IoT

MDPI

Article

Developing a Low-Order Statistical Feature Set Based on Received Samples for Signal Classification in Wireless Sensor Networks and Edge Devices

George D. O'Mahony [1,*], Kevin G. McCarthy [1], Philip J. Harris [2] and Colin C. Murphy [1]

[1] Department of Electrical and Electronic Engineering, University College Cork, T12 K8AF Cork, Ireland; k.mccarthy@ucc.ie (K.G.M.); colinmurphy@ucc.ie (C.C.M.)
[2] Raytheon Technologies Research Center, T23 XN53 Cork, Ireland; harrispj@rtx.com
* Correspondence: george.omahony@umail.ucc.ie

Abstract: Classifying fluctuating operating wireless environments can be crucial for successfully delivering authentic and confidential packets and for identifying legitimate signals. This study utilizes raw in-phase (I) and quadrature-phase (Q) samples, exclusively, to develop a low-order statistical feature set for wireless signal classification. Edge devices making decentralized decisions from I/Q sample analysis is beneficial. Implementing appropriate security and transmitting mechanisms, reducing retransmissions and increasing energy efficiency are examples. Wireless sensor networks (WSNs) and their Internet of Things (IoT) utilization emphasize the significance of this time series classification problem. Here, I/Q samples of typical WSN and industrial, scientific and medical band transmissions are collected in a live operating environment. Analog Pluto software-defined radios and Raspberry Pi devices are utilized to achieve a low-cost yet high-performance testbed. Features are extracted from Matlab-based statistical analysis of the I/Q samples across time, frequency (fast Fourier transform) and space (probability density function). Noise, ZigBee, continuous wave jamming, WiFi and Bluetooth signal data are examined. Supervised machine learning approaches, including support vector machines, Random Forest, XGBoost, k nearest neighbors and a deep neural network (DNN), evaluate the developed feature set. The optimal approach is determined as an XGBoost/SVM classifier. This classifier achieves similar accuracy and generalization results, on unseen data, to the DNN, but for a fraction of time and computation requirements. Compared to existing approaches, this study's principal contribution is the developed low-order feature set that achieves signal classification without prior network knowledge or channel assumptions and is validated in a real-world wireless operating environment. The feature set can extend the development of resource-constrained edge devices as it is widely deployable due to only requiring received I/Q samples and these features are warranted as IoT devices become widely used in various modern applications.

Keywords: classification; decision tree; edge devices; IoT; machine learning; WSNs; XGBoost; ZigBee

Citation: O'Mahony, G.D.; McCarthy, K.G.; Harris, P.J.; Murphy, C.C. Developing a Low-Order Statistical Feature Set Based on Received Samples for Signal Classification in Wireless Sensor Networks and Edge Devices. *IoT* **2021**, *2*, 449–475. https://doi.org/10.3390/iot2030023

Academic Editor: Antonio Guerrieri

Received: 9 July 2021
Accepted: 28 July 2021
Published: 1 August 2021

1. Introduction

Wireless sensor networks (WSNs) are increasingly being integrated into safety-critical applications such as, for example, the Internet of Things (IoT), which has evolved WSNs into essential elements of current technology. These networks are being adopted across a diverse application space, including missile defense [1], health care (wireless body area networks) [2], remote patient monitoring [3], space exploration [4,5], aerospace, surveillance, industrial sensing, control and monitoring systems [6]. This broad array of machine-to-machine and machine-to-people deployments creates new challenges relating to security, spectral coexistence and threat identification, due to radio spectrum variations and diverse fluctuating operating environments. This is clearly evident in the utilization of WSNs as the communication link from the "things" to the access point in IoT applications (Figure 1). The emerging area of the IoT [7] is rapidly changing the wireless landscape, as advances

are directly affecting (or creating) broadly accepted models such as smart cities/homes [8], edge/cloud computing and big data analytics', amongst others. These critical applications will, likely, continue to embrace WSNs in the modern cost-centered age, due to enabling easier design, installation and maintenance, while simultaneously providing new deployment opportunities.

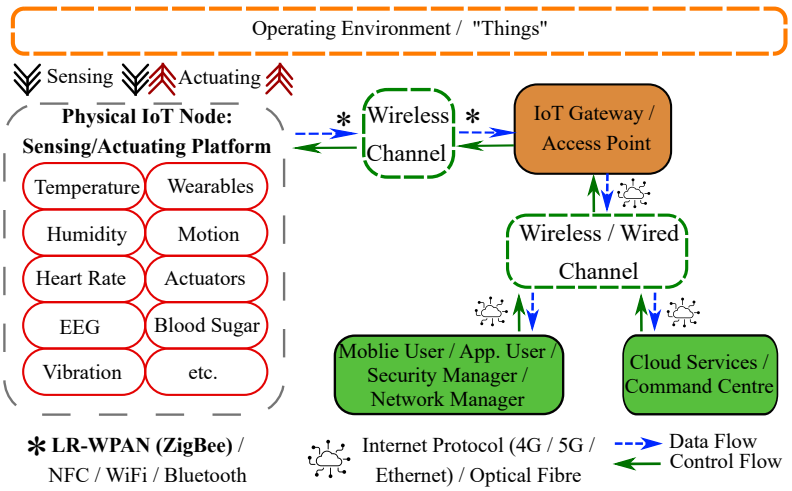


Figure 1. An example IoT architecture showing the potential utilization of WSNs and the security vulnerability produced by the wireless channel.

