Jamming Detection in Low-BER Mobile Indoor Scenarios via Deep Learning

Savio Sciancalepore*†, Fabrice Kusters*, Nada Khaled Abdelhadi*, Gabriele Oligeri‡

***Abstract*—The current state of the art on jamming detection relies on link-layer metrics. A few examples are the bit-error rate (BER), the packet delivery ratio, the throughput, and the signal-to-noise ratio (SNR). As a result, these techniques can only detect jamming *ex-post*, i.e., once the attack has already taken down the communication link. These solutions are unfit for mobile devices, e.g., drones, which might lose the connection to the remote controller, being unable to predict the attack. Our solution is rooted in the idea that a drone unknowingly flying toward a jammed area is experiencing an increasing effect of the jamming, e.g., in terms of BER and SNR. Therefore, drones might use the abovementioned phenomenon to detect jamming before the increase of the BER and the decrease of the SNR completely disrupt the communication link. Such an approach would allow drones and their pilots to make informed decisions and maintain complete control of navigation, enhancing security and safety. This paper proposes Bloodhound+, a solution for jamming detection on mobile devices in low-BER regimes. Our approach analyzes raw physical-layer information (I-Q samples) acquired from the wireless channel. We assemble this information into grayscale images and use sparse autoencoders to detect image anomalies caused by jamming attacks. To test our solution against a broad set of configurations, we acquired a large dataset of indoor measurements using multiple hardware, jamming strategies, and communication parameters. Our results indicate that Bloodhound+ can detect indoor jamming up to 20 meters from the jamming source at the minimum available relative jamming power, with a minimum accuracy of 99.7%. Our solution is also robust to various sampling rates adopted by the jammer and to the type of signal used for jamming.**

***Index Terms*—Wireless Security; Artificial Intelligence for Security; Drones Security; Mobile Security.**

I. Introduction

Drones, a.k.a. Unmanned Aerial Vehicles (UAVs), represent an evolution of the Internet of Things (IoT) paradigm. Since we define the IoT as a network of physical objects (things) that are embedded with sensors, software, and other technologies to connect and exchange data with other devices and systems over the Internet [1],[2], it is immediate to include drones in the IoT domain. They are battery-powered devices, featuring limited onboard storage space and integrating Central Processing Units (CPUs) with heterogeneous computational capabilities. Recent papers even coined the notion of *Flying IoT* to specifically identify drones as a specification of the IoT paradigm [3], [4], [5], [6].

Today, drones are increasingly used both outdoors, e.g., for disaster management, search and rescue, and goods delivery [7], and indoors, e.g., for inventory management, intralogistics of items, inspection and surveillance [8], and leading companies such as IKEA and Amazon are actively experimenting products for warehouse management and home surveillance [9], [10]. In this context, leading market analysis companies estimate a current market value of up to 100.37 billion USD by 2029, with a compound annual growth rate of 25.5% in the next years [11].

As their role becomes more central, drones are increasingly the main target of several cybersecurity attacks. In particular, due to the reliance on wireless channels for communication, video streaming, and telemetry, attackers can easily disrupt drones' operation through *jamming* attacks [12]. Jamming can significantly disrupt wireless communications in a given area by injecting high-power noise into the same channel used by legitimate communication parties [13]. Depending on the firmware onboard, the drone might return to the mission's starting point, land, or even crash, with potential hazards for people in the area, especially for indoor applications [14].

At the time of writing, most of the available solutions for jamming detection work by identifying the deterioration of the Bit-Error-Rate (BER), the Packet Delivery Rate (PDR), or the Signal to Noise Ratio (SNR) of the communication link. Some of them also use the *spectrogram* of the signal in the Physical (PHY) layer, looking for sudden anomalies (see Section II for an overview). Such solutions work reliably and effectively when applied to static deployments. However, they can detect jamming mainly *ex-post*, i.e., once jamming has already disrupted the regular operations of the communication link. Applying such approaches to drones would trigger the default actions listed above, possibly causing hazards to the drone and surrounding people. In this context, mobile devices such as drones typically experience an increasing effect of jamming (increasingly high BER, low PDR, low SNR) while approaching the jammed area. Drones might exploit the mentioned phenomena to deploy a solution to detect jamming in a low-BER regime, i.e., before entering an area where the high-power noise injected by the jammer completely disrupts the communication link. Such a solution is especially critical for indoor applications, where people and drones often work together in limited space.

Contribution. In this paper, we propose *BloodHound+*, an innovative solution for jamming detection in low-BER mobile indoor scenarios leveraging state-of-the-art Deep Learning

This is a personal copy of the authors. Not for redistribution. The final published version of the paper accepted in IEEE Internet of Things Journal is available through the IEEExplore Digital Library, with the DOI: 10.1109/JIOT.2023.3343611.

*Eindhoven University of Technology, Eindhoven, Netherlands.
†Eindhoven Artificial Intelligence Systems Institute (EAISI). Email: s.sciancalepore@tue.nl, {f.e.kusters, n.k.s.abdelhadi}@student.tue.nl
‡Division of Information and Computing Technology (ICT), College of Science and Engineering (CSE), Hamad Bin Khalifa University (HBKU), Doha, Qatar. Email: goligeri@hbku.edu.qa

(DL) techniques. *BloodHound*+ allows to carry out jamming detection by converting raw PHY data, that is, I-Q samples, into images while detecting anomalies in their shape by resorting to autoencoders.

Although our approach can apply to any mobile device, we specifically consider the case of drones operating indoors, e.g., for warehouse inspections [15], inspired by the recent deployment of real-world use cases, such as the one managed by IKEA [9] and Amazon [10]. When applied to autonomous or remotely piloted vehicles and drones, *BloodHound*+ can detect the approach of a jammed area with very low BER values, allowing the remote entity to detect jamming while maintaining complete control of the communication link. To verify the effectiveness of our proposed approach, we conducted an extensive measurement campaign emulating UAVs through Ettus Research X310 and LimeSDR radios, using multiple hardware devices (multiple Ettus Research X310 and LimeSDR radios), communication link configurations, and jamming conditions. Using such measurements, we tested the effectiveness of *BloodHound*+ and other competing approaches for detecting jamming in a low-BER regime. Our results show that *BloodHound*+ can detect jamming in scenarios with a lower BER ($\approx 1e-6$) compared to benchmark solutions, e.g., with an accuracy of 0.997 when the adversary jams at a distance of 10 m from the target with a Relative Jamming Power (RJP) of 0.1. Our solution is also very robust to: (i) the distance from the jammer, (ii) the training set size, (iii) the number of acquired samples, (iv) the sampling ratios at the jammer and the receiver, (v) the type of jamming signal (tone, Gaussian, or deceptive), as well as (vi) the adoption of different jamming hardware and radio types. We envision that *BloodHound*+ can be used during regular operations of the UAV by acquiring physical layer information from the (already existing) UAV communications channel. In addition, it can be activated when desired, so as not to cause significant overhead on the UAV.

This contribution extends and completes our previous work published in [16] by providing the following new content.

- We focus on an indoor scenario, providing a brand new range of data considering additional hardware, modulation techniques, and jamming strategies.
- We consider a stronger adversary model, assuming that the adversary knows the sampling rate and modulation techniques of the legitimate communication link. This knowledge allows the adversary to optimize the parameters of the jamming attack to boost its effectiveness and avoid detection simultaneously.
- We design a new optimized methodology for jamming detection based on a one-class classifier of black and white images extracted from raw I-Q samples using *sparse autoencoders*.
- We experimentally compare our new methodology with those proposed in [16] and [17], showing remarkable performances and improvements concerning the RJP at the receiver, distance from the jammer, number of I-Q samples per image, training set size, and invariance to the hardware used for training.
- We provide additional results on a new dataset gathered using new hardware, namely, the LimeSDR.
- We provide new results on the data collected using various sampling rates at the jammer and receiver.
- We provide new results on using a new jamming strategy, i.e., deceptive jamming using the same modulation (Binary Phase-Shift Keying (BPSK)) and signal of the legitimate communication link.

We acknowledge that *BloodHound*+ takes inspiration from anomaly detection strategies applied in other research domains, e.g., intrusion detection and computer vision [18]. However, to the best of our knowledge, none of the contributions in the current literature provided a structured methodology to apply such strategies to detect jamming attacks on the wireless RF spectrum. Also, as described in more detail in Sect. II, none of the contributions focused on low-BER regimes, investigating how to detect jamming while still maintaining remote communication capabilities.

Roadmap. The rest of this paper is organized as follows. Section III introduces preliminary notions; Section IV describes the scenario and adversarial model; Section V provides the rationale and details of *BloodHound*+; Section VI discusses our extensive measurement campaign and performance assessment of our solution, and finally Section VII draws the conclusion and outlines future work.

II. Related Work

Several scientific papers recently considered drones for indoor applications, focusing on aspects such as localization [37], navigation [38], [39], and visualization [40]. However, none of them investigates jamming attacks and anti-jamming approaches for indoor scenarios, thus mainly referring to the literature on generic (outdoor) jamming detection. In the scientific community, jamming detection is usually achieved by applying various types of analysis on one or more metrics extracted from the primary communication link.

Regarding the metrics, several parameters have been analyzed, such as the Received Signal Strength (RSS) of the signals [19], the PDR as in [20] and [28], the Carrier-to-Noise density power ratio [21], retransmission attempts [26], and [29], the packet re-transmission profile, as in [30], or modulation-specific metrics, such as for Orthogonal Frequency Division Multiplexing (OFDM) in [31]. Such metrics have been used in several scenarios and communication technologies, e.g., Massive MIMO [22], Wireless Sensor Networks (WSN) [23], GPS [27], IEEE 802.11 [28], and spread spectrum-based communication technologies [24], [25].

At the same time, due to the increasing popularity of Artificial Intelligence (AI), ML and DL approaches have been recently used extensively for detecting ongoing jamming. Such tools include Convolutional Neural Networks (NNs)s (CNNs) such as in [32] and [33], genetic algorithm-based Cumulative Sum (CUSUM) methods such as in [34], Bayesian networks such as in [35] and, finally, autoencoders such as in [36]. All such approaches utilize as the main source of information the PHY layer of the communication stack due to its direct

TABLE I: Qualitative comparison of *BloodHound+* with related literature on jamming detection. The symbol ● denotes that a specific feature is supported, the symbol ◐ denotes that the specific feature is partially supported, while the symbol ○ denotes that the feature is not supported.

Ref.	Jamming Detection Metric	Jamming Detection Technique	Robustness to Jamming Distance	Robustness to Jamming Signal Type	Jamming Detection in Low-BER Regime
[19]	RSS	Geometric and Arithmetic Mean ratio	○	○	○
[20]	PDR	Query-based procedure	○	○	○
[21]	Carrier-to-Noise density power ratio	Sum-of-Squares Paradigm	○	○	○
[22]	Coherence blocks	Generalized Likelihood Ratio Test	○	○	○
[23]	RSS	Predetermined knowledge, Error Correcting Codes, and Limited node Wiring	○	○	○
[24], [25]	PDR	Code Tree	○	○	○
[26]	Re-transmissions	Statistics-based	○	○	○
[27]	Automatic Gain Control (AGC)	Static tests	○	●	○
[28]	PDR	Random forests	○	○	○
[29]	PDR	Channel probing	○	○	○
[30]	Retransmisions	Message Invalidation Ratio	○	●	○
[31]	OFDM parameters	Machine Learning (ML)	○	○	○
[32]	Power spectral density, spectrogram, raw constellation	ML	●	○	○
[33]	Spectrogram	ML	●	●	○
[34]	Nonlinear alternating current	CUSUM	●	○	○
[35]	PHY, Radio Link Control and Packet Data Convergence Control parameters	LSTM	○	●	○
[36]	I-Q	Autoencoders	●	○	○
[17]	I-Q	CNN	●	○	◐
[16]	I-Q	CNN	●	○	◐
BloodHound+	I-Q	Sparse Autoencoders	●	●	●

relationship with the wireless channel, where jamming occurs. However, although some of the contributions cited analyzed the performance of the proposed jamming detection technique with low SNR, none of them considered the BER of the communication link. As a result, the proposed approaches mostly confirm that the root cause of the drop in BER is jamming. However, they cannot detect such attacks even when the jamming effect is so low as not to significantly affect the BER of the communication link. As explained above, such consideration is particularly relevant in mobile scenarios for remotely controlled equipment, not to lose control of the mobile entity completely before detecting jamming. In this context, the only contribution to the literature achieving such a property is our previous proposal in [16]. As shown in Section VI, the methodology shown in this paper significantly outperforms both the solution proposed in [16] and the improvements of such a methodology, as the one proposed in [17]. We summarize our comparison with the current literature in Tab. I.

III. Preliminaries

In this section, we introduce preliminary notions that are useful to the readers of this manuscript, i.e., digital modulation techniques (Section III-A) and autoencoders (Section III-B).

A. Digital Modulation

Digital modulation schemes adopted in wireless communication systems preprocess baseband signals to make them suitable for transmission at high frequencies [41]. Typically, modulation techniques divide the bit-stream to be transmitted into two orthogonal components, namely the I vector and the Q vector, linked in a complex value of type $I + jQ$, where the I vector is the real component and the Q vector is the imaginary component. Due to their orthogonality, such components can be transmitted together on the wireless channel without interfering. They can also be recovered and assembled at the receiver to reconstruct the original bit-stream. In this context, a typical way to represent complex I-Q signals is through the I-Q plane, as shown in Fig. 1. In particular, the number of expected I-Q values at the receiver (denoted as n) indicates the number of bits that can be recovered through a single complex I-Q value. In general, we can recover $\log_2 n$ bits from n symbols, and thus, with reference to Fig. 1 showing the I-Q plane of a BPSK modulation, we can recover $n = 2$ bits, i.e., $[i = -1, q = 0]$ (b=0) and $[i = 1, q = 0]$ (b=1). We also notice that the values of the I-Q samples at the receiver always differ from those at the transmitter because of noise introduced by the hardware components of the devices and the

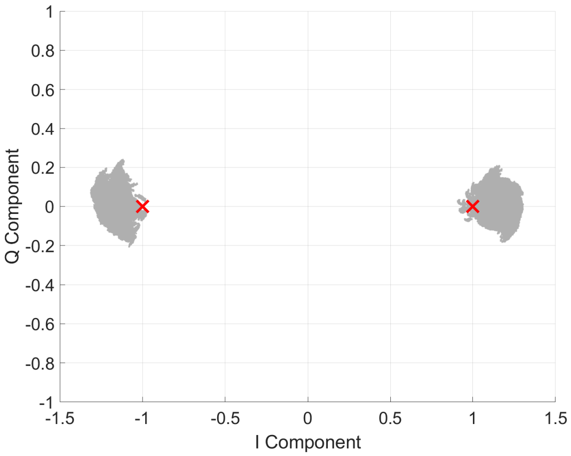


Fig. 1: I-Q plane of a BPSK modulation. The receiver expects two symbols, i.e., $[i=-1, q=0]$ and $[i=1, q=0]$ (red crosses). However, due to the wireless channel, the received symbols are displaced (light grey area).

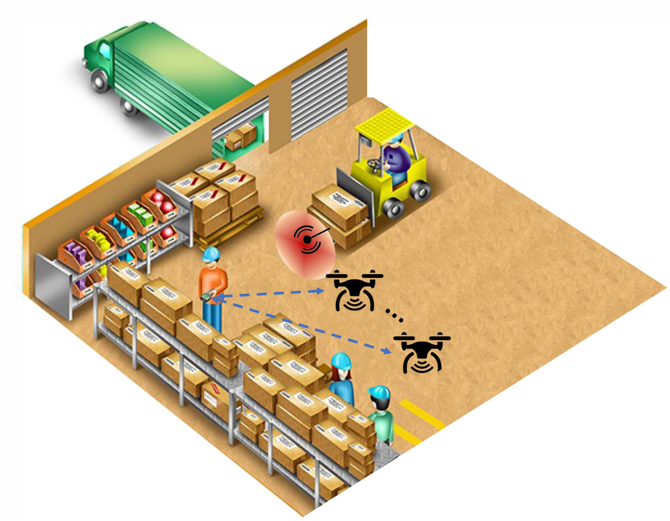


Fig. 2: Reference Scenario. While operating indoors, a drone tries to detect jamming (red color) in a low-BER regime, i.e., before the jamming affects the quality of the legitimate communication link (blue color).

fluctuations of the wireless channel between the transmitter and the receiver. To recover the original transmitted symbol, the receiver associates the expected symbol with the received I-Q sample whose distance from the received one is the shortest. Thus, the higher the noise impact, the higher the chance that a received I-Q sample is associated with the wrong expected symbol, leading to an error and, therefore, a higher BER. At the same time, with a given noise profile affecting the communication channel, the higher the modulation order n, the higher the amount of expected symbols and, thus, the higher the BER. The rationale described above is adopted by lower-order modulation schemes, e.g., communication links affected by high noise levels, such as satellite transmissions and mobile indoor applications. The intuition driving our work is that the collective displacement of I-Q samples from the expected one can be used to discriminate the presence of various levels of intentional interference, i.e., jamming, affecting the communication link. We will provide more details about our approach in Section V.

B. Autoencoders

Without loss of generality, *autoencoders* are a special type of Artificial NN (ANN) which can be trained to reconstruct their input [42]. Formally, the problem autoencoders solve is to find an encoder $A : \mathbb{R}^d \rightarrow \mathbb{R}^p$ and decoder $B : \mathbb{R}^p \rightarrow \mathbb{R}^d$ satisfying Eq. 1.

$$\arg\min_{A,B} E[\Delta(\mathbf{x}, B \circ A(\mathbf{x}))], \tag{1}$$

where the symbol "$\circ$" represents the composition operator, i.e., $B \circ A(\mathbf{x}) = B(A(\mathbf{x}))$, E represents the expectation of the distribution of the input $\mathbf{x}$, '$A(\mathbf{x})$' the encoded version of the input, known as the *bottleneck* of the autoencoder when $p < d$, and finally Δ is the reconstruction loss function, which measures the distance between the input of the ANN and the reconstruction of the input [42]. For our purposes, in line with many scientific contributions such as [42] and [43], Δ is the mean-squared-error (mse) function, as defined in Eq. 2 on two reference distributions $\mathbf{x}$ and $\mathbf{y}$.

$$\text{mse}(\mathbf{x}, \mathbf{y}) = \frac{1}{d} \cdot \|\mathbf{x} - \mathbf{y}\|_2^2 \text{ for all } \mathbf{x}, \mathbf{y} \in \mathbb{R}^d, \tag{2}$$

being $d = M \cdot N$.

Traditionally, autoencoders have been used mainly for image generation, particularly for creating sets of images similar to the input ones. However, in the cybersecurity research domain, they are mainly used for anomaly detection. Let c be an autoencoder trained on samples from a probability distribution P. Next, let Q be a probability distribution such that $P \neq Q$. Then, we expect c to have a smaller reconstruction error when tested on unseen samples from P than when tested on unseen samples from Q. Therefore, the magnitude of the reconstruction error of c in an unseen sample measures the probability that such an unseen sample is not sampled from P but from another distribution [43]. Consequently, we can define a specific error value τ as a classification boundary. All samples in which the auto-encoder c makes a higher error than τ can be classified as 'not from P'. In the literature, τ is often referred to as a *threshold* [44], [45].

In this work, we use autoencoders to build a statistical profile of the channel experienced between the mobile transmitter and the receiver under regular operating conditions. We provide more details in Section V.

IV. System and Adversary Model

Fig. 2 shows the scenario and adversary model considered in this work, inspired by the real-world use case discussed in [9]. We consider an indoor scenario where drones operate to achieve (semi)-automatized tasks, e.g., warehouse inspection or home surveillance. The relevance of such a scenario is confirmed by recent news, reporting that leading companies such as IKEA and Amazon are already experimenting with commercial products for this purpose [9], [10]. We consider the existence of a communication link between the drone and a Ground Control Station (GCS), which can be used either to pilot the drone, in the case of a Remotely-Piloted Aircraft System (RPAS), or to report telemetry data, in the case of (semi) autonomous operations [46]. Independently of the communication link usage, we do not make any assumption on the nature of the drone operations, which can be either instructed by a human pilot or (semi-) autonomous. Also, we do not make assumptions on the presence of a positioning technology, besides that it is not affected by spoofing [47]. Without loss of generality, we consider that the communication link between the drone and the GCS adopts the BPSK modulation scheme. The mentioned assumption is reasonable, as such a scheme allows one to mitigate the noise affecting indoor communication channels as much as possible, being also used in modern WiFi standards.

We also consider the deployment of a static jammer in the area, which injects noise into the wireless channel used for communication between the drone and the GCS. We assume that such a jammer continuously emits interfering signals with the highest possible transmission power, to affect ongoing wireless communications in the deployment area as much as possible. Unlike the contribution in [16], we do not make any assumptions about the specific jamming signal: it can be Additive White Gaussian Noise (AWGN), a single tone, and even deceptive jamming, adopting the same modulation scheme used by the legitimate communication link. Also, being possibly unaware of the sampling rate of the legitimate communication link, the jammer transmits signals with the highest possible sampling ratio, limited only by the hardware used to carry out the attack. Note that when the attacker is unaware of the modulation used by the legitimate communication link, they can perform modulation-agnostic jamming, e.g. by injecting AWGN or a single tone centered on the channel of interest. Instead, suppose that the attacker is aware of the modulation used by the target link. In that case, it can use *deceptive jamming*, i.e., injecting a signal characterized by the same digital modulation (and possibly the same message pattern) of legitimate messages, further complicating the detection process.

Being bounded by the maximum achievable transmission power, the jammer significantly impacts ongoing communications only in a specific area around its location. In fact, the RSS associated with the jamming signal depends on the distance between the jammer and the receiver location (drone), which is highest in the proximity of the jammer and decreases with further movement. The described wireless propagation effect generates a *jammed area*, disrupting wireless communications. In this context, the mobile receiver (drone), moving toward the jammed area, wants to promptly detect the jamming signal in a low-BER regime, i.e., before the effect of the jamming on the quality of the communication link becomes noticeable, causing a significant increase in the BER. In fact, such a jamming detection mechanism would improve drone situational awareness, as it would allow GCS to be aware of imminent jamming and take action immediately without relying on a predefined set of steps (e.g., landing, returning to the starting point).

TABLE II: Notation and brief description.

Notation	Description
n	Number of I-Q samples per image.
M, N	Dimensions of the input image.
$a_{m,n}$	Generic pixel of the input image.
d	Dimension of vectors within autoencoders, with $d = M \cdot N$.
K	Encoder units.
J	Decoder units.
τ	Autoencoder threshold value.
MSE_{train}	mse value obtained at training time.

V. Methodology

This section describes *BloodHound*+, i.e., the methodology we propose to detect jamming in a low-BER regime. In summary, *BloodHound*+transforms jamming detection into an anomaly detection problem on images generated by encoding the current state of the communication channel. Overall, we can identify two main building blocks of our solution: the *Image Generation* and the *Jamming Detection*, described below. Tab. II summarizes the primary notation used below, with a short description.

Image Generation. Fig. 3 provides a graphical overview of the image generation process used in *BloodHound*+. The input to the process is represented by a sequence of raw I-Q samples. Such samples can be collected using a SDR or any hardware capable of obtaining PHY wireless channel information (e.g., spectrum analyzers). The amount of samples used to generate images, namely n, is one of the degrees of freedom of our solution and can be configured by trading off the overall accuracy with the computational requirements of the solution (see Section VI for a detailed evaluation of the impact of this parameter). We represent the sequence of I-Q samples through the traditional I-Q plane, where we display the component I on the x-axis and the component Q on the y-axis. In a benign scenario (no jamming), such a representation generates several clouds of I-Q values, approximately centred on the value of the expected *symbol*, as explained in Section III-A. Based on this representation, we build a bi-variate histogram. Specifically, we divide the I-Q plane around the cloud of points into tiles $N \times M$, where the values of N and M depend on the dimensions of the images we want to obtain. Then, for each tile $a_{m,n}$, we evaluate the number of I-Q samples that fall into the tile itself. When the number of I-Q samples falling on a tile exceeds the value 255, we truncate it to the maximum value, to ensure $a_{m,n} \in [0, 255], \forall (m, n)$. We consider the output of such a process as a pixel value. As a consequence, the output of the image generation process is a grayscale image corresponding to the received profile of I-Q samples. Note that, in principle, we might also work with colored images (3-D matrices). In Section VI, we evaluate this configuration, adopted in [17], and show its pros and cons in physical-layer jamming detection.

Jamming Detection. The jamming detection process is the building block of *BloodHound*+ dedicated to the timely

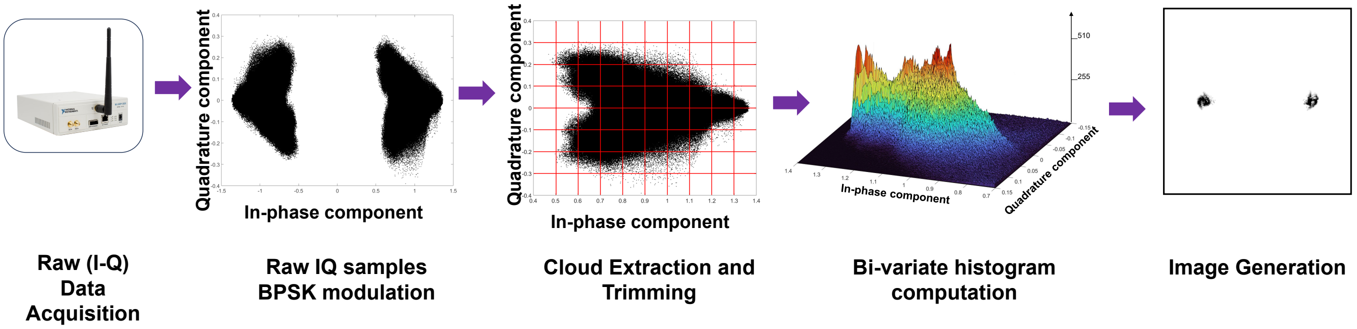


Fig. 3: Graphical overview of the image generation process of *BloodHound*+. We acquire raw I-Q data through a generic Software-Defined Radio (SDR), plot them through the traditional I-Q plane representation, compute a bi-variate histogram based on the density of samples in given areas of the plane, and then obtain an image.

detection of possible jamming affecting the wireless communication channel. It is a DL-based process using *sparse autoencoders*, so it involves a *training* and *testing* process. We highlight that our manuscript does not aim to provide a new autoencoder architecture. Instead, as discussed in the related work (Sect. II) and summarized in Tab. I, the innovation of our manuscript is that *BloodHound*+ is the first solution applying autoencoders to solve a jamming detection problem. Fig. 4 shows the architecture of the adopted *autoencoder*.

The input to the autoencoder is represented by the images obtained as a result of applying the *image generation* process to I-Q samples collected from the wireless channel. At training time, we acquire I-Q samples corresponding to the typical behavior of the communication link. We denote images corresponding to such a scenario configuration as *unjammed images*. We acquire n I-Q samples and generate images of size $M \times N$. In our deployment, $n = 10^5$ and $M = N = 224$, to match the size of images used as input to various NN (we compare our performance to two benchmark solutions using CNN in Section VI). As encoder, we used logarithmic sigmoid functions with $K = 16$ units (a.k.a. neurons). As a result, we obtain a latent representation of the input image consisting of $K = 16$ dimensions. Such a latent representation summarizes the relevant features of the input image, significantly compressing its dimension (compared to the input image). Then, we submit the latent representation vectors to the decoder using a linear decoder transfer function with a total number of $J = 50,176$ neurons. In our context, using the logistic sigmoid activation function in the decoder's units did not allow our solution to converge to a good solution. We suspect *vanishing gradients* to be the cause, further supported by the better convergence we obtain when using linear activation functions in our decoder layer. In principle, autoencoders allow the use of multiple hidden layers. Overall, the higher the number of hidden layers, the better the performance of the classifier at run-time, but also the higher the computational overhead of the methodology. Here, we use two hidden layers and the sparsity regularization technique, in line with the architecture of *sparse nonlinear autoencoders*. As we show in Section VI, such a choice allows us to obtain remarkable classification accuracy while achieving a computational cost lower than that of more complex architectures. This process provides a reconstructed image of the same dimension as the input. We first convert the matrix of size $m \times n$ into a vector of dimension $d = M{\cdot}N$, concatenating the rows of the image one after the other. Then, we compute the mse loss function as in Eq. 2 (see Section III-B). During training, we acquire several images corresponding to the regular (expected) behavior of the wireless channel, building a corresponding profile of such a channel when displayed through images (i.e., our hypothesis). We compute a threshold τ on such a profile, as explained in Section III-B, to distinguish the *regular channel conditions* from the unexpected one. At testing time, we compare the mse value obtained from a run-time acquisition of the wireless channel with the threshold τ previously cited. Suppose that the mse of the input image is equal to or greater than the threshold. In that case, the autoencoder produces a *positive prediction*, meaning that a jammer affects the communication channel. Otherwise, when mse is lower than the threshold, the autoencoder outputs a *negative prediction*, meaning there is no jamming. In line with the logic of any ML and DL solution, the rationale of *BloodHound*+ aligns with statistical hypothesis testing methods. In the following, we provide more details on the autoencoder training process and the threshold selection methodology.

Training the autoencoder. For training the autoencoder, we use only *unjammed images*, i.e., images generated from I-Q samples acquired when no jamming affects the communication link. We do not use any *jammed images*, i.e., images obtained from a jammed communication channel. We applied this strategy mainly since jamming can be performed in many different ways, typically unknown and unpredictable to the legitimate parties. Instead, our intuition is that we can build a profile of the expected conditions of the communication link, even in very noisy scenarios, and detect jamming as a deviation from such expected conditions. In this context, to guarantee reliable operations for the autoencoder, it is crucial to gather I-Q samples that cover the most extensive possible set of expected conditions of the communication channel. In fact, the reliability of the auto-encoder in identifying anomalies leading to jamming depends on the variety of conditions affecting the communication channel, thus reducing false positive events.

Threshold selection. Optimal selection of the decision threshold of an autoencoder is an actively researched problem,

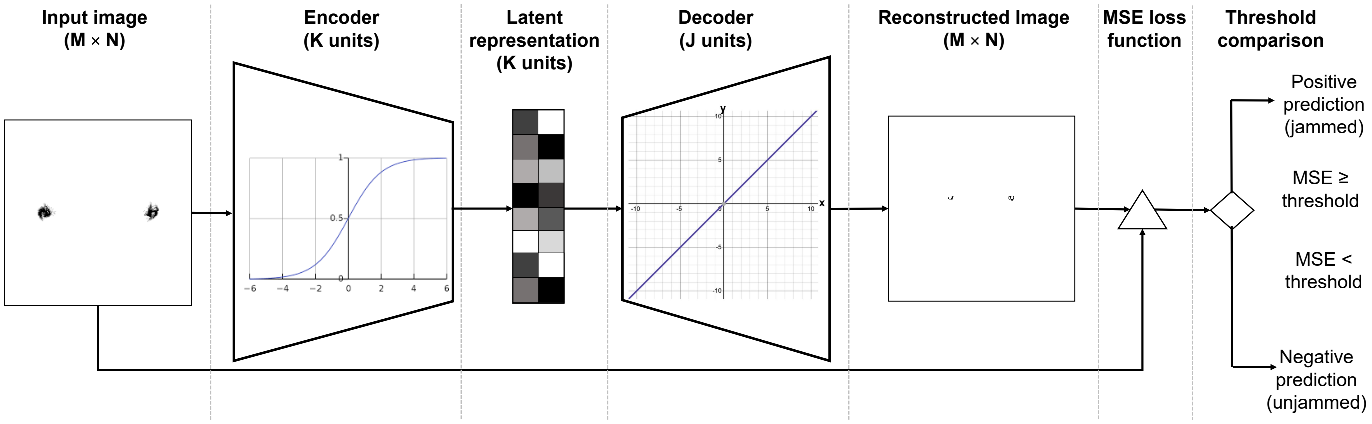


Fig. 4: Autoencoder architecture. We use a *logsig* encoder transfer function and a *purelin* decoder transfer function, with a total number of two hidden layers, a sparsity regularization term of 0.05, and the mse as a loss function, coming up with a *sparse nonlinear autoencoder*.

which has yet to have a universally optimal solution [48], [45]. In this work, we adopt the approach suggested by the authors in [44], i.e., we compute the threshold according to Eq. 3.

$$\tau = mean(MSE_{\text{train}}) + 3.5 \cdot std(MSE_{train}), \qquad (3)$$

being MSE_{train} the set of mses that the autoencoder compute on the training data, $mean$ the statistical average and std the standard deviation. As demonstrated by the authors in [44] and confirmed by the authors in [45], such a choice is reasonable in scenarios where no anomalous samples are used in the training phase, as in our scenario. Furthermore, as acknowledged by the authors in [45], this choice reduces false negatives compared to the standard option, i.e., setting τ to the maximum mse observed in the training samples. In turn, such a choice increases the chances of detecting jamming in a low-BER regime. Figure 5 reports an example of the threshold selection process on actual data acquired with jammer jamming with $RJP = 0.6$ at a distance of 10 meters from the receiver (see Section VI-A for details). We notice that the distribution of the mse values for unjammed images is characterized by smaller values compared to the one of jammed images, with only minimal overlap at the tails of the distributions. Setting the threshold according to Eq. 3 allows one to reduce false negatives without affecting performance.

Finally, note that deploying an autoencoder for image-based jamming detection involves setting optimal values of the hyperparameters of such a tool. We discuss the selection of auto-encoder hyperparameters in Section VI.

VI. Experimental Assessment

In this section, we provide the details of our extensive experimental assessment, carried out to evaluate *BloodHound+* in real indoor scenarios. We introduce the actual measurements used for the following analysis in Section VI-A, while in Section VI-B we describe the experimental settings. Then Section VI-C reports the performance of our approach and compares it with two solutions available in the literature. We extend such results further, evaluating *BloodHound+* with different hardware and various channel sampling rates (Section VI-D) and investigating the capability of identifying deceptive jamming (Section VI-E).

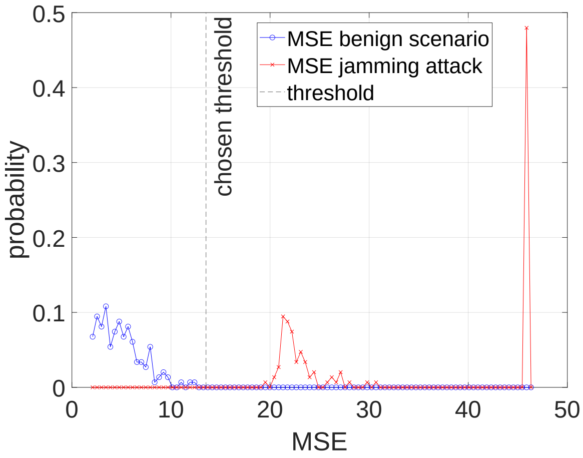


Fig. 5: Sample distribution of the mse of unjammed and jammed images and optimal threshold selection. The reported data refer to a jammer jamming with $RJP = 0.6$ at a distance of 10 meters from the receiver.

A. Measurements

In this paper, we build on top of the data provided as part of our contribution in [49] and extend such a dataset with new measurements obtained with new hardware and different configuration parameters.

All the measurements discussed below have been acquired in an indoor office environment during a regular working day, with people moving around and possibly across the legitimate communication link. Such conditions match, as much as possible, those described in Section IV.

The general setup of our measurements includes three entities, i.e., a transmitter, a receiver, and a jammer. In the first set of experiments, we considered SDR Ettus Research X310 [50] featuring a daughterboard UBX160 as reference hardware for all the entities involved in our measurement campaign. Here, we placed the transmitter and the jammer close to each other while moving the receiver at various distances from the transmitter (see below). For the second set of experiments, we used different hardware, i.e., the LimeSDR [51]. It is a low-cost, open-source, SDR platform supporting any wireless communication standard. For these experiments, we placed the transmitter and receiver 3 meters away from each other and placed the jammer between them at a distance of 1.5 meters from both entities. We connected the SDRs either via Ethernet (Ettus X310) or USB 2.0 (LimeSDR) to two laptops, one controlling the data transmission and jamming processes and the other taking care of data reception. The received I-Q samples were stored on the laptop connected to the receiving SDR and subsequently uploaded to a centralized server for data analysis. Specifically, we used the High-Performance Computing (HPC) cluster available at TU/e, Eindhoven, The Netherlands, providing a CPU E2124 with four cores running at 3.3 GHz and 32 GB of RAM, as well as 2 GPUs Tesla V1000 running at 32 with 256 GB of RAM.

Regarding software, for both setups, we used the GNURadio v3.8 development toolkit [52]. We set the carrier frequency $f_c = 900$ MHz for both the legitimate communication link and the jamming. We configured the transmitter and receiver to exchange packets containing a repeating sequence of 256 bytes, encoded by a *Constellation Modulation* block using the regular BPSK modulation scheme.

For the first setup using the SDR Ettus X310, we configured a sample rate of 1M samples per second at the transmitter, the receiver, and the jammer. We set the normalized transmission power and receiver gain to the maximum value of 1, corresponding to approximately 15 dBm (32 mW) of transmission power. At the receiver, we set up the reception chain of the BPSK modulation, including (i) an *Adaptive Gain Control* (AGC) block, to mitigate the signal level fluctuations introduced by the multipath fading; (ii) a *Symbol Sync*, which performs timing synchronization; (iii) a *Costas Loop*, which locks to the center frequency of the signal and down-converts it to baseband; and, (iv) finally, a *Constellation Decoder* block, which decodes the constellation points. We saved the I-Q data obtained as the output of the *Constellation Decoder* block. We did not use any channel estimation techniques to filter out any channel effects beneficial for jamming detection. Regarding the jammer, we chained two blocks: (i) a signal source, which can be an analog $\sin$ wave (tone jammer) or a digital sequence of Gaussian-distributed values (Gaussian jammer); and (ii) the *USRP Sink block*, which sends the signal to the radio for actual transmission. To emulate the scenario described in Section IV, we placed the entities at different distances and, to further mimic the movement, we changed the relative values of the jammer transmission power between 0 and 0.8, i.e., between 0 and 7.94mW (9dBm), respectively. Values greater than 0.8 cause a complete disruption of the BER of the signal (see Section VI-C), making our solution unnecessary. In fact, when using a static setup, the reduction of the transmission power of the jammer makes the received jamming power level at the receiver weaker, allowing us to investigate the effect of a jammer located further away from the communication link.

For the second setup using the LimeSDR, we considered different sample rate values at the receiver and the jammer. In particular, consider the formula $t_s = K \times t_{S,R}$, where $t_{S,R}$ is the reference sample rate of 1M samples per second, t_s is the actual sample rate used in the measurement, and K an oversampling factor. We carried out experiments considering different values of the oversampling ratio both at the receiver and at the jammer, namely, the *Receiver Oversampling Ratio (ROR)* and *Jamming Oversampling Ratio (JOR)*. Specifically, we varied both the ROR and the JOR in the range $[1, 4]$. For all such experiments, we tested two jamming strategies: tone jamming, i.e., jamming with a sinusoid signal, and deceptive jamming, i.e., jamming with precisely the same bit-sequence BPSK-modulated signal delivered between the legitimate transmitter and the receiver.

In general, the two measurement setups described above allowed us to investigate the effectiveness of *BloodHound*+ while varying an extensive range of configuration parameters, including: (i) the RJP at the receiver, (ii) the distance of the jammer from the legitimate communication link, (iii) the oversampling ratios at the receiver and the jammer, and (iv) the types of jamming and the type of radios used for the experiments.

B. Experimental Settings

For our experiments, we found the best configuration of the hyperparameters of the autoencoder of *BloodHound*+ and compared its performance with the approach based on the binary CNN *Resnet-18* used in [16] and the solution based on 3-D images proposed in [17].

Autoencoder deployment. We fine-tuned the hyperparameters of the autoencoder used in *BloodHound*+to find the best trade-off between classification accuracy and the general validity of the solution, i.e., to avoid overfitting. Specifically, we used the Matlab-provided implementation of autoencoders, version R2022b. As mentioned in Section V, we used the *Pure Linear* (*purelin*) transfer function as the decoder transfer function. We used a formal hyperparameter optimization method performed on a subset of our dataset for all remaining hyperparameters of the autoencoder. Specifically, we selected $1,500$ images from I-Q samples acquired in the scenario with the most data available, i.e., with the receiver positioned 10 meters from the transmitter and the jammer emitting jamming with a RJP of 0.5. We evaluated all combinations of the following hyperparameters: (i) hidden size (i.e., the size of the latent representation), with considered values being 8, 16, 32 and 64; (ii) sparsity regularization, with considered values being 1, 0.5, and 0; (iii) L2-regularization term, with considered values being 0.01, 0.001, and 0.0001; and finally, (iv) encoder transfer function, with considered values being *logsig* and *satlin*. Finally, we set all remaining hyperparameters to their default values provided by Matlab, specified at [53]. Such various configuration parameters led us to test a total number of 72 configurations. For such tests, we used the methodology described in the following, inspired by the k-fold cross-validation technique used by the authors in [54].

- We divide all jammed images in the selected dataset into 10 disjoint subsets of equal size.
- We divide all unjammed images in the selected dataset into evenly sized disjoint subsets of 10.
- For each combination of hyperparameters considered α, for every $i \in \{1, \ldots, 10\}$, we do the following:
  - We train an autoencoder on all subsets of unjammed images, except for the i-th one. During training, we use the hyperparameters in α.
  - We compute the mse values of the autoencoder on the images in the i-th subset of unjammed images.
  - We compute the mse values of the autoencoder on the images in the i-th subset of jammed images.
  - Using the mse values collected as part of the two steps above, we compute the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) curve corresponding to the autoencoder.
- For each combination of hyperparameters α, we take the average of the AUC-values found in the iterations of the last cited step.
- Finally, we pick the combination of hyperparameters for which the last step mentioned above yielded the largest result.

Note that, like the authors in [54], we selected the AUC as the optimization metric since it measures the quality of a classifier independently of the threshold selection process. Following such a hyperparameter selection process, we set the hidden size value to 16, the sparsity regularization term to 0.5, the L2-regularization term to 0.01 and the encoder transfer function to the Logistic Sigmoid (*logsig*). We also selected the number of epochs by examining how many iterations the autoencoder takes to reliably converge in the worst case, i.e., when setting the size of the latent representation to 64. We empirically established that this would occur after 250 epochs, and thus, we selected such a value to trade-off between classification accuracy and training time. We use such a hyperparameter configuration to train the autoencoder of *BloodHound+* for all the results discussed below on unseen data to avoid overfitting.

Benchmark Approaches. For all experiments, we compare *BloodHound+* with two benchmark solutions, i.e., the former version of *BloodHound+* published in [16] and the approach proposed by the authors in [17]. We selected such solutions as they work successfully on PHY data, i.e., I-Q samples, significantly outperforming other solutions in challenging scenarios, such as the one considered in this manuscript. Both approaches use the residual CNN *Resnet-18*, pre-trained on the *ImageNet* dataset [55] and with the necessary modifications to the output layer necessary to fit the nature of the jamming detection problem. Specifically, we modified the output layer of CNN to consider two possible classes as output, i.e., either *No-Jamming* or *Jamming*. Regarding the input layer, CNN works on images of size 224 × 224 constructed over I-Q samples collected from the wireless channel, which is optimal for comparison to *BloodHound+*. To train CNN, we used the automated procedure *trainNetwork* provided by Matlab. We set the batch size to 32, the number of epochs to 1, and the solver to *adam*, similar to [16]. We set all remaining hyperparameters to their default values, available at [56].

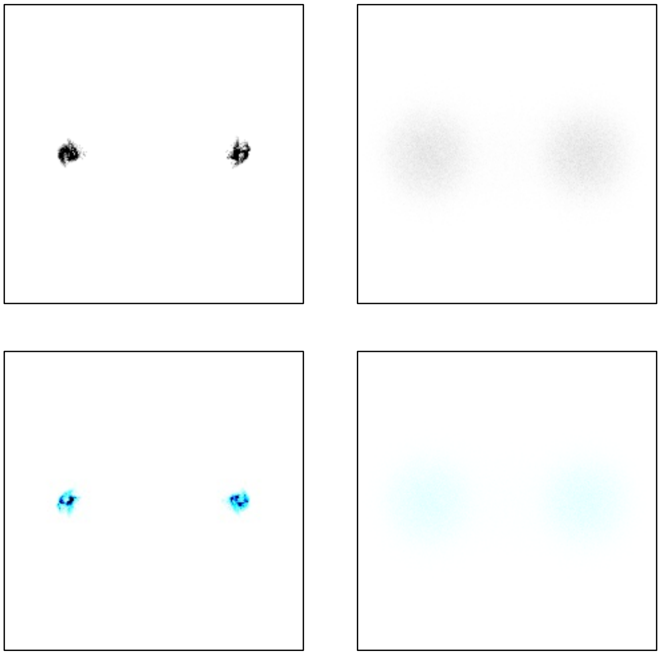


Fig. 6: Sample grayscale and colour *unjammed* (left) and *jammed* (right) images generated from the same set of I-Q samples, used for the experimental assessment. We generated the *jammed* images using samples collected at 10 meters from the jammer, with RJP = 0.5.

The solution provided in [16] works on grayscale images, while the proposal in [17] uses color images characterized by three layers rather than one. We follow the same procedure described in [17] to set up such images. As an example, Fig. 6 compares a grayscale and a color image generated over the same set of I-Q samples collected from the wireless channel. The lower images in Fig. 6 have been generated by considering one layer for each primary colour component (red, green, and blue). Therefore, assuming an image constituted by a three-layer matrix, i.e., [224 × 224 × 3] (one layer for each primary colour), and the pixel value between 0 and 255, in line with [17], we assign each value of the tile through the following rule.

- $0 \leq x_T \leq 255$, then $p_R = 0, p_G = 0, p_B = x_T$,
- $256 \leq x_T \leq 511$, then $p_R = 0, p_G = x_T - 255, p_B = 255$,
- $x_T > 511$, then , then $p_R = x_T - 510, p_G = 255, p_B = 255$,

where x_T represents the value of the tile from the bi-variate histogram, while p_R, p_G and p_B are the pixel values, i.e., red, green and blue, respectively. Finally, we observe that if $x_T > 767$, it is clipped to 767—this issue can also be controlled by properly adjusting the chunk size. Instead, for the upper figures, in line with the logic in [16], we only consider one layer and thus, $0 \leq x_T \leq 255$.

Measurements Characterization. As an introduction to the presentation of our results, with reference to the setup using the USRP X310 SDRs, in Fig. 7, we show the BER of the

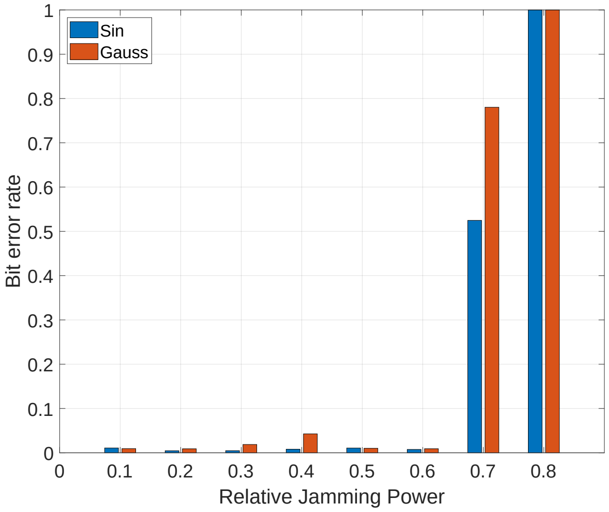


Fig. 7: Analysis of the BER of the TX-RX communication link using the SDRs USRP X310, with various levels of RJP and two jamming signals, i.e., tone jamming and Gaussian jamming. We placed the receiver 10 m away from the jammer.

TX-RX communication link experienced with different values of the RJP, under different jamming signals.

We highlight that when $RJP < 0.7$, the impact of the jammer on the communication link in terms of BER is minimal, i.e., very few bits are corrupted. We achieved the lowest BER value with $RJP = 0.1$, where $BER \approx 1e{-}6$. On the contrary, most bits are corrupted when $RJP \geq 0.7$. Recall that in this manuscript, we are specifically interested in detecting jamming in a mobile scenario in a low-BER regime, i.e., before its impact on the communication link becomes significant and significantly affects the throughput of the communication link. In this context, we are particularly interested in improving the jamming detection performance when $RJP < 0.7$. For higher values of RJP, other techniques based on the analysis of BER can already detect jamming *ex-post*, i.e., once the communication link is significantly affected.

Finally, in this context, note that we configured the LimeSDR setup with an absolute transmission gain at the jammer of 23 dBm. This configuration allows us to have a BER of the legitimate communication link of approximately 0.001, which enables us to study the impact of different configuration parameters of the scenario while matching the conditions of the low-BER regime described above.

Performance Metrics. We compare the performance of the methodologies introduced above mainly concerning their *accuracy*, obtained as $acc = \frac{TP+TN}{TP+FP+FN+TN}$, being TP the number of true positives (i.e., jammed images correctly classified), TN the true negatives (i.e., unjammed images correctly classified), FP the false positives (i.e., unjammed images wrongly classified as jammed ones) and FN the false negatives (i.e., jammed images wrongly classified as unjammed ones). For some of the results shown below, we show the True Positive Ratio (TPR) and True Negative Ratio (TNR), computed as $TPR = \frac{TP}{TP+FP}$ and $TNR = \frac{TN}{TN+FN}$, respectively. We obtain our estimates of accuracy, TPR, and TNR using the cross-validation approach used by the authors of [54], using 10-fold cross-validation. For each result, we report the mean and the 95% confidence intervals, computed using the tool *tinv* provided by Matlab. Tab. III summarizes all the experiments discussed below, together with the relevant parameters.

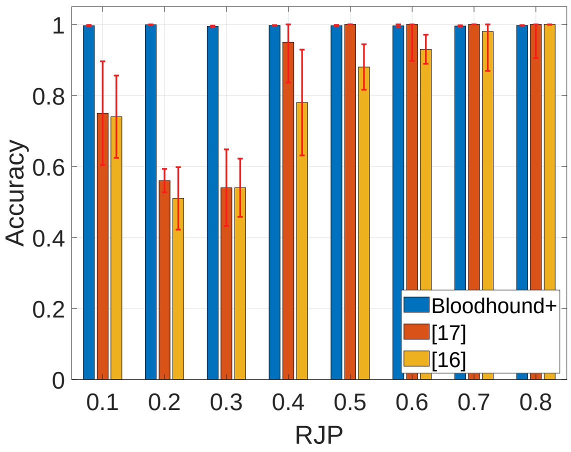


Fig. 8: Classification accuracy of *BloodHound+*, the proposal in [17] and the solution in [16] with various levels of RJP.

C. Jamming Detection Robustness in Low-BER Regime

We first consider the impact of the received jamming power at the receiver on the capability of *BloodHound+* to detect the presence of the jammer. We consider the setup using the Ettus X310 SDR, and precisely, the measurements where we placed the receiver 10 meters away from the jammer. Here, we generate images using $n = 10^5$ I-Q samples and evaluate the performance of *BloodHound+*, the proposal in [17] and the solution in [16] to detect jamming, in terms of overall classification accuracy. We report the results of our investigation in Fig. 8.

First, we note that the approaches in [17] and [16], based on CNNs, reliably identify jamming only when $RJP \geq 0.4$. With lower values of the RJP, their performances do not follow a unique trend, and the results also exhibit high variance (see the red bars indicating the confidence intervals of the measurements). On the contrary, *BloodHound+* reports remarkable performances for every value tested of RJP, with a minimum accuracy of 0.997. We believe such a result is due to the rationale of autoencoders used in *BloodHound+*, which work only on *unjammed images*. This configuration allows autoencoders to build a profile of the regular behavior of the wireless channel to identify more minor differences (anomalies) reliably. We also note that the performances of *BloodHound+* do not depend on the speed of the involved entities nor the smoothness of the change of channel conditions. Indeed, our approach processes chunks of n I-Q samples and compares the

TABLE III: Experiments carried out in the manuscript and related parameters tested.

Techniques	RJP	Distances [m]	Samples per Image	Training Set Size	Jamming Radios	Jamming Oversampling Rates	Jamming Signal Type
BloodHound+	0.1 - 0.8	3, 5, 7, 10, 13, 16, 19, 21	10,000 50,000 100,000 500,000 1,000,000	2, 9, 18, 36, 54, 72	4, 5, 6, 7	1, 2, 3, 4	AWGN, BPSK
[16]	0.1 - 0.8	3, 5, 7, 10, 13, 16, 19, 21	10,000 50,000 100,000 500,000 1,000,000			1, 2, 3, 4	AWGN, BPSK
[17]	0.1 - 0.8	3, 5, 7, 10, 13, 16, 19, 21	10,000 50,000 100,000 500,000 1,000,000				

images created from such samples with the expected channel conditions acquired during training time.

We also investigated the impact of the receiver's distance from the jammer on the performance of the cited solutions. To this end, using the same setup used for the previous tests, we set $RJP = 0.5$ and move the receiver away from the jammer to a distance of 21 meters. We stopped at such a distance due to the physical limitations of the involved hardware, i.e., at distances higher than 21 meters, the BER of the legitimate communication link increases significantly, preventing us from performing the test reliably. We first acknowledge that the receiver experiences increasing interference while moving closer to the jammer. This phenomenon is shown in Fig. 9, highlighting that the process is not linear but follows a polynomial model while increasing distance, i.e., $ax^2 + bx + c$. Note that our result is consistent with the findings of Tedeschi et al. reported in [57]. We also notice that, for specific distance values, significantly different RSS values may be experienced, due to the fast-fading process affecting wireless communication channels. However, recall that lower RSS does not necessarily imply higher BER. As our manuscript focuses on jamming detection in low-BER scenarios, our analysis considers a condition where the BER of the communication channel is low to provide jamming detection before the loss of the communication link. We report in Fig. 10 the results of our experiments involving jamming detection via *BloodHound+* and competing approaches.

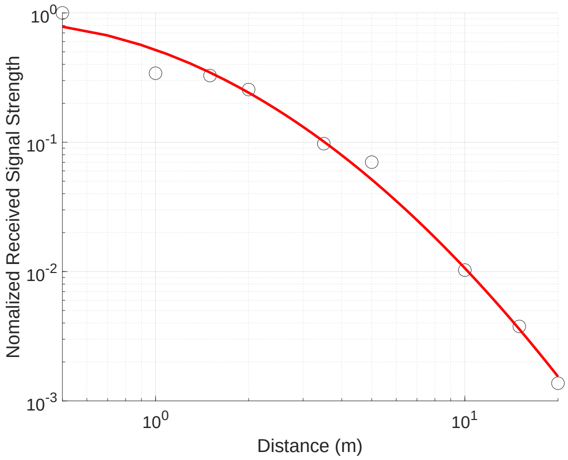


Fig. 9: Normalized Received Signal Strength (NRSS) as a function of the distance: the NRSS decreases when the receiver moves far away from the transmitter. The solid red line shows the model that best fits the experimental data (black circles).

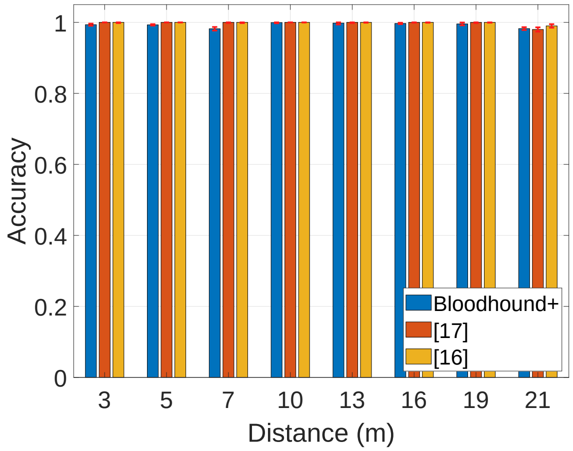


Fig. 10: Classification accuracy of *BloodHound+*, the proposal in [17] and the solution in [16] when positioning the receiver at various distances from the jammer.

Here, we notice that all tested solutions report remarkable performance with minimal differences. All three approaches reliably detect jamming at various distances, with an average accuracy well above 0.99. Overall, the distance between the receiver and the transmitters does not affect the classification accuracy, which remains very high even when the receiver is 21 meters away. We present such a result primarily to show that, similarly to competing approaches, *BloodHound+* can detect jamming even at a significant distance from the jamming source. In addition to such a result, we present additional results below that show the enhanced robustness of *BloodHound+* compared to other approaches.

Another critical parameter of *BloodHound+* is the number of samples used to generate images, namely n. The higher the value of this parameter, the higher the number of samples to use for image generation. Thus, the longer the receiver has

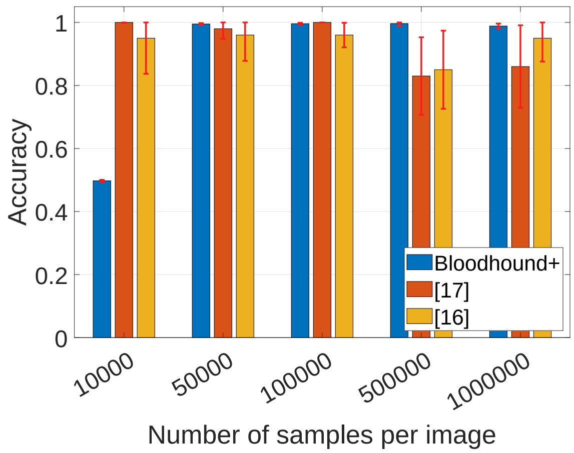


Fig. 11: Classification accuracy of *BloodHound*+, the proposal in [17] and the solution in [16] considering an increasing number of samples N in the *Image Generation* process.

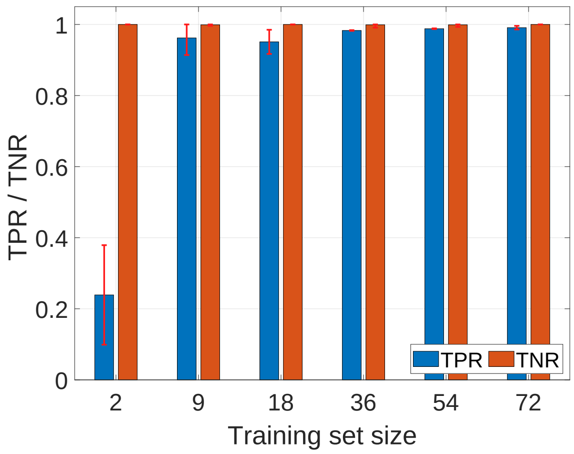


Fig. 12: TPR and TNR of *BloodHound*+ when varying the training set size, i.e., the number of *unjammed* images used for training the autoencoder.

to acquire samples from the wireless channel, the higher the processing overhead of the solution. For this analysis, in the same setup as in the previous experiments, we considered the data acquired with the receiver located 10 m away from the jammer and $RJP = 0.5$, and tested the performance of the three approaches while increasing the number of samples used in the image generation phase, from $10,000$ to $1,000,000$. Fig. 11 reports the results of our analysis.

Note that the solutions in [17] and [16] report a higher classification accuracy than *BloodHound*+ for a low number of samples. For example, when the $n = 10,000$, *BloodHound*+ reports an accuracy of 0.498 while such values amount to 0.99 and 0.95 for the solution in [17] and [16], respectively. When the available number of samples increases, the accuracy of the benchmark approaches is still high, but the variance becomes larger (see the red bars). On the contrary, when $n \geq 50,000$, not only the accuracy of *BloodHound*+ is very high (always higher than 0.99), but the variance is also minimal (less than 0.001 for all tests), indicating greater robustness and reliability. Such results further motivate the deployment of *BloodHound*+ and highlight its superiority compared to the benchmark solutions.

To provide further insight into the performance of *BloodHound*+, we investigated the impact of additional configuration parameters. In particular, in the same setup as the last cited test, considering $n = 100,000$, we evaluated the effect of the training set size. Figure 12 summarizes the results of our investigation, distinguishing the achieved TPR and TNR of our proposed solution.

To perform reliably, *BloodHound*+ requires a minimum training set of only 9 images, reporting TPR and TNR values of 0.962 and 0.99, respectively. Performance remains almost constant when increasing the training set size. However, increasing the training set could be particularly relevant in very noisy scenarios characterized by a broader range of *expected* wireless channel fluctuations. We recall that we need to train *BloodHound*+ only once before deployment, and such results do not affect the deployability of our solution at runtime.

We also investigated further any bias of our results concerning the specific hardware used for the experiments. Taking into account the same scenario as in previous experiments, i.e., the receiver located 10 meters away from the jammer, we evaluated the TPR and TNR of *BloodHound*+ when changing the hardware used for jamming among the five available radios. We highlight that this methodology prevents the autoencoder from fingerprinting both the transmitter and the receiver, these being the same for all measurement classes. Moreover, we considered different hardware for the jammer during our measurements, i.e., we mutually excluded the ones adopted for training from the ones adopted for testing. The mentioned strategy eventually guarantees that the autoencoder learns the characteristics of the legitimate signal only while being independent of the transmitter, the receiver, and the jammer hardware (we do not use jammed signals for training). Specifically, we consider all unjammed images obtained when placing the receiver at a distance of 10 meters from the transmitter, using the radio x as the jammer. Next, we consider all the jammed images generated with the receiver located at a distance of 10 meters and use radio y as a jammer. We separated the unjammed and jammed images into 10 folds, and we trained *BloodHound*+ on 9 of the folds containing unjammed images, holding the i-th one to estimate the TNR. Next, we estimate the TNR on the i-th fold of the unjammed images and evaluate the TPR on the i-th fold of the jammed images. Fig. 13 reports the result of our analysis. The tick labels on the x-axis are of the form (x, y), as described above.

Note that the TPR and TNR remain almost unchanged while varying the hardware used for jamming and the radio considered for training. Therefore, we can safely assume that *BloodHound*+ is not biased by the specific radios used in the experiments. Still, it can extract the features of the wireless

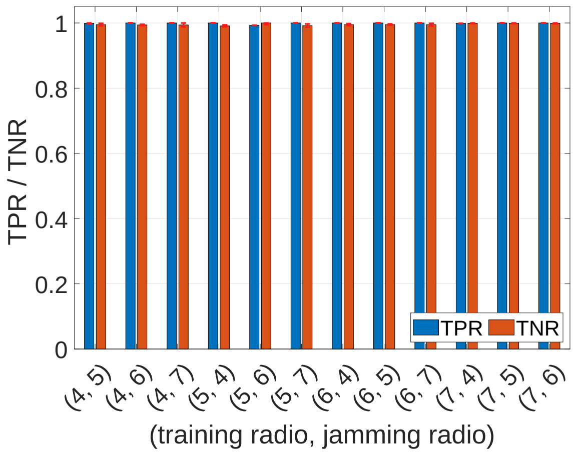


Fig. 13: TPR and TNR of *BloodHound+* when varying the hardware used for jamming and the radio used for training.

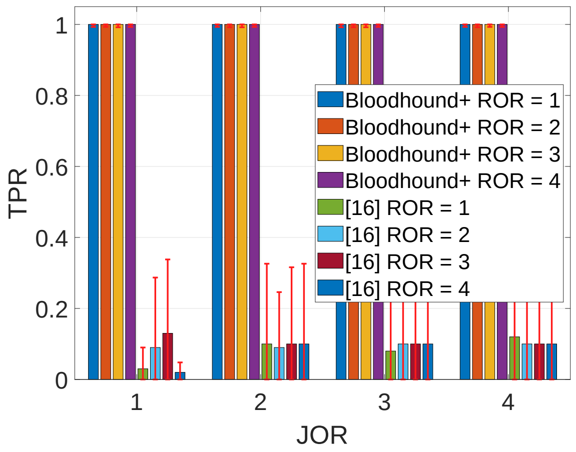


Fig. 14: True Positive Ratio (TPR) of *BloodHound+* and the proposal in [16], with various ROR and JOR values, when the jammer injects Gaussian noise (Gaussian random jamming).

channel useful for detecting jamming independently of the particular hardware.

D. Impact of Different Hardware and Sampling Rate

We obtained all the results shown in the previous subsection using a single hardware brand (Ettus Research USRP X310) and a single configuration of the sample rate, i.e., 1 Msa/s. To further extend our experimental assessment, we first analyzed the data collected through the second setup described in Section VI-A, adopting the hardware LimeSDR Mini. An important consideration about the scenario described in Section IV is that the jammer might not know the adopted sampling rate in advance. Thus, to disrupt ongoing communications as much as possible, in real-life scenarios, the jammer might emit jamming using the maximum achievable sampling ratio, likely higher than the one adopted by the legitimate communication link. At the same time, the receiver might oversample the signal, obtaining more helpful information for jamming detection. This additional information might be discarded for communication but might benefit jamming detection. Therefore, in our tests, we trained *BloodHound+* on unjammed images of the legitimate communication link obtained under $RJP = 0.5$ and a distance of 3 meters. Then, we tested using unjammed images (disjoint from those used for training) and jammed images obtained from I-Q samples generated with various levels of JOR. For this test, we let the jammer emit random Gaussian noise. We also compared the performance of *BloodHound+* with the solution in [16]. Fig. 14 summarizes the results of our analysis in terms of TPR (for comparison purposes). Note that the TNR of *BloodHound+* was also excellent, at an average of 0.987 over all values of JOR.

We can distinguish two effects from the reported results. Considering the configuration with $ROR = 1$ and $JOR = 1$, we notice that the solution in [16] already reports very low TPR (0.03). We highlight that this result is not related to different oversampling rates but to the new hardware used for the experiments. Indeed, the LimeSDR is a cheaper hardware, which introduces additional inaccuracies and imperfections in the I-Q samples received. Such inaccuracies affect the shape of the I-Q samples, which is now more spread around the expected symbol than before, leading to jammed images different from the ones in the training set of the solution in [16]. The approach in [16] does not catch these variations, thus failing to perform reliable jamming detection. On the contrary, in such a configuration, the performance of *BloodHound+* does not change compared to the results shown in Section VI-C, demonstrating once again the enhanced robustness offered by the autoencoders used in *BloodHound+* compared to the CNNs used in the competing solution. With higher ROR and JOR values, the performance of the approach in [16] remains well below 0.2, confirming the unsuitability of such a solution for detecting jammers in the wild. Instead, *BloodHound+* can mitigate and overcome the impact on the wireless channel of different oversampling ratios, being able to detect jamming also when the JOR is very high.

E. Deceptive Jamming

In previous experiments, we mainly considered two jamming models, that is, tone jamming (using a sinusoid signal) and random jamming (using AWGN). When the adversary does not know the modulation used by the legitimate communication link, the jamming models mentioned and investigated above are the most reasonable options to disrupt the channel as much as possible. However, more powerful attackers might know in advance or become aware at runtime (e.g., by eavesdropping) of the modulation used by the legitimate communication link. Based on this knowledge, they might use the signal as part of a jamming attack optimized to disrupt the communication link as much as possible. Recall that the legitimate communication link in our experiments adopts the BPSK modulation scheme. In such a scenario, the usage of a jamming signal modulated also as a BPSK allows to boost

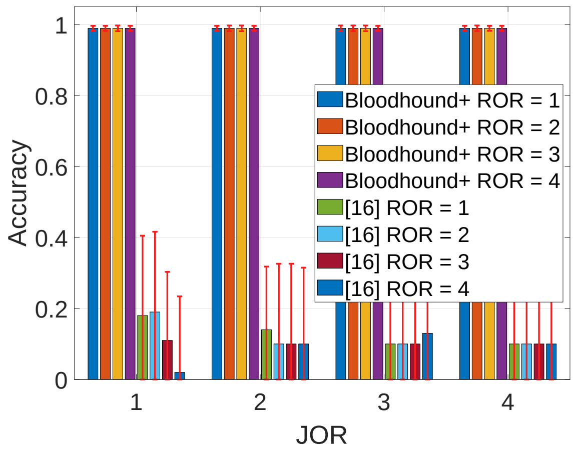


Fig. 15: True Positive Ratio (TPR) of *BloodHound+* and the proposal in [16], with various ROR and JOR values, when the jammer injects the same signal from the legitimate communication link (deceptive jamming).

the effectiveness of the jamming activity, as acknowledged by several scientific contributions both analytically [58] and experimentally [28].

To this end, using the same setup as the previous experiments in Section VI-D, we evaluated the ability of both *BloodHound+* and the solution in [16] to detect *deceptive jamming*, i.e., a jammer injecting the same signal used as part of the legitimate communication link. Fig. 15 summarizes the results of our analysis in terms of TPR (for comparison purposes). Note that also in this case, the TNR of *BloodHound+* is 0.987, on average over all JOR, since we trained the autoencoder in *BloodHound+* with the same data as in Fig. 14.

Even when the adversary uses deceptive jamming, *BloodHound+* significantly outperforms the benchmark solution in all tested configurations, showing perfect TPR and being robust to adopting a high JOR by the adversary.

Overall, the results reported above demonstrate the superiority of *BloodHound+* compared to benchmark approaches and the robustness of our solution to a wide range of configuration parameters and scenarios, making it the preferred solution for jamming detection in a low-BER regime.

F. Considerations on Interference

Various factors, such as interference and obstacles, might affect the performance of *BloodHound+*. We show in Fig. 16 the effect on the I-Q samples of people passing through our experimental setup. We notice that the shape of the I-Q samples at the receiver is significantly affected compared to a LOS scenario. Also, compared to jamming, note that these phenomena are much quicker, and the shape of the I-Q samples returns to the regular shape much faster than under jamming attacks. Thus, to allow *BloodHound+* not to declare jamming in such cases, we need to experience such phenomena during the training phase to make them part of the regular conditions of the communication channel, especially indoors. These considerations motivate the deployment of our solution and the experiments in an office environment during working hours: we experience several instances of such phenomena during training, and they contribute to creating an expected profile of unjammed scenarios which takes interferences into account. During jamming attacks, the shape of the I-Q samples is displaced from the expected profile for a much extended time, contributing to enhancing the performance of our solution.

VII. Conclusion

In this paper, we have presented *BloodHound+*, an approach that allows drones and possibly other mobile devices to detect jamming at the physical layer (PHY) of the communication stack. Our solution works on raw I-Q samples extracted from the communication link, converts them into grayscale images, and uses *sparse autoencoders* to detect discrepancies with the expected profile of the channel. Therefore, *BloodHound+* can detect jamming in low-BER regimes, i.e., well before its effect could cause a significant decrease in the quality of the main communication link. At the same time, our solution allows drones to efficiently avoid the jammed area and maintain complete control and safety. To test the effectiveness of our solution, we conducted an extensive measurement campaign, acquiring real-world data with different hardware, jamming strategies, and scenario configurations. We also tested the performance of *BloodHound+* depending on various parameters, such as SNR of the communication link, the distance from the jammer and the transmitter, the size of the training set, the number of samples acquired from the channel, the jammer oversampling ratio, and the jamming strategies. Our experimental assessment demonstrates, through an extensive collection of results, the superiority of our solution compared to the current state-of-the-art across all the analyzed configuration parameters. In general, our solution contributes to taking a step further toward the safe and secure integration of drones into daily life. As part of our future work, we plan to investigate further the effectiveness of *BloodHound+* e.g., when applied for outdoor applications.

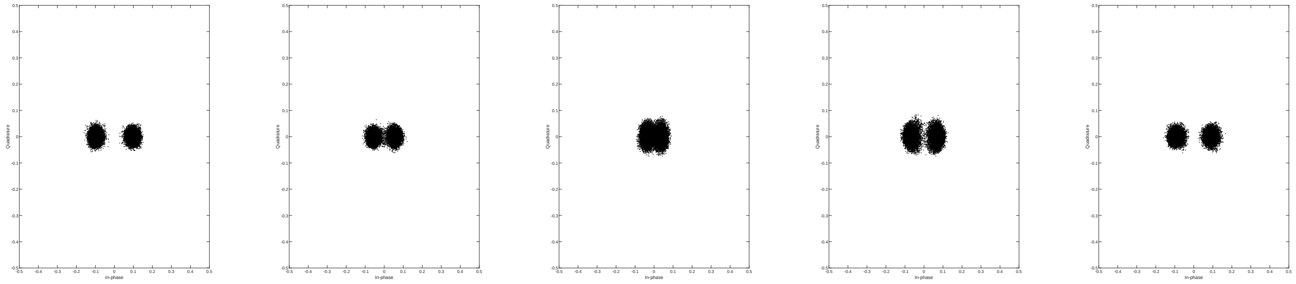
Acknowledgements

This work has been partially supported by the INTERSECT project, Grant No. NWA.1162.18.301, funded by the Netherlands Organization for Scientific Research (NWO). Any opinions, findings, conclusions, or recommendations expressed in this work are those of the author(s) and do not necessarily reflect the views of NWO. Moreover, this publication was made possible by the NPRP12C-0814-190012-SP165 awards from the Qatar National Research Fund (a member of Qatar Foundation).

References

[1] IBM, “What is the internet of things?” https://www.ibm.com/topics/internet-of-things, (Accessed: 2023-Dec-12).

[2] Oracle, “What is IoT?” https://www.oracle.com/internet-of-things/what-is-iot/, (Accessed: 2023-Dec-12).



(a) t = 0 (b) t = 1 (c) t = 2 (d) t = 3 (e) t = 4

Fig. 16: The effect of multipath on the transmitter-receiver link across five time windows. Compared to an ideal (static) scenario (Figs. (a) and (e)), the shape of the I-Q samples at the receiver (modulated through the BPSK scheme) is significantly affected (see Figs. (b), (c), and (d)) when an event is happening, e.g., moving objects.

[3] H. Genc, Y. Zu, T. Chin, M. Halpern, and V. Reddi, "Flying IoT: Toward Low-Power Vision in the Sky," *IEEE Micro*, vol. 37, no. 06, pp. 40–51, Nov 2017.

[4] E. Wisse, P. Tedeschi, S. Sciancalepore, and R. Di Pietro, "A^{2}RID-Anonymous Direct Authentication and Remote Identification of Commercial Drones," *IEEE Internet of Things Journal*, 2023.

[5] A. Yazdinejad, R. M. Parizi, A. Dehghantanha, H. Karimipour, G. Srivastava, and M. Aledhari, "Enabling drones in the internet of things with decentralized blockchain-based security," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6406–6415, 2020.

[6] M. A. Hoque, M. Hossain, S. Noor, S. R. Islam, and R. Hasan, "IoTaaS: Drone-based Internet of Things as a service framework for smart cities," *IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 12 425–12 439, 2021.

[7] L. Abualigah, A. Diabat, P. Sumari, and A. H. Gandomi, "Applications, Deployments, and Integration of Internet of Drones (IoD): A Review," *IEEE Sensors J.*, vol. 21, no. 22, pp. 25 532–25 546, 2021.

[8] L. Wawrla, O. Maghazei, and T. Netland, "Applications of Drones in Warehouse Operations," *Whitepaper. ETH Zurich, D-MTEC*, p. 212, 2019.

[9] The Verge, "Ikea adds stock-counting drones to more of its stores," https://www.theverge.com/2023/3/20/23648156/ikea-verity-drones-stock-counting-stores, (Accessed: 2023-Dec-12).

[10] J. Schneider, "Amazon Wants You To Test its Ring Flying Indoor Drone Camera," https://petapixel.com/2021/09/28/amazon-wants-you-to-test-its-ring-flying-indoor-drone-camera/, (Accessed: 2023-Dec-12).

[11] Global Indoor Robots Market – Industry Trends and Forecast to 2029. (Accessed: 2023-Sep-13). [Online]. Available: https://www.databridgemarketresearch.com/reports/global-indoor-robots-market

[12] T. Multerer, A. Ganis, U. Prechtel, E. Miralles, A. Meusling, J. Mietzner, M. Vossiek, M. Loghi, and V. Ziegler, "Low-cost jamming system against small drones using a 3D MIMO radar based tracking," in *European radar conference (EURAD)*. IEEE, 2017, pp. 299–302.

[13] H. Pirayesh and H. Zeng, "Jamming Attacks and Anti-Jamming Strategies in Wireless Networks: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, 2022.

[14] R. Ferreira, J. Gaspar, P. Sebastião, and N. Souto, "A Software Defined Radio Based Anti-UAV Mobile System with Jamming and Spoofing Capabilities," *Sensors*, vol. 22, no. 4, p. 1487, 2022.

[15] S. Chambers, "A.P. Moller Holding invests in drone inventory tracking solution," https://splash247.com/a-p-moller-holding-invests-in-drone-inventory-tracking-solution, (Accessed: 2023-Dec-12).

[16] S. Alhazbi, S. Sciancalepore, and G. Oligeri, "BloodHound: Early Detection and Identification of Jamming at the PHY-layer," in *IEEE Consumer Communications & Networking Conference (CCNC2023)*, 2023.

[17] S. Alhazbi, S. Sciancalepore, and G. Oligeri, "The Day-After-Tomorrow: On the Performance of Radio Fingerprinting over Time," in *2023 Annual Computer Security Applications Conference (ACSAC)*, 2023.

[18] D. Gong, L. Liu, V. Le, B. Saha, M. R. Mansour, S. Venkatesh, and A. v. d. Hengel, "Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1705–1714.

[19] S. Saxena, A. Pandey, and S. Kumar, "RSS based multistage statistical method for attack detection and localization in IoT networks," *Pervasive and Mobile Computing*, vol. 85, p. 101648, 2022.

[20] M. Çakıroğlu and A. T. Özcerit, "Jamming detection mechanisms for wireless sensor networks," in *International ICST Conference on Scalable Information Systems*, 2010.

[21] D. Borio and C. Gioia, "Real-time jamming detection using the sum-of-squares paradigm," in *International Conference on Localization and GNSS (ICL-GNSS)*. IEEE, 2015, pp. 1–6.

[22] H. Akhlaghpasand, S. M. Razavizadeh, E. Björnson, and T. T. Do, "Jamming detection in massive MIMO systems," *IEEE Wireless Communications Letters*, vol. 7, no. 2, pp. 242–245, 2017.

[23] M. Strasser, B. Danev, and S. Čapkun, "Detection of reactive jamming in sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 7, no. 2, pp. 1–29, 2010.

[24] J. T. Chiang and Y.-C. Hu, "Cross-layer jamming detection and mitigation in wireless broadcast networks," in *Proc. of the 13th annual ACM international conference on Mobile computing and networking*, 2007, pp. 346–349.

[25] J. T. Chiang and Y.-C. Hu, "Cross-Layer Jamming Detection and Mitigation in Wireless Broadcast Networks," *IEEE/ACM Transactions on Networking*, vol. 19, no. 1, pp. 286–298, 2011.

[26] A. Marttinen, A. M. Wyglinski, and R. Jäntti, "Statistics-based jamming detection algorithm for jamming attacks against tactical MANETs," in *2014 IEEE Military Communications Conference*. IEEE, 2014, pp. 501–506.

[27] E. Axell, F. M. Eklöf, P. Johansson, M. Alexandersson, and D. M. Akos, "Jamming detection in GNSS receivers: Performance evaluation of field trials," *NAVIGATION: Journal of the Institute of Navigation*, vol. 62, no. 1, pp. 73–82, 2015.

[28] O. Puñal, I. Aktaş, C.-J. Schnelke, G. Abidin, K. Wehrle, and J. Gross, "Machine learning-based jamming detection for IEEE 802.11: Design and experimental evaluation," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*. IEEE, 2014, pp. 1–10.

[29] D. Liu, et al., "Efficient and Timely Jamming Detection in Wireless Sensor Networks," in *IEEE Int. Conf. on Mob. Ad-Hoc and Sensor Systs.*, 2012, pp. 335–343.

[30] Z. Lu, et al., "Modeling, Evaluation and Detection of Jamming Attacks in Time-Critical Wireless Applications," *IEEE Trans. on Mob. Comput.*, vol. 13, no. 8, pp. 1746–1759, 2014.

[31] J. Pawlak, Y. Li, J. Price, M. Wright, K. Al Shamaileh, Q. Niyaz, and V. Devabhaktuni, "A Machine Learning Approach for Detecting and Classifying Jamming Attacks Against OFDM-Based UAVs," in *Proceedings of the 3rd ACM Workshop on Wireless Security and Machine Learning*, ser. WiseML '21, 2021, p. 1–6.

[32] C Swinney, et al., "GNSS Jamming Classification via CNN, Transfer Learning & the Novel Concatenation of Signal Representations," in *IEEE Int. Conf. on Cyber Situat. Awaren., Data Analytics and Assessm.*, 2021, pp. 1–9.

[33] Y. Li, J. Pawlak, J. Price, K. Al Shamaileh, Q. Niyaz, S. Paheding, and V. Devabhaktuni, "Jamming Detection and Classification in OFDM-

Based UAVs via Feature- and Spectrogram-Tailored Machine Learning," *IEEE Access*, vol. 10, pp. 16 859–16 870, 2022.

[34] K.-D. Lu and Z.-G. Wu, "Genetic Algorithm-Based Cumulative Sum Method for Jamming Attack Detection of Cyber-Physical Power Systems," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–10, 2022.

[35] Y. Wang, S. Jere, S. Banerjee, L. Liu, S. Shetty, and S. Dayekh, "Anonymous Jamming Detection in 5G with Bayesian Network Model Based Inference Analysis," in *IEEE 23rd International Conference on High Performance Switching and Routing (HPSR)*, 2022, pp. 151–156.

[36] H. Bouzabia, T. N. Do, and G. Kaddoum, "Deep Learning-Enabled Deceptive Jammer Detection for Low Probability of Intercept Communications," *IEEE Systems Journal*, pp. 1–12, 2022.

[37] A. Famili, A. Stavrou, H. Wang, and J.-M. J. Park, "PILOT: High-Precision Indoor Localization for Autonomous Drones," *IEEE Transactions on Vehicular Technology*, pp. 1–15, 2022.

[38] G. Raja, S. Suresh, S. Anbalagan, A. Ganapathisubramaniyan, and N. Kumar, "PFIN: An efficient particle filter-based indoor navigation framework for UAVs," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 5, pp. 4984–4992, 2021.

[39] S. Jung, S. Hwang, H. Shin, and D. H. Shim, "Perception, Guidance, and Navigation for Indoor Autonomous Drone Racing using Deep Learning," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2539–2544, 2018.

[40] X. Gao, L. Zhu, H. Cui, Z. Hu, H. Liu, and S. Shen, "Complete and Accurate Indoor Scene Capturing and Reconstruction Using a Drone and a Robot," *IEEE Sensors Journal*, vol. 21, no. 10, pp. 11 858–11 869, 2020.

[41] T. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed. USA: Prentice Hall PTR, 2001.

[42] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *arXiv preprint arXiv:2003.05991*, 2021. [Online]. Available: https://arxiv.org/pdf/2003.05991.pdf

[43] G. Oligeri, S. Sciancalepore, S. Raponi, and R. Di Pietro, "PAST-AI: Physical-layer authentication of satellite transmitters via deep learning," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 274–289, 2022.

[44] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning," *Pattern Recognition*, vol. 58, pp. 121–134, 2016.

[45] S. S. Khan and B. Taati, "Detecting unseen falls from wearable devices using channel-wise ensemble of autoencoders," *Expert Systems with Applications*, vol. 87, pp. 280–290, 2017.

[46] P. Tedeschi, S. Sciancalepore, and R. Di Pietro, "Lightweight Privacy-Preserving Proximity Discovery for Remotely-Controlled Drones," in *Proc. of Annual Computer Security Applications Conference*, 2023, pp. 178–189.

[47] S. Raponi, S. Sciancalepore, G. Oligeri, and R. Di Pietro, "Road Traffic Poisoning of Navigation Apps: Threats and Countermeasures," *IEEE Security & Privacy*, vol. 20, no. 3, pp. 71–79, 2021.

[48] H. Torabi, S. L. Mirtaheri, and S. Greco, "Practical autoencoder based anomaly detection by using vector reconstruction error," *Cybersecurity*, vol. 6, no. 1, p. 1, 2023.

[49] S. Alhazbi, S. Sciancalepore, and G. Oligeri, "A Dataset of physical-layer measurements in indoor wireless jamming scenarios," *Data in Brief*, vol. 46, p. 108773, 2023.

[50] Ettus Research, "USRP X310," https://www.ettus.com/all-products/x310-kit/, 2020, (Accessed: 2023-Dec-12).

[51] LimeSDR Microsystems, "LimeSDR," https://limemicro.com/products/boards/limesdr/, 2020, (Accessed: 2023-Dec-12).

[52] T. J. O'shea and N. West, "Radio Machine Learning Dataset Generation with GNU Radio," in *Proceedings of the GNU Radio Conference*, vol. 1, no. 1, 2016.

[53] Matlab R2023a documentation, "TrainAutoencoder," https://www.mathworks.com/help/deeplearning/ref/trainautoencoder.html, (Accessed: 2023-Dec-12).

[54] N. Shvetsova, B. Bakker, I. Fedulova, H. Schulz, and D. V. Dylov, "Anomaly detection in medical imaging with deep perceptual autoencoders," *IEEE Access*, vol. 9, pp. 118 571–118 583, 2021.

[55] O. Russakovsky et al., "ImageNet Large Scale Visual Recognition Challenge," *Int. Jour. of Comp. Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[56] Matlab R2023a documentation, "TrainNetwork," https://www.mathworks.com/help/deeplearning/ref/trainnetwork.html, (Accessed: 2023-Dec-12).

[57] P. Tedeschi, S. Sciancalepore, and R. Di Pietro, "Modelling a Communication Channel under Jamming: Experimental Model and Applications," in *IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*. IEEE, 2021, pp. 1562–1573.

[58] S. Amuru and R. M. Buehrer, "Optimal Jamming Against Digital Modulation," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 10, pp. 2212–2224, 2015.