Article

AI-Based Detection of Jamming Attacks In 6G Drone Networks

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

[1] Affiliation 1; e-mail@e-mail.com
[2] Affiliation 2; e-mail@e-mail.com
* Correspondence: sergio.cibecchini@gmail.com; Tel.: (optional; include country code; if there are multiple corresponding authors, add author initials) +xx-xxxx-xxx-xxxx (F.L.)
† Current address: Affiliation.
‡ These authors contributed equally to this work.

Keywords: keyword 1; keyword 2; keyword 3 (List three to ten pertinent keywords specific to the article; yet reasonably common within the subject discipline.)

1. Introduction

The shift from 4G to 5G has been a generational leap has revolutionized connectivity around the world. Despite 5G being still in its early adoption rate 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 the increases in both data rates and latency but this in turn will also bring forth new security challenges, specific to those applications.

One of the main differences between 5G and 6G technology will be the focus of the 6G standard in regards to AI integration. 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, by integrating artificial intelligence and machine learning directly into the network. 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 security threats. By leveraging Diagnostic Analytics, a collection of insights into the status of the networks, security teams will be able to train specific AI models to detect and respond to security threats in real-time.

In this paper we will focus on a specific aspect of the security of 6G networks, namely we will provide a machine learning based approach for the detection of jamming attacks in networks of drones.

The paper is divided in 5 sections:

1. **Topic overview:** In this section we will discuss how drones can benefit from integration in 6G network, analyze jamming attacks, their types, mitigation and detection. Finally we will discuss the advantages of an edge AI approach for jamming detection.
2. **Materials and methods:** In this section we will define the scenario that we decided to analyze as well as the dataset, algorithm and evaluation metrics we chose for our tests.
3. **Numerical Results:** In this section we will present the testing methodology as well as the numerical results that we obtained.
4. **Discussion:** In this section we will discuss the results obtained in the previous section and highlight notable trends.
5. **Conclusions:** In this section we will summarize the results and discuss possible future research directions.

Citation: Cibecchini, S.; Chiti, F.; Lastname, F. AI-Based Detection of Jamming Attacks in 6G Drone Networks. *Future Internet* **2024**, *1*, 0. https://doi.org/

Received:
Revised:
Accepted:
Published:

2. Related Works

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

3. Topic Overview

3.1. Drones in 6G Networks

Drones, also known as Unmanned Aerial Vehicles (UAVs) are defined as *all aircraft designed to fly without a pilot on board* [3]. This technology has experienced rapid growth in recent years and is expected to keep growing in both the consumer sector as well as the commercial and military sectors [4].

In report 22.886 [5] 3GPP, the organ responsible for the development and maintenance of the 5G standard, identifies some of the envisioned use cases for 5G V2X (Vehicle-to-Everything) communication services. Among these use cases, the report identifies vehicle platooning, advanced and remote driving end extended situational awareness as some of the main benefits of V2V communication. All these use cases 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 soil and crop health assessment as well as irrigation in agriculture, delivery of life saving supplies and locating of survivors in disaster response, in the delivery sector as a more eco friendly alternative to traditional delivery options and possibly in the mobility sector as a complement to traditional taxis and public transportation [6].

The effectiveness of drones in the military sector is widely recognized, both for high and low end models. For example, in the Ukraine war even cheap FPV drones mounted with explosives were successfully used to destroy much more expensive equipment [7].

Security 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 communication between two or more devices. This is achieved by transmitting a powerful signal on the same frequency as the one used by the devices to communicate. If the jamming signal is strong enough, it is able to overwhelm the legitimate signal, effectively blocking the communication between the devices [6]. 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 classification [8]. 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 as the devices it wants to disrupt.
- **Periodic Jamming:** The jammer simply transmits a strong signal on the same frequency as the devices it wants to disrupt for a certain period of time t_a, then stops transmitting for another period of time t_b. This type of jamming attack is particularly effective against devices that are not able to change their frequency and has the ability to disrupt the transmission of a sequence of consecutive packets. [9]
- **Random Jamming:** In random jamming, the jammer is active at random intervals, giving each transmitted packet a probability p of being jammed [9].
- **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 operation types, as it only transmits its signal when it is needed[16].
- **Smart Jamming:** A smart jammer is a more sophisticated type of jammer that is able to adapt its jamming signal to maximize the disruption of the communication between the devices. Smart jammers are able to modify their attack pattern based on the transmission specifics of the devices they are targeting and are able to adapt to changes in the communication channel[11].

An effective Jamming detection AI model should achieve a high detection score against all the different types of jamming attacks.

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, 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 landing procedure or a hover in place procedure. All of these procedures 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.

One real world example of this is the capture of an American drone by Iran in 2011. In December 2011 a Lockheed Martin RQ-170 Sentinel drone operated by the United States Air Force was flying over Iran when its operators lost control of the vehicle. The US government initially claimed that the drone had crashed due to a technical malfunction, but later reports revealed that the drone had been captured by the Iranian military. Iranian electronic warfare specialists claimed to have brought it down using a Jamming attack, that forced the drone into a return to base procedure, in combination with a GPS spoofing attack, that made the drone land into a designated area [12]. After successfully capturing the drone, the Iranian government managed to reverse-engineer the drone and produce a working replica, which was then used in their military operations [13].

3.4. Centralized vs Edge approach for Jamming detection

When presenting a machine learning 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 but also in terms of internal storage and battery capacity [14]. 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 central server, that returns the results of the AI model to the device. While this approach might be feasible in some cases, in a real-time situation such as a jamming attack, the latency introduced by the communication with the central server means less time to react to the attack. Also, as jamming attacks degrade or sometimes completely block the communication between the device and the base station, a centralized approach might not be feasible in this case. An lightweight edge approach on the other hand, while not as powerful as a centralized approach, would to provide real-time results to the device and would be able to operate even when the communication is disrupted.

3.5. Detection and Mitigation of Jamming Attacks

When attempting to mitigate jamming attacks, the first step is always to detect that an attack is taking place. This can be done by monitoring the communication channel for signs of jamming. The most common approaches involve the analysis of metrics such as the Signal to Noise Ration (SNR), the Received Signal Strength Indicator (RSSI) and the Packet Delivery Ratio (PDR) [15]. The most simple way 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 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 machine learning 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 [9]. 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.

4. Materials and Methods

4.1. Scenario Definition and Attacker Classification

In the proposed scenario 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 algorithm module, trained on nominal traffic RSS values. The module takes the sampled RSS values and classifies them as either normal or anomalous. Thanks to this classification module, the drone is able to determine wether 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 point in time by transmitting a strong 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 the drone has to rely on its internal classification algorithm 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 shows a diagram of the proposed scenario.

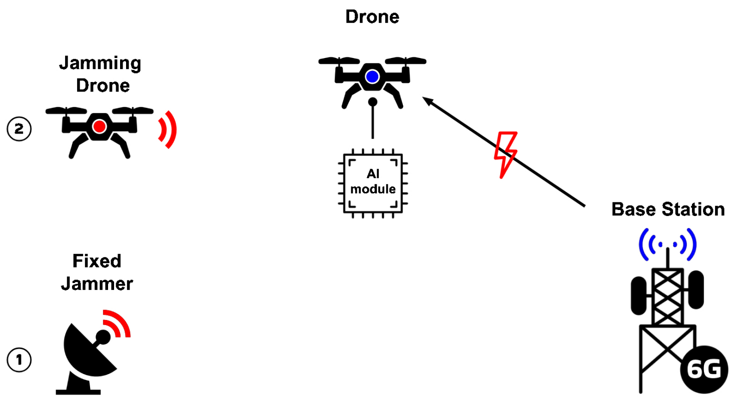


Figure 1. Proposed jamming scenario. The Drone communicates with the 6G base station and receives information about how to behave. The fixed jammer **(1)** and the mobile jammer **(2)** target the communication between the drone and the base station. The drone has an internal classification module that is able to detect jamming attacks.

In our scenario, the drone will be subject to two types of jamming attacks, a constant jamming attack and a periodic jamming attack. These types of attacks can be attributed to different types of jammers. The constant jamming attack could be caused by a ground jammer, as maintaining a constant jamming signal requires a large power source, while the periodic jamming attack could be caused by a drone jammer, as drones are usually battery powered and can only jam for a limited amount of time. Employing a periodic jamming attack could help the malicious drone conserve energy and reduce the chances of being detected.

Table 1 shows the attacker classification in the proposed scenario [16]:

Table 1. 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 available, we chose an open-source dataset [17] that analyses the RSS values received by a Raspberry Pi 3 that is subject to periodic and constant jamming attacks.

The dataset 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 chosen to receive the jamming signal and the RSS values and was connected to a Raspberry Pi 3. The jamming radio operates at a frequency of 2.412 GHz 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 3 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 sample. The samples us the *dBm (decibel-milliwatts)* unit of measurement a logarithmic scale that measures the power of a signal in relation to 1 milliwatt. Conversion can be done using the formula [18] shown in equation 1:

$$P_{dBm} = 10 \cdot \log_{10}\left(\frac{P_{mW}}{1mW}\right) \tag{1}$$

Figures 2 and 3 show respectively plots of the RSS values sampled during a constant jamming attack and a periodic jamming attack.

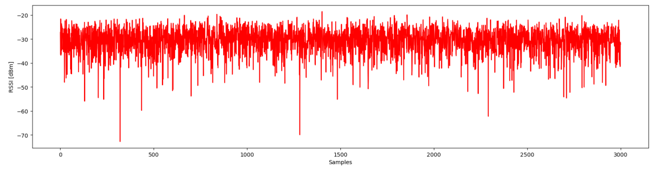


Figure 2. Constant jamming attack RSS values

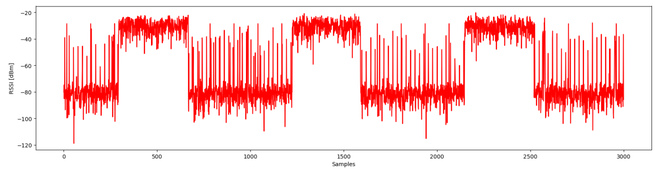


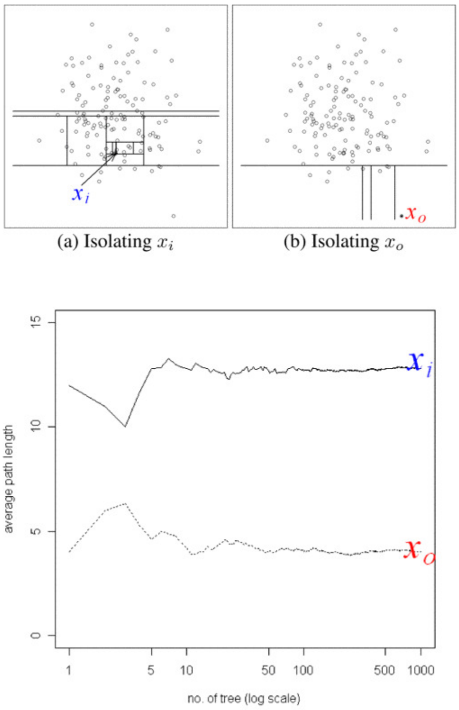
Figure 3. Periodic jamming attack RSS values

4.3. Algorithm Choice

When choosing the algorithm for the classification module, we first had to define the requirements that the algorithm had to meet. The algorithm has to be able to classify anomalous samples with a high degree of accuracy, while at the same time being lightweight enough 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 redetermine the normal transmission RSS values in a constantly changing environment. If the model didn't periodically redefine normal RSS values, it would be prone to false positives caused by the mobile nature of drones. The model should be also required a limited amount of storage to be run effectively, as drones are usually lacking in internal storage.

The algorithm that best fit these requirements was the *Isolation Forest* algorithm [19]. The Isolation Forest algorithm is a state-of-the-art unsupervised machine learning algorithm for anomaly detection, introduced by Liu et al. in 2008. Unlike traditional anomaly detection algorithms, which require the definition of a normal class, Isolation Forest is able to detect anomalies without explicitly defining the characteristics of the normal class.

This is achieved by leveraging the properties of anomalies themselves, i.e being few 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 that were required to isolate is used to determine the anomaly score of the data point. The higher the number of splits, the more likely the data point is to be an anomaly. Isolation Forest is an *ensamble* algorithm, meaning that the final anomaly score is calculated by averaging the anomaly scores of all the trees in the forest. More trees mean more accuracy but also more computational power required.



(a) Isolating x_i (b) Isolating x_o

(c) Average path lengths converge

Figure 4. Isolation forest in action. The anomaly **x0** is isolated in less splits compared to the normal datapoint **x1**[19].

The chosen implementation for isolation forest is the one provided by the *Scikit-learn* library [20] for Python 3. The algorithm was tested on a Desktop PC with an Intel i7-6700 CPU and 16GB of RAM as well as on a Raspberry Pi 3 with a Quad Core 1.2GHz CPU and 1GB of RAM [21]. The graphs shown are the ones obtained on the Desktop PC for convenience, but the ability of the algorithm to run effectively on a resource constrained device like the Raspberry Pi 3, that has hardware comparable to the one found in high-end commercial drones, was verified.

4.4. Evaluation Metrics

The performance of the model was evaluated using the state-of-the-art evaluation metrics for machine learning classification algorithms. The metrics are all based on the number of true positives, false positives, true negatives and false negatives. A true positive is a data point that is correctly classified as an anomaly, while a false positive is a data point that is incorrectly classified as an anomaly. A true negative on the other hand is a data point that is correctly classified as a normal data point, while a false negative is a data point that is incorrectly classified as a normal data point. The evaluation metrics that we chose to use are the following:

- **Accuracy:** The accuracy is the ratio of correctly classified data points to the total number of data points 2.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

- **Precision:** The precision is the ratio of correctly classified anomalies to the total number of data points classified as anomalies 3.

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

- **Recall:** The recall is the ratio of correctly classified anomalies to the total number of anomalies 4.

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

- **F1 Score:** The F1 score is the harmonic mean of the precision and recall 5.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5}$$

The combination of these metrics gives us an insight into the model's performance.

5. Results

5.1. Parameters Tuning Phase

The first phase in our testing methodology was the tuning of the hyperparameters of the model. The tuning was performed by making the parameters vary between a minimum and a maximum value, evaluating the impact on the performance metrics (see section 4.4) and then choosing the best performing parameters. From the parameters available in the *Scikit-learn* implementation of the Isolation Forest, the parameters we decided to tune were the following:

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

During the tuning phase, the untested hyperparameters were set to their default values of the *Scikit-learn* implementation of the Isolation Forest (2).

Table 2. Scikit-Learn Isolation Forest hyperparameters default values.

Parameter	Default Value
n_estimators	100
max_samples	'auto'
contamination	0.1

During he tuning process, as well as the subsequent model testing phase, the model was evaluated using as input 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 classify correctly anomalous samples but also reduce the number of false positives during normal operation.

The values whe chose for *normal_traffic_size* and *jamming_size* are shown in table 3.

Algorithm 1 Test input definition

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

Table 3. Dataset sizes used for the tuning and testing phases.

Dataset	Size
jamming samples n.	2000
nominal samples n.	20000

The dataset is purposefully unbalanced, 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 the figure which employs constant jamming. Periodic jamming always showed comparable trends as constant jamming in all the tests performed.

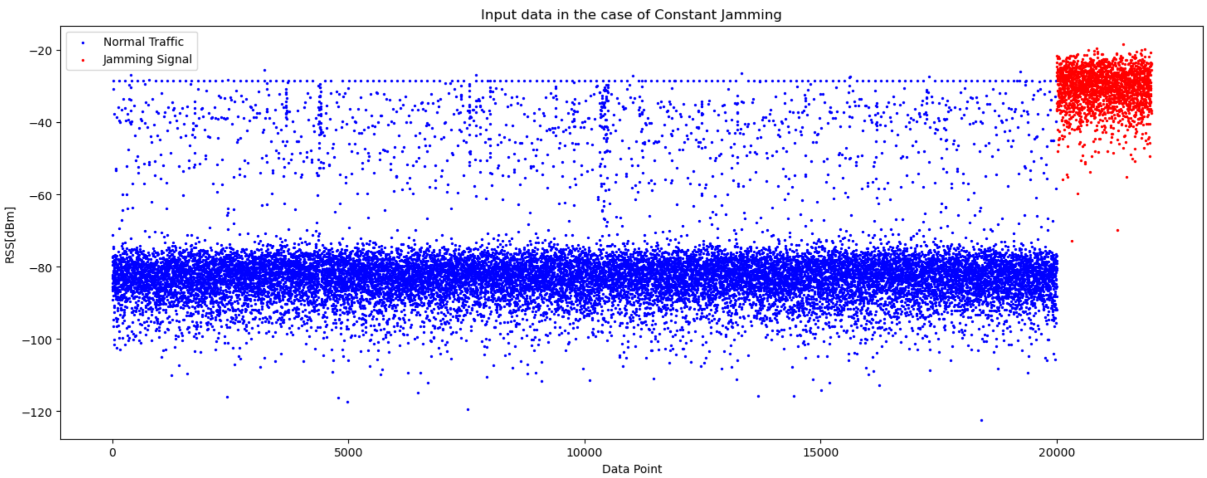


Figure 5. Input signal in the case of Constant Jamming. The normal traffic signal is concatenated with the jamming signal.

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 4.

Table 4. 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.

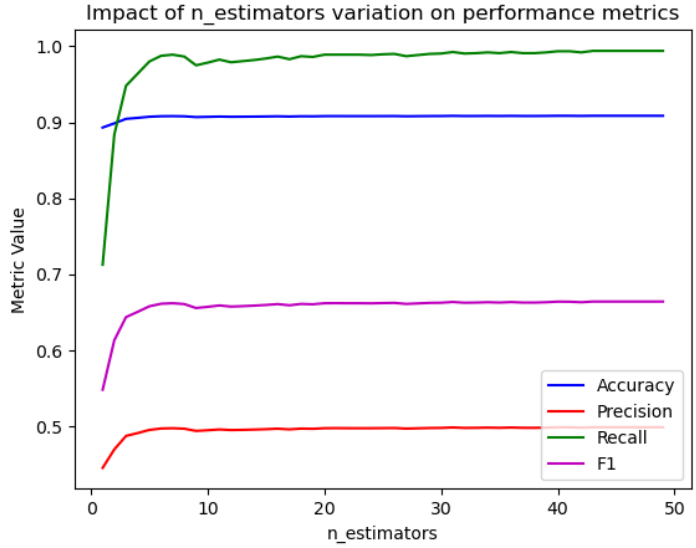
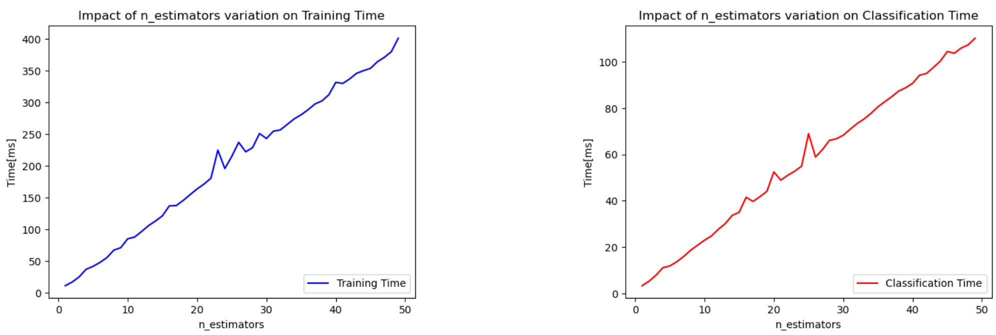


Figure 6. n_estimators tuning results.

As we can see from the graph, a growing number of estimators initially correlates with a better model performance, but after a certain point the performance stabilizes. In graphs 7a and 7b is shown 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.

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

5.1.2. max_samples tuning

The second parameter that we decided to tune was the *max_samples* parameter. As shown in table 5, the tuning values were chosen to be between 1 and 20000, i.e the same number of normal traffic samples in the test input.

Table 5. max_samples tuning values.

max_samples	Values
Minimum	1
Maximum	100
Step	1

From figure 8 we can see that the model performance is not greatly affected by the *max_samples* parameter. This is most likely due to the great difference in RSS values between the normal traffic samples and the jamming attack samples, meaning the model can develop the capability to correctly classify the input even with a small training set. 323 324 325 326

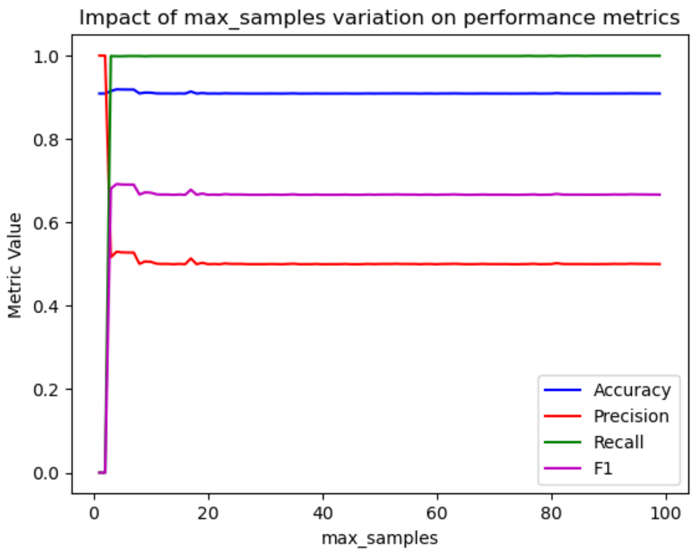
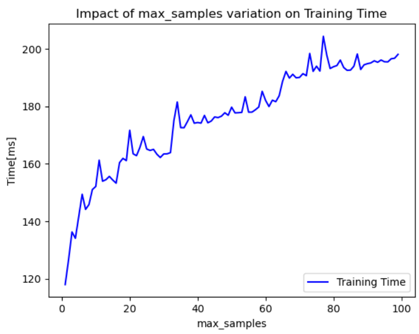
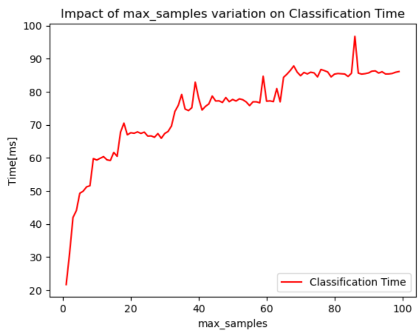


Figure 8. max_samples tuning results.

Still, as the model has to build bigger trees, the *max_samples* parameters as considerable influence on the training and classification time, as shown in figure 9. Given minimal performance differences but the impact on the time metrics, chose the value of **max_samples = 10**. 327 328 329 330



(a) max_samples training time.



(b) max_samples classification time.

Figure 9. Comparison of max_samples training and classification times.

5.1.3. contamination tuning 331

The final parameter we focused on is the *contamination* parameter. The tuning values are shown in table 6. 332 333

Table 6. contamination tuning values.

contamination	Values
Minimum	0.01
Maximum	0.5
Step	0.01

The contamination parameter was by far the one that had the most impact on the model performance, as shown in figure **??**. As recall was the metric that we wanted to prioritize as highlights the number of detections, we chose the value of **contamination = 0.09**, as it provided the best metric values with our dataset.

Our testing showed that, as expected, tuning of the contamination parameter had a 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, recapped in table 7 and compared against the default parameters.

Table 7. 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 10 we can see the results of the model classification of the input signal with the tuned hyperparameters.

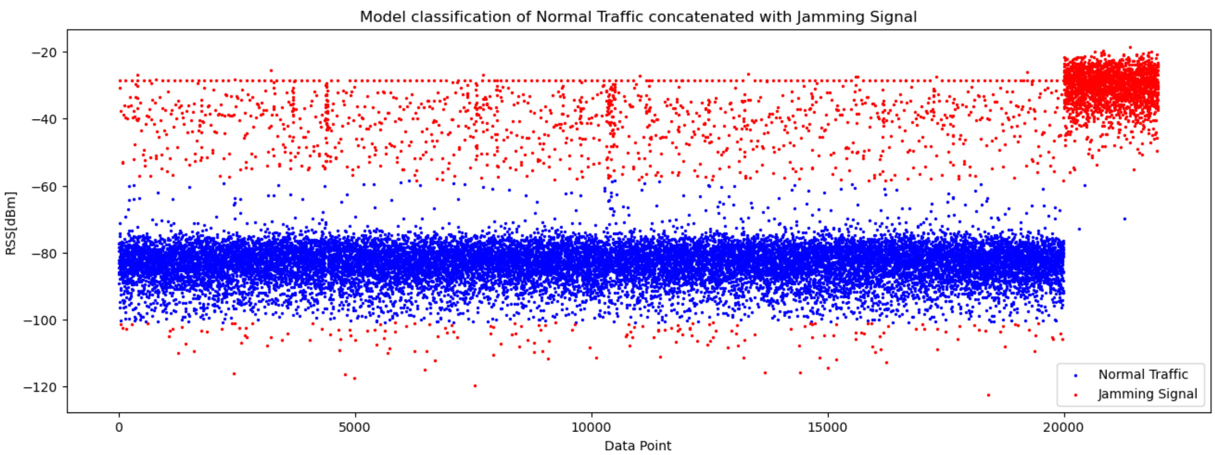


Figure 10. Classification results from the tuned Isolation Forest model

Table 8 and 9 show a comparison between the model performance with the tuned hyperparameters and the default hyperparameters.

Table 8. Confusion Matrix Components Comparison.

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

Table 9. 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. Still, 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 10, many of the normal traffic samples have RSS values that are similar to the ones of the jamming attack samples, leading the model to falsely classify them as anomalies. This is most likely because of how the dataset was created, as the normal traffic samples were taken from a real-world scenario where the RSS values and be effected by external conditions.

Instead of seeing this as a setback, we see this as a positive. The fact that the dataset contains noise means it is more representative of a real-world scenario, where the drone might be moving trough 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 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 Majority Rule module, as shown in figure 11.

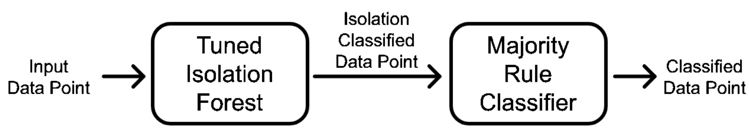


Figure 11. Majority Rule system diagram. The Tuned Isolation Forest model results are passed to the Majority Rule module, that then provides the final classification.

The mechanism of the Majority Rule module is simple. When a data point is passed to the Majority rule module, its classification class is inserted into a sliding window. If the datapoint is classified by the Tuned Isolation forest as an anomaly, the Majority Rule module checks the contents of the sliding window. If the majority of the datapoints in the window are classified as anomalies, the data point is classified as an anomaly. If the majority of the datapoints in the window are classified as normal, the data point is classified as normal.

To determine the optimal value for the *window_size* parameter, we tested the model in the same way as we did for the hyperparameters tuning phase. This time, the model

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
```

parameters were set to the best performing values from the tuning phase (table 7). Table 10 shows the tuning values for the *window_size* parameter.

Table 10. window_size tuning values.

window_size	Values
Minimum	1
Maximum	100
Step	1

From figure 12 we can see that the *window_size* parameter has a considerable impact on the model performance. The optimal value for the *window_size* parameter was found to be **window_size = 39**.

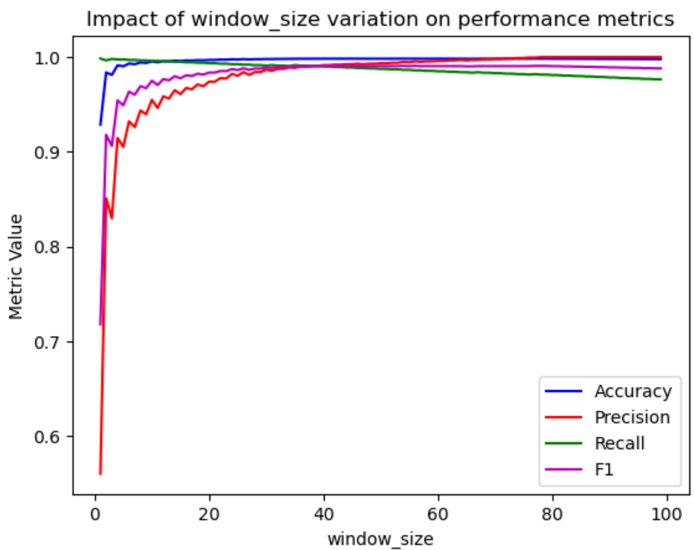


Figure 12. window_size tuning results.

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 13.

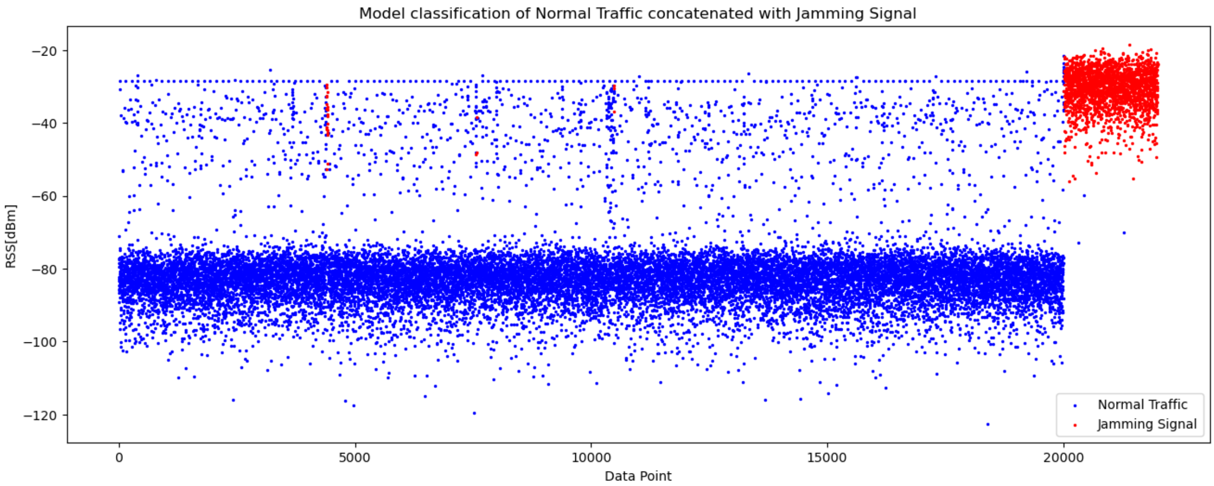


Figure 13. Classification results from the Majority Rule Isolation Forest model

In table 11 and 12 we can see a comparison between the tuned model and the Majority Rule model.

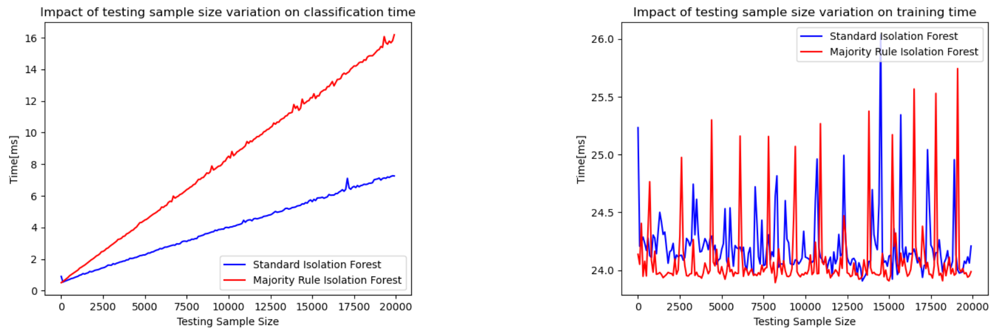
Table 11. Confusion Matrix Components Comparison.

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

Table 12. 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 proved effective in mitigating the high number of false positives that was present in the first version of the model. Given the resource constrained nature of the scenario, we decided to test the computational impact of the Majority Rule module against the standard tuned model. Classification times and training times were measured for both models and the results are shown in figure pair 14.



(a) Classification time comparison. **(b)** Training time comparison.

Figure 14. Comparison of the standard tuned model and the Majority Rule model training and classification times.

Graph 14a shows that there is indeed a difference in classification time, with the standard model being faster. Figure 14b on the other hand shows that, as expected, training time is not affected by the Majority Rule module.

5.2.3. Comparison against existing solution

As mentioned in section 4.2, this work is based on the work of [17], 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 a non reliable classification of normal traffic. We decided to compare our solution against the solution of the authors of the dataset we used. We decided to use two significant figures as the results provided in the compared paper were in this format. The results in the case of constant jamming were very similar, with both models scoring 1.0 in all the metrics. Table 13 shows the comparison between the two models in the case of periodic jamming and table 14 shows the comparison between the two models in classification of normal traffic.

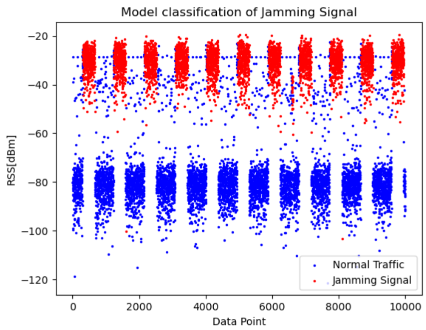
Table 13. Comparison between the proposed approach and the approach of [17] in the case of periodic jamming.

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

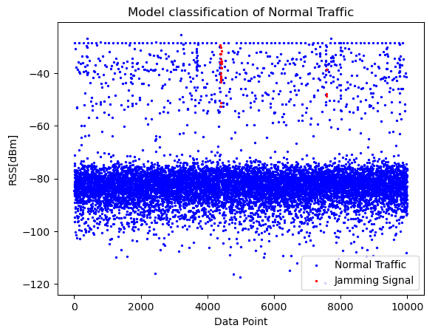
Table 14. Comparison between the proposed approach and the approach of [17] in the case of normal traffic classification.

Metric	Proposed Approach	CNN Approach
Accuracy	1.00	N.D
Precision	1.00	0.80
Recall	1.00	0.70
F1 Score	1.00	0.75

In figures 15a and 15b we can see the classification results of our proposed solution in the case of a testing sample size of 10000 samples.



(a) Periodic Jamming classification.



(b) Normal Traffic classification.

Figure 15. Classification results of the proposed solution.

6. Discussion

In this article we analyzed wether 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 Isolation Forest 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 suffered too much from a high number of false positives, caused by the inherent noise of the dataset. To mitigate this issue, we proposed the integration of the Isolation Forest model with a Majority Rule module (section 5.2.2). The *window_size* parameter of the Majority Rule module was tuned to provide the best performance. the integration of the Majority Rule module proved extremely effective in reducing the number of false positives, greatly improving the model's precision.

The integration of the Majority Rule module impacted total classification time as shown in figure 14a, but the linear trend [19] that we expected from the isolation forest 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 worth it, 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 of the classification means that the algorithm is suitable for a resource constrained environment such as the proposed 6G drone scenario.

As shown in section 5.2.3, the proposed solution outperformed the solution proposed by [17] in both the detection of periodic jamming attacks and the classification of normal traffic, while matching the performance against constant jamming. 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

7.1. Real world integration

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. After tuning of the model, the drone would periodically sample incoming signals and classify them using the Majority Rule module. If the model detects a jamming attack, the drone could activate built in countermeasures like frequency hopping or DSSS to mitigate the effects of the attack.

7.2. Future Work

Future research could test the model against a wider range of jamming attacks types, as well as analyze the impact of real world noise on the model performance. Mounting the model on an actual drone would be the best way to achieve a real-world test of the model and would provide valuable insights into how to improve the model further.

Devices integrated into 6G networks are expected to have a higher degree of sensing capabilities compared to current devices [2]. Future research could leverage these additional metrics to improve the model's performance. Thanks to the properties of the Isolation Forest to work on high dimensional data, the model could be easily expanded to include more metrics, such as the RSV metric proposed in [17].

8. Patents

This section is not mandatory, but may be added if there are patents resulting from the work reported in this manuscript.

Author Contributions: For research articles with several authors, a short paragraph specifying their individual contributions must be provided. The following statements should be used "Conceptualization, X.X. and Y.Y.; methodology, X.X.; software, X.X.; validation, X.X., Y.Y. and Z.Z.; formal analysis, X.X.; investigation, X.X.; resources, X.X.; data curation, X.X.; writing—original draft preparation, X.X.; writing—review and editing, X.X.; visualization, X.X.; supervision, X.X.; project administration, X.X.; funding acquisition, Y.Y. All authors have read and agreed to the published version of the manuscript.", please turn to the CRediT taxonomy for the term explanation. Authorship must be limited to those who have contributed substantially to the work reported.

Funding: Please add: "This research received no external funding" or "This research was funded by NAME OF FUNDER grant number XXX." and and "The APC was funded by XXX". Check carefully that the details given are accurate and use the standard spelling of funding agency names at https://search.crossref.org/funding, any errors may affect your future funding.

Institutional Review Board Statement: In this section, you should add the Institutional Review Board Statement and approval number, if relevant to your study. You might choose to exclude this statement if the study did not require ethical approval. Please note that the Editorial Office might ask you for further information. Please add "The study was conducted in accordance with the Declaration of Helsinki, and approved by the Institutional Review Board (or Ethics Committee) of NAME OF INSTITUTE (protocol code XXX and date of approval)." for studies involving humans. OR "The animal study protocol was approved by the Institutional Review Board (or Ethics Committee) of NAME OF INSTITUTE (protocol code XXX and date of approval)." for studies involving animals. OR "Ethical review and approval were waived for this study due to REASON (please provide a detailed justification)." OR "Not applicable" for studies not involving humans or animals.

Informed Consent Statement: Any research article describing a study involving humans should contain this statement. Please add "Informed consent was obtained from all subjects involved in the study." OR "Patient consent was waived due to REASON (please provide a detailed justification)." OR "Not applicable" for studies not involving humans. You might also choose to exclude this statement if the study did not involve humans.

Written informed consent for publication must be obtained from participating patients who can be identified (including by the patients themselves). Please state "Written informed consent has been obtained from the patient(s) to publish this paper" if applicable.

Data Availability Statement: We encourage all authors of articles published in MDPI journals to share their research data. In this section, please provide details regarding where data supporting reported results can be found, including links to publicly archived datasets analyzed or generated during the study. Where no new data were created, or where data is unavailable due to privacy or ethical restrictions, a statement is still required. Suggested Data Availability Statements are available in section "MDPI Research Data Policies" at https://www.mdpi.com/ethics.

Acknowledgments: In this section you can acknowledge any support given which is not covered by the author contribution or funding sections. This may include administrative and technical support, or donations in kind (e.g., materials used for experiments).

Conflicts of Interest: Declare conflicts of interest or state "The authors declare no conflicts of interest." Authors must identify and declare any personal circumstances or interest that may be perceived as inappropriately influencing the representation or interpretation of reported research results. Any role of the funders in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript; or in the decision to publish the results must be declared in this section. If there is no role, please state "The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results".

Abbreviations

The following abbreviations are used in this manuscript:

SDN	Software Defined Networks
UAVs	Unmanned Aerial Vehicles
3GPP	3rd Generation Partnership Project
V2X	Vehicle to Everything
FPV	First Person View
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
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. Unmanned aircraft (drones). Available online: https://transport.ec.europa.eu/transport-modes/air/aviation-safety/unmanned-aircraft-drones_en (accessed on 13 September 2024).
4. Laricchia, F. Consumer and commercial drones - statistics and facts. Available online: https://www.statista.com/topics/7939/drones/#topicOverview (accessed on 13 September 2024).
5. Sultan, A. Study on enhancement of 3GPP support for 5G V2X services. Tech. Rep. *3GPP*, 2018.
6. 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.
7. Rao, A.; Zafra, M.; Hunder, M.; Kiyada, S. How drone combat in Ukraine is changing warfare. Available online: https://www.reuters.com/graphics/UKRAINE-CRISIS/DRONES/dwpkeyjwkpm/ (accessed on 13 September 2024).
8. Cawthra, J. Data Integrity: Detecting and Responding to Ransomware and Other Destructive Events. *NIST Special Publication* **2020**, *1800-26A*.
9. 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.

16. 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.

11. 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.
12. 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).
13. Gross, J.A.; TOI Staff. Iranian UAV that entered Israeli airspace seems to be American stealth knock-off. Available online: https://www.timesofisrael.com/iranian-uav-that-enteredisraeli-airspace-seems-to-be-american-stealth-knock-off (accessed on 16 May 2024).
14. 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.
15. 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.
16. 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.

17. 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.
18. Sobot, R. *Wireless Communication Electronics: Introduction to RF Circuits and Design*; Springer: 2012; p. 252. ISBN 9783030486303.
19. 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.
20. Scikit-learn: Isolation Forest. Available online: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html (accessed on 14 September 2024).
21. Raspberry Pi 3 Model B. Available online: https://www.raspberrypi.com/products/raspberry-pi-3-model-b/ (accessed on 14 September 2024).
22. 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.