As the adoption of WSN deployments increases so too does the number of active edge devices. It is predicted that over 29 billion connected devices are likely by 2023 [9] (including 14.7 billion machine-to-machine connections) and will increase in the subsequent years. Any loss of service (or transmissions) from these edge devices can have significant consequences for privacy, safety and system performance [10]. These, typically, computationally and energy constrained edge devices can use information regarding the fluctuating wireless environment [11] in making decentralized decisions. Wireless transmission performance is heavily linked to the quality of the wireless channel and, so, edge devices reacting autonomously to channel variations can accelerate the optimal response. This hypothesis can eradicate obsolete central controller responses due to transmission latency. It is enhanced if the data used are always available to a functioning receiver, that is raw received in-phase (I) and quadrature-phase (Q) samples.

In this paper, raw I/Q samples are collected using software-defined radios (SDRs) in a typical live wireless environment and visualized and analyzed across time, frequency and space (probability density). The analyzed signals (transmitted from commercial and SDR sources) are visualized using a Tektronix real-time spectrum analyzer (RTSA) and associated Digital Phosphor Technology (DPX) [12] in Figure 2. No prior knowledge, except that signals can have different bandwidths, including relationships between samples and symbols or channel assumptions, is used. Classification decisions are based entirely on low-order features extracted from the raw I/Q samples. The contribution and novelty here are the use of received raw I/Q samples and no channel assumptions to develop a feature set for industrial, scientific and medical (ISM) band wireless signal classification using a constant sampling rate based on the WSN protocol, ZigBee [13]. This investigation explores the hypothesis of making high-level edge device/network decisions based entirely on the lowest available data, raw I/Q samples, and low-order statistics. The use of the I/Q samples will be shown to enable the classification of different wireless channels/signals and to distinguish between two different IEEE802.11 versions/transmitters. Utilizing the raw I/Q samples is validated by focusing on, typically, fundamental machine learning algorithms that are verified as being fit for purpose. The developed feature set is verified by analyzing the achieved accuracy and ability to generalize to unseen data. The algorithms

used are support vector machines (SVMs), Random Forest [14], k nearest neighbors (k-NNs), XGBoost and deep neural networks (DNNs).

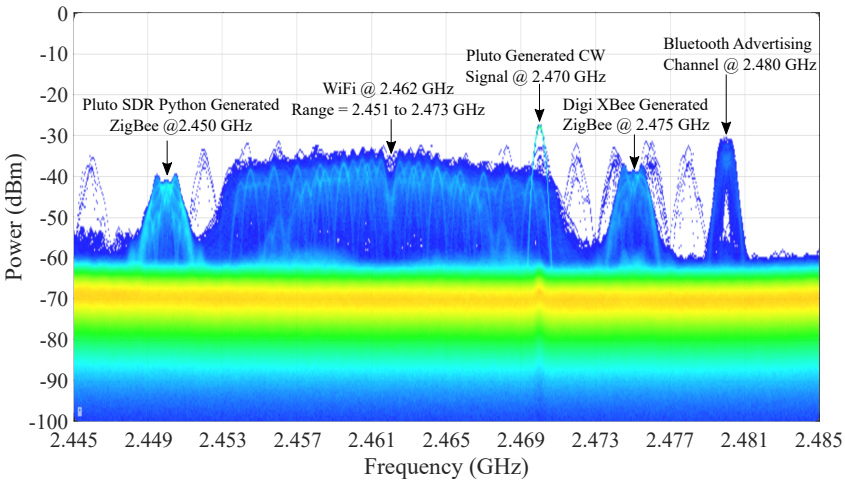


Figure 2. DPX visualization of noise, ZigBee (commercial node), ZigBee (Pluto SDR), CW (Jammer) and coexistence with Bluetooth and WiFi.

The desired deployment is on embedded edge devices (i.e., typical WSN and IoT devices), which can use the designed signal classification model to autonomously adapt procedures in accordance with identified channel information. This operation will enable enhancements in edge device operation and efficiency. This study concentrates on open source platforms, with low cost yet high performance, to enable interoperability. Machine learning is appropriate for this application as the computationally intensive model training and optimization stage can be performed off-line. Thus, only the optimized model is required to be uploaded to the edge device. Here, models are fine-tuned using available test data, including unseen data, and implemented on a Raspberry Pi embedded device.

