MDPI

Article

A Lightweight AI-based Approach For Drone Jamming Detection

Sergio Cibecchini*, Francesco Chiti [1] and Laura Pierucci[2]

[1] Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Firenze, Florence, Italy; francesco.chiti@unifi.it
[2] Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Firenze, Florence, Italy; laura.pierucci@unifi.it
* Correspondence: sergio.cibecchini@edu.unifi.it

Abstract: The future integration of drones in 6G networks will significantly enhance their capabilities, enabling a wide range of new applications based on autonomous operation. However, drone networks are particularly vulnerable to Jamming attacks, a type of Availability attack that can disrupt network operation and hinder drone functionality. In this paper, we propose a low complexity unsupervised machine learning approach for the detection of constant and periodic jamming attacks, using the Isolation Forest algorithm. We detail the tuning of the base model as well as the integration with a Majority Rule module which significantly reduced the number of false positives caused by environmental noise, achieving high accuracy and precision. Our approach outperforms the standard Isolation Forest model in the detection of both constant and periodic jamming attacks, while still correctly identifying nominal traffic. Finally, we discuss the potential integration of the proposed solution in 6G-enabled drone networks, as a lightweight edge-based solution for enhancing security against jamming attacks.

Keywords: Jamming attacks, 6G Drone Networks, Isolation Forest, Edge AI, IoT Security

1. Introduction

The shift from 4G to 5G has been a generational leap and has revolutionized connectivity. Despite 5G still being in its early stages of adoption at the time of writing [1], the research community is already working on the next generation of wireless communication technologies, 6G. 6G technology is expected to enable a wide variety of new use cases, thanks to massive coverage as well as improvements in both data rates and latency. All of these advancements will in turn bring forth new security challenges specific to those applications.

One of the main differences between 5G and 6G technology will be a much deeper integration with Artificial Intelligence (AI). While Software Defined Networks (SDN) have played a key role in improving the efficiency and security of 5G networks, 6G is expected to take this a step further, with the networks being designed to be AI native from the ground up. This is what the authors of [2] define as the shift from *Softwarization* to *Intelligentization*.

AI integration in 6G networks will greatly strengthen the security of the network against potential threats. By leveraging Diagnostic Analytics, a collection of insights into the status of the network, security teams will be able to train specific AI models to detect and respond to security threats in real-time.

In this paper we focus on a specific aspect of the security of 6G networks, namely we provide a lightweight edge-based Machine Learning approach for the detection of jamming attacks in networks of drones. We conceptualize a scenario where a drone is subject to constant and periodic jamming and has to rely on an internal AI model to detect the attack and apply the appropriate countermeasures.

In the future, IoT devices are expected to come equipped with AI-specific chips that will allow them to run AI models directly on the edge, without the need to offload computation to a central server. This is especially useful in the case of jamming attacks,

Citation: Cibecchini, S.; Chiti, F.; Pierucci, L. A Lightweight AI-based Approach For Drone Jamming Detection. *Future Internet* **2024**, *1*, 0. https://doi.org/

Received:
Revised:
Accepted:
Published:

where the communication between the device and the server is disrupted. Our solution is designed to run on a resource-constrained IoT device like a drone and to be lightweight, both in terms of memory usage and computational power. Despite the low resource requirements, the proposed approach is able to detect both constant and periodic jamming attacks with a high degree of accuracy, as well as correctly classify normal traffic.

The paper is organized in five Sections:

1. **Topic overview:** General overview of the benefits of drone integration in 6G networks, analysis of jamming attacks: types, mitigation and detection. Discussion of the advantages provided by an edge AI approach for jamming detection.
2. **System Model:** Description of the analyzed scenario, along with the dataset, algorithms, and evaluation metrics chosen for the tests.
3. **Model Performance:** Explanation of the testing methodology, including model tuning, testing phases, and detailed description of the performance of the model.
4. **Discussion:** Analysis of the results from the previous section and the advantages of the proposed approach.
5. **Conclusions:** Overview of how the proposed solution applies to real-world scenarios, with a discussion on potential future research directions.

2. Related Works

Detection of jamming attacks in wireless networks has been a topic of interest for the research community for many years. A wide variety of approaches have been proposed for different types of networks, with most of the work focusing on the detection of jamming attacks in Vehicular Ad-Hoc Networks (VANETs)[3].

The authors of [4] analyze a wide range of supervised Machine Learning approaches for the detection of Jamming attacks, including Random Forest, Support Vector Machines and Neural Networks. The authors show that Random Forests were able to achieve a high detection of jamming attacks. The authors of [5] on the other hand, propose the use of an unsupervised approach in the context of VANETs and test its performance in a variety of real-world scenarios.

The authors of [6] compare the use of Multi-Layer Perceptrons (MLPs) and Decision Trees for the detection of jamming attacks in drone networks, showing that both methods are able to achieve good detection performance.

This work builds upon the research done in [7], where the authors analyze a Machine Learning (ML) based approach for the detection of jamming attacks. The authors propose the use of Convolutional Neural Networks (CNNs) to detect jamming attacks in an IoT drone scenario. The authors of [7] show that their CNN model is able to achieve a high detection rate against constant jamming attacks, but struggles in the classification of periodic jamming and normal traffic.

We propose a lightweight approach that integrates a specifically tuned Isolation Forest with a Majority Rule module to reduce the number of false positives and improve resistance against environmental noise. Our solution leverages the IF algorithm for jamming detection in the scenario of a 6G drone network, testing the model against the dataset created by [7].

Isolation forest has been successfully applied in a variety of scenarios for the detection of anomalies. The authors of [8] propose an Isolation Forest based approach for the detection of GPS spoofing attacks, while the authors of [9] successfully applied a combination of decision trees and Isolation Forest to classify and detect jamming attacks in a mobile scenario.

Table 1 shows a summary of the detailed approaches for jamming detection in scientific literature.

Table 1. Existing approaches for jamming detection in scientific literature

Reference	Data Source	Method	Metrics
[4]	Simulation	Random Forest	BPR, PDR, RSS, CCA
[7]	Real data	CNN	RSS
[4]	Simulation	Linear SVM	BPR, PDR, RSS, CCA
[9]	Real data	DT + IF	PDR, BPR, SNR, RSSI
[6]	Real + Simulated data	DT + MLP	Throughput, PDR, RSSI
Our solution	Real data	IF	RSS

We decided on the Isolation Forest algorithm for its low computational and memory requirements and good performance in the detection of anomalies. Our approach reaches a high detection rate against both constant and periodic jamming attacks, while also being able to classify normal traffic with a high degree of accuracy. We also detail the tuning of the model's hyperparameters to fit our specific scenario and compare it against the default model. In the tuning phase, the resource-constrained nature of the scenario was a key deciding factor in the selection of the more appropriate hyperparameters.

3. Topic Overview

3.1. 6G Drone Networks

Drones, also known as Unmanned Aerial Vehicles (UAVs), are defined as *all aircraft designed to fly without a pilot on board* [10]. This technology has experienced rapid growth in recent years and is expected to continue growing in both the consumer sector as well as the commercial and military sectors [11].

In technical report 22.886 [12], 3GPP identifies some of the envisioned use cases for 5G V2X (Vehicle-to- Everything) communication services. Among these use cases, advanced and remote driving, vehicle platooning and extended situational awareness are identified as some of the main benefits of V2V communication. All these capabilities can be leveraged by a 6G drone network to achieve fast and reliable drone-to-drone communication that, with the integration of artificial intelligence, would allow the drones to act autonomously in a coordinated manner.

This would prove useful in a variety of fields: from autonomous irrigation as well as soil and crop health assessment in agriculture, delivery of life-saving supplies and identification of survivors in disaster response, in the delivery sector as a more eco-friendly alternative to traditional delivery options, and potentially in the mobility sector as a complement to traditional taxis and public transportation [13].

In the military sector, the effectiveness of drones is widely recognized, with both high and low end models being used in a variety of roles, from more passive roles like surveillance and intelligence gathering, to more active roles like delivery of explosives and targeted strikes. Securing the communication channel against jamming attacks is crucial in all these applications, especially in a safety critical environment.

3.2. Understanding Jamming Attacks

Jamming attacks are a type of Denial of Service (DoS) attack that aims at disrupting the physical communication between two or more devices. This is achieved by transmitting a signal on the same frequency as the one used by the devices to communicate. If the jamming signal power level is high enough, it is able to overwhelm the legitimate signal, effectively blocking the communication between the devices [13]. Since the ability to communicate is affected, jamming attacks falls under the umbrella of attacks that target the *Availability* of the service in the CIA triad [14]. Jamming attacks can be classified into 5 main categories, based on the attack pattern of the jammer:

- **Constant Jamming:** The jammer continuously transmits a strong signal on the same frequency used by the devices it wants to disrupt to communicate.

- **Periodic Jamming:** The jammer transmits a strong signal for a certain period of time t_{on}, then stops transmitting for another period of time t_{off}. This cycle is repeated until the attack ends.
- **Random Jamming:** In random jamming, the jammer is active at random intervals, jamming each transmitted packet with a probability p based on a random pattern [3].
- **Reactive Jamming:** A reactive jammer starts transmitting its jamming signal only when it senses energy in the communication channel, indicating that a legitimate transmission is taking place. This type of jamming attack is more power-efficient compared to other attack patterns, as it only transmits its signal when it knows that it can actually disrupt the communication [15].
- **Smart Jamming:** A smart jammer is a more advanced and computationally intensive type of jammer that is able to adapt its jamming signal to maximize the disruption of the communication between the devices. Smart jammers employ a traffic analysis module, meaning they are able to modify their attack pattern based on the transmission specifics of the devices they are targeting and adapt to changes in the communication channel [16].

3.3. Jamming attacks against drone networks

Jamming attacks are particularly effective against drone networks, as drones usually rely on external input to navigate and operate correctly. If a jammer were able to completely block the communication between the drone and the Base Station (BS), the drone would be left without any indication on how to behave and would need to activate an internal failsafe mechanism. This usually comes in the form of either a *return to base* procedure, a *hover in place* procedure or a *land in place* procedure. All of these approaches leave the drone in a vulnerable position, as a bad actor could potentially capture the drone and use it for malicious purposes. This is especially true when jamming attacks are used in combination with other types of attacks, such as spoofing attacks. A jamming attack can be initiated against a drone, making it lose connection with the central server. The drone would be forced to start a failsafe procedure, like *return to base*. The attacker could then employ a GPS spoofing attack to make the drone believe that the Base Station is in a different location, leading to the drone landing in a spot designated by the attacker [17].

3.4. Centralized vs Edge approach for Jamming detection

When presenting a ML based approach in an IoT setting, the question of where the AI model should be placed often arises. By their nature, IoT devices, and in turn drones, are usually resource-constrained, both in terms of computational power and also in terms of internal storage and battery capacity [18]. This means that complex AI models and algorithms are usually not feasible to be run on the device itself.

A centralized approach offloads the computational burden to a remote central server, that returns the results of the AI model to the device. While this scheme might be favorable in some cases, in a real-time situation, such as a jamming attack, the latency introduced by the communication with the server reduces the time to react to the threat. Also, as jamming attacks degrade or sometimes completely block the communication between the device and the BS, a centralized approach might not be feasible in this case. A collaborative edge approach faces similar issues, as the communication between the devices would likely be disrupted by the attack. Implementing a lightweight on-device edge approach, on the other hand, while not as precise as a centralized approach, would provide real-time results and would be able to operate even when the communication is disrupted, as is the case in a jamming situation.

3.5. Detection and Mitigation of Jamming Attacks

The state-of-the-art approaches for jamming detection involve monitoring of the communication link and the analysis of metrics such as the Signal to Noise Ratio (SNR), the Received Signal Strength (RSS) or the Packet Delivery Ratio (PDR) [19]. The simplest

approach when developing a method to detect jamming attacks is to set a static threshold for these metrics and trigger an alarm when the threshold is crossed. While this approach is easy to implement and might be effective in some cases, it is not able to adapt to changes in the communication channel and might be prone to false positives. Implementing a ML based approach for jamming detection would allow the system to adapt to the changes in the communication channel and would be able to provide a more accurate detection of jamming attacks [3]. This is especially useful in mobile scenarios like drone networks, where the environment is constantly changing and the signal strength can vary greatly.

Once an attack is successfully detected, state-of-the-art jamming mitigation techniques, like Direct Sequence Spread Spectrum (DSSS), Frequency Hopping (FHSS) and advanced signal processing techniques can be put in place to mitigate the effects of the attack.

The use of an AI model for jamming detection is particularly suitable for a 6G network, as the improved sensing capabilities would allow the model to train on a wide variety of diagnostic data, improving the detection rate and accuracy of the model. Furthermore, as the network is expected to be integrated with AI from the ground up, it is likely that in the future drones will be equipped with AI-specific chips for a more efficient execution of AI models.

4. System Model

4.1. Scenario Definition and Attacker Classification

In the proposed scenario (Figure 1) we have a flying drone that is subject to a jamming attack. The drone communicates with a ground station and periodically samples the Received Signal Strength (RSS) of the signal it receives. The drone implements a simple unsupervised machine learning module, trained on nominal traffic RSS values. The module takes the sampled RSS values and classifies them as either normal or anomalous, making the drone able to determine whether or not a jamming attack is taking place.

We assume that a jammer is deployed in the network and that it starts a jamming attack at a certain instant by transmitting a high-power signal on the same frequency as the one used by the drone to communicate with the ground station. We assume that the jammer is able to completely block the communication between the drone and the ground station, meaning that the drone has to rely on its internal resources to determine if a jamming attack is taking place and react accordingly. We assume that the jammer type is unknown to the drone.

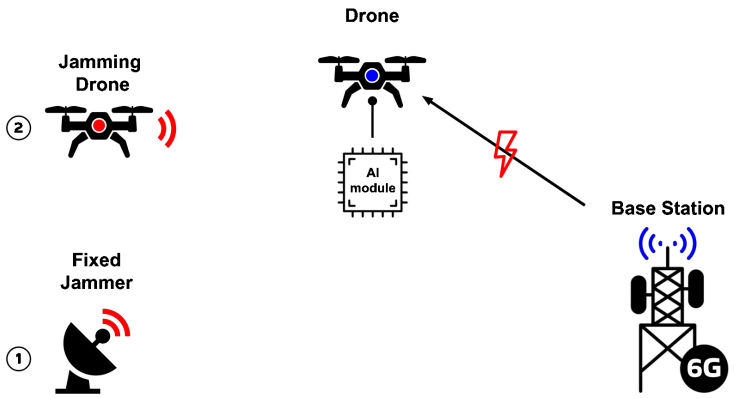


Figure 1. Proposed jamming scenario. The Drone communicates with the 6G BS and receives information about how to behave. The fixed jammer **(1)** and the mobile jammer **(2)** target the communication between the drone and the Base Station. The drone has an internal classification module capable of detecting jamming attacks.

In our scenario, the drone is subject to two types of jamming attacks, a constant and a periodic jamming attack that can be attributed to two different types of jammers. The

constant jamming attack could be caused by a fixed ground jammer, as maintaining a constant jamming signal requires a large power source, while the periodic jamming attack could be caused by a mobile jammer, for example a drone, as drones are usually battery powered and can only jam for a limited amount of time. Employing a periodic jamming attack allows the malicious drone to preserve energy and lowers the chances of detection.

The attacker classification [15] of the proposed scenario is detailed in Table 2:

Table 2. Attacker classification details.

Classification	Description
Active	The attacker is actively trying to disrupt network operation by transmitting a jamming signal
External	The attack takes place at level 1 of the OSI model, meaning that the attacker is not part of the network
Local	The attack is local, as it is targeted at a specific drone or drone cluster and not at the entire network
Malicious	Jamming attacks are considered malicious as their main goal is to disrupt correct network operation

4.2. Dataset Choice

As 6G network traffic is not yet widely available, we chose an open-source dataset [7] that analyses the RSS values received by a Raspberry Pi 3 which is subject to periodic and constant jamming attacks.

The dataset in [7] was created using a software-defined radio (SDR) connected to a laptop that was programmed to transmit a jamming signal using the open-source software *GNU Radio*. A second SDR radio was connected to a Raspberry Pi 3, designated to receive the jamming signal. The jamming radio operates at a frequency of $2.412GHz$ with a bandwidth of $40MHz$, while the Raspberry Pi 3 was programmed to sample the RSS values with a frequency of $32K$ samples per second.

The dataset contains three different *.txt* files that store the RSS values sampled by the Raspberry Pi 3. The first file contains the RSS values sampled during a constant jamming attack, the second file contains the RSS values sampled during a periodic jamming attack and the third file contains the RSS values sampled during normal network operation. The samples are stored in a single column, with each row representing a single sampled RSS value, expressed in *dBm (decibel-milliwatts)*.

Figures 2 and 3 show, respectively, plots of the RSS values sampled during a constant jamming attack and a periodic jamming attack, while Figure 4 shows the RSS values sampled during normal network operation.

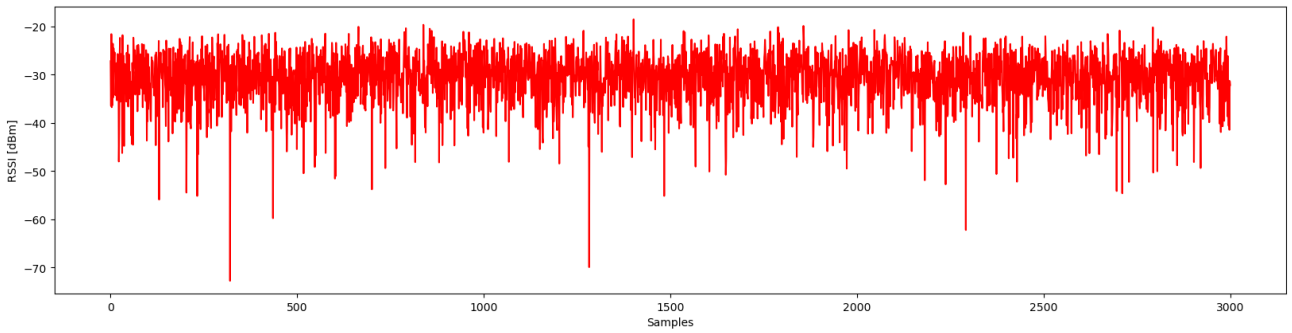


Figure 2. Constant jamming attack RSS values

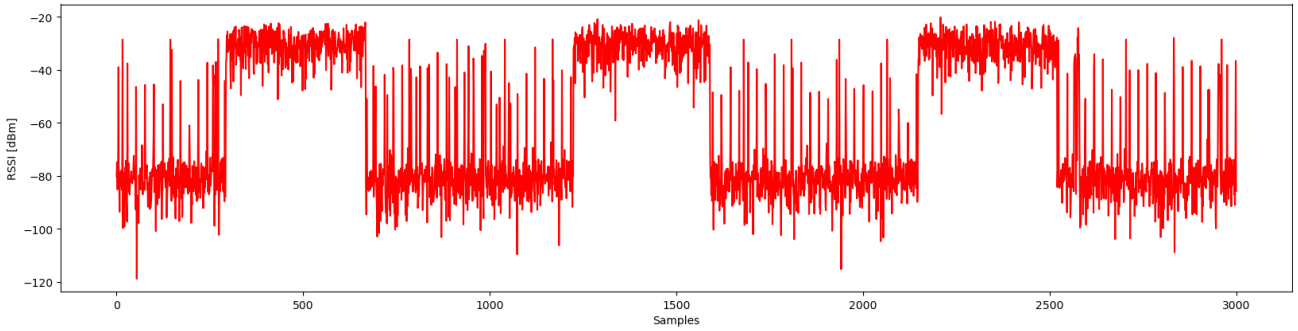


Figure 3. Periodic jamming attack RSS values

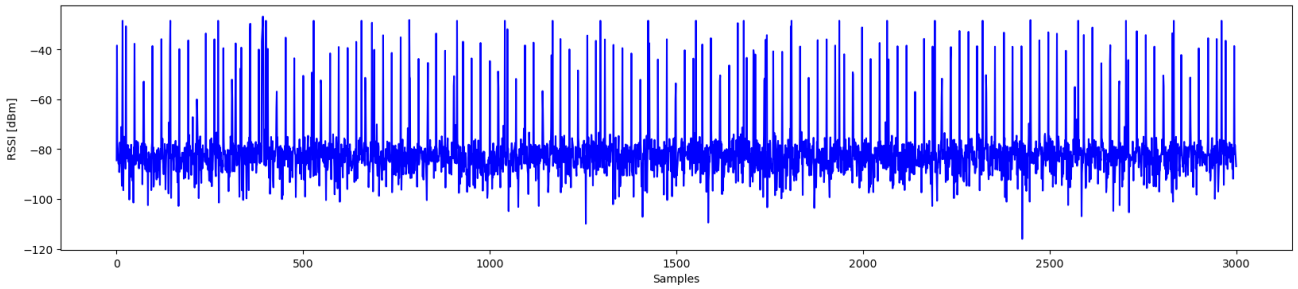


Figure 4. Normal network operation RSS values

4.3. Detection Algorithm

When choosing the algorithm for the classification module, we first had to define the requirements that it has to meet. The algorithm has to have the ability to classify anomalous samples with a high detection rate (Recall), while at the same time being lightweight enough in terms of memory usage and computational power to be run on a resource-constrained device like a drone. The training phase should also be fast, as this would allow the drone to quickly determine the normal transmission RSS values in a constantly changing environment. A periodic redefinition of normal RSS values would limit the number of false positives caused by drone mobility. The model should also require a small amount of storage to run effectively, as drones are usually very limited in internal memory.

The algorithm that we found to be the best fit for these requirements is the *Isolation Forest* algorithm [20]. The Isolation Forest (IF) algorithm is a state-of-the-art unsupervised ML algorithm for anomaly detection, introduced by Liu et al. in 2008. Unlike traditional anomaly detection algorithms, which require the definition of a normal class, IF is able to detect anomalies without explicitly defining their characteristics. This is achieved by leveraging the properties of anomalies themselves, i.e., being rare and separated from the majority of the data points.

The algorithm works by building a forest of isolation trees. Each tree is built by randomly selecting one of the features and then splitting the data points based on a value randomly selected between the minimum and maximum value of the feature. This partitions the data into the left and right branches of the tree. The process is repeated recursively until all the data points are isolated. The height of a node, meaning the number of splits required to isolate it, is used to determine the anomaly score of the data point. The greater the number of splits, the more likely the data point is to be an anomaly. IF is an *ensemble* algorithm, meaning that the final anomaly score is calculated by averaging the anomaly scores of all the trees in the forest. More trees result in greater accuracy but also more computational power required.

The algorithm was chosen for its simplicity, low computational requirements (both in terms of memory and computational power) and proven effectiveness in the detection

of anomalies[20]. While more complex solutions like Neural Networks or Support Vector Machines might prove more effective in certain scenarios, they are best suited in a situation where computation power is not a limiting factor. The unsupervised nature of the algorithm was also a determining factor in its selection, as it would allow for a periodical redefinition of the normal RSS values based on the current environment, without the need to rely on a fixed labeled dataset. Also, thanks to the ability to work on high dimensional data, the algorithm could potentially be scaled to work with more complex datasets in the future, if the drones were to be equipped with more advanced sensors.

The chosen implementation for IF is the one provided by the *Scikit-learn* library [21] for Python 3.

4.4. Evaluation Metrics

The performance of the model was evaluated using the state-of-the-art evaluation metrics for ML classification algorithms. The metrics are all based on the number of true positives, false positives, true negatives and false negatives. The chosen evaluation metrics are the following:

- **Accuracy:** the ratio of correctly classified data points to the total number of data points.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

- **Precision:** the ratio of correctly classified anomalies to the total number of data points classified as anomalies.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

- **Recall:** the ratio of correctly classified anomalies to the total number of anomalies.

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

- **F1 Score:** the harmonic mean of the precision and recall.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

The combination of these metrics gives us concrete insight into the performance of the model.

5. Model Performance

5.1. Parameters Tuning Phase

The first phase in our testing methodology involves tuning the model's hyperparameters by varying them between a minimum and a maximum value, evaluating the impact on the performance metrics (detailed in Section 4.4) and then choosing the best possible values. In our scenario, a value is considered good if it provides a high metric value while at the same time not impacting too drastically the execution time. In cases where improving the value of a metric requires a significant increase in execution time, we chose the value that provides the best trade-off between the two.

Among the parameters available in the *Scikit-learn* implementation of the Isolation Forest, we decided to tune the following:

- **n_estimators:** the number of base estimators in the ensemble, i.e., the number of isolation trees used to compute the anomaly score of each data point.
- **max_samples:** the max number of samples to draw from the dataset to train each tree.
- **contamination:** the amount of outliers present in the training dataset.

During the tuning phase, the untested hyperparameters were set to the default values of the *Scikit-learn* implementation of the Isolation Forest (Table 3).

Table 3. Scikit-Learn Isolation Forest hyperparameters default values.

Parameter	Default Value
n_estimators	100
max_samples	'auto'
contamination	0.1

The model was evaluated using input composed of normal traffic samples concatenated with jamming attack samples (see code snippet 1). This was done to simulate a real-world scenario where the model has to be able to correctly classify anomalous samples but also reduce the number of false positives during normal operation.

Algorithm 1 Test input definition

```
normalTraffic ← ReadAndParseFile(NORMAL_TRAFFIC_FILE, normal_traffic_size)
jamming ← ReadAndParseFile(JAMMING_FILE, jamming_size)
testInput ← Concatenate(normalTraffic, jamming)
```

The selected values for *normal_traffic_size* and *jamming_size* are shown in Table 4.

Table 4. Dataset sizes used for the tuning and testing phases.

Dataset	Size
normal_traffic_size	20,000
jamming_size	2,000

The dataset is intentionally unbalanced between normal data points and anomalous points, as jamming attacks are usually rare events compared to normal network operation. In Figure 5 we can see a representation of the input signal in the case of a constant jamming attack. The proposed results, unless otherwise specified, are based on the input signal shown in Figure 5, which employs constant jamming. Periodic jamming has always shown comparable trends as constant jamming in all the performed tests.

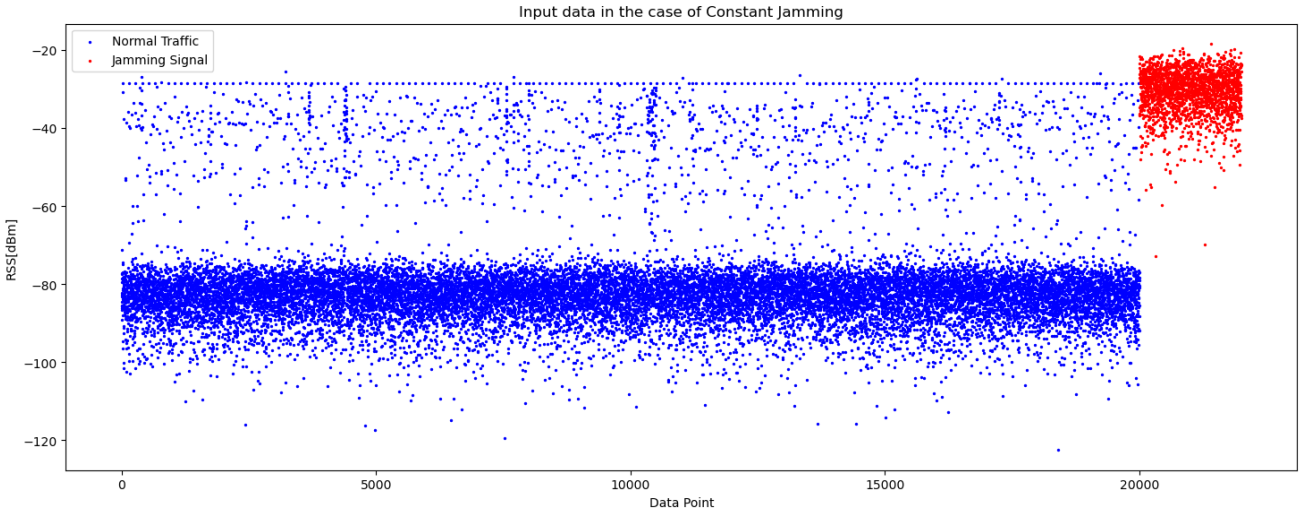


Figure 5. Input signal in the case of Constant jamming. The normal traffic signal is concatenated with the jamming signal.

5.1.1. n_estimators tuning

The first parameter that we decided to tune was the *n_estimators* parameter. The tuning values are shown in Table 5.

Table 5. n_estimators tuning values.

n_estimators	Values
Minimum	1
Maximum	50
Step	1

The effect on the evaluation metrics of the model is shown in Figure 6.

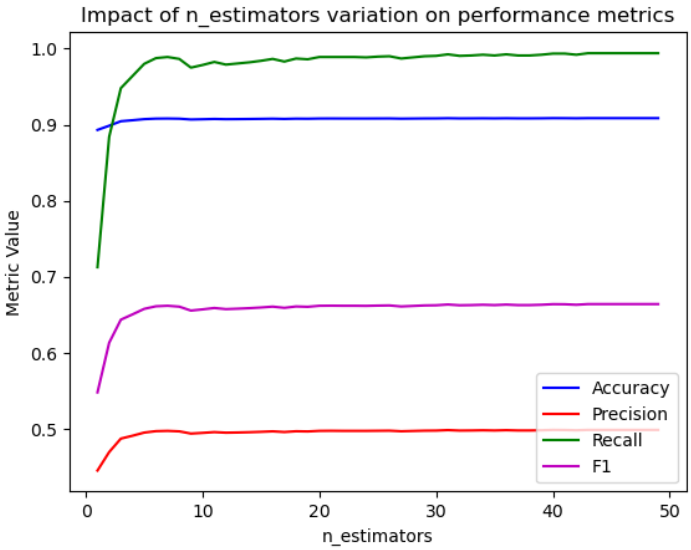
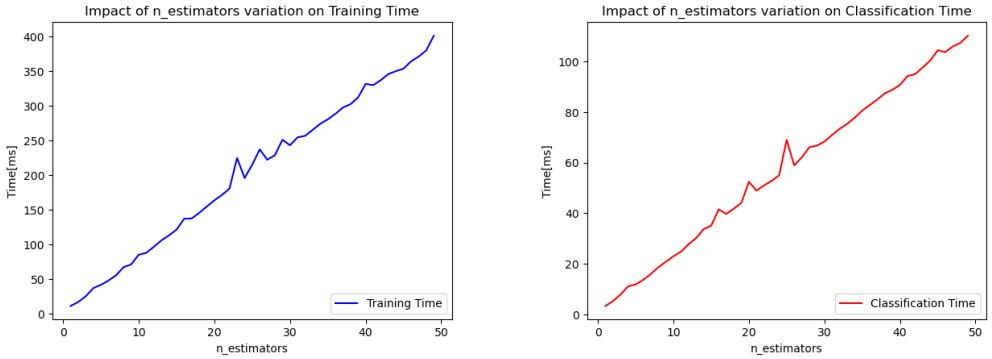


Figure 6. n_estimators tuning results. Performance of the model in all analyzed metrics increases with an increasing number of estimators

As we can see from the graph, performance increases as the number of estimators increase, converging to a stable value. Graphs 7a and 7b show that the time required both for training and classification increases linearly with the number of estimators. This means that choosing an appropriate number of estimators is crucial, as it can greatly affect the model performance.



(a) n_estimators training time. **(b)** n_estimators classification time.

Figure 7. Comparison of n_estimators training and classification times. Both training and classification times increase linearly with the number of estimators.

We accordingly selected **n_estimators = 15**, as it provided a good balance between model performance and computational resources required. The model metrics keep slowly improving until 45 estimators, but the performance increase is minimal, making it not worth such a large increase in training and classification time, especially considering the resource-constrained nature of the scenario under consideration.

5.1.2. max_samples tuning

The second parameter that we decided to tune was the *max_samples* parameter. The tuning values are shown in Table 6.

Table 6. max_samples tuning values.

max_samples	Values
Minimum	1
Maximum	100
Step	1

From Figure 8 we can notice that the model performance is not greatly affected by the *max_samples* parameter. This is most likely due to the significant difference in RSS values between the normal traffic samples and the jamming attack samples, meaning the model can develop the ability to correctly classify the input even with a limited training set.

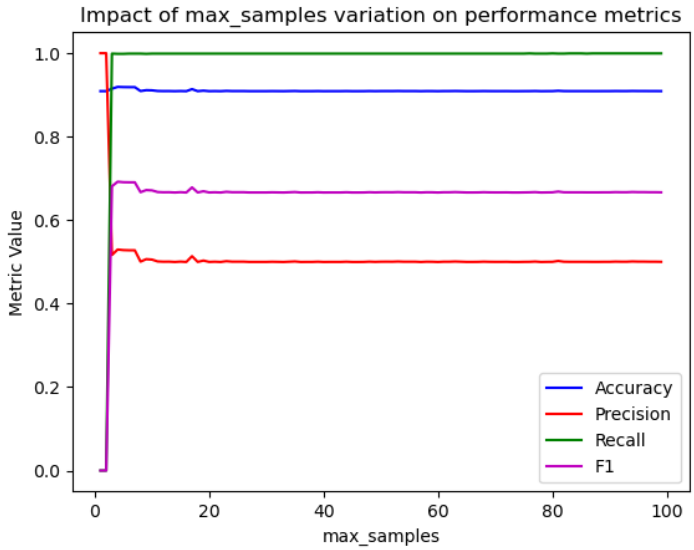
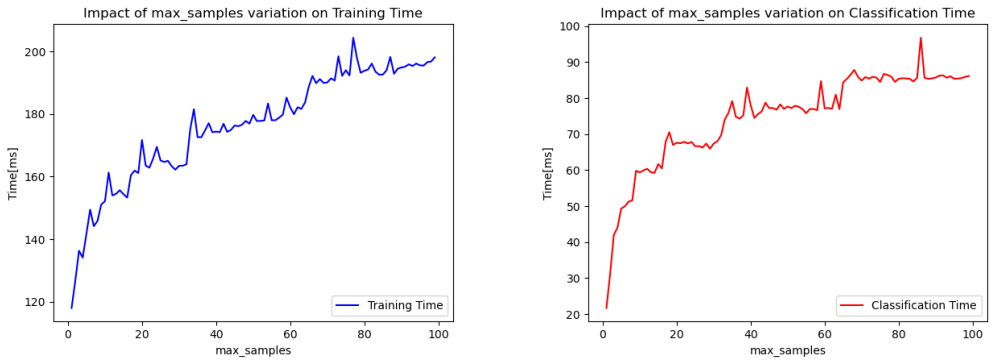


Figure 8. max_samples tuning results. The max samples parameter has limited impact on the model's performance given the large difference in RSS values between normal and anomalous samples.

However, as the model has to build bigger trees, the *max_samples* parameter has considerable influence on the training and classification time, as shown in Figure 9. Given the minimal performance differences but substantial impact on the time metrics, we chose the value of **max_samples = 10**.



(a) max_samples training time. **(b)** max_samples classification time.

Figure 9. Comparison of max_samples training and classification times. Both training and classification times follow a logarithmic trend.

5.1.3. contamination tuning

The last parameter we focused on is the *contamination* parameter. The tuning values are shown in Table 7.

Table 7. contamination tuning values.

contamination	Values
Minimum	0.01
Maximum	0.5
Step	0.01

The contamination parameter is by far the one that has the greatest impact on the model performance, as shown in Figure 10. Since recall is the metric that we want to prioritize as it highlights the number of detections, we chose the value of **contamination = 0.09**.

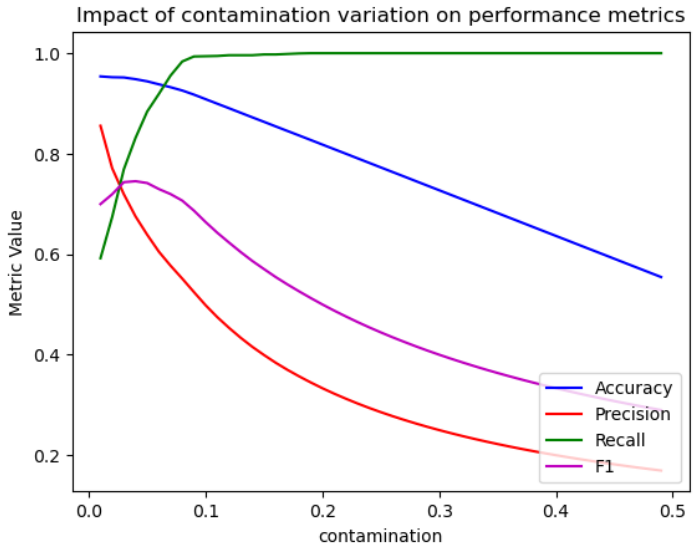


Figure 10. contamination tuning results. While recall score benefits from a higher contamination, all the other metrics are negatively impacted by a higher contamination value.

Our testing pointed out that, as expected, tuning of the contamination parameter has minimal impact on the training and classification time.

5.2. Model Testing Phase

5.2.1. Standard Model Testing

After the tuning phase, we tested the model using the best performing hyperparameters, detailed in Table 8 and compared against the default parameters.

Table 8. Tuned hyperparameters and default values

Parameter	Tuned Value	Default Value
n_estimators	15	100
max_samples	10	'auto'
contamination	0.09	0.1

The model was tested against the input signal shown in Figure 5. In Figure 11 we can see the results of the model classification of the input signal with the tuned hyperparameters.

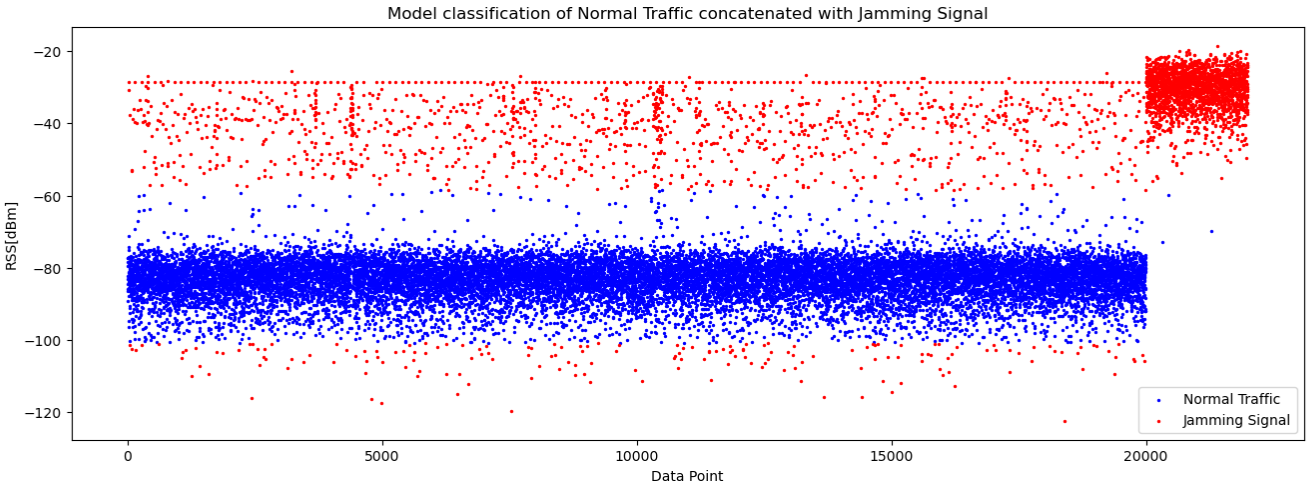


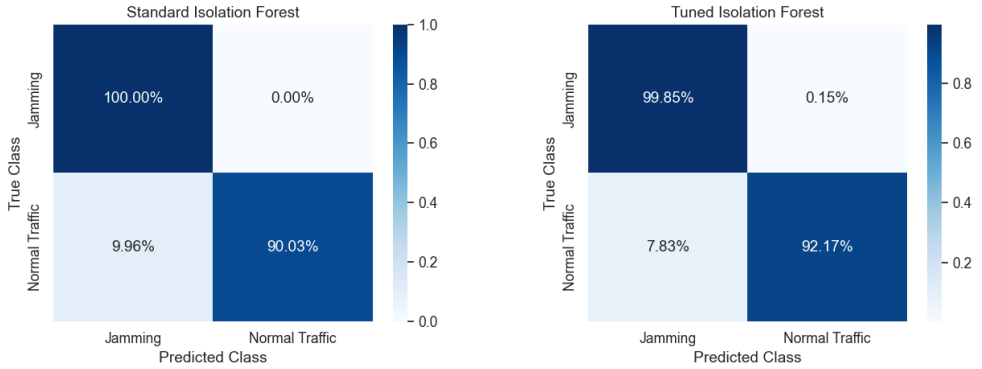
Figure 11. Classification results from the tuned Isolation Forest model.

Table 9 and 10 show a comparison between the model performance with the tuned and the default hyperparameters.

Table 9. Confusion Matrix Components Comparison.

Metric	Tuned Isolation Forest	Default Isolation Forest
TP	1997	2000
FP	1565	1993
TN	18435	18007
FN	3	0

Figures 12a and 12b show, respectively, the confusion matrix for the default and the tuned Isolation Forest model.



(a) **(b)**

Figure 12. Confusion Matrix for the default Isolation Forest model **(a)** and the tuned Isolation Forest model **(b)**. The tuned version reduces false positives while suffering a negligible increase in false negatives compared to the default model.

Table 10. Performance Metrics Comparison.

Metric	Tuned Isolation Forest	Default Isolation Forest
Accuracy	0.929	0.909
Precision	0.561	0.501
Recall	0.999	1.000
F1 Score	0.718	0.667

The tuning phase proved effective in improving the model's performance compared to the default hyperparameters, with an increase of 2% in accuracy, 6% in precision, and 5.1% in the F1 score, while maintaining a nearly identical recall (0.1% drop). However, despite reaching a very high Recall score in both scenarios, the model is still too prone to false positives, as indicated by the value of the Precision metric. This is most likely due to the nature of the normal traffic signal. As we see from Figure 11, many of the normal traffic samples have RSS values that are similar to the jamming attack samples, leading the model to classify them as anomalies. This is most likely due to how the dataset has been created, as the normal traffic samples were measured in a real-world scenario where the RSS values would be affected by external noise.

Instead of seeing this as an impairment, we view it as an improvement opportunity. The fact that the dataset contains noise means it is more representative of a real-world scenario, where the drone might be moving through different environments and thus be subject to environmental noise.

5.2.2. Majority Rule model testing

Having understood the nature of the problem, we decided to implement a Majority Rule (MR) system to reduce the number of false positives. The improved model would use the Tuned Isolation Forest model's results and then pass them to a MR module, as shown in Figure 13.

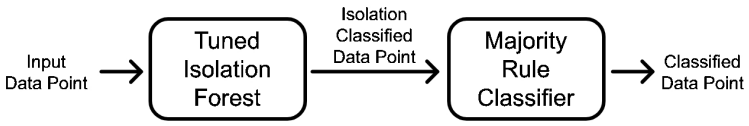


Figure 13. Majority Rule system diagram. The Tuned Isolation Forest model results are passed to the MR module, which then provides the final classification.

The logic behind the Majority Rule module is simple. The MR takes as input the classification results of the IF model and inserts them into a sliding window. If the data point is classified by the Tuned IF as an anomaly, the MR module checks the contents of the sliding window. If the majority of the data points in the window are classified as anomalies, the data point is classified as an anomaly. If the majority of the data points in the window are classified as normal, then the data point is classified as normal (see code snipped 2).

Algorithm 2 Majority Rule Algorithm

```
    window ← []
for each index in range(len(classification)) do
    if length of window equals predefined window size then
        window.pop(0)
    end if
    window.append(classification[index])
    if current classification is OUTLIER and the count of OUTLIERS in the window
     is less than or equal to half the window size then
        classification[index] ← INLIERS
    end if
end for
    return classification
```

To determine the optimal value for the *window_size* parameter, we test the model in the same way as in the hyperparameters tuning phase. The model parameters were set to the best performing values from the tuning phase (Table 8). Table 11 shows the tuning values for the *window_size* parameter.

Table 11. window_size tuning values.

window_size	Values
Minimum	1
Maximum	100
Step	1

From Figure 14 we can see that the *window_size* parameter has a considerable impact on the model performance.

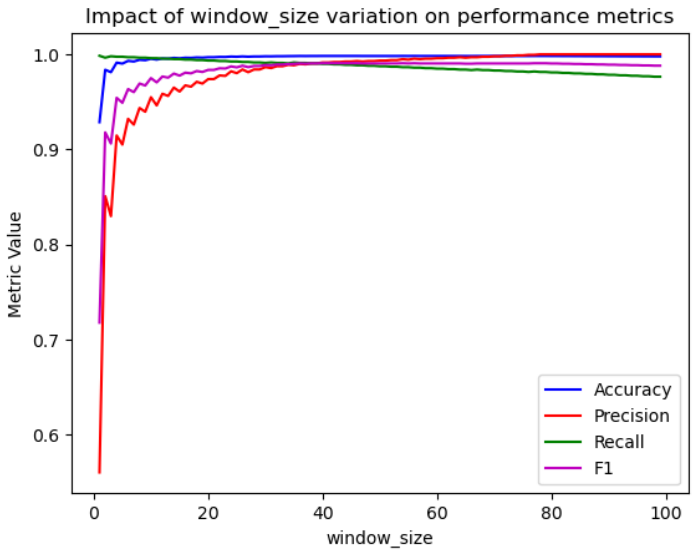


Figure 14. window_size tuning results. Most metrics initially benefit from a larger window size except for recall that suffers a linear decrease.

While accuracy and precision benefit from a larger window size, we can observe a linear decrease in recall as the window size increases. This is due to the nature of the majority rule, as a larger window size means that a larger number of anomalous data points must be present in the window in order to classify the current data point as an anomaly. The optimal value for the *window_size* parameter was found to be **window_size = 39**, as it provides high accuracy and precision while maintaining a high recall.

After defining the value for the *window_size* parameter, we tested the model against the same input signal used for the standard tuned model (Figure 5). The results of the model classification are shown in Figure 15.

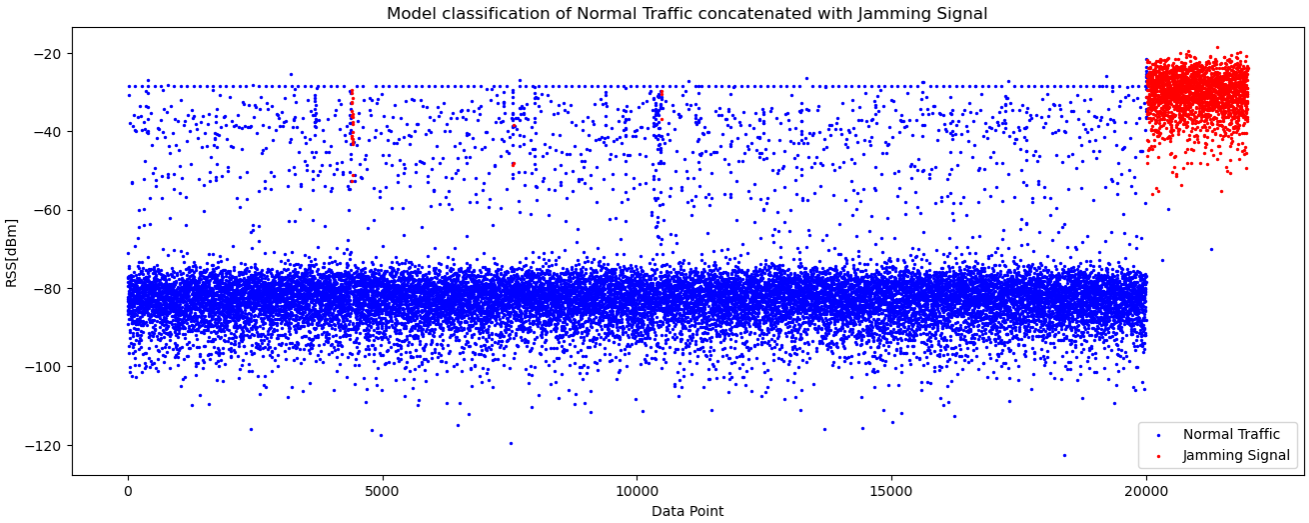
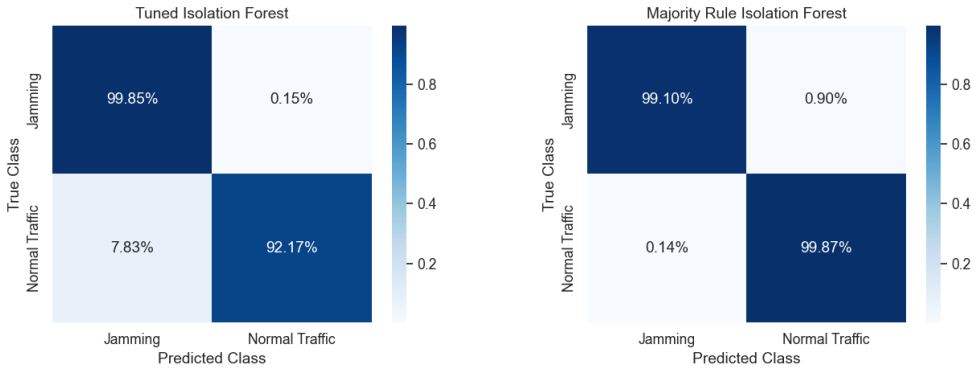


Figure 15. Classification results from the MR Isolation Forest model

In Tables 12 and 13 we can see a comparison between the tuned model and the MR model.

Table 12. Confusion Matrix Components Comparison.

Metric	Tuned Isolation Forest	Majority Rule Isolation Forest
TP	1997	1982
FN	3	18
FP	1565	27
TN	18435	19973



(a) **(b)**

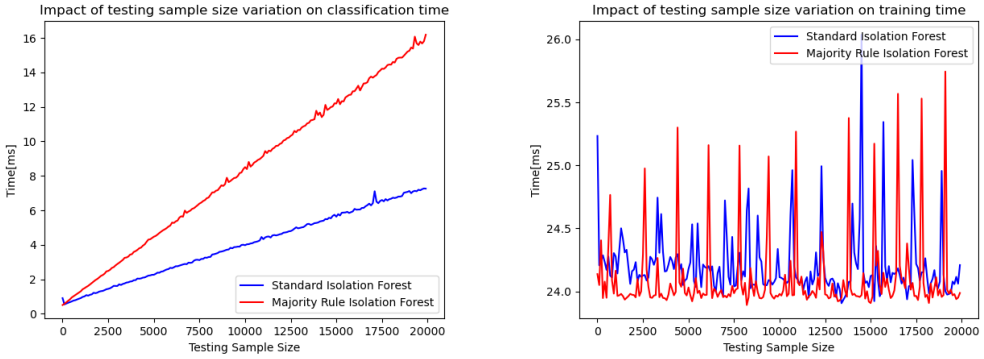
Figure 16. Confusion Matrix for the tuned Isolation Forest model **(a)** and the Majority Rule Isolation Forest model **(b)**. The MR model significantly reduces false positives while maintaining a rate of true positives of over 99%.

Table 13. Performance Metrics Comparison.

Metric	Tuned Isolation Forest	Majority Rule Isolation Forest
Accuracy	0.929	0.998
Precision	0.561	0.987
Recall	0.999	0.991
F1 Score	0.718	0.989

The results show that the Majority Rule Model is effective in mitigating the high number of false positives that were present in the first version of the model. The MR integration resulted in an improvement in Precision of 42.6%, in Accuracy of 6.9% and in F1 Score of 27.1%, while suffering only a less than 1% drop in Recall. In the original model, most of the false positives occurred due to the normal traffic samples containing a high number of isolated data points with similar RSS values to those of the jamming samples. The MR module addresses this by checking if the previous points were also classified as anomalies. This approach proves to be effective, as jamming attacks are rarely composed of isolated data points.

Given the resource-constrained nature of the scenario, we decided to test the computational impact of the MR module against the standard tuned model. Classification times and training times were measured for both models and the results are shown in Figures 17a and 17b.



(a) **(b)**

Figure 17. Comparison of the standard tuned model and the MR in terms of classification time **(a)** and training time **(b)**.

Figure 17a shows that there is indeed a difference in classification time, with the standard model being faster. This is due to the additional computational complexity added by the Majority Rule module, which requires updating and checking of the sliding window for each classified data point. Figure 17b on the other hand shows that, as expected, training time is not affected by the MR module.

5.2.3. Comparison against existing solution

As mentioned in Section 4.2, this work is based on the work of [7], where the authors proposed a solution for jamming attack detection based on CNNs. In their paper, one of the drawbacks that the authors cite with their solution is the low detection rate against periodic jamming, as well as an unreliable classification of normal traffic. We decided to compare our solution against the solution of the authors of the dataset we used. The results show two significant Figures as the results provided in the compared paper were in this format. Results in the case of constant jamming were very similar, with both models scoring 1.0 in all the metrics and are therefore not shown.

Table 14 shows the comparison between the two models in the case of periodic jamming and Table 15 shows the comparison between the two models in classification of normal traffic.

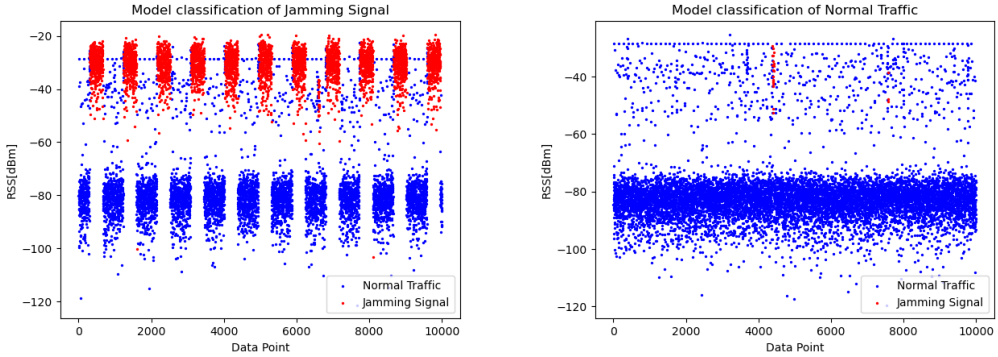
Table 14. Comparison between the proposed approach and the approach of [7] in the case of periodic jamming.

Metric	Proposed Approach	CNN Approach
Accuracy	0.98	0.72
Precision	0.99	0.73
Recall	0.95	0.83
F1 Score	0.97	0.78

Table 15. Comparison between the proposed approach and the approach of [7] in the case of normal traffic classification.

Metric	Proposed Approach	CNN Approach
Accuracy	1.00	0.87
Precision	1.00	0.80
Recall	1.00	0.70
F1 Score	1.00	0.75

In Figures 18a and 18b we can see the classification results of our proposed solution in the case of a testing sample size of 10,000 samples.



(a) Periodic Jamming classification. **(b)** Normal Traffic classification.

Figure 18. Classification results of the proposed solution. The model is able to correctly classify periodic jamming and normal traffic with high accuracy.

Our solution is able to achieve considerably better results in the classification of normal traffic and periodic jamming, while maintaining the same level of performance in the case of constant jamming.

6. Discussion

In this article we analyzed whether it was possible to employ an Isolation Forest model to detect jamming attacks in a 6G drone scenario. We tuned the hyperparameters of the model and tested it against the default IF model. We achieved a considerable improvement in performance thanks to the tuning process (Section 5.2.1), but despite the high recall and accuracy scores, the model experienced a high number of false positives, caused by the inherent noise of the dataset. To mitigate this issue, we proposed the integration of the IF model with a Majority Rule module (Section 5.2.2). The *window_size* parameter of the MR module was tuned to provide the best performance. The addition of the MR module proved extremely effective in reducing the number of false positives, greatly improving the model's precision.

Integration of the MR module also impacted total classification time as shown in Figure 17a, but the linear trend [20] that we expected from the IF model was preserved, meaning that the classification time complexity was kept at $O(n)$. In our opinion, the trade-off between the increased classification time and the improved performance is justified, as the model is now much more reliable in a real-world scenario, where noise from the surrounding environment is almost always present. The linear time complexity and low memory usage make the proposed solution suitable for a resource-constrained environment such as the analyzed 6G drone scenario.

As shown in Section 5.2.3, the proposed solution outperformed the solution proposed by [7] in both the detection of periodic jamming and the classification of normal traffic, while achieving the same performance against constant jamming attacks. We believe that the importance of correct classification of normal traffic is often underestimated, as a high number of false positives can lead to the activation of resource-intensive countermeasures, even at times when they are not needed.

7. Conclusions

In this article we analyzed the use of the Isolation Forest unsupervised machine learning algorithm for the detection of jamming attacks. We compared the performance of the base IF model against a version specifically tuned to detect jamming attacks. During the tuning phase, we paid particular attention to the computational impact of each parameter and tried to find the right balance between performance and efficiency. We then proposed the integration of a Majority Rule module to reduce the number of false positives caused by the inherent noise of the dataset. The MR module was tuned to provide the best performance and proved effective in mitigating the problem of false positives, while still

maintaining a linear complexity in classification time. We tested the model against a dataset that included normal traffic, constant jamming, and periodic jamming attacks, with an unbalanced number of normal traffic samples and jamming attack samples to better reflect a real-world scenario. We compared the performance of the proposed solution against the solution proposed by [7] and found that our solution outperformed the existing solution in both the detection of periodic jamming and the classification of normal traffic, while retaining the same performance against constant jamming attacks.

In a real world scenario, a 6G drone integrating the proposed solution would periodically run the tuning phase of the algorithm, selecting the best hyperparameters for the current environment. This could happen periodically or whenever there is a change in the environment caused by the drone moving to a different location. The dynamic tuning of the model would ensure that the parameters of the model are always optimized for the current environment, providing the best possible performance. After the tuning phase, the drone would periodically sample incoming signals and classify them using the MR-integrated IF. If the model detects a jamming attack, the drone could activate built-in countermeasures like Frequency Hopping or Direct Sequence Spread Spectrum to mitigate the effects of the attack.

Future research could explore the model's performance against a broader range of jamming attacks types, including smart, reactive and random jamming, and evaluate the impact of real world noise on its accuracy. Deploying the model on a drone would provide opportunity for real-world testing and valuable insights for further refinement.

As discussed in Section 3.5, devices integrated into 6G networks are expected to have a higher degree of sensing capabilities compared to current devices [2]. Thanks to the ability of the Isolation Forest to work on high dimensional data, the model could easily be expanded to include more classification features, which would improve its situational awareness and performance.

Author Contributions: Conceptualization, Francesco Chiti and Sergio Cibecchini; methodology, Sergio Cibecchini; software, Sergio Cibecchini; validation, Francesco Chiti and Laura Pierucci; writing—original draft preparation, Sergio Cibecchini; writing—review and editing, Francesco Chiti and Laura Pierucci; visualization, Sergio Cibecchini; supervision, Francesco Chiti and Laura Pierucci. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The dataset used in this study is openly available and provided by the authors of [7] at the following URL: https://github.com/AMHD/Jamming-Detection-in-IoT-Wireless-Networks-An-Edge-AI-Based-Approach. The code for the tests and the jamming detection model developed in this study is available in the following repository: https://github.com/cibecs/JammingAttacksAnomalyDetection.git.

Acknowledgments: This work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, in partnership with the project "Telecommunications of the Future" (PE00000001—program "RESTART"). Additionally, this article publication is based upon work from COST Action CA22168 for "Physical Layer Security for Trustworthy and Resilient 6G Systems (6G-PHYSEC)," supported by COST (European Cooperation in Science and Technology).

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

SDN	Software Defined Networks
UAVs	Unmanned Aerial Vehicles
3GPP	3rd Generation Partnership Project
ML	Machine Learning
V2X	Vehicle to Everything
FPV	First Person View
BS	Base Station
MR	Majority Rule
DOS	Denial Of Service
SNR	Signal to Noise Ratio
RSSI	Received Signal Strength Indicator
PDR	Packet Delivery Ratio
CIA	Confidentiality, Integrity, Availability
IoT	Internet of Things
AI	Artificial Intelligence
DSSS	Direct Sequence Spread Spectrum
FHSS	Frequency Hopping Spread Spectrum
OSI	Open Systems Interconnection
dBm	Decibel-milliwatts
IF	Isolation Forest
CNN	Convolutional Neural Network
RSV	Relative Speed Variation

References

1. 5G - statistics & facts. Available online: https://www.statista.com/topics/3447/5g/topicOverview (accessed on 13 September 2024).
2. Letaief, K.B.; Chen, W.; Shi, Y.; Zhang, J.; AI, B. The Roadmap to 6G: AI Empowered Wireless Networks. *IEEE Communications Magazine* **2019**, *57*, 84–90. https://doi.org/10.1109/MCOM.2019.1900271.
3. Lyamin, N.; Samuylov, A.; Gaidamaka, Y.; Vinel, A.; Koucheryavy, Y. AI-Based Malicious Network Traffic Detection in VANETs. *IEEE Network* **2018**, *32*, 15–21. https://doi.org/10.1109/MNET.2018.1800074.
4. Arjoune, Y.; Salahdine, F.; Islam, M.S.; Ghribi, E.; Kaabouch, N. A Novel Jamming Attacks Detection Approach Based on Machine Learning for Wireless Communication. In Proceedings of the 2020 International Conference on Information Networking (ICOIN), Barcelona, Spain, 2020; pp. 459–464. https://doi.org/10.1109/ICOIN48656.2020.9016462.
5. Karagiannis, D.; Argyriou, A. Jamming attack detection in a pair of RF communicating vehicles using unsupervised machine learning. *Vehicular Communications* **2018**, *13*, 56–63. https://doi.org/10.1016/j.vehcom.2018.05.001. Available online: https://www.sciencedirect.com/science/article/pii/S221420961730222X.
6. Greco, C.; Pace, P.; Basagni, S.; Fortino, G. Jamming detection at the edge of drone networks using Multi-layer Perceptrons and Decision Trees. *Applied Soft Computing* **2021**, *111*, 107806. https://doi.org/10.1016/j.asoc.2021.107806. Available online: https://www.sciencedirect.com/science/article/pii/S1568494621007274.
7. Hussain, A.; Zhang, M.; Bhatia, S.; Cheng, L. Jamming Detection in IoT Wireless Networks: An Edge-AI Based Approach. In Proceedings of the 12th International Conference on the Internet of Things, IoT '22; Association for Computing Machinery: Delft, Netherlands, 2023; pp. 57–64. ISBN: 9781450396653. https://doi.org/10.1145/3567445.3567456.
8. Zuo, S.; Liu, Y.; Zhang, D.; Xin, P.; Liu, T. Detection of GPS Spoofing Attacks Based on Isolation Forest. In Proceedings of the 2021 IEEE 9th International Conference on Information, Communication and Networks (ICICN), Xi'an, China, 2021; pp. 357–361. https://doi.org/10.1109/ICICN52636.2021.9673863.
9. Hong, S.; Kim, K.; Lee, S.-H. A Hybrid Jamming Detection Algorithm for Wireless Communications: Simultaneous Classification of Known Attacks and Detection of Unknown Attacks. *IEEE Communications Letters* **2023**, *27*, 1769–1773. https://doi.org/10.1109/LCOMM.2023.3275694.
10. Unmanned aircraft (drones). Available online: https://transport.ec.europa.eu/transport-modes/air/aviation-safety/unmanned-aircraft-drones_en (accessed on 13 September 2024).
11. Laricchia, F. Consumer and commercial drones - statistics and facts. Available online: https://www.statista.com/topics/7939/drones/#topicOverview (accessed on 13 September 2024).
12. Sultan, A. Study on enhancement of 3GPP support for 5G V2X services. Tech. Rep. *3GPP*, 2018.
13. Hassija, V.; Kumar, R.; Gupta, H.; Singh, S.; Sharma, P. Fast, Reliable, and Secure Drone Communication: A Comprehensive Survey. *IEEE Communications Surveys and Tutorials* **2021**, *23*, 2802–2832. https://doi.org/10.1109/COMST.2021.3097916.
14. Cawthra, J. Data Integrity: Detecting and Responding to Ransomware and Other Destructive Events. *NIST Special Publication* **2020**, *1800-26A*.
15. Boualouache, A.; Engel, T. A Survey on Machine Learning-Based Misbehavior Detection Systems for 5G and Beyond Vehicular Networks. *IEEE Communications Surveys and Tutorials* **2023**, *25*, 1128–1172. https://doi.org/10.1109/COMST.2023.3236448.

16. Feng, S.; Haykin, S. Cognitive Risk Control for Anti-Jamming V2V Communications in Autonomous Vehicle Networks. *IEEE Transactions on Vehicular Technology* **2019**, *68*, 9920–9934. https://doi.org/10.1109/TVT.2019.2935999.
17. Owano, N. RQ-170 drone's ambush facts spilled by Iranian engineer. Available online: https://phys.org/news/2011-12-rq-drone-ambush-facts-iranian.html (accessed on 14 September 2024).
18. Chorti, A.; Hollanti, C.; Koorapaty, H.; Poor, H.V. Context-Aware Security for 6G Wireless: The Role of Physical Layer Security. *IEEE Communications Standards Magazine* **2022**, *6*, 102–108. https://doi.org/10.1109/MCOMSTD.0001.2000082.
19. Sciancalepore, S.; Kusters, F.; Abdelhadi, N.K.; Oligeri, G. Jamming Detection in Low-BER Mobile Indoor Scenarios via Deep Learning. *arXiv* **2023**, eprint: 2306.10912. Available online: https://arxiv.org/abs/2306.10912.
20. Liu, F.T.; Ting, K.M.; Zhou, Z.-H. Isolation Forest. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, Pisa, Italy, 2008; pp. 413–422. https://doi.org/10.1109/ICDM.2008.17.
21. Scikit-learn: Isolation Forest. Available online: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html (accessed on 14 September 2024).