BloodHound: Early Detection and Identification of Jamming at the PHY-layer

Saeif Alhazbi*, Savio Sciancalepore†, Gabriele Oligeri*
*Division of Information and Computing Technology (ICT)
College of Science and Engineering (CSE), Hamad Bin Khalifa University (HBKU), Doha, Qatar
{salhazbi, goligeri}@hbku.edu.qa
†Eindhoven University of Technology, Eindhoven, Netherlands
s.sciancalepore@tue.nl

***Abstract*— Traditional jamming detection techniques, adopted in static networks, require the receiver (under jamming) to infer the presence of the jammer by measuring the effects of the jamming activity (packet loss and received signal strength), thus resulting only in a-posteriori analysis. However, in mobile scenarios, receivers (e.g., drones, vehicles, etc.) typically experience an increasing jamming effect while moving toward the jamming source. This phenomenon allows, in principle, an *early* detection of the jamming activity—being the communication not yet affected by the jamming (no packet loss). Under such an assumption, the mobile receiver can take an informed decision before losing the radio connection with the other party.**

To the best of our knowledge, this paper represents the first attempt toward the detection of a jammer before the radio link is fully affected by its activity. The proposed solution, namely, BloodHound, can early detect the approach to a jammer in a mobile scenario, i.e., before losing the capability of communicating, thus enhancing situational awareness and robustness. We performed an extensive measurement campaign, and we proved our solution to be able to detect the presence of a jammer with an accuracy higher than 0.99 even when the bit error rate is less than 0.01 (early detection), by varying several configuration parameters of the scenario.

***Index Terms*—Jamming Detection; Machine Learning for Security; Mobile Security.**

I. Introduction

Jamming is nowadays still one of the most relevant threats affecting the availability of wireless networks [1]. By simply injecting high-power signals into the wireless communication channel, an adversary can disrupt any wireless links, preventing all Radio Frequency (RF) communications to occur in a specific geographical area [2]. Today, the jamming threat is particularly relevant especially in the context of mobile Remotely-Piloted Aircraft Systems (RPASs) and autonomous vehicles, relying on wireless communications for controlling the vehicle or receiving (video) feeds from the environment where the entities are located. When subject to jamming, such entities lose the remote control, and have to rely on a local logic to become available again [3]. Before any action can be triggered as a countermeasure, the target entity has to detect jamming [4]. Then, the most effective countermeasure to the specific type of jamming attack can be selected, if any, or the target can leverage a set of pre-defined actions to mitigate the attack. Indeed, the better the jammer characterization in terms of time, frequency, and nature of the injected interference, the more effective the countermeasure [5]. Several jamming detection strategies for wireless networks have been proposed in the literature (see Sec. VII for an overview). Most of them rely on the analysis of metrics related to the reliability of the traffic in the network, such as the bit error rate (ber), throughput, and Packet Delivery Ratio (PDR), and the quality of the communication link, such as the Received Signal Strength (RSS) and the Signal-to-Noise Ratio (SNR). When the reliability of the communication link decreases (increasing ber and decreasing PDR) together with the quality of the link (low SNR), high chances are that the link is subject to jamming. Although the remarkable detection accuracy such solutions can achieve, the above-mentioned strategies can detect jamming only *ex-post*, i.e., once the jamming has already taken down the communication link (or significantly affected it). As a result, at the jamming detection time, the target node cannot reliably communicate further, and it is required to autonomously take an action.

To close this gap, we propose *BloodHound*, a new solution for early jamming detection and identification in a mobile scenario, based on the analysis of the physical (PHY) layer of the communication link. *BloodHound* detects in advance (i.e., earlier) the jamming of the communication link by analyzing the raw I-Q samples of the signal acquired at the PHY layer, and evaluating the shifting from the expected profile through Deep Learning (DL) techniques. Moreover, *BloodHound* can also identify the type of signal injected by the jammer, among the traditional *Gaussian* noise and *tone-jamming*. Our investigation reports outstanding classification performances, higher than 0.99 in almost all the analyzed scenarios. At the same time, *BloodHound* requires limited acquisition time (500 msec at 1 M samples per second) and

negligible processing time. The rest of this paper is organized as follows. Sec. II describes the scenario and the problem considered in this work, Sec. III describes our measurement setup, Sec. IV provides the measurements characterization, Sec. V shows the details of the used classification tool, Sec. VI investigates the effectiveness of our solution with different configurations, Sec. VII reviews related work, and finally, Sec. VIII tightens the conclusions.

II. Scenario and Problem Description

In this section, we introduce the system and adversary models assumed in this work (Sec. II-A), as well as the necessary background on the PHY layer and I-Q samples (Sec. II-B).

A. Scenario and Adversary Model

We consider a mobile entity, such as a drone or a connected car, moving around to accomplish a specific task, e.g., goods delivery or surveillance, to name a few. While moving, the entity is communicating wirelessly with a control station, either for receiving commands or for uploading information acquired at run-time, such as video-streaming or sensors readings. An adversarial jammer is deployed in a static location, to prevent any communications in a given area of interest. We consider a jammer able to transmit powerful radio signals and cause interference at any communication frequency in the radio spectrum, including the one(s) used by the mobile entity to communicate with the ground control station. Moreover, the jammer broadcasts its jamming signal continuously, at the maximum available power. Being bounded by the transmission power, the jammer significantly impacts ongoing communications only in a specific geographical area around its location. Indeed, the RSS associated with the jamming signal depends on the distance between the jammer and the receiver location, being maximum in the proximity of the jammer and decreasing while moving farther. The described wireless propagation effect generates a *jammed area*, where any communication is disrupted. In this context, the mobile receiver, moving toward the jammed area, wants to promptly detect the approaching of the jammed area, before the effect of the jamming on the quality of the communication link would be noticeable. Indeed, such an early jamming detection mechanism would enhance the situational awareness of the mobile entity, as it would allow the ground station to be aware of the upcoming jamming and to take action immediately, without relying on a pre-defined set of actions.

B. I-Q samples

Modulation schemes adopted in digital communication systems allow base-band information to be converted to a format suitable for transmission at any frequency in the RF spectrum (high-frequency). In this context, the *I-Q representation* is the most adopted one, because of its spectrum efficiency, low cost, and ease of design; indeed, it is often realized in hardware through inexpensive System on Chips (SoCs) [6]. Figure 1 shows a general diagram of an I-Q modulator,

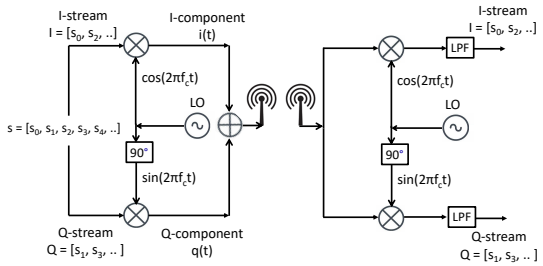

Fig. 1: I-Q (de)-modulation. The symbol-stream s is divided into two streams, I and Q, which are modulated by two orthogonal signals (phase-shifted by 90 degrees). At the receiver, dual operations are performed to recover the original symbol stream.

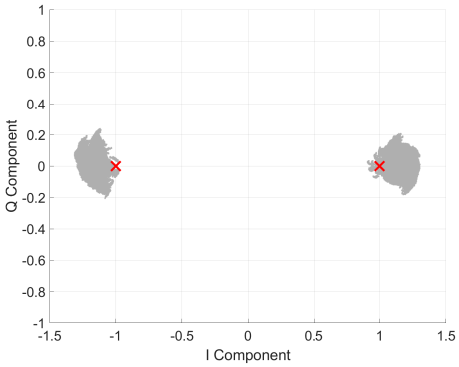

Fig. 2: Real BPSK constellation. Symbols can take two possible ideal values: $[i = -1, q = 0]$ and $[i = 1, q = 0]$ (red crosses). However, radio propagation introduces a shift that moves them from the ideal position (light grey area).

including a local oscillator (LO) working at the reference frequency f_c (carrier) and two mixers, used to modulate the *I-component* and *Q-component* of the signal. Subsequently, the two signals ($i(t)$ and $q(t)$) are combined and transmitted onto the radio spectrum. Since the two components are phase-shifted by 90 degrees, they do not mutually interfere and can be decoded correctly by an I-Q de-modulator (right side of Fig. 1). The I-Q symbols displacement (constellation) and the information carried by such symbols define the specific modulation format. The Binary Phase Shift Keying (BPSK) is one of the modulation formats that is most used in practice, especially when dealing with long-range RF communications, featuring two I-Q symbols, as shown in Fig. 2. The BPSK modulation is widely used nowadays in many applications, including satellite and low-range communications, requiring high reliability. Overall, the higher the number of expected symbols in the I-Q plane, the higher the amount of information that can be delivered, but also the lower the robustness to noise. Indeed, radio communications are subject to noise that, in turn, affects the I-Q symbols by shifting their expected position (defined at the transmitter). Such a shift can be caused by

several factors, such as background noise, other simultaneous interfering data transmissions, or a jammer (as in the case study of this paper).

Our intuition is that the shifting affecting the I-Q samples (and the associated patterns) can be used to detect and identify the presence of a malicious interference (jammer). As a toy example, let us consider real I-Q samples associated with the symbol $[1, 0]$, i.e., the right symbol in Fig. 2. Figure 3 shows the "clouds" of I-Q samples associated with the symbol $[1, 0]$ grouped over time ($5 \cdot 10^5$ I-Q samples per image), considering different scenarios: No-Jamming (only regular background radio noise), jamming performed with a *sin* wave (tone-jamming), and finally, jamming via a Gaussian distributed pattern (*Gauss*) signal. The image indexes (#Img) refer to the measurement timeline, assuming an overall acquisition duration of 600 seconds, equivalent to a total of 290 images. Note that the clouds have similar shapes over time when considering the same jamming scenario, i.e., No-Jam, *sin* or *gauss*. At the same time, they are very different from each other. In the remainder of this paper, we show how to leverage a standard image detection technique based on Convolutions Neural Network (CNN) to detect the presence of a malicious interference (jammer) and characterize its nature (*sin* or *gauss*). We will also test our solution against several configuration parameters.

III. Measurement Setup

In this section, we introduce the hardware (Sec. III-A) and the software setup (Sec. III-B) of our measurement campaign.

A. Hardware Setup

Our general setup includes three radios, i.e., a receiver and two transmitters, taking the role of the actual data transmitter and the jammer, respectively. The reference hardware adopted in this work is the Software-Defined Radio (SDR) USRP Ettus Research X310 [7], featuring the UBX160 daughterboard. The SDRs are connected via Ethernet to two Laptops Dell XPS15 9560, one responsible for controlling the jamming and data transmission, and the other one being in charge of data reception. The laptops are equipped with 32GB of RAM and an Intel 7th Gen Core i5-7300HQ processor, running at 2.80 GHz. As for the antenna, we resort to the VERT2450 omni-directional stylo antenna, provided by the same vendor. It is worth noting that the transmitter and the jammer are close to each other (specifically, one on top of the other) during all the measurements considered in this work, while the location of the receiver varies according to the specific measurement conditions. Moreover, we highlight that all the measurements have been taken in an office environment during working hours, thus implying people moving close to both the transmitters and the receiver. Note that, with reference to the scenario described in Sec. II, the indoor environment is characterized by more challenging channel conditions compared to the outdoor one, due to the presence of stronger multipath fading and missing line-of-sight affecting the signal propagation. Thus, our results can be conceived as the ones in the worst-case scenario. Finally, our deployment is characterized by the absence of the Line-of-Sight (NLoS) between the transmitters and the receivers for all the measurements: the parties were separated by one or more thin walls, further complicating jamming detection.

B. Data collection and parsing

Our software platform is constituted of the well-known GNURadio development toolkit. We set the reference frequency $f_c = 900$ MHz for both communication and jamming, while we considered a repeating sequence of 256 bytes to be transmitted before being encoded by a *Constellation Modulation* block, by using a regular BPSK modulation scheme. We considered a sample rate of 1M samples per second at both the transmitter, receiver and the jammer (bandwidth of 1MHz). The normalized transmission power and receiver gain have been set to 1—this being equal to about 15dBm (32mW) for the transmission power given our hardware set-up. The receiver chain is constituted of the following blocks: (i) an *Adaptive Gain Control* (AGC) block, to mitigate the signal level fluctuations due to 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. No channel estimation techniques have been considered in this work. The data is collected at the receiver side after the *Constellation Decoder* block. The jammer includes 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), in line with the constant jamming signals mostly adopted in real-life; and (ii) the *USRP Sink block*, which sends the signal to the radio for the actual transmission. To emulate the scenario described in Sec. II, we placed the transmitter, the receivers, and the jammer at different distances and, to emulate the movement further, we changed the relative values of the jammer's transmission power between 0 and 0.8, i.e., between 0 and 7.94mW (9dBm), respectively. Indeed, with a static setup, reducing the transmission power of the jammer makes the received jamming power level weaker at the receiver, allowing us to investigate the effect of a jammer located farther away from the communication link. Finally, the dataset is publicly available here [8].

IV. Measurements characterization

In this section, we characterize our measurements in terms of ber, SNR, *Amplitude*, and *Phase*, showing that such metrics cannot detect jamming in the analyzed scenarios.

Bit-error-rate (ber). We consider the reference measurement scenario constituted of three radios: a transmitter, a receiver and a jammer. We recall that the transmitter and the jammer are deployed one on top of the other, while the receiver is located approx. 10 meters far away from both of them. The ber analysis has been carried out by comparing the position of the I-Q samples in the I-Q plane and the value of the transmitted bit, as previously described in Sec. II-B. While we

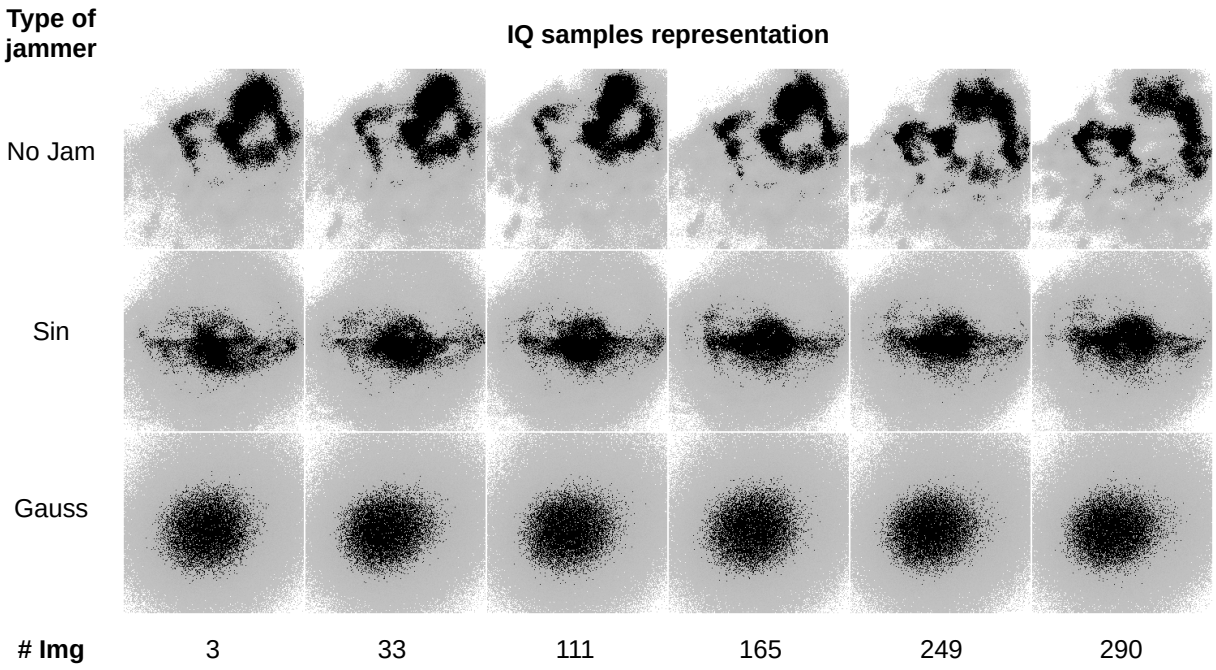

Fig. 3: I-Q samples representation. We consider 6 images taken from three measurements (one for each jamming behaviour) of 600 seconds constituted of 290 images ($5 \cdot 10^5$ I-Q samples per image). Images are organized by type of jammer (No Jamming, *sin*, and *gauss*), and by time, where we considered images 3, 33, 111, 165, 249, and 290, respectively.

set the transmission power and receiver gain to 1, we varied the relative jamming power from 0.1 to 0.8, and we report the bit error rate as a function of the two types of jammers considered in this work (*sin* and *gauss*). Figure 4 shows the mean ber, as a function of the adopted relative jamming power. We observe that the jamming signal does not affect the communication when the relative jamming power is less than 0.6. Conversely, the ber significantly increases when the jamming power is 0.7 and 0.8—it is worth mentioning that, in the latter case, all the received bits are corrupted. Finally, we observe small ber values for all the jamming powers. We can explain them by recalling the scenario where the measurement has been taken. Indeed, during the data collection, people were moving around close to the radios, causing unpredictable perturbations.

Signal-to-Noise ratio (SNR). Various link quality metrics, such as the SNR, have been considered by previous contributions to infer the presence of artificial (and malicious) noise sources [9], [10], [11]. We consider the same deployment previously described, i.e., three (3) radios, constituted of one transmitter, one jammer (at the same position of the transmitter), and one receiver about 10 meters far away from the previous ones. We also assumed three different jamming behaviours, i.e., silent (No Jamming), *sin*, and *gauss*. Figure 5 shows the probability density function associated with the SNR, where the SNR has been computed as in Eq. 1.

$$SNR\Big|_{dB} = P_{rx}\Big|_{dBm} - N\Big|_{dBm}. \quad (1)$$

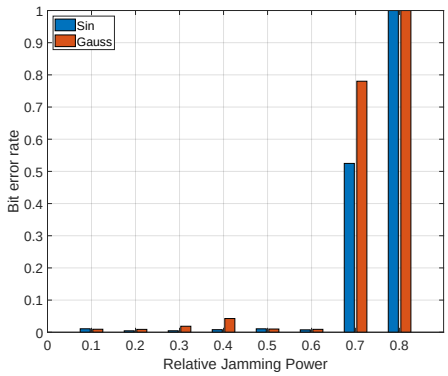

Fig. 4: Bit error rate (ber) as a function of the relative jamming power and the types of jamming, i.e., *sin*, and *gauss*.

where P_{rx} and N are the RSS and the power associated with the noise, respectively, computed from the I-Q samples as in Eq. 3.

$$P_{rx}\Big|_{dBm} = 30 + 10\log_{10}(I^2 + Q^2), \quad (2)$$

$$N\Big|_{dBm} = 30 + 10\log_{10} E\left[\left(\sqrt{I^2 + Q^2} - \mu\right)^2\right]. \quad (3)$$

where $\sqrt{E\left[\left(\sqrt{I^2 + Q^2} - \mu\right)^2\right]}$ is the standard deviation computed over a window of 10 samples of I and Q values, while μ is the mean value computed over the same set.

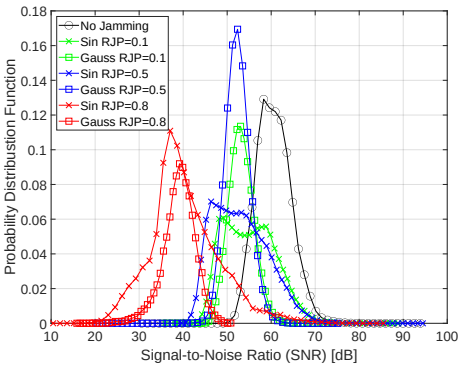

Fig. 5: Signal-to-Noise ratio (SNR) associated with different jamming behaviour, i.e., No Jamming, *sin*, and *gauss*, and varying the relative jamming power between 0.1 and 0.8.

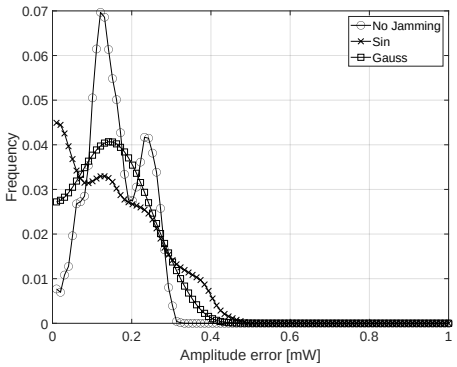

Fig. 6: Amplitude error analysis associated with the three different jammer behaviour, i.e., No Jamming, *sin*, and *gauss*. Note that the three curves are overlapping, making this statistic not suitable for jammer detection and classification.

Figure 5 shows the SNR (dB) assuming the different jamming behaviours, while varying the relative jamming power (RJP), i.e., $RJP = 0.1$ (green solid line), $RJP = 0.5$ (blue solid line), and finally, $RJP = 0.8$ (red solid line). It is worth noting that the presence of the jammer (blue, red, and green lines) decreases the SNR compared to a regular communication link affected by environmental radio noise (black line). Moreover, we observe that the application of SNR-based analysis strategies makes the jamming discrimination task very challenging. Indeed, the *sin* and *gauss* curves are overlapping for all the considered jamming powers, being indistinguishable. In the following, we consider two additional statistics associated with the received signal, i.e., the *amplitude error* and the *phase error*. We specifically focus on the scenario where the receiver is deployed 10 meters from the transmitter and the jammer, and the relative jamming power has been fixed to 0.5.

Amplitude error analysis. We define the amplitude error α_ϵ as in Eq. 4.

$$\alpha_\epsilon = \left| 1 - \sqrt{I^2 + Q^2} \right|. \tag{4}$$

Figure 6 shows the frequency of the amplitude errors computed on the three jamming behaviours, i.e., no-jamming (circles), *sin* (crosses), and *gauss* (squares). Note that detecting the jammer from the error amplitude might be challenging. Indeed, the three curves are overlapping, making identification very hard. Finally, we highlight that the error amplitude has been computed only for the I-Q samples associated with *correct* bits, i.e., bits whose values have not been complemented, while we did not take into account statistics associated with corrupted bits.

Phase error analysis. We define the phase error ϕ_ϵ as in Eq. 5.

$$\phi_\epsilon = \begin{cases} \arctan\left(\frac{I}{Q}\right), & \text{if } I > 0 \\ \pi - \arctan\left(\frac{I}{Q}\right), & \text{otherwise} \end{cases} \tag{5}$$

Figure 7 reports the phase error analysis, considering the jammer behaviours of No Jamming, *sin* and *gauss*. As in the previous case, we consider only the I-Q samples associated with the bits correctly received, i.e., we are excluding the bits that are received as corrupted (whose value has been complemented). The phase error analysis for the No Jamming case highlights very small errors associated with the I-Q samples, i.e., $\phi_\epsilon < |0.1|$ for most cases. The presence of the jammer significantly increases the phase error up to $\phi_\epsilon < 0.3$, although the two jammers cannot be distinguished by just looking at this statistic.

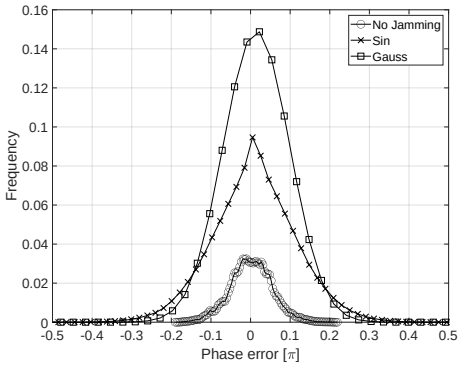

Fig. 7: Phase error analysis associated with the three different jammer behaviour, i.e., No Jamming, *sin* and *gauss*. Note that the three curves are overlapping, making this statistic not suitable for jammer detection and classification.

V. Classification of I-Q samples via CNN

In this section, we address the problem of classifying I-Q samples to infer the presence of a malicious transmitter (jammer). Driven by the intuition presented in Sec. II-B, we set up a two-stage process consisting of: (i) generating images from I-Q samples; and (ii) classifying the generated images to detect the presence of the jammer. Finally, the full process has been implemented in Matlab 2021b.

Image generation. To convert an arbitrarily large sequence of I-Q samples to images, we consider the *bivariate histogram*. Specifically, we parted the I-Q plane in a sequence of adjacent tiles, and we counted how many I-Q samples belong to the same tile. The output of this process is finally considered as a pixel value, comprised between 0 and 255. The tiling process (number of rows and columns) should be properly chosen according to the input size required by the image classification algorithm. The images have been generated considering the clouds associated with the two symbols $[1, 0]$ and $[-1, 0]$ independently, and removing the quantiles 1 and 99 associated with both the I and Q components—this is to perform preliminary filtering of inconsistent I-Q samples (outliers). Finally, note that some of the tiles might collect more than 255 samples, based on the predefined number of samples per image and the tiling process. In the afore-cited case, we removed from the count the exceeding I-Q samples, setting the maximum value to 255.

Images classification. To classify the images, in line with other solutions based on transfer learning [12], [13], we resort to a standard image classification algorithm leveraging Convolutional Neural Networks (CNNs), based on a Residual Network with 18 layers (*ResNet-18*) pre-trained on the *ImageNet* dataset [14]. The input images needed for *ResNet-18* should have a size $[224 \times 224]$, thus biasing the previously discussed tiling process. Moreover, the default *ResNet-18* design provides an output of 1000 classes—this one being the number of classes of the *ImageNet* dataset. To adapt the CNN to our scenario, we changed the output classes to three (3), since we require to distinguish between No-Jamming, Sin, and Gauss jamming behaviours. Finally, we selected empirically a training process characterized by 30 epochs, a validation frequency of 5 iterations, and a mini-batch size of 512 elements. It is worth noting that the choice of ResNet is only one of the many possible options, to show that CNNs can effectively detect jamming earlier, thus demanding more research in this field.

VI. Jamming detection and Identification

In this section, we investigate the performance of *BloodHound*, i.e., our methodology to detect and identify the presence of a jammer. We first analyze a reference scenario, constituted of a transmitter and a jammer deployed one on top of the other and a receiver, located far away from them. For each measurement session, we considered three configurations of the jammer behaviour: No Jamming, *sin*, and *gauss*. Finally, to validate our methodology, we varied several configuration parameters, as described in the following.

Relative Jamming Power. We considered the generic scenario where the receiver is located 10 meters away from both the transmitter and the jammer (the later ones close to each other), and we varied the relative jamming power from 0.1 to 0.8. For each considered value of the relative jamming power, we tested two different SDR jamming hardware, by training on the first one and testing on the second one—this is to guarantee that there are no biases due to the specific jamming hardware, thus excluding the possibility of physical layer fingerprinting of the radio transmitter. We run each measurement session for 600 seconds, collecting about 146M samples. By considering $5 \cdot 10^5$ samples per image, we generated a total number of 293 images for each of the three classes, i.e., No Jamming, *sin*, and *gauss*. The training and validation processes have been carried out by randomly shuffling the datasets previously obtained, by splitting the original one into 60% and 40% for training and validation, respectively. We stress that for the testing phase we adopted a brand new dataset, collected using a different jammer while keeping the same transmitter and receiver. Figure 8 shows the classification accuracy of *BloodHound*, varying the relative jamming power. We observe that the accuracy of the detection algorithm is maximum when the relative jamming power is above 0.4. Indeed, for values less than 0.4, the effect of the jamming on the signal is so low that it does not significantly affect the shape of the I-Q samples, preventing early classification. At the same time, when the relative jamming power overcomes 0.4, the I-Q samples distribution is significantly affected by the jammer, thus enabling a successful detection—although the bit error rate is almost zero (recall Fig. 4). The I-Q shifting is even more noticeable when the relative jamming power is higher (greater than 0.6). Indeed, the resulting signal (the result of the superposition of the transmission, jamming, and radio noise) is so degraded that it might not be possible to decode and receive it, resulting in large fluctuations in performance. The red dashed line in Fig. 8 highlights the interpolation of the median values in the interval $[0.1 - 0.5]$. Finally, we observe that the maximum classification performance is achieved in the range between 0.4 and 0.6, where the bit error rate is almost zero (recall Fig. 4). Such a finding confirms the feasibility of detecting the jammer well in advance compared to traditional methods, even when the bit error rate is not affected. For the sake of completeness, Fig. 9 reports the confusion matrix associated with the performance when the jamming power is set to 0.5. We stress that the training/validation processes and the testing operation have been carried out on different

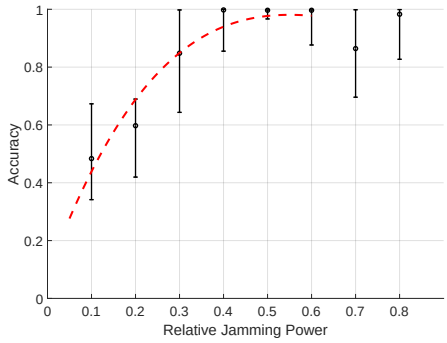

Fig. 8: Classification accuracy as a function of the relative jamming power, with 95% confidence interval.

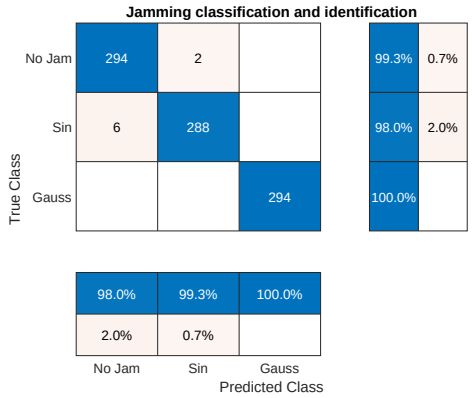

Fig. 9: Testing confusion matrix: Jamming detection and identification assuming a relative jamming power value of 0.5.

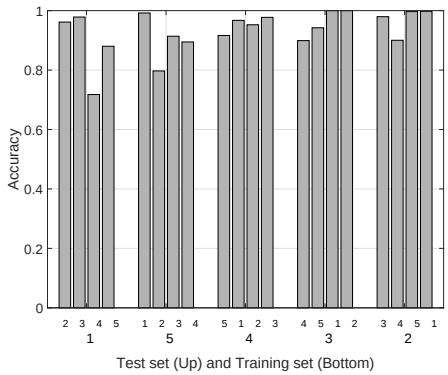

Fig. 10: Classification accuracy as a function of the hardware. The upper x-labels report the indexes associated with the SDR used for testing, while the lower ones report the SDRs used for training.

datasets, generated using different jammers (different SDRs). At the same time, we recall that, in the same scenario, it is not possible to either detect or identify the type of jammer by using traditional solutions based on link-quality parameters (recall Section IV).

Different hardware (SDRs). We consider the reference scenario of a receiver located 10 meters away from both the transmitter and the jammer (close to each other). In particular, we adopted five (5) different radios as jamming stations, while we kept the same radio hardware for the transmitter and the receiver. This methodology prevents the neural network to fingerprint either the transmitter or the receiver—these ones being the same for all the 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. This eventually guarantees that the CNN learns the features of the sum of the over-the-air signal and the jammer (if present) while being independent of the transmitter, the receiver and the jammer hardware.

Moreover, as a reference scenario, we set the relative jamming power to 0.5 (best performance from Fig. 8), while keeping the other parameters the same as before: measurement duration 600 seconds, $5 \cdot 10^5$ I-Q samples per image, turning out into more than 290 images for each of the three classes. For each classification process, we trained and validated on a dataset (split as 60/40) generated from a specific receiver station, and subsequently, we adopted the remaining four 4 measurements (from the other radios) for the testing process. Figure 10 shows the overall classification accuracy of *BloodHound*, as a function of all the possible combinations we considered. The lower labels in the x-axis refer to the radio receiver adopted in the training process, while the upper labels refer to the radio used for the testing. The results show very high accuracy (over 0.9), except for two isolated cases. Such cases should not be brought back (exclusively) to the adopted hardware combination, but also the (uncontrollable) background radio noise and the multipath fading process—we

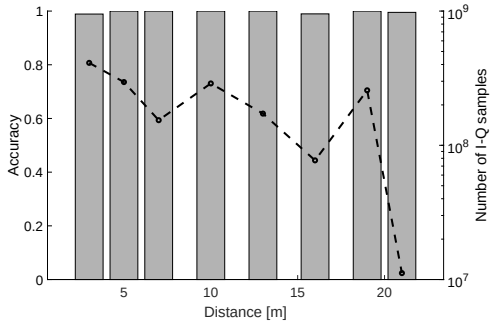

Fig. 11: Classification accuracy (left y-axis) and number of available I-Q samples (right y-axis), increasing the distance between the receiver and the transmitters. We stop our measurements at 21 meters since the number of I-Q samples (correctly-received bits) was significantly lower.

recall that all the measurements have been taken in an office environment, with people walking around all the time.

Distance. We consider once again our reference scenario constituted of three radios, including one receiver and two transmitters, i.e., the actual data transmitter and the jammer. We varied the distance between the receiver and the transmitters, between 3 and 21 meters. Moreover, we set the relative jamming power to 0.5, and we collected measurements of 600 seconds, considering $5 \cdot 10^5$ I-Q samples per image, turning out into more than 290 images for each of the three classes. For each distance, we split the collected data into three subsets, i.e., 60%, 20%, and 20% of the original one, representing the training, validation, and testing set, respectively. Finally, we did not consider distances larger than 21 meters due to logistics issues although they will be taken into account in future works. Figure 11 shows the classification accuracy of *BloodHound*, varying the distance between the receiver and

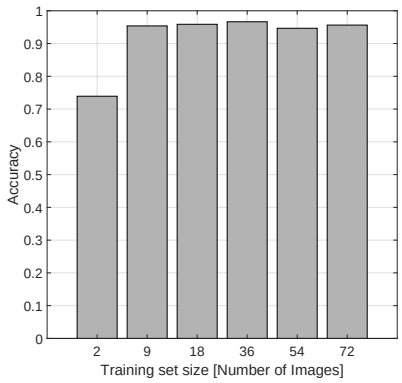

Fig. 12: Classification accuracy as a function of the training set size (number of images used as input to the classifier).

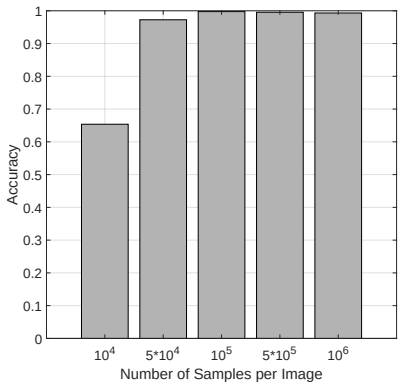

Fig. 13: Classification accuracy as a function of the number of I-Q samples considered to generate each image.

both the transmitter and the jammer (close to each other). The results show that 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. Nevertheless, we highlight that the purpose of this analysis is to investigate how the classification accuracy is affected when the RSS of the transmitter and the jammer decreases compared to the power of the background radio noise. We stop our measurements at 21 meters since the received I-Q samples associated with correctly received bits were detrimentally affected by the combination of jamming and background radio noise—but still preserving patterns allowing their correct classification.

Training set size. We investigate the impact of the training set size (number of images as input to the classifier) by setting the number of I-Q samples per image at $5 \cdot 10^5$. The measurement scenario is constituted of the same three radios considered for the previous analyses, i.e., the transmitter and the jammer located 10 meters far away from the receiver (with the transmitter and the jammer close to each other). Moreover, the training and testing processes have been performed by resorting to different jamming radio hardware to be independent of potential biases due to the radio hardware fingerprinting. Therefore, we collected one measurement of 600 seconds for the training process and one more measurement for the testing process resorting to two different SDRs for the jamming. Figure 12 reports the accuracy of the classification as a function of the training set size (number of images adopted for the training). The achieved performance highlights a very strong bias in the image patterns generated by the I-Q samples since, with only a few images in the training set (more than 9), it is already possible to achieve reasonably high performance (classification accuracy higher than 0.9).

Number of samples per image. Our last analysis focuses on the number of I-Q samples considered to generate each image, providing an indication of the time required for *BloodHound* to detect and classify the jammer. We considered the same measurement setup as before, including one data transmitter, one jammer, and one receiver located 10 meters away from the transmitter and the jammer (close to each other). Moreover, for each configuration, we swap the jammer by adopting two different SDRs. Subsequently, we considered each pair of measurements as one for training and the other one for testing—this is to avoid any bias due to hardware fingerprinting of the transmitting source. Figure 13 shows the classification accuracy as a function of the number of I-Q samples used to generate each image, where the number of I-Q samples has been varied between 10^4 and 10^6 samples per image. As previously mentioned, the usage of a higher number of I-Q samples reduces the number of images provided as input to the algorithm (given a predefined measurement duration); therefore, for the vast majority of our analysis, we assumed a number of I-Q samples being equal to $5 \cdot 10^5$ while accepting the number of images as a function of the measurement duration. By adopting $5 \cdot 10^5$ I-Q samples per image and a sample rate of 10^6 samples per second (the value we used to collect our samples), only 500 msec are needed to generate an image and test it for jamming detection and identification. Such performance proves the reduced time and the flexibility characterizing our solution.

VII. Related Work

Jamming detection has been a hot topic in the scientific community, and several approaches have been proposed. The most effective and usable are the ones leveraging traffic-related features, such as the PDR [15], throughput, number of retransmissions [16], RSS [17], and a combination thereof [18]. While the cited schemes passively detect jamming, other schemes, such as [19], actively detect intentional interference by embedding secrets in the messages, to detect illegitimate transmissions. Other solutions, such as the ones by Wood et al. [20] and Di Pietro et al. [21], detect jammed topology areas, to redirect the traffic toward more reliable links. Many studies in the literature also proposed the adoption of Machine Learning (ML) models to detect jamming in wireless networks, still considering traffic-related features. Wu et al. [22] designed several CNN models to classify five different types of jamming

signals, i.e., audio jamming, narrow-band jamming, pulse jamming, sweep jamming, and spread spectrum jamming. Gecgel et al. [23] implemented and compared the performance of two neural network architectures, i.e., CNNs and RNNs, to detect the presence of a jammer in a wireless communication network using the Orthogonal Frequency Division Multiplexing (OFDM) technique. Arjoune et al. [24] studied the performance of three ML techniques for the recognition of jamming in 5G networks, i.e., Support Vector Machine (SVM), Random Forest and neural networks. They used several features, including the RSS, bad packet ratio, PDR, and the assessment of the channel condition. Cai et al. [10] used frequency spectrum waterfall plots and CNNs models to discriminate among three jamming types, i.e., sweep jamming, single-tone jamming, and multi-tone jamming. A similar solution based on black and white spectrogram images has been used by Morales et al. in [25], to detect intentional interference activities in Global Navigation Satellite Systems (GNSS). They used SVM and CNNs to discriminate among Amplitude Modulated (AM) jammers, chirp jammers, Frequency Modulated (FM) jammers, pulse jammers, and Narrow Band (NB) jammers. Building on the previous work, Swinney et al. [26] used a combination of traffic-related features to build reference images for jamming scenarios, then used the well-known VGG16 model as input for features extraction, and finally, they used SVM, Logistic Regression, and Random Forest for jamming classification.

VIII. Conclusion

We have proposed *BloodHound*, a solution for early detection of jamming leveraging Convolutional Neural Networks. *BloodHound* infers on the presence of a jammer earlier than traditional solutions, even if the bit-error rate associated with the link is close to zero. We have shown that the accuracy of *BloodHound* is higher than 0.99—being able to detect and identify three types of jamming conditions, i.e., no-jamming, tone-jamming, and random Gaussian jamming. We also tested *BloodHound* against several configuration parameters, such as distance to the jamming source, different jamming hardware, power of the jamming source, and finally, training set size. Our results show that early jamming detection is possible, effective, and also efficient, as only 500 ms are required to detect the presence of the jammer.

Acknowledgements

This publication was made possible by an award GSRA7-1-0510-20045 from Qatar National Research Fund (a member of Qatar Foundation). The contents herein are solely the responsibility of the author[s].

References

[1] A. Mpitziopoulos, et al., "A survey on jamming attacks and countermeasures in WSNs," *IEEE Commun. Surveys & Tuts.*, vol. 11, no. 4, pp. 42–56, 2009.

[2] K. Grover, et al., "Jamming and Anti–Jamming Techniques in Wireless Networks: a Survey," *Int. Jour. of Ad Hoc and Ubiq. Comput.*, vol. 17, no. 4, pp. 197–215, 2014.

[3] Tedeschi, P. et al., "Modelling a Communication Channel under Jamming: Experimental Model and Applications," in *IEEE Intl Conf on Parallel Distrib. Process. with Apps.*, 2021, pp. 1562–1573.

[4] A. Marttinen, et al., "Statistics-based jamming detection algorithm for jamming attacks against tactical MANETs," in *IEEE Military Commun. Conf.* IEEE, 2014, pp. 501–506.

[5] M. Strasser, et al., "Detection of Reactive Jamming in Sensor Networks," *ACM Trans. on Sensor Netw. (TOSN)*, vol. 7, no. 2, pp. 1–29, 2010.

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

[7] Ettus Research, "USRP X310," https://www.ettus.com/all-products/x310-kit/, 2020, (Accessed: 2022-01-19).

[8] S. Alhazbi, S. Sciancalepore, and G. Oligeri, "A Dataset of IQ samples in Indoor Jamming Scenarios," Oct. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.7119040

[9] Q. Qu, et al., "JRNet: Jamming Recognition Networks for Radar Compound Suppression Jamming Signals," *IEEE Trans. on Veh. Technol.*, vol. 69, no. 12, pp. 15 035–15 045, 2020.

[10] Y. Cai, et al., "Jamming pattern recognition using spectrum waterfall: A deep learning method," in *IEEE Int. Confe. on Comput. and Commun.* IEEE, 2019, pp. 2113–2117.

[11] G. Shao, et al., "Convolutional neural network-based radar jamming signal classification with sufficient and limited samples," *IEEE Access*, vol. 8, pp. 80 588–80 598, 2020.

[12] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. on knowledge and data engineer.*, vol. 22, no. 10, pp. 1345–1359, 2009.

[13] G. Oligeri et al., "PAST-AI: physical-layer authentication of satellite transmitters via deep learning," *CoRR*, vol. abs/2010.05470, 2020. [Online]. Available: https://arxiv.org/abs/2010.05470

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

[15] M. Çakiroğlu and A. T. Özcerit, "Jamming Detection Mechanisms for Wireless Sensor Networks," in *Proc. of the Int. Conf. on Scalable Informat. Systs.*, 2008, pp. 1–8.

[16] 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.

[17] 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.

[18] N. Sufyan, et al., "Detection of jamming attacks in 802.11 b wireless networks," *EURASIP Journ. on Wirel. Commun. and Netw.*, vol. 2013, no. 1, pp. 1–18, 2013.

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

[20] A. Wood, et al., "JAM: A jammed-area mapping service for sensor networks," in *RTSS*. IEEE, 2003, pp. 286–297.

[21] R. D. Pietro and G. Oligeri, "Gopjam: Key-less jamming mitigation via gossiping," *J. Netw. Comput. Appl.*, vol. 123, pp. 57–68, 2018.

[22] Z. Wu, et al., "Jamming signals classification using convolutional neural network," in *IEEE Int. Symp. on Signal Process. and Informat. Technol.* IEEE, 2017, pp. 062–067.

[23] S. Gecgel, et al., "Jammer detection based on artificial neural networks: A measurement study," in *Proc. of the ACM Worksh. on Wirel. Secur. and Machine Learning*, 2019, pp. 43–48.

[24] Y. Arjoune, et al., "A novel jamming attacks detection approach based on machine learning for wireless communication," in *Int. Conf. on Informat. Network.* IEEE, 2020, pp. 459–464.

[25] R. Morales Ferre, et al., "Jammer classification in GNSS bands via machine learning algorithms," *Sensors*, vol. 19, no. 22, p. 4841, 2019.

[26] 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.