The remainder of this paper is organized as follows: Section 2 describes related literature and how this study differentiates itself. The materials and methods as discussed in terms of the data collection process and associated hardware in Section 3.1 and the wireless I/Q data feature engineering approach in Section 3.2. The signal classification and initial implementation results are specified in Section 4. Section 5 concludes this paper and outlines future work.

2. Related Work

A brief review of the main related topics and what differentiates this study from the literature are presented in this section. The topics include concentrating on the use of raw I/Q samples and wireless signal classification methods. It is worth noting that focusing on using received raw I/Q samples as the basis for decisions is a relatively unexplored concept. In [15], the authors exploit I/Q component characteristics of a transmitter and deep learning techniques for uncooperative direction finding using a single uncalibrated directional receiver. The authors in [16] use the I/Q imbalance information to learn high-dimensional features for transmitter identification. A generative adversarial network is then employed to detect rogue transmitters, while convolutional and fully connected deep neural networks are used to classify different trusted transmitters. In [17], the authors used I/Q information and recurrent neural networks (RNN) to predict primary user (SDR transmitters) activity in dynamic spectrum access networks. Similarly, in [18], I/Q data and RNNs are leveraged to identify specific universal software-defined radio peripherals. While in [19], deep learning and I/Q samples are used once again for radio device identification (fingerprinting) by learning unchanging hardware-based characteristics of individual transmitters. Finally, in [20], Matlab simulations showed how I/Q samples have promise in detecting interference in WSNs. The majority of the literature using raw I/Q samples

focuses on transceiver or directional identification, which proves the value of raw I/Q samples and opens up new areas for investigation.

The majority of approaches use I/Q samples as that is how the data are received. However, the samples are generally used to gain access to higher levels of data for analysis, for example, symbols/chips and/or bits. These higher-level samples are then used in transforms (Wavelet, Discrete Evolutionary, etc.) to extract information for modulation scheme identification. Chip sequence error patterns are utilized in [21] for channel identification and, as a result, the emitting interference/co-existing signal. Chip analysis is just above the level of I/Q samples but has implementation drawbacks as it requires devices to buffer known patterns for classification.

Wireless signal identification is evolving with hardware and software enhancements. Typically, techniques are used to classify wireless signals based on their modulation schemes. This crucial technology enables transceivers to efficiently utilize available resources [22], especially in cognitive networks. A maximum likelihood-based approach for coherent and non-coherent modulation estimations was proposed in [23], where optimal performance is achieved through the application of mathematical channel models with end-to-end settings. However, likelihood-based methods are, typically, unsuitable for real-world deployments due to high computational costs and, typically, poor generalization in complex environments [22].

Traditional feature-based signal classification consists of data pre-processing, feature extraction and classification (decision algorithm). Designed approaches focus, generally, on digital modulation classification and produce robust performance and low complexity [24]. This is achieved by efficiently extracting features from a statistical analysis of the signals. Various features have been employed in this area, such as cyclic features [25], wavelet transforms [26], higher-order statistics [27] and cumulants [28–31]. The authors in [22] employ spectral features and higher-order cumulants as input features to machine learning-based classifiers. In [32], a Matlab simulation approach is demonstrated, which uses higher-order cumulants, derived from the received signals, as the input features for various classifiers. The work in [33] focused on combining features extracted from a spectral correlation analysis and SVMs. These different features are employed across various machine learning approaches including decision trees [34], SVMs [26], k nearest neighbors (k-NNs), Back Propagation (BP) neural networks [32] and Naive Bayes. These were typically chosen for their performance in pattern recognition but their performance typically generalizes poorly to previously unseen data/scenarios [22]. This study demonstrates that fundamental algorithms can indeed generalize to unseen data, given a sufficiently descriptive feature set.

Focusing on deep learning, O'Shea et al. [35] have outlined the compelling possibility of using deep learning techniques for radio signal identification, based on modulation schemes, and provide methods for real-world adoption. In [36], a SDR prototype for spectrum sharing is developed that utilizes a convolutional neural network for real-time wireless signal modulation classification. This concept is continued in [37], where deep learning methods are shown to outperform both a maximum–minimum eigenvalue ratio-based method and a frequency domain entropy-based method. Other successful examples of deep learning for signal classification based on modulation schemes include [29,38,39]. The majority of these approaches leverage deep learning's ability to learn the features automatically from the input data. This is a data-driven approach and training such data-driven deep networks on large volumes of data typically requires appropriate computational resources and extensive time, both of which are rarely found in deployed communication systems [40]. The approach in [41] focuses on low-cost sensors and a reduced data-driven model, while, crucially for this investigation, identifying wireless classification over frequency, time and space dimensions as an active research problem.

The majority of previous work concentrates on identifying radio devices and/or the modulation scheme being implemented. Typically, statