Thesis Dissertation

IMPLEMENTING AND EXPERIMENTING WITH INDOOR LOCALIZATION IN WSN AND IOT NETWORKS

Styliana Maria Zymara

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

May 2025

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

Implementing and Experimenting with Indoor Localization in WSN and IoT Networks

Styliana Maria Zymara

Supervisor

Dr. Vasos Vassiliou

A thesis submitted in partial fulfilment of the requirements for the award of a Bachelor's degree in Computer Science at the University of Cyprus

May 2025

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor Associate Professor Dr. Vasos Vassiliou for his invaluable guidance, insightful feedback, and continuous support throughout the process of writing this thesis. His encouragement and expertise have been instrumental in helping me navigate challenges and grow academically. I am also sincerely thankful to my family for their unwavering support and encouragement throughout the years of my studies at the University of Cyprus.

Abstract

As Internet of Things (IoT) deployments grow across smart cities and critical infrastructures, ensuring the security and reliability of wireless communication becomes increasingly vital. LoRaWAN, being one of the popularly employed Low-Power Wide-Area Network (LPWAN) protocols, is used extensively for low-energy and long-distance communication. Its unlicensed frequency utilization and centralized star topology, however, make it a particularly vulnerable protocol to jamming attacks. Despite numerous intrusion detection system (IDS) solutions, most do not address the spatial dimension of such attacks, leaving the source of disruption unlocalized.

This study proposes a two-part framework dedicated to LoRaWAN networks: (i) a machine learning-oriented jamming classification system using features such as Packet Loss Ratio (PLR), Received Signal Strength Indicator (RSSI), and Signal-to-Noise Ratio (SNR) to classify jammed or normal intervals, and (ii) an ensemble localization framework that combines four methods, including an adapted version of the WSN-oriented MMLAW algorithm for LoRaWAN's gateway architecture and the novel Jamming Impact Weighted Centroid (JIWC) method, to accurately estimate the jammer's location.

Experimental results demonstrate that combining multiple features yields robust jamming detection, even against imbalanced datasets. The ensemble approach mitigates individual method weaknesses, while impact scoring and aggregation provide reliable estimates despite the absence of ground truth coordinates. This work contributes a practical solution for both detecting and localizing jamming threats in LoRaWAN systems, setting the foundation for enhanced security in low-power IoT networks.

Contents

Chapter 1

Introduction

1.1 Motivation

The rise in Internet usage has prompted Internet of Things (IoT) protection companies to develop more advanced technologies and standard security practices to safeguard IoT devices from intrusions. Numerous methods have been proposed and discussed in the literature for detecting anomalies in IoT and Wireless Sensor Networks (WSNs), highlighting the critical need for effective Intrusion Detection Systems (IDS).

While many ML-based IDS solutions show high accuracy in detecting attacks, they frequently overlook the spatial aspect, i.e., the localization of the attacker or source of disruption, such as a jammer. In particular, indoor localization of attackers is underexplored, especially in LoRaWAN networks, despite its critical role in enabling effective mitigation strategies. This gap is primarily caused by LoRaWAN's unique architecture, which makes it challenging to use traditional localization techniques, as it depends on centralized communication via gateways and lacks device-to-device interactions.

Motivated by the shortcomings in current research, this thesis aims to bridge the identified gap, by providing a comprehensive review of state-of-the-art localization techniques and

proposing a framework for both jamming detection and jammer localization tailored towards LoRaWAN environments. Our interest in LoRaWAN is further motivated by its expanding worldwide popularity as well as its open and flexible deployment strategy.

1.2 Objective and Contribution

The main objective of this project is to investigate and implement a framework for the detection and localization of jamming attacks in LoRaWAN networks. One of the main challenges we faced was the limited availability of public datasets related specifically to LoRaWAN and jamming events.

More precisely, this work provides a comprehensive review of existing localization technologies and detection methods applicable to LoRaWAN. We implemented and evaluated various machine learning models for jamming detection using a LoRaWAN dataset. Key network metrics, such as signal-to-noise ratio (SNR), received signal strength indicator (RSSI), and packet loss ratio (PLR) were carefully selected and processed to be combined in the jamming indicator (JI).

Finally, despite the absence of ground truth location data, we developed an ensemble localization framework for a LoRa-based dataset. By analyzing consensus across methods and their convergence patterns, we hypothesized and validated the jammer's position at (0,0) through error metrics, demonstrating that our approach outperforms standalone methods in stability and precision.

1.3 Methodology

This thesis adopts a two-stage pipeline to address jamming in LoRaWAN networks: first, jamming detection is performed on a labeled LoRaWAN dataset using multiple machine learning (ML) models with features such as RSSI, SNR, and a carefully calculated Packet Loss Ratio (PLR) used to improve detection accuracy.

For the second stage, jammer localization is carried out on a separate LoRa-based dataset using an ensemble of four different methods to mitigate individual weaknesses. One of the main approaches, MMLAW (Modified Multilateration Localization Algorithm with

Weights), was originally developed by Dr. Michalis Savva for WSNs, but was modified to suit LoRaWAN's centralized gateway architecture. Additionally, this study proposes the Jamming Impact Weighted Centroid (JIWC) method, another localization solution.

1.4 Thesis Organization

Next, in Chapter 2 we provide the theoretical background on IoT, WSNs, LPWAN technologies, localization techniques, and jamming. Chapter 3 reviews related work on anomaly detection, jamming detection and localization in both WSN and LoRa-based networks. In Chapter 4, we include the detailed implementation of machine learning methods for our jamming detection framework, and accordingly, in Chapter 5 we present the design and application of multiple jammer localization methods on a separate LoRa dataset. Chapter 6 summarizes and discusses the results of both jamming detection and localization, including evaluation insights. Lastly, in Chapter 7 we provide conclusions, highlight key findings, limitations, and potential directions for future research.

Chapter 2

Theoretical Background

In this section, multiple key definitions will be provided along with the necessary theoretical background, information that is essential support the remainder of this study. First, all the relevant technologies for indoor localization will be presented, with particular emphasis on LoRaWAN. Then, an overview of the most relevant localization techniques will be exhibited, with a special sub-section dedicated to Machine Learning approaches. Following this, we will discuss possible attacks in IoT and WSN, focusing mostly on jamming, which is the central concern of this research. Finally, we explore jamming attacks in IoT and WSN networks to understand their impact on communication and localization performance.

2.1. WSNs and IoT Networks

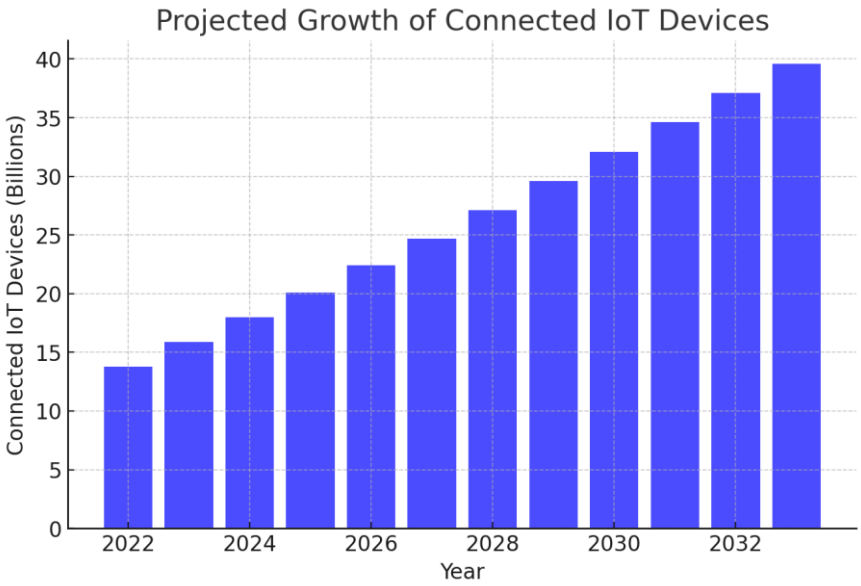


Figure 2.1.1: Number of Internet of Things (IoT) connections worldwide from 2022 to 2023, with forecasts from 2024 to 2033 [1].

A wireless sensor network (WSN) consists of hundreds to thousands of sensor nodes, each designed primarily for distributed data sensing. These nodes communicate wirelessly and send their collected data to a central base station for processing and analysis. However, they are limited by various constraints like power, energy, efficiency and deployment challenges [2].

Internet of Things (IoT) has gained substantial traction in the industrial and academic communities and extends this paradigm by building an intelligent, interconnected network through the combination of cyber and physical systems. This self-organized network involves sensors, actuators, and mobile devices (also referred to as “things”) that communicate data through standardized network protocols [3].

Healthcare applications, sensor networks, smart homes, smart cities, corporate networks, smart grid technologies, and web applications are just a few of the application areas that have made use of IoT technology. As these applications develop across various fields, they bring to the surface several challenges including keeping devices and networks safe,

preventing attacks on IoT systems and managing networks that have limited resources [4].

The widespread deployment of IoT systems allows one to have increased data movement and more complex device-to-device interactions, creating new opportunities for cybercriminals [5]. There are plenty of advantages to using tiny internet-connected devices, such as helping individuals be more efficient, but they also introduce security risks. The sheer number of IoT devices enables malicious actors to seek new ways of exploiting vulnerable systems. Compounding these challenges is the inherent heterogeneity of IoT infrastructure, where the amalgamation of various technologies yield unique vulnerabilities in security that typical protection measures like access control, encryption, and authentication often fail to adequately counteract [4].

Localization is a critical component in WSNs and IoT systems [5] and represents the process of determining the physical position of stationary or movable devices (e.g., smartphones, beacons, drones, etc.) within a network. While the underlying concept of localization traces back to NASA's satellite-based tracking systems developed in the 1960s [3], modern implementations go far beyond conventional navigation and object tracking. Of particular importance is localization's crucial role in cybersecurity frameworks, where Intrusion Detection Systems (IDS). Pinpointing an attacker's location (e.g., a jammer) becomes essential for maintaining network integrity, especially in wireless environments that are susceptible to disruption [6].

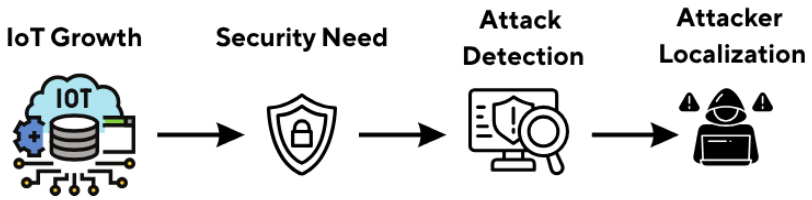


Figure 2.1.2: Motivation for jammer localization in IoT environments.

2.2 Localization Technologies Overview

Global Navigation Satellite Systems (GNSS) are the cornerstone of outdoor positioning and offer worldwide coverage, with many GPS systems using this technology (United

States, Russia, China, European Union [5]). GNSS employs a triangulation technique to estimate location by receiving signals from multiple satellites. However, GNSS struggles significantly when applied to interior navigation due to the presence of signal distortions. Signal attenuation, multipath effects caused by objects (e.g., walls or roofs) and general environmental noise are some of the main issues that make standalone GNSS impractical for indoor settings.

Nevertheless, GNSS can be used in conjunction with other technologies to enhance indoor positioning accuracy and availability. One solution, for instance, is GNSS/INS (Inertial Navigation System) integration, where inertial sensors (i.e., accelerometers and gyroscopes) compensate for GNSS signal dropouts by providing continuous motion tracking during indoor transitions [7].

Radio Frequency Technologies

Radio frequency (RF) technology employs wireless communication through the use of electromagnetic waves spanning between 3 kHz to 300 GHz in the frequency spectrum. The corresponding radio waves are enabled to propagate through the medium, allowing data transmission and distance measurements to be taken, making them a foundational component of localization systems. The RF technology is widely used for location estimation because of rampant availability and inexpensive supporting hardware like Wi-Fi access points (APs) and Bluetooth beacons [8].

Indoor positioning relies heavily on an equally accessible technology, **Wi-Fi (IEEE 802.11)** operating in 2.4 and 5 GHz bandwidths, also known as ISM (Industrial, Scientific, Medical) bands [5]. Wi-Fi chipsets are found in most modern devices, including computers, smartphones, smartwatches and other gadgets, which further contribute to the IoT infrastructure. Therefore, localization becomes much more accessible and cost-effective. Essentially, the existing Wi-Fi infrastructure can be utilized for most indoor localization systems with no additional hardware needed, since APs provide signal measurements such as Received Signal Strength Indicator (RSSI), Channel State Information (CSI), and round-trip time (RTT), among others.

On top of that, Wi-Fi has been significantly extended and covers distances up to 1 km, a considerable improvement from its earlier 100m limit [8]. With its broader coverage area and higher throughput compared to Bluetooth, Wi-Fi enables faster data communication, making it more viable for most applications. Specific methods that use Wi-Fi technology will be presented in Section 2.1.2, but namely some main approaches include Geometric approaches (e.g., Angle of Arrival (AoA), Time of Flight (ToF), Round Trip Time (RTT)), RSSI fingerprinting, and Channel State Information (CSI)-based localization.

Bluetooth, particularly its Low Energy variant **(BLE)**, is a short-range wireless personal area network (WPAN) technology which relies on electromagnetic waves within the frequency range of 2.4-2.48 GHz [9]. Similar to Wi-Fi, this technology is also prominent in localization systems because of its global availability and support by personal devices. Through the advancements introduced in Bluetooth 5.0, BLE has significantly augmented its range functionalities. In an ideal scenario, the operating range can be up to 40 meters indoors and 200 meters outdoors.

The main advantage of Bluetooth is its very low power consumption permitting months of operation on a single charge [5], which makes it an excellent fit for power-limited IoT devices. Additionally, features like adaptive frequency hopping help mitigate interference from other wireless signals.

However, Bluetooth does have shortcomings when it comes to indoor localization performance. Aside from the common challenges which all RF signals experience, i.e. multipath and signal attenuation, it typically provides sub-meter accuracy and latency of several seconds. If centimeter-level precision or instantaneous updates are desired for the localization system, Bluetooth alone is insufficient. Furthermore, due to Bluetooth's limited range in indoor environments, it is generally not preferred for localization across large-scale areas [8].

These constraints render Bluetooth ideal for proximity-based applications in retail, healthcare, airports and transportation stations, where localization is performed using BLE technology installed integrated into smartphones (e.g., iBeacons for Apple and Eddystone for Google) [8].

Ultra-Wideband (UWB) is a highly suitable technology for indoor localization due to its wide bandwidth (over 500 MHz) and low-power operation, with carrier frequencies above 2.5GHz. Because of its extremely low energy usage, UWB results in such a large bandwidth [5].

Some of the key advantages of UWB technology include high data rates and remarkable resistance to multipath fading because of the technology's distinct spectrum [8]. UWB's high time resolution and short pulse duration are fully utilized by techniques such as Time of Arrival (ToA), Time Difference of Arrival (TDoA), and Angle of Arrival (AoA) to achieve centimeter-level accuracy, outperforming RSSI-based methods. All the above characteristics make UWB a strong candidate for high-precision indoor localization in IoT and WSN applications.

Zigbee is a popular low-power, low-data-rate wireless communication technology built on the IEEE 802.15.4 standard and commonly used in the IoT era. It operates in the 868 MHz, 915 MHz and 2.4 GHz unlicensed ISM bands [5] with a maximum data rate of 250 kilobits per second and a range of up to 30 meters. Multiple topologies are supported by Zigbee such as tree, star and mesh. Even in intricate indoor environments, mesh networking enables devices to efficiently forward data through intermediate nodes to reach their destination. Because it operates in unlicensed ISM bands, it is prone to interference from other signals in the same spectrum, a shared vulnerability with both Wi-Fi and Bluetooth.

In indoor localization using Zigbee deployments, while RSSI is also the most widely adopted metric, alternative indicators like Link Quality Indication (LQI) can be used to reduce noise in ML-based localization and improve accuracy in dense, dynamic environments [3]. Although Zigbee's accuracy is generally lower than that of Wi-Fi or UWB, it can be an excellent choice for large-scale, low-power localization systems where strict precision is not critical, and energy efficiency is highly desired.

RFID (Radio-Frequency Identification) is a wireless radio wave-based technology that is used for object identification and tracking via tags with unique IDs [5]. The specific tags come in three types: active, passive, and semi-active.

Active RFID tags, powered by an internal battery, enable detection up to 100 meters and operate in the UHF (Ultra-high frequency) and SHF (Super high frequency) bands. However, they are not appropriate for sub-meter accuracy and are not usually integrated into personal mobile devices, limiting their use in user-side indoor localization [8].

Conversely, passive RFID tags are battery-free and very popular across multiple use cases due to their low cost, small size, and easy deployment.. Passive tags typically operate in the UHF band (860–960 MHz) and have a smaller range of 10 meters but are capable of sub-meter level detection under ideal conditions.

The way RFID systems operate, includes backscattering communication between mentioned RFID tags and RFID readers, with middleware for the data processing [8]. By employing multiple reference tags which act as a transmitter, RSSI values from nearby readers are used to estimate the position of a target tag by comparing signal similarities. In indoor localization, RFID systems are valued for their low cost, scalability, and resistance to occlusion and environmental interference.

Optical-based Technologies

Optical-based indoor localization techniques offer an alternative to radio frequency (RF) systems, particularly in RF-sensitive environments (e.g., hospitals). Optical technology uses visible or infrared light to fulfill specific functions, and in this case, aid in localization. Two of the well-known technologies under this category are Infrared (IR) and Visible Light Communication (VLC).

IR systems rely on line-of-sight (LoS) communication using IR-emitting devices (e.g., LEDs) and IR sensors (e.g., photodiodes). IR emitting devices send a signal with a unique ID which the sensor can detect to determine the target's location [5]. The benefits of this technology lie in its immunity to electromagnetic interference, a characteristic not applicable to RF-based systems. However, IR systems are sensitive to fluorescent light and sunlight interference and involve relatively costly hardware [8].

On the contrary, **VLC** systems determine position by using modulated visible lights (e.g., LEDs) and a sensor (smartphone camera, photodiode) to determine position. In fact, in many scenarios, VLC systems have better accuracy than Wi-Fi and benefit from LEDs' low power consumption, long lifespan, and immunity to sunlight interference.
Both VLC and IR, however, require line-of-sight (LoS) conditions between the transmitter and receiver in order to successfully estimate the location.

Inertial Sensors (Dead Reckoning)

Inertial sensors, which are commonly embedded in smartphones and IoT devices, use a technique often referred to as dead reckoning or Pedestrian Dead Reckoning (PDR) in order to enable relative indoor localization [5]. Inertial measuring units (IMU) include, among others: accelerometers (distance estimation via step counts), gyroscopes (angle or direction measurements), magnetometers (magnetic field measurements), pressure sensors (elevation estimation). However, localization accuracy degrades over time due to these sensors' susceptibility to drift and noise, particularly indoors.

To counteract this, inertial systems are typically paired with other technologies such as Wi-Fi, Bluetooth, GNSS, or UWB, and filtered with the use of Kalman or particle filters to improve position estimates [8].

Cellular-based Technologies

5G networks revolutionize cellular-based localization by leveraging advanced technologies which contribute to ultra-low latency (1-10ms), high throughput and sub-meter (or even centimeter-level) accuracy . The co-existence of IoT and 5G networks led to multiple adaptation technologies, such as mMIMO (massive multiple-input multiple-output) and mmWave (millimeter-wave) [10].

5G's high bandwidth and strict synchronization improve Time-of-Arrival (ToA) and Angle-of-Arrival (AoA) estimations which are ideal for IoT and WSN applications. Recent studies show (ML)-enhanced ToA tracking with 5G downlink

signals, providing 0.5-meter accuracy indoors by eliminating multipath effects via Kalman filtering [11].

With speeds 20 times faster than 4G, 5G allows mass and simultaneous device connections with real-time localization [10], and forms the backbone for industrial automation, emergency response systems, and smart cities.

Challenges remain, including hardware complexity as components of this network include 5G base stations (gNBs) and small cells, but despite their higher cost, offer more scalable solutions for large IoT deployments. Moreover, it is needed to strike a balance in power consumption for battery-powered IoT nodes and ensure privacy in location-based applications. Nevertheless, with edge computing and AI-driven analytics, 5G cellular localization is poised to underpin the next generation of IoT innovations.

2.3 LPWAN Technologies

Low-Power Wide Area Networks have emerged as a group of communication technologies and protocols, offering innovative solutions in IoT. As the name suggests, LPWAN technologies cover a very large area (kilometers level) through base stations while keeping the power usage and throughput of the end devices minimum [12]. They achieve this by performing adjustments to transmission rate, signal power, and intervals of communication, resulting in very little energy consumption.

While short-range applications frequently employ GHz-based technologies (Wi-Fi, Bluetooth, ZigBee), long-range communications require more energy-efficient solutions, with LPWANs operating in the MHz band [13].These networks are particularly suited for IoT deployments in smart cities, industrial IoT, and other large-scale applications where devices must operate for extended periods on battery power.

LoRa, SigFox and NB-IoT fall under the category of LPWAN technologies, each with distinct communication features and trade-offs. NB-IoT operates typically in licensed bands, while LoRa and SigFox operate on unlicensed industrial, scientific, and medical (ISM) bands.

2.3.1 LoRa (Long Range)

LoRa, short for "Long Range," is a wireless communication system designed to support long-range data transmission with ultra-low power consumption and is promoted by the LoRa Alliance. Since LoRa can offer long-distance communication (~15 km outdoors) at minimal power usage, it has been one of the most adopted LPWAN technologies so far. This makes it a well-suited technology to a broad set of applications, including industrial automation, smart cities, agriculture, environmental monitoring and urban infrastructure. LoRa is built on two layers: (i) Physical Layer (PHY) and (ii) MAC layer, also known as LoRaWAN [14].

LoRa Physical Layer (PHY) is the lower layer of the protocol stack and is responsible for the data transmission, i.e. the actual radio communication. The LoRa PHY is proprietary technology owned by Semtech Corporation [14]. It's closed source, meaning its core modulation technology is not publicly open.

LoRa's physical layer employs Chirp Spread Spectrum (CSS) Modulation, spreading the signal across a wide bandwidth (e.g., 125–500 kHz). Specifically, information is encoded in radio sinusoidal waves called "chirps", which are signals that linearly increase or decrease in frequency over time. As a result, the CSS method provides resilience against effects like multipath fading, Doppler effects, and interference, enabling reliable long-range communication in the presence of a significant amount of channel noise [15].

LoRaWAN MAC Layer is the upper layer built on top of the physical layers designed mainly for sensor networks, and it defines how devices organize communication in a shared radio environment by coordinating the access to the medium. Unlike the proprietary PHY, LoRaWAN is open and maintained by the LoRa Alliance and it was first introduced in 2015 [12].

As of 2025, LoRa Alliance allows public, private and hybrid networks to be deployed catering to various use cases and locations [16]. The Alliance has expanded to nearly 200 LoRaWAN network operators, providing coverage in almost every country worldwide. Moreover, the Alliance has emphasized the growth and variety of LoRaWAN network models, pointing out that during a three-year period, public LoRaWAN networks

increased by 66%, mostly due to the deployment of new LPWAN IoT infrastructure by satellite, community, and license-exempt network operators [17].

LoRaWAN Architecture

A typical LoRaWAN network consists of several key components working in concert to enable low-power, wide-area communication, and is usually organized in a star-of-stars topology [12]. Some of the key components are the end-devices, the gateways, the network server and the application server.

End devices, most commonly sensors or actuators, collect data or perform actions and communicate wirelessly with one or multiple gateways. These gateways subsequently forward the arriving frames to a network server, the network's central point of intelligence, which manages devices and forwards data to the appropriate application server for processing. The network server is responsible for removing duplicate messages, decoding the data and sending the messages back to the end-devices [14]. The communication between end devices and gateways uses the LoRa physical layer, which employs Chirp Spread Spectrum (CSS) modulation [15].

LoRa devices are not connected with only a single gateway, they simply broadcast their data and any nearby gateways that are within range can forward the packets to the network server where all the decision-making takes place. It must be highlighted that device-to-device communication is not supported by LoRaWAN, meaning that any packet from one device to another must be relayed via the network server.

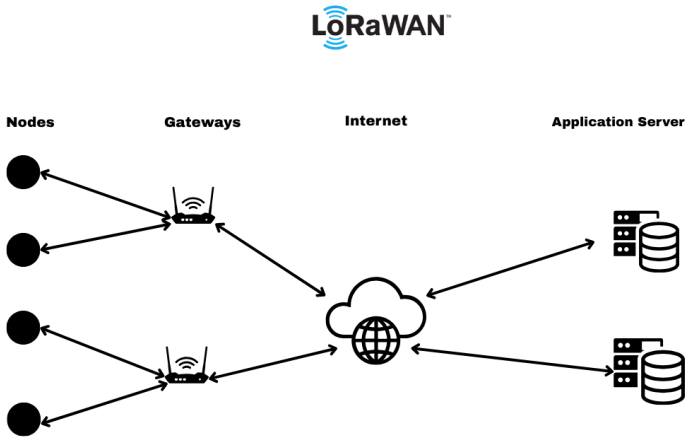


Figure 2.3.1.1: LoRaWAN architecture design

Furthermore, LoRaWAN end-devices are separated into three different classes:

Class A devices are the most energy efficient. They are bi-directional and can send data (uplink) whenever needed and open two short downlink receive windows afterwards to listen for replies. This class is ideal for battery-powered devices since downlink messages can only be sent right after an uplink, with the tradeoff of limited responsiveness.

Class B devices add scheduled receive windows, allowing the server to send data at specific times. These schedules are synced using beacons from the gateway, offering a balance between power use and downlink flexibility.

Class C devices keep their receive window open continuously, allowing for immediate downlink communication. This makes them highly responsive but it also results in maximum power usage, making them mostly suitable for plugged-in devices.

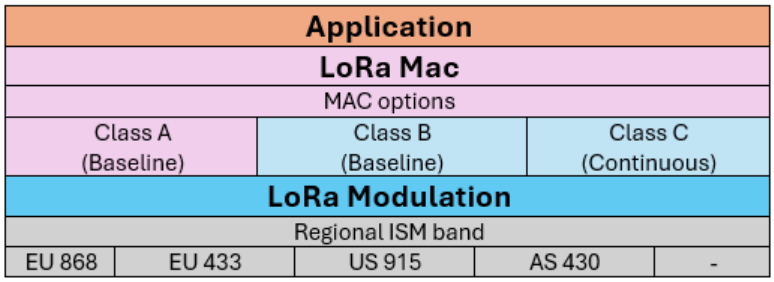


Figure 2.3.1.2: LoRaWAN Protocol Stack

Key Transmission Parameters

- **Carrier Frequency**

LoRa mostly operates in sub-GHz ISM bands (e.g., 868 MHz in Europe, 915 MHz in North America), where lower frequencies offer better penetration and range, but lower data rates [13]. Regional regulations dictate channel plans, for example, EU LoRaWAN uses 863–870 MHz with duty cycle restrictions (1% for most channels) to minimize interference [18].

- **Duty Cycle**

LoRaWAN operates in unlicensed ISM bands, and thus is subject to regional regulatory constraints, with one being the duty cycle. The duty cycle defines the maximum percentage of time a device is allowed to transmit on a given channel within a specific time period [12]. After each transmission, the device must wait for the remainder of the duty cycle window before it can transmit again, contributing to a fair channel utilization and reduced collision risks. Although this mechanism supports spectrum sharing and scalability, it also limits throughput in dense deployments.

- **Coding Rate (CR)**

The CR (e.g., 4/5, 4/6, 4/7, or 4/8) determines the ratio of error-correction bits to payload data. In simpler words, CR in regulates the number of extra bits used for error correction. For example, i) a CR of 4/5 means: for every 4 bits of actual data, 1 extra bit is added for error correction (total 5 bits sent), ii) a CR of 4/8 means: for every 4 bits of data, 4 extra bits are added (total 8 bits sent), and so on.

The tradeoff is that higher CRs (e.g., 4/8) are more reliable because they are better at handling noise, but at the cost of reduced throughput (more redundant bits transmitted). Accordingly, lower CRs have higher throughput, thereby are faster, but less reliable. This trade-off is critical for battery-constrained IoT devices [19].

- **Spreading Factor (SF)**

SF (ranging from SF7 to SF12) controls the number of chirps per symbol and directly affects range and data rate. LoRa's SF essentially defines the duration of the signal being stretched over time i.e. how slowly or quickly data is being transmitted. Higher SFs (e.g., SF12) increase sensitivity (up to −148 dBm) and support longer communication range, but reduce throughput (slower) and increase air time, making them vulnerable to prolonged jamming attacks [19]. In contrast, lower SFs (e.g., SF7) use fewer chirps per symbol, resulting in faster data transmission, shorter airtime, and hence, reduced jamming exposure.

Spreading Factor (SF)	Data Rate	Range	Air Time	Power Consumption	Jamming Vulnerability
SF7	High	Short	Very Low (~50 ms)	Low	Low
SF8	Medium-High	Short-Medium	Low (~100 ms)	Medium	Medium
SF9	Medium	Medium	Moderate (~200 ms)	Medium	Medium
SF10	Medium-Low	Medium-Long	High (~400 ms)	High	High
SF11	Low	Long	Very High (~800 ms)	High	High
SF12	Very Low	Very Long	Extremely High (~1000+ ms)	Very High	Very High

Table 2.3.1.1: Comparison of LoRa spreading factors and their trade-offs.

- **Bandwidth (BW)**

LoRa's BW (typically 125, 250, or 500 kHz) affects both data rate and spectral efficiency [19]. Narrow BWs enhance receiver sensitivity for longer communication distances and tolerance to noise at the cost of lower data rate. On the other hand, wider bandwidths offer the potential for higher data rates at the cost of lower coverage range and link budget due to increased power consumption and susceptibility to co-channel interference. Adaptive BW selection takes advantage of real-time SNR measurements to adaptively select bandwidth, i.e., automatically switch to 125 kHz in interference-heavy environments or 500 kHz when handling time-sensitive sensor data [20].

- **Adaptive Data Rate (ADR)**

Adaptive Data Rate (ADR) is another vital component of LoRaWAN that accommodates dynamic adaptation of several transmission parameters, including the spreading factor (SF), bandwidth (BW), and coding rate (CR). The rationale behind this mechanism is to optimize energy efficiency, maximize network capacity and reliability based on the existing state of the network. For instance, when the devices are near a

gateway and the signal strength is high the ADR mechanism may decrease the SF but increase the data rate in order to minimize air-time and power usage. Conversely, during poor signal conditions ADR may increase the SF at the expense of lower data rates to guarantee reliability. This flexibility of LoRaWAN can significantly contribute to large-scale IoT deployments, where varying distances and environmental noise demand dynamic optimization for high performance and battery life.

Limitations in LoRaWAN

While LoRaWAN offers great advantages such as long-range communication and low power consumption, it also suffers from certain inherent drawbacks due to design trade-offs [21]. A primary limitation is that it operates at a low bit rate and, as a result, messages experience long air-time, increasing their susceptibility to collisions, especially in dense network deployments. Since the protocol does not use carrier sensing (like CSMA) or timing synchronization to prevent overlaps, it relies mainly on the inherently low end-device transmission rate to reduce the chances of collision. Bi-directional communication further increases packet loss. This is not only due to collisions but also because acknowledgment messages (ACKs) can quickly exhaust a device's duty cycle limit, limiting further communication. [21]

Thanks to LoRa's Chirp Spread Spectrum (CSS) modulation, not all simultaneous transmissions lead to data loss. Gateways can successfully decode overlapping packets if they are sent using different Spreading Factors (SFs) and have similar RSSI. However, issues arise in two main scenarios: i) When two messages collide using the same SF, they interfere destructively resulting in loss of both. ii) When two messages have unequal signal strengths, i.e., if one message is significantly stronger, it can drown out the weaker one, even if the SFs are different, so only the stronger signal is successfully received.

Beyond these, LoRa's limited bandwidth and payload size constrain its use in applications that require high data throughput or real-time performance. Operating in unlicensed frequency bands also raises the risk of interference from external interference (non-LoRa devices). Lastly, deploying a LoRaWAN network may involve considerable setup complexity and cost.

2.3.2 Sigfox

Sigfox is a French LPWAN technology which has rapidly expanded its global footprint, used in more than 70 countries and has a coverage of 40 kilometers [22]. Although Sigfox and LoRaWAN both make use of license-free bands, their operation modes differ [23]. Sigfox operates as a centralized global network provider, managing infrastructure and services centrally, whereas LoRaWAN supports a decentralized topology supporting public, private, and hybrid network deployments.

Sigfox is based on a one-hop star topology and Aloha as its medium access protocol [22], relying on the backhaul connectivity of mobile operators. This solution simplifies end-user deployment but diminishes network flexibility and management of the infrastructure. Additionally, it employs Ultra NarrowBand (UNB) technology that can offer long-range communication, minimal power consumption and limits the number of dense base station (BS) placements in the area [23]. Modulation schemes include Differential Binary Phase Shift Keying (DBPSK) for uplink and Gaussian Frequency Shift Keying (GFSK) for downlink.

Nodes are constrained to send a maximum of 140 messages daily, which can't exceed 12 bytes (uplink communication). For downlink communication, devices can receive up to 4 messages per day, each limited to 8 bytes in size [22] and can only be received after the device prompts the network to deliver a downlink message (downlink request flag = 1).

Key benefits of Sigfox include low hardware and infrastructure costs, as well as extremely low power consumption with data rates of approximately 100 bps, enabling devices to last for years on a single battery. Lastly, Sigfox offers flexibility in hardware selection as it supports chipsets of various manufacturers, as opposed to LoRaWAN [23].

Sigfox has a number of drawbacks despite its benefits. Its centralized operational model restricts user control and requires registration and service fees. Moreover, given that all data are stored on Sigfox servers, privacy issues are raised [23]. While UNB technology reduces BS density and installation costs, it also restricts localization accuracy as higher BS density typically improves positioning capabilities. Finally, because Sigfox has limited downlink connection, it is best suited for applications where devices predominantly send data to the cloud with little need to receive data.

2.3.3 NB-IoT

Narrowband Internet of Things (NB-IoT) is a 3GPP-standardized Low Power Wide Area (LPWA) technology included in the LTE standard [24], which starter later than LoRaWAN and Sigfox. It uses the same licensed frequency bands as LTE, as opposed to LoRa and Sigfox which both use license-free bands. NB-IoT can operate under three modes: i) stand-alone ii) guard-band and iii) in-band. It is specifically designed to enable large-scale connectivity for IoT devices by decreasing deployment costs and battery usage. In order to achieve the aforementioned goals, it omits several LTE features (e.g., handover*, carrier aggregation, dual connectivity) [24].

The NB-IoT architecture is typically structured into three main layers: the perception layer, the network layer, and the application layer. The perception layer consists primarily of NB-IoT devices (sensors, actuators, or other IoT nodes) whose role is to capture data from the physical environment. The network layer serves as the communication backbone, comprising cellular base stations (eNBs) and core network entities to transmit and process the captured data. Finally, the application layer provides specific services and applications tailored to end-user needs, by using the processed data that was delivered by the network layer.

NB-IoT is strongly promoted by telecommunication operators and can be supported by existing telecommunication base stations (BSs). The technology also benefits from the security and quality guarantees of licensed spectrum [23]. However, due to the use of licensed bands it is more expensive to develop.

**Handover refers to the process of transferring a connection from one node or technology to another without disrupting the user experience.*

LPWAN Technologies Comparison

Feature	LoRaWAN	Sigfox	NB-IoT
Frequency Band	Unlicensed ISM (e.g., 868/915 MHz)	Unlicensed ISM (e.g., 868/902 MHz)	Licensed LTE bands
Modulation Scheme	Chirp Spread Spectrum (CSS)	DBPSK (uplink), GFSK (downlink)	OFDMA/SC-FDMA (LTE-based)
Network Model	Decentralized (supports public, private, hybrid)	Centralized global network provider	Fully centralized (operated by telecom providers)
Data Rate	0.3–50 kbps (adaptive)	~100 bps	~20–250 kbps
Topology	Star-of-stars topology	One-hop star topology	Cellular star topology
Localization Suitability	Moderate (multiple gateways can enhance accuracy)	Limited (low BS density reduces accuracy)	Good (dense cellular infrastructure = higher accuracy)

Table 2.3.1: LPWAN Technologies Comparison Table

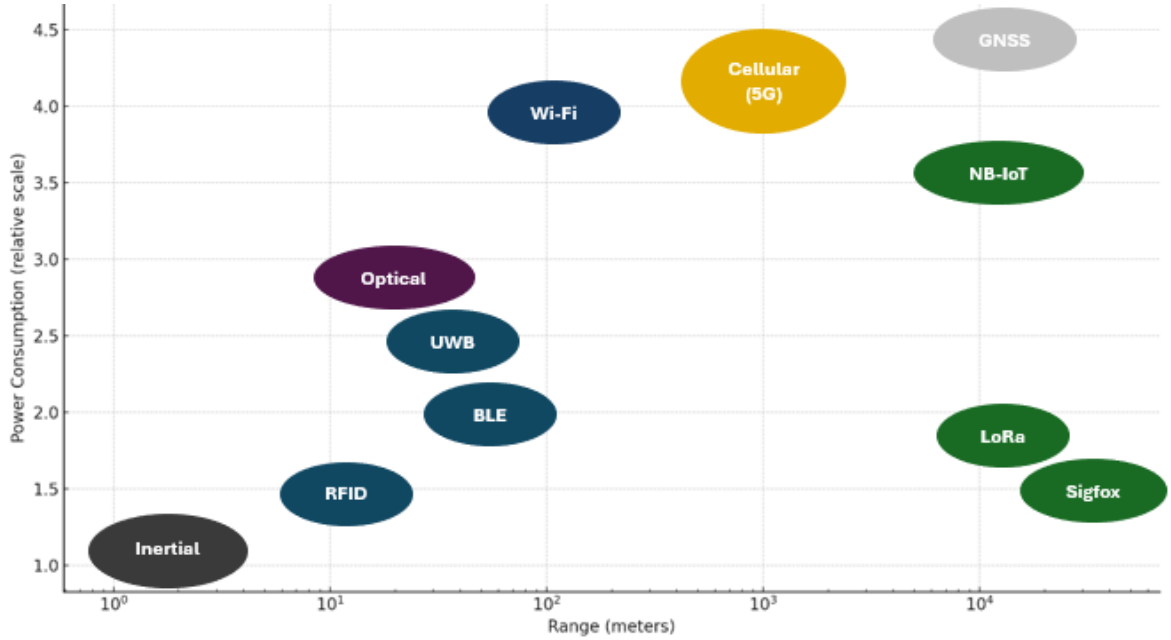


Figure 2.3.1: Localization Technologies by Range and Power Consumption

2.4 Localization Techniques Overview

Localization in the context of Wireless Sensor Networks (WSN) and IoT, refers to the process of determining the precise or approximate geographical coordinates of a device or a sensor [2]. Accurate localization is essential to a multitude of applications, including environmental monitoring, asset tracking, health care, security systems etc.

The substantial amounts of data collected by sensor nodes, regardless of their quality, lack the critical spatial context which is needed for informed decision-making. In cases where an attacker wants to target the security of the network (e.g., via jamming), pinpointing its position in the area is crucial.

Although localization mechanisms are indispensable in both traditional WSN and LoRaWAN networks, their methodologies and requirements differ significantly due to differences in operational conditions, e.g., communication coverage, energy demand, and hardware capability. Therefore, this chapter aims to focus on the available indoor localization methods in LoRaWAN and WSN/IoT networks and address the complexities of jamming localization.

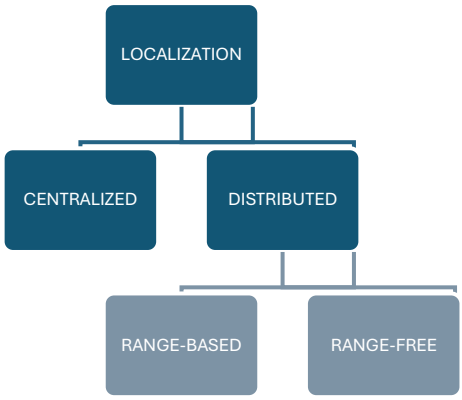


Figure 2.1.4: Classification of localization techniques

2.4.1 A Broad Taxonomy of Localization Methods

Localization methods in WSN and IoT networks can generally be classified along several characteristics [2]. By categorizing these techniques, our goal is to understand the

underlying principles related to each strategy and their suitability in various application settings.

Classification based on architecture

One primary way to classify localization methodologies is by the architecture of the localization process: centralized vs. distributed [2].

i) Centralized localization

In **centralized localization**, a designated central entity, normally a base station or a network server, gathers measurement data from all sensor nodes within the network. This central entity then processes the global information collected from the nodes using sophisticated algorithms to determine between-node distances [2].

One of the main benefits of centralized algorithms is that they possess a comprehensive global view of the network, which could potentially make them more accurate in location estimation. However, a centralized model can pose some problems concerning scalability and increasing computational complexity, because of the additional overhead placed on the central base station. All these factors reduce the practicability of centralized approaches on extremely large-scale sensor networks [2].

Furthermore, the reliance on one central anchor automatically introduces a single point of failure, which can influence the system's robustness. Consequently, recovery mechanisms must be employed to mitigate this risk.

ii) Distributed localization

In **distributed localization** we can observe a more autonomous approach, where each sensor node independently estimates its own position based on information obtained from its immediate neighbors or a limited set of reference points. Unlike centralized approaches, each node calculates the distances to neighbors and anchors by collecting measurements with different methods [2].

A prominent example of distributed localization is the Bayesian filter localization, which utilizes noisy measurements for determining the location of a sensor. More specifically,

this algorithm outputs probabilities of each predicted location commonly referred to as "beliefs." These algorithms iteratively refine these beliefs through repeated interactions among sensor nodes, a mechanism known as belief propagation [2].

Another simple method used within distributed localization is particle filtering. Particle filters, known for their effectiveness in handling non-Gaussian noise and providing quantifiable location uncertainties, are especially suitable for distributed implementations. They require fewer computational resources and iterations to converge and can easily handle dynamic and noisy environments, thus improving accuracy and reliability in location predictions.

Distributed localization algorithms have greater robustness than centralized algorithms, mainly because there is no single point of failure. Additionally, they easily scale to larger networks since computations are performed locally at each node rather than relying on a central anchor. Lastly, this peer-to-peer approach has reduced communication overhead and power consumption since nodes only exchange information with neighbors and communication is primarily localized.

On the other hand, distributed algorithms pose greater complexity in terms of local interactions compared to centralized algorithms. In order to achieve global consistency in location predictions, multiple iterations of message exchange are required among nodes. That happens because distributed methods rely on single-hop communication, whereas centralized algorithms use multi-hop communication. As a result, the accuracy of distributed localization can be lower compared to centralized methods due to the potential accumulation of errors as location estimates propagate through the network.

Distributed localization techniques are commonly further categorized into range-based and range-free methods, each suitable for different scenarios, as detailed further in this section.

Classification based on type of information

Another fundamental classification of localization techniques revolves around the type of information used to estimate the location: range-based vs. range-free. The choice of a suitable IoT localization method depends on the specific requirements of the application.

The description along with the benefits and drawbacks of each category will be discussed in the following section.

i) Range-based localization

Range-based localization methods rely on directly estimating the distance or angle between sensor nodes, typically to a set of anchor nodes whose positions are known. These distance or angle measurements can be determined using a range of methods such as Time of Arrival (ToA), Time Difference of Arrival (TDoA), Angle of Arrival (AoA), and Received Signal Strength Indicator (RSSI) [25]. Once these measurements are acquired, geometric principles i.e. trilateration, triangulation, or multilateration, are employed to calculate the position of the unknown node. For instance, LoRaWAN topologies often utilize TDoA because of their gateway-centric structure, leveraging synchronized timestamps across many gateways to triangulate device positions with meter-grade precision [26].

Range-based methods are typically more precise in location estimation [27], but they often require additional hardware (e.g., antennas, infrared sensors, etc.), strict time synchronization (for ToA/TDoA) or are affected by environmental variability (e.g. RSSI) [27]. In addition, range-based methods can cause higher energy consumption and typically require a more complicated implementation [26], with the lack of reference points also being an issue that affects the accuracy of calculations.

ii) Range-free localization

Range-free localization techniques do not depend on direct distance or angle measurements. Instead, these methods utilize different types of information regarding the network's topology in order to estimate the position of a node. For example, they take into consideration information such as the connectivity between nodes, the number of hops to anchor nodes, power of the wireless signal or patterns in the device's data [26], [27].

Range-free localization methods are typically more cost and energy efficient, simpler to use and easier to deploy due to the fact that they do not need specialized hardware to function. However, they generally offer lower accuracy compared to range-based

methods [26], and their performance is often influenced by the density and distribution of nodes in the network.

iii) Hybrid localization

The research community is showing strong interest in **hybrid localization techniques**, as they combine the strengths of both range-based and range-free methods in order to alleviate the limitations of standalone approaches and aiming for an enhanced performance and accuracy [28]. Beyond combining various estimation techniques, hybrid localization may also include scenarios where different technologies (e.g. Wi-Fi, Bluetooth) are integrated together into a single localization system. In the detailed overview (Section 2.4.2), two types of hybrid approaches will be discussed: the Joint technique, which aims to combine different localization methods and Data fusion, which mainly leverages a single method while integrating many communication technologies.

A key advantage of hybrid methods is their ability to mitigate environmental challenges. In the dynamic settings of IoT environments, machine learning (ML) algorithms are increasingly used to harmonize heterogeneous data streams. For instance, neural networks and deep learning models can handle a set of metrics such as RSSI, CSI, and hop-count to eliminate signal distortions caused by obstacles or interference, aiming to improve accuracy and energy efficiency [27].

Despite their benefits, hybrid approaches are fraught with challenges. Calibrating conflicting measurements from different modalities (e.g., RSSI vs. hop count) requires robust algorithms, and system complexity increases as it requires advanced hardware or synchronized protocols [29].

2.4.2 Detailed Overview of Localization Methods

i) Range-based localization techniques

Range-based localization methods achieve location estimation by using measurements such as distances or angles between end devices and anchors. These measurements are then used in several techniques to geometrically calculate the location of the unknown node. Some of the most common techniques will be listed below:

Time-based techniques

Time of Arrival (ToA), which is also referred to as Flight Time, is a technique that estimates the distance between a transmitter and a receiver by precisely measuring the time it takes for a signal to propagate from one to the other [2]. By knowing the travelling speed of the signal (typically the speed of light for radio waves, $c=3\times10^8$m/s), the distance can be directly estimated by multiplying the travel time by the speed.

Formula:

$$T = t_1 - t_0$$

$$d=c\times T$$

where:

- t_0 = *timestamp when the sender sends the signal*
- t_1 = *timestamp when the receiver receives the signal*
- d = *estimated distance (meters)*
- c = *speed of light (approximately* $c=3\times10^8$*m/s)*
- T = *time of flight (seconds)*

Although ToA can offer high accuracy in clear line-of-sight scenarios, there are some key limitations to address. First, the two nodes (receiver and sender of the signal) must be precisely aligned, thus, time synchronization for both nodes is crucial [2]. Furthermore, especially in indoor settings, ToA can be highly influenced by obstacles that deflect the emitted signals [26] and multipath propagation, which can cause signals to travel longer distances or arrive at different times. These issues can affect distance calculations and lead to inaccurate predictions.

It must be noted that accurate localization requires measurements from at least three base stations (anchor nodes) [5].

Time Difference of Arrival (TDoA) is slightly more flexible than ToA, as it overcomes the need for strict synchronization between the transmitter and the receiver nodes. TDoA requires synchronization only between base stations, because this method uses only the

difference in arrival times of a signal received at multiple geographically dispersed receivers [2]. By obtaining these measurements the location of a node can be determined by methods such as multilateration [26], which is based on the geometry of the hyperbolas formed by the constant time differences. Thus, TDoA does not require knowledge of the exact transmission time from the sender [5], as the ToA method does.

Formula:

$$\Delta T_{ij} = t_j - t_i$$

$$\Delta d_{ij} = c \times \Delta T_{ij}$$

where:

- t_i = *timestamp when receiver i receives the signal*
- t_j= *timestamp when receiver j receives the signal*
- ΔT_{ij} = *time difference of arrival between receivers i and j (seconds)*
- Δd_{ij} = *distance difference between the transmitter to receivers i and j (meters)*
- c = *speed of light (approximately* $c=3\times10^8 m/s$*)*

The drawbacks of the TDoA method are similar to those described for the ToA method. While TDoA eliminates the strict clock synchronization requirement between the target and anchors—a crucial requirement for the aforementioned ToA approach, it still necessitates accurate time synchronization among the fixed stations (e.g. gateways) [5]. Moreover, the accuracy of both ToA and TDoA can be influenced by noise, multipath propagation and interference.

In the context of LoRaWAN, TDoA is a commonly employed network-based localization method, which offers a lower power consumption profile for the end devices compared to GPS, but comes with the tradeoff of lower accuracy [30].

Angle-based techniques

Angle of Arrival (AoA) is a directional localization technique that estimates the angle at which a signal arrives at a receiver [2]. To be precise, AoA localization estimates the location at the center of gravity within the intersection which was created by the sight

triangles between the anchors and the target [26]. This method utilizes directional antennas to measure angles from transmitters to receivers, in order to determine the direction of the incoming signal. Then, triangulation can be used to approximate the location of the target, thus, at least two fixed positioned anchors are necessary.

Although AoA provides reasonable accuracy, it requires specialized hardware (e.g. antenna arrays) which increases the cost and complexity of the infrastructure. Moreover, this method is also affected by environmental limitations (e.g. shadowing, multipath) [2] and its accuracy is highly dependent on the directionality of the antennas. Therefore, AoA typically demands a clear Line-of-Sight (LOS) area for achieving optimal results.

RSSI radio propagation technique

Received Signal Strength Indicator (RSSI) is perhaps one of the most widely used signal metrics in localization techniques due to its simplicity and low hardware cost. More specifically, RSSI is readily available in most wireless communication devices/sensors, and no additional hardware is required to obtain it. The main idea behind RSSI is that the strength of a radio signal becomes weaker as the distance from the transmitter increases [2].

RSSI-based radio propagation techniques estimate the position of a device by first measuring the RSS from multiple base stations or wireless access points [5]. Then, path loss models use the collected measurements to calculate the distance between the target and reference point. After estimating distances, localization algorithms such as multilateration or trilateration can determine the coordinates of the target.

Although the RSSI measurement is easily available, its accuracy is largely sensitive to environmental factors. Physical obstacles that create Non-Line-Of-Sight conditions, such as walls, can cause signal fluctuations due to attenuation, reflection, diffraction, and scattering. These limitations lead to significant variations in the received signal strength that do not solely depend on distance. Additionally, noise and external interference from other wireless devices will distort RSSI readings as well, which makes it an unreliable standalone metric—especially for indoor environments.

Another primary challenge of this technique is that it requires prior knowledge of the environment's path loss behavior, typically determined by offline calibration. More specifically, in order to apply this technique successfully, parameters like the path loss exponent, the reference RSSI value at a known distance, and environmental propagation factors must be carefully tuned to the deployment conditions.

The inverse relationship between the strength of the signal and the distance can be expressed by a path loss model, which takes into account characteristics of the propagation environment.

Path Loss Model Formula:

$$RSSI(d)=RSSI(d_0)-10n\log_{10}(d/d_0)$$

where:

- *RSSI(d) = received signal strength (dBm) at distance d*
- *RSSI(d_0) = reference RSSI (dBm) at known distance d_0 (usually 1 meter)*
- *n = path loss exponent (depends on environment, e.g. 2 for free space, 3–4 indoors)*
- *d = distance between transmitter and receiver (meters)*
- *d_0 = reference distance (meters)*

In LoRaWAN networks, RSSI-based propagation models are most suited for outdoor localization, where the surroundings are stable and signal fading is more predictable. Simple path loss models may limit the effectiveness of localization in LoRaWAN indoor environments, therefore, fingerprinting and hybrid methods are generally preferred in the literature, with ML-based solutions increasingly gaining attention [31].

Geometric-based Algorithms

Once distance or angle measurements have been obtained using range-based techniques (e.g., ToA, TDoA, AoA, or RSSI), the next step usually involves the application of geometric methods to estimate the position of the target node. The three most widely used approaches are trilateration, multilateration, and triangulation.

Trilateration calculates a node's position by measuring its distance from at least three anchor nodes with known coordinates. After determining the distances, which are also referred to as the circle radii, the intersection of spheres (or circles) can estimate the location of the target [2].

Given distances d_i to anchors at (x_i,y_i,z_i) the system solves:

$$\text{2-D Trilateration: } \sqrt{(x-x_i)^2+(y-y_i)^2} = d_i$$

$$\text{3-D Trilateration: } \sqrt{(x-x_i)^2+(y-y_i)^2+(z-z_i)^2} = d_i$$

Trilateration can be reliable about predicting correct locations when precise information is provided. However, that is not always the case in practical scenarios (e.g. RSSI fluctuations or ToA inaccuracies), and the circles might not intersect to any point. To address this, the mathematical approach of Least-Squares Estimation (LSE) can often be used to linearize this nonlinear system above.[5]

Multilateration (Hyperbolic Positioning) can be used with more than three nodes using the distance estimates via Time Difference of Arrival (TDoA) [2]. It locates a device by comparing time differences of signals received at multiple anchors, forming hyperbolic curves (2D) or surfaces (3D). For anchors i and j the TDoA equation defines hyperbolic position solutions:

$$c \cdot (t_i - t_j) = \sqrt{(x-x_i)^2+(y-y_i)^2} - \sqrt{(x-x_j)^2+(y-y_j)^2}$$

$$\tan(\theta_i) = \frac{y-y_i}{x-x_i}$$

where:

- c: Signal propagation speed (e.g., 3×10^8 m/s for radio waves)
- t_i, t_j: Signal arrival times at anchors i and j (seconds)
- (x, y): Unknown target coordinates
- (x_i, y_i), (x_j, y_j): Known positions of anchors i and j

As discussed in the TDoA description, this approach depends only on anchor nodes synchronization instead of demanding synchronization between anchors and end devices. Advanced solvers such as Maximum Likelihood Estimation (MLE), are commonly employed to mitigate nonlinearity and noise.

Triangulation is a geometric localization technique which uses angle measurements collected from at least two known anchors (in 2D) or three (in 3D). The target node calculates its location by determining the AoA from each anchor [2]. As a result, lines radiating from each anchor are formed and the location of the target node corresponds to where these bearing lines intersect. The mathematical model for each anchor i is given:

$$\tan(\theta_i) = \frac{y - y_i}{x - x_i}$$

where:

- θ_i is the measured angle at anchor i,
- (x_i, y_i) is the coordinate of anchor i,
- (x, y) is the unknown target position.

There are a few practical considerations, which involve the efficiency of the necessary specialized hardware (e.g. directional antennas or antenna arrays), as antenna misalignment in indoor settings can drastically influence the performance of the localization. These hardware requirements, although essential for accurate angle estimation, increase both the complexity and the cost of deployment. Moreover, the triangulation technique is particularly vulnerable to environmental distortions in indoor spaces, just like the other geometric-based methods. There is a requirement of clear line-of-sight (LoS), as multipath interference and shadowing can potentially degrade the accuracy of the system.

ii) Range-Free Localization Techniques

Range-free localization methods present an alternative strategy to determining the location of nodes in WSN and IoT networks. The primary distinction between range-based and range-free localization methods is their reliance on physical distance or angle measurements. The former requires precise signal-based metrics (e.g., RSSI, ToA, or AoA) to estimate positions, while the latter uses network connectivity and proximity data (e.g., hop counts or anchor node positions) without direct distance calculations [26].

Fingerprinting method

Extensive research and implementations have been undertaken concerning location estimation with Fingerprinting methods because RSSI measurements can be easily obtained [5].

The term "fingerprint-based" localization derives from the concept of human fingerprints, which are always unique to an individual, just as a signal fingerprint is intended to be unique to a specific point in space within a building. A core foundation of this localization method involves the mapping of the fingerprints with their corresponding physical coordinates within the indoor space.

Creating unique fingerprints entails considering the specific signal features, which are measurable properties. The most common signal feature, as previously discussed, is the Received Signal Strength Indicator (RSSI) used to represent the power of a wireless signal arriving at a device. RSSI is easy to retrieve with minimal hardware requirements, making it a practical choice for most indoor localization systems.

However, since RSSI also suffers from indoor interference, multipath effects and all the noise in indoor dynamic environments, advancements in wireless technology have enabled the use of more sophisticated features, such as Channel State Information (CSI) [32]. CSI provides richer information about the wireless channel, including amplitude and phase information across multiple subcarriers, which can enable more precise localization. [33]

Another challenge with fingerprinting is the requirement for a large number of measurements to build the database. On top of that, any changes in the environment may have an impact on the recorded signal properties, meaning the database must be updated regularly which takes extra effort and time.

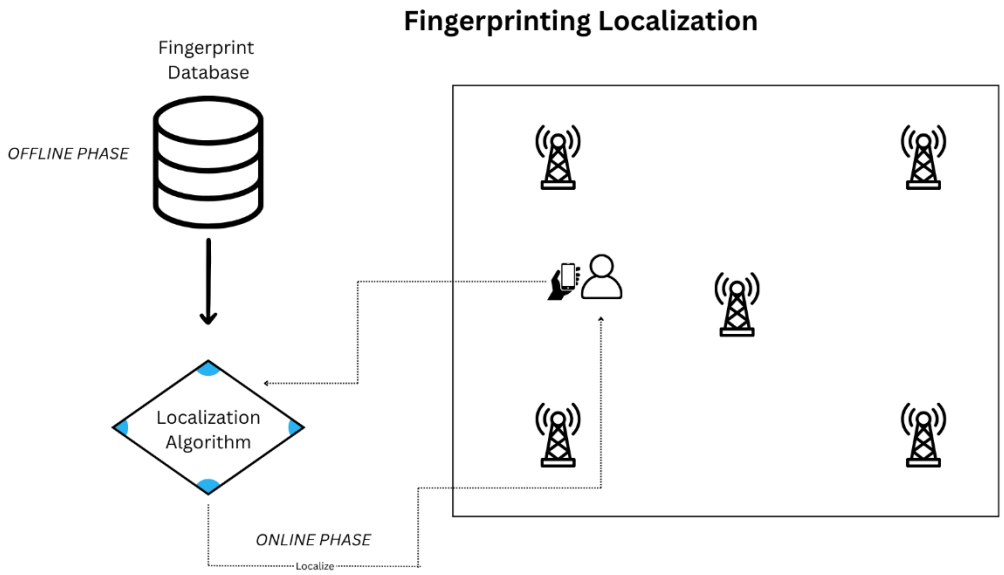


Figure 2.4.2.1: Fingerprinting Localization

Traditional vs AI-Based Fingerprinting

Fingerprinting methods can be further categorized into traditional approaches and artificial intelligence approaches [34]. Both categories consist of two operational stages: (i) the offline phase, also referred to as training, and (ii) the online phase, also referred to as localization stage. The main difference lies in how they process acquired fingerprints: traditional approaches estimate the target's position based on the closest reference points collected during the offline phase, while AI-based approaches use ML to train models and estimate the target node's position [34].

Once RSSI is combined with ML algorithms the impact of environmental factors can be mitigated. This is relevant also for LoRaWAN setups, which normally utilize RSSI metrics in conjunction with ML for fingerprint-based localization. For instance, a study in [12] achieved 98.8% accuracy in indoor localization using LoRaWAN RSSI and SNR data via neural networks. The study demonstrates how ML converts noisy

RSSI readings into precise location estimates, minimizing the signal fluctuations caused by environmental factors (e.g. multipath).

The result of the offline phase is the creation of a fingerprint database, also known as a radio map. This database serves as the foundational reference for the entire localization system, as it stores the collected signal features (the fingerprints) along with the precisely known coordinates (or labels) of their corresponding reference points RPs.

Moving on to the online phase, once the fingerprint database has been established, the system can be used to determine the location of a device in real-time. This real-time measurement of the device generates a new fingerprint at its current unknown location, which will be compared with the rest of the collected values in the database.

Lastly, the density of RPs within the environment presents a trade-off: higher density requires longer data collection efforts but can possibly capture finer-grained variations in the signal environment and thus result in more accurate location estimates. In contrast, lower density reduces the data collection burden but can lead to a less precise localization capability [35].

Below, a comparison of the main characteristics of RSSI-based fingerprinting and RSSI radio propagation techniques will be presented.

Aspect	**Fingerprinting Method** **(Range-Free)**	**RSSI Radio Propagation Method** **(Range-Based)**
Principle	Matches measured RSSI values to a pre-collected database (radio map)	Uses path loss models to convert RSSI to distance and applies trilateration/multilateration
Main Calculation Approach	Pattern matching or ML classification	Analytical distance estimation via formula
Environmental Sensitivity	High (Updates required if the environment changes significantly)	High (Errors if path loss parameters are misestimated)

Complexity	Medium to high (especially with AI-based methods)	Low to medium (simpler implementation but depends on accurate modeling)
Adaptability	Better with AI-based systems (can adapt to noise/variations)	Less adaptable
Use Cases	Common in indoor positioning systems (Wi-Fi, LoRaWAN, Bluetooth)	Used in simpler setups

Table 2.4.2.1: Comparison between RSSI fingerprinting (range-free) and RSSI propagation model (range-based) localization techniques across key aspects.

Range-Free Localization Techniques (cont.)

Proximity-based localization does not determine the precise location of a device. Instead, it estimates the position of an end device according to a predefined area [5] by relying on metrics that indicate whether a device is within a specific radius R of an anchor (coverage-based) or within direct communication range (connectivity-based) [26]. When a device is detected by an anchor, it is concluded that the specific device is inside the detector's proximity region.

Although it cannot provide relative or absolute location estimations, the proximity-based localization technique has been used extensively in IoT and WSN literature because of its simple implementation, minimal energy and computational requirements. It has also been widely used in most of GSM-based localization systems [5].

The **DV-Hop (Distance Vector Hop)** algorithm is a prominent range-free technique which uses a hop-based propagation model [26]. This algorithm estimates the distance between anchor and sensor nodes based on the number of hops in the shortest path between them and an average hop size [2].

The algorithm typically operates in three phases: First, anchor nodes broadcast their location information throughout the network, each node begins with a hop count set to zero and records the number of hops to reach each anchor node. During this iterative process, nodes update their tables as packets are received and replace their hop count values with lower ones [26]. As a result, all nodes will have the minimum hop count

recorded for each anchor ($HopCount_i$). Second, each anchor node calculates its average hop size ($HopSize_i$) by dividing the actual distance to other anchor nodes by the number of hops between them. This average hop size is then broadcast to all other neighbor nodes in the network. Finally, each target node uses the following formula in order to determine its distance from each anchor node based on the received information:

$$d_i = HopCount_i \times HopSize_i$$

Where:

- d_i = estimated distance to anchor node i
- $HopCount_i$ = number of hops from the target node to anchor i
- $HopSize_i$ = average physical distance per hop for anchor i

Once these estimated distances to several anchor nodes have been established, the position of an unknown node can be determined using geometric methods like trilateration or multilateration.

DV-Hop is known for its simplicity and scalability, as it does not require specialized ranging hardware. It is also particularly useful for constrained nodes which cannot process the entire network [25] and can localize a node with less than 3 neighbor anchors [36].

A limitation of DV-Hop is the assumption that all hops have a uniform distance (same physical length), which may not always be true in real-world networks with irregular topologies or obstructions.

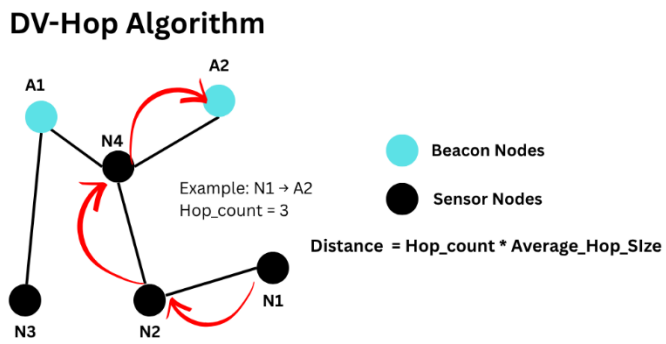


Figure 2.4.2.2: DV-Hop Localization Algorithm with Example Hop Count Propagation

The **Centroid** method is another simple range-free localization technique utilized in WSN and IoT networks. It estimates the location of an unknown node as the geometrical center (or centroid) of all neighboring in-range anchor nodes, averaging their known (x, y) coordinates [25].

Centroid-based localization has two main variations:

i. **Simple centroid**, which computes the arithmetic mean of anchor node coordinates, treating all anchors with equal importance.

   For a node M with N adjacent anchors at positions (xi, yi), its estimated position ($x_{centroid}$,$y_{centroid}$) is:

$$x_{\text{centroid}} = \frac{1}{N}\sum_{i=1}^{N} x_i\,, \quad y_{\text{centroid}} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

   Example: If three anchors $A1, A2, A3$ have coordinates $(x1, y1), (x2, y2), (x3, y3)$ the centroid is: $\left(\frac{x_1+x_2+x_3}{3}, \frac{y_1+y_2+y_3}{3}\right)$

ii. **Weighted Centroid Localization algorithm (WCL)**, which allocates weights as a function of proximity of the node to critical network entities (e.g. jamming attacks) [25] in order to attract the estimated position to close reference points. This weighting approach helps to mitigate errors caused by environmental interference and is capable of achieving more accurate results compared to the standard WCL method [37].

$$x_{\text{WCL}} = \frac{\sum_{i=1}^{N} w_i x_i}{\sum_{i=1}^{N} w_i}, y_{\text{WCL}} = \frac{\sum_{i=1}^{N} w_i y_i}{\sum_{i=1}^{N} w_i}$$

   The weights w_i can be derived from various proximity or quality metrics. Common choices include:

- Inverse Distance Models:

$$w_i = \frac{1}{d_i} \quad \text{or} \quad w_i = \frac{1}{d_i^k} \quad (k = 1,2)$$

where: *di* is the estimated distance to anchor *i*.

- RSSI-Based Models:

$$w_i = \text{RSSI}_i \quad \text{or} \quad w_i = \text{RSSI}_i^p \quad (p = 1,2)$$

where: RSSI is the received signal strength from anchor *i*.

While centroid methods are computationally very simple and cost-effective, they are highly dependent on anchor density and distribution in the area, thus, it is prone to significant localization errors.

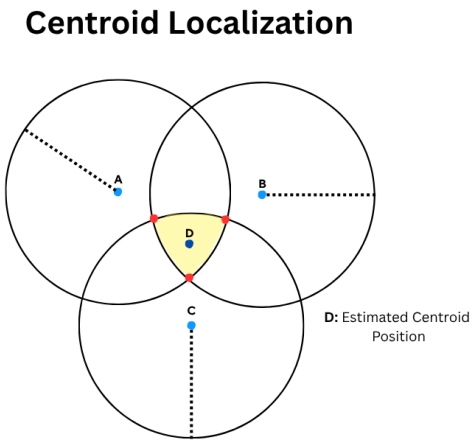


Figure 2.4.2.3: DV-Hop Localization Algorithm with Example Hop Count Propagation

Approximate Point in Triangle (APIT) works by having every unknown node determine its location relative to three anchors forming a triangle. The node checks whether it lies inside or outside of the formed triangle by comparing signal strength indicators from nearby non-anchor nodes. This process is repeated for every triplet of

neighboring anchors and by aggregating results from multiple triangles, the node estimates its position as the intersection of feasible regions.

APIT performs well in irregular network topologies and as a range-free method it does not require any distance measurements. Nevertheless, its performance still depends on the density of anchor nodes and the communication range of the anchors.

Amorphous localization shares similarities with DV-Hop in terms of its approach to location estimation. Similar to DV-Hop, Amorphous typically comprises an initial phase in which anchor flood both their location information and hop counts to the entire network. The key difference, however, is that Amorphous uses a probabilistic model to manage varying node densities, whereas the DV-Hop uses a uniform hop distance. More precisely, amorphous uses the neighbor connectivity to improve positions iteratively, limiting errors and resulting in better accuracy than DV-Hop in non-uniform settings.

Similar to the other range-based methods, its performance strongly depends on network density and anchor distribution. Although Amorphous requires higher computational overhead than DV-Hop, the trade-off is improved positioning accuracy, making it a preferable solution for real world applications where node distribution becomes irregular.

Virtual Force Iterative Localization (VFIL) is a range-free algorithm that utilizes virtual forces to estimate node positions in WSNs and IoT deployments. The method works by simulating attractive and repulsive forces between anchor nodes (known positions) and unknown nodes while refining positions iteratively to minimize localization errors. Furthermore, VFIL has been used as an enhancement of the Centroid Localization method in jamming scenarios in wireless networks [38].

This algorithm is effective in large-scale, low-power LoRaWAN networks, where sparse anchor distribution and RSSI variability challenge traditional methods. It can also be useful in scenarios where nodes are spread out uneven in the area and balance is desirable [25]. Ultimately, the underlying principle of using virtual forces is to optimize the network organization and lead to more energy-efficient communication solutions, a critical achievement for IoT resource-constrained environments.

iii) Hybrid localization techniques:

The **joint technique** in hybrid localization systems attempts to combine two or more distinct methods, such as proximity or multilateration. By leveraging the strengths of each method, hybrid systems can become more robust under varying network conditions.

A notable example is the Hybrid DV-Hop algorithm which was proposed in [29]. It improves the system's accuracy by combining RSSI (for one-hop neighbors) and hop-count (for multi-hop nodes). However, this approach introduces added computational complexity and demands careful fine-tuning.

The second approach involves integrating multiple types of sensor data or communication technologies (e.g., Wi-Fi, Bluetooth, Zigbee) into a unified localization framework, also referred to as "**data fusion**" [26]. An effective hybrid system must fuse diverse measurements from one or multiple devices to enable seamless localization and adaptive services. The most widespread technology in IoT deployments is without a doubt Wi-Fi, and thus, it can be integrated with more available technologies.

An example of a data-fusion framework is presented in [39], and is tailored for Wi-Fi, Bluetooth, Zigbee and UWB protocols. The system offers high accuracy because it takes into account the best features of each technology, such as Wi-Fi's broad coverage, UWB's precision, and Bluetooth's low energy consumption, while compensating for their respective weaknesses.

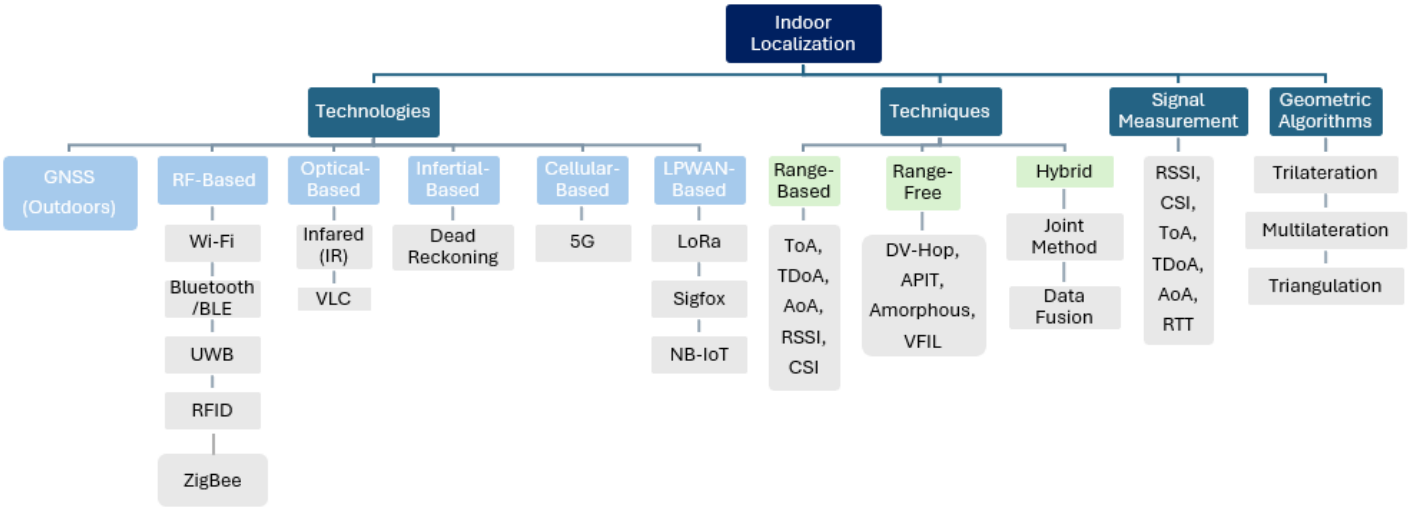


Figure 2.4.2.4: Indoor Localization Taxonomy

2.5 Machine Learning Algorithms in Fingerprint-Based Localization

Fingerprint-based indoor positioning systems have been increasingly making use of machine learning methodologies to deliver accurate and reliable indoor localization, in environments where GPS signals are either attenuated or completely unavailable. The initial attraction of the fingerprinting approach stemmed from its capacity to operate without a strict line of sight (LoS) between the receiver and the source of the signal, which presented a significant benefit compared to traditional GPS in indoor environments.

Along with the development of the Internet of Things (IoT) and wireless sensor networks (WSNs), machine learning (ML) technology has been applied to indoor localization systems more and more markedly. Recent advancements in ML have enabled more sophisticated algorithms to be used in fingerprint-based systems, which facilitate better handling of signal noise, dynamic environmental conditions, and device heterogeneity. In the following sections, we will explore various ML algorithms that are widely mentioned in the literature.

Supervised learning is a machine learning paradigm where models are trained on labeled datasets, that is, input data (e.g. RSSI , CSI etc.) are paired with known ground-truth coordinates. The goal is to learn a mapping function that accurately predicts locations for

unseen sensor data. This approach is widely used in fingerprint-based localization due to its high accuracy in structured environments, though it requires extensive labeled datasets, which can be costly to collect [40]
The key supervised learning methods used in indoor localization systems will be discussed below.

A fundamental technique employed in fingerprint identification is the **K-Nearest Neighbor (KNN) algorithm**, which is a supervised learning method. This specific algorithm operates by selecting the K-nearest annotated measurements from an offline radio map and using a weighted average of their location coordinates to approximate an unknown location of a target. To achieve that, the algorithm calculates the inverse of the Euclidean distance between the observed RSS (Received Signal Strength) and its K-nearest training samples as weights [41]. Despite its simplicity, the KNN algorithm necessitates the maintenance of a massive radio map, as it must store all the RSS training values for frequent calculations [42].

The estimated position $\hat{p}$ of the target is calculated as the weighted average of the positions of the K nearest neighbors:

$$\hat{p} = \frac{\sum_{i=1}^{K} w_i \cdot p_i}{\sum_{i=1}^{K} w_i}$$

Where:

- p_i is the position of the i^{th} neighbor,
- $w_i = \frac{1}{d_i}$ is the weight (inverse of Euclidean distance),
- $d_i = |r - r_i|$ is the distance between the observed RSS vector r and the i^{th} stored RSS vector r_i.

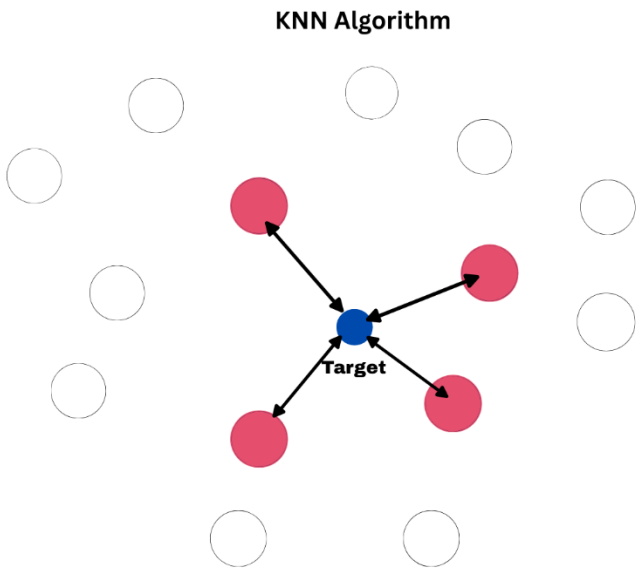


Figure 2.5.1: K-Nearest Neighbor (KNN) Algorithm Example

Support Vector Machines (SVM) are also conventionally used for classification problems within this domain. As stated in [43], the kernel approach within SVM allows for enhanced generalization in modeling both linear and non-linear interactions between fingerprint classes. By using the kernel functions, the SVM method aims to solve the randomness and incompleteness of the RSS measurements but has a limitation of high computing complexity [42].

A **Multi-Layer Perceptron (MLP)** is a type of feedforward artificial neural network (ANN) inspired by biological neural networks [44] and composed of an input layer, one or more hidden layers, and a target output layer. Each layer is made up of interconnected neurons that enable the network to understand complicated patterns by applying nonlinear transformations to the input data. More specifically, resilient fingerprint features are extracted through the training MLPs with backpropagation. Backpropagation is an optimization algorithm that minimizes prediction errors by tuning or changing synaptic weights via gradient descent. Generally, the hierarchical nature of the MLP makes it inherently well suited for Wi-Fi fingerprinting for complex and indoor environments where signal variance due to interference and moving objects poses major challenges [44].

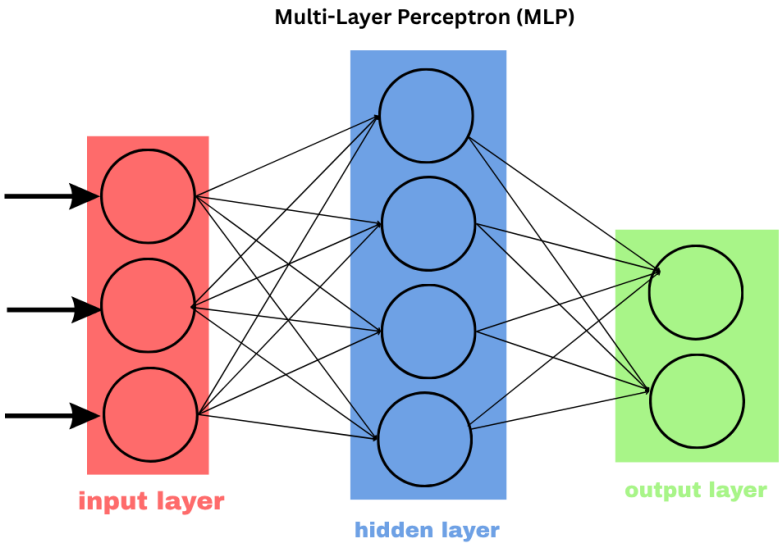


Figure 2.5.2: Multi-Layer Perceptron (MLP) Architecture

However, all of the above models may risk overfitting when trained on limited annotated fingerprint data [43].

Unsupervised learning has been explored to automatically learn meaningful representations from unlabeled data, often for dimensionality reduction or clustering in fingerprint-based localization systems [33]. The purpose of simplifying the data by identifying natural groupings or similar signal patterns (clustering) offers more efficient processing and storage of the data, which is crucial for IoT environments. Some commonly used unsupervised learning techniques are Autoencoders and Principal Component Analysis (PCA).

More specifically, **autoencoders** are neural networks that compress input data into a lower-dimensional latent space (encoder) and reconstruct it (decoder) in order to preserve essential features for tasks like Wi-Fi fingerprint denoising or feature extraction. For instance, [45] employs an autoencoder to transform sparse Wi-Fi RSSI measurements into compact, trainable feature sets.

Principal Component Analysis (PCA) is a linear transformation that projects high-dimensional data into orthogonal components while retaining the highest variance values. As highlighted by Alhmiedat [34], PCA helps localization minimize storage costs while still providing the discriminative accuracy needed for accurate positioning. Overall, these methods strike a balance between computational costs and localization precision for scalable deployment across resource-constrained IoT networks.

Semi-supervised learning offers a promising solution to localization by bridging the gap between supervised and unsupervised learning approaches, while using small sets of labeled data together with larger sets of unlabeled data. In the majority of indoor localization scenarios, collecting accurate labeled location data (i.e., sensor readings paired with precise coordinates) can be expensive and usually very time-consuming [45] [40].

However, this is the case where unlabeled data can be useful (e.g. raw sensor measurements without location tags), which are readily available and often easier to acquire. By incorporating the acquired unlabeled data into the training process, semi-supervised learning can help to uncover underlying patterns and structures that might not be apparent from the labeled data alone, as demonstrated in fingerprint-based indoor positioning systems [40].

Techniques like pseudo-labeling and self-training have shown promising results in indoor positioning tasks, such as higher localization accuracy, robustness to noise and an overall improvement of model generalization. The effectiveness of semi-supervised learning can be observed particularly in wireless signal-based localization, where the propagation of signals is governed by the physical environment [40].

In indoor positioning systems which use Wi-Fi or Bluetooth signals, obtaining labeled data typically involves recording signal strength and manually annotating these values with their corresponding coordinates. To address this problem, the author of [45] introduces a Semi-Supervised Generative Adversarial Network (SSGAN) that generates synthetic labeled Wi-Fi fingerprint data, improving landmark localization accuracy by 35% compared to a supervised deep neural network when labeled data is scarce.

To address the above issue, **Ensemble learning** methods have emerged as a solution by combining multiple models, often termed “weak learners” (e.g. decision trees, k-NN) to achieve a high-accuracy meta-model, a superior predictor also known as a "strong learner" [43]. Considering the strengths of different algorithms, ensemble methods aim to reduce both bias and variance in order to produce reliable predictions [46].

Recent trends highlight two prominent ensemble approaches, bagging (e.g. Random Forest) and boosting (e.g. AdaBoost, XGBoost). The general principle of bagging (short for Bootstrap Aggregating) is training multiple base learners in parallel and independently on different samples of the data, getting multiple predictions at the same time, and then aggregating them into a final prediction [47]. Boosting is another prominent ensemble learning technique that focuses on creating a strong predictive model. Unlike bagging, which trains base learners in parallel, boosting training process happens sequentially. More specifically, each new model attempts to correct the errors made by previous models in the sequence. Similarly to the bagging method, the final prediction of the boosted ensemble occurs after combining all the predictions of the models in a weighted approach [46].

Several studies have explored the use of **Random Forest** classifiers to improve localization accuracy for fingerprint-based localization. Random Forest ensures robustness to noise and outliers, which are common in wireless signal measurements, especially in indoor environments [48].

Similarly, **AdaBoost** has been applied to Wi-Fi fingerprint indoor localization systems to enhance precision, particularly in conjunction with noise-reduction techniques such as Probability-One (PONE) Access Point filtering [49]. The results of the study in [49] show a clear improvement of the localization performance with 95.5% accuracy and lowering the 2D errors to 0.25 meters in multi-floor buildings.

It must be noted that, since the deployment of machine learning on resource-constrained IoT edge devices has been rapidly increasing, it is a necessity to focus on computational efficiency as well. Ensemble learning methods, although they tend to provide superior

accuracy, their computational demands could be a challenge. Nevertheless, certain ensemble techniques, particularly gradient boosting frameworks like LightGBM and XGBoost, have been optimized for efficiency and are being explored for use in IoT environments.

LightGBM is designed to be a fast and efficient framework, especially in the case of large datasets. It employs a leaf-wise growth strategy and histogram-based techniques to limit memory usage and accelerate training, as demonstrated in [50]. Similarly, parallel processing and regularization in XGBoost enable the detection of attacks in real time without compromising performance [50]. Thus, these optimizations are suitable for the continuous data generated by IoT devices as they reduce the computational burden while maintaining accuracy.

Deep learning techniques have demonstrated a strong ability to fuse multi-dimensional data sources (such as RSSI, CSI and inertial sensor data) to improve localization accuracy. This ability to fuse heterogeneous streams of information allows for a more comprehensive understanding of the environment, especially in challenging indoor settings. Due to the fact that deep learning architectures can capture spatial and temporal dependencies, they are particularly suitable for fingerprinting-based localization methods [10]. When trained on these spatial fingerprints (e.g. RSSI or CSI values at various reference points), the models learn to associate certain patterns of a signal with particular locations.

Reinforcement learning (RL) has also gained attention for its potential to support autonomous navigation and location prediction. This approach learns a set of behaviors to achieve a specific objective, and it does that by using information about its current state and the environment [43]. Consequently, RL-based systems can provide localization strategies without solely relying on labeled datasets, unlike supervised methods. Traditional supervised learning approaches require large collections of sensor readings and their associated ground truth locations, which can be often costly and time-consuming to acquire.

On the contrary, an RL-based system will learn optimal policies through trial-and-error interactions with the surrounding environment, making it adaptable to real-world uncertainties like multipath effects and variability. In this way, RL can address the main challenge for resource-constrained IoT edge devices by avoiding data labeling, a time consuming and costly process [51]. Consequently, this makes RL particularly valuable in the dynamic environments of IoT and WSN network settings that are either expensive to get labeled data from or have none at all.

For instance, Deep Q-Networks (DQNs) being a specific type of deep reinforcement learning algorithm have been used to dynamically fuse multi-modal sensor data (e.g., RSSI, AoA, PDR) with respect to the environment dynamics and interference. In the context of localization, state space can include the sensor readings and the agent's current position estimate, whereas actions might involve refining the position estimate or deciding on the next move. The RL-IFF framework employed in [51] depicts improved accuracy by optimizing fusion weights via Q-learning, achieving a mean squared error (MSE) of <0.01m in hybrid BLE-based systems, outperforming standalone AoA, RSSI, and PDR methods.

Transfer learning is a robust approach in machine learning whereby models trained in one environment (the source domain) can be adapted and employed in another environment (the target domain) with minimal labeled data [52]. Among the advantages of using Transfer Learning in localization is the reduction in effort in data collection. Instead of needing to build a complete fingerprint database from scratch in a new environment, it is possible to transfer a pre-trained model in another but comparable environment using only partial data. This is particularly beneficial in dynamic indoor settings where conditions such as furniture arrangements cause signal variations and lead to domain shifts that degrade traditional localization models.

Therefore, maintaining localization systems becomes cheaper and requires less time, especially in applications that change frequently, which highlights the practical value of Transfer Learning in real-world localization tasks (e.g. smart homes and industrial IoT) [52]. Furthermore, Transfer learning has proved to be particularly effective in cross-domain RSSI-based positioning systems, whereby the environmental characteristics can

vary considerably. RSSI-based localization, although widely used due to its reliance on existing Wi-Fi infrastructure has been shown to be environment-sensitive, such as variations in furniture layout and the presence of people [53]. This limitation can lead to significant differences in the RSSI signal distributions, which further complicates the process of directly applying a trained model in one environment to another.

One of the ways to address this problem is using Transfer Learning, which as previously stated, offers a way to fine-tune models trained in a source environment to that of a target environment with different RSSI behaviors. For example, a framework outlined in [53] relies on transferring generic RSSI data between different homes, even when the data collection protocols vary, which showcased improved indoor localization performance.

More specifically, the framework proposes a Conditional Generative Adversarial Network (ConGAN)-based augmentation in combination with a transfer learning framework (T-ConGAN). Their approach entails pre-training the model with RSSI samples from multiple houses and subsequently focusing on a 'target' house. By doing so, this method uses general knowledge to generate room-specific signals. Their work demonstrates a remarkable 51% improvement in accuracy of room-level localization around some of the most difficult areas like the staircases, which is particularly useful in the context of healthcare.

This systematic understanding of various algorithms and learning methods illustrates the breadth of research in fingerprint-based indoor localization systems and serves as a reference for future works in this domain.

2.6 Jamming Attacks in WSNs and LoRa

Radio jamming is defined as the deliberate act of transmitting signals on the same radio frequencies used by a target network with the intention of disrupting or preventing legitimate communication. A jamming source can disrupt an entire network or a smaller portion, depending on how powerful is the attacker [25]. Wireless networks are particularly vulnerable to radio jamming attacks due to their straightforward nature, and

also because a jammer can easily launch an attack without any specialized hardware or detailed knowledge of the control system [54].

Jamming is well studied in many radio technologies, such as Wi-Fi, Bluetooth, Zigbee, etc., however, LoRaWAN jamming in particular hasn't been studied for long. This type of attack can undermine multiple IoT applications, such as alarm systems, fire detection, and environmental monitoring. [13]

LoRa jamming refers to the intentional disruption of LoRaWAN communications by exploiting vulnerabilities, with jammers targeting specific channels or spreading factors (SF) in the protocol's physical and MAC layers. Moreover, research has shown that synchronized jamming can negatively affect LoRa communications by flooding the gateway with interference, which leads to a drastic reduce in network throughput [13].

Another consideration when jamming attacks occur in a LoRa network is that the gateway can become a single point of failure. A LoRa gateway can be jammed by malicious attackers making it unable to receive any packets from devices that are connected to the network. [15]

2.6.1 Common Jamming Attack Types

Continuous jamming is a brute-force attack in which an attacker repeatedly floods the channel with high-power noise to keep it busy and completely disrupt all communications in the network by overwhelming legitimate signals. In LoRa networks, this attack is highly effective due to the low data rates and long-range transmission supported by the protocol, as the constant noise blocks both uplink and downlink messages, rendering the network unusable [55].

Unlike selective or triggered jamming, continuous jamming is readily detectable via energy detection mechanisms [25] but poses a serious threat because of its simplicity and catastrophic impact on network availability. Previous research reports that there is around 55% throughput drop when continuous jamming occurs in a LoRaWAN environment

[56]. The simplicity of this attack makes it a common threat in both LoRaWAN and WSN deployments.

Triggered jamming occurs when a malicious node will selectively disrupt communication only upon detecting specific packet signatures or protocol patterns, and hence minimizes its energy expenditure while maximizing interference [57]. For IoT networks, such an attack in most cases targets LoRaWAN gateways by intercepting preamble signals or MAC commands, then flooding the channel with noise in order to corrupt authentic transmissions [58]. Unlike constant jamming, triggered jamming evades simple energy detection countermeasures, and hence is a stealthy threat to low-power networks [59]. Malicious devices can detect the start of a legitimate transmission and quickly transmit interfering signals, jamming only specific parts of the message. This makes the attack more energy-efficient and harder to detect.

Triggered jamming has been studied in [59] and it demonstrated how the long-air time of LoRa messages made this specific type of jamming possible and effective. More specifically, when the trigger jammer was active, only 0.5% of the messages managed to reach the gateway. The energy efficiency of triggered jamming makes it a significant concern in energy-constrained deployments, like IoT networks.

A **random jammer** disrupts network communications by transmitting packets containing unrecognizable data, and in random moments. They can attack either by using (i) specific signal shape i.e. following a specific sequence or (ii) an arbitrary signal shape, as it was discussed by M. Savva [25]. For example, a random jammer can follow a sequence of jamming for x milliseconds, sleep for y milliseconds and repeat. Whereas random jammers that follow the (ii) approach create jamming signals completely randomly and unplanned. One limitation of these type of jammers is that they cannot jam during sleeping mode, thus, they may have the lowest accuracy. In essence, a random jammer introduces unpredictability into the communication channel, so that legitimate nodes struggle to establish or maintain stable connections.

Selective jamming is a sophisticated attack where an adversary targets specific “high” importance packets or messages in a wireless network, by analyzing protocol headers or

payload content to disrupt critical communications [60]. Selective jamming conserves the attacker's energy and reduces the likelihood of detection, making it particularly dangerous for resource-constrained networks like LoRaWAN [61]. For example, the attacker can corrupt key bits during transmission, which can lead to failure in cyclic redundancy check (CRC) validation, causing the gateway to drop the message entirely.

It must be noted that selective jamming makes the detection very challenging for the operator, because it jams only specific devices or messages, leaving the rest of the network untouched. In contrast, triggered and continuous jamming affects all the devices at a certain frequency uniformly, which can easily be marked as jamming, and thus take action (e.g. channel hopping) [21].

Reactive jamming is an advanced form of jamming where an attacker dynamically disrupts communications only upon detecting active transmissions. So, if an action is identified by the jammer, it immediately sends a signal in order to collide with the existing signal that was identified [25]. Moreover, a reactive jammer adapts its strategy based on the network's response to interference, making its detection more challenging with additional effort required [13]. Unlike continuous or selective jamming, reactive jammers monitor the wideband spectrum in real-time in order to emit high-power noise precisely when legitimate signals are detected. As a result, reactive jammers increase their chances and effectiveness of interference because they target frequencies that are currently in use.

In LoRaWAN networks, this attack exploits the protocol's long preamble and CSS modulation, in a way that once a transmission is detected, the jammer rapidly aligns its interference to the same frequency and SF, causing significant packet loss. Unlike RTS/CTS or ACK-targeting reactive jammers used in other wireless networks like Wi-Fi [25], this jammer leverages the extended preamble duration and deterministic nature of LoRa modulation to disrupt communication effectively.

Synchronized jamming attacks represent a potentially more damaging threat to LoRaWAN networks, as they directly exploit the fundamental signal processing mechanisms at the physical layer. In this type of attack, the attacker transmits jamming chirps that are precisely aligned in both time and frequency with the legitimate LoRa

chirps, making it extremely difficult for the LoRaWAN gateway to separate between the intended signal and the interference [15]. One reason why synchronized jamming is effective relies on the ability of jamming chirps to arrive at the gateway with higher power than the legitimate signal, leading the gateway to demodulate the stronger jamming signal instead of the intended data.

Moreover, this technique can bypass existing collision recovery mechanisms that rely on the misalignment of chirp boundaries in the time or frequency domain. Ultimately, one of the most concerning aspects of synchronized jamming is its ability to create a single point of failure in the LoRaWAN network specifically targeting the gateway. Research has empirically demonstrated the effectiveness of synchronized jamming attacks in significantly degrading the performance of LoRa communication, even in the presence of prior countermeasures [15].

Deceptive jamming involves transmitting signals designed to appear as authentic packets to the receiving system, with the primary goal being introducing false information causing the victim to make incorrect conclusions [62]. A modified deceptive jammer has been described by M. Savva [25], which employes an ON-OFF pattern. When in the ON state, the jammer emits interference signals, but unlike a constant jammer, it transmits a specific data sequence, allowing nodes to transfer packets during the OFF state. This approach is still simple to implement, but the deceptive jammer is more challenging to detect than a constant jammer, as it transmits seemingly legitimate packets instead of random bits. Lastly, similar to the constant jammer, the deceptive jammer is energy-inefficient due to its continuous transmission [25].

Advanced Jamming Attack Strategies

The aforementioned jamming techniques, ranging from straightforward continuous interference to more sophisticated deceptive strategies, represent common attack vectors explored in literature. Building upon these fundamental concepts, several studies have tried to create more advanced and adaptive jammers to evaluate their effectiveness in jamming the network.

M. Savva [25] introduces the **complex jammer** which encompasses four jamming behaviors: Constant, Deceptive, Random and Reactive, and can successfully switch between them during attacks. The key findings of the proposed approach conclude that the increased complexity of the jammer makes it much more challenging to detect, as it demonstrated the lowest detection accuracy in experiments. Besides detection, complex jammer's mobility-like behavior of switching strategies has also complicated the localization algorithm (MMLAW) used in the research.

A wormhole attack is a network-layer attack where two or more malicious nodes collude to secretly relay packets between distant parts of a network. One of the malicious devices receives normal messages and sends them over to the second device though a low-latency link. To carry out an attack, the second device is responsible to replay them in a different area of the network [21].

In traditional wireless sensor networks (WSNs), this attack creates fake short-cut routes, leading to routing loops, energy depletion, man-in-the-middle attacks or can also be used simply to convince two distant nodes that they are neighbors by relaying packets between the two of them [63].

However, in LoRaWAN, which uses a star-of-stars topology, classic wormhole attacks are ineffective due to the lack of multi-hop routing and replay attack protection via Message Integrity Codes (MICs). Instead, a **novel wormhole-based selective jamming attack** was investigated and executed by Aras et al. (2017) [21] : one malicious node (a sniffer) captures legitimate LoRa transmissions while another (a jammer) simultaneously blocks them at the gateway, allowing delayed replay of recorded messages to manipulate sensor data undetected. By replaying normal messages during a jamming attack, the attacker can simultaneously block alerts and make the system appear normal.

Attack Type	Description	Impact on LoRa	Detection Difficulty	Energy Efficiency
Continuous Jamming	Repeatedly floods channel with noise.	Blocks all communication (uplink/downlink), high throughput drop.	Easy	Low
Triggered Jamming	Disrupts on specific signal pattern or event (e.g., preamble)	Corrupts selective transmissions, often targets gateways.	Hard	Medium
Random Jamming	Unpredictable noise.	Causes unstable links, irregular interference.	Medium	Variable (depends on ON-OFF pattern)
Selective Jamming	Targets important packets.	Corrupts key data, affects specific devices/messages.	Very Hard	High
Reactive Jamming	Dynamically jams any ongoing transmissions.	Exploits LoRa preamble and CSS modulation, causes high packet loss.	Hard	High
Synchronized Jamming Attacks	Transmits precisely aligned jamming chirps.	Can be indistinguishable from legitimate signals, may lead to gateway failure.	Hard	Medium - High
Deceptive Jamming	Transmits 'fake' authentic signals.	Injects false data, doesn't fully block real traffic.	Medium	Low – Medium

Table 2.6.1 Overview of Jamming Attack Types and Their Characteristics in LoRaWAN

2.7 Wi-Fi Jamming Limitations in LoRaWAN

Although numerous studies have already been conducted on jamming attacks in traditional Wireless Sensor Networks (WSNs) and there are a lot of common vulnerabilities that overlap with LoRaWAN, many Wi-Fi jamming techniques often prove ineffective against LoRaWAN due to key architectural and protocol-level differences.

Wi-Fi depends on mechanisms like carrier-sense multiple access with collision avoidance (CSMA/CA), orthogonal frequency-division multiplexing (OFDM), Direct Sequence Spread Spectrum (DSSS) and Frequency Shift Keying (FSK). However, LoRaWAN takes a fundamentally different approach and uses a simple ALOHA-based random access mechanism [19] in conjunction with Chirp Spread Spectrum (CSS) modulation. In addition, LoRaWAN utilizes the adaptive data rate (ADR) function to dynamically adjust

spreading factors (SFs) and maintain connectivity in noisy environments. Added on the previous features, LoRaWAN's heavy reliance on uplink traffic and duty-cycle restrictions may complicate jamming attacks and detection mechanisms even more [19].

An example of a specific-function jammer that was described in [25] for Wi-Fi (IEEE 802.11) networks is the channel-hopping jammer. Frequency hopping is a common anti-jamming technique, but its implementation might be more straightforward in some WSN protocols compared to standard LoRaWAN [64]. This jammer relies on MAC-layer mechanisms (e.g. CSMA) which are typical in Wi-Fi but largely absent in LoRaWAN.

Finally, the network topology can play a role. LoRaWAN typically uses a star topology, where all devices communicate with a central gateway [15]. In contrast, traditional WSNs can use more complex topologies like mesh networks, that might offer some resilience to jamming because data can be sent through alternative routes if a node is attacked. Therefore, some adaptations and redefinitions of jamming attacks in Wi-Fi networks along with their detection strategies, must be taken into consideration to fit the unique characteristics of low-power wide-area networks (LPWAN).

Feature	**LoRaWAN**	**Traditional WSNs**
Frequency Bands	Sub-GHz ISM bands (regional variations)	2.4 GHz ISM band, Sub-GHz (protocol dependent)
Modulation	CSS	DSSS, FSK, GFSK, O-QPSK
Typical Data Rates	Low (0.3 kbps to 50 kbps)	Higher (~250 kbps for Zigbee, up to 2 Mbps for BLE)
Typical Transmission Range	Several kilometers (urban), up to 15 km (rural)	Shorter range (tens to hundreds of meters)
Network Topology	Star (mostly)	Star, Mesh, Tree
Susceptibility to Long On-Air Time Attacks	Higher (longer airtime)	Lower generally (shorter airtime)
Energy Constraints	High (battery-powered devices common)	High (sensor nodes often battery-powered)

Table 2.7.1: Comparative Overview of Key Characteristics Between LoRaWAN and Traditional WSN Technologies

Chapter 3

Literature Review and Related Work

Chapter 3 delves into the existing literature that has been published over the past decade, focusing on the growth of WSN and IoT technologies in the context of indoor localization and their associated security issues. Emphasis is placed on jamming detection and localization, but also broader anomaly detection. This review specifically refers to studies about LoRaWAN, examining this technology's current state with respect to jamming detection and localization.

While numerous papers have broached anomaly and jamming detection in both WSN and IoT networks, the available literature on explicit jamming localization remains insufficient. The majority of such studies utilize machine learning-based frameworks to demonstrate high accuracy in detecting anomalies or generic attacks, yet they do not extend to physically localizing the attacker. This limitation is even more evident in LoRa and LoRaWAN-based systems, where research on indoor localization has primarily been focused on improving the accuracy of positions through RSSI fingerprinting or ensemble learning/hybrid methods, while attack-specific localization is rarely explored. This represents a significant research gap that my study will attempt to address: implementing and experimenting with jamming localization techniques in LoRa-based IoT and WSN environments.

Yang et al. (2021) [65] provide a comprehensive survey of indoor localization in WSN and IoT networks. The authors highlight that ML algorithms, particularly deep learning, improve localization accuracy under NLOS conditions, while filter-based, such as Kalman Filters, contribute to real-time tracking. They verify that hybrid methods (e.g., CNN-LSTM) provide sub-meter localization errors in indoor environments, mitigating multi-path effects. However, the survey does not address any security threats and focuses solely on benign environments.

3.1 Anomaly Detection in WSN and IoT

Several attempts have been made which aim to explore more advanced security measures since the escalating threat of cyberattacks on IoT networks. Ullah and Mahmoud [66] propose a deep learning-based anomaly detection model for IoT networks, focusing on identifying cyberattacks such as DDoS, malware, and data theft. They argue that traditional methods struggle with unpredictable network technologies, and hence they propose multiclass classification model with CNNs, by utilizing 1D, 2D, 3D configurations along with transfer learning. Model validation is performed using multiple open-source IoT datasets, such as BoT-IoT [67] and IoT-23 [68], achieving a high accuracy in detecting anomalies, up to 99.97% and a low false alarm rate. Although the paper's framework demonstrates robust anomaly detection for network and application-layer attacks, it does not address physical-layer attacks such as jamming attacks, nor does it investigate localization-based intrusions.

Boush et al. (2025) [4] tackle this issue by presenting an efficient IoT attack detection system, "IoT-SecureNet" which includes efficient feature extraction and ensemble machine learning (ML) algorithms, i.e., XGBoost, LightGBM, and CatBoost. The ensemble technique leverages the strengths of each model while efficient feature extraction ensures that the model is trained only on the most crucial features in order to optimize detection accuracy. To combine all the models' detection results (attack vs. no attack), two methods are outlined: i) Majority voting technique, which for binary classification jobs is the simplest and ii) Weighted Averaging, which assigns weight to each model based on its performance during the training process. Overall, this study

demonstrates the potential of ensemble ML techniques for enhancing security in IoT networks.

3.2 Anomaly Detection in LoRa

Although indoor localization is not specifically covered by Babazadeh (2020) [69], their edge-based anomaly detection method for LoRa-based WSNs includes useful details regarding resource-efficient signal processing. The study's approach to timestamped event logging and compression-rate thresholds (e.g. filtering compression rate, FCR<50% → anomaly) demonstrates how edge devices can preprocess data to reduce bandwidth overhead. With the aid of RSSI/CSI measures, this technique may be potentially modified and adaptable for indoor interference detection. This study, however, prioritizes broad anomaly detection in environmental monitoring rather than specific jamming detection or localization, which differs from my thesis' scope. Regardless, their application of Channel Activity Detection (CAD) towards packet collision reduction draws attention to trade-offs between real-time responsiveness vs. detection accuracy, a consideration equally relevant to dense indoor IoT deployments.

Kurniawan & Kyas (2022) [70] suggested a ML-based system for generic anomaly detection in LoRaWAN gateways, evaluating 11 algorithms (including CBLOF, PCA, and Isolation Forest) on real-world network traffic. Their methodology focuses on packet-level analysis (join-request and data-request patterns) using RSSI/LSNR features to identify threats like DoS, replay attacks, and MITM. The main results of their work indicated that CBLOF achieved the highest performance scores when detecting anomalies, whereas PCA and HBOS were computationally efficient when working with large datasets. Their work differs from my thesis focus as it examines general network anomalies rather than specifically targeting jamming signals or localizing the jammer.

Ensemble learning techniques have also been explored in the field of indoor localization. K. Hettiarachchige [43] reviews the application of ensemble machine learning techniques to improve LoRa-based indoor localization systems by using RSSI data. More specifically, the study revolves around optimizing localization accuracy and predictive performance by evaluating several supervised ensemble methods (Random Forest,

Gradient Boosting, LightGBM, Bayesian post-hoc regularization). The methodology involves testing and evaluating the mentioned algorithms under different hyperparameters, which leads to the proposed model: an ensemble model combining Gradient Boosting and LightGBM with a Voting Classifier. By achieving accuracy at around 91%, the ensemble approach outperformed individual novel models and harmonizes the strengths of both models, leading to a more robust and resilient predictive tool.

Senol et al. (2024) [71] tackle the critical challenge of securing LoRa-based IoT networks by detecting tampered radio-frequency transmissions using ML strategies. The study detects generic signal tampering (jamming, spoofing, replay, unspecified distortions etc.) but does not provide granular classification. Their approach uses image-based anomaly detection, converting frequency signals into visual representations to recognize deviations in frequency patterns. They employ five algorithms commonly used in image processing and computer vision: Local Outlier Factor (LOF), Isolation Forest, Autoencoder, Variational Autoencoder (VAE), and Principal Component Analysis (PCA) on a dataset of real-world transmission recordings of normal and abnormal signal images.

LOF achieves the highest accuracy (97.78%) while Isolation Forest is the least reliable with 84.49% accuracy. The strong performance of LOF validates the broader effectiveness of density-based outlier detection in RF anomaly detection, aligning with previous findings from Kurniawan & Kyas (2022) [70], where CBLOF, a clustering-enhanced LOF variant, also performed well.

Their use of image-based data distinguishes their work, as it captures subtle anomalies often missed by traditional signal analysis when complex patterns are present. Potential limitations of this framework, as stated by the authors, can be the dependence of most algorithms on the quality and quantity of the training dataset. The research also omits real-time deployment challenges, such as computational latency in resource-constrained IoT devices, a critical factor for indoor localization systems.

3.3 Jamming Detection in LoRaWAN

Aras et. al [21] in their paper "Selective Jamming of LoRaWAN using Commodity Hardware" empirically prove that LoRaWAN's design choices i.e. long packet air-time (due to CSS modulation) and unencrypted MAC-layer headers, enable practical, low-cost jamming attacks. The authors demonstrate how to exploit these characteristics of LoRa while using low-cost hardware for three attack variants: triggered jamming, selective jamming, and a combined selective jamming-wormhole attack. The key findings of their study reveal that LoRaWAN is extremely vulnerable to jamming, achieving over 98% success rates of selective jamming over spreading factors (SFs). A notable discovery is the inverse relationship between SF and jamming effectiveness, i.e., higher SFs (e.g., SF12) are easier to jam (longer air-time) but require stronger jamming signals. Similarly, lower SFs (e.g., SF7) which have shorter air-time, evade jamming if the attacker's reaction is too slow. Although this paper does not address jamming detection or localization, it has proven exploitable weaknesses in LoRaWAN and validated jamming attacks in real-world testing.

The first official work on jamming detection for LoRaWAN, was conducted by Danish et al. (2018) [72], who focused on the join procedure, and proposed a Network Intrusion Detection System (NIDS) which uses the Hamming distance between consecutive join-request messages. Hamming distance is a metric that counts the differences between bit positions in two equal-length binary strings and thus can point out anomalies that signal jamming events. As a result, their system attains 98% accuracy with a 5% false alarm rate.

In a similar manner, Martinez (2021) [73] focuses on jamming attacks in LoRaWAN networks and addresses the problem from a more holistic point of view. The author suggests two primary tools: i) a mathematical model to estimate the impact of jamming on LoRaWAN performance and ii) an extended ns-3 simulation module for evaluating realistic scenarios. Key findings of the study reveal that jamming severely degrades network performance and interferes communication, especially in terms of throughput and energy usage. Retransmission mechanisms can be applied to mitigate some impacts, but for IoT networks which rely on battery-powered nodes, the retransmission-energy

tradeoff must also be taken into account. An interesting proposition is the use of LSTM networks for jamming detection, achieving high performance in identifying anomalies, and which will be evaluated on our approach later. This thesis does not go a step further to address jamming localization, a gap my research aims to tackle.

3.4 Jamming Detection and Localization in WSNs and IoT

In contrast to the limited research on jamming detection in LoRa-based systems, in the realm of 'traditional' Wireless Sensor Networks (WSNs), much work has been carried out on the potential of jamming detection. Upadhyaya et al. (2019) [74] evaluated several Machine Learning algorithms (decision tree, random forest, SVM) for jamming detection in WSNs, relying primarily on RSSI.

In the domain of Vehicular Ad-Hoc Networks (VANETs), the author in [75] also utilized a ML approach (Random Forest) for detecting jamming by analyzing metrics such as Channel Busy Ratio (CBR), Packet Delivery Ratio (PDR), and Inactivity Time (IT). Given the highly mobile and dynamic nature of VANETs, the suggested technique demonstrated excellent efficacy with an accuracy of up to 97%.

Another study carried out by Osanaiye et. al (2018) [76] presents a statistical methodology by implementing the Exponentially Weighted Moving Average (EWMA) algorithm to detect jamming attacks in WSNs. The key benefit of EWMA is that it uses aggregation of recent data and historical (past) data and can easily identify small changes in time-series. The proposed algorithm monitors the Inter-Arrival Time (IAT) of packets as an evaluation metric to identify abnormalities caused by jamming with a 100% detection rate, according to the authors. This result highlights the effectiveness of lightweight statistical techniques in physical-layer interference detection in WSNs, though such methods remain underexplored in more complex LPWAN settings like LoRaWAN.

M. Savva (2024) [25] addresses the critical need for practical anti-jamming methods and proposes a comprehensive framework for detecting, localizing, and recovering from jamming attacks in WSN and IoT networks. He focuses on enhancing security against

intelligent jammers (constant, deceptive, random, reactive) that employ more advanced ML algorithms to attack a network. A notable contribution is the introduction of a novel adaptive behavior "complex jammer" which highlights the system's ability to handle evolving attack strategies.

Concerning the detection part, the author employs a lightweight fuzzy logic intrusion detection system (FLIDS) specifically designed to detect jamming attacks. During the localization phase, the modified Multilateration Localization Algorithm with Weights (MMLAW) combines metrics from the data link and network layers, such as Expected Transmission Count (ETX) and retransmissions in order to pinpoint the attacker's location. Lastly, the recovery phase includes network-layer rerouting techniques and node blacklisting within the affected area. However, the author explicitly excludes LoRa and LoRaWAN from the scope of the study, which my thesis' scope explicitly includes.

Acronyms for Table 3.1 columns:

IL – Indoor Localization
AD – Anomaly Detection
JD – Jamming Detection
JL – Jamming Localization

Authors	Technology	Technique	Key Finding	Methodology	Limitations	IL	AD	JD	JL
Yang et al. (2021) [65]	WSN/IoT Networks	Survey of ML/filter-based methods	ML and filters improve accuracy in NLOS/dynamic environments	Survey	1. No security threats mentioned	✓	✗	✗	✗
Ullah et al. (2021) [66]	IoT Networks	Convolutional Neural Networks (1D, 2D, 3D), Transfer Learning	Achieved 99.97% accuracy in multiclass attack detection.	Feature selection, CNN training, transfer learning on merged IoT datasets.	No physical-layer attacks (e.g., jamming).	✗	✓	✗	✗
Boush et al. (2025) [4]	IoT	Ensemble ML (XGBoost, LightGBM, CatBoost)	Ensemble model achieved 96.2% accuracy in attack detection	Supervised ML on IoT network traffic data with ensemble learning and feature extraction	No real-time evaluation.	✗	✓	✗	✗
Babazadeh (2020) [69]	LoRa WSN	Edge analytics + CAD scheduling	Achieved 50% bandwidth reduction via compression-rate thresholds	Sensor-side data compression, Centralized anomaly reconstruction	1. No multipath analysis. 2.Scalability constraints in dense networks	✗	✓	✗	✗

			(FCR<50% = anomaly)						
Kurniawan & Kyas (2022) [70]	LoRaWAN	Various ML algorithms	CBLOF achieved the highest F1 (0.92) PCA most efficient	Packet-level RSSI/LSNR analysis and 11 ML models	No multipath tests	✗	✓	✗	✗
K. Hettiarach-chige (2024) [43]	LoRa	Ensemble ML (Gradient Boosting, LightGBM, VotingClassifier)	Achieved 91% accuracy for localization	Supervised ML on RSSI data with extensive hyperparameter tuning	No real-world evaluation	✓	✗	✗	✗
Senol et al. (2024) [71]	LoRa	Machine Learning (LOF, VAE, PCA, Autoencoder, Isolation Forest)	Achieved ~ 98% accuracy (LOF) in detecting tampered RF signals.	Image-based ML (LOF, VAE, PCA) on spectrograms from HackRF/MKRWA N1310 testbed.	1. No real-time evaluation. 2. No attack-type granularity	✗	✓	✗	✗
Danish et al. (2018) [72]	LoRaWAN	Hamming distance-based NIDS	98% detection accuracy with 5% false alarms	Analyzing bit differences in join-request messages	Only detects join-request jamming	✗	✓	✓	✗
Martinez (2021) [73]	LoRaWAN	Mathematical modeling & LSTM detection	LSTM achieved high detection accuracy	Simulation-based evaluation with ns-3 module	No real-world validation	✗	✓	✓	✗
Upadhyaya et al. (2019) [74]	WSN	ML (DT, RF, SVM)	Comparative performance evaluation of ML classifiers	RSSI-based feature analysis	Limited adaptability to new environments.	✗	✓	✓	✗
VANETs study) [75]	VANET	Random Forest	97% detection accuracy	CBR, PDR, IT metrics analysis	1. Vehicle-specific dynamics 2. High mobility focus	✗	✓	✓	✗
Osanaiye et al. (2018) [76]	WSN	EWMA statistical method	100% detection rate (claimed)	IAT monitoring for anomalies	No real-world validation	✗	✓	✓	✗
Savva (2024) [25]	WSN / IoT Networks	FLIDS + MMLAW (using ETX, retransmissions)+ recovery routing	Proposed full framework (detection-localization-recovery) for jamming,	Fuzzy Logic IDS for detection, Multilateration with Weights for localization, Recovery via rerouting/blacklisting	LoRaWAN explicitly excluded	✓	✓	✓	✓

Table 3.1: Literature Review Summary

Chapter 4

Jamming Detection Implementation

This section presents the implementation of several Machine Learning and Deep Learning methods for performing jamming detection on a LoRa-based dataset. The dataset is public and can be accessed on GitHub [77]. The goal is to detect and analyze jamming attacks in LoRa networks using network metrics like PLR, RSSI and SNR. The dataset is described in Table 4.1.1:

4.1 Dataset Description

Total Samples	31,919
Time Range	February 2, 2023 - February 9, 2023
Recording Interval	Every minute
Data Format	CSV
Devices	5 static LoRa end devices (Mote01 – Mote05)
Features	14 total (including LoRa PHY settings, RSSI, SNR, Jamming label, FCnt)
Jamming Label	Binary label: 1 = Normal , -1 = Jamming
Jamming Period	14:15 until 17:00 on February 8, 2023

Table 4.1.1: Chirp Dataset Overview

Dataset Limitations:

While the dataset provides essential communication metrics like RSSI, SNR, and frame counters (fcnt), it lacks location coordinates, or any location information at all, which makes localization of jamming sources infeasible. Therefore, on this dataset only jamming detection can be performed, which is later evaluated on the ground truth values (known jamming period). Additionally, the dataset is highly imbalanced, with the jamming event occurring for only a few hours on a single day, while the remaining data reflects normal operation.

4.2 Packet Loss Ratio (PLR) Calculation

Despite that the dataset contains RSSI and SNR which can be used for identifying possible attack patterns, those metrics weren't enough for accurate results. Therefore, it was necessary to determine PLR through the frame counter (fcnt) values.

PLR was chosen as a key measure for jamming detection because it directly measures the reliability of communication, which is severely impacted by jamming attacks. Since jammers disrupt transmissions and cause more packet loss, the measure is a good indicator of the malicious interference, complementing RSSI and SNR for increased accuracy in detection.

PLR is defined as the ratio of lost packets to the total expected packets (over a given period):

$$PLR = \frac{\text{Expected Packets} - \text{Received Packets}}{\text{Expected Packets}}\ x\ 100$$

The calculation of this metric was initially per-device over fixed time intervals. Then, PLR was calculated for the whole system (aggregating PLR across all motes) which yielded to better results. This might happen because per-device PLR can be prone to outliers from transient local interference, whereas combining data from all motes can amplify the persistent signal of jamming and average out random noise.

The other two metrics considered for detecting jamming are the Signal-to-Noise Ratio (SNR) and Received Signal Strength Indicator (RSSI). SNR quantifies signal quality by measuring how much the desired signal stands out from noise. We expect SNR to drop when jamming interference is active.

$$\text{SNR (dB)} = 10 \times \log_{10}\left(\frac{P_{\text{signal}}}{P_{\text{noise}}}\right)$$

Where:

- P_{signal} is the received signal power
- P_{noise} is the background noise power.

RSSI measures absolute received power, including both signal and noise. While useful for detecting overpowering jammers, it cannot distinguish intentional jamming from natural signal attenuation, and it must be integrated with other metrics for reliable detection.

$$\text{RSSI (dBm)} = 10 \times \log_{10}(P_{\text{received}})$$

As discussed, the PLR was calculated initially per device. However, it was later concluded that using the collective data of all devices provided a better approximation. The reasoning behind the calculation of PLR is identical in both approaches, differing only in the dataset split utilized: per-device vs. system-wide aggregated data.

The process begins by reading and pre-processing an aggregated data set of time-stamped transmission logs of all the devices involved. The data is sorted chronologically and grouped by device within each interval of fixed duration (15 minutes). For every such interval, the algorithm extracts the minimum and maximum frame counter ($fcnt$) values for each device to estimate the number of packets that should have been transmitted. The expected packet count for a device in a given interval is calculated as:

$$\text{Expected}_i = \text{Last\_fcnt}_i - \text{First\_fcnt}_i + 1$$

The actual received packets are counted directly from the data, and lost packets are calculated as:

$$\text{Lost}_i = \text{Expected}_i - \text{Received}_i$$

The expected and received counts are then accumulated across all devices to obtain a total expected and total received value for that interval. The system-wide PLR is calculated using the following formula:

$$\mathrm{PLR}_{\mathrm{system}} = \left(\frac{\sum_{i=1}^{N} \mathrm{Lost}_i}{\sum_{i=1}^{N} \mathrm{Expected}_i} \right) \times 100$$

Where:

- N is the number of active devices in the interval.

For example, assume that during a 15-minute interval, a device had a first frame counter of 102 and a last frame counter of 108. If only 5 packets were actually received during this interval, the calculations would be as follows:

$$\mathrm{Expected_i} = 108 - 102 + 1 = 7$$
$$\mathrm{Lost_i} = \mathrm{Expected_i} - \mathrm{Received_i} = 7 - 5 = 2$$
$$\mathrm{PLR_i} = \left(\frac{2}{7}\right) \times 100 \approx 28.57\%$$

This process is repeated per device, and system-wide PLR (for that interval) is computed by summing expected and received packets across all devices.

ALGORITHM 4.2.1: SYSTEM-WIDE PLR CALCULATION

Input: Chirp dataset df, interval duration Δt (in minutes)
Output: Packet Loss Ratio (PLR) for each interval
Initialize start_time ← first timestamp, end_time ← start_time + Δt
while start_time < last timestamp **do**
    interval_data ← filter entries with Local Time ∈ [start_time, end_time)
    total_expected, total_received ← 0
    for each device in interval_data:
        Calculate expected = max(fcnt) − min(fcnt) + 1
        Calculate received = number of packets
        Accumulate total_expected and total_received
    end for
    PLR ← $\left(\frac{\text{total_expected} - \text{total_received}}{\text{total_expected}} \right) \times 100$
    Store [start_time, end_time, PLR]
    start_time ← end_time; end_time ← start_time + Δt
end while
Return PLR values

RSSI and SNR preprocessing

For each interval, in addition to PLR, the average Received Signal Strength Indicator (RSSI) and Signal-to-Noise Ratio (SNR) are computed by averaging all values recorded during that time. Each interval is then labeled as either jamming (-1) or normal (1) based on whether its start time falls within the known jamming period (in this case, between 14:15 and 17:00 on February 8, 2023).

$$\text{Avg\_RSSI} = \frac{1}{M}\sum_{j=1}^{M} \text{RSSI}_j \qquad \text{Avg\_SNR} = \frac{1}{M}\sum_{j=1}^{M} \text{SNR}_j$$

Finally, the results consisting of the interval start and end times, total expected and received packets, PLR percentage, average RSSI and SNR, and the jamming label are written to a CSV file.

4.3 Jamming Detection Methods:

Several machine learning models were employed for jamming detection, with the LSTM Autoencoder pseudocode described in more detail due to its sequence-based architecture. For the remaining models, a similar detection pipeline was followed, and only the key differences in parameters are summarized in their corresponding pseudocodes.

1. **LSTM Autoencoder**

The LSTM Autoencoder was utilized to detect jamming by learning normal patterns from the multivariate time series data of RSSI, SNR and PLR. Training was done using non-jamming periods, and instances of jamming were used for the testing part. The data was normalized and split into overlapping sequences of length T=10, which were used as input to the model. After training the model for 100 epochs with MSE and Adam optimizer, it reconstructed all input sequences. Anomalies were identified based on high reconstruction error, calculated as:

$$[\text{MSE}_\text{i} = \frac{1}{T \cdot F}\sum_{t=1}^{T}\sum_{f=1}^{F}\left|x_{t,f}^{(i)} - \widehat{x_{t,f}^{(i)}}\right|]$$

Where:

- i is the index for the sequence (i.e., the i-th time window).
- T is the number of time steps in the sequence (10 if the sequence length is 10).
- F is the number of features (RSSI, SNR, PLR → F = 3).

A threshold set at the 98th percentile of these errors was used to flag jamming events.

ALGORITHM 4.3.1: LSTM AUTOENCODER FOR JAMMING DETECTION

Input: Chirp dataset df, interval duration Δt (in minutes)
Output: Packet Loss Ratio (PLR) for each interval
Normalize PLR, RSSI, and SNR using MinMaxScaler
Define jamming_start and jamming_end timestamps
Split data:
    train_data ← entries outside jamming period
    full_data ← entire dataset
Create train_sequences:
    for i in 0 to len(train_data) − T:
    sequence ← train_data[i : i+T]
    append to train_sequences
Define LSTM Autoencoder:
    Encoder: LSTM(64) → LSTM(32)
    Decoder: RepeatVector(T) → LSTM(32) → LSTM(64) → TimeDistributed(Dense(3))
Train model on train_sequences
Create all_sequences from full_data using sliding window of length T
Reconstruct all_sequences using trained model
Compute reconstruction error (MSE) for each sequence
Set anomaly threshold ← 98th percentile of training MSE
Detect anomalies:
    for each sequence error mse_i:
        if mse_i > threshold: flag as anomaly
        else: flag as normal
Return anomaly labels

2. One-Class Support Vector Machine (SVM)

The One-Class SVM was applied as an unsupervised anomaly detector trained exclusively on non-jamming data. The input consisted of normalized PLR, RSSI, and

SNR values. The model used an RBF kernel with parameters $\gamma = 1$ and $nu = 0.01$ to learn the distribution of normal data. Once trained, the model predicted on the entire dataset, assigning +1 to normal points and −1 to anomalies. Predictions were converted to binary labels (0 for normal, 1 for anomaly).

ALGORITHM 4.3.2: SVM FOR JAMMING DETECTION

Input: Normal training data X_train, full dataset X_full
Output: Anomaly labels (0 = normal, 1 = anomaly)
1 Define One-Class SVM with kernel = 'rbf', gamma = 1, nu = 0.01
2 Train SVM model on X_train
3 Predict labels on X_full: +1 = normal, −1 = anomaly
4 Convert predictions: anomaly = 1 **if** label == -1 **else** 0
5 Return anomaly labels

3. **Binary Logistic Regression (BLR)**

Binary Logistic Regression was employed as a supervised classification model to separate jamming and non-jamming intervals based on PLR, RSSI, and SNR. The features were normalized, and jamming periods were labeled as 1, and the rest as 0. The model was trained with class balancing using a regularization parameter $C = 0.5$ and a maximum of 500 iterations. Predictions were made as probabilities, and a decision threshold of 0.8 was applied to classify jamming.

ALGORITHM 4.2.3: BLR FOR JAMMING DETECTION

Input: Normalized dataset X, ground truth labels y
Output: Anomaly labels (0 = normal, 1 = anomaly)
1 Define logistic regression with class_weight = 'balanced', C = 0.5, max_iter = 500
2 Train model on X, y
3 Predict probability scores p_i for each sample
4 Set threshold $\tau = 0.8$, classify: $\text{anomaly}_i = 1$ **if** $p_i > \tau$, **else** 0
5 Return anomaly labels

4. **Extreme Gradient Boosting (XGBoost)**

XGBoost gradient boosting classifier was used to classify jamming with normalized multivariate input of PLR, RSSI, and SNR. The classifier used 100 trees with a learning rate of 0.05, max depth of 6, and class balancing weight of 100 to counter label imbalance. The classifier was trained over the full labeled dataset and predicts probabilities, which

were thresholded at 0.5 for anomaly detection. Anomalies corresponded to high predicted probability of class 1 (jamming).

ALGORITHM 4.3.4: XGBOOST FOR JAMMING DETECTION

Input: Normalized dataset X, ground truth labels y

Output: Anomaly labels (0 = normal, 1 = anomaly)

1 Define XGBoost classifier with scale_pos_weight = 100, max_depth = 6, n_estimators = 100, learning_rate = 0.05
2 Train model on X, y
3 Predict probability scores p_i **for** each sample
4 Set threshold $\tau = 0.5$, classify: $\text{anomaly}_i = 1$ **if** $p_i > \tau$, **else** 0
5 Return anomaly labels

5. K-Means Clustering

K-Means was applied in an unsupervised setting to cluster normalized data into groups based on similarity in PLR, RSSI, and SNR. 10 clusters were created using the k-means++ initialization. The least occurring cluster was considered to represent jamming conditions, while the most frequent cluster corresponded to normal behavior.

ALGORITHM 4.3.5: K-MEANS FOR JAMMING DETECTION

Input: Normalized dataset X

Output: Anomaly labels (0 = normal, 1 = anomaly)

1 Define K-Means with n_clusters = 10, init = 'k-means++', max_iter = 300, tol = $1e^{-4}$
2 Train model on X
3 Predict cluster labels for each sample
4 Identify anomaly cluster as the one with the fewest members
5 Assign anomaly = 1 **if** in anomaly cluster, **else** 0
6 Return anomaly labels

6. Random Forest (RF)

Random Forest was used as a supervised ensemble classifier to detect jamming using the same three features. The model was trained with 100 decision trees using $class\_weight = 'balanced'$ to handle class imbalance. Input features were normalized, and the jamming interval was labeled with '1'. The model output was class probabilities, and samples with probability > 0.5 were classified as jamming.

ALGORITHM 4.3.6: RF FOR JAMMING DETECTION

Input: Normalized dataset X, ground truth labels y

Output: Anomaly labels (0 = normal, 1 = anomaly)

1 Define Random Forest with n_estimators = 100, class_weight = 'balanced', random_state = 42

2 Train model on X, y

3 Predict probability scores p_i for each sample

4 Set threshold τ = 0.5; classify: $\text{anomaly}_i = 1$ **if** $p_i > \tau$, **else** 0

5 Return anomaly labels

7. 1D Convolutional Neural Network (1D-CNN)

The 1D Convolutional Neural Network was trained to detect jamming patterns by learning temporal dependencies in overlapping sequences of length $T = 10$. The CNN architecture included stacked Conv1D and MaxPooling1D layers followed by fully connected layers, which were trained employing binary cross-entropy loss and Adam optimizer. The model predicted binary labels on all the sequences, with high confidence jamming labels being labelled as "jamming".

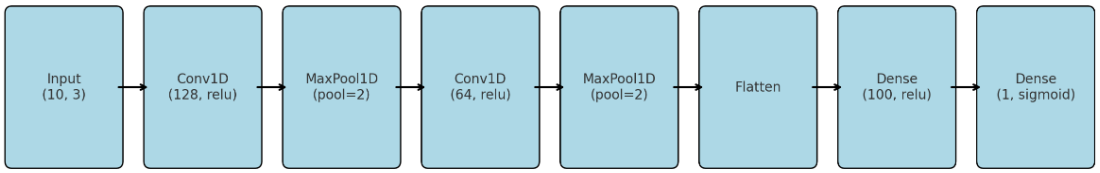


Figure 4.3.1 : 1-D CNN Architecture

ALGORITHM 4.3.7: 1D CNN FOR JAMMING DETECTION

Input: Normalized dataset X, ground truth labels y

Output: Anomaly labels (0 = normal, 1 = anomaly)

1 Create multivariate sequences of fixed length T from X and y

2 Split sequences into training and testing sets

3 Define CNN model: Conv1D(128) → MaxPooling1D → Conv1D(64) → MaxPooling1D → Flatten → Dense(100) → Dense(1, sigmoid)

4 Train CNN using binary crossentropy and class weights

5 Predict anomaly probabilities and apply threshold τ = 0.5: $\text{anomaly}_i = 1$ **if** $p_i > \tau$, **else** 0

6 Return anomaly labels

A trial-and-error procedure was used to select all the jamming detection models' hyperparameters. Each method was tuned individually to identify the parameter combination that yielded the highest performance metrics during evaluation. The final selected parameters for each algorithm are summarized in Table 4.3.1.

Method	Type	Key Parameters
LSTM Autoencoder	Unsupervised Deep	Sequence length $T = 10$ epochs=100 optimizer=Adam loss=MSE encoder=LSTM(64→32) decoder=RepeatVector → LSTM(32→64) → Dense(3)
Logistic Regression	Supervised	$C = 0.5$ class_weight='balanced' max_iter=500
Random Forest	Supervised Ensemble	n_estimators=100 class_weight='balanced' random_state=42
XGBoost	Supervised Boosted	n_estimators=100 learning_rate=0.05 max_depth=6 scale_pos_weight=100
One-Class SVM	Unsupervised	kernel='rbf' gamma=1 nu=0.01
1D-CNN	Supervised Deep	Sequence length $T = 10$ loss=binary_crossentropy optimizer=Adam
K-Means	Unsupervised	n_clusters=10 init='k-means++' max_iter=300 tol=1e-4

Table 4.3.1: Jamming Detection Methods and Their Parameters

Chapter 5

Jamming Localization Implementation

This chapter presents the implementation of four jammer localization methods developed using LoRa-based drone communication data. This study utilizes the "Drone Communication Dataset" which is publicly available [78] and designed for research purposes in areas such as anomaly detection and cybersecurity in drone networks. The data was collected from a simulated drone communication network spanning from November 1, 2019 to December 31, 2024, with data recorded on an hourly basis.

5.1 Dataset Description

Total Samples	44,016
Time Range	November 1, 2019 - December 31, 2024
Recording Interval	Hourly
Data Format	CSV
Features	26 input features, 8 multilabel anomaly labels

Table 5.1.1: Dataset Overview

For the objective of this study, a subset of relevant features was used primarily to achieve jammer localization. The columns (features) that were considered are summarized below.

Feature Name	Description	Unit / Type
signal_strength	Received Signal Strength Indicator (RSSI)	dBm (decibels)
packet_loss_rate	Percentage of lost communication packets	%
signal_noise_ratio	Signal-to-Noise Ratio (SNR) of the received signal	dB (decibels)
drone_gps_coordinates	Geographic location of the drone (latitude, longitude)	Tuple (float, float)
drone_identification	Unique id for each drone	Integer
communication_protocol	Communication protocol filtered for LoRa only	Categorical (LoRa)
label_jamming	Binary indicator (1 = jammed, 0 = normal)	Binary (0 or 1)

Table 5.1.2: Relevant Extracted Features from the Dataset

The dataset contains several labels for network anomalies such as jamming, spoofing, MITM attack etc. The primary target variable was ***label_jamming*** and the data was filtered for detected jamming events, i.e. $label\_jamming$ = 1. Therefore, only localization was performed on this dataset, considering we have the jamming detection by the flag.

5.2 Localization Algorithms

Each method estimates the jammer's location using different principles, ranging from machine learning predictions to signal degradation analysis.
The four implemented methods are:

1. Random Forest Regression with KMeans Centroid
2. XGBoost Regression with KMeans Centroid
3. Modified Multilateration with Weights (MMLAW)
4. Jamming Impact Score Weighted Centroid (JIWC)

Machine Learning Methods (1. RF and 2. XGB)

The following two methods both use the same input features, which were chosen considering they are all relevant metrics to network conditions.

Input features:

- Signal Strength (RSSI)
- Signal-to-Noise Ratio (SNR)
- Packet Loss Rate (PLR)
- Sequence Number Gap
- Base Station Load
- Transmission Power
- Uplink/Downlink Quality

The models take as input physical and network features above which reflect signal conditions under jamming and output the predicted jammer's location.

$$x^{(i)} = \left[x_1^{(i)}, x_2^{(i)}, \dots, x_7^{(i)}\right], \quad y^{(i)} = \left[\text{lat}^{(i)}, \text{lon}^{(i)}\right]$$

5.2.1 Random Forest (RF) Regression with Weighted Centroid

This method uses a trained Random Forest (RF) regression model to predict the jammer's location from the perspective of each affected drone.

A Random Forest is an ensemble of M decision trees. Each tree is trained on a random subset of the data and outputs its own prediction:

$$\widehat{y_j} = T_j(x), \quad \text{for } j = 1,2, \dots, M$$

The final RF prediction is the average of all trees, which produces a 2D coordinate (lat, lon) representing where the jammer is likely located from that drone's point of view.

$$\widehat{\boldsymbol{y}} = \frac{1}{M}\sum_{j=1}^{M} T_j(\boldsymbol{x})$$

Lastly, after all predicted locations are collected $\{(\widehat{\text{lat}}_i, \widehat{\text{lon}}_i)\}_{i=1}^{N}$ from N jammed drones, they are aggregated using a weighted centroid, where each prediction is weighted based on a composite jamming indicator score derived from PLR, RSSI, and SNR. (Localization Step). This technique gives higher influence to drones more strongly affected by the jammer.

The RF prediction centroid is then:

$$\hat{y} = \frac{\sum_{i=1}^{N} w_i \cdot y_i}{\sum_{i=1}^{N} w_i}$$

5.2.2 XGBoost Regression with Weighted Centroid

This method follows the same pipeline as the Random Forest approach but uses XGBoost (Extreme Gradient Boosting) as the underlying regression model. XGBoost is a gradient boosting algorithm that builds ensemble of regression trees f_k trained sequentially to minimize error using gradient descent.

$$\hat{\boldsymbol{y}} = \sum_{k=1}^{K} f_k(\boldsymbol{x}), \quad f_k \in \mathcal{F}$$

Where:

- f_k is the k-th regression tree
- $\mathcal{F}$ is the space of all possible regression trees
- $\hat{\boldsymbol{y}}$ is the predicted output (latitude, longitude)

Each tree f_k is trained to minimize the residual error of the previous prediction. The model optimizes the following regularized objective function:

$$\mathcal{L} = \sum_{i} \mathrm{l}\left(y^{(i)}, \widehat{y^{(i)}}\right) + \sum_{k} \Omega(f_k)$$

Where:

- $\mathrm{l}(.)$ is a differentiable loss function, such as mean squared error,
- $\Omega(f_k)$ is a regularization term that penalizes tree complexity.

Lastly, after making per-drone predictions, the estimated jammer location is computed using the same weighted formulation:

$$\hat{y} = \frac{\sum_{i=1}^{N} w_i \cdot y_i}{\sum_{i=1}^{N} w_i}$$

ALGORITHM 1 & 2: RANDOM FOREST AND XGBOOST JAMMER LOCALIZATION

Input: Jammed drone dataset df_jam

Output: Trained RF_Model and XGB_Model ready for prediction

1 Normalize PLR, RSSI, SNR in df_jam → plr_n, rssi_n, snr_n

2 Compute jamming_indicator = 0.4 * plr_n + 0.3 * (1 - rssi_n) + 0.3 * (1 - snr_n)

3 Define features ← [
signal_strength, signal_noise_ratio, packet_loss_rate,
sequence_number_gap, base_station_load,
transmission_power, uplink_downlink_quality
]

4 Define target ← [lat, lon]

5 X ← extract features from df_jam

6 y ← extract target from df_jam

7 Initialize RF_Model with n_estimators = 200 and random seed

8 Train RF_Model using (X, y)

9 Initialize XGB_Model with n_estimators = 200 and random seed

10 Train XGB_Model using (X, y)

11 For each jammed drone i: predict lat/lon using RF_Model and XGB_Model

12 Compute RF weighted centroid:

$$\left(\widehat{\text{lat}_{\text{RF}}}, \widehat{\text{lon}_{\text{RF}}}\right) = \left(\frac{\sum_{i=1}^{N} w_i \cdot \widehat{\text{lat}_{\text{RF}}^{(i)}}}{\sum_{i=1}^{N} w_i}, \frac{\sum_{i=1}^{N} w_i \cdot \widehat{\text{lon}_{\text{RF}}^{(i)}}}{\sum_{i=1}^{N} w_i}\right)$$

13 Compute XGB weighted centroid:

$$\left(\widehat{\text{lat}_{\text{XGB}}}, \widehat{\text{lon}_{\text{XGB}}}\right) = \left(\frac{\sum_{i=1}^{N} w_i \cdot \widehat{\text{lat}_{\text{XGB}}^{(i)}}}{\sum_{i=1}^{N} w_i}, \frac{\sum_{i=1}^{N} w_i \cdot \widehat{\text{lon}_{\text{XGB}}^{(i)}}}{\sum_{i=1}^{N} w_i}\right)$$

14 Return $(\widehat{\text{lat}_{\text{RF}}}, \widehat{\text{lon}_{\text{RF}}})$ and $(\widehat{\text{lat}_{\text{XGB}}}, \widehat{\text{lon}_{\text{XGB}}})$

Jamming Impact Methods (3. MMLAW and 4. JIWC)

To support jammer localization, a composite impact score was defined to quantify how severely each drone was affected by jamming. This score combines three physical-layer metrics that are sensitive to signal disruption: i) Packet Loss Rate (PLR), ii) RSSI (signal strength), and iii) SNR (signal-to-noise ratio). Each metric is min-max normalized to a [0, 1] scale, and then weighted based on its relative importance.

$$\text{Impact Score}_i = w_{\text{plr}} \cdot \text{PLR}_{\text{norm},i} + w_{\text{rssi}} \cdot \left(1 - \text{RSSI}_{\text{norm},i}\right) + w_{\text{snr}} \cdot \left(1 - \text{SNR}_{\text{norm},i}\right)$$
$$\text{where: } w_{\text{plr}} = 0.4, \quad w_{\text{rssi}} = 0.3, \quad w_{\text{snr}} = 0.3$$

PLR was given the highest weight (0.4), as it directly reflects transmission failure, while RSSI and SNR were each weighed at 0.3. Inverted forms of RSSI and SNR are used to ensure that lower values (indicating worse signal quality) correspond to higher impact. The resulting impact score provides a unified measure of signal degradation and is used in both the MMLAW (as a proxy for distance) and JIWC (as a weight in centroid estimation).

5.2.3 Modified Multilateration with Weights (MMLAW)

The MMLAW algorithm applied in my solution is a modified version of the localization algorithm proposed by M. Savva in his dissertation [25]. The original MMLAW algorithm was designed for wireless sensor networks using Contiki OS and Cooja, making use of network-layer metrics such as Retransmissions or ETX (Expected Transmission Count) as a basis for making distance estimates in multilateration.

In contrast, my implementation targets LoRa networks, where such network-layer metrics used here aren't present or even defined. Consequently, I replaced them with physical-layer jamming indicators (impact scores) better applicable to LoRa: RSSI, SNR, and Packet Loss Rate (PLR). These were min-max normalized and inverted to simulate distances, maintaining the underlying assumption that higher effect indicates closeness to the jammer.

$$\text{Impact Score}_i = 0.4 \cdot \text{PLR}_{\text{norm},i} + 0.3 \cdot \left(1 - \text{RSSI}_{\text{norm},i}\right) + 0.3 \cdot \left(1 - \text{SNR}_{\text{norm},i}\right)$$

In summary, while the initial algorithm made use of ETX-weighted distance, my implementation maintains the same formulation using least-squares multilateration but adjusts the input metric in accordance with LoRa's communication characteristics.

Step 1: Simulated Distance Calculation

Each drone is assigned a simulated distance d_i based on its jamming indicator value. The indicator is min-max normalized and inverted, so that higher impact corresponds to a shorter distance.

$$d_i = 1 - \frac{z_i - z_{min}}{z_{max} - z_{min}}, \quad \text{for } i = 1,2,\dots,N$$

Where:

- z_i is the jamming indicator for drone i
- d_i is the simulated distance
- N is the number of affected drones

Step 2: Multilateration Setup

Let (x_i, y_i) be the GPS coordinates of drone i, and assume the jammer is located at (x, y). We define the system of equations based on differences of squared distances between the first drone (reference) and the others:

$$A_i = 2(x_i - x_1) \quad 2(y_i - y_1), \text{ for } i = 2,\dots,N$$
$$B_i = d_1^2 - d_i^2 - x_1^2 + x_i^2 - y_1^2 + y_i^2$$

This yields a linear system in matrix form:

$$A \cdot [xy] = B$$

Where:

- $A \in R^{(N-1)\times 2}$,
- $B \in R^{N-1}$

Step 3: Least-Squares Estimation

We solve for the jammer coordinates (x, y) using the least-squares solution:

$$[\hat{x}\hat{y}] = \arg\min_{x,y} |A \cdot [xy] - B\,|^2$$

Which is computed using the pseudo-inverse or via:

$$[\hat{x}\hat{y}] = (A^\top A)^{-1} A^\top B$$

The final estimated jammer location is:

$$(\text{lat}_{\text{est}}, \text{lon}_{\text{est}}) = (\hat{x}, \hat{y})$$

ALGORITHM 3: MMLAW JAMMER LOCALIZATION (MODIFIED MULTILATERATION WITH WEIGHTS)

Input: Jammed drone dataset sample_df, jamming indicator column indicator_col
Output: Estimated jammer coordinates $(\hat{x}, \hat{y})$

Extract coordinates ← sample_df[['lat', 'lon']]
Extract indicator ← sample_df[indicator_col]
Normalize and invert indicator to simulate distances:
d ← 1 − MinMaxScaler().fit_transform(indicator)
if number of coordinates < 3, **then**
    Return None, None
x ← latitude values, y ← longitude values
Initialize matrix A of size (N − 1) × 2
Initialize vector B of size (N − 1)
for i = 1 to N − 1 **do**
    A[i−1, 0] ← 2 × (x[i] − x[0])
    A[i−1, 1] ← 2 × (y[i] − y[0])
    B[i−1] ← $d[0]^2 - d[i]^2 - x[0]^2 + x[i]^2 - y[0]^2 + y[i]^2$
end for
Estimate jammer coordinates using least-squares: $(\hat{x}, \hat{y})$ ← lstsq(A, B)
Return $(\hat{x}, \hat{y})$

5.2.4 Jamming Impact Score Weighted Centroid (JIWC) – Proposed method

The Jamming Indicator Weighted Centroid (JIWC) method is proposed and developed as part of this thesis and was designed specifically to suit the characteristics of LoRa networks. JIWC is a signal-impact-based localization approach and estimates the jammer's location by computing a weighted average (centroid) of drone coordinates, where each drone is weighted based on how severely it was affected by the jamming signal.

The key idea of this approach is that drones with higher jamming impact scores are most likely closer to the jammer. Therefore, the jammer position is estimated closer to drone

with higher jamming indicators. To summarize, JIWC provides a simple yet effective geometric estimation without distance computation and iterative solving.

Step 1: Jamming Impact Score Calculation

As mentioned earlier, a composite jamming impact score is computed as a function of three physical-layer measures for each drone: Packet Loss Rate (PLR), RSSI (Received Signal Strength), and SNR (Signal-to-Noise Ratio). These are initially min-max normalized:

$$\mathrm{PLR}_{\mathrm{norm}},\ \mathrm{RSSI}_{\mathrm{norm}},\ \mathrm{SNR}_{\mathrm{norm}} \in [0,1]$$

The impact score s_i for each drone i is then calculated as:

$$s_i = w_{\mathrm{plr}} \cdot \mathrm{PLR}_{\mathrm{norm}} + w_{\mathrm{rssi}} \cdot (1 - \mathrm{RSSI}_{\mathrm{norm}}) + w_{\mathrm{snr}} \cdot (1 - \mathrm{SNR}_{\mathrm{norm}})$$

Where:

$$w_{\mathrm{plr}} = 0.4, \quad w_{\mathrm{rssi}} = 0.3, \quad w_{\mathrm{snr}} = 0.3$$

This reflects that higher PLR, lower RSSI, and lower SNR contribute more strongly to jamming impact.

Step 2: Impact Weight Normalization

The impact scores s_i are min-max normalized to produce weights $w_i \in [0,1]$ for each drone:

$$w_i = \frac{s_i - s_{min}}{s_{max} - s_{min}}$$

Where:

- w_i is the normalized weight used in the centroid,
- s_{min} and s_{max} are the minimum and maximum impact scores across all jammed drones.

Step 3: Weighted Centroid Calculation

Let (x_i, y_i) be the GPS coordinates of drone i. The estimated jammer position is computed as a weighted centroid:

$$x_{\text{est}} = \frac{\sum_{i=1}^{N} x_i \cdot w_i}{\sum_{i=1}^{N} w_i}, y_{\text{est}} = \frac{\sum_{i=1}^{N} y_i \cdot w_i}{\sum_{i=1}^{N} w_i}$$

Where:

- N is the number of jammed drones,
- $(x_{\text{est}}, y_{\text{est}})$ is the estimated jammer location.

The final estimated jammer location in geographic coordinates is:

$$(\text{lat}_{\text{est}}, \text{lon}_{\text{est}}) = (x_{\text{est}}, y_{\text{est}})$$

ALGORITHM 4: JAMMING IMPACT WEIGHTED CENTROID (JIWC)

Input: Jammed drone dataset df_jam

Output: Estimated jammer coordinates $(\text{lat}_{\text{est}}, \text{lon}_{\text{est}})$

Define features ← [packet_loss_rate, signal_strength, signal_noise_ratio]

Min-max normalize each feature → plr_norm, rssi_norm, snr_norm

Compute impact score for each packet:
impact_score = 0.4 × plr_norm + 0.3 × (1 - rssi_norm) + 0.3 × (1 - snr_norm)

Group by drone ID and compute average of: latitude, longitude, impact score

Min-max normalize the average impact scores → impact_weight

Let N be the number of jammed drones

Initialize total weight $W \leftarrow 0$

Initialize sums: $sum\_lat \leftarrow 0,\ sum\_lon \leftarrow 0$

for $i = 1$ to N **do**

    Retrieve drone latitude x_i, longitude y_i, and weight w_i

    $\text{sum\_lat} \leftarrow \text{sum\_lat} + x_i \cdot w_i$

    $\text{sum\_lon} \leftarrow \text{sum\_lon} + y_i \cdot w_i$

    $W \leftarrow W + w_i$

end for

Return $\text{lat}_{\text{est}} \leftarrow \frac{\text{sum\_lat}}{W}, \text{lon}_{\text{est}} \leftarrow \frac{\text{sum\_lon}}{W}$

5.3 Ensemble Localization Framework

Methodology

Since the dataset lacks ground truth jammer coordinates, direct evaluation of localization accuracy was infeasible. To address this, we adopted an aggregation-based approach to assess consensus among four distinct localization methods: RF (Random

Forest), XGB (XGBoost), MMLAW, and JIWC. Since these methods each rely on different localization principles, combining them forms an **ensemble** localization framework that leverages the strengths of each technique to produce more robust jammer position estimates. The goal was to determine whether the methods converge toward a consistent estimate as the number of observed jammed devices increases.

ENSEMBLE LOCALIZATION FRAMEWORK

Input: Jammed drone dataset sample_df, jamming indicator column indicator_col
Define scenario_groups ← [(10,10), (5,20), (3,33), (2,50)]
Initialize intermediate_centers ← []
for each (num_scenarios, num_devices) in scenario_groups **do**
    Generate random_scenarios ← draw num_scenarios subsets of size num_devices from unique drone IDs
    Initialize scenario_estimates ← []
    for each scenario in random_scenarios **do**
        Extract sample ← entries in df_jam from scenario's drone IDs
        Compute:
            JI_Centroid ← weighted average of coordinates using jamming indicator
            MMLAW ← least-squares multilateration based on inverted indicator
            RF ← KMeans centroid of RF predictions
            XGB ← KMeans centroid of XGB predictions
        Collect estimates ← [JI_Centroid, MMLAW, RF, XGB]
        Compute scenario_center ← mean of estimates
        Compute scenario_radius ← max distance of any estimate to center
        Append estimates to scenario_estimates
    end for
    Compute group_center ← mean of all scenario_estimates
    Compute group_radius ← max distance from group_center to any estimate
    Append (group_center, group_radius) to intermediate_centers
end for
Compute final_center ← mean of all group_centers
Compute final_radius ← max distance from final_center to any group_center
Return final_center, final_radius

Chapter 6

Results and Evaluation for Jamming Detection and Localization

6.1 Jamming Detection Results

To visualize the network's data, we first plotted RSSI over Time (Figure 6.1.1) for each device. It can be determined from the plot that RSSI data suddenly changes during the middle of 08/02/2023, which is also the known jamming period time.

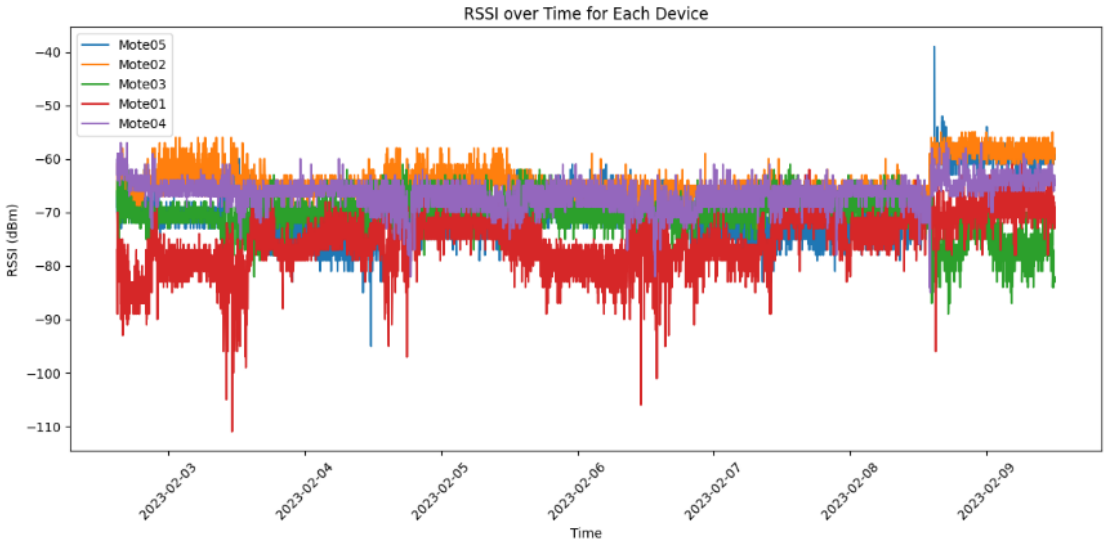


Figure 6.1.1: RSSI over Time for Mote01 – Mote05.

After calculating the new metric of Packet Loss Ratio (PLR) as described in Chapter 5, we also plot the PLR over time for the whole network in one plot (Mote01 – Mote05).

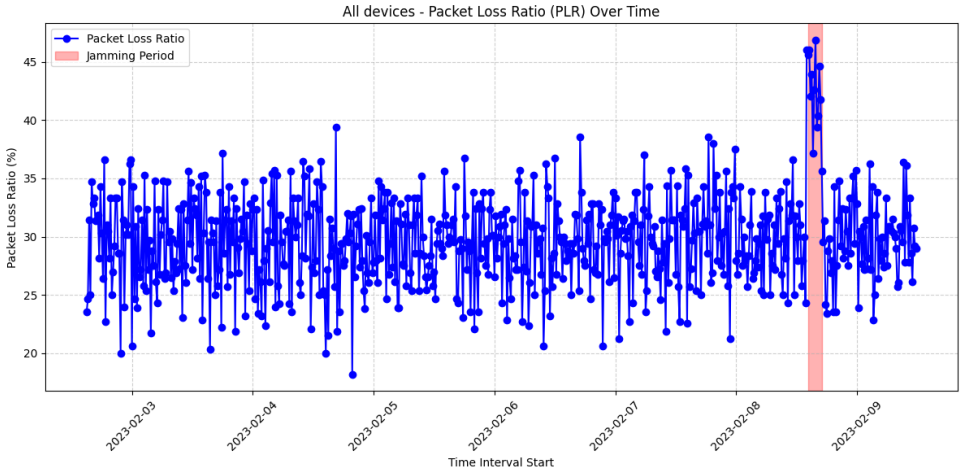


Figure 6.1.2: Network's PLR over time

A clear rise in PLR values during the defined jamming period (February 8th, 14:15–17:00) can be observed, confirming the correlation between jamming and packet disruption.

ML Methods Results (Plots)

Machine learning-based detection methods were visualized with anomaly scatter plots and evaluated against ground truth labels and confusion matrices. Each model successfully detected a significant portion of the jamming period (which is shadowed with grey in scatter plots). All anomalies are marked considering the 20-minute intervals that were used for the PLR estimation and visualized as red dots (Figures 6.1.4 – 6.1.10).

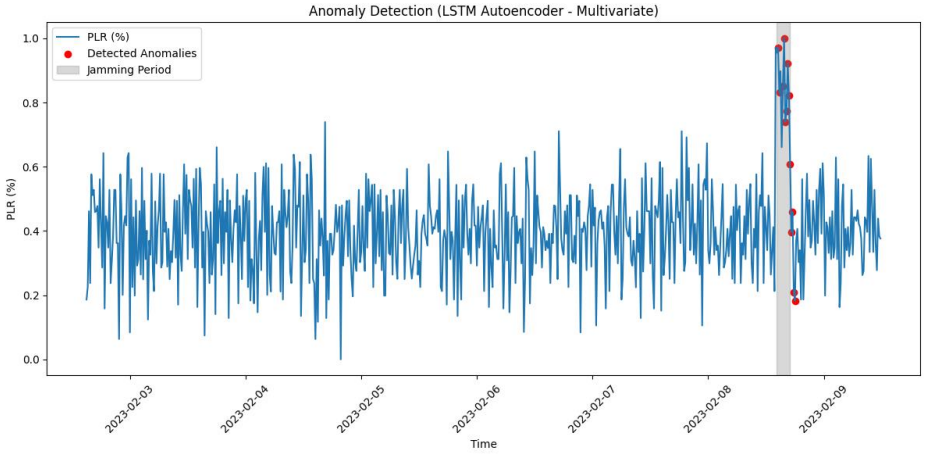


Figure 6.1.3: LSTM jamming detection across dataset

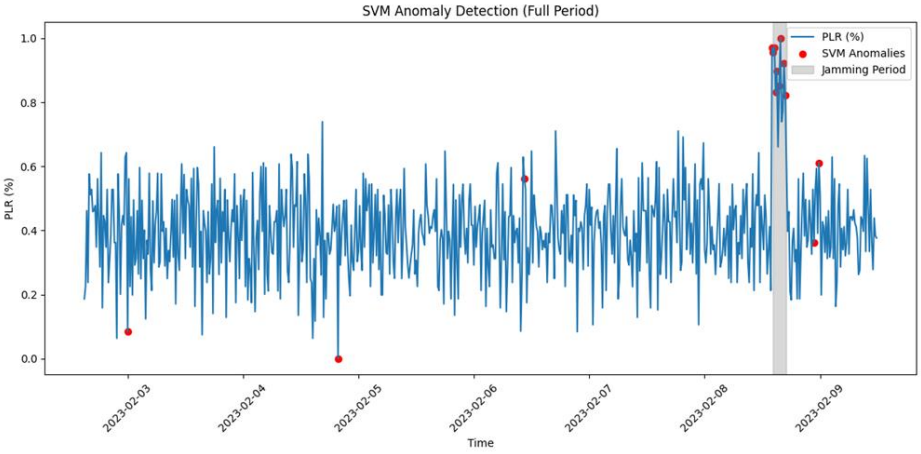


Figure 6.1.4: SVM jamming detection across dataset

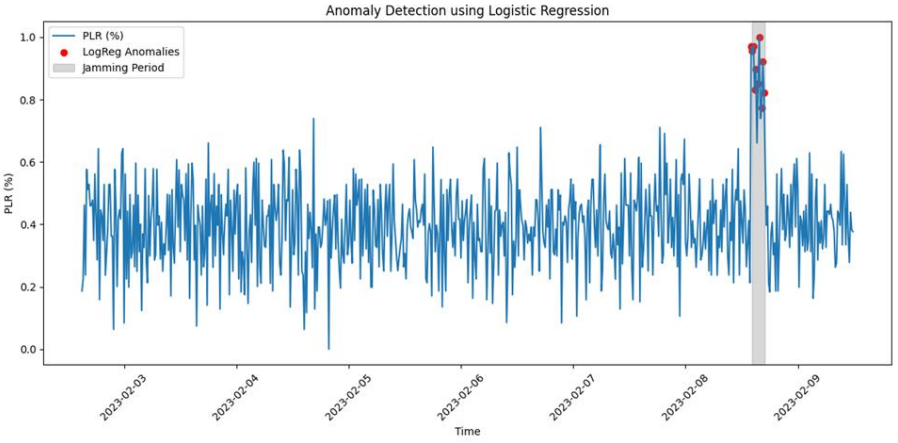


Figure 6.1.5: BLR jamming detection across dataset

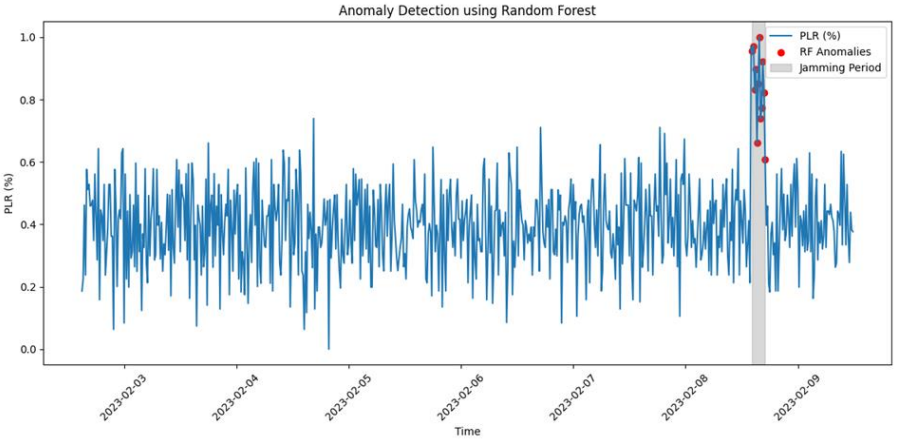


Figure 6.1.6: Random Forest jamming detection across dataset

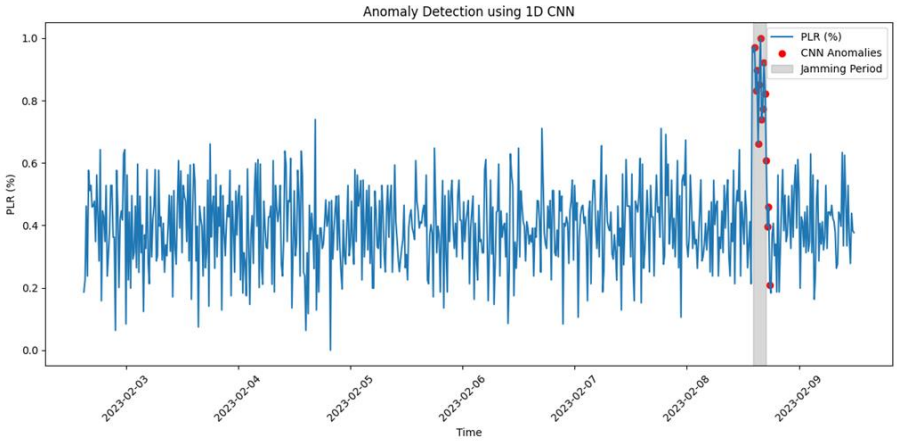


Figure 6.1.7: CNN 1-D jamming detection across dataset

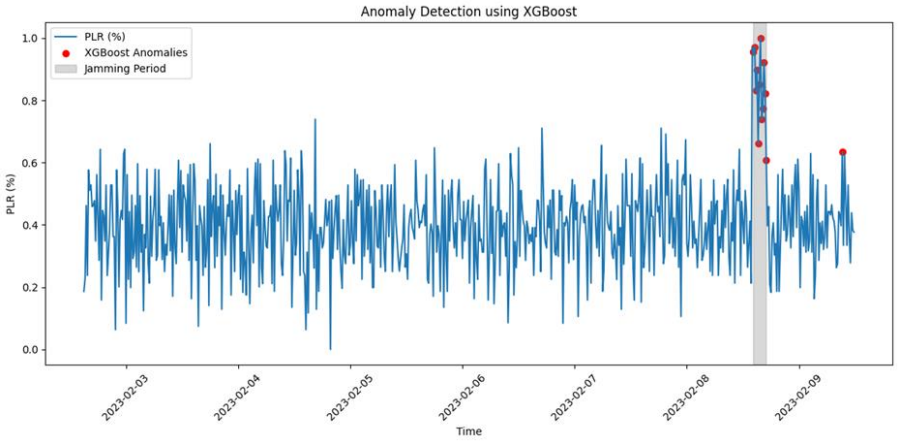


Figure 6.1.8: XGBoost jamming detection across dataset

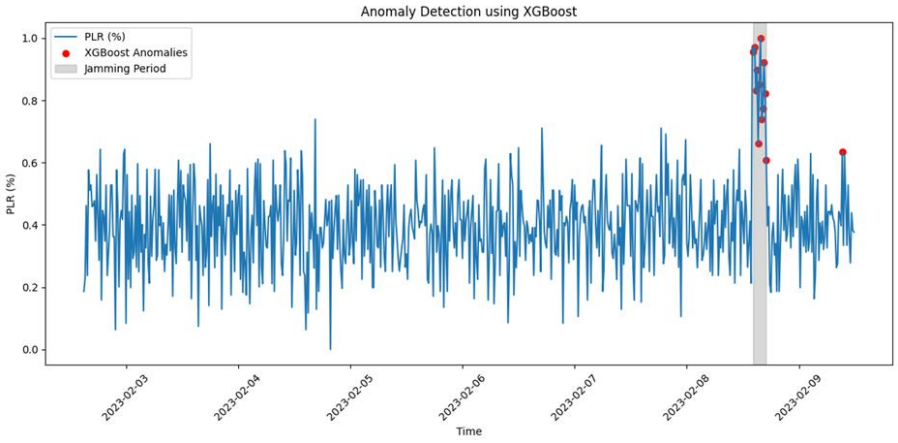


Figure 6.1.9: K-Means jamming detection across dataset

6.2 Jamming Detection Evaluation

Confusion matrices showed each model's predictive performance and can be easily compared to determine the most effective ML technique. In these matrices, the true positives (TP) represent correctly identified jamming intervals, while true negatives (TN) correspond to correctly identified normal periods. In contrast, false positives (FP) showcase incorrect jamming detection instances (i.e., false alarms) and false negatives (FN) correspond to incorrectly identified normal events (i.e., real attacks that are overlooked).

Term	Meaning
True Positive (TP)	Jamming occurred and was correctly predicted by the model
True Negative (TN)	No jamming occurred and was correctly predicted as normal by the model
False Positive (FP)	No jamming occurred, but the model incorrectly predicted it as jamming (false alarm).
False Negative (FN)	Jamming occurred, but the model failed to detect it and predicted normal (missed attack).

Table 6.2.1: Classification Terms Explanation in Jamming Detection

A high number of TP and TN indicates accurate detection, which is important for minimizing both missed jamming events (FN) and false alarms (FP).
Essentially, for effective jamming detection, both FN and FP should be low FN, with FN being more important (thus, must be minimized) to ensure real attacks are not overlooked.

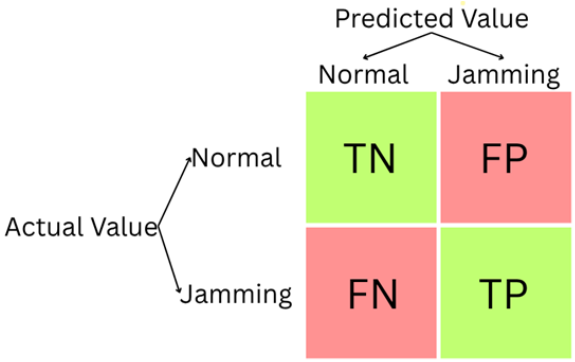
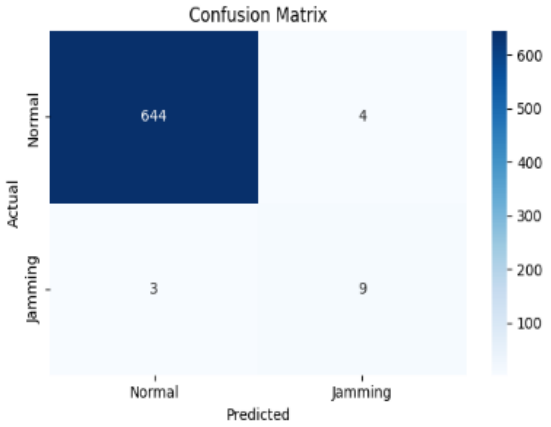
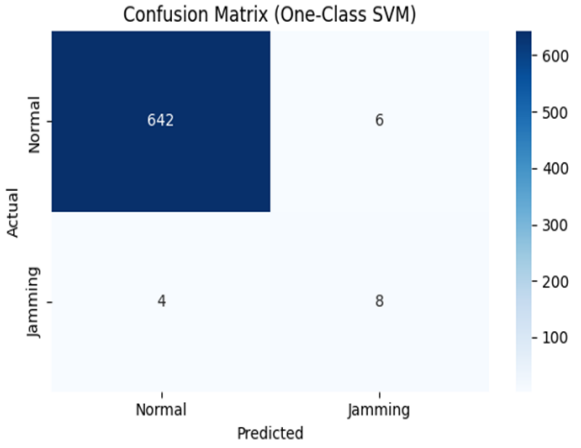


Figure 6.2.1: Confusion Matrix Representation for Jamming Detection

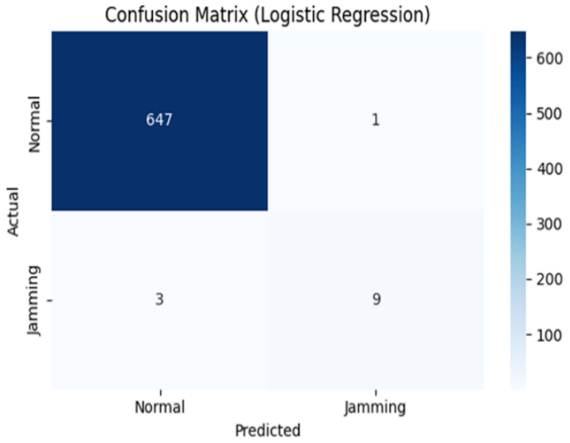
1. LSTM Autoencoder



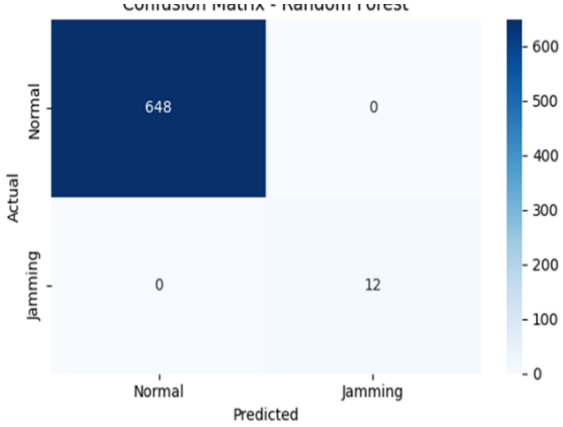
2. SVM



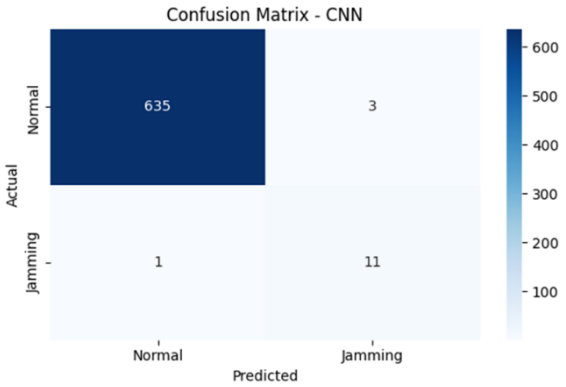
3. BLR



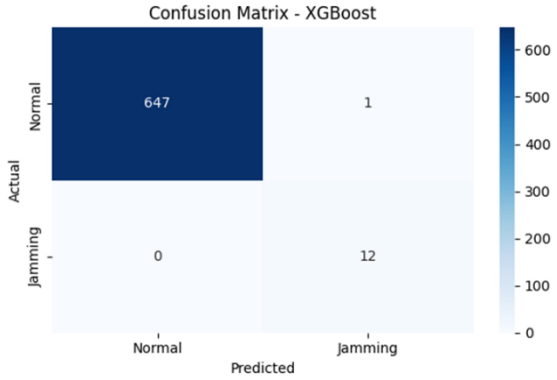
4. Random Forest



5. CNN 1-D



6. XGBOOST



7. K-MEANS

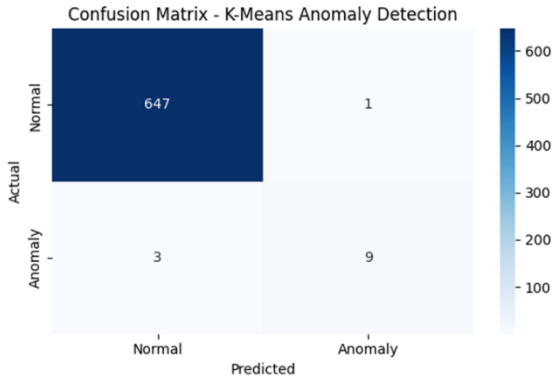


Figure 6.2.2: Confusion Matrices across all methods

Method	TP	TN	FP	FN
LSTM	9	644	4	3
SVM	8	642	6	4
BLR	9	647	1	3
RF	12	648	0	0
1D-CNN	11	635	3	1
XGBoost	12	647	1	0
K-Means	9	647	1	3

Table 6.2.2: Classification Results for all Machine Learning techniques.

Among all methods, Random Forest achieved perfect results (TP = 12, FP = 0) and XGBoost near-perfect results (TP = 12, FP = 1) both correctly identifying all jamming

instances and minimizing false predictions. These models achieved the highest overall reliability, with no false negatives, ensuring no jamming events were missed.
Conversely, One-Class SVM and LSTM exhibited higher false positive and false negative rates, indicating lower sensitivity and more frequent misclassifications. For instance, One-Class SVM missed 4 jamming periods (FN) and incorrectly flagged 6 normal periods as jamming (FP).

Evaluation Metrics Overview

Accuracy measures the overall correctness of a model by evaluating how many predictions were correct over the total number of predictions. It can be misleading in imbalanced datasets, as it may remain high even if the model fails to detect rare classes (in our case, jamming events).

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

Precision quantifies how many of the instances predicted as positive (jamming) were actually correct. It is especially important to be considered when false alarms (false positives) need to be minimized.

$$Precision = \frac{TP}{TP + FP}$$

Recall, also known as sensitivity or true positive rate, measures how well the model detects actual positive cases (jamming intervals). A high recall means fewer jamming events are missed.

$$Recall = \frac{TP}{TP + FN}$$

Specificity measures how well the model identifies negative cases correctly. It complements recall by focusing on the true negative rate.

$$Specificity = \frac{TN}{TN + FP}$$

Lastly, the **F1-score** which is defined as the harmonic mean of precision and recall, provides a balanced metric especially useful for imbalanced datasets. It is more informative than accuracy when class distributions are uneven.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

The F1-score is a crucial evaluation metric in this study because the dataset is imbalanced, with jamming events occurring during only a small portion of the total monitoring period. This imbalance was further emphasized when the data was split into short time intervals for PLR calculation.

In such cases, relying on metrics like accuracy alone can be misleading, therefore the F1-score attempts to address this issue by combining both precision and recall. It provides a balanced view of the model's performance, especially on the minority class (jamming).

Method	**Accuracy**	**Precision**	**Recall (Sensitivity)**	**Specificity**	**F1-Score**
RF	100.00%	100.00%	100.00%	100.00%	100.00%
XGBoost	99.85%	92.31%	100.00%	99.85%	96.00%
1D-CNN	99.38%	78.57%	91.67%	99.53%	84.62%
BLR	99.39%	90.00%	75.00%	99.85%	81.82%
K-Means	99.39%	90.00%	75.00%	99.85%	81.82%
LSTM	98.94%	69.23%	75.00%	99.38%	72.00%
SVM	98.48%	57.14%	66.67%	99.07%	61.54%

Table 6.2.3: Evaluation metrics ranked by F1-score for all techniques.

The table above (Table 6.2.3) shows all the evaluation metrics for each method, ranked by the F1-score, as it's the most reliable performance metric for imbalanced datasets.

Based on the evaluation metrics, Random Forest (RF) achieved perfect performance across all indicators. This suggests it is the most reliable model for jamming detection in this study, with no false positives (FP) or false negatives (FN). XGBoost also performed exceptionally well, achieving high recall (100%) and precision (92.31%), indicating that it successfully identified all jamming instances with minimal false alarms. 1D-CNN showed strong generalization with high recall (91.67%) and a balanced F1-score (84.62%), making it a solid deep learning alternative.

On the other hand, SVM and LSTM underperformed compared to the other methods, particularly in precision and F1-score, suggesting a tendency to misclassify normal data as jamming.

Overall, tree-based models like Random Forest and XGBoost provided the best trade-off between detection accuracy and false alarm reduction, making them the most suitable for this specific deployment in LoRaWAN jamming detection.

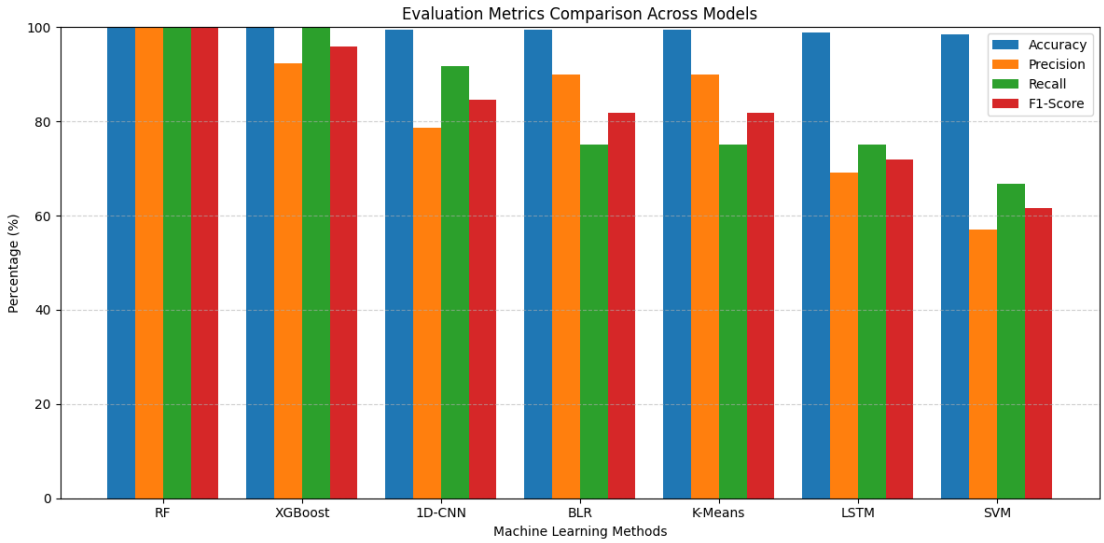


Figure 6.2.3: Bar chart evaluation metrics comparison across models

It must be highlighted that accuracy which remains high across all models, might not be the most reliable performance metric in this case. This is due to the imbalanced nature of the dataset, where normal (non-jamming) intervals significantly outnumber jamming ones.

Lastly, because this dataset lacks geographic coordinates for each mote and the jammer, no direct localization can be applied. However, the jamming impact (JI) score can still be calculated for each mote. The Jamming Impact (JI) score is the key metric used for the second part of the thesis' implementation, i.e. the localization part on another dataset.

The JI score for each mote is calculated using a weighted combination of the change in Packet Loss Ratio (PLR), RSSI deviation, and SNR drop between the pre-jamming and jamming periods:

$$JI = w_{plr} \times \left(\mathrm{PLR}_{jamming} - \mathrm{PLR}_{normal}\right) + w_{rssi} \times \left(\mathrm{RSSI}_{normal} - \mathrm{RSSI}_{jamming}\right) + w_{snr} \times \left(\mathrm{SNR}_{normal} - \mathrm{SNR}_{jamming}\right)$$

Where:

$$w_{plr} = 0.4, \qquad w_{rssi} = 0.3, \qquad w_{snr} = 0.3$$

Mote	**Impact Score**
Mote3	8.60
Mote4	5.39
Mote1	4.55
Mote2	3.45
Mote5	0.62

Table 6.2.4: Jamming Impact Scores per Mote

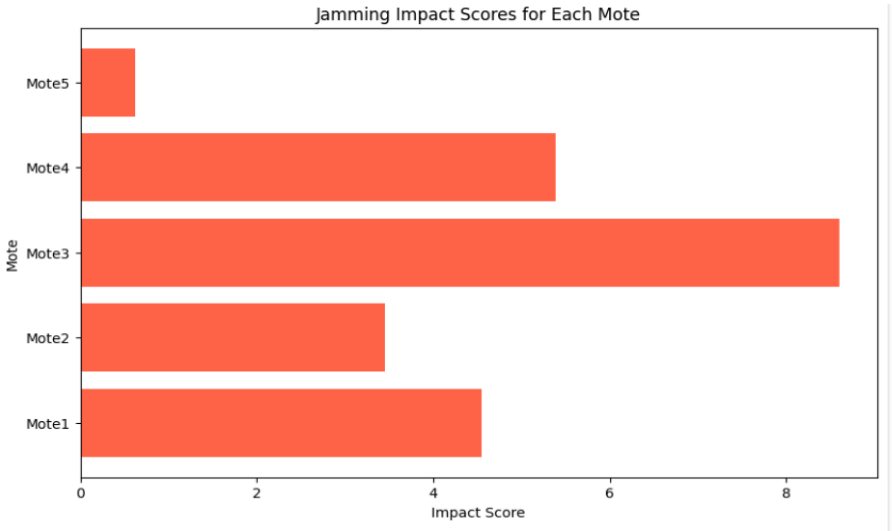


Figure 6.2.4: Bar Chart Comparison of Jamming Impact Scores per Mote

In conclusion, Mote3 experienced the highest impact (JI = 8.60), followed by Mote4 and Mote1, indicating proximity to or stronger exposure from the jammer. This scoring approach serves as a proxy for impact-based localization, since the absence of coordinate data prevents the application of more precise localization techniques (e.g. trilateration).

6.3 Jamming Localization Results

Experimental Design

Our approach applies all four implemented localization algorithms (RF, XGB, MMLAW, JIWC) across controlled subsets of jammed devices. The dataset was first filtered to include only communication entries corresponding to the LoRa protocol, with a jamming label of 1. From the filtered dataset, 100 unique jammed drone identifiers were available for experimentation. These were partitioned into progressively larger groups to analyze scalability (Table 6.3.1). For each scenario, all four methods were applied to the same subset of devices, generating independent jammer coordinate estimates.

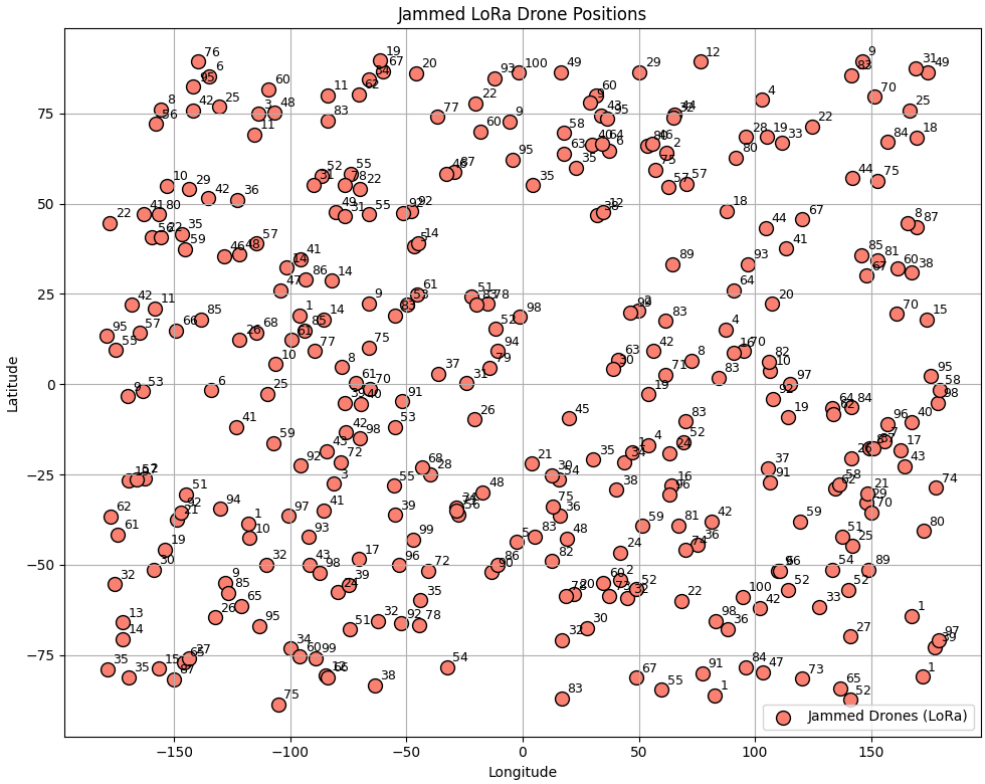


Figure 6.3.1: Geographic locations of jammed drone devices participating in the LoRa network.

Scenario Definition

The scenario design process involved dividing the dataset into five progressively larger groups to observe how localization performance scales with increased data availability. The following scenario configurations were used:

Configuration	Number of scenarios	Number of devices (in each scenario)
1	10	10
2	5	20
3	3	33
4	2	50
5	1	100

Table 6.3.1: Groups of random data splits for the 100 unique drones

For each configuration, the drone IDs were selected randomly using a random seed. Within each scenario, all four localization methods were applied independently on the same subset of drones to generate four distinct (latitude, longitude) estimates representing the predicted jammer position. In the following section, we provide the results with all the plots required. Before showing all the plots, we will present the two types of plots along with some explanations.

Scenario-Level Aggregation (Plot Type 1)

For every scenario (e.g., a group of 10 devices), all four localization methods (RF, XGB, MMLAW, JIWC) were applied independently, yielding four predicted jammer coordinates. This set of predictions is then processed using ensemble logic by averaging the coordinates, allowing us to reduce method-specific bias and isolate the most probable jammer location per scenario. The results are visualized in Plot Type 1 (Fig. 6.3.2), which shows:

1. **Jammed Drones**: Plotted with color intensity based on their jamming impact score (darker red indicates stronger impact), a metric used by the methods.
2. **Method Estimates**: Each of the four methods' predictions shown as distinct markers.

3. **Scenario Center**: Computed as the mean of all method estimates (gray dot).
4. **Estimation Radius**: The maximum Euclidean distance from any method's estimate to the center (dashed gray circle), quantifying the worst-case dispersion of predictions.

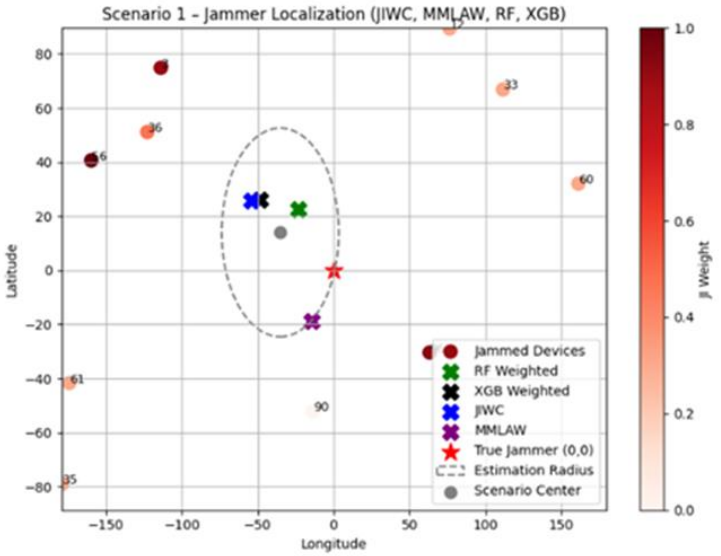


Figure 6.3.2: Example of Plot Type 1

Configuration-Level Aggregation (Plot Type 2)

Plot Type 2 (Fig. 6.3.3) synthesizes intermediate results across all scenarios within a configuration (e.g., five 20-device groups) to reveal system-level trends. More specifically, it superimposes all scenario estimation circles and centers (gray dots) from Type 1 plots and displays the final configuration-level jammer position (black dot) as the average of all scenario centers.

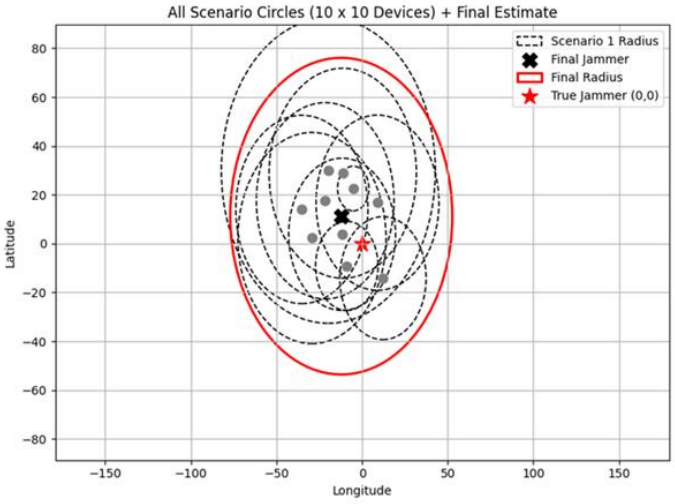
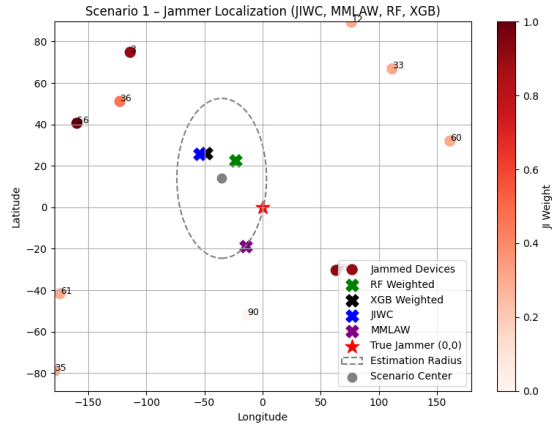
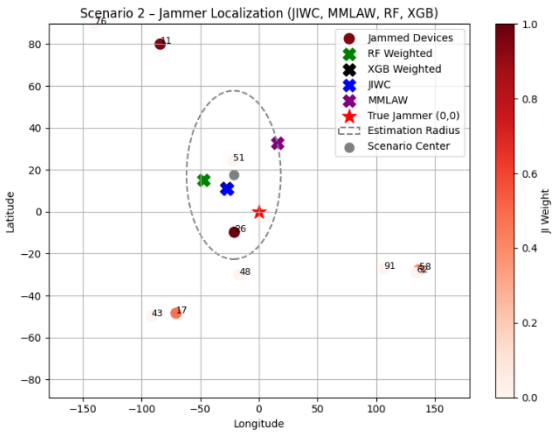
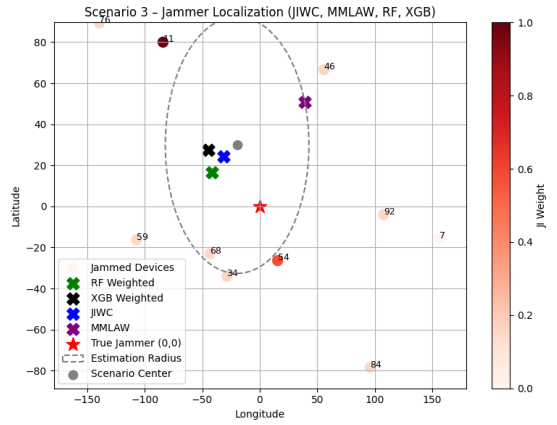


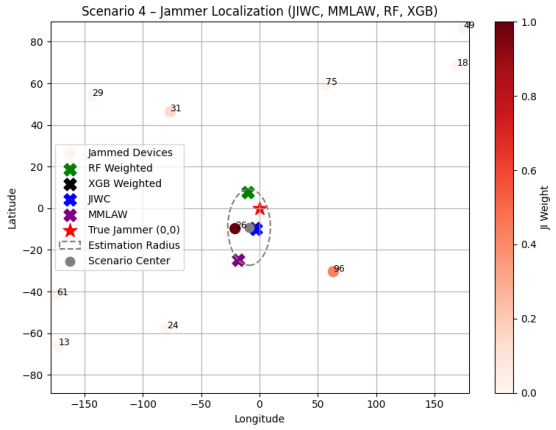
Figure 6.3.3: : Example of Plot Type 2

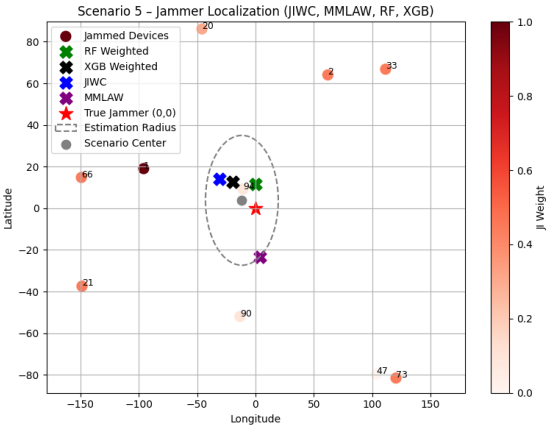
Configuration 1: 10 scenarios x 10 devices

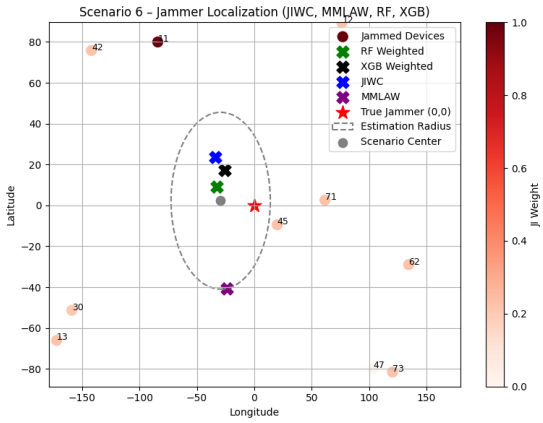












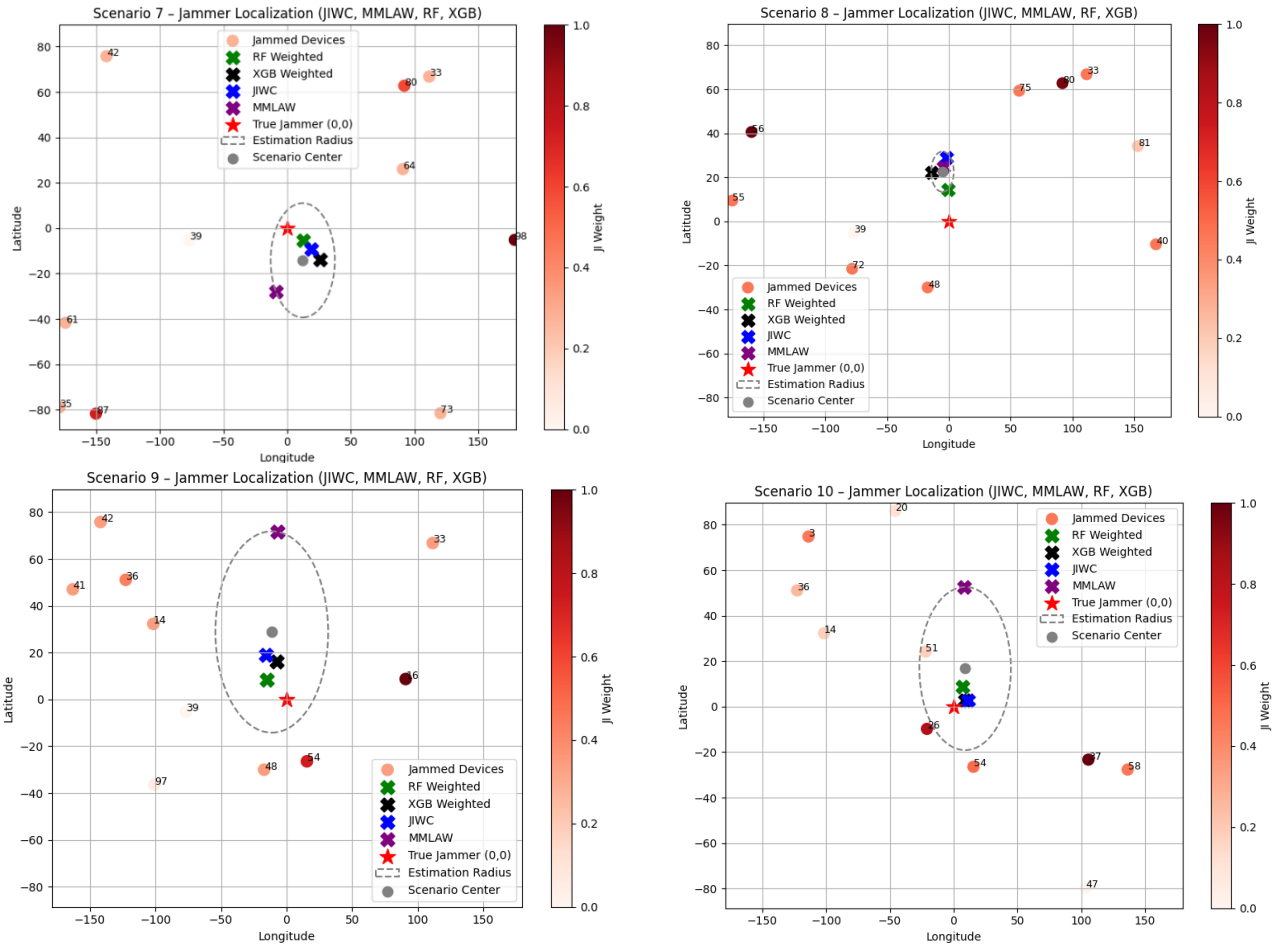


Figure: 6.3.4: 10 Scenario Plots for 10x10 Configuration (Type 1)

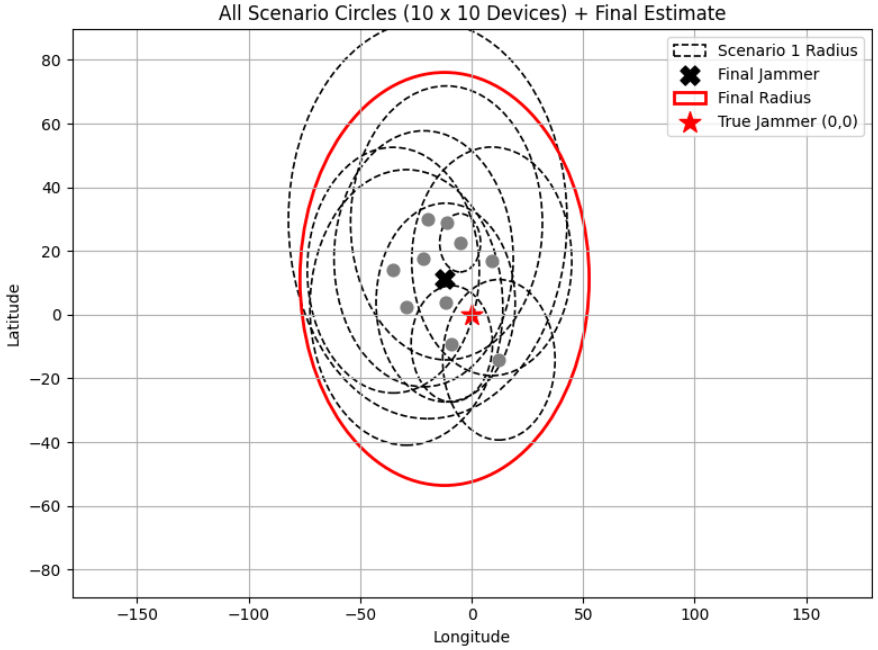


Figure: 6.3.5: Intermediate Result Plot for 10x10 Configuration (Type 2)

Configuration 2: 5 scenarios x 20 devices

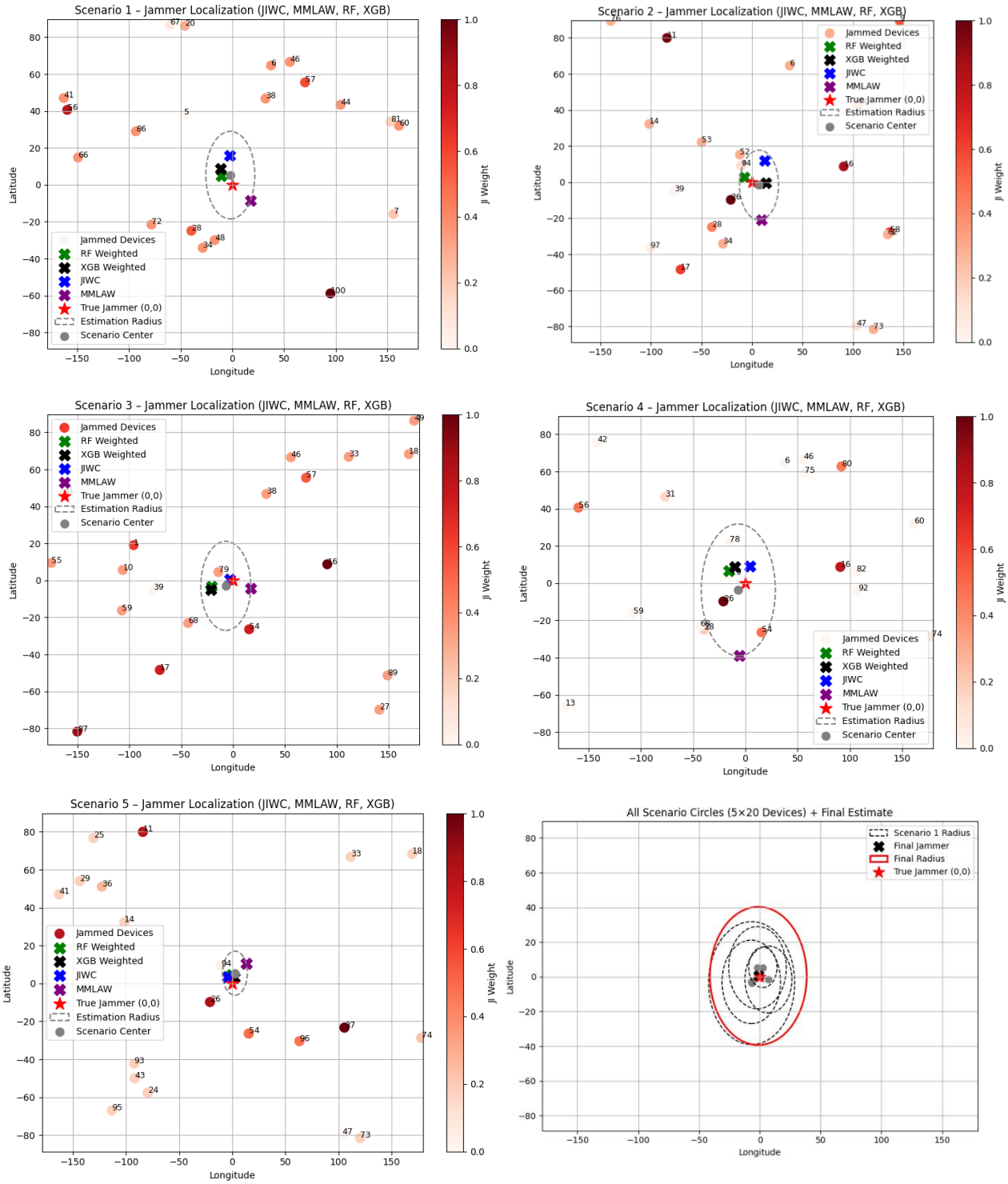


Figure: 6.3.6: 5 Scenario Plots for 5x20 Configuration (Type 1) and Intermediate Result Plot (Type 2)

Configuration 3: 3 scenarios x 33 devices

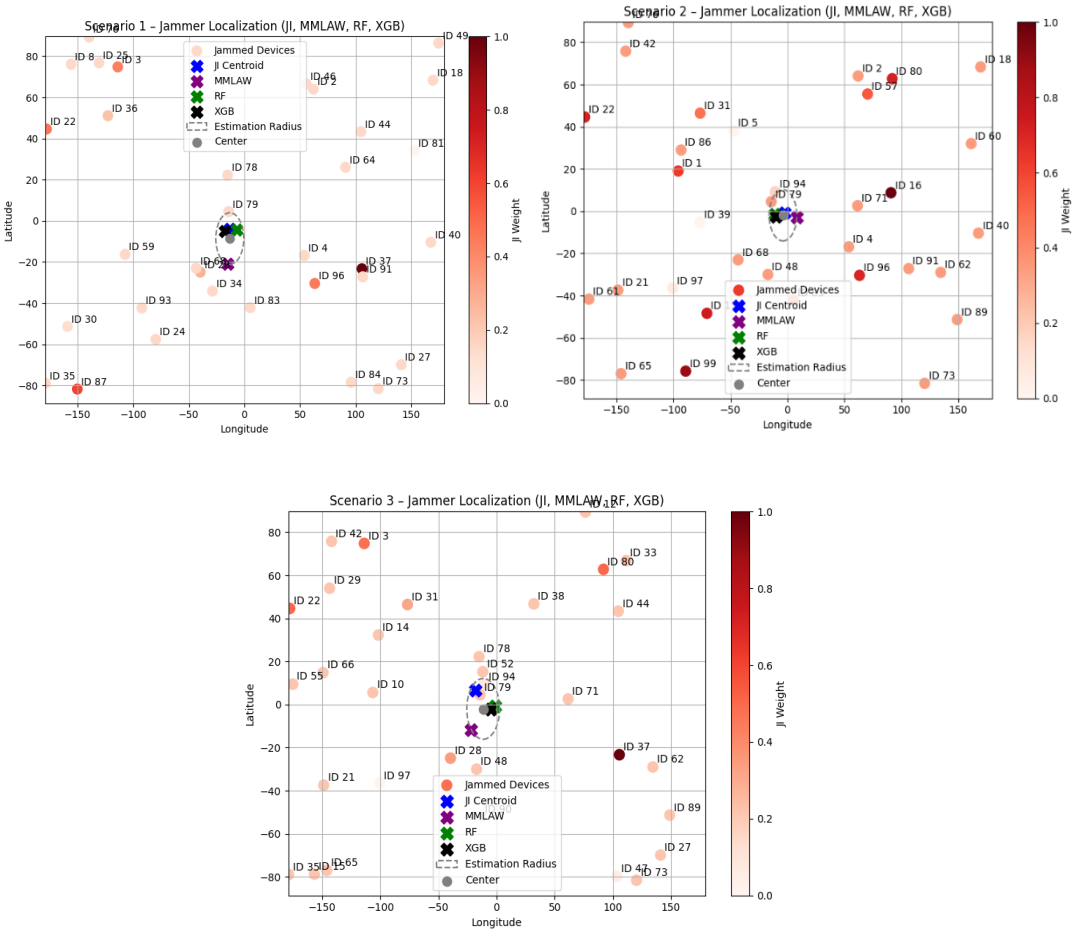


Figure: 6.3.7: 3 Scenario Plots for 3x33 Configuration (Type 1)

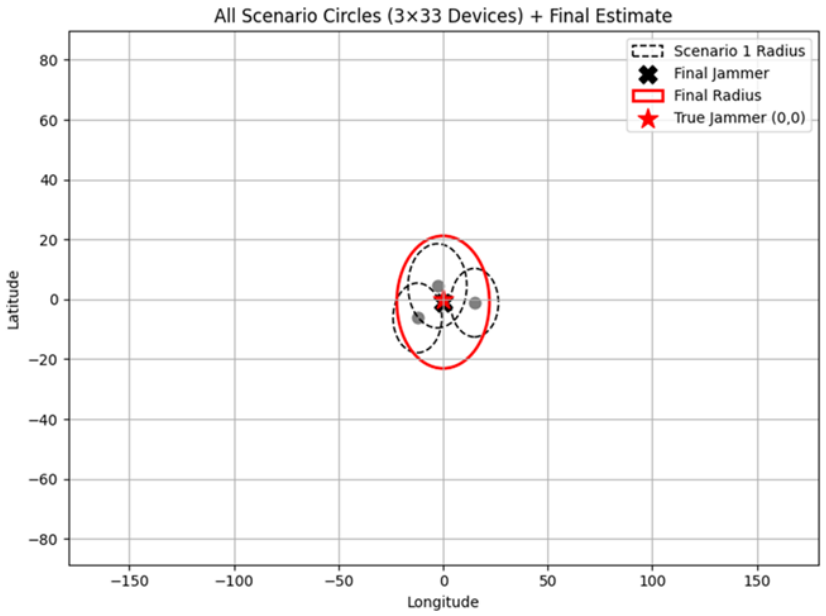


Figure: 6.3.8: Intermediate result plot for 3x33 configuration (Type 2)

Configuration 4: 2 scenarios x 50 devices

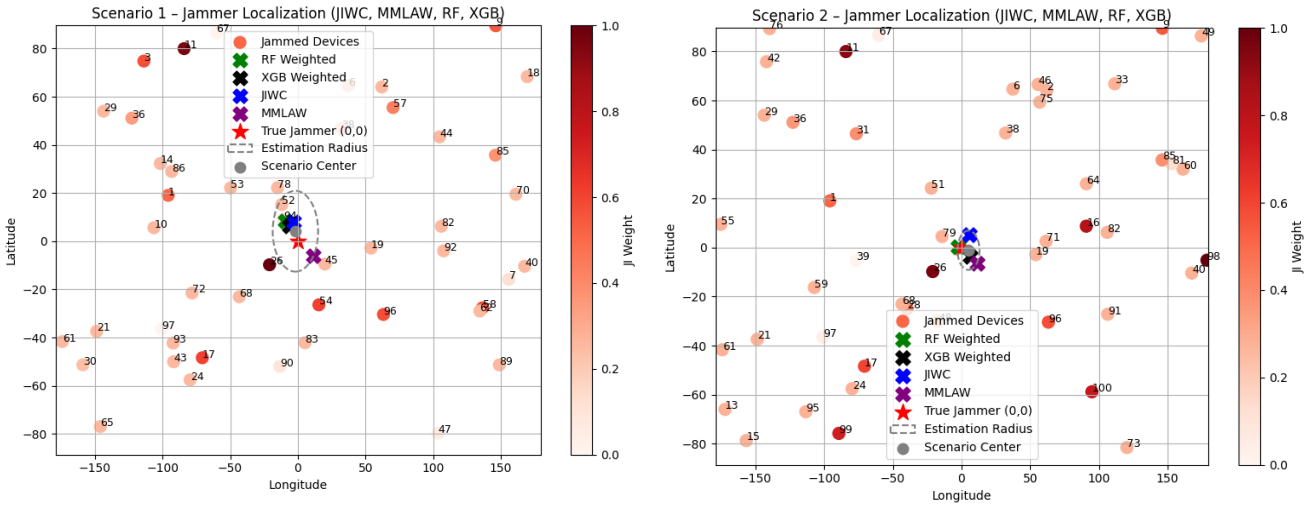


Figure: 6.3.9: 3 Scenario Plots for 3x33 Configuration (Type 1)

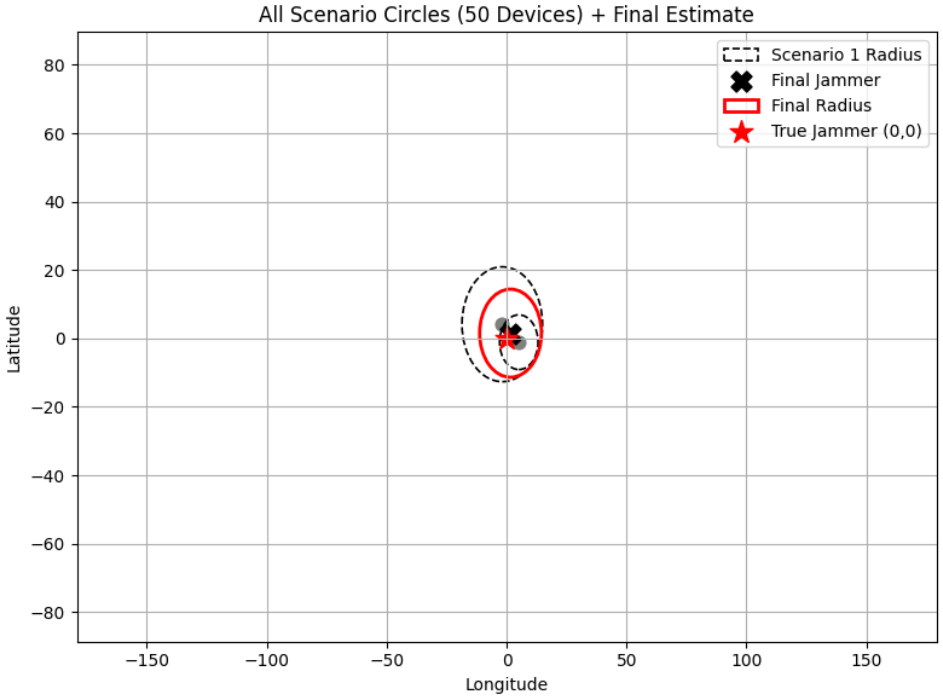


Figure: 6.3.10: Intermediate result plot for 2x50 configuration (Type 2)

Configuration 5: 1 scenario x 100 devices

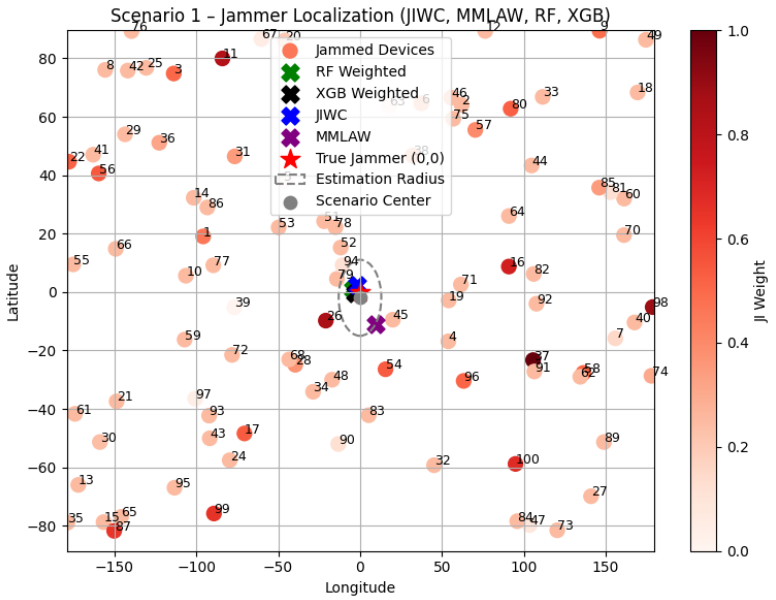


Figure: 6.3.11: 1 Scenario Plot for 1x100 Configuration (Type 1)

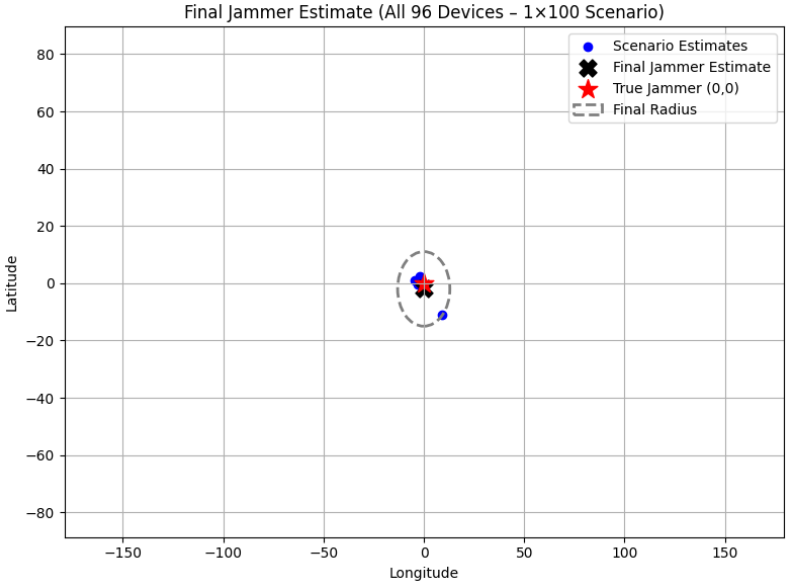


Figure: 6.3.12: Intermediate result plot for 1x100 configuration (Type 2)

Devices	Latitude	Longitude	Radius (degrees)
10	11.243996	-12.138278	64.82
20	0.524562	-1.187424	39.81
33	-0.929483	-0.002457	22.13
50	1.540155	1.641561	12.88
100	-1.958810	-0.064555	13.02

Table 6.3.2: Intermediate results after execution of the four configurations

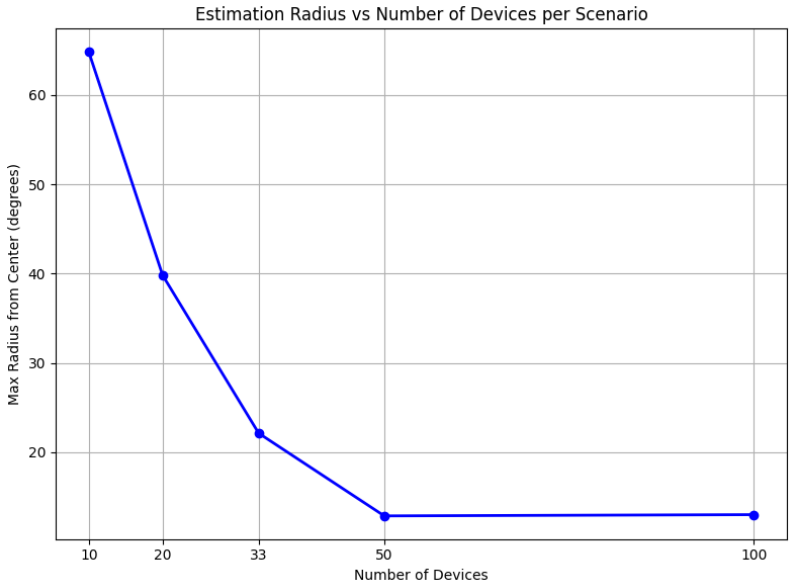


Figure 6.3.13: Impact of Device Count on Localization Accuracy (Estimation Radius)

The figure 6.3.13 demonstrates that as the number of jammed devices per scenario increases, the estimation radius consistently decreases. The estimation radius decreases from 64.82° (10 devices) to 13.02° (100 devices) - an 80% reduction in dispersion. This indicates improved localization accuracy with larger device groups, suggesting that a higher number of input data points leads to more consistent and convergent jammer predictions. In other words, more devices lead to greater agreement among localization methods and more reliable jammer detection.

For the final jammer localization, all previous intermediate results are aggregated together. The final plot (Figure 6.3.12) presents the overall results from all four groups (10x10, 5x20, 3x33, 2x50 and 1x100 groups), showing their estimated jammer positions. Based on the four results, final location is approximated by the mean center. Table 6.3.3 shows all the coordinates used for the plot and table 6.3.4 shows the final jammer estimation coordinates (grey dots).

Devices in Each Scenario	Latitude	Longitude
10	11.2440	-12.1383
20	0.5246	-1.1874
33	-0.9295	-0.0025
50	1.5402	1.6416
100	-1.9588	-0.0646

Table 6.3.3: Jammer Localization Estimates by Scenario

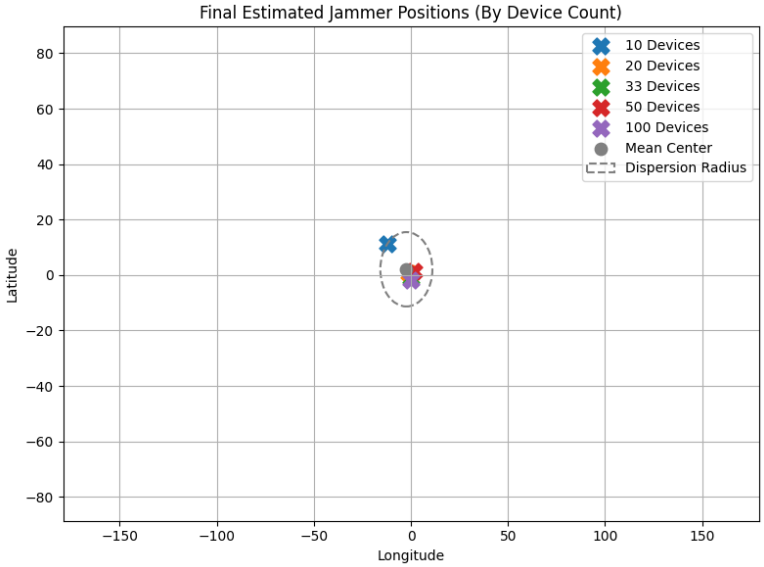


Figure 6.3.14: Final Jammer Location Estimation (including 10x10 Configuration)

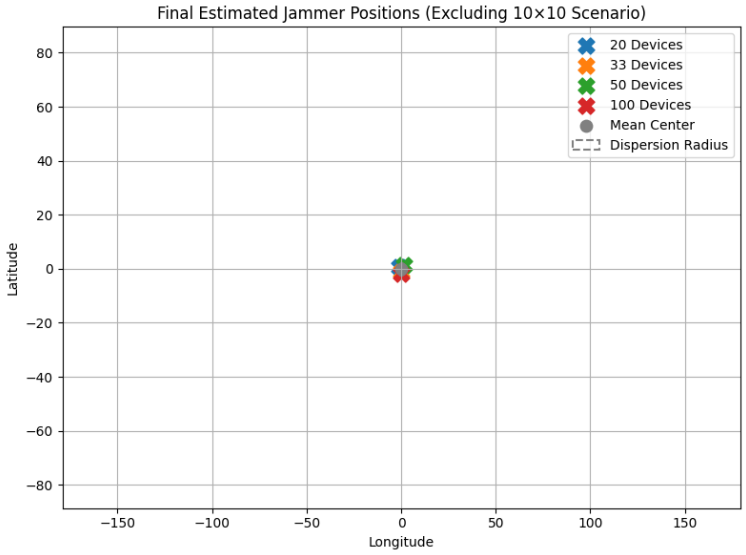


Figure 6.3.15: Final Jammer Location Estimation (excluding 10x10 configuration)

Hence, since Figure 6.3.13 evidently shows a radius of dispersion decreasing with increasing device groups per scenario, it can be concluded that larger device groups lead to more consistent and reliable jammer localization. Furthermore, the final plots in Figures 6.3.14 and 6.3.14 show that the estimated jammer locations for each of the four different group configurations are closely clustered around a common center. This convergence indicates that despite the absence of ground truth coordinates, the localization methods and aggregation strategy point to a consistent region, which is an internal consistency of the proposed approach.

While the final mean jammer position was initially calculated including all configurations, we observed that the 10x10 configuration exhibited the largest dispersion radius and highest inconsistency. For completeness, we also computed the mean center excluding the 10x10 configuration.

Final Mean Center Estimation	**Latitude**	**Longitude**
Mean center (incl. 10x10 scenario)	2.084084	-2.350231
Mean center (excl. 10x10 scenario)	-0.205894	0.096781

Table 6.3.4: Final Mean Center Estimates

Notably, all configurations converge toward (0,0), with 33-device scenarios averaging (-0.93, -0.002) and 100-device scenarios reaching (-1.96, -0.065). This consistent progression strongly suggests the jammer was intentionally placed at the origin (0,0) during data collection. Additionally, as shown in Table 6.3.4, the exclusion leads to a mean estimate much closer to the origin (0,0). This further strengthens our conclusion regarding the jammer's true position.

We will use this hypothesis as a reference point to evaluate localization accuracy and method performance in Section 6.4, where we analyze and compare the estimated jammer positions against the assumed true location.

6.4 Jamming Localization Evaluation

Since the dataset used in this study does not give ground truth coordinates for the actual position of the jammer, direct error-based evaluation, i.e., estimation of localization accuracy or distance to true point, is not feasible. In the absence of labeled coordinates, we rely instead on internal consistency indicators to assess the effectiveness of the proposed localization techniques. These indicators include agreement among different methods, reduction in dispersion (i.e., estimation radius), and convergence toward a stable center as the number of jammed devices increases.

To overcome the lack of ground truth, we introduced a hypothesis based on observed convergence patterns: the jammer was likely positioned at the origin (0,0) during data collection. This assumption is supported by consistent localization trends across all configurations that were described in Section 6.3. Using this hypothesis, we can evaluate each method's performance by comparing estimated coordinates against the (0,0) reference and measuring their Euclidean distance. This strategy enables relative error calculations and facilitates ranking of methods across different device group sizes.

The following figures and tables present the evaluation results based on this hypothesis.

Configuration	**Latitude**	**Longitude**	**Lat Distance (°)**	**Lon Distance (°)**	**Euclidean Distance (°)**
10 Devices	11.2440	-12.1383	11.2440	12.1383	16.4643
20 Devices	0.5246	-1.1874	0.5246	1.1874	1.3005
33 Devices	-0.9295	-0.0025	0.9295	0.0025	0.9295
50 Devices	1.5402	1.6416	1.5402	1.6416	2.2522
100 Devices	-1.9588	-0.0646	1.9588	0.0646	1.9599
Mean (incl. 10×10)	2.0841	-2.3502	2.0841	2.3502	3.1493
Mean (excl. 10×10)	-0.2059	0.0968	0.2059	0.0968	0.2280

Table 6.4.1: Localization Estimates and Distances from Jammer Position (0,0)

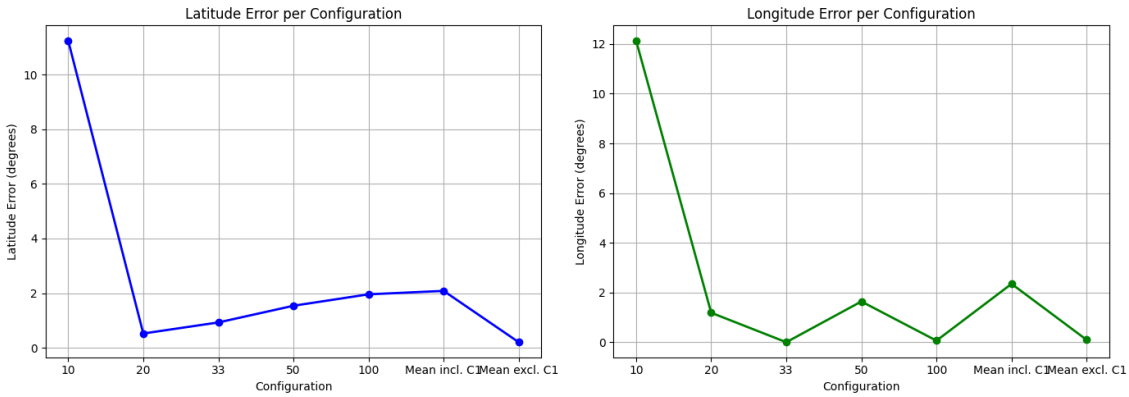


Figure 6.4.1: Latitude and Longitude Error of Jammer Location Estimates Across Configurations and Aggregated Mean Centers (Including and Excluding the 10x10 Scenario)

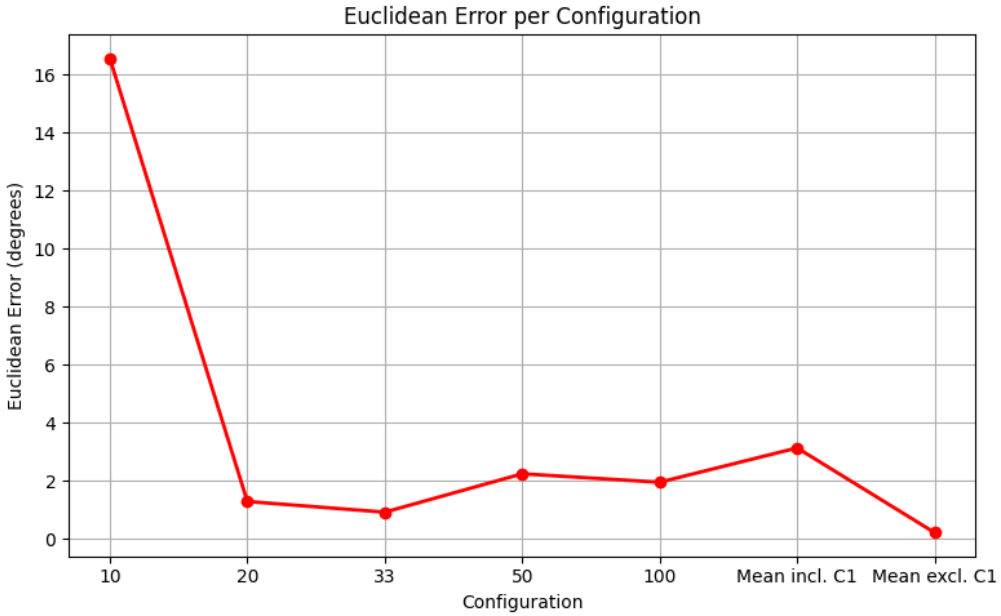


Figure 6.4.2: Euclidean Error of Jammer Location Estimates Across Configurations and Aggregated Mean Centers (Including and Excluding the 10x10 Scenario)

The quantitative analysis demonstrates significant convergence of localization estimates toward the hypothesized jammer position at (0,0). The 10-device scenario shows the highest deviation (16.46° error) and was previously shown to have the largest estimation radius, therefore its outlier behavior justifies treating it separately in the final analysis. Nevertheless, all larger device groups achieve sub-2° accuracy, with the 33-device configuration yielding the most precise estimate (0.93° error). This progression further confirms that increased data availability improves localization reliability.

The final mean center estimates, computed by aggregating results across configurations, further confirm the impact of noisy scenarios. When all configurations are included, the mean center deviates by approximately 3.15° from the origin. In contrast, excluding the 10-device configuration yields a final mean center with a remarkably low error of just 0.23°.

Method Evaluation

To assess the performance of the four proposed jammer localization methods (JIWC, MMLAW, RF, and XGBoost) we calculated their average Euclidean error across the five different scenario configurations (10x10, 5x20, 3x33, 2x50, and 1x100). For each configuration, the predicted jammer location by each method was compared against our hypothesized ground truth at the origin (0,0), and the Euclidean distance in degrees serving as our accuracy metric.

Table 6.4.2 presents the average Euclidean errors for each method across the five configurations and Table 6.4.3 calculates the aggregated average error for each method. As expected, all methods generally improve with larger device groups. Notably, RF achieved the lowest average error (11.2°), followed closely by JIWC (12.5°) and XGBoost (13.2°) also performing reliably. MMLAW demonstrated the highest average error (22.44°), suggesting that geometric localization may be more sensitive to distribution or noise in the input dataset.

Configuration	**RF**	**XGB**	**MMLAW**	**JIWC**
10x10	24.188	28.529	40.539	30.187
5x20	12.535	13.442	23.191	10.494
3x33	7.249	12.855	21.183	10.465
2x50	7.508	8.220	12.889	8.007
1x100	4.525	2.993	14.421	3.329

Table 6.4.2: Average Euclidean Errors (°) per Method and Configuration

Rank	Method	Avg. Error (°)
1	RF	11.201
2	JIWC	12.496
3	XGB	13.208
4	MMLAW	22.444

Table 6.4.3: Average Method Performance Ranking Across All Configurations

Chapter 7

Conclusion

7.1 Summary

This project explored the intersection of indoor localization and machine learning with a keen interest in jamming detection and localization in LoRaWAN IoT networks. A thorough literature review outlined the evolution of wireless technologies, fingerprinting methods, and security concerns across WSN, IoT, and LPWAN systems. Several supervised, unsupervised, and ensemble machine learning models were evaluated for anomaly detection, showing varied performance based on the nature of the data and model complexity. The framework's implementation provided a multi-step process, from jamming detection via PLR/SNR/RSSI anomalies to estimating attacker location using an ensemble approach. Evaluation results confirmed the effectiveness of machine learning and deep learning models for accurate detection, while simpler scoring methods provided a practical approach for estimating jamming impact without device coordinates.

7.2 Challenges

Some of the challenges we encountered during the development of the suggested jammer detection-localization framework:

- Dataset Inconsistency and Missing Location Data

It was not possible to directly apply localization algorithms on the initial dataset of jamming events, since it lacked location (coordinates) information of both gateways and

end devices. That didn't affect the anomaly detection part, which was originally implemented with various ML and DL models on the initial dataset.

- Selection of machine learning algorithms and optimization of hyperparameters

The dataset properties and localization requirements should be considered when selecting the best machine learning methods and hyperparameters. Finding the most suitable method requires choosing a model based on performance indicators and cross-validation.

7.3 Future Work

Future work could expand on this study by applying the proposed methods to datasets that include ground-truth coordinates, allowing for direct localization performance comparisons. It should also be considered extending the classification of jamming types (e.g., reactive, random) and introducing mobile attackers to enhance the robustness and realism of future indoor localization and security systems.

References

[1] "Number of Internet of Things (IoT) connections worldwide from 2022 to 2023," Statista. Accessed: May 19, 2025. [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

[2] V. Sneha and M. Nagarajan, "Localization in Wireless Sensor Networks: A Review," *Cybern. Inf. Technol.*, vol. 20, no. 4, pp. 3–26, Nov. 2020, doi: 10.2478/cait-2020-0044.

[3] T. Ahmad, X. J. Li, M. Ashfaq, M. Savva, I. Ioannou, and V. Vassiliou, "Location-enabled IoT (LE-IoT): Indoor Localization for IoT Environments using Machine Learning," in *2024 20th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, Abu Dhabi, United Arab Emirates: IEEE, Apr. 2024, pp. 392–399. doi: 10.1109/DCOSS-IoT61029.2024.00065.

[4] Mawahib Sharafeldin Adam Boush, "Enhancing IoT Network Attack Detection with Ensemble Machine Learning and Efficient Feature Extraction," *J. Inf. Syst. Eng. Manag.*, vol. 10, no. 20s, pp. 288–298, Mar. 2025, doi: 10.52783/jisem.v10i20s.3054.

[5] S. M. Maghdid and H. Maghdid, "A Comprehensive Review of Indoor/Outdoor Localization Solutions in IoT era: Research Challenges and Future Perspectives," Aug. 13, 2021. doi: 10.36227/techrxiv.15138609.

[6] M. Savva, I. Ioannou, and V. Vassiliou, "Evaluating Localization Algorithms in IoT Networks Under Jamming Attacks," in *2024 IFIP Networking Conference (IFIP Networking)*, Thessaloniki, Greece: IEEE, Jun. 2024, pp. 627–633. doi: 10.23919/IFIPNetworking62109.2024.10619065.

[7] V. Di Pietra, P. Dabove, and M. Piras, "Loosely Coupled GNSS and UWB with INS Integration for Indoor/Outdoor Pedestrian Navigation," *Sensors*, vol. 20, no. 21, p. 6292, Nov. 2020, doi: 10.3390/s20216292.

[8] H. Obeidat, W. Shuaieb, O. Obeidat, and R. Abd-Alhameed, "A Review of Indoor Localization Techniques and Wireless Technologies," *Wirel. Pers. Commun.*, vol. 119, no. 1, pp. 289–327, Jul. 2021, doi: 10.1007/s11277-021-08209-5.

[9] M. Rahman, M. NagshvarianJahromi, S. S. Mirjavadi, and A. M. Hamouda, "Compact UWB Band-Notched Antenna with Integrated Bluetooth for Personal

Wireless Communication and UWB Applications," *Electronics*, vol. 8, no. 2, p. 158, Feb. 2019, doi: 10.3390/electronics8020158.

[10] S. F. Ahmed *et al.*, "Toward a Secure 5G-Enabled Internet of Things: A Survey on Requirements, Privacy, Security, Challenges, and Opportunities," *IEEE Access*, vol. 12, pp. 13125–13145, 2024, doi: 10.1109/ACCESS.2024.3352508.

[11] Z. Liu, L. Chen, X. Zhou, Z. Jiao, G. Guo, and R. Chen, "Machine Learning for Time-of-Arrival Estimation With 5G Signals in Indoor Positioning," *IEEE Internet Things J.*, vol. 10, no. 11, pp. 9782–9795, Jun. 2023, doi: 10.1109/JIOT.2023.3234123.

[12] T. Perković, L. Dujić Rodić, J. Šabić, and P. Šolić, "Machine Learning Approach towards LoRaWAN Indoor Localization," *Electronics*, vol. 12, no. 2, p. 457, Jan. 2023, doi: 10.3390/electronics12020457.

[13] J. Šabić, T. Perković, D. Begušić, and P. Šolić, "Practical Realization of Reactive Jamming Attack on Long-Range Wide-Area Network," *Sensors*, vol. 25, no. 8, p. 2383, Apr. 2025, doi: 10.3390/s25082383.

[14] A. Augustin, J. Yi, T. Clausen, and W. Townsley, "A Study of LoRa: Long Range & Low Power Networks for the Internet of Things," *Sensors*, vol. 16, no. 9, p. 1466, Sep. 2016, doi: 10.3390/s16091466.

[15] N. Hou, X. Xia, and Y. Zheng, "Jamming of LoRa PHY and Countermeasure," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, Vancouver, BC, Canada: IEEE, May 2021, pp. 1–10. doi: 10.1109/INFOCOM42981.2021.9488774.

[16] "LoRa Alliance," LoRa Alliance®. Accessed: Apr. 14, 2025. [Online]. Available: https://lora-alliance.org/

[17] "Network Options." Accessed: Apr. 14, 2025. [Online]. Available: https://resources.lora-alliance.org/private-vs-public-networks

[18] "RP002-1.0.4 Regional Parameters," LoRa Alliance®. Accessed: Apr. 14, 2025. [Online]. Available: https://resources.lora-alliance.org/technical-specifications/rp002-1-0-4-regional-parameters

[19] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the Limits of LoRaWAN," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 34–40, 2017, doi: 10.1109/MCOM.2017.1600613.

[20] Y. Yu *et al.*, “Adaptive Multi-Channels Allocation in LoRa Networks,” *IEEE Access*, vol. 8, pp. 214177–214189, 2020, doi: 10.1109/ACCESS.2020.3040765.

[21] E. Aras, N. Small, G. S. Ramachandran, S. Delbruel, W. Joosen, and D. Hughes, “Selective Jamming of LoRaWAN using Commodity Hardware,” in *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, Nov. 2017, pp. 363–372. doi: 10.1145/3144457.3144478.

[22] H. Alqurashi, F. Bouabdallah, and E. Khairullah, “SCAP SigFox: A Scalable Communication Protocol for Low-Power Wide-Area IoT Networks,” *Sensors*, vol. 23, no. 7, p. 3732, Apr. 2023, doi: 10.3390/s23073732.

[23] Y. Li *et al.*, “Location-Enabled IoT (LE-IoT): A Survey of Positioning Techniques, Error Sources, and Mitigation,” *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4035–4062, Mar. 2021, doi: 10.1109/JIOT.2020.3019199.

[24] R. S. Sinha, Y. Wei, and S.-H. Hwang, “A survey on LPWA technology: LoRa and NB-IoT,” *ICT Express*, vol. 3, no. 1, pp. 14–21, Mar. 2017, doi: 10.1016/j.icte.2017.03.004.

[25] M. Savva, “A Framework for the Detection, Localization, and Recovery from Jamming Attacks in the Internet of Things,” University of Cyprus, Nicosia, Cyprus, 2024. doi: 10.13140/RG.2.2.35507.95525.

[26] G. Pettorru, V. Pilloni, and M. Martalò, “Trustworthy Localization in IoT Networks: A Survey of Localization Techniques, Threats, and Mitigation,” *Sensors*, vol. 24, no. 7, p. 2214, Mar. 2024, doi: 10.3390/s24072214.

[27] R. Shahbazian, G. Macrina, E. Scalzo, and F. Guerriero, “Machine Learning Assists IoT Localization: A Review of Current Challenges and Future Trends,” *Sensors*, vol. 23, no. 7, p. 3551, Mar. 2023, doi: 10.3390/s23073551.

[28] G. Oguntala, R. Abd-Alhameed, S. Jones, J. Noras, M. Patwary, and J. Rodriguez, “Indoor location identification technologies for real-time IoT-based applications: An inclusive survey,” *Comput. Sci. Rev.*, vol. 30, pp. 55–79, Nov. 2018, doi: 10.1016/j.cosrev.2018.09.001.

[29] O. Cheikhrouhou, G. M. Bhatti, and R. Alroobaea, “A Hybrid DV-Hop Algorithm Using RSSI for Localization in Large-Scale Wireless Sensor Networks,” *Sensors*, vol. 18, no. 5, p. 1469, May 2018, doi: 10.3390/s18051469.

[30] N. Podevijn *et al.*, “LoRaWAN Geo-Tracking Using Map Matching and Compass Sensor Fusion,” *Sensors*, vol. 20, no. 20, p. 5815, Oct. 2020, doi: 10.3390/s20205815.

[31] T. Perković, L. Dujić Rodić, J. Šabić, and P. Šolić, “Machine Learning Approach towards LoRaWAN Indoor Localization,” *Electronics*, vol. 12, no. 2, p. 457, Jan. 2023, doi: 10.3390/electronics12020457.

[32] T. Yang, A. Cabani, and H. Chafouk, “A Survey of Recent Indoor Localization Scenarios and Methodologies,” *Sensors*, vol. 21, no. 23, p. 8086, Dec. 2021, doi: 10.3390/s21238086.

[33] J. Jiao, X. Wang, and C. Han, “Robust Indoor Localization in Dynamic Environments: A Multi-source Unsupervised Domain Adaptation Framework,” Feb. 11, 2025, *arXiv*: arXiv:2502.07246. doi: 10.48550/arXiv.2502.07246.

[34] T. Alhmiedat, “Fingerprint-Based Localization Approach for WSN Using Machine Learning Models,” *Appl. Sci.*, vol. 13, no. 5, p. 3037, Feb. 2023, doi: 10.3390/app13053037.

[35] X. Yu, H. Wang, and J. Wu, “A method of fingerprint indoor localization based on received signal strength difference by using compressive sensing,” *EURASIP J. Wirel. Commun. Netw.*, vol. 2020, no. 1, p. 72, Dec. 2020, doi: 10.1186/s13638-020-01683-8.

[36] L. Gui, T. Val, A. Wei, and R. Dalce, “Improvement of range-free localization technology by a novel DV-hop protocol in wireless sensor networks,” *Ad Hoc Netw.*, vol. 24, pp. 55–73, Jan. 2015, doi: 10.1016/j.adhoc.2014.07.025.

[37] J. Blumenthal, R. Grossmann, F. Golatowski, and D. Timmermann, “Weighted Centroid Localization in Zigbee-based Sensor Networks,” in *2007 IEEE International Symposium on Intelligent Signal Processing*, Alcala de Henares, Spain: IEEE, 2007, pp. 1–6. doi: 10.1109/WISP.2007.4447528.

[38] H. Liu, Z. Liu, Y. Chen, and W. Xu, “Determining the position of a jammer using a virtual-force iterative approach,” *Wirel. Netw.*, vol. 17, no. 2, pp. 531–547, Feb. 2011, doi: 10.1007/s11276-010-0295-6.

[39] X. Guo, N. Ansari, L. Li, and L. Duan, “A Hybrid Positioning System for Location-Based Services: Design and Implementation,” *IEEE Commun. Mag.*, vol. 58, no. 5, pp. 90–96, May 2020, doi: 10.1109/MCOM.001.1900737.

[40] P. Chen *et al.*, “Semi-Supervised Learning-Enhanced Fingerprint Indoor Positioning by Exploiting an Adapted Mean Teacher Model,” *Electronics*, vol. 13, no. 2, p. 298, Jan. 2024, doi: 10.3390/electronics13020298.

[41] H. Liu, H. Darabi, P. Banerjee, and J. Liu, “Survey of Wireless Indoor Positioning Techniques and Systems,” *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 37, no. 6, pp. 1067–1080, Nov. 2007, doi: 10.1109/TSMCC.2007.905750.

[42] X. Wang, L. Gao, S. Mao, and S. Pandey, “CSI-based Fingerprinting for Indoor Localization: A Deep Learning Approach,” *IEEE Trans. Veh. Technol.*, pp. 1–1, 2016, doi: 10.1109/TVT.2016.2545523.

[43] V. S. Kulathunga Hettiarachchige, “Ensemble Machine Learning Techniques for LoRa-based Wireless Indoor Localization Systems,” Dissertation, Uppsala University, 2024. [Online]. Available: https://uu.diva-portal.org/smash/get/diva2:1901610/FULLTEXT01.pdf

[44] Q. Zhu, Q. Xiong, K. Wang, W. Lu, and T. Liu, “Accurate WiFi-based indoor localization by using fuzzy classifier and mlps ensemble in complex environment,” *J. Frankl. Inst.*, vol. 357, no. 3, pp. 1420–1436, Feb. 2020, doi: 10.1016/j.jfranklin.2019.10.028.

[45] J. Yoo, “Wi-Fi Fingerprint Indoor Localization by Semi-Supervised Generative Adversarial Network,” *Sensors*, vol. 24, no. 17, p. 5698, Sep. 2024, doi: 10.3390/s24175698.

[46] Encord, “What is Ensemble Learning?,” Encord Blog. Accessed: Apr. 07, 2025. [Online]. Available: https://encord.com/blog/what-is-ensemble-learning/

[47] IBM, “What Is Bagging? | IBM,” IBM. [Online]. Available: https://www.ibm.com/think/topics/bagging

[48] Y. Wang, C. Xiu, X. Zhang, and D. Yang, “WiFi Indoor Localization with CSI Fingerprinting-Based Random Forest,” *Sensors*, vol. 18, no. 9, p. 2869, Aug. 2018, doi: 10.3390/s18092869.

[49] S. Jin and D. Kim, “WiFi Fingerprint Indoor Localization Employing Adaboost and Probability-One Access Point Selection for Multi-Floor Campus Buildings,” *Future Internet*, vol. 16, no. 12, p. 466, Dec. 2024, doi: 10.3390/fi16120466.

[50] Mawahib Sharafeldin Adam Boush, “Enhancing IoT Network Attack Detection with Ensemble Machine Learning and Efficient Feature Extraction,” *J. Inf. Syst.*

Eng. Manag., vol. 10, no. 20s, pp. 288–298, Mar. 2025, doi: 10.52783/jisem.v10i20s.3054.

[51] M. Salimibeni and A. Mohammadi, “Hybrid Indoor Localization via Reinforcement Learning-based Information Fusion,” 2022, *arXiv*. doi: 10.48550/ARXIV.2210.15132.

[52] J. Wang, Y. Fu, H. Feng, and J. Wang, “Transfer Learning for Indoor Localization Algorithm Based on Deep Domain Adaptation,” *Sensors*, vol. 23, no. 23, p. 9334, Nov. 2023, doi: 10.3390/s23239334.

[53] T. Suwannaphong, R. McConville, and I. Craddock, “Transfer Learning of RSSI to Improve Indoor Localisation Performance,” 2024, *arXiv*. doi: 10.48550/ARXIV.2412.09292.

[54] H. Pirayesh and H. Zeng, “Jamming Attacks and Anti-Jamming Strategies in Wireless Networks: A Comprehensive Survey,” *IEEE Commun. Surv. Tutor.*, vol. 24, no. 2, pp. 767–809, 2022, doi: 10.1109/COMST.2022.3159185.

[55] J. Petajajarvi, K. Mikhaylov, M. Hamalainen, and J. Iinatti, “Evaluation of LoRa LPWAN technology for remote health and wellbeing monitoring,” in *2016 10th International Symposium on Medical Information and Communication Technology (ISMICT)*, Worcester, MA, USA: IEEE, Mar. 2016, pp. 1–5. doi: 10.1109/ISMICT.2016.7498898.

[56] I. Martinez, P. Tanguy, and F. Nouvel, “On the performance evaluation of LoRaWAN under Jamming,” in *2019 12th IFIP Wireless and Mobile Networking Conference (WMNC)*, Paris, France: IEEE, Sep. 2019, pp. 141–145. doi: 10.23919/WMNC.2019.8881830.

[57] W. Xu, W. Trappe, Y. Zhang, and T. Wood, “The feasibility of launching and detecting jamming attacks in wireless networks,” in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, Urbana-Champaign IL USA: ACM, May 2005, pp. 46–57. doi: 10.1145/1062689.1062697.

[58] T.-H. To and A. Duda, “Simulation of LoRa in NS-3: Improving LoRa Performance with CSMA,” in *2018 IEEE International Conference on Communications (ICC)*, Kansas City, MO: IEEE, May 2018, pp. 1–7. doi: 10.1109/ICC.2018.8422800.

[59] E. Aras, G. S. Ramachandran, P. Lawrence, and D. Hughes, “Exploring the Security Vulnerabilities of LoRa,” in *2017 3rd IEEE International Conference on*

Cybernetics (CYBCONF), Exeter, United Kingdom: IEEE, Jun. 2017, pp. 1–6. doi: 10.1109/CYBConf.2017.7985777.

[60] A. Proano and L. Lazos, "Selective Jamming Attacks in Wireless Networks," in *2010 IEEE International Conference on Communications*, Cape Town, South Africa: IEEE, May 2010, pp. 1–6. doi: 10.1109/ICC.2010.5502322.

[61] J. M. Marais, R. Malekian, and A. M. Abu-Mahfouz, "LoRa and LoRaWAN testbeds: A review," in *2017 IEEE AFRICON*, Cape Town: IEEE, Sep. 2017, pp. 1496–1501. doi: 10.1109/AFRCON.2017.8095703.

[62] H. C. Yildirim, M. F. Keskin, H. Wymeersch, and F. Horlin, "Deceptive Jamming in WLAN Sensing," Jan. 02, 2024, *arXiv*: arXiv:2401.01101. doi: 10.48550/arXiv.2401.01101.

[63] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures," in *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, 2003.*, Anchorage, AK, USA: IEEE, 2003, pp. 113–127. doi: 10.1109/SNPA.2003.1203362.

[64] C. Del-Valle-Soto, L. J. Valdivia, R. Velázquez, J. A. Del-Puerto-Flores, J. Varela-Aldás, and P. Visconti, "Adaptive Jamming Mitigation for Clustered Energy-Efficient LoRa-BLE Hybrid Wireless Sensor Networks," *Sensors*, vol. 25, no. 6, p. 1931, Mar. 2025, doi: 10.3390/s25061931.

[65] F. Yang, N. Shu, C. Hu, J. Huang, and Z. Niu, "Jammer Location-Aware Method in Wireless Sensor Networks Based on Fibonacci Branch Search," *J. Sens.*, vol. 2023, no. 1, p. 2261730, Jan. 2023, doi: 10.1155/2023/2261730.

[66] I. Ullah and Q. H. Mahmoud, "Design and Development of a Deep Learning-Based Model for Anomaly Detection in IoT Networks," *IEEE Access*, vol. 9, pp. 103906–103926, 2021, doi: 10.1109/ACCESS.2021.3094024.

[67] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Future Gener. Comput. Syst.*, vol. 100, pp. 779–796, Nov. 2019, doi: 10.1016/j.future.2019.05.041.

[68] S. Garcia, A. Parmisano, and M. J. Erquiaga, "IoT-23: A labeled dataset with malicious and benign IoT network traffic." Zenodo, Jan. 20, 2020. doi: 10.5281/ZENODO.4743746.

[69] M. Babazadeh, “LoRa-Based Anomaly Detection Platform: Center and Sensor-Side,” *IEEE Sens. J.*, vol. 20, no. 12, pp. 6677–6684, Jun. 2020, doi: 10.1109/JSEN.2020.2976650.

[70] A. Kurniawan and M. Kyas, “Machine Learning Models for LoRa Wan IoT Anomaly Detection,” in *2022 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, Depok, Indonesia: IEEE, Oct. 2022, pp. 193–198. doi: 10.1109/ICACSIS56558.2022.9923439.

[71] N. S. Senol, A. Rasheed, M. Baza, and M. Alsabaan, “Identifying Tampered Radio-Frequency Transmissions in LoRa Networks Using Machine Learning,” *Sensors*, vol. 24, no. 20, p. 6611, Oct. 2024, doi: 10.3390/s24206611.

[72] S. M. Danish, A. Nasir, H. K. Qureshi, A. B. Ashfaq, S. Mumtaz, and J. Rodriguez, “Network Intrusion Detection System for Jamming Attack in LoRaWAN Join Procedure,” in *2018 IEEE International Conference on Communications (ICC)*, Kansas City, MO: IEEE, May 2018, pp. 1–6. doi: 10.1109/ICC.2018.8422721.

[73] I. Martinez, “(PDF) Jamming on LoRaWAN Networks : from modelling to detection,” Institut National des Sciences Appliquées de Rennes, 2021. Accessed: Apr. 11, 2025. [Online]. Available: https://www.researchgate.net/publication/350939008_Jamming_on_LoRaWAN_Networks_from_modelling_to_detection

[74] B. Upadhyaya, S. Sun, and B. Sikdar, “Machine Learning-based Jamming Detection in Wireless IoT Networks,” in *2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS)*, Singapore: IEEE, Aug. 2019, pp. 1–5. doi: 10.1109/VTS-APWCS.2019.8851633.

[75] O. Punal, I. Aktas, C.-J. Schnelke, G. Abidin, K. Wehrle, and J. Gross, “Machine learning-based jamming detection for IEEE 802.11: Design and experimental evaluation,” in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, Sydney, Australia: IEEE, Jun. 2014, pp. 1–10. doi: 10.1109/WoWMoM.2014.6918964.

[76] O. Osanaiye, A. S. Alfa, and G. P. Hancke, “A Statistical Approach to Detect Jamming Attacks in Wireless Sensor Networks,” *Sensors*, vol. 18, no. 6, p. 1691, May 2018, doi: 10.3390/s18061691.

[77] Srichard2, *Srichard2/LoRa-Jamming-Dataset*. (Mar. 16, 2023). [Online]. Available: https://github.com/Srichard2/LoRa-Jamming-Dataset

[78] Zenodi, “Drone Communication Dataset.” Kaggle, 2025. doi: https://www.kaggle.com/dsv/10702965.