Applied Soft Computing 111 (2021) 107806

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

# Jamming detection at the edge of drone networks using Multi-layer Perceptrons and Decision Trees

Claudia Greco [a], Pasquale Pace [a,*], Stefano Basagni [b], Giancarlo Fortino [a]

[a] DIMES-University of Calabria, Rende 87036, Italy
[b] Institute for the Wireless Internet of Things, Northeastern University, Boston, MA 02115, United States

ARTICLE INFO

*Article history:*
Received 31 January 2021
Received in revised form 19 July 2021
Accepted 9 August 2021
Available online 13 August 2021

*Keywords:*
Machine learning
Jamming detection
Drone networks

ABSTRACT

As wireless networks play an increasingly key role in everyday life, it is necessary to secure them from radio frequency attacks, such as jamming, which are hard to detect, especially because they may be easily mistaken for other network conditions. Within this challenging context, the paper proposes a framework for jamming detection in drone networks, relying on a distributed approach based on supervised machine learning techniques, namely, Multi-layer Perceptrons and Decision Trees. Given a reference data packet trace set, our framework computes the features of some predefined metrics, such as throughput, PDR and RSSI, which vary during a jamming attack, and that can therefore be used to detect it. We evaluate our framework using datasets from publicly available standardized jamming attack scenarios with IEEE 802.11p radio data, and via ns3-based simulation datasets from networks of drones using WiFi. We show that the performance of the classifiers improves as the sampling time of the packets decreases. We also show that the Multi-layer Perceptron can be effectively generalized to achieve jamming detection accuracy superior to that of Decision Trees even when applied to communication scenarios for which it has not been specifically trained. Our proposed framework reaches a satisfactory accuracy level of 96%, while requiring low computational and hardware capabilities, thus proving to be suitable for resource-constrained drone networks.

## 1. Introduction

The rapid evolution of the computational and sensory capabilities of increasingly small devices has made the use of networks of mobile small nodes, like Unmanned Aerial Vehicles (UAVs) or *drones*, widespread and reasonably inexpensive. As flying devices without humans on board drone networks are ideal for supporting strategic missions in which they are used for surveillance and intelligence operations and, if armed, also for offensive missions. In recent years drone networks have been also used to other areas of interest, spanning from civilian to commercial domains, such as remote sensing, reconnaissance, search and rescue, commercial aerial surveillance, film-making, disaster relief, relay communication, etc. [1–3].

Given the increasing deployment of these networks, it is critical to make them as secure as possible. There are numerous threats that undermine the security of drone networks and, more generally, of wireless networks. Some of these threats can be addressed through appropriately designed network architectures and routing protocols [3–7]. Existing security mechanisms applied to drone networks are based on cryptography to ensure message privacy and node authentication. However, there are some attacks that, by their nature, cannot be addressed by conventional security mechanisms and therefore need further defense. Radio interference attacks, or jamming attacks, are a typical example of these threats.

Drone networks, often called FANETs (for Flying Ad hoc Networks), are particularly subject to jamming, as drone in the network communicate using wireless radio links, which are subject to interference. Jamming is considered the wireless version of a Denial-of-Service (DoS) attack [8]: the attacker may damage the operations of nodes within its radio range, preventing other devices from communicating correctly. Although it is a well-known problem in wireless networks, jamming is still an open issue and to date there is no effective solution to this threat.

### 1.1. Research context and motivations

Many solutions have been proposed for reacting to jamming attacks, such as frequency hopping [9,10] and channel hopping [11]. Clearly, in order to react to a jamming attack, we must detect it first. Unfortunately, this is not an easy task since there is no single network parameter which, by itself, allows us to recognize a jammed condition. Also, jamming makes it impossible for the network nodes to communicate with each other. However,

* Corresponding author.
*E-mail address:* p.pace@dimes.unical.it (P. Pace).

https://doi.org/10.1016/j.asoc.2021.107806
1568-4946/© 2021 Elsevier B.V. All rights reserved.

this can also happen due to other network conditions, such as congestion or non-malicious interference [12]. Therefore, it is necessary to combine more than one metric to ascertain whether impaired communication is due to jamming or not.

Most studies carried out on jamming concern general wireless scenarios. Works focusing on jamming attacks on drone networks are lacking, arguably because detecting jamming in these networks is made even more challenging because of the nature of drones themselves, namely, small devices, with limited computational resources and power consumption constraints.

To tackle this challenge, on-device Artificial Intelligence (AI) can be applied, due to its communication efficiency. However, current solutions relying on AI techniques follow a centralized approach [13–15]. This kind of solutions is not appropriate for drone networks because of their highly mobile and decentralized nature, their highly mobile nodes, low response times and power constraints.

### 1.2. Objectives and contributions

Our purpose in this work is to present a new framework for detecting jamming attacks in drone networks. We propose a distributed supervised learning-based on-device jamming attack detection security framework that relies on an analysis of network parameters whose combination can be a symptom of jamming. Particularly, we identify throughput, Packet Delivery Rate (PDR) and Received Signal Strength Indicators (RSSI) as metrics that may noticeably vary when a jamming attack occurs, and whose combination can be therefore used to detect jamming.

We choose these features as attributes of the datasets used to induce our classification models, according to supervised machine learning approaches.

The framework operates following two main steps:

1. given a set of packet traces, it extract a series of default features;
2. once a value is assigned to each feature, it detects whether jamming is occurring or not.

We propose a distributed solution, in which each node is responsible to detect its own jamming condition without relying on any other network entity to discover if an attacker is jamming it. The detection is performed through two machine learning (ML) techniques which are *Multi-layer Perceptron* and *Decision Tree* whose operations are detailed in Section 3.3.

We evaluate our framework with datasets from publicly available standardized jamming attack scenarios by CRAWDAD [16], and analyze its applicability to drone networks also evaluating our framework with datasets generated through NS3 simulations. Then, we analyze the obtained results, expressed in terms of accuracy values, which result in the capability to detect known attacks and more. In particular, we compare the two ML techniques in various network communication settings verifying that the Multi-layer Perceptron has a higher generalization capability overcoming the Decision Tree (96% Vs 50%) in larger and more complex communication scenarios for which had not been trained. Since our goal is not only to provide a framework that is capable of jamming detection, but also to be used in the context of drone networks, in the end we perform an analysis of the required hardware resources, bearing in mind the limited resources with which drones are generally equipped.

The rest of the paper is structured as follows. Section 2 introduces background concepts about drone networks and jamming attacks, and the main techniques known in literature to counteract it. In Section 3 we present our framework and its functionalities, highlighting the characteristics of the metrics on which it is based. We also provide some introductory concepts on the machine learning techniques used in the paper. Results from the performance evaluation of our framework is reported in Section 4, comparing the performance obtained by using different communication technologies and networks. Section 5 concludes the paper and discusses future work directions.

## 2. Background and related works

This section presents the issues related to the emerging drone networks technologies also presenting few schemes for jamming classification, possible strategies for jamming detection and the main jamming indicators useful to detect jamming attacks.

### 2.1. Drone networks: single-UAV and multi-UAV systems comparison

A drone network is a wireless system whose nodes can be UAVs (Unmanned Aerial Vehicles), commonly known as *flying drones*, consisting of unmanned flying devices, usually controlled by a computer embedded on board, called Flight Computer System (FCS), via a pre-programmed software. Its early usage consisted of single-UAV systems, which are networks with just one drone, executing one or many tasks [17].

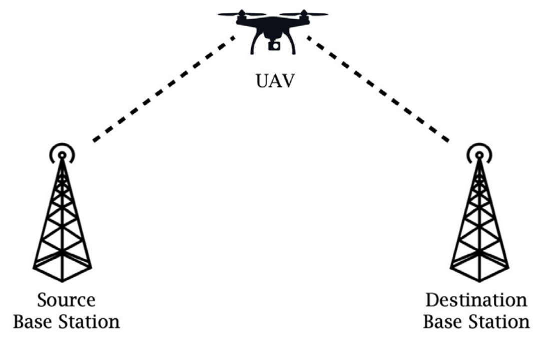
In single-UAV applications it is very simple to set up a communication environment, since all the ground nodes are linked to the aerial node in a star topology network strategy and therefore they can easily communicate with each other indirectly [18]. However, these systems pose challenges in the peer-to-peer communication such as sending more data or expanding the communication range, since if a drone flies outside the coverage of the ground base, it loses connection. Furthermore, in order to carry powerful processors, sensors and communication hardware, the drone of a single-UAV system is generally huge and this can be dangerous and expensive in case of failure.

Single-UAV systems have been in use for a long time, until the spread of a trend toward miniaturization, from the 1990s, has led to the development of small tools and equipment. Since small drones have lower acquisition and maintenance costs, this phenomenon has made them affordable to academics and even hobbyists, allowing their use in other sectors besides the military one [19]. With a smaller and lighter architecture, drones are hard to detect and do not represent a real threat to human life or the environment in which they move. Moreover, they can be transported by hand without any support and can be reused in different applications. However, the capability of a mini-UAV is restricted by many aspects, such as power, sensing and computation, and therefore these devices are designed to be integrated into multi-UAV systems. Within a multi-UAV system, drones must coordinate and collaborate, thus leading to the creation of a system whose capabilities exceed those of a single drone. Indeed, drone systems consisting of multiple small devices can offer several advantages over the use of just one large, in terms of cost, scalability, survivability, fault tolerance and other aspects [20].

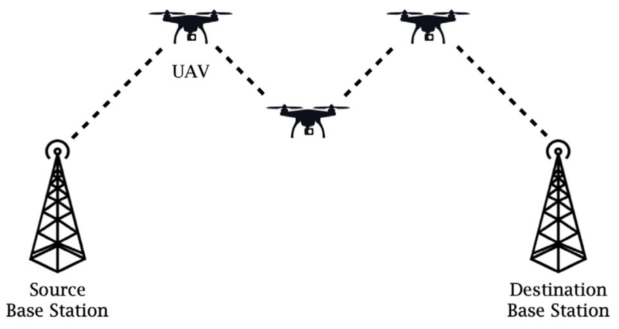
Multi-UAV systems allow to expand the area of interest of a mission as well as to achieve goals faster, parallelizing individual tasks. Moreover, the failure of one drone does not imply the failure of the entire task, which can be carried out by the other UAVs.

Ultimately, the many advantages offered by multi-UAV systems compared to single-UAV systems have the cost of greater communication and coordination complexity.

The topologies of single-UAV and multi-UAV networks are shown in Fig. 1, and the main differences between these systems are summarized in Table 1.



(a) Single-UAV system



(b) Multi-UAV system

**Fig. 1.** UAV systems topologies.

**Table 1**
Single-UAV, Multi-UAV systems comparison.

| | Single-UAV | Multi-UAV |
|---|---|---|
| Size | Large and heavy | Small and light |
| Cost | High | Low |
| Portability | Needs support | Hand-carried |
| Dangerous | High | Low |
| Survivability | Single drone life | Fault tolerance |
| Scalability | Low | High |
| Multitasks capability | Low | High |
| Goal achievement time | High | Low |
| Complexity | Low | High |

### *2.2. Jamming attacks*

Jamming attacks are the equivalent of Internet Denial-of-Service (DoS) attacks for radio signals. In particular, jamming is the act of disturbing radio (wireless) communications by causing their Signal-to-Interference-plus-Noise-Ratio (SINR) to decrease, typically transmitting on the same frequency and with the same modulation as the signal to be disturbed.

To understand how jamming attacks work, it is important to distinguish them from interference. Interferences are unintentional forms of disruption, usually due to device malfunctions, accidentally caused; on the contrary, jamming attacks are conducted by an attacker with the malicious intention to interrupt or prevent legitimate communications in the network. A pictorial representation of jamming is shown in Fig. 2.

Radio users also distinguish between *intentional* and *unintentional* jamming. We talk about unintentional jamming when a node transmits on a busy frequency without first checking whether it is in use, or when it is not able to hear stations using that frequency.

In general, attackers jam the ongoing transmissions via injecting malicious signal to the wireless channel in use.

The entity – either a malicious or unintentional wireless device – that launches such radio interference is referred to as *jammer* or *interferer*.

In order to avoid the occurrence of unintentional jamming phenomena, CSMA (Carrier Sense Multiple Access) is often used. It is a media access control (MAC) protocol according to which each node in the network, before transmitting on a shared medium, verifies the absence of other traffic. This protocol by itself actually is not enough to avoid collisions, and other mechanisms are usually adopted. Some types of jammers, as explained below, transmit without respecting the CSMA. We can define a jammer as a device that does not comply to legitimate PHY or MAC protocols and intentionally tries to hinder communications by interfering with transmitter or receiver.

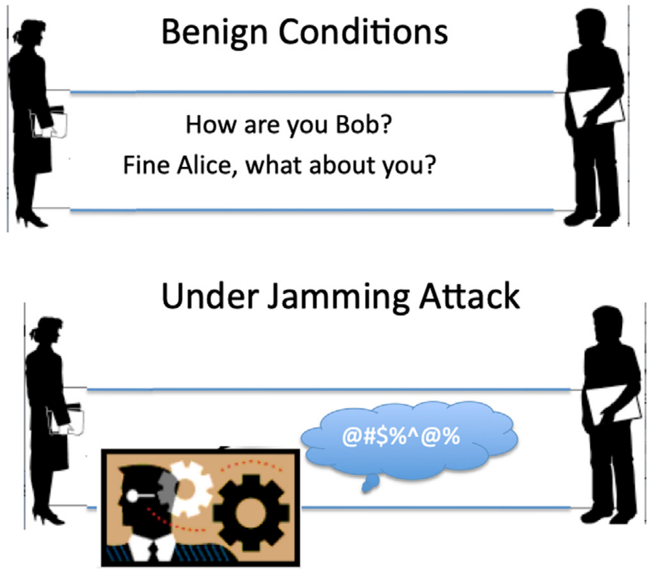


**Fig. 2.** Pictorial representation of a jamming entity [21].

The effect of a jammer depends on several aspects, such as its radio transmitter power, location and influence on the network or the targeted node.

There are many ways in which a jammer may jam a wireless network, among which it chooses the most effective one, and there are also many different ways to classify them. Some divide jamming techniques in two main categories, which are *noise techniques* and *repeater techniques*.

A detailed classification of jamming attacks, summarized in Fig. 3, among with a description of a jammer possible operating modes is proposed in [22].

In our analysis we decided to focus on reactive jamming, which is more sophisticated compared to constant jamming. Reactive jammers are considered the most critical adversarial threat to compromise networks since they operate by injecting interfering signals only when target devices are transmitting. In particular, during reactive jamming, interference is only generated when a transmission is taking place over the network, staying quiet when the channel is idle, in contrast to constant jamming, which consists in a nonstop interference. This makes reactive jamming not only cost-effective for the jammer, but also more challenging to detect, track and compensate.

In any case, the goal of a jammer is to interfere with legitimate wireless communications, and it can achieve this by acting both

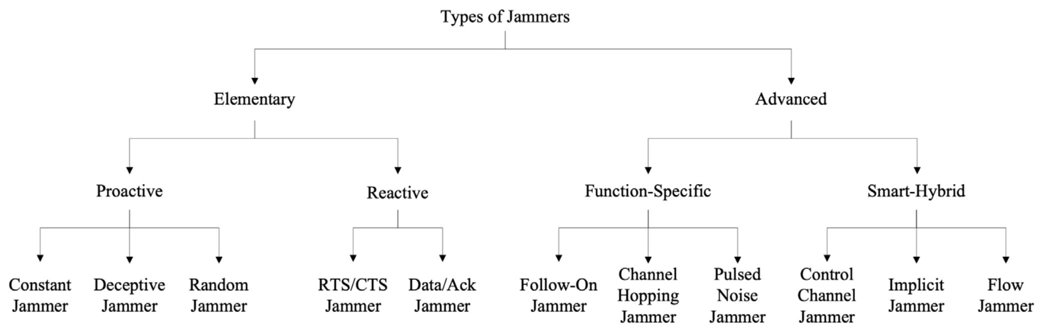


**Fig. 3.** Types of Jammers in Wireless Networks [23].

in transmission and in reception, that is by preventing the source node from sending packets or the destination node from receive legitimate packets.

### 2.3. Related works

Several techniques are known in the literature for detecting and reacting to jamming attacks, as well as some metrics whose value can be symptomatic of the presence of a jammer in action. Clearly, these techniques vary according to the type of jammer considered and its characteristics. Also, as seen before, not all jamming attacks are equally difficult to detect. For example, since reactive jammers are triggered only when packet transmission occurs over the network, they may be not so easily detected.

Once a jammer and the area in which it acts have been recognized, the other nodes in the network can counter the attack by redirecting traffic, changing channels or moving to a safe location. Some well known techniques are *channel hopping* and *frequency hopping*.

The channel hopping is the most popular countermeasure to jamming and works particularly well in the case of proactive jamming. It consists in continuously changing the communication channel after a certain period of time.

The frequency hopping, instead, selects the frequencies considered "good" (not jammed) and, if anomalous interferences are detected on those used, it changes the transmission frequencies.

In addition, many techniques can be combined, such as the direct-sequence spread spectrum (DSSS) and the frequency-hopping spread spectrum (FHSS) [24]. The DSSS uses a modulation that makes the transmitted signal wider in bandwidth than the information bandwidth [25]. In this way, the attacker detects data signals as white noise and it is not able to identify the communication radio band. The FHSS is used to avoid interference, instead.

Game theory has also often been used to model jamming attack scenarios. More specifically, the interaction between a network node and a jammer can be formulated as a Stackelberg game, in which the latter, acting as a *follower*, determines its transmission power based on the action observed by the former, which instead acts as a *leader* [26,27].

Almost all the studies carried out on jamming concern general wireless scenarios, and the literature lacks work that focuses on this attack in the specific context of drone networks, which is even more challenging because of its characteristics.

However, it is worth considering that drone networks are usually made up of small devices, with limited computational resources and power consumption constraints.

Recently, on-device Artificial Intelligence (AI) has gained significant notice thanks to its communication efficiency [28]. This is an unexplored territory: very few works apply Machine Learning for the detection of jamming attacks. In [13] an unsupervised machine learning approach is proposed through the use of clustering to detect jamming in the radio frequency communication between two vehicles. A framework based on supervised machine learning techniques for detecting jamming in optical networks is presented in [15]. In [14] the use of Reinforcement Learning against jamming attacks is evaluated in a VANET, including relying UAVs.

The success gained by *Reinforcement Learning* (RL) in the decision-making problems has led to the application of Q-learning (QL) to build anti-jamming wireless communications. According to this approach, the jammer, base on the environment, chooses its jamming action in order to maximize the reward [29]. This is long-term cumulative reward, which is determined through a series of time events. The Q-learning is a RL method, which can learn the optimal strategy based on the long-term cumulative reward with an end-to-end approach.

Although all these solutions take advantage of machine learning methods against jamming attacks, they rely on a centralized approach, which is not suitable for UAV networks.

A centralized solution is impractical for several reasons: drones have high mobility and may not always be in the communication range of the ground station, and also they usually require an immediate response. In addition, continuous communications to a centralized system is not effective, since it can cause severe power consumption [18].

## 3. Proposed framework

Detecting jamming attacks is a critical step toward building a secure and reliable wireless network. This is not a simple task, since a jammer can operate in several different ways; moreover, we have to distinguish jamming attacks scenarios from other legitimate network situations, such as congestion.

Jamming attacks detection can be implemented on dedicated devices or in customized programs running on the communication devices themselves. Generally, the latter case is preferred, since it involves lower costs.

In this section, we present a distributed solution, consisting of a framework for jamming attacks detection to be executed on each node of the network, whose objective is to indicate, starting from raw packets, the presence or absence of jamming.

In particular, we examine the metrics that can be symptoms of a jamming attack. We also present two supervised machine learning techniques, and explain how they can be used within a specific framework, combined to the chosen measures, for the detection of jamming attacks in wireless networks. In particular we used (i) a Multi-Layer Perceptron [30] (ii) and a Decision Tree [31].

The considered scenarios are controlled, which means that the network nodes have information on both sides of the communication in which they participate, transmitter and receiver.

The framework was developed using Python 3.7.2 and its code relies on the SciKit-Learn library to implement Machine Learning techniques [32].

Since there is no publicly available dataset relating to jamming attacks in drone networks, which is our problem of interest, we used the trace set collected in a VANET communications scenario [16], made available to the scientific community through CRAWDAD [33], an archive that stores wireless trace data, as already done in [34].

This decision is justified by the following reasons: *(i)* FANETs (Flying Ad hoc NETworks) [18,35] are a subset of VANETs and the two networks therefore share common characteristics; *(ii)* communication takes place using the 802.11p protocol, which, although meant for VANET, is perfectly suited to FANET, as explained in [36]; *(iii)* according to the Federal Aviation Administration, drones usually maintain the same altitude during their flight, so the third dimension of the UAVs in a FANET can be considered fixed.

### 3.1. Jamming indicators analysis

During a jamming attack, the value of some network parameters is generally subject to alterations and therefore they can be used as jamming indicators. However, there are many papers showing that there is no metric which is itself enough to conclude that a jamming attack is in progress [37]. According to these observations, we decided to analyze the combined effect of three metrics, which are Throughput, PDR and RSSI.

**Throughput.** It is a measure used to quantify the actually used transmission capacity of a telecommunication channel. Its value is lower than that of the theoretical capacity, which expresses the maximum transmission frequency at which data can travel. Throughput is the amount of data transmitted in a unit of time; therefore, it is usually expressed in bits per second (bit/s or bps), and sometimes in data packets per second (p/s or pps).

**Packet Delivery Ratio (PDR).** It is defined as the ratio of the number of packets successfully received by a destination node, compared to the number of packets sent by a source node. If a node does not receive any packet at all, the PDR is 0, however, in order to estimate the PDR, an exchange of messages must take place between the nodes of the network and, since a jamming attack can be interrupted at any time, this exchange of messages must be very frequent.

**Received Signal Strength Indicator (RSSI).** It is a measure of the relative power present in a received radio signal in a wireless network; thus, the greater the RSSI value, the stronger the signal. Generally, the RSSI is not provided from the receiving device to the user; however, IEEE 802.11-based devices make their RSSI values available to the user.

The goal of a jamming attack is to downgrade the quality of the channel surrounding a node, and therefore essentially reduce the ability of the node to send or receive packets. These three measures can undergo variations in both cases in which a jammer attacks the sender, which may not be able to transmit, and when it attacks the destination, which may not be able to receive packets.

PDR is a central measure for detecting jamming attacks: it is effective enough in distinguishing jamming scenarios, in which it may assume very close to 0 values, from congestion scenarios, in which it usually assumes lower values, but not so close to 0, as explained in [37]. However, it is not nearly as effective at distinguishing other network dynamics that can downgrade the PDR value just like a jamming attack would. This occurs for example when the sender moves out of the communication range of the receiver, or when its battery is discharged.

To address this question we rely on the results obtained by *Xu et al.* [37], which show that an enhanced jammer detection, able to differentiate between these two scenarios, can be achieved by combining the values of PDR and Signal Strength (SS). In particular, they show that, in a normal interference-free scenario, a high signal strength implies a high PDR and a low signal strength implies a low PDR. On the other hand, if the PDR is low, the SS is not necessarily low.

Two cases in which a node of the network $X$ may have low PDR values are the following: its neighboring nodes are down or the node $X$ is jammed. In the first case the SS values are also low, in the second they are high. This signal strength consistency-based consideration allows us to distinguish the two cases, and then reduce the number of false positive.

This is what happens when the Reactive jammer is active analyzed in [38]: a period of no reception, which means PDR=0, starts and ends without any corresponding decrease or increase of the RSSI.

In conclusion, there is no single network parameter that, by itself, is enough to determine that a jamming attack is in progress; therefore, it is preferred to combine multiple metrics rather than develop detection techniques that focus exclusively on the values of one of them [34,37].

### 3.2. Feature extraction

Since the dataset contains raw packets, a feature extraction is necessary in order to proceed with the analysis. By feature extraction, we mean the process of creating a new set of features out of the original one, which came with the raw data. This is done through some algorithm or transformation dependent on the nature of the data.

First of all, the CRAWDAD dataset was analyzed and processed in order to generate the dataset to be used during the classifier training and setting phases.

The data relating to each communication in the CRAWDAD trace set is stored in two files, respectively containing packets sent by the transmitter node and packets received by the receiver node. Since these two files are not synchronized, the packets stored in one of them do not necessarily coincide with all and only the packets stored in the other. Therefore, a first step of data processing consists in identifying the packets, on the transmitter and receiver side, belonging to the same time interval.

Each sender side tuple represents a packet which, among other things, contains the sending time stamp, the protocol adopted (C2X for data packets and MGM for signaling packets) and the packet size (194 bytes for data packets, shorter lengths for signaling packets). Each tuple on the receiver side includes the same information, as well as the RSSI. A second step therefore consists in filtering, on both sides, the data packets.

At this point, the remaining packets are divided among smaller and fixed-size time windows, in order to aggregate the data of the packets linked to each window in a single instance.

The basic idea is to compute, starting from the data associated with a window, the minimum, maximum, average and standard deviation values of Throughput, PDR and RSSI.

The instances extraction was done through a Python script whose steps are summarized by Algorithm 1.

The resulting dataset is made up of 12 features, 4 for each metric, plus the class label. Regarding the size of the windows, we set it to values equal to 2, 3, 4, 5 s. For each of these values, two datasets were produced: in one, the windows follow one another without overlapping and each packet falls into only one of them; in the other, there is a 50% overlap between each window and the next. In conclusion, eight datasets were generated, overall.

**Algorithm 1:** Instance Extraction

```
1  Input: Sent packets s_pck, received packets r_pck, window
   size win_size
2  Output: A labeled instance
3  Identify packets from s_pck and r_pck falling within the
   same time interval
4  Filter data packets
5  Split packets into win_size sized windows
6  for each window do
7      compute min, max, mean, dev_std Throughput
8      compute min, max, mean, dev_std PDR
9      compute min, max, mean, dev_std RSSI
10     associate label
11 end
```

**Table 2**
Class distribution.

| Datasets | | | |
|---|---|---|---|
| Window size | Instances | Normal | Jamming |
| 2 s | 1085 | 310 | 775 |
| 2 s with overlap | 2167 | 620 | 1547 |
| 3 s | 715 | 204 | 511 |
| 3 s with overlap | 1417 | 403 | 1014 |
| 4 s | 530 | 151 | 379 |
| 4 s with overlap | 1052 | 299 | 753 |
| 5 s | 419 | 118 | 301 |
| 5 s with overlap | 826 | 232 | 594 |

Table 2 shows for each of the generated datasets the number of instances and how many of them are labeled as jamming attacks.

As we can see, the datasets are not balanced with respect to the class label: the percentage of instances belonging to the Jamming class is approximately 72%, while the Normal instances constitute the remaining 28%, in each dataset.

### 3.3. Training and testing

The datasets obtained as explained in the previous section were used to train and evaluate the classifiers.

As classifiers we analyzed the behavior of Multi-Layer Perceptrons and Decision Trees, adopting the k-fold cross validation technique, with $k = 5$ to split the dataset into training set and test set. The training set and test set generated at each step of this process are balanced with respect to the distribution of labels in the starting dataset.

An Artificial Neural Network (ANN) [39] is a computational model, inspired by the human brain's neural network. It is made up of layers of interconnected nodes, called *neurons*, which are processing units, and adapts its own structure based on external or internal information, that flows through the network during the training phase. An ANN can have a variable number of layers, but it always has at least two: the *input layer* and the *output layer*. The neurons that follow the input layer and precede the output layer are organized in *hidden layers*. The network receives signals through its input layer, the nodes of which are connected with the internal nodes, arranged on the various layers. Each node processes the received signals and transmits the result to subsequent nodes. The last layer of neurons constitutes the output layer. In the construction of a neural network, neurons are assigned an *activation function*, which defines the output of that node given an input. A feed-forward neural network, commonly called a *feed-forward network*, is an ANN where connections between neurons do not form cycles, unlike recurrent neural networks.

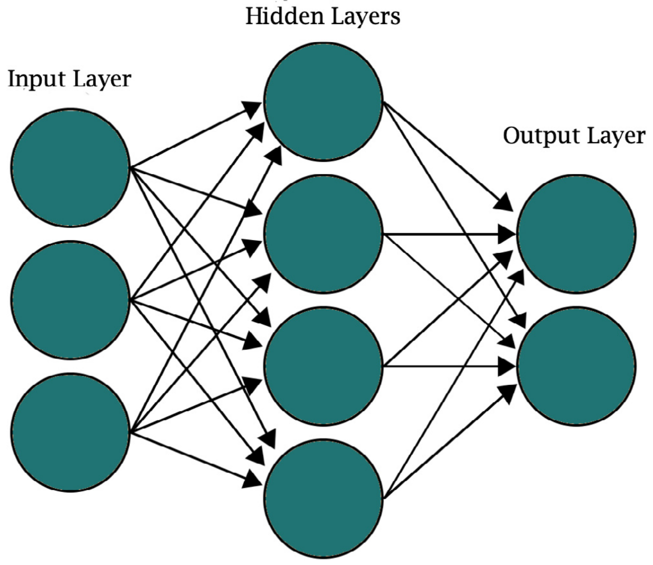


**Fig. 4.** Multi-Layer Perceptron.

The simplest feed-forward network is the Single-Layer Perceptron (SLP). A SLP consists only of an input layer and an output layer, without any hidden layer. Each input neuron is connected to each output neuron. In other words, this type of neural network has a single layer that performs data processing, hence the name. A feed-forward network having at least one hidden layer is called a Multi-Layer Perceptron (MLP) [40]. Each layer has connections coming from the previous layer and going to the next, therefore the propagation of the signal occurs forward without cycles and without transverse connections, as shown in Fig. 4.

In this case, we build a MLP with one single hidden layer, consisting of three neurons. The network uses the rectified linear unit function as activation function, which returns

$$f(x) = max(0, x).$$

A Decision Tree (DT) consists of a classification model, in which each path from the root node to a leaf node represents a conjunction of conditions that lead to the classification of an instance. Each internal node represents a set of instances and defines a split, that is a condition on a feature of the dataset, on the basis of which the data is divided. Its branches represent possible outcomes. The root node represents the set of all instances, while the leaf nodes make up the class labels. The structure of a decision three is shown in Fig. 5. A decision tree is built using learning techniques starting from the initial data set, which is divided into the two subsets we know: the training set, on the basis of which the tree structure is created, and the test set, which is used to test the accuracy of the predictive model thus created.

In order to define the split of each node, it is necessary to identify, from time to time, the relevance and importance of each of the attributes, thus place the most important as the basis of the split test. Here we chose the Gini index as test split criteria. The *Gini index* or Gini impurity is a statistical measure of distribution, which measures the degree or probability of a particular variable being wrongly classified when it is randomly chosen.

Formally, it is defined as:

$$Gini = 1 - \sum_{i=1}^{n} p_i^2$$

where $p_i$ is the probability of an object being classified to a particular class.

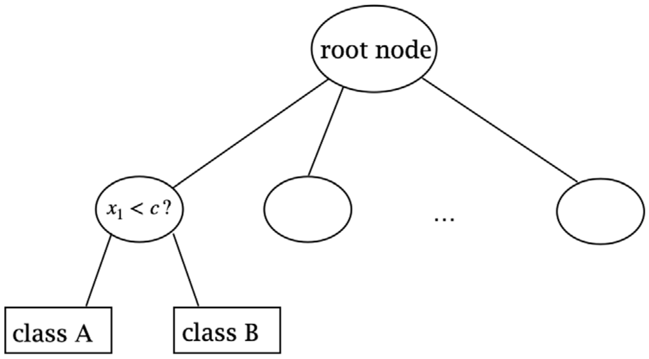


**Fig. 5.** Decision Tree.

While building the decision tree, we would prefer choosing the feature with the least Gini index as the root node.

The process of extracting the dataset and building the classifier is schematized in Fig. 6.a.

### 3.4. Jamming detection

Once the classifiers have been trained and tested offline, as explained in the previous sections, they can be used in real-time. The basic assumption is that the necessary information, which is the trace data on both the transmitter and receiver sides, is available on the node. The task is designed to collect traces for a period of time equal to the chosen window size. The processing of this data trace, which consists of the steps discussed above, leads to one unlabeled instance, which is given as input to the classifier selected among those available on the node.

The online operation of the framework is schematized in Fig. 6.b.

## 4. Performance evaluation

In this section we analyze the quality of the proposed framework and its applicability in a drone network scenario, testing the various configurations and evaluating different indicators.

In particular, we tested the goodness of our proposal on different tracesets coming out from both real and simulated communication scenarios, with the aim of investigating a wider range of cases.

### 4.1. Basic concepts for model evaluation

In order to evaluate the model, we introduce some basic concepts. Given a binary classification problem having as possible outcomes Positive and Negative labels, we denote by:

- True Positive ($TP$) $\rightarrow$ an outcome *correctly* predicted by the classifier as belonging to the Positive class;
- True Negative ($TN$) $\rightarrow$ an outcome *correctly* predicted by the classifier as belonging to the Negative class;
- False Positive ($FP$) $\rightarrow$ an outcome *incorrectly* predicted by the classifier as belonging to the Positive class;
- False Negative ($FN$) $\rightarrow$ an outcome *incorrectly* predicted by the classifier as belonging to the Negative class;

These definitions above can be also generalized to non-binary classification problems. At this point, we denote by:

- **Accuracy** — the fraction of correctly classified instances formally defined as $Accuracy = (TP+TN)/(TP+TN+FP+FN)$
- **Precision** — the fraction of correctly Positive classified instances over all the Positive classified instances formally defined as $Precision = TP/(TP + FP)$
- **Recall** — the fraction of correctly Positive classified instances over all the correctly classified instances formally defined as $Recall = TP/(TP + FN)$
- **F-measure** — the harmonic mean of precision and recall. It is an accuracy metrics and, formally, it is defined as $F = 2 \cdot (Precision \cdot Recall)/(Precision + Recall)$

To evaluate the trained classifier, we compare the labels predicted by it on the test set with the real ones, assigned by the domain expert. In this way we obtain the TP, TN, FP and FN values to be used to calculate the metrics. In the case of *k-fold* cross validation these measures are calculated as the average of their values obtained at each execution.

### 4.2. Attacker model

We assume that, according to the configuration of the experiments set in the collection of data traces, the attacker is a reactive jammer. A reactive jammer aims to compromise the reception of a message. Hence, it starts jamming only when it observes a network activity to occur on a certain channel [22]. Therefore, in order to be active, it must first detect a transmitter and to be able to impair packet delivery, it must also generate sufficient interference power at the receiver. This attack is less energy efficient than other jamming operating modes because it has to constantly monitor the network to determine when it has to transmit, but it is also much more difficult to detect.

The jammer generates a blind area, whose extension and intensity may vary, and within which the communication between the nodes is affected. Therefore, whether the nodes move in the same direction or toward each other (see Fig. 7), when they reach the blind area, they may not be able to communicate so easily.

### 4.3. Working with real datasets

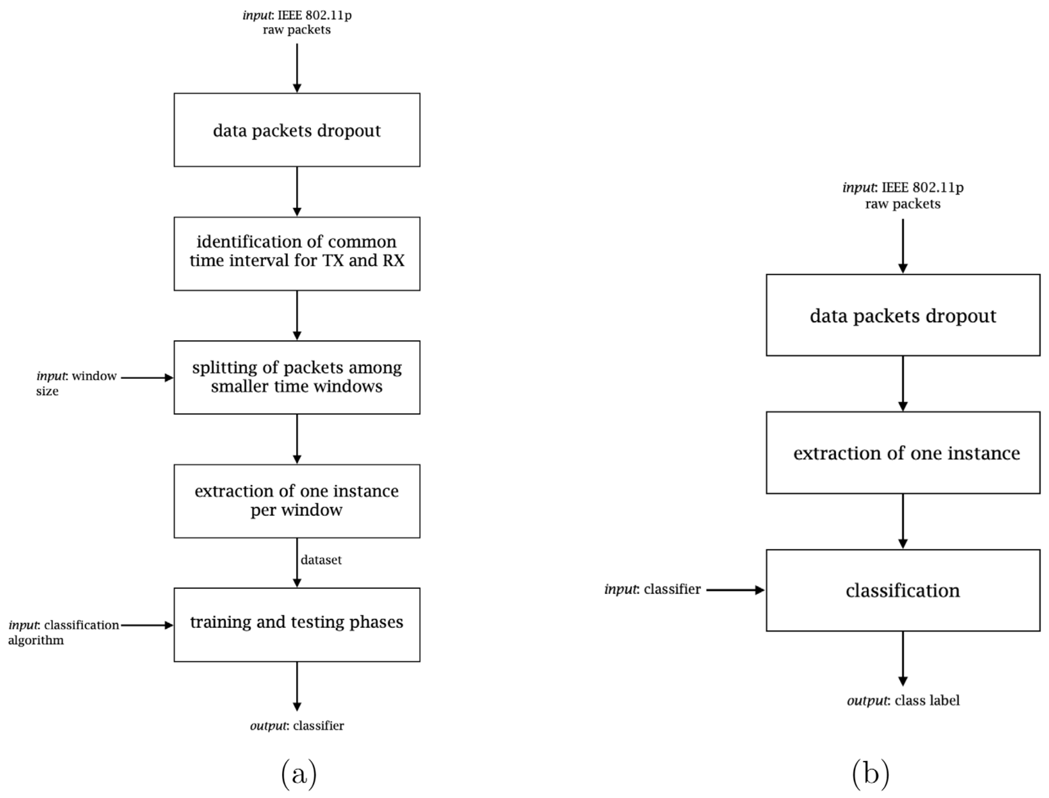
The proposed framework has been tested using 8 different datasets, which have been generated from tracesets [16], as a result of a feature extraction process. For sake of completeness, we remark that the reference tracesets consists of IEEE 802.11p raw packets exchanged among nodes of a VANET, with and without the presence of a jammer. The jammer can acts in two operation modes, constant and reactive, and the traces were collected in both indoor and outdoor scenarios.

The analysis is conducted by highlighting the values of *Accuracy*, *Precision*, *Recall* and *F-measure* gained by several classifiers operating in different network configurations.

In detail, we compare classifiers by varying: *(i)* the classification algorithm, which can be a MLP or a Decision Tree; *(ii)* the dataset used to fit the classifiers, which depends on the size of the time window, set at 2, 3, 4, 5 s, and the presence of overlapping windows.

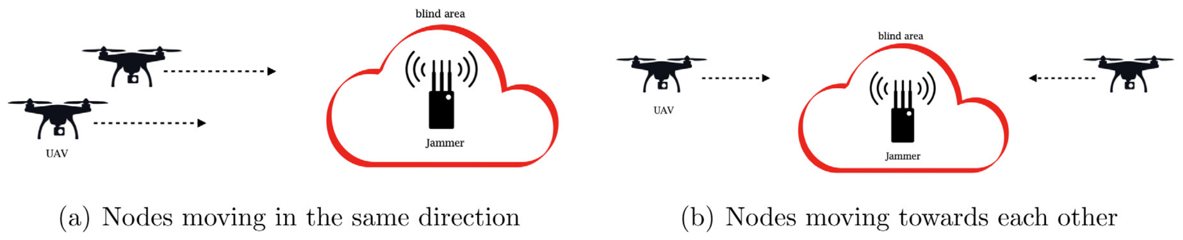
To evaluate the system performances with the aim of obtaining meaningful statistical results [41], we made 10 replications, repeating the test 10 times and averaging the obtained results; moreover, we set the confidence interval level to 0.95 and we excluded the first seconds of the reference tracesets from the statistical error computation in order to verify the correctness of the statistical analysis for the obtained results also reducing the system's transient effects.

In the following we analyze separately the quality of neural networks and decision trees, and then finally compare the two models.



input: IEEE 802.11p raw packets
data packets dropout
identification of common time interval for TX and RX
input: window size
splitting of packets among smaller time windows
extraction of one instance per window
dataset
input: classification algorithm
training and testing phases
output: classifier
(a)
input: IEEE 802.11p raw packets
data packets dropout
extraction of one instance
input: classifier
classification
output: class label
(b)

**Fig. 6.** (a) Offline operation of the framework (b) Online operation of the framework.



blind area
UAV
Jammer
(a) Nodes moving in the same direction
blind area
UAV
Jammer
(b) Nodes moving towards each other

**Fig. 7.** Attacker Model.

### *4.3.1. MLP evaluation*

By looking at the results we make two main points. First, by comparing each result obtained for the datasets with and without overlapping for a fixed window size, we observe that models induced from the former achieve higher accuracy values. It may be due to the fact that datasets with overlapping contain a greater number of instances. Also, we can observe that the performance of the MLP classifier decreases as the size of the time window increases. Fixed the number of nodes, we notice that datasets having $window\ size = 2$ s always perform better that those having $window\ size = 5$ s, in both with and without overlapping cases. These results are shown in Fig. 8.a.

MLP classifiers reach precision values between about 70% and 75%, as shown in Fig. 8.b. This is a symptom of the presence of False Positives in the classification. This is mostly due to the imbalance of the datasets, which present a majority of instances belonging to the Jamming class.

The recall values follow the same accuracy trend, as shown in Fig. 8.c where the classifiers trained on datasets with overlapped windows perform better than their respective non-overlapped ones and the performance worsens as the number of nodes in the network increases. The best case is obtained by setting $window\ size = 2$ s with overlapping.

Usually, we refer to precision and recall separately when in the problem addressed, reducing the number of False Positives is more important than reducing that of False Negatives, or vice versa.

In our case, however, minimizing the number of False Negatives is important as reducing that of False Positives, since countermeasures must be taken in response to the presence of jamming. Therefore, we analyze the F-measure, which combines the two measures, as defined in 4.1.

Also regarding the F-measure, we observe the same trend: the performance of the classifiers worsens with the increasing of the size of the time window. As seen before, datasets with overlapping lead to a better behavior of the model.

In conclusion, by comparing all the accuracy values obtained for all the possible window sizes, we can assert that accuracy drops as the window size increase.

### *4.3.2. Decision tree evaluation*

As in the previous case, we start by comparing the two cases of maximum and minimum size of the time window.

The accuracy assessment on DTs is very similar to that made for MLPs: it decreases as the size of the time window increase, and the datasets with overlapping perform better than the others.

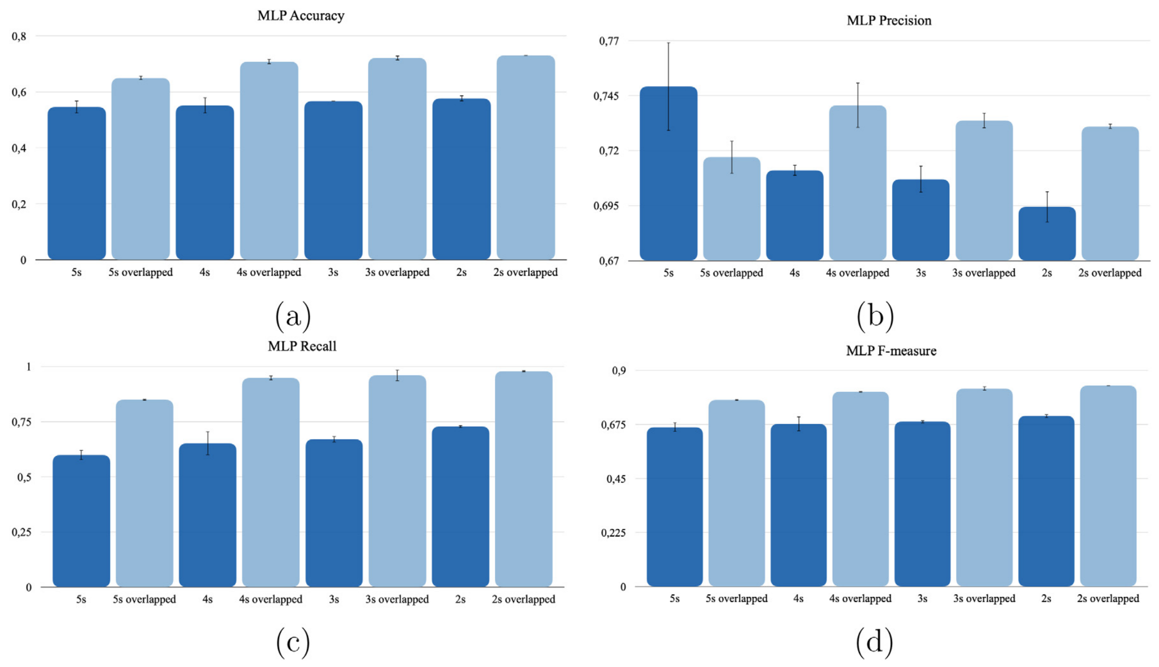


**Fig. 8.** MLP performances for *window size = 2 and 5 s* — (a) Accuracy; (b) Precision; (c) Recall; (d) F-measure.

However, unlike what happens for MLP classifiers, the differences between the performances reached by the various decision trees are slighter, as shown in Fig. 9.a. The same goes for the Precision and Recall values, and therefore for the F-measure, which reaches 95.36% for the $dataset = 2sec$ *with overlapping* case, as shown in Fig. 9.c and 9.d respectively.

### 4.3.3. Classifiers comparison

By comparing the results of MLP and DT we observe that the latter achieves better results, in all the scenarios. However, we can see the same trend followed by both the models, which can be summarized as follows: (i) the performances decrease as the size of the time window increases; (ii) the datasets with overlapping induce more performing models than those without overlap.

Fig. 10 shows a comparison between the two models and the accuracy values that they reach in the best case scenario, which is window size = 2 s with overlapping.

## 4.4. Working with simulated scenarios

It is important to emphasize that in all the scenarios in which the CRAWDAD dataset was collected, only two nodes, exchanging IEEE 802.11p packets, are involved in communication and therefore the network design does not take into account the possible presence of other devices that transmit legitimately. Moreover, those nodes only move in two directions being terrestrial vehicles and not flying drones. These simplifications could constitute a limitation and not a faithful representation of UAVs communications. However, according to the Federated Aviation Administration, Fact Sheet on Unmanned Aircraft Regulations [42] the UAVs usually maintain a fixed altitude because of which the third dimension of the UAVs in the FANET can be considered fixed. Therefore, given that no other standard UAV jamming attack datasets are currently available, the jamming attack dataset in the vehicular ad hoc network performs as the closest fit for the numerical analysis and it can be used as a proof-of-concept for jamming detection in FANET.

By following the previous considerations, aiming at generating and studying more realistic drone network communication scenarios to validate our proposal, we also simulated several ns-3 [43] based Flying Ad hoc Network topologies with three-dimensional mobility model in an ad hoc setting, communicating over the WiFi physical standard 802.11n [44,45]. A jammer node was introduced with reactive RF jamming signals which interferes with the communication between UAV nodes playing the roles of simple transmitters and an UAV equipped with long-range communication technology playing the role of a receiver server node, in the three-dimensional UAV ad hoc Gauss Markov mobility model [46]. We extracted 15786 instances with 12 features each consisting of PDR, RSSI, and Throughput considering a window size of 2 s with overlapping.

Two network topologies were considered: linear and grid. The linear scenarios consist of nodes arranged in a straight line, while in the grid ones the transmitters surround the receiver, making up each side of a square. As an example, this last configuration can represent a real application scenario in which the receiver drone can work as an edge gateway collecting the information coming from the other neighboring drones in order to process data locally at the edge or to forward data to a remote server using long-range communication technologies. In any case, the jammer is going after the receiver, in a *follow-me* pattern. All the nodes are moving at similar speeds, between 25 and 35 km/h and at an altitude between 5 and 10 m.

For each of the considered settings, the jammer was placed at two different distances from the receiver in order to emulate two types of attack: the closer it is, the stronger the attack.

Moreover, in order to test the proposed framework in different channel conditions, we varied the average interference values in terms of RSSI experienced by the drone acting as receiver. In this way we could collect and analyze communication data traces in which the effect of radio interference was considered. In particular, according to the work in [47] and the selected radio communication technology, we considered two opposite channel conditions (*i.e.*, good and poor) corresponding to mean RSSI values of $\sim -63$ dBm and $\sim -88$ dBm respectively; we also tested intermediate conditions of low interference ($\sim -74$ dBm) and medium interference ($\sim -81$ dBm). Table 3 summarizes the main simulation parameters.

We started by analyzing three basic settings, which are shown in Fig. 11: two linear, with one and two transmitting nodes respectively, and a grid, with four transmitting nodes. In details, in

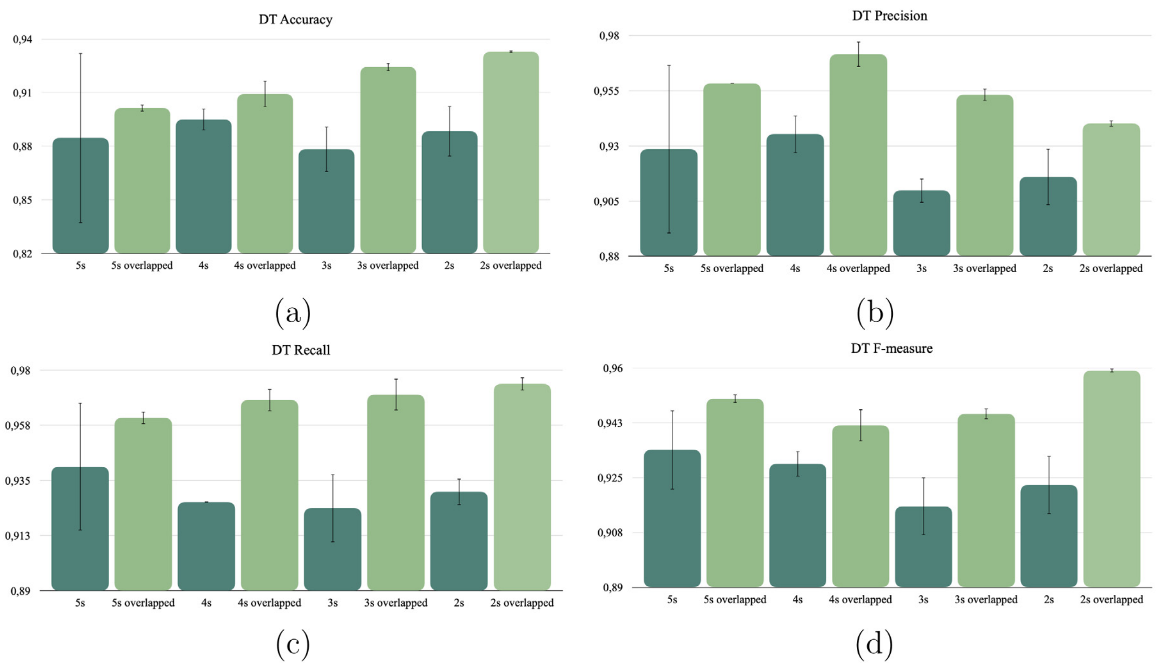


**Fig. 9.** TREE performances for *window size = 2 and 5 s* — (a) Accuracy; (b) Precision; (c) Recall; (d) F-measure.

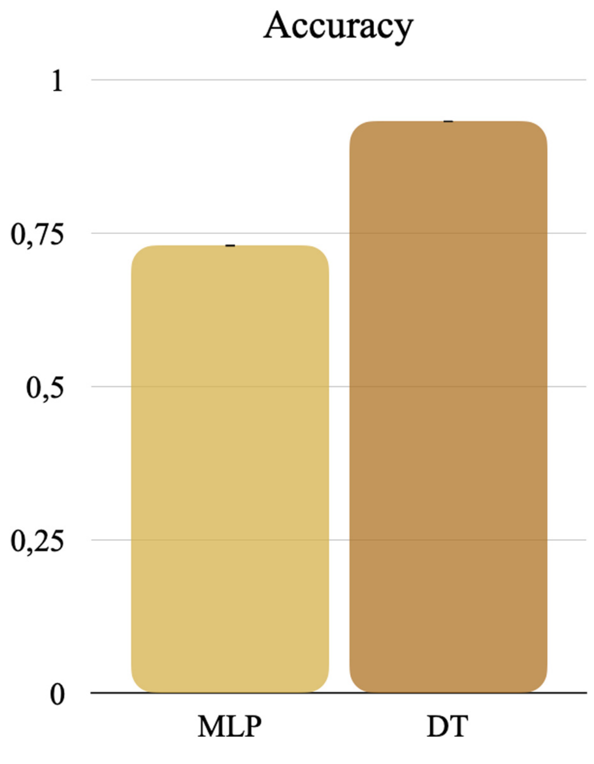


**Fig. 10.** MLP/DT Accuracy comparison for *window size = 2 s with overlapping*.

order to emulate a fairly effective attack, the jammer was placed at 5 m and at 20 m in the linear scenario with one transmitter, at 20 m and at 50 m in the linear scenario with two jammers, and at 30 m and at 80 m in the grid scenario with four transmitters.

Later, we generated some more complex settings, still following the linear and grid network topologies, but with a larger number of transmitters: linear with 4 transmitters, linear with 8 transmitters, linear with 10 transmitters, grid with 8 transmitters, and grid with 10 transmitters.

**Table 3**
Simulation parameters.

| Parameter | Value |
|---|---|
| Drones speed | [25 km/h–35 km/h] uniformly distributed |
| Drones height | [5 m–10 m] uniformly distributed |
| Flying time | 5 min |
| Radio propagation model | Friis |
| Communication technology | IEEE 802.11n |
| Average receiver interference | high, medium, low |
| Data rate | 90 packets/s |
| Jamming type | Reactive |
| Windows size | 2 s with overlapping |

### 4.4.1. Results

In this section we present the results obtained throughout the simulations in terms of classifiers accuracy.

First of all, we generated several datasets collecting the packets exchanged among nodes in each of the previously depicted basic scenarios, and we used them to build a series of classifiers, whose accuracy values are shown in Table 4. It is worth to note that, in these basic scenarios, the DT classifiers always achieves better performances compared to the MLPs, regardless of the interference and jamming levels. Afterward, we combined the datasets related to the same scenario obtaining three new datasets comprising of different interference and jamming levels, and we trained new DTs and MLPs on these datasets. Then, these three datasets were merged into a single one, which we used to train two new classifiers: a DT and a MLP, reaching an accuracy of 93% and 86.7% respectively. These two classifiers were tested on several datasets corresponding to both the basic and more complex scenarios, aiming at making more general the conducted study. The resulting accuracy values are listed in Table 5.

By comparing the performances of the classifiers on each dataset individually, we can observe that DT performs better than MLP only for those datasets used to build the classifiers. In other words, MLP seems to be more capable of generalization, achieving higher accuracy values than DT when the testset is not included in the initial dataset.

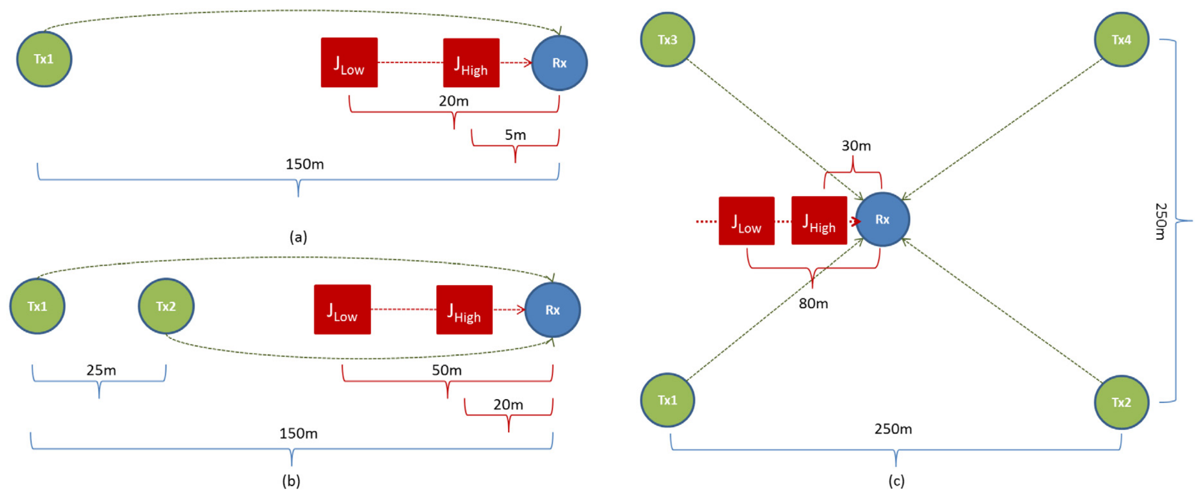


**Fig. 11.** Simulated drone network topology — (a) Linear-1Tx; (b) Linear-2Tx; (c) Grid-4Tx;.

**Table 4**
Decision Tree (DT) and Multi Layer Perceptron (MLP) accuracy in reference communication scenarios.

| Communication scenario | Interference level | Jamming level | Accuracy DT | Accuracy MLP |
|---|---|---|---|---|
| Linear-1Tx | None | Low | 0,9 | 0,87 |
| | | High | 0,94 | 0,93 |
| | Low | Low | 0,87 | 0,85 |
| | | High | 0,92 | 0,91 |
| | Medium | Low | 0,81 | 0,79 |
| | | High | 0,87 | 0,85 |
| | High | Low | 0,71 | 0,67 |
| | | High | 0,76 | 0,74 |
| Linear-2Tx | None | Low | 1 | 0,52 |
| | | High | 1 | 0,51 |
| | Low | Low | 0,93 | 0,53 |
| | | High | 0,92 | 0,507 |
| | Medium | Low | 0,91 | 0,52 |
| | | High | 0,9 | 0,51 |
| | High | Low | 0,98 | 0,505 |
| | | High | 0,976 | 0,5 |
| Grid-4Tx | None | Low | 0,998 | 0,73 |
| | | High | 1 | 0,75 |
| | Low | Low | 0,998 | 0,73 |
| | | High | 0,998 | 0,81 |
| | Medium | Low | 0,998 | 0,75 |
| | | High | 0,998 | 0,79 |
| | High | Low | 0,998 | 0,75 |
| | | High | 0,998 | 0,83 |

**Table 5**
Accuracy of trained classifiers on new and more complex communication scenarios.

| Linear scenarios | DT | MLP | Grid scenarios | DT | MLP |
|---|---|---|---|---|---|
| 1Tx | 0.85 | 0.82 | 4Tx | 1 | 0.94 |
| 2Tx | 0.92 | 0.80 | 8Tx | 0.5 | 0.9 |
| 4Tx | 0.9 | 0.95 | 10Tx | 0.5 | 0.96 |
| 8Tx | 0.5 | 0.79 | Training Set Linear 1Tx +2Tx +Grid 4Tx | 0.93 | 0.86 |
| 10Tx | 0.36 | 0.92 | | | |

### 4.5. Framework execution time and memory occupation

The proposed framework allows to perform a jamming attack detection activity in wireless drone networks.

Since our goal is to integrate it into a multi-UAV system having limited computing powers, we carried out an analysis in terms of execution time and memory occupation in order to verify the feasible implementation of the framework.

**Table 6**
Online execution times of MLP and DT.

| Window size | MLP [s] | DT [s] |
|---|---|---|
| 2 s | 0.399 | 0.378 |
| 3 s | 0.392 | 0.381 |
| 4 s | 0.383 | 0.376 |
| 5 s | 0.401 | 0.390 |

The framework has been tested on a constrained device such as the Raspberry Pi 3, whose main hardware characteristics are:

- CPU: 4×ARM Cortex-A53, 1.2 GHz
- GPU: Broadcom VideoCore IV
- RAM: 1GB LPDDR2 (900 MHz)
- Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy

These features are supported by most commercial drones, so it is realistic to think that the framework can also run on other UAVs having similar computing power.

By executing the framework we discovered that the memory space occupied by an MLP varies between 10256 and 16552 bytes. An instance of the Decision Tree occupies 2192 bytes, instead.

We also observed a time execution which ranges between 0.378 s and 0.4 s The exact values are listed in Table 6. Clearly, this time refers to the online execution of the framework, that is to the actual detection activity, while the offline one varies between 5.02 s and 7.5 s for the MLP, and between 0.44 s and 0.8 s for the DT.

Ideally, execution times should vary according to the size of the window: the wider the window, the greater the number of traces that fall within it. This, however, is not true in the case of possible interference or jamming: during an attack, in a 5 s window no packet could be sent (or received), while, in normal conditions, in a 2 s window much more data could be transmitted.

In conclusion we verified that Off-the-Shelf flying drones such as the *Intel Aero Ready-to-Fly* [48] have hardware features that exceed those listed above, with a 2.4 GHz Quad Core CPU and 4GB of RAM; thus, these devices are easily able to run the framework even shorter times.

## 5. Conclusions

Due to the shared nature of the communication medium, wireless networks are subject to various radio frequency attacks, such as jamming. However, many of the existent solutions are not suitable for the context of specific wireless networks, such as drone networks, whose nodes are highly mobile and usually have limited computational capabilities and energy resources. For these reasons we designed a specific framework for jamming detection by developing and training two different machine learning techniques (DT and MLP) using datasets coming from both real and simulated communication scenarios. Although both the designed classifiers obtain good performance in terms of jamming detection accuracy, we verified that the MLP is more effective than the DT when applied to communication scenarios for which it has not been trained. Finally, we note that given its execution times and hardware requirements our proposed framework is suitable for most off-the-shelf aerial drones, which are in need of swift decisions and are resource constrained.

Future works will be focused on the integration of the proposed framework within a wider routing protocol framework for drone networks, such as in [49], to make the drone network safer. We also intend to analyze other types of jamming attacks and extend the framework to more complex network scenarios with multiple transmitters, receivers, and jammers. Other jamming strategies can be simulated once again by using NS3, which beside reactive allows to model and implement constant, random, and eavesdropper jammers working in different communication conditions. Even though some jamming strategies are easier to detect, such as constant jamming, this is still a challenging task because, similarly to the presented analysis, further communication topologies and specific network scenarios need to be tested with the aim of designing a more general detection strategy to best discern jamming attacks from other legitimate network situations, such as congestion or standard channel interference.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgment

This work has been supported by the Italian MIUR, PRIN 2017 Project "Fluidware" (CUP H24I17000070001).

## References

[1] A.I. Hentati, L.C. Fourati, Comprehensive survey of UAVs communication networks, Comput. Stand. Interfaces 72 (2020) 103451, http://dx.doi.org/10.1016/j.csi.2020.103451.

[2] H. Shakhatreh, A.H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N.S. Othman, A. Khreishah, M. Guizani, Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges, IEEE Access 7 (2019) 48572–48634, http://dx.doi.org/10.1109/ACCESS.2019.2909530.

[3] C.C. Baseca, J.R. Díaz, J. Lloret, Communication ad hoc protocol for intelligent video sensing using AR drones, in: 2013 IEEE 9th International Conference on Mobile Ad-Hoc and Sensor Networks, 2013, pp. 449–453, http://dx.doi.org/10.1109/MSN.2013.115.

[4] J.-A. Maxa, M.S.B. Mahmoud, N. Larrieu, Secure routing protocol design for uav ad hoc networks, in: 2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC), IEEE, 2015, 4A5–1.

[5] C. Karlof, D.A. Wagner, Secure routing in wireless sensor networks: attacks and countermeasures, Ad Hoc Netw. 1 (2–3) (2003) 293–315, http://dx.doi.org/10.1016/S1570-8705(03)00008-8.

[6] P. Papadimitratos, Z. Haas, Secure routing for mobile ad hoc networks, in: Communication Networks and Distributed Systems Modeling and Simulation Conference, CNDS 2002, (CONF) SCS, 2002.

[7] C. Cambra Baseca, S. Sendra, J. Lloret, J. Tomas, A smart decision system for digital farming, Agronomy 9 (5) (2019) 216, http://dx.doi.org/10.3390/agronomy9050216.

[8] R.H. Jhaveri, S.J. Patel, D.C. Jinwala, Dos attacks in mobile ad hoc networks: A survey, in: 2012 Second International Conference on Advanced Computing & Communication Technologies, IEEE, 2012, pp. 535–541.

[9] V. Navda, A. Bohra, S. Ganguly, D. Rubenstein, Using channel hopping to increase 802.11 resilience to jamming attacks, in: INFOCOM 2007. 26th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 6-12 May 2007, Anchorage, Alaska, USA, IEEE, 2007, pp. 2526–2530, http://dx.doi.org/10.1109/INFCOM.2007.314.

[10] W. Xu, K. Ma, W. Trappe, Y. Zhang, Jamming sensor networks: attack and defense strategies, IEEE Netw. 20 (3) (2006) 41–47, http://dx.doi.org/10.1109/MNET.2006.1637931.

[11] S. Djuraev, J. Choi, K. Sohn, S.Y. Nam, Channel hopping scheme to mitigate jamming attacks in wireless LANs, EURASIP J. Wirel. Commun. Netw. 2017 (2017) 11, http://dx.doi.org/10.1186/s13638-016-0785-z.

[12] M. Di Felice, A. Trotta, L. Bedogni, L. Bononi, F. Panzieri, G. Ruggeri, V. Loscrí, P. Pace, STEM-mesh: Self-organizing mobile cognitive radio network for disaster recovery operations, in: 9th International Wireless Communications and Mobile Computing Conference, IWCMC, 2013, pp. 602–608.

[13] D. Karagiannis, A. Argyriou, Jamming attack detection in a pair of RF communicating vehicles using unsupervised machine learning, Veh. Commun. 13 (2018) 56–63, http://dx.doi.org/10.1016/j.vehcom.2018.05.001.

[14] L. Xiao, X. Lu, D. Xu, Y. Tang, L. Wang, W. Zhuang, UAV relay in VANETs against smart jamming with reinforcement learning, IEEE Trans. Veh. Technol. 67 (5) (2018) 4087–4097, http://dx.doi.org/10.1109/TVT.2018.2789466.

[15] M. Bensalem, S.K. Singh, A. Jukan, On detecting and preventing jamming attacks with machine learning in optical networks, in: 2019 IEEE Global Communications Conference, GLOBECOM 2019, Waikoloa, HI, USA, December 9-13, 2019, IEEE, 2019, pp. 1–6, http://dx.doi.org/10.1109/GLOBECOM38437.2019.9013238.

[16] O. Puñal, C. Pereira, A. Aguiar, J. Gross, CRAWDAD dataset uportorwthaachen/vanetjamming2012 (v. 2014-05-12), 2014, Downloaded from https://crawdad.org/uportorwthaachen/vanetjamming2012/20140512.

[17] M. Aljehani, M. Inoue, Communication and autonomous control of multi-UAV system in disaster response tasks, in: G. Jezic, M. Kusek, Y.J. Chen-Burger, R.J. Howlett, L.C. Jain (Eds.), Agent and Multi-Agent Systems: Technology and Applications, 11th KES International Conference, KES-AMSTA 2017, Vilamoura, Algarve, Portugal, June 21-23, 2017, Proceedings, in: Smart Innovation, Systems and Technologies, vol. 74, Springer, 2017, pp. 123–132, http://dx.doi.org/10.1007/978-3-319-59394-4_12.

[18] O.K. Sahingoz, Networking models in flying ad-hoc networks (FANETs): Concepts and challenges, J. Intell. Robot. Syst. 74 (1–2) (2014) 513–527, http://dx.doi.org/10.1007/s10846-013-9959-7.

[19] H. Chao, Y. Cao, Y. Chen, Autopilots for small fixed-wing unmanned air vehicles: A survey, in: 2007 International Conference on Mechatronics and Automation, IEEE, 2007, pp. 3144–3149.

[20] I. Bekmezci, O.K. Sahingoz, S. Temel, Flying ad-hoc networks (FANETs): A survey, Ad Hoc Netw. 11 (3) (2013) 1254–1270, http://dx.doi.org/10.1016/j.adhoc.2012.12.004.

[21] K. Pelechrinis, M. Iliofotou, S.V. Krishnamurthy, Denial of service attacks in wireless networks: The case of jammers, IEEE Commun. Surv. Tutor. 13 (2) (2011) 245–257, http://dx.doi.org/10.1109/SURV.2011.041110.00022.

[22] K.-M. Xu, M. Zhang, Z.A. Eitzen, S.J. Ghan, S.A. Klein, X. Wu, S. Xie, M. Branson, A.D. Del Genio, S.F. Iacobellis, et al., Modeling springtime shallow frontal clouds with cloud-resolving and single-column models, J. Geophys. Res.: Atmos. 110 (D15) (2005).

[23] K. Grover, A.S. Lim, Q. Yang, Jamming and anti-jamming techniques in wireless networks: a survey, Int. J. Ad Hoc Ubiquitous Comput. 17 (4) (2014) 197–215, http://dx.doi.org/10.1504/IJAHUC.2014.066419.

[24] A. Mpitziopoulos, D. Gavalas, G.E. Pantziou, C. Konstantopoulos, Defending wireless sensor networks from jamming attacks, in: Proceedings of the IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC 2007, 3-7 September 2007, Athens, Greece, IEEE, 2007, pp. 1–5, http://dx.doi.org/10.1109/PIMRC.2007.4394775.

[25] Wikipedia contributors, Direct-sequence spread spectrum — Wikipedia, the free encyclopedia, 2020, https://en.wikipedia.org/w/index.php?title=Direct-sequence_spread_spectrum&oldid=954223274. (Online; Accessed 9 June 2020).

[26] L. Jia, F. Yao, Y. Sun, Y. Xu, S. Feng, A. Anpalagan, A hierarchical learning solution for anti-jamming stackelberg game with discrete power strategies, IEEE Wirel. Commun. Lett. 6 (6) (2017) 818–821, http://dx.doi.org/10.1109/LWC.2017.2747543.

[27] L. Xiao, T. Chen, J. Liu, H. Dai, Anti-jamming transmission stackelberg game with observation errors, IEEE Commun. Lett. 19 (6) (2015) 949–952, http://dx.doi.org/10.1109/LCOMM.2015.2418776.

[28] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: A. Singh, X.J. Zhu (Eds.), Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA, in: Proceedings of Machine Learning Research, vol. 54, PMLR, 2017, pp. 1273–1282, http://proceedings.mlr.press/v54/mcmahan17a.html.

[29] N. Gao, Z. Qin, X. Jing, Q. Ni, Anti-intelligent UAV jamming strategy via deep Q-networks, in: 2019 IEEE International Conference on Communications, ICC 2019, Shanghai, China, May 20-24, 2019, IEEE, 2019, pp. 1–6, http://dx.doi.org/10.1109/ICC.2019.8762016.

[30] S.K. Pal, S. Mitra, Multilayer perceptron, fuzzy sets, and classification, IEEE Trans. Neural Netw. 3 (5) (1992) 683–697, http://dx.doi.org/10.1109/72.159058.

[31] S.R. Safavian, D. Landgrebe, A survey of decision tree classifier methodology, IEEE Trans. Syst. Man Cybern. 21 (3) (1991) 660–674.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, CoRR abs/1201.0490 (2012) arXiv:1201.0490.

[33] D. Kotz, T. Henderson, I. Abyzov, J. Yeo, CRAWDAD dataset dartmouth/campus (v. 2009-09-09), 2009, Downloaded from https://crawdad.org/dartmouth/campus/20090909.

[34] N.I. Mowla, N.H. Tran, I. Doh, K. Chae, Federated learning-based cognitive detection of jamming attack in flying ad-hoc network, IEEE Access 8 (2020) 4338–4350, http://dx.doi.org/10.1109/ACCESS.2019.2962873.

[35] M.A. Khan, A. Safi, I.M. Qureshi, I.U. Khan, Flying ad-hoc networks (FANETs): A review of communication architectures, and routing protocols, in: First International Conference on Latest Trends in Electrical Engineering and Computing Technologies, INTELLECT 2017, Karachi, Pakistan, November 15-16, 2017, IEEE, 2017, pp. 1–9, http://dx.doi.org/10.1109/INTELLECT.2017.8277614.

[36] V.M. Vishnevskiy, K.E. Samouylov, D.V. Kozyrev (Eds.), Distributed computer and communication networks - 23rd international conference, DCCN 2020, moscow, Russia, september 14-18, 2020, revised selected papers, in: Lecture Notes in Computer Science, vol. 12563, Springer, 2020, http://dx.doi.org/10.1007/978-3-030-66471-8.

[37] W. Xu, W. Trappe, Y. Zhang, T. Wood, The feasibility of launching and detecting jamming attacks in wireless networks, in: P.R. Kumar, A.T. Campbell, R. Wattenhofer (Eds.), Proceedings of the 6th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2005, Urbana-Champaign, IL, USA, May 25-27, 2005, ACM, 2005, pp. 46–57, http://dx.doi.org/10.1145/1062689.1062697.

[38] O.P. nal, A. Aguiar, J. Gross, In VANETs we trust?: characterizing RF jamming in vehicular networks, in: J.B. Kenney, J. Gozálvez, F. Bai, R. Kravets (Eds.), Proceedings of the Ninth ACM International Workshop on Vehicular Inter-Networking, Systems, and Applications, VANET '12, Low Wood Bay, Lake District, UK, June 25, 2012, ACM, 2012, pp. 83–92, http://dx.doi.org/10.1145/2307888.2307903.

[39] P. Kumar, P. Sharma, Artificial neural networks-a study, Int. J. Emerg. Eng. Res. Technol. 2 (2) (2014) 143–148.

[40] M.W. Gardner, S. Dorling, Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences, Atmos. Environ. 32 (14–15) (1998) 2627–2636.

[41] A.M. Law, W.D. Kelton, W.D. Kelton, Simulation Modeling and Analysis, vol. 3, McGraw-Hill, New York, 2000.

[42] M. Mozaffari, W. Saad, M. Bennis, Y. Nam, M. Debbah, A tutorial on UAVs for wireless networks: Applications, challenges, and open problems, IEEE Commun. Surv. Tutor. 21 (3) (2019) 2334–2360, http://dx.doi.org/10.1109/COMST.2019.2902862.

[43] Network simulator 3, http://www.nsnam.org/. (Online; Accessed 04 January 2021).

[44] M. Asadpour, D. Giustiniano, K.A. Hummel, S. Heimlicher, Characterizing 802.11n aerial communication, in: P.R. Kumar, D. Ghose, K. Namuduri, Y. Wan (Eds.), Proceedings of the Second ACM MobiHoc Workshop on Airborne Networks and Communications, ANC@MobiHoc 2013, Bangalore, India, July 29, 2013, ACM, 2013, pp. 7–12, http://dx.doi.org/10.1145/2491260.2491262.

[45] N. Goddemeier, S. Rohde, C. Wietfeld, Experimental validation of RSS driven UAV mobility behaviors in IEEE 802.11s networks, in: Workshops Proceedings of the Global Communications Conference, GLOBECOM 2012, 3-7 December 2012, Anaheim, California, USA, IEEE, 2012, pp. 1550–1555, http://dx.doi.org/10.1109/GLOCOMW.2012.6477816.

[46] J.P. Rohrer, E.K. Çetinkaya, H. Narra, D. Broyles, K. Peters, J.P.G. Sterbenz, Aerorp performance in highly-dynamic airborne networks using 3D Gauss-Markov mobility model, in: MILCOM 2011 - 2011 IEEE Military Communications Conference, Baltimore, MD, USA, November 7-10, 2011, IEEE, 2011, pp. 834–841, http://dx.doi.org/10.1109/MILCOM.2011.6127781.

[47] M. Tauber, S.N. Bhatti, Low RSSI in WLANs: Impact on application-level performance, in: International Conference on Computing, Networking and Communications, ICNC 2013, San Diego, CA, USA, January 28-31, 2013, IEEE Computer Society, 2013, pp. 123–129, http://dx.doi.org/10.1109/ICCNC.2013.6504066.

[48] Intel® aero ready to fly drone, https://www.intel.it/content/www/it/it/drones/aero-ready-to-fly-brief.html. (Online; Accessed 04 January 2021).

[49] L. Bertizzolo, S. D'Oro, L. Ferranti, L. Bonati, E. Demirors, Z. Guan, T. Melodia, S. Pudlewski, SwarmControl: An automated distributed control framework for self-optimizing drone networks, in: 39th IEEE Conference on Computer Communications, INFOCOM 2020, Toronto, on, Canada, July 6-9, 2020, IEEE, 2020, pp. 1768–1777, http://dx.doi.org/10.1109/INFOCOM41043.2020.9155231.