Real-time Classification of Jamming Attacks against UAVs via on-board Software-defined Radio and Machine Learning-based Receiver Module

J. Price[(1)], Y. Li[(1)], K. Al Shamaileh[(1)], Q. Niyaz[(1)], N. Kaabouch[(2)], and V. Devabhaktuni[(3)]

[(1)] Electrical and Computer Engineering Department, Purdue University Northwest, Hammond 46323, IN, USA
[(2)] School of Electrical Engineering and Computer Science, University of North Dakota, Grand Forks 58202, ND, USA
[(3)] Electrical and Computer Engineering Department, The University of Maine, Orono 04469, ME, USA
E-mail: kalshama@pnw.edu

2022 IEEE International Conference on Electro Information Technology (eIT) | 978-1-6654-8009-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/eIT53891.2022.9813923

***Abstract*—In this article, real-time jamming detection against unmanned aerial vehicles (UAVs) is proposed via the integration of a software-defined radio (SDR) with an on-board Raspberry Pi processor. The SDR is utilized for capturing and forwarding the radio frequency signals to a receiver module hosted in the processor. This module extracts signal features characterized by orthogonal frequency division multiplexing (OFDM) parameters, energy parameters, and signal-to-noise ratio (SNR) parameters. Upon feature extraction, the aforementioned module exploits a machine learning (ML) classifier for detecting and classifying four jamming types; namely, barrage, single-tone, successive-pulse, and protocol-aware. The resulting configuration yielded in an overall detection rate (DR) of 93% and a false alarm rate (FAR) of 1.1%, which are in proximity to their counterparts obtained during the validation stage of the receiver module.**

***Index Terms*—Jamming classification, machine learning (ML), orthogonal frequency division multiplexing (OFDM), software-defined radio (SDR), unmanned aerial vehicles (UAVs).**

I. INTRODUCTION

UNMANNED arial vehicles (UAVs) have recently seen a widespread use for a variety of applications, such as research missions, mailing delivery, and disaster management. With autonomous control, UAVs (e.g., drones) are inherently susceptible to a variety of cyberattacks aiming for sabotaging their operations or accessing their data and trajectory information. In other words, a malfunction resulting from a cyberattack (i.e., jamming) potentially compromises sensitive payloads or, in some extreme scenarios, leads to aerial collisions. Hence, the risks of cyberattacks associated with the functionality of UAVs must be acknowledged and mitigated.

Cyberattacks on UAVs branch into data interception, data manipulation, and denial-of-service. Data interception is often encountered with broadcast authentication protocols, which use cryptographic and non-cryptographic techniques [1–5]. On the other hand, secure location verification (e.g., multilateration) were adopted to sideline the impacts of data manipulation attacks [6,7]. However, these solutions are inefficient for jamming detection provided that attackers can easily launch interference with software-defined radios (SDRs) to disrupt the trajectory of a UAV. As a result, developing jamming detection techniques that also adhere to the existing standards is of grave importance. In this work, radiometric transmissions (i.e., signal features) are used to train a machine learning (ML) algorithm within a receiver module for jamming detection and classification. To this end, SDR units are exploited to launch jamming attacks and collect signal features. The underlying approach facilitates minimal modifications to standards and considers realistic attack setups unlike other reported efforts that assume software/hardware changes and simulation-based scenarios [8–17]. It is noteworthy to point out that ML was proposed for satellite communications, vehicle Ad Hoc networks, 5G networks, Internet of Things, and UAVs with applications to jamming detection, trajectory optimization, swarm communication, situational awareness, and malicious attack mitigation [18–25].

The groundwork of this research was laid for in a previous effort that resulted in large experimental datasets of signal features [26]. These features were obtained via SDRs and GNURadio flowgraphs built with an external computer. Feature extraction was followed by training and validating multiple ML algorithms for detecting/classifying different jamming attacks. The work herein extends on [26] in the following aspects:

1. The ML algorithm with the highest detection rate (DR) is integrated with the GNURadio feature extraction flowgraph to create an all-inclusive seamless receiver module that is capable of preforming real-time decision-making for jamming detection/classification.
2. The developed module is incorporated into a drone via on-board Raspberry Pi processor paired with a low-profile SDR (i.e., HackRF One). This entails loading the processor with the Python code generated from the overall module and verifying the jamming classification process.

This article is organized as follows: Section II details the integration of the ML algorithm and GNURadio feature extraction flowgraph into one receiver module. Here, the extracted features are fed directly to the classification algorithm. Section III elaborates on exploiting the module to enable a drone for detecting and classifying specific jamming types. This setup is realized with on-board SDR and Raspberry Pi processer. Conclusions are provided in Section IV.

978-1-6654-8009-3/22/$31.00 ©2022 IEEE

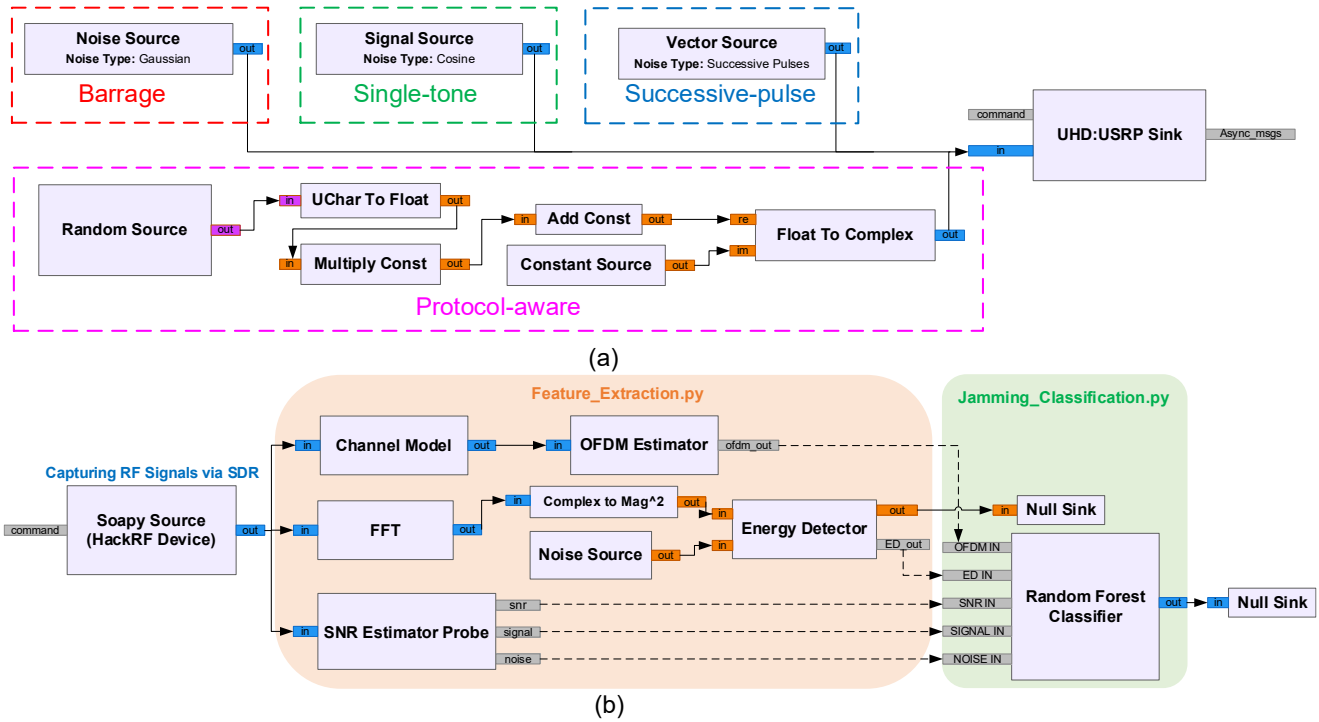


Fig. 1. Simplified GNURadio flowgraphs for (a) launching jamming attacks and (b) receiver module for extracting signal features and executing jamming detection/classification. Further details on each jamming attack and ML training/testing are available in [26].

II. Module Development and Validation

In this section, a real-time jamming detection/classification module is developed and tested on a computer prior porting to the drone's on-board processor. This module enables the detection and classification of four types of jamming attacks: barrage, single-tone, successive-pulse, and protocol-aware. This module also considers IEEE 802.11 orthogonal frequency division multiplexing (OFDM) communication technology at 2.4 GHz center frequency. Fig. 1(a) shows a simplified GNURadio flowgraph that conveys how each jamming type is launched, whereas Fig. 1(b) illustrates the feature extraction and ML classification blocks at the receiver side. The five-class random forest (RF) classifier developed in a previous effort is utilized here for its high DR, i.e., 92% in comparison to the other investigated algorithms [26]. In this setup, eight signal features are extracted. Of these features, three are OFDM-specific (i.e., subcarrier length, cyclic prefix length, subcarrier spacing). The subcarrier length indicates the number of the subcarriers being used. The cyclic prefix length controls symbol overlapping, and the subcarrier spacing is the frequency separation among subcarriers [27]. The OFDM Estimator block shown in Fig. 1(b) is used to extract these features [28]. Two other features are energy-specific; namely, average received power and threshold. The latter is a binary indicator that returns 1 once the average received power exceeds a certain level and returns 0 otherwise. Such features are extracted with the use of the Energy Detector block [28]. Finally, three additional features; specifically, signal-to-noise ratio (SNR), average signal power, and average noise power are extracted from the SNR Estimator Probe block. It is worthy to be pointed out that the average received power obtained from the Energy Detector block conveys noise energy; whereas the average signal power obtained from the SNR Estimator Probe presents the estimated signal power excluding noise power. For the training and validation of the RF classifier, a total of 23,565 signal samples were collected following the attack scenario reported in [26, Fig. 2]. Of these samples, 10,071 were obtained under no jamming, whereas 3,392, 3,367, 3,378, and 3,357 were obtained in the presence of barrage, single-tone, successive-pulse, and protocol-aware jamming, respectively. The complete dataset with the 23,565 samples is provided in [29]. While developing the classifier, 10-fold cross-validation was used and grid search is utilized for finding the optimal hyper-parameters.

The jamming classifcation block receives all features from the feature extraction block and writes classification parameter C, $C = 0, 1 \ldots 4$, to a file according to the jamming type as depicted in Table 1. The resulting C values are utilized in evaluating the classification accuracy (i.e., DR) by comparing the predictions of the module with the actual launched jamming type. Fig. 2 illustrates the pseudo code for the receiver module. Also, Fig. 3 shows a sample output file with prediction results for $C = 0$ (i.e., No Jamming), which results in 100% DR. This DR is calculated and updated for every set of extracted features. The *Prediction* together with the *Jamming Type* are printed to validate the functionality of the developed receiver module.

TABLE I
Classification Parameter and Corresponding Jamming Type

Parameter C	Jamming Type
0	No Jamming
1	Barrage
2	Single-tone
3	Successive-pulse
4	Protocol-aware

Algorithm: Real-Time Jamming Classification

Given: Features – Feature_Extraction.py
Model – Jamming_Classification.py
YActual – List with actual jamming type
YPredict – List of predicted outputs
C – Jamming Type // $C \in \{0,4\}$

1: **Procedure:** Real_Time_Jamming_Classification_Module()
2: Prediction = Jamming_Classification.predict(Features)
3: YActual.append(*C*) // Replace with jamming type
4: YPredict.append(Prediction)
5: DR = DetectionRate(YActual, YPredict)
6: *Loop*: **for** each Prediction **do**
7: print Prediction to console window
8: **end for**
9: Print DetectionRate to console window
10: File = open(Results_File.txt)
11: File.write(Prediction, DR×100, *C*)
12: **end Procedure**

Fig. 2. Pseudocode of the receiver module for jamming classification.

The receiver module in Fig. 1(b) is also tested by calculating the DR while a jammer is in proximity to a receiver terminal (i.e., computer). To collect experimental data, two computers are utilized. The first computer uses an SDR to function as a receiver running the developed module. This computer has a 64-bit Windows 10 machine with Intel®Core™i7- 7700HQ CPU @ 2.8 GHz processor and 32 GB of memory. The second computer is for launching each of the jamming attacks via another SDR. Fig. 4 presents sample outputs of 15 predictions for each attack type. The receiver module demonstrates a DR of 100% for barrage and successive-pulse types. Moreover, the resulting DR for single-tone and protocol-aware are 93.33% and 40%, respectively. The high misprediction in protocol-aware is attributed to its spectral similarities to barrage jamming and is expected to improve with the increase of samples. It is to be stressed that the DR is updated as samples are added. For instance, Fig. 4(b) shows that sample 1 is mispredicted. Hence, DR assumes one misprediction out of a single attempt, i.e., 0%. After correctly predicting sample 2, the DR is updated with one correct prediction out of two attempts, i.e., 50%. Finally, after receiving all samples, the overall DR is found to be 14 correct predictions of 15 predictions, i.e., 93.33%. Accordingly, the DR can be expressed as follows:

	Prediction	DR(%)	Jamming Type
1:	0	100.00	0
2:	0	100.00	0
3:	0	100.00	0
4:	0	100.00	0
5:	0	100.00	0
6:	0	100.00	0
7:	0	100.00	0
8:	0	100.00	0
9:	0	100.00	0
10:	0	100.00	0
11:	0	100.00	0
12:	0	100.00	0
13:	0	100.00	0
14:	0	100.00	0
15:	0	100.00	0

Fig. 3. Sample output file under no jamming attack ($C = 0$).

	Prediction	DR(%)	Jamming Type
1:	1	100.00	1
2:	1	100.00	1
3:	1	100.00	1
4:	1	100.00	1
5:	1	100.00	1
6:	1	100.00	1
7:	1	100.00	1
8:	1	100.00	1
9:	1	100.00	1
10:	1	100.00	1
11:	1	100.00	1
12:	1	100.00	1
13:	1	100.00	1
14:	1	100.00	1
15:	1	100.00	1

(a)

	Prediction	DR(%)	Jamming Type
1:	1	0.00	2
2:	2	50.00	2
3:	2	66.67	2
4:	2	75.00	2
5:	2	80.00	2
6:	2	83.33	2
7:	2	85.71	2
8:	2	87.50	2
9:	2	88.89	2
10:	2	90.00	2
11:	2	90.91	2
12:	2	91.67	2
13:	2	92.31	2
14:	2	92.86	2
15:	2	93.33	2

(b)

	Prediction	DR(%)	Jamming Type
1:	3	100.00	3
2:	3	100.00	3
3:	3	100.00	3
4:	3	100.00	3
5:	3	100.00	3
6:	3	100.00	3
7:	3	100.00	3
8:	3	100.00	3
9:	3	100.00	3
10:	3	100.00	3
11:	3	100.00	3
12:	3	100.00	3
13:	3	100.00	3
14:	3	100.00	3
15:	3	100.00	3

(c)

	Prediction	DR(%)	Jamming Type
1:	2	0.00	4
2:	2	0.00	4
3:	2	0.00	4
4:	2	0.00	4
5:	2	0.00	4
6:	2	0.00	4
7:	2	0.00	4
8:	4	12.50	4
9:	4	22.22	4
10:	4	30.00	4
11:	4	36.36	4
12:	1	33.33	4
13:	4	38.46	4
14:	1	35.71	4
15:	4	40.00	4

(d)

Fig. 4. Sample output file for each jamming type: (a) barrage, (b) single-tone, (c) successive-pulse, and (d) protocol-aware.

$$\mathrm{DR} = \frac{\text{Correctly Predicted Samples}}{\text{Samples in the Dataset}} \tag{1}$$

III. Module Implementation

The receiver module depicted in Fig. 1(b) is adopted in a Clover 4.2 open-source drone from COEX, as featured in Fig. 5. This drone is equipped with Raspberry Pi 4 that facilitates 64-bit ARM Cortex-A72 quad-core CPU @ 1.5 GHz processor and 1 GB of memory [30]. This processor is interfaced with HackRF One SDR via micro-USB cable for receiving the radio frequency transmissions at 2.4 GHz (i.e., drone operating frequency). Once these transmissions are captured, two Python-based subroutines are executed: Feature_Extraction.py for extracting the eight above-mentioned features and Jamming_Classification.py for exploiting these features for running, in real-time, a five-class ML classifier for detecting jamming presence and specifying its type.

Prior implementing the module, the training and validation of the RF classifier are repeated using the Raspberry processor to calculate the detection/classification time and to establish a comparison between the computer and Raspberry processors considering DR, F-score (FS), and false-alarm rate (FAR). The calculations of these evaluation metrics are performed as follows:

$$\text{F-score} = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2}$$

$$\mathrm{FAR} = \frac{\text{False Positive Samples}}{\text{False Positive+True Negative Samples}} \tag{3}$$

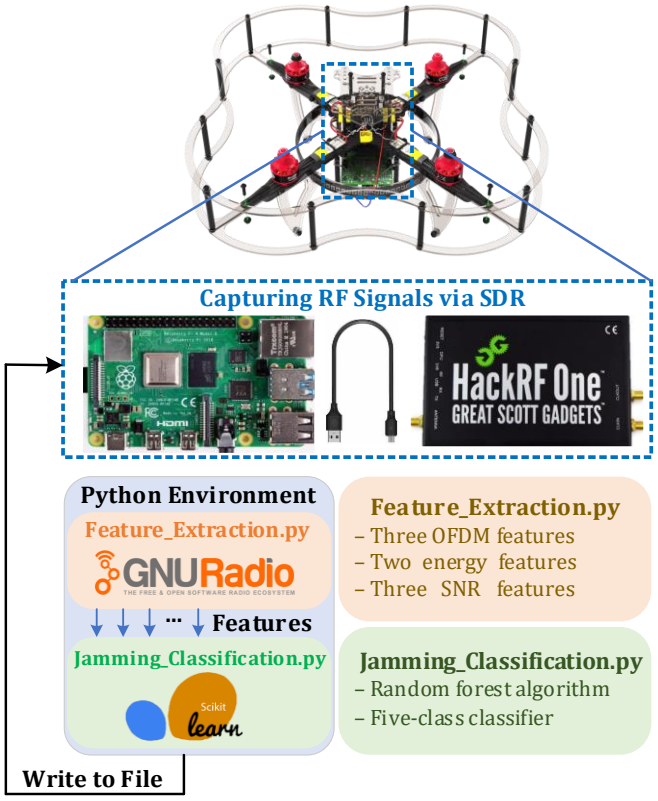
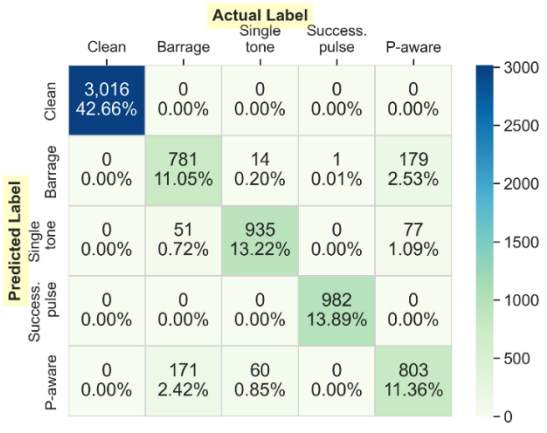


Fig. 5. Hardware-software configurations for classifying jamming.

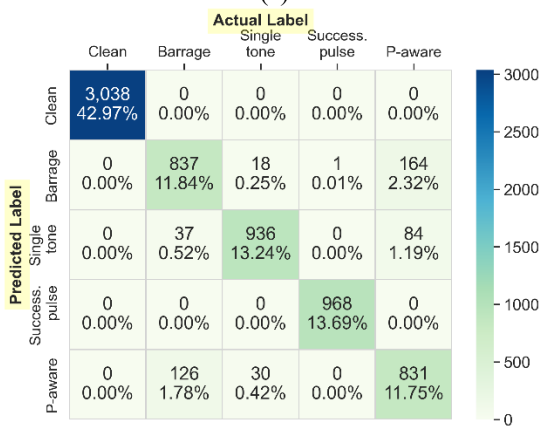
In (2), the "Precision" is defined as the number of positive samples predicted as positive (i.e., true positive) divided by the sum of true positive and negative samples predicted as positive (i.e., false positive). Also, the "Recall" is the number of true positive samples divided by the sum of true positive and positive samples predicted as negative (i.e., false negative). F-score is computed from the Precision and Recall to represent their harmonic mean. Finally, the FAR is the number of false positive samples divided by the sum of false positive and true negative samples predicted by the classifier. Table II illustrates that the validation accuracy (VA) and DR of the developed classifier are higher than 92% for both processors. In addition, the testing time with the use of the Raspberry processor is 10.64 seconds. Since this testing time results from classifying nearly 7,070 samples (i.e., 30% of the overall dataset), the average processing time of the five-class RF model to classify a sample is 1.5 ms, enabling real-time jamming detection/classification. Fig. 6 illustrates the resulting confusion matrices of the classifier after completing the training/testing with the computer and Raspberry Pi processors. Such matrices indicate that none of the jamming attacks are misclassified as No Jamming. Misclassification, however, occurs mostly among barrage and protocol-aware jamming types mainly due to their similar spectral properties. The weighted FARs are computed from these matrices to be 1.33% and 1.10% for the computer and raspberry Pi processors, respectively. Furthermore,

TABLE II
MODEL EVALUATION METRICS UTILIZING COMPUTER AND RASPBERRY PI

	VA (%)	DR (%)	FS	Training (sec)	Testing (sec)
Computer	92.20 ± 0.600	92.20	0.92	5.404	0.411
Raspb. Pi	93.57 ± 0.006	93.49	0.91	971.69	10.64



(a)



(b)

Fig. 6. Confusion matrices for testing the five-class RF model with (a) computer and (b) Raspberry Pi processors.

TABLE III
UPDATED ACCURACY SCORE FOR EACH JAMMING CASE FOR 100 SAMPLES

Parameter *C*	0	1	2	3	4
No. of Misclassifications	0	31	0	0	40
DR (%)	100	69	100	100	60

it can be inferred from the confusion matrix in Fig. 6(b) that, of the 7,070 testing samples, the resulting DRs of the classifier for $C = 0$, 1, 2, 3, and 4 are 100%, 83.7%, 95.1%, 99.9%, and 77%, respectively. After validating the classifier using Raspberry, the Python code obtained from the overall GNURadio receiver module is built in the drone as described in Fig. 5. All five scenarios (i.e., no jamming, barrage, single-tone, successive-pulse, protocol-aware) are tested while the drone is idling. In these tests, an SDR is used to launch each jamming type considering a 1.5-meter jammer-drone separation to enable the receiver module for capturing and classifying 100 samples per scenario. The DR for all scenarios is calculated and updated as elaborated in Section II and is summarized in Table III. The developed module showed 100% accuracy in detecting jamming presence. Also, no misclassification occurs when there is no jamming or when either single-tone or successive-pulse is launched. On the other hand, barrage is misclassified 31% of the time (18% as single-tone, 13% as protocol-aware); whereas protocol-aware is misclassified 40% of the time (33% as barrage, 5% as single-tone, 2% as successive-pulse). Such results suggest that the proposed receiver module provides a viable solution for jamming detection and classification.

V. Conclusion

Real-time jamming detection and classification configuration with applications to UAVs is proposed via the integration of an SDR with on-board Raspberry Pi processor. The SDR is utilized for capturing and forwarding the radio frequency signals to a receiver module hosted in the processor. This module extracts OFDM, energy, and SNR signal features and exploits an ML algorithm for detecting and classifying four jamming types: barrage, single-tone, successive-pulse, and protocol-aware. The resulting configuration yielded in an overall DR and FAR of 93% and 1.10%, respectively. These values are in proximity to their counterparts obtained during the validation of the receiver module. Future work entails testing the developed module while the drone is in operation and implementing jamming mitigation protocols such as path rescheduling and beamforming front-end circuitry.

Acknowledgment

This research is funded by the National Science Foundation, Secure and Trustworthy Cyberspace under Award no. 2006662.

References

[1] M. Strohmeier, V. Lenders, and I. Martinovic. "On the security of the automatic dependent surveillance-broadcast protocol," *IEEE Commun. Surveys & Tutorials*, vol. 17, no. 2, pp.1066–1087, 2014.

[2] M. Manesh and N. Kaabouch, "Analysis of vulnerabilities, attacks, countermeasures and overall risk of the Automatic Dependent Surveillance-Broadcast (ADS-B) system," *Int. J. Critical Infra. Protec.,* vol. 19, pp. 16–31, 2017.

[3] K. Wesson, T. Humphreys, and B. Evans, "Can cryptography secure next generation air traffic surveillance?" *IEEE Sec. Priv. Mag.*, draft (2014), last accessed on March 2, 2022.

[4] C. Giannatto, Jr., "Challenges of implementing automatic dependent surveillance broadcast in the nextgen air traffic management system," Ph.D. dissertation, Univ. Maine, Orono, ME, USA, 2015.

[5] B. Danev, H. Luecken, S. Capkun, and K. El Defrawy, "Attacks on physical-layer identification," in *Proc. 3rd ACM Conf. Wireless Net. and Sec.* (*WiSec*), 2010, pp. 89–98.

[6] S. Brands and D. Chaum, "Distance-bounding protocols," in *Proc. Workshop Theory Appl. Cryptograph. Techn*. Berlin, Germany: Springer, 1993, pp. 344–359.

[7] B. Xiao, B. Yu, and C. Gao, "Detection and localization of sybil nodes in VANETs," in *Proc. Workshop Dependability Issues Wireless Ad Hoc Netw. Sensor Netw.* (*DIWANS*), 2006, pp. 1–8.

[8] M. Sliti, W. Abdallah, and N. Boudriga, "Jamming attack detection in optical UAV networks," in *Proc. 20th Int. Conf. Transparent Opt. Netw.* (*ICTON*), 2018, pp. 1–5.

[9] D. Karagiannis and A. Argyriou, "Jamming attack detection in a pair of RF communicating vehicles using unsupervised machine learning," *Veh. Commun*., vol. 13, pp. 56–63, 2018.

[10] Y. Arjoune, F. Salahdine, M. S. Islam, E. Ghribi, and N. Kaabouch, "A novel jamming attacks detection approach based on machine learning for wireless communication," in *Proc. Int. Conf. Inf. Netw.* (*ICOIN*), 2020, pp. 459–464.

[11] L. Mokdad, J. Ben-Othman, and A. T. Nguyen, "DJAVAN: Detecting jamming attacks in vehicle ad hoc networks," *Perform. Eval.*, vol. 87, pp. 47–59, 2015.

[12] A. Nguyen, L. Mokdad, and J. Ben Othman, "Solution of detecting jamming attacks in vehicle ad hoc networks,'' in *Proc. 16th ACM Int. Conf. Model., Anal. Simul. Wireless Mobile Syst*., 2013, pp. 405–410.

[13] J. Grover, N. K. Prajapati, V. Laxmi, and M. Gaur, "Machine learning approach for multiple misbehavior detection in vanet," in *Proc. 1st Int. Conf. Adv. Comput. Commun.* (*ACC*), Kochi, India, 2011, pp. 644–653.

[14] H. Liu, B. Lang, M. Liu, and H. Yan, "CNN and RNN based payload classification methods for attack detection," *Knowl.-Based Syst.*, vol. 163, pp. 332–341, 2019.

[15] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[16] X. Wang, X. Wang, and S. Mao, "RF sensing in the Internet of Things: A general deep learning framework," *IEEE Commun. Mag*., vol. 56, no. 9, pp. 62–67, 2018.

[17] C. Liu, J. Wang, X. Liu, and Y.-C. Liang, "Deep CM-CNN for spectrum sensing in cognitive radio," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2306–2321, Oct. 2019.

[18] S. Gecgel and G. K. Kurt, "Intermittent jamming against telemetry and telecommand of satellite systems and a learning-driven detection strategy," in *Proc. 3rd ACM Workshop Wireless Secur. Mach. Learn.* (*WiseML*), 2021, pp. 43–48.

[19] S. Gecgel, C. Goztepe, and G. K. Kurt, "Jammer detection based on artificial neural networks: A measurement study," in *Proc. ACM Workshop Wireless Secur. Mach. Learn.* (*WiseML*), 2019, pp. 43–48.

[20] 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 *Proc. IEEE Int. Symp. World Wireless, Mobile Multimedia Netw.*, 2014, pp. 1–10.

[21] B. Upadhyaya, S. Sun, and B. Sikdar, "Machine learning-based jamming detection in wireless IoT networks," in *Proc. IEEE VTS Asia Pacific Wireless Commun. Symp.* (*APWCS*), 2019, pp. 1–5.

[22] M. Hachimi, G. Kaddoum, G. Gagnon, and P. Illy, "Multi-stage jamming attacks detection using deep learning combined with kernelized support vector machine in 5G cloud radio access networks," in *Proc. Int. Symp. Netw., Comput. Commun.* (*ISNCC*), 2020, pp. 1–5.

[23] P. Bithas, E. Michailidis, N. Nomikos, D. Vouyioukas, and A. G. Kanatas, "A survey on machine-learning techniques for UAV-based communications," *Sensors*, vol. 19, no. 23, p. 5170, 2019.

[24] Q. Wu, H. Wang, X. Li, B. Zhang, and J. Peng, "Reinforcement learningbased anti-jamming in networked UAV radar systems," *Appl. Sci.*, vol. 9, no. 23, p. 5173, 2019.

[25] X. Lu, L. Xiao, C. Dai, and H. Dai, "UAV-aided cellular communications with deep reinforcement learning against jamming," *IEEE Wireless Commun.*, vol. 27, no. 4, pp. 48–53, 2020.

[26] 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. 16859–16870, 2022.

[27] Y. Cho, J. Kim, W. Yang, and C. Kang, "*Introduction OFDM*. Wiley, pp. 111–151, 2010. [Online]. Available: https://ieeexplore.ieee.org/book/5675894.

[28] S. Müller and C. Richardson. GitHub—Gnuradio/Gr-Inspector: Signal Analysis Toolbox for GNU Radio. Accessed: Dec. 21, 2021. [Online]. Available: https://github.com/gnuradio/gr-inspector.

[29] GitHub: UAVs Jamming Detection and Classification. Accessed: Mar. 1, 2022. Available: https://github.com/michaelevol/uavs_jamming_detection.

[30] COEX. Accessed: Jan. 12, 2022. [Online]. Available: https://coex.tech/clover.