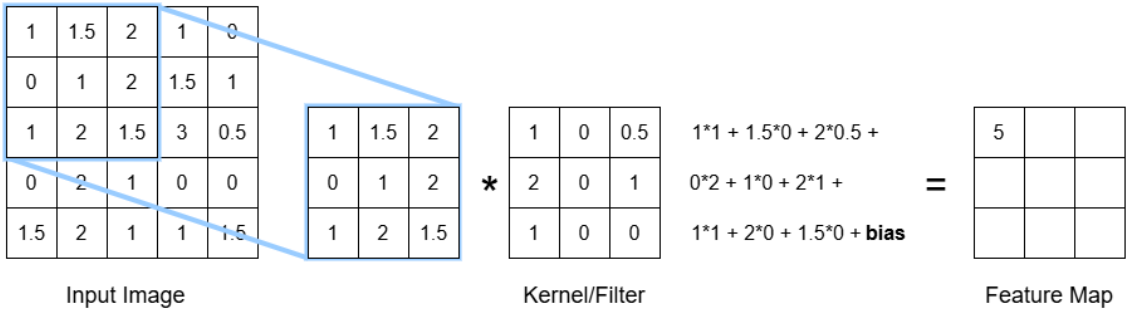


Figure 2.7: An illustrative example demonstrating the convolution operation using a kernel (filter). A kernel is applied to a matrix of inputs, the operation, and the resulting scalar.

Pooling layers are used to downscale feature maps because an image usually contains redundant or irrelevant information. By applying a pooling operation, the spatial resolution is reduced while the most "important" features are kept [27]. Figure 2.8 illustrates the effect of a max-pooling filter. The filter has a size of 2x2 and is swept across the input map. For each window, it selects the largest value, thereby decreasing the dimensions but preserving the most important activations. The figure shows the matrix before pooling and the resulting matrix after pooling, where only the maximal values remain.

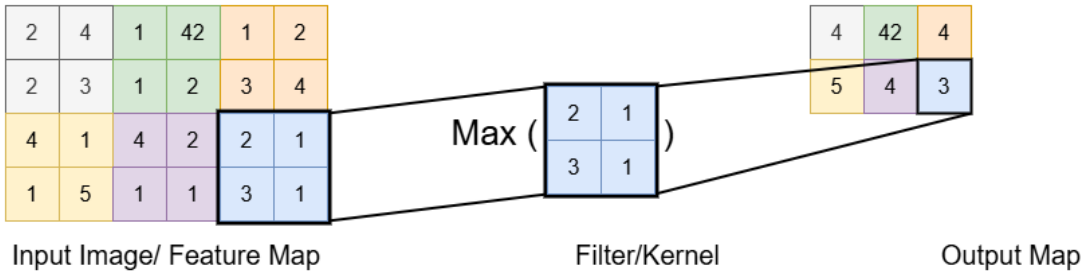


Figure 2.8: An illustrative example demonstrating the operation of a max-pooling kernel (filter) when applied to an input image or feature map. Showing the original matrix, the kernel, and the resulting downsized feature map.

Dropout is another layer that is frequently used in CNNs. The purpose of a dropout layer is to *randomly* deactivate a fraction of the network's weights during each training iteration [27]. Although forgetting weights may appear counterintuitive, the technique promotes generalization. Instead of allowing the network to adjust all weights so as to fit the training data perfectly, a subset of weights is temporarily removed. Thereby, the model cannot rely on a few specialized pathways, it must learn to achieve the same output through many different configurations. This forces the network to develop a representation that generalizes beyond the specific training examples.

There are also *batch normalization* layers, which aim to normalize the output of each preceding layer [29]. Their purpose is to centre the activations around 0 and to apply standard-scaling, thereby reducing the impact of a phenomenon that frequently makes CNN training difficult: the *covariate shift* [29]. Covariate shift refers to a change in the statistical distribution of the inputs between the training phase and deployment in a different environment, which can cause the model's performance to deteriorate. By normalizing the activations during training, batch normalization mitigates this shift and makes the CNN more robust to variations in the data.

Thus, in summary, a CNN is built from a sequence of distinct layers that are applied after the input.

- **Convolutional layers** perform feature extraction by convolving learnable kernels with the input.
- **Pooling layers** reduce spatial resolution and select important activations.
- **Dropout & Batch normalization** act as regularizers, improving the network's ability to generalize.
- **Activation-function layers** introduce non-linearity, enabling the model to capture complex relationships

These layers are stacked in a fixed order where the input image passes through the pipeline until the final stage, where a fully-connected layer maps the extracted features to a class score or other variables. This final mapping provides the decision-making component of the CNN.

3
Method

The initial phase of the methodology involved the generation of synthetic GPS signal data using MATLAB. Thereafter, real GPS signal recordings were collected from publicly available repositories. Based on these datasets, a data processing pipeline was implemented to enable the loading of binary files and then perform satellite acquisition. During the acquisition stage, two-dimensional search images, Doppler frequency estimates, and corresponding code delay values were computed and stored. These acquisition maps were then used to train a CNN, where the CNN was provided with both spoofed and authentic acquisition images as labelled input. For performance evaluation, the following metrics were considered: classification accuracy, balanced accuracy, Matthews correlation coefficient, and timelines illustrating the classification result.

3.1 GPS Signal Data Simulated Using MATLAB

To generate simulated GPS signal data using MATLAB, a scenario was first defined that included a hypothetical position of a receiver, the time of day, and the set of visible satellites. The receiver was positioned on the ground by manually specifying GPS coordinates in an open, flat area. In the simulation, foliage, signal reflections from buildings, and other obstacles did not influence the receiver's position. The simulated recording time was varied to produce different visible satellite constellations. The true GPS constellation corresponding to each selected day and receiver location was obtained from GPS almanacs [30]. The GPS receiver model also allowed configuration of a minimum elevation angle above the horizon and the minimum elevation angle used for every simulated scenario was 5 degrees. Figure 3.1 shows an example of satellite visibility for a given scenario. Based on this setup, the signal properties of Doppler shift and time delay were calculated from each satellite relative to the receiver.

The GPS L1-C/A signal was generated using the MATLAB gpsWaveformGenerator object [31]. This object produced a baseband waveform at a sampling rate of 5 MHz for all recordings. For each satellite, the corresponding C/A PRN code was applied at its native chip rate of 1.023 MHz, and the navigation message bits were modulated onto the resulting waveform. The navigation bits were obtained from the GPS almanac, which was downloaded for the specific scenario under consideration. The P(Y)-code was not generated with gpsWaveformGenerator, as it does not convey information intended for civilian users, and its omission does not affect

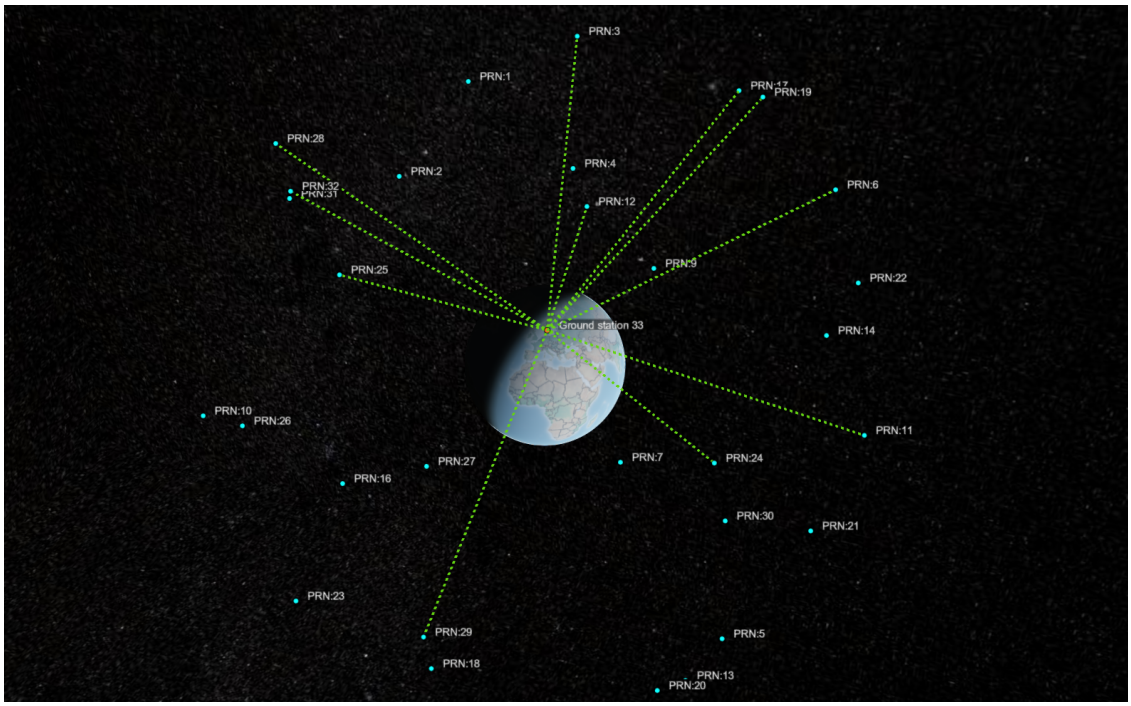


Figure 3.1: A satellite scenario created using MATLAB. Satellites visible to the ground station are annotated using lines between the satellites and the receiver. Receiver situated in Sweden, the scenario played out at 10:00 the 16th of February.

the characteristics of the L1-C/A signal. At this stage, the signal was still ideal, consisting solely of the original waveform with no distortions. To approximate a realistic received signal, several impairments were introduced, including the computed Doppler shift and propagation delay, together with noise to represent passage through atmospheric layers and other environmental disturbances. Effects were incorporated to degrade the signal-to-noise ratio by adding additive white Gaussian noise. The recordings were generated under three C/N_0 conditions: 40–42 dB-Hz, 44–46 dB-Hz, and 48–50 dB-Hz.

Each simulation was divided into two segments: an initial portion of the recording containing only authentic signals, followed by the point in time at which spoofing was introduced. All simulated recordings had a total duration of 75 seconds, with spoofing starting at the 35-second mark. Once spoofing began, all visible satellite PRN IDs were spoofed. For each visible PRN ID, a Doppler shift and code-phase delay were drawn uniformly from a range of Doppler shifts of -1000 to 1000 Hz and a time delay range of 1 to 10 μs. The generated spoofed signal was added on top of the authentic signal at the time mark of spoofing introduction. The set values for the specific PRN ID were static until the end of the recording. The power of the spoofing signal, measured in dB-Hz, was alternated for each simulated recording and not changed during a recording. A full summary of the different configurations is found in Table 3.1.

Note that there are two important remarks regarding how the data were simu-

C/N_0 (dbHz)	40-42	40-42	40-42	44-46	44-46	44-46	48-50	48-50	48-50
Spoofer Advantage (dBhz)	1-2	4-6	9-11	1-2	4-6	9-11	1-2	4-6	9-11

Table 3.1: A summary of C/N_0 configurations and spoofing power advantages used in generated MATLAB recordings. All recordings utilize a sampling rate of 5 MHz and a bit width of 16 (I/Q). In total, there were 9 recordings, one of each C/N_0 and spoofer advantage.

lated using MATLAB. First, the generated signals are Signal-in-Space (SiS) and have thus not undergone receiver processing. Thereby, the simulated signals will differ from signals that have been acquired, tracked, and processed by an actual receiver. Secondly, for the sake of reproducibility, it should be noted that the GPS signal generation in MATLAB was initially based on a MATLAB guide describing the generation of GNSS signals [32]. The original guide was, however, extensively modified to support the storage of raw signal recordings and metadata, and enable spoofing injection by adding stronger authentic-like signals with configurable time and Doppler offsets onto the original signal.

3.2 Open Source GPS Signal Datasets

Two publicly available datasets were selected to examine the transition from simulated data to real-world recordings. The first dataset repository from the Finnish Geospatial Research Institute (FGI) contains four distinct spoofing scenarios: an untargeted attack, meaconing, a targeted single-frequency attack, and a targeted dual-frequency attack in which the GPS L5 band is also spoofed [19]. The second dataset, the Oak Ridge Spoofing and Interference Test Battery (OAKBAT), consists of seven scenarios, all of which involve targeted spoofing attacks but one [33]. These scenarios encompass a broader range of conditions, including both static receiver placement and dynamic configurations in which the receiver was placed on a vehicle. One of the OAKBAT scenarios contains a clean recording with only authentic GNSS signals. A summary of the scenarios in the FGI and OAKBAT repositories is provided in Table 3.2.

Beyond spoofing type, bit width, and sample rate, the two collections differ in hardware setup. In OAKBAT, both authentic live-sky signals and spoofed signals are generated by a hardware transmitter, allowing the recordings to be captured with a GPS receiver [33]. In contrast, the FGI recordings contain genuine live-sky signals captured at a fixed site [19]. The spoofed signals were then injected via hardware simulators because true spoofed signals are not available at the measurement location. Consequently, both repositories rely on hardware-simulated spoofed signals as a necessity, since acquiring genuine spoofed transmission from an adversary is infeasible for controlled experiments.

Scenario	Type of spoofing	Sample Rate	Bit width	Description
FGI [19]				
TGS	Targeted, time & position synchronous (static receiver)	26 MHz	8 bit (I)	Single-frequency spoofing (L1); L5 left authentic.
TGD	Targeted, time & position synchronous (static receiver)	26 MHz	8 bit (I)	Dual-frequency spoofing across L1 and L5.
UTD	Untargeted, time & position asynchronous (static receiver)	26 MHz	8 bit (I)	Dual-frequency; spoofed position ∼15 km off, time advanced ∼10 h.
MCD	Meaconing (re-radiation)	26 MHz	8 bit (I)	Dual-frequency delayed rebroadcast of authentic signals (∼15 min delay).
OAKBAT [33]				
CleanStatic	No spoofing present	5 MHz	16 bit (I/Q)	A recording to use as baseline.
os2	Targeted, static receiver, time push	5 MHz	16 bit (I/Q)	2 µs time push with 10 dB spoofing power advantage.
os3	Targeted, static receiver, time push	5 MHz	16 bit (I/Q)	2 µs time push with 1.3 dB spoofing power advantage.
os4	Targeted, static receiver, position push	5 MHz	16 bit (I/Q)	600 m position push with 0.4 dB spoofing power advantage.
os5	Targeted, dynamic receiver, time push	5 MHz	16 bit (I/Q)	2 µs time push with 9.9 dB spoofing power advantage.
os6	Targeted, dynamic receiver, position push	5 MHz	16 bit (I/Q)	600 m position push with 0.8 dB spoofing power advantage.

Table 3.2: A summary of the scenarios found in the FGI and OAKBAT spoofing repositories. Tabulating the type of spoofing in each recording, its sample rate, bit width, and a description of the scenario.

3.3 Acquisition Map Extraction

As described in the theory section, each GPS satellite transmitted a unique code that repeats every 1 ms in the GPS L1 C/A signal. The code is used to acquire a signal by searching over a two-dimensional grid of Doppler-frequency offsets and code-phase delays. For each Doppler, the locally generated PRN was correlated with a 1 ms data segment, producing an *acquisition map* (also referred to as a search map or CAF-map in this thesis). The map indicated the pair of Doppler shift and time delay, which the received PRN matched the locally generated replica.

The Doppler search covered the interval of -10 kHz $\leq f_{\text{Doppler}} \leq 10$ kHz relative to the intermediate frequency. Rather than evaluating every hertz, a Doppler-bin resolution was utilized. With a 200 Hz bin size, the 20 kHz span resulted in 100 bins (20000 Hz /200 Hz = 100 bins). To study the effect of resolution, five different bin sizes (50 Hz, 75 Hz, 100 Hz, 125 Hz, and 150 Hz) were examined.

The size of the acquisition map depended on the sampling rate. A 5 MHz sampling rate (used for the MATLAB simulations and OAKBAT) produced 5000 samples per 1 ms snippet. Consequently, with a 200 Hz Doppler resolution, the correlation of the 5000 sample segment against 100 Doppler values tested generated a raw map of 5000 x 100 (code-phase samples x Doppler bins). Because only a narrow region around the correlation peak contained useful information, the full raw map was not used.

3.3.1 Region-Of-Interest (ROI) Acquisition Map

To obtain a zoomed-in view of the dominant peak, a Region-of-Interest (ROI) was defined. The ROI was centred on the largest correlation value in the raw map and sized to 128 code-phase samples x 64 Doppler bins. With a 5 MHz sampling rate, each sample Δt corresponds to

$$\Delta t = \frac{1}{f_s} = \frac{1}{5 \cdot 10^6} = 0.2\mu s \tag{3.1}$$

So the ROI spans $\pm 64 \cdot 0.2\mu s = \pm 12.8\mu s$ around the peak, allowing observation of secondary peaks within this interval. The Doppler value view depends on the chosen bin resolution. For the coarsest resolution (150 Hz), the ROI covers $\pm 32 \cdot 150$ Hz = ± 4.8 kHz. The ROI size, therefore, acts as a hyperparameter. It was chosen to encompass the expected spoofing delay while limiting computational cost

3.3.2 Global (Down-scaled) Acquisition Map

In some scenarios, dual-peak signatures may end up outside of the ROI. To retain a global view, the raw map was down-scaled to 1024 x 64 (code-phase x Doppler) bins. The choice of 1024 code-phase bins is determined by the C/A code length of 1023 bits (chips). Since the C/A code repeats every 1 ms, and consists of 1023 bits (chips), the global map can be at least 1023 code-phase samples wide, information regarding peaks might be lost by squishing different peaks together. The Doppler dimension was fixed at 64 bins to match the ROI resolution. The downsizing was performed differently for each axis of the data:

- **Code-phase axis:** max-pooling was applied, keeping the peak amplitude by selecting the maximum value within each pooling window
- **Doppler axis:** bilinear interpolation was used, which is appropriate because Doppler peaks are relatively broad and the interpolation does not distort their shape.

3.4 Network Design, Fine-Tuning Approach, and Experimental Resources

The proposed CNN utilizes a dual-branch design that processes both the ROI map and the global acquisition map in parallel. Two similar feature-extractor branches are built from the same fundamental building blocks, and at the end, their outputs are concatenated and fed to a common fully-connected classifier. Note that all images (the acquisition maps), both ROI and global, were normalized to the range of [0,1] before being passed to the CNN.

Two basic blocks were first defined (Table 3.3 and Table 3.4). Each block consists of a sequence of convolutional layers, batch-normalisation, ReLU activations, max-pooling, and dropout. The blocks share the same depth and ordering but differ in kernel size. The CAFConvBlock uses a 3x3 kernel, while the CAFGlobalBlock uses a 7x1 and 3x1 kernels to capture elongated structures along the code phase axis of the global map, in order to allow for detection of multiple correlation peaks.

Layers	Kernel Dimensions
Convolutional	3×3
BatchNorm	n/a
ReLU	n/a
Convolutional	3×3
BatchNorm	n/a
ReLU	n/a
MaxPool	2×2
Dropout	n/a

Table 3.3: The internal architecture of the CAFConvBlock. All layers are 2-D variants.

Layer	Kernel Dimensions
Convolutional	7×1
BatchNorm	n/a
ReLU	n/a
Convolutional	3×1
BatchNorm	n/a
ReLU	n/a
MaxPool	2×1
Dropout	n/a

Table 3.4: The internal architecture of the CAFGlobalBlock. All layers are 2-D variants.

The architecture of each branch alternates the appropriate block type: the ROI branch stacks CAFConvBlocks, whereas the global stacks CAFGlobalBlocks (see Table 3.5). After the last block, the two branches are merged by shared fully-connected layers and Softmax output, which returns the probability of authentic versus spoofed acquisition.

Beyond the layer components of the network, the training phase was conducted with a cross-entropy loss function, an AdamW optimizer, and a Cosine learning rate scheduler with an initial learning rate of $\alpha = 10^{-3}$. Additionally, a global weight decay of $\omega = 10^{-3}$ was also utilized. 80 % of the data were used as training data, 10 % as validation data, and the remaining 10 % as test data, and this was the same for all training and fine-tuning scenarios.

ROI Branch	Global Branch
	Input
	CAFGlobalBlock
Input	CAFGlobalBlock
CAFConvBlock	CAFConvBlock
CAFConvBlock	CAFConvBlock
CAFConvBlock	CAFConvBlock
CAFConvBlock	CAFConvBlock
Fully Connected Layer	Fully Connected layer
ReLU	ReLU
Dropout	Dropout
Fully Connected Layer	Fully Connected Layer
Softmax	Softmax

Table 3.5: The internal architecture of the dual branch network for global and ROI input acquisition maps ("images"). Where the divider ends, the dual branches merge and use the same layer, which is at the first fully connected layer.

3.4.1 Fine-Tuning Approach

Fine-tuning of the CNN layer weights was performed on models that had been initially trained exclusively on synthetically generated MATLAB simulation data. During fine-tuning, selected layers are unfrozen (i.e., their weights are unlocked) to allow further parameter adjustment [27]. This approach was also utilized when training the models on FGI and OAKBAT data.

The classifier layers, the fully connected layers listed in Table 3.5, were fine-tuned using the same learning rate as in the initial training on MATLAB data, that is $\alpha = 10^{-3}$. In contrast, the convolutional layers were unfrozen in a blockwise (as CAFConvBlock and as CAFGlobalBlock) manner and optimized with different learning rates depending on their position within the network (early vs. late layers).

In particular, the feature extraction layers in the first two CAF-blocks of both branches, the first two CAFConvBlocks in the ROI branch and the first two CAF-GlobalBlocks in the global branch were updated using a learning rate that was one hundred lower than the original learning rate, $\alpha_{\text{first}} = \alpha \cdot 0.01$. The later blocks and convolutional layers, i.e., those located after the first two blocks in each branch and before the final fully connected layers, were fine-tuned with a learning rate one order of one tenth lower than the original value, $\alpha_{\text{middle}} = \alpha \cdot 0.1$.

3.4.2 Computational Resources

Computations were executed on a server with shared resources managed by a resource management system called SLURM [34]. The following computational resources were reserved and used for the initial extraction of acquisition maps, model training, fine-tuning procedures, and later validation of the trained models:

- **Central Processing Unit (CPU)** – 8 computational cores from an INTEL XEON GOLD 6548N (2 × 64-core processors).
- **Random Access Memory (RAM)** – 32 GiB.
- **Graphics Processing Unit (GPU)** – One NVIDIA L4 with 24 GiB of on-board memory.

3.5 Evaluation Criteria

The differentiation of GPS signals as either spoofed or authentic was evaluated using accuracy, balanced accuracy, Matthews correlation coefficient (MCC), and timeline plots. The reason for applying a multitude of evaluation criteria is to handle the problem of class imbalances and also adhere to the evaluation criteria used within the field of GNSS spoofing detection, such as standard accuracy.

Authentic signal acquisitions were assigned a class label of 0 (as negative), whilst spoofed signal acquisitions were assigned 1 (as positive). Since this was a binary classification problem, four different outcomes were possible:

- **True Positive (TP)** - the model correctly labelled a spoofed acquisition as spoofed.
- **True Negative (TN)** - the model correctly labelled an authentic acquisition as authentic.
- **False Positive (FP)** - the model incorrectly labelled an authentic acquisition as spoofed, creating a false alarm.
- **False Negative (FN)** - the model incorrectly labelled a spoofed acquisition as authentic, not detecting spoofed attacks.

The proportion of all acquisitions correctly classified, i.e., accuracy, is defined as

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.2}$$

Due to class imbalances between the number of spoofed and authentic acquisitions in particular datasets, the metric of balanced accuracy was also used for evaluation. Balanced accuracy is defined as

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right) \tag{3.3}$$

Since the problem was a binary classification problem, MCC was also considered due to its statistical robustness [35]. MCC provides a singular evaluation metric that ranges between -1 and 1, where 1 is perfect classification, 0 is no better than random, and -1 is perfectly inverted predictions. MCC is defined as

$$\text{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{3.4}$$

All metrics considered provide a singular value for each scenario. Thus, in order to provide a visual overview of the result that reflects what the trained model assigns for the probability of spoofing of each acquisition, a heatmap timeline for each GPS recording was plotted. The timeline shows the tracked satellites that are able to be acquired during the run, and the probability of spoofing the CNN model assigned for each acquisition performed. The heat-map timelines also illustrate the point at which spoofing is introduced in the recordings, and thus allow for visual aid from the start of the recording to the end, and which labels were assigned for each sample.

4
Results

Global and ROI acquisition maps were plotted to visualize the data, and a representative subset was selected for presentation. Both authentic and spoofed acquisitions were extracted and displayed. The performance of the trained CNN model was evaluated for several Doppler-search precision values. For each Doppler setting, the best result in each scenario was tabulated. Model generalization was assessed by applying trained CNNs to recordings from the OAKBAT and FGI datasets. Finally, the inference times for acquisition, preprocessing, and CNN processing were measured.

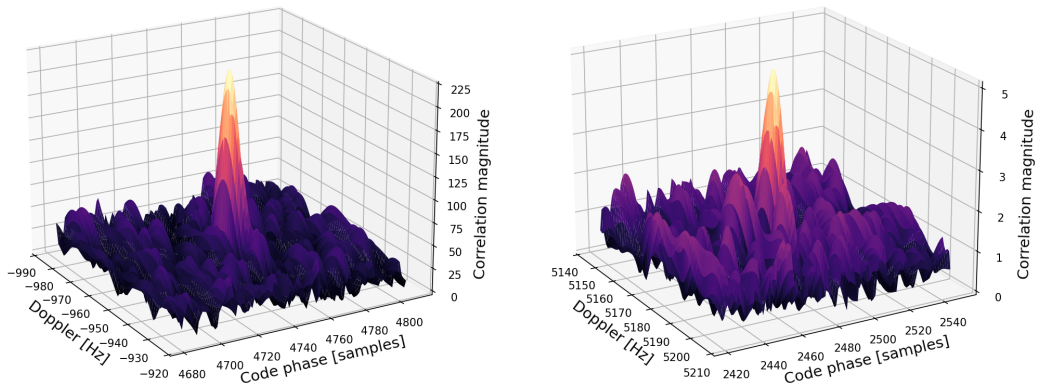
4.1 Global & ROI Acquisition Plots

Approximately 80,000 acquisitions were performed for each Doppler-search precision. From each acquisition, one global map comprised of 1024x64 pixels and one ROI map comprised of 128x64 pixels were extracted, yielding a total of 800,000 2D images for the five Doppler-search values. A representative subset of authentic and spoofed plots was then selected for presentation in order to illustrate typical acquisition map scenarios. All images, together with the corresponding Doppler bin precisions, were deposited in a public GitHub repository [36].

4.1.1 Authentic

Authentic signals, as defined in this project, contained no intentional interference. Thereby, authentic acquisition maps exhibited a single dominant peak both globally and locally. Local maps surrounding the dominant peak are shown in Figure 4.1. The figure includes an example generated from MATLAB simulations and an example obtained from the FGI UTD recording. Correlation magnitudes differed between the plots, but this did not affect the results because every 2-D map was normalized to the interval [0,1] before CNN training. Hence, absolute magnitude differences were irrelevant and only the relative contrast between peak and background mattered. The illustrations hold absolute values to convey that correlation magnitude can vary across recordings and signal-strength conditions.

A global acquisition map from an authentic sample is illustrated in Figure 4.2. Most of the global image consisted of noise, with a single, barely visible peak corresponding to the authentic signal. Signal-strength variations influenced the noise background: a stronger signal produced a darker background because the peak's amplitude was notably higher than the surrounding noise.



(a) Acquisition from MATLAB simulations showcasing an authentic scenario.

(b) Acquisition from FGI UTD showcasing an authentic scenario.

Figure 4.1: ROI acquisition maps from authentic signals. Illustrated in the sample domain, where the peak has been cut out and no resizing performed.

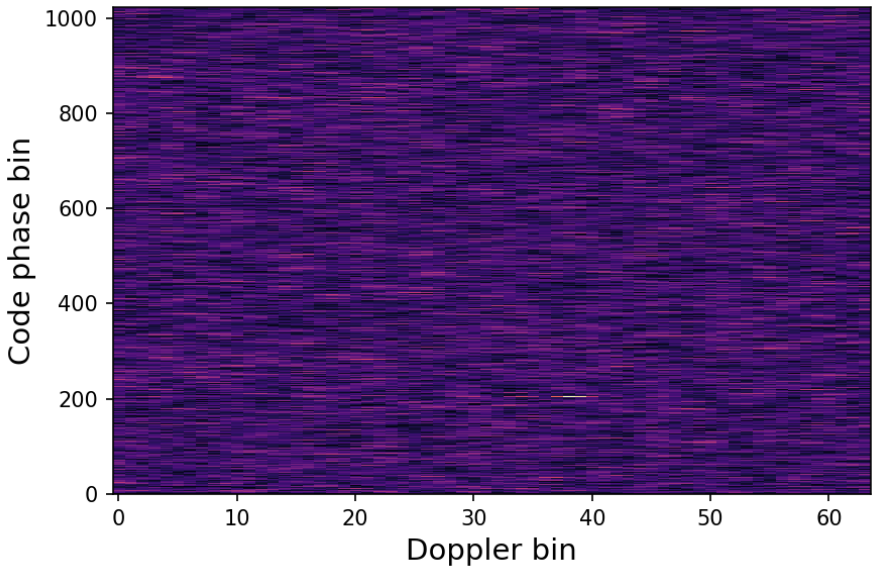
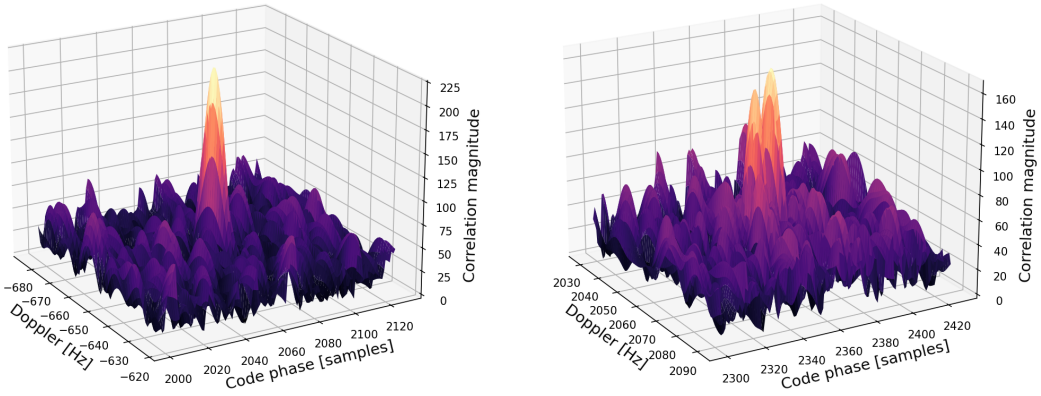


Figure 4.2: A global authentic acquisition map from the OAKBAT os6 recording. Peak at approximately Doppler bin 40 and code phase bin 200. Illustrating the single peak and the surrounding noise.

4.1.2 Spoofed

All spoofed signals contained intentional interference, which could produce up to two peaks in the acquisition maps. However, a second peak was not always observable because a stronger signal could suppress the weaker one down into the noise floor. Both the dual-peak and suppressed-peak cases were illustrated in Figure 4.3, which shows acquisitions extracted from simulated data. The left sub-image resembled an authentic acquisition, displaying a single dominant peak, whereas the right sub-image exhibited two peaks because the spoofing advantage was relatively low.



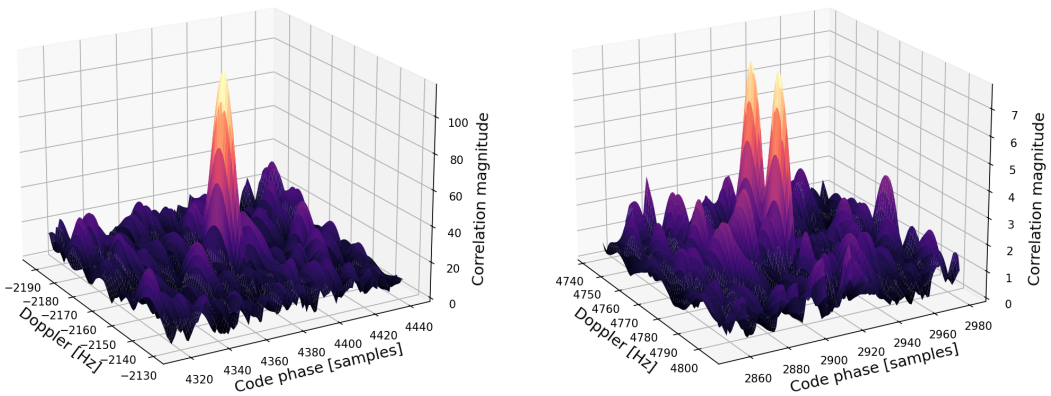
(a) Acquisition from MATLAB simulations showcasing a single spoofed peak scenario.

(b) Acquisition from MATLAB simulations showcasing a spoofed dual peak scenario.

Figure 4.3: ROI acquisition maps from spoofed simulated signals. Illustrated in the sample domain, where the peak has been cut out, and no resizing has been performed.

Figure 4.4 displays spoofed ROI images from the OAKBAT and FGI recordings, reinforcing the similarity between simulated and real data. In the left sub-image (OAKBAT os2), the spoofing advantage was 10 dB-Hz, resulting in the authentic peak being completely buried in the noise and therefore unrecoverable. The right sub-image (FGI UTD) showed a clear dual-peak pattern, even though the exact spoofing advantage of this recording is unknown. It demonstrates that dual-peak scenarios can also be detected in non-simulated recordings.

A global acquisition map from a spoofed scenario is displayed in Figure 4.5. This map does not contain a dual peak but instead a single dominant peak that suppresses the background noise, indicating a relatively strong spoofed signal.



(a) Acquisition from OAKBAT os2 showcasing a single spoofed peak scenario.

(b) Acquisition from FGI UTD showcasing a spoofed dual peak scenario.

Figure 4.4: ROI acquisition maps from spoofed OAKBAT and FGI signals. Illustrated in the sample domain, where the peak has been cut out, and no resizing has been performed.

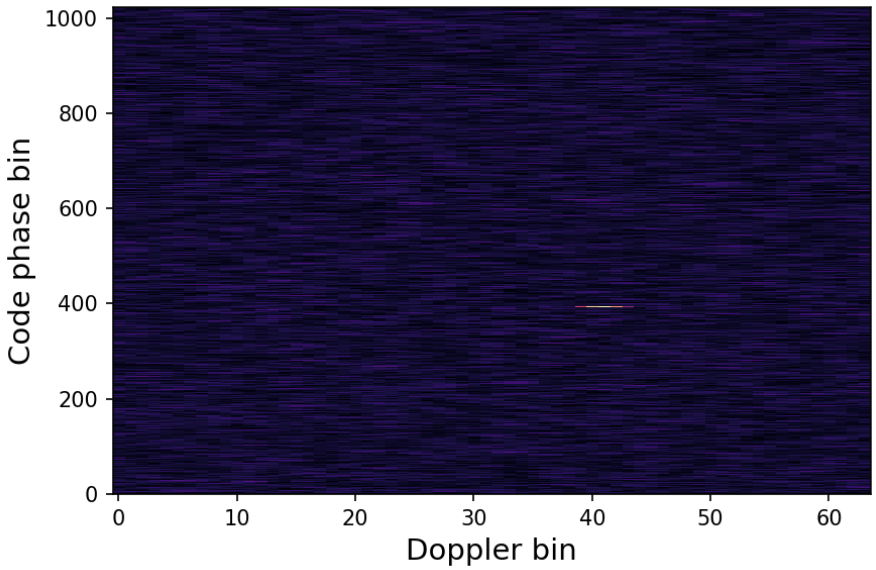


Figure 4.5: A global spoofed acquisition map from the FGI TGS scenario. Peak at approximately Doppler bin 40 and code phase bin 400. Illustrating the single peak and the surrounding suppressed noise.

4.2 Influence of Doppler-Search Resolution on Classification Results

Training, test, and validation sets were balanced so that the number of authentic and spoofed samples was equal. Consequently, standard classification accuracy was used as the primary performance metric. Figure 4.6 shows the test-set accuracy (10% of the whole data were held out and never exposed to the network during training) for three training scenarios.

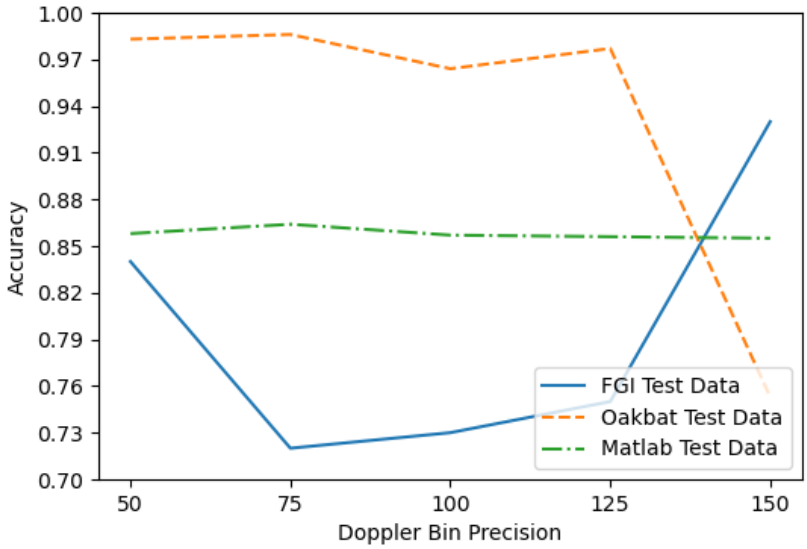


Figure 4.6: A line plot of the accuracy on the test data for varying Doppler bin precision for each scenario.

- **Scenario 1 - MATLAB Simulated Data**
  The network was trained exclusively on simulated data in order to quantify the gap between simulation and real-world recordings. The dataset consisted of 24,000 acquisitions. Accuracy remained almost constant across all Doppler bin precision, with a slight increase at 75 Hz Bin width.
- **Scenario 2 - Fine-tuning on OAKBAT recordings (os2, os4, os6)**
  A MATLAB-pre-trained model was fine-tuned on a domain-specific subset consisting of 7200 acquisitions. Test set accuracy was lowest for a 150 Hz Doppler bin and highest for 75 Hz.
- **Scenario 3 - Fine-tuning on FGI recordings (MCD, FGS, UTD)**
  A MATLAB-pre-trained model was fine-tuned on FGI data consisting of 7200 acquisitions. In this case, the best accuracy was obtained with a coarser 150 Hz Doppler bin.

Overall, the results indicate that the optimal Doppler bin precision depends on the source domain of the fine-tuning data, where models trained solely on simulations are relatively insensitive to this parameter. The best achieved Doppler precision for a specific scenario will be used when examining the remaining results.

4.2.1 MATLAB-Simulated Data - Cross-Domain Generalization

In this experiment, the CNN was trained exclusively on MATLAB-simulated acquisitions to quantify the performance gap between synthetic and real-world data. The simulated recordings were generated with the same sampling rate (5 MHz) and I/Q bit width (16-bit) as the OAKBAT recordings, whereas the FGI recordings used a 26 MHz sampling rate and only an 8-bit (I) format. Thereby, FGI recordings had to be resampled to 5 MHz to be compatible with the trained network.

The generalization results, shown in Table 4.1, showed a strong dependence on the recording configuration. For the FGI recordings (TGS and TGD), which are similar to spoofing attacks present in OAKBAT, the model achieved near-random performance. The meaconing recording (MCD) performed even worse than random guessing, indicating a complete failure of generalization to the FGI domain. A timeline plot of the model's assigned probability of each acquisition is included in the Appendix as Figure A.1.

On the other hand, when the model was evaluated on OAKBAT recordings that it had never seen, it showed non-random behaviour, thereby providing a proof-of-concept of cross-domain transfer. On the CleanStatic recording, the model classified 72 % of the frames as authentic. The best performance was obtained on os2 (spoofing advantage of 10 dB-Hz, where only a single dominant peak existed). A timeline plot is also provided for the os2 recording in the Appendix as Figure A.2. The most challenging case was os4 (spoofing advantage of 0.4 dB-Hz). Note that the smallest spoofing advantage present in the synthetic training set was 1-2 dB-Hz, and thereby, the model's ability to detect weaker spoofing attacks on OAKBAT suggests a degree of successful generalization.

Scenario	#Authentic	#Spoofed	Accuracy	Balanced Accuracy	MCC
FGI [19]					
TGS	588	2565	0.83	0.54	0.26
TGD	660	2497	0.81	0.54	0.26
UTD	603	806	0.58	0.51	0.09
MCD	802	1364	0.49	0.41	-0.23
OAKBAT [33]					
CleanStatic	5033	0	0.72	-	-
os2	1284	4641	0.95	0.87	0.84
os3	1212	3300	0.92	0.86	0.79
os4	1204	2985	0.81	0.79	0.56
os5	1215	4641	0.95	0.87	0.83
os6	1222	3122	0.82	0.80	0.58

Table 4.1: Results obtained after training the CNN model on data simulated using MATLAB. All evaluations utilized a Doppler precision of 75 Hz.

4.2.2 Fine-tuning on OAKBAT recordings (os2, os4, os6)

The CNN that had been trained exclusively on MATLAB-simulated data (Doppler bin precision of 75 Hz) was subsequently fine-tuned using a subset of OAKBAT recordings. Three recordings (os2, os4, os6) were selected to assess the model's ability to generalise to previously unseen data from the same dataset. From the first few seconds of the authentic and spoofed segments, a total of 7200 acquisitions were extracted.

The performance of the fine-tuned model is summarised in Table 4.2. Results from OAKBAT fine-tuning show that the CleanStatic recording achieved an 89 % overall accuracy for labelling authentic-as-authentic, reducing but not eliminating false alarms. The weakest case was that of os4, with a balanced accuracy of 0.96 and an MCC of 0.91. The highest performance recording result was that of os5 (spoofing advantage of 9.9 dB-Hz). This recording had not been used during fine-tuning, indicating that the model can recognize stronger spoofing attacks.

In contrast, fine-tuning on OAKBAT data did not improve performance on the FGI recordings. The results remained close to random guessing, and for the meaconing case (MCD), performance was even below chance. This suggests that the domain gap is not solely between synthetic and real-world data, but also involves differences in dataset configurations.

Scenario	#Authentic	#Spoofed	Accuracy	Balanced Accuracy	MCC
FGI [19]					
TGS	588	2565	0.82	0.52	0.17
TGD	660	2497	0.80	0.52	0.15
UTD	603	806	0.58	0.50	0.07
MCD	802	1364	0.48	0.38	-0.32
OAKBAT [33]					
CleanStatic	5033	0	0.89	-	-
os2 *	1284	4641	0.99	0.98	0.97
os3	1212	3300	0.98	0.97	0.96
os4 *	1204	2985	0.96	0.96	0.91
os5	1215	4641	0.99	0.99	0.97
os6 *	1222	3122	0.97	0.96	0.92

Table 4.2: Results obtained after fine-tuning the CNN model originally trained on MATLAB-simulated data using OAKBAT datasets. All evaluations utilized a Doppler precision of 75 Hz. * The model was trained and evaluated on samples drawn from the same recording.

4.2.3 Fine-tuning on FGI recordings (MCD, FGS, UTD)

The best test-set result for the FGI data (Figure 4.6) was obtained with a Doppler bin precision of 150 Hz. The three recordings of MCD, TGS, and UTD were selected out of the available FGI recordings since they covered a wide variety of spoofing attacks. The TGD recording was left out for validation purposes since it contains a similar attack to the spoofing attack in TGS. As with the OAKBAT experiment, 7200 acquisitions were extracted from the initial seconds of the authentic and spoofed segments of each recording, and the class labels were kept balanced.

The fine-tuned model showed a marked improvement over the model trained on MATLAB data. It no longer behaved as a random guesser but classified the acquisition maps correctly in the majority of cases. The FGI UTD recording yielded the highest performance with a balanced accuracy of 0.96 and an MCC of 0.93. This recording also contained the fewest acquired samples. The meaconing scenario remained the most difficult case, achieving an MCC of 0.62. Figure A.5 in the appendix displays the heatmap timeline for the MCD recording. The model's output probability hovers around 50 % during the onset of spoofing, indicating uncertainty. The previously unseen TGD recording is visualized in Figure A.4, also in the appendix.

Fine-tuning on FGI data, however, degraded the model's performance on the OAKBAT recordings. All OAKBAT test sets fell to or below random guessing levels after the FGI adaptation. All Timeline plots for every scenario are provided in the public GitHub repository [36], which offers further insight into the cases not discussed in detail here.

Scenario	#Authentic	#Spoofed	Accuracy	Balanced Accuracy	MCC
FGI [19]					
TGS *	588	2565	0.95	0.90	0.86
TGD	660	2497	0.86	0.92	0.89
UTD *	603	806	0.93	0.96	0.93
MCD *	802	1364	0.73	0.83	0.62
OAKBAT [33]					
CleanStatic	5033	0	0.19	-	-
os2	1284	4641	0.59	0.24	-0.43
os3	1212	3300	0.55	0.33	-0.35
os4	1204	2985	0.59	0.36	-0.31
os5	1215	4641	0.58	0.22	-0.46
os6	1222	3122	0.59	0.35	-0.33

Table 4.3: Results obtained after fine-tuning the CNN model originally trained on MATLAB-simulated data using the FGI dataset. All evaluations utilized a Doppler precision of 150 Hz. * The model was trained and evaluated on samples drawn from the same recording.

4.3 Inference Time

The processing time of the data pipeline were measured for each tested Doppler bin precision and is summarized in Table 4.4 and Figure 4.7. The acquisition stage required roughly 19 ms independently of the Doppler resolution, while the preprocessing step added about 2.7 ms. The CNN processing forward pass contributed approximately 1 ms. Consequently, the total inference time per satellite varied only marginally across the tested Doppler bin widths, ranging from 22.26 ms (75 Hz) to 22.67 ms (150 Hz). The reported uncertainty corresponds to one standard deviation across all measurements from the OAKBAT and FGI datasets for a given Doppler bin precision.

Doppler (Hz)	**Acquisition (ms)**	**Preprocessing (ms)**	**CNN (ms)**	**Total (ms)**
50	18.93 ± 1.26	2.68 ± 0.13	0.98 ± 0.14	22.59 ± 1.53
75	18.66 ± 0.87	2.63 ± 0.08	0.97 ± 0.12	22.26 ± 1.07
100	18.85 ± 0.68	2.71 ± 0.06	0.99 ± 0.17	22.35 ± 0.91
125	18.95 ± 0.79	2.72 ± 0.09	1.00 ± 0.16	22.67 ± 1.04
150	18.96 ± 0.71	2.71 ±0.07	1.00 ± 0.17	22.67 ± 0.95

Table 4.4: All measurements refer to the processing time per individual satellite. Aggregated inference-time measurements across all evaluated Doppler bin precisions, averaged over all GPS recordings from OAKBAT and FGI. The reported uncertainty (±) corresponds to one standard deviation of the timing measurements.

The total inference time across the tested Doppler bin precisions differed only slightly, by at most a few milliseconds. Notably, finer Doppler bin resolutions did not result in a notable increase in total inference time. This behaviour is most likely explained by the fixed computational cost of performing Fast Fourier Transformations (FFT) during the acquisition stage, as well as the underlying Python library implementation, which utilizes dual-core processing capabilities [37].

In addition, there is an uncertainty in the timing measurements, which is reflected in the larger variability observed for the smallest Doppler bin precision. This may be explained by the execution environment, which was managed by SLURM and involved shared computational resources distributed among multiple users. Thereby, the inference-time results are primarily relevant in demonstrating that smaller Doppler bin precision does not necessarily lead to longer inference time.

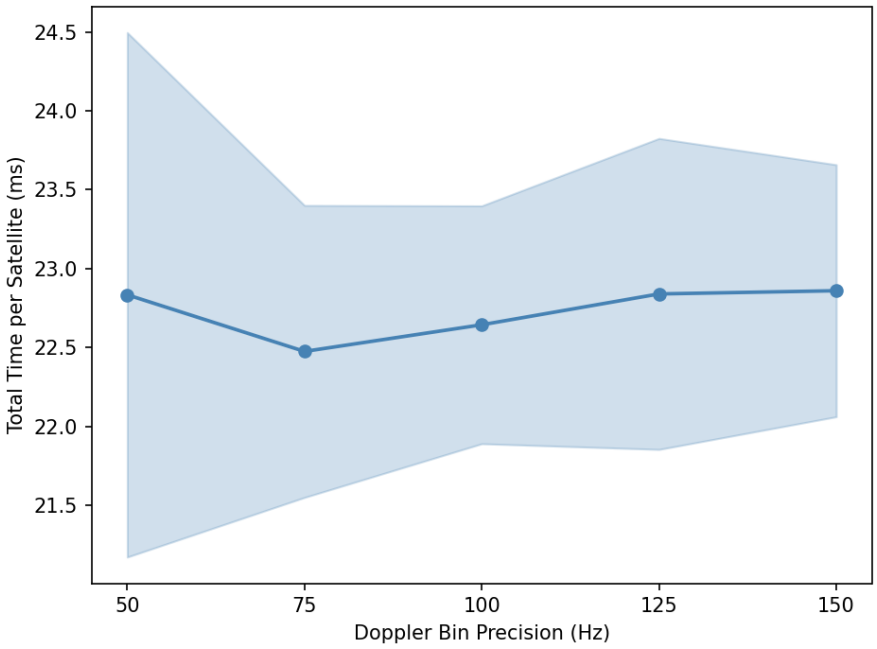


Figure 4.7: Inference time and uncertainty for each PRN ID processing. Total processing time (ms), including satellite ID acquisition, the duration of preprocessing from raw samples to global and ROI maps, and the inference time of the CNN model, evaluated as a function of Doppler bin resolution (Hz). The reported uncertainty (±) corresponds to one standard deviation of the timing measurements.

5
Discussion

The results presented in this work raise several points worth discussing in greater depth. The following discussion addresses four main aspects: the ability of the model to generalize across different datasets and recording conditions, a reflection on what features the CNN may be looking at when classifying signals, the feasibility of spoofing mitigation using the proposed approach, and finally, the ambiguity surrounding how spoofing itself is defined. Together, these points aim to contextualize the results and identify directions for future work.

5.1 Cross-Domain Generalization

Models that initially were trained on the MATLAB-simulated recordings demonstrated generalization to the OAKBAT domain but not to the FGI domain, and this conversely held for the models fine-tuned on FGI data, making them incompatible with the remaining datasets. This distinction between datasets is therefore clearly shown in the results. The generalization from simulated to OAKBAT was expected, given that both share a sampling rate of 5 MHz and a bit width of 16, thereby minimizing the differences between the domains. In contrast, the FGI recordings, consisting of other parameters, required resampling to 5 MHz before performing any acquisition.

A specific factor that may have contributed to the generalization failure toward the FGI domain is the resampling procedure applied to the FGI recordings. Since the FGI data were recorded at different sampling rates. This process may have introduced resampling artifacts, alternating the signal characteristics in ways that were not present in either the simulated data or the OAKBAT recordings. If such artifacts were introduced, the model trained on simulated data would not have been exposed to these distortions during training, potentially explaining why generalization to FGI failed.

Furthermore, a likely source of domain discrepancy is the difference in carrier-to-noise ratio between simulated and real recordings. Simulated data tends to produce cleaner signals, with fewer unpredictable noise sources, whereas real recording captures environmental and hardware noise. The difference in C/N_0 characteristics may have made the simulated data insufficiently representative of real-world conditions, further limiting the mode's ability to generalize across domains.

Identifying the most important factor of cross-domain generalization remains unanswered, as a multitude of variables are present. One further contributing factor may be that of hardware configuration utilized: the OAKBAT scenario relies on simulation hardware for both the live-sky and the spoofed signals, but the FIG repository used real-live sky signals mixed with simulated spoofing [19, 33]. Given that FGI represents the more realistic recording conditions while OAKBAT offers reproducibility, the results suggest that the transition from simulated to real-world data is the most critical domain gap. This implies that, for finding reported in review articles such as [10] to hold broader significance, cross-domain generalization capability should be weighted more than accuracy metrics on a specific dataset.

Beyond providing real-sky signals, the FGI repository also introduces more spoofing variations. However, more spoofing variations come at the cost of a reduced range of spoofing advantage values, time-offsets, and the absence of dynamic receiver scenarios, representing a different set of limitations compared to OAKBAT [19, 33]. Furthermore, fine-tuning experiments conducted on FGI data were inherently constrained by these factors. Nevertheless, fine-tuning yielded promising results, suggesting that the cross-domain generalization gap is bridgeable. The most fundamental outstanding challenge, however, remains the lack of publicly available repositories containing non-simulated spoofing signals.

5.2 Interpretation of Model Predictions

That GPS spoofing can result in a dual peak scenario at the acquisition stage is an established observation [10]. This is further supported by the results presented in the previous section, where Figure 4.4 demonstrates that the MATLAB simulation successfully reproduces this scenario, and Figure 4.3 shows it occurring in the "real-world" FGI UTD recordings. Dual peak scenarios, therefore, represent a strong indicator of spoofing for the automatic feature extraction performed by the CNN. However, a dual peak is not always present. When the spoofing power advantage becomes sufficiently large, the authentic signal is buried beneath the noise floor, as illustrated in the OAKBAT os2 acquisition map in Figure 4.3, where a 10 dB spoofing power advantage is applied.

Despite this, the model successfully detects spoofing even when the authentic peak is suppressed. Several explanations are possible for what the CNN may be looking for in this case. One possibility is that a strong spoofing advantage suppresses the background noise floor, resulting in an unusually clean noise floor that the CNN learned to associate with spoofing. Another possibility relates to peak characteristics: high-power spoofing may produce a peak that is fundamentally different in shape compared to an authentic signal. As seen in the OAKBAT os2 example in Figure 4.3, the spoofed peak appears tall and narrow, whereas authentic peaks tend to be lower and more rounded. Such differences in peak shape are well-suited for a CNN to detect.

Considering both cases together, there may exist a spectrum of detection cues avail-

able to the CNN depending on the spoofing power advantage. At lower spoofing advantages, dual peaks remain visible and serve as the primary detection cue. As the spoofing advances, increases, and the authentic peaks become buried, the CNN may instead rely on secondary cues such as noise floor suppression or peak shape characteristics. This raises the question of whether a middle ground exists where neither cue is present, potentially representing the most challenging scenario for detection.

It should also be noted that the CNN developed in this work does not rely solely on a single branch. In addition to the local branch, which examines a region of interest around the peak in fine detail, the global branch is included to capture the full acquisition map. The global branch was introduced specifically to detect dual peaks that fall outside the region of interest, as may occur when the time-offset injection is large. In the global view, what CNN looks at is likely more straightforward, since the map consists predominantly of noise with only true signal peaks standing out. This branch was designed with untargeted spoofing in mind, where the spoofed signals may not be aligned with authentic signals. However, if the spoofed signals' current constellation does not overlap with any visible satellite, no dual peak scenario arises. The authentic signal would appear with only its own peak, and the spoofed signal would similarly produce only a single peak in its own receiver channel. Under these conditions, detection may be missed, as neither dial peak nor a strong power advantage is present to serve as a cue.

A natural extension of this analysis would be the use of explainability tools, such as gradient-weighted class activation mapping, to more rigorously identify which regions of the acquisition map the CNN responds to [38]. This was not done in the present work due to scope limitations, but it represents a promising direction for future research.

5.3 On the Feasibility of Spoofing Mitigation

The focus of this work, as with much of existing literature, is on spoofing detection rather than mitigation [10], and this is a reasonable prioritization given the inherent challenges involved. As previously discussed, when the spoofing power advantage is sufficiently large, the authentic signal becomes buried in the noise floor, leaving only the spoofed signal recoverable. In principle, if two peaks are visible in the acquisition map, mitigation could be attempted by tracking the lower-power signal, which could correspond to the authentic one. However, this is far from straightforward.

The CNN developed in this work classifies signals as spoofed or authentic, but does not explicitly determine whether two peaks are present, since the methodology demonstrated that spoofing can be detected even in the absence of a dual peak scenario. If dual peak detection were pursued, a separate algorithm would be needed to locate the second peak, as the second high correlation magnitude in an acquisition map does not necessarily correspond to the second distinct peak - it may simply be the second highest point of the dominant peak. In this sense, what is trivial

for a human observer to identify visually is considerably more difficult to formalize algorithmically.

Further complications arise when considering the case where two peaks overlap, making peak separation impossible. Additionally, in untargeted spoofing scenarios, the spoofed signals may not correspond to any satellite in the currently visible constellation, meaning no dual-peak scenario arises at all. Taken together, these factors suggest that spoofing mitigation using acquisition stage maps and a CNN-based classification poses significant unsolved challenges, and whether true mitigation is achievable through this approach remains an open question.

5.4 Ambiguity in Spoofing Types - Which Scenarios Are Intended to Be Detected?

In the scope of this work, the spoofing scenario of interest is implicitly *targeted spoofing*, i.e., the attacker transmits signals that mimic the current satellite constellation so that the tracked PRN identifiers are identical to those of the authentic satellites. This description is deliberately broad and therefore ambiguous since it encompasses a variety of attacks ranging from strong spoofer advantages that introduce temporal and Doppler offsets to simple meaconing attacks with short rebroadcast delays and a transmitter located close to a receiver. Consequently, the receiver may observe either two distinct correlation peaks or a single dominant peak for each tracked PRN.

The ambiguity becomes evident because the proposed method was also evaluated on untargeted spoofing and on a meaconing scenario from the FGI dataset, as well as on a dual-frequency (TGD) dataset in which the L5 channel was never considered. The core strength of the approach lies in its ability to exploit the presence of the two peaks, yet it also demonstrated a degree of robustness when only a single peak was present. After modest fine-tuning on the FGI data, the method got promising results, suggesting that a detection technique designed for a specific spoofing class should be tested under alternative attack models, since related scenarios do not behave identically.

One point is that GPS spoofing should not be treated as a homogeneous concept. The recent 2024 review by [10] highlights this problem of treating spoofing as a homogenous concept and underscores the difficulty of comparing methods with a clear taxonomy. Establishing a more explicit classification of spoofing types with the GNSS-spoofing literature would help research find the intended detection domain, and then, to avoid redundant research, and to allow for meaningful comparisons.

6
Conclusion

The acquisition of GPS signals with a GPS receiver is performed using a 2-D search that yields an acquisition map indicating code-phase and Doppler offsets. This map can be split into a region-of-interest (ROI) segment and a global segment, enabling the detection of several spoofing attacks. The resulting one-dimensional images can be fed to a CNN that automatically extracts features and classifies the acquisition map as authentic or spoofed.

The performance of the CNN depends on the recordings used during training, where cross-domain generalization was observed to be limited. Simulated recordings, when generated with parameters matching those of the target dataset, can act as a useful springboard because they provide a variety of spoofing configurations. When CNN models are fine-tuned on data from either OAKBAT or FGI, the results within the same domain improve significantly, achieving balanced classification accuracy up to 99%. However, cross-domain spoofing detection between OAKBAT and FGI does not succeed. The exact features learned by the network remain unsure, but it is reasonable to assume that the model considers the noise floor, the number and shape of correlation peaks, and the overall peak structure. Furthermore, the network was capable of recognizing varying spoofer power advantages.

Bibliography

[1] European Union Agency for the Space Programme. What is gnss. Accessed 2026-05-20. [Online]. Available: https://www.euspa.europa.eu/eu-space-programme/galileo/what-gnss

[2] E. D. Kaplan and C. J. Hegarty, *Understanding GPS/GNSS - Principles and Applications (3rd Edition)*. Artech House, 2017. [Online]. Available: https://app.knovel.com/hotlink/toc/id:kpUGPSGNS4/understanding-gps-gnss/understanding-gps-gnss

[3] P. K. Enge, "The global positioning system: Signals, measurements, and performance," *International Journal of Wireless Information Networks*, vol. 1, no. 2, pp. 83–105, Apr. 1994. [Online]. Available: https://doi.org/10.1007/BF02106512

[4] D. Hoey and P. Benshoof, "Civil gps systems and potential vulnerabilities," *Proceedings of the 18th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2005)*, p. 6, 10 2005.

[5] M. Cuntz, A. Konovaltsev, A. Dreher, and M. Meurer, "Jamming and spoofing in gps/gnss based applications and services – threats and countermeasures," in *Future Security*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 196–199.

[6] F. Asplund. (2025) Gps-störningar längs kusten - ambulanser drabbas varje dag. Accessed 2026-01-19. [Online]. Available: https://www.sverigesradio.se/artikel/gps-storningar-langs-kusten-ambulanser-drabbas-varje-dag

[7] A. Brantemo and E. Foric. (2025) Mattias drabbades av gps-störningar – "befann sig" plötsligt utanför kaliningrad. Accessed 2026-05-05. [Online]. Available: https://www.svt.se/nyheter/lokalt/ost/mattias-drabbades-av-gps-storningar-befann-sig-plotsligt-utanfor-kaliningrad

[8] M. L. Psiaki and T. E. Humphreys, "Gnss spoofing and detection," *Proceedings of the IEEE*, vol. 104, no. 6, pp. 1258–1270, 2016.

[9] GPSPATRON. (2025) Gnss interference in the baltic sea: A collaborative study by gpspatron and gdynia maritime university. Accessed 2026-04-07. [Online]. Available: https://gpspatron.com/gnss-interference-in-the-baltic-sea-a-collaborative-study-by-gpspatron-and-gdynia-maritime-university

[10] K. Radoš, M. Brkić, and D. Begušić, "Recent advances on jamming and spoofing detection in gnss," *Sensors*, vol. 24, no. 13, 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/13/4210

[11] Z. Chen, J. Li, J. Li, X. Zhu, and C. Li, "Gnss multiparameter spoofing detection method based on support vector machine," *IEEE Sensors Journal*, vol. 22, no. 18, pp. 17 864–17 874, 2022.

[12] K. Radoš, M. Brkić, and D. Begušić, "Gnss signal classification based on machine learning methods," in *2024 47th MIPRO ICT and Electronics Convention (MIPRO)*, 2024, pp. 806–811.

[13] D. R. Kartchner, R. Palmer, and S. K. Jayaweera, "Satellite navigation anti-spoofing using deep learning on a receiver network," in *2021 IEEE Cognitive Communications for Aerospace Applications Workshop (CCAAW)*, 2021, pp. 1–5.

[14] G. Aissou, S. Benouadah, H. El Alami, and N. Kaabouch, "Instance-based supervised machine learning models for detecting gps spoofing attacks on uas," in *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, 2022, pp. 0208–0214.

[15] E. Shafiee, M. R. Mosavi, and M. Moazedi, "Detection of spoofing attack using machine learning based on multi-layer neural network in single-frequency gps receivers," *Journal of Navigation*, vol. 71, no. 1, p. 169–188, 2018.

[16] P. Borhani-Darian, H. Li, P. Wu, and P. Closas, "Detecting gnss spoofing using deep learning," *EURASIP Journal on Advances in Signal Processing*, vol. 2024, no. 1, Jan. 2024. [Online]. Available: http://dx.doi.org/10.1186/s13634-023-01103-1

[17] G. Marchand, A. Toumi, G. Seco-Granados, and J. A. López-Salcedo, "Machine learning assessment of anti-spoofing techniques for gnss receivers," in *Proceedings of the Work-in-Progress in Hardware and Software for Location Computation (WIPHAL)*. Castellón, Spain: CEUR Workshop Proceedings, Jun. 2023, work-in-Progress paper.

[18] S. Semanjski, I. Semanjski, W. De Wilde, and A. Muls, "Use of supervised machine learning for gnss signal spoofing detection with validation on real-world meaconing and spoofing data—part i," *Sensors*, vol. 20, no. 4, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/4/1171

[19] S. Islam, M. Z. H. Bhuiyan, M. Liaquat, I. Pääkkönen, and S. Kaasalainen, "An open gnss spoofing data repository: characterization and impact analysis with fgi-gsrx open-source software-defined receiver," *GPS Solutions*, vol. 28, no. 4, p. 176, 2024. [Online]. Available: https://libkey.io/10.1007/s10291-024-01719-2

[20] Y.-H. Sung, S.-J. Park, D.-Y. Kim, and S. Kim, "Gps spoofing detection method for small uavs using 1d convolution neural network," *Sensors*, vol. 22, no. 23, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/23/9412

[21] J. Li, X. Zhu, M. Ouyang, W. Li, Z. Chen, and Z. Dai, "Research on multi-peak detection of small delay spoofing signal," *IEEE Access*, vol. 8, pp. 151 777–151 787, 2020.

[22] European Space Agency. (2026) Gps navigation message. Accessed 2026-03-17. [Online]. Available: https://gssc.esa.int/navipedia/index.php?title=GPS_Navigation_Message

[23] M. Braasch and A. van Dierendonck, "Gps receiver architectures and measurements," *Proceedings of the IEEE*, vol. 87, no. 1, pp. 48–64, 1999.

[24] European Space Agency. (2014) Tracking loops. Accessed 2026-04-23. [Online]. Available: https://gssc.esa.int/navipedia/index.php/Tracking_Loops

[25] F. Chollet, *Deep Learning with Python*, 1st ed. Shelter Island, NY: Manning Publications, 2017.

[26] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *Neurocomputing*, vol. 503, pp. 92–108, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231222008426

[27] M. Elgendy, *Deep Learning for Vision Systems*, 1st ed. Shelter Island, NY: Manning Publications, 2020.

[28] PyTorch. torch.nn. Accessed 2026-05-13. [Online]. Available: https://docs.pytorch.org/docs/2.11/nn.html

[29] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[30] Navigation Center, United States Coast Guard. GPS NANUS, Almanacs, OPS Advisories, & SOF. Accessed 2026-02-24. [Online]. Available: https://www.navcen.uscg.gov/gps-nanus-almanacs-opsadvisories-sof

[31] Mathworks. gpsWaveformGenerator - Generate GPS waveform (legacy L1 and L2, modernized L1C, L2C, and L5). Accessed 2026-02-24. [Online]. Available: https://se.mathworks.com/help/satcom/ref/gpswaveformgenerator-system-object.html

[32] Mathworks. (2026) Gnss signal transmission using software-defined radio. Accessed 2026-02-19. [Online]. Available: https://se.mathworks.com/help/satcom/ug/gnss-signal-transmission-using-sdr.html

[33] A. Albright, S. Powers, J. Bonior, and F. Combs, "A tool for furthering gnss security research: The oak ridge spoofing and interference test battery (oakbat)," *Proceedings of the 33rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2020)*, pp. 3697–3712, 9 2020.

[34] SLURM. Slurm - overview. Accessed 2026-05-13. [Online]. Available: https://slurm.schedmd.com/overview.html

[35] D. Chicco, N. Tötsch, and G. Jurman, "The matthews correlation coefficient (mcc) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation," *BioData Mining*, vol. 14, 02 2021.

[36] B. Pettersson. (2026) All master thesis results. Accessed 2026-04-23. [Online]. Available: https://github.com/BenjaminTWP/ALL-Master-Thesis-Results

[37] Berkly Statistics. Parallel processing in python. Accessed 2026-05-13. [Online]. Available: https://computing.stat.berkeley.edu/tutorial-parallelization/parallel-python.html#basic-parallelized-loopsmapsapply-using-ipyparallel

[38] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," *International Journal of Computer Vision*, vol. 128, no. 2, p. 336–359, Oct. 2019. [Online]. Available: http://dx.doi.org/10.1007/s11263-019-01228-7

[39] The United Nations. The 17 goals. Accessed 2026-05-18. [Online]. Available: https://sdgs.un.org/goals

A

Appendix 1: Heatmap Timelines

The following appendix holds the heatmap timelines of a subset plots chosen to illustrate interesting cases and underscore findings. All heatmap plots can be found in the following GitHub repository [36].

The timeline plots shows the tracking, not ordinary GPS receiver tracking, but the acquisition search carried out once every second for all the PRN IDs present in the recording. The heatmap essentially shows the CNN model's classification confidence, where the more sure it is about spoofing, the more red the classification, where it is unsure, the classification is yellow, and where it is authentic, the acquisition is green.

A.1 CNN Model Trained on MATLAB Simulations

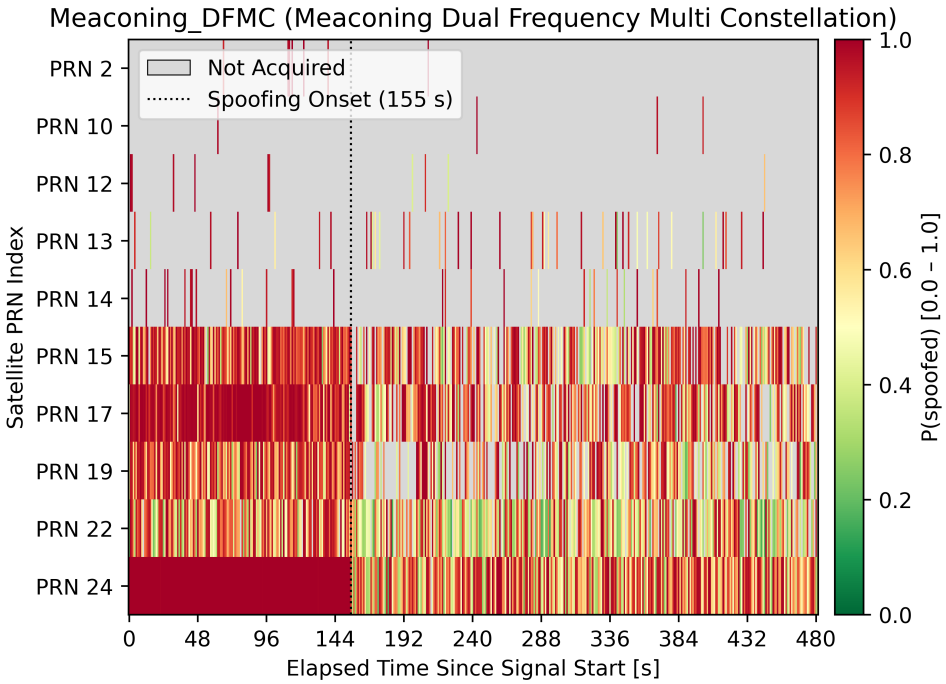


Figure A.1: Classification confidence heatmap of the CNN trained exclusively on MATLAB data and evaluated on the FGI MCD recording. Acquisition performed once per second for all satellite PRN IDs. Non-acquired signal marked as grey. Spoofing introduced at 155 seconds. Illustrating the overall confidence of the CNN model during tracking by performing acquisition searches.

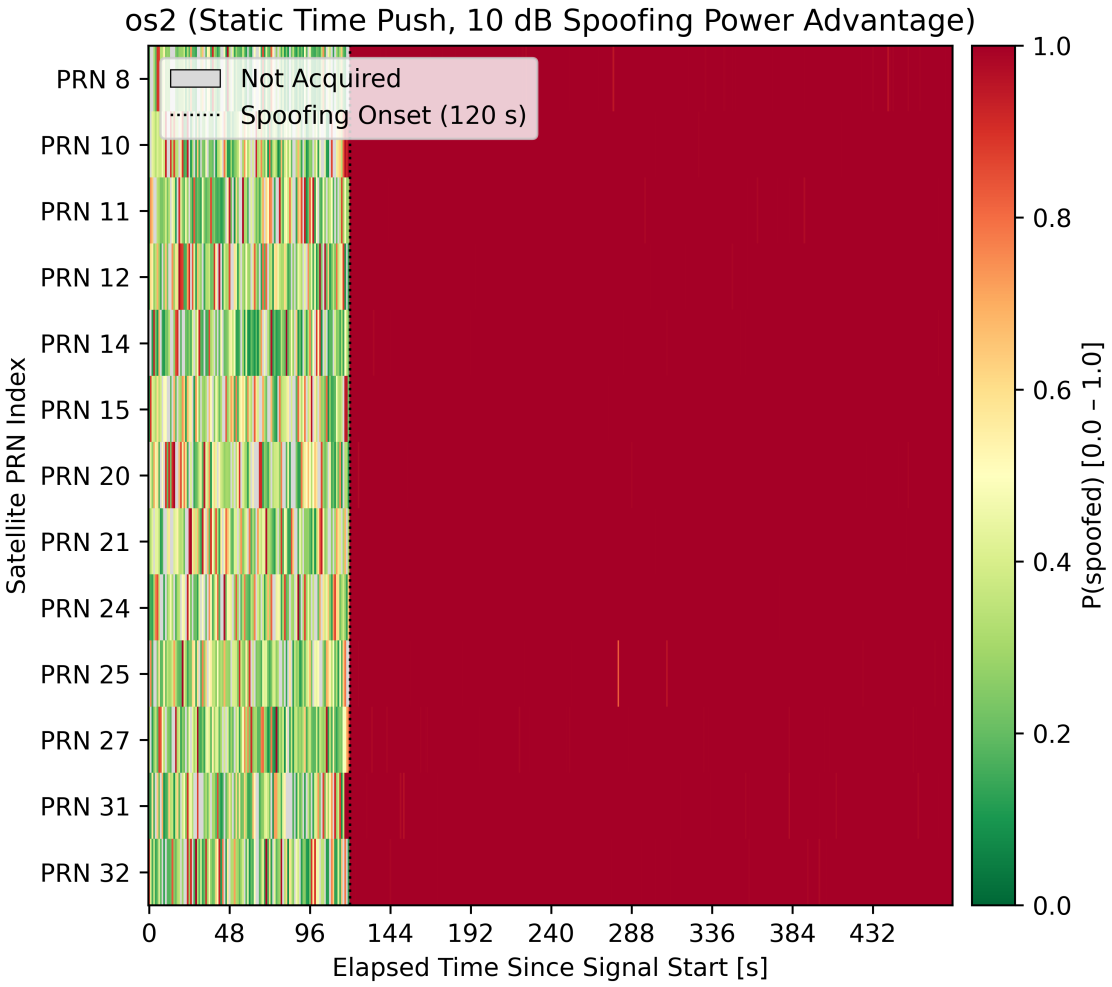


Figure A.2: Classification confidence heatmap of the CNN trained exclusively on MATLAB data and evaluated on the Oakbat os2 recording. Acquisition performed once per second for all satellite PRN IDs. Non-acquired signal marked as grey. Spoofing introduced at 120 seconds. Illustrating the overall confidence of the CNN model during tracking by performing acquisition searches.

A.2 Fine-tuning on Oakbat recordings (os2, os4, os6)

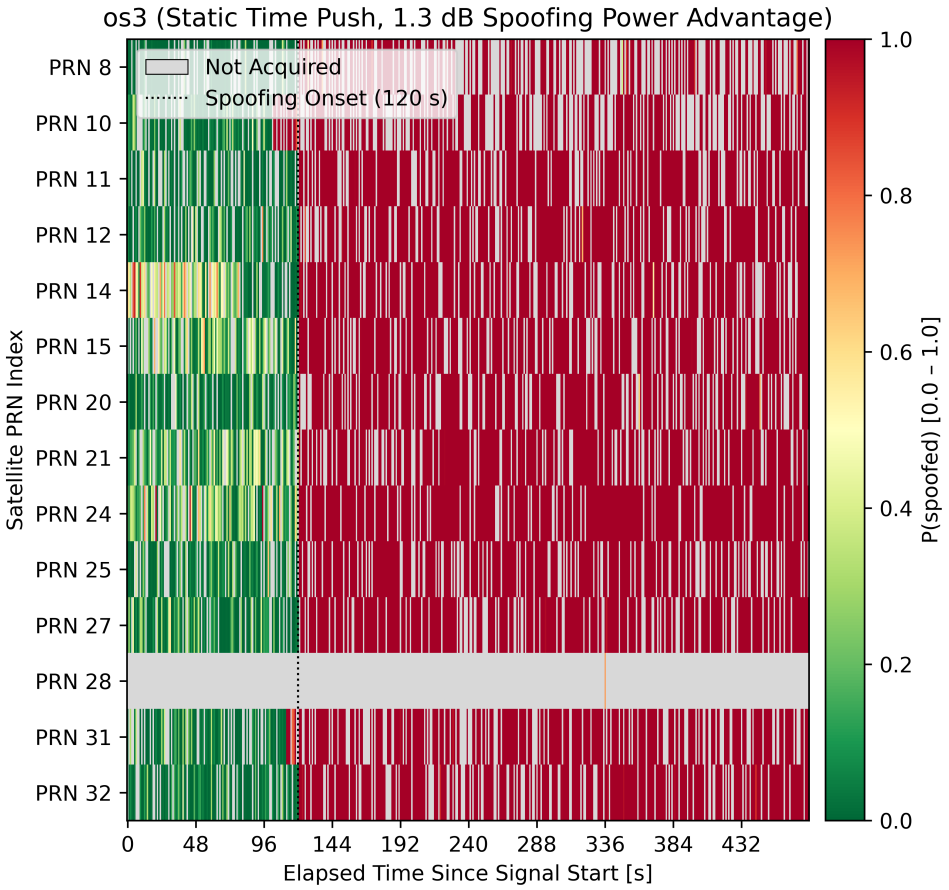


Figure A.3: Classification confidence heatmap of the CNN fine-tuned on Oakbat data and evaluated on the Oakbat os3 recording. Acquisition performed once per second for all satellite PRN IDs. Non-acquired signal marked as grey. Spoofing introduced at 120 seconds. Illustrating the overall confidence of the CNN model during tracking by performing acquisition searches.

A.3 Fine-tuning on FGI recordings (MCD, FGS, UTD)

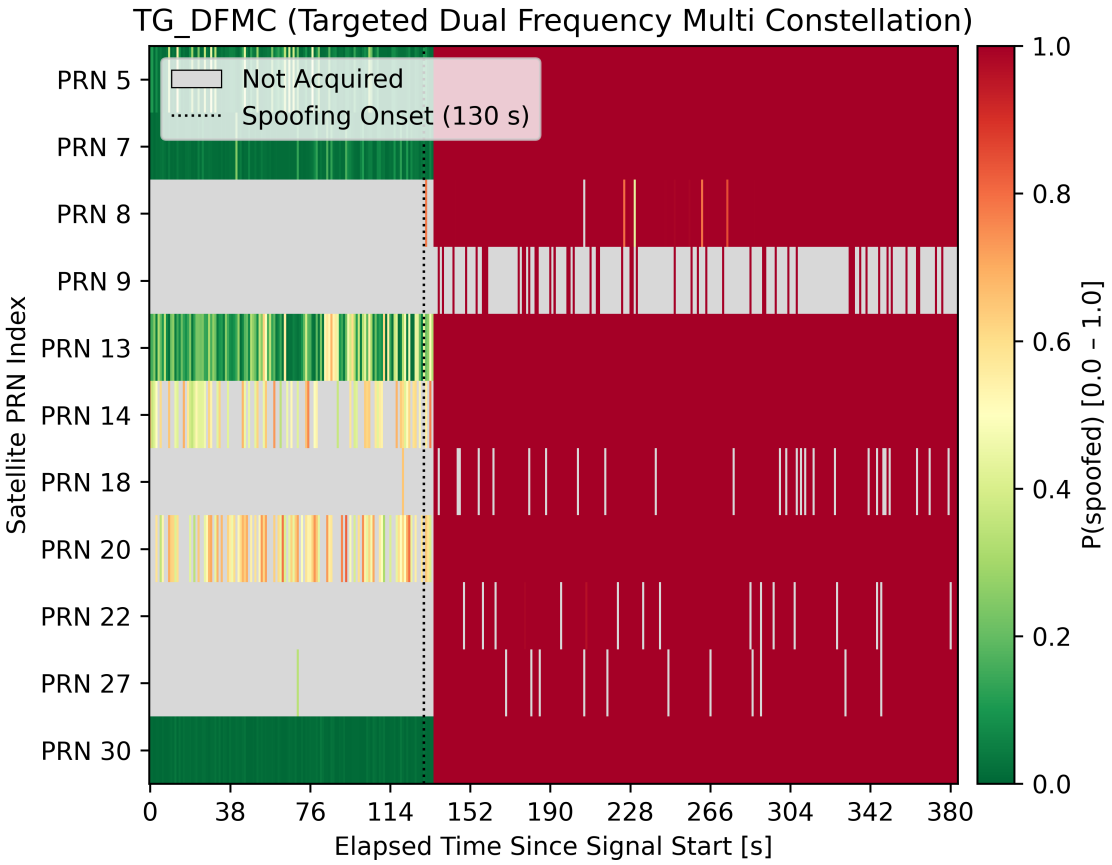


Figure A.4: Classification confidence heatmap of the CNN fine-tuned on FGI data and evaluated on the FGI TGD recording. Acquisition performed once per second for all satellite PRN IDs. Non-acquired signal marked as grey. Spoofing introduced at 130 seconds. Illustrating the overall confidence of the CNN model during tracking by performing acquisition searches.

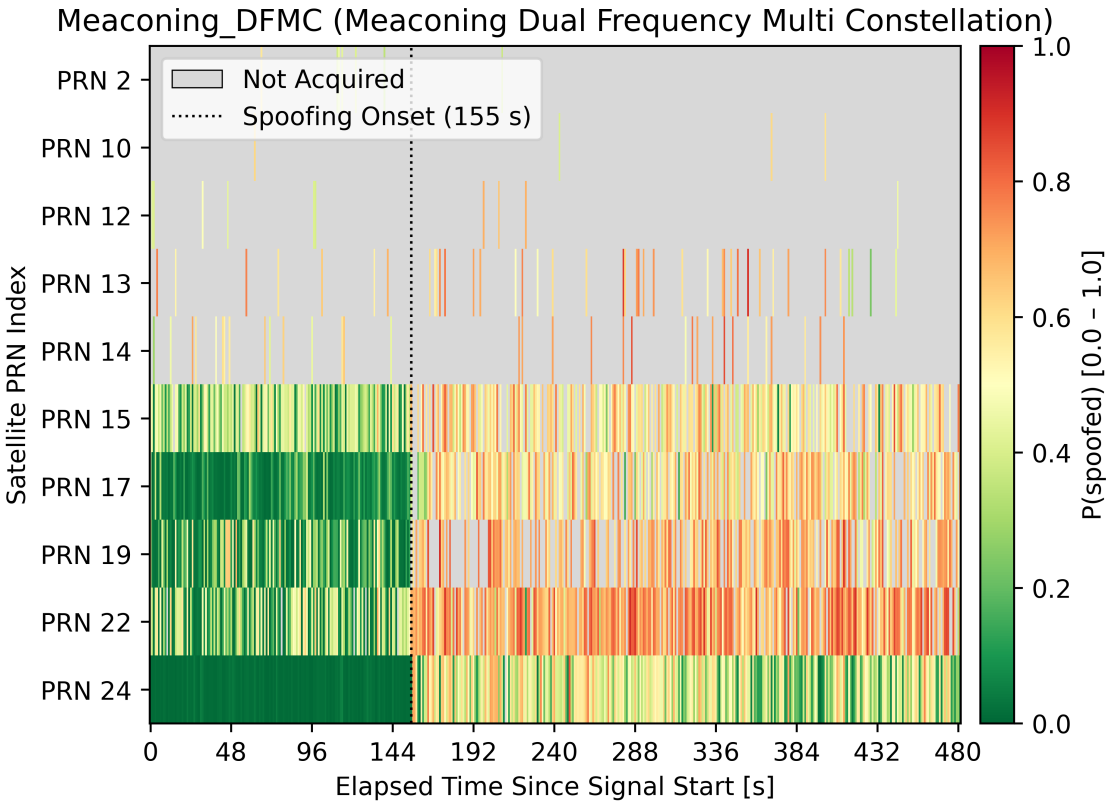


Figure A.5: Classification confidence heatmap of the CNN fine-tuned on FGI data and evaluated on the FGI MCD recording. Acquisition performed once per second for all satellite PRN IDs. Non-acquired signal marked as grey. Spoofing introduced at 155 seconds. Illustrating the overall confidence of the CNN model during tracking by performing acquisition searches.

B

Appendix 2: The Use of Generative AI

Following is an acknowledgement of the use of generative AI, such as large language models, within this project. Generative AI were used to refine text by asking for possible improvements, following is a prompt used.

"""""""""""""

I am writing the [NAME_OF_SECTION] section of my master's thesis. I am a student at a data science and AI master's program, and I am writing my thesis in the faculty of "space, earth, and observatory". My master's thesis concerns spoofing detection using a convolutional neural network at the acquisition stage of a GPS receiver.

Since I am writing by myself, I have no one to proofread and help me discuss phrasing and vocabulary used in the text, thus, I would like you to take on this role. I will provide paragraphs of my writing, and I would like you to suggest alternatives that can help me keep a scientific style and adhere to the correct tense as expected in a [NAME_OF_SECTION] section.

Note that I do not want to lose the structure of what I have been writing and how I have written it. Thereby, help me keep tense and suggest improvements for the paragraphs I provide. Keep answers short, and reply with improvements/suggestions. The paragraph to first examine is:

\section {Results}
Global and ROI acquisition maps were first plotted to get a visualization, a selection of these were chosen to be presented. Both authentic and spoofed acquisitions were extracted and presented. The trained CNN model performance were evaluated for different values of Doppler search precision. The best results from each Doppler value for each scenario were tabulated. The models generalization performance, were evaluated by applying the trained model on different recordings from Oakbat and FGI datasets. Lastly, the inference time for acquiring, preprocessing, and CNN processing time were measured.
"""""""""""""

Parts of the response were then selected if it seemed like a reasonable change to make. One example of how text were reworked with the answer to this prompt can be found as the first paragraph of the Results section.

C

Appendix 3: Sustainable Development Aspects

This thesis focuses on detecting spoofing and discuses avenues of spoofing detection, with the main goal of ensuring accurate GPS positioning and timing services. This objective aligns with several of the UN's Sustainable Development Goals (SDGs) [39].

SDG 11 - Sustainable Cities and Communities. As mentioned in the introduction, GPS interference is relatively common and can affect critical infrastructure, such as medical transport services [6]. Other instances of GPS interference have also been reported [7, 9]. Therefore, this work supports the sustainability of urban environments by indirectly working towards reliable navigation and timing solutions for emergency services. Furthermore, GPS interference may misdirect aerial vehicles and, in the absence of a human pilot, could even completely disrupt the operation of an unmanned aerial vehicle (UAV).

SDG 9 - Industry, Innovation, and Infrastructure. Unmanned aerial vehicles are now widely used in industry and form part of everyday infrastructure. Drones are used for tasks such as food delivery, and UAVs support productivity in agriculture and many other sectors. All these applications depend on accurate navigation provided by GNSS systems. They promote economic growth, but are also vulnerable to disruption if navigation systems are interfered with. To ensure sustainable industry and infrastructure, GPS spoofing detection needs to be integrated.

In summary, this thesis has examined the work from a sustainable development perspective and demonstrates that GPS spoofing introduces vulnerabilities that can be exploited.

DEPARTMENT OF PHYSICS AND ASTRONOMY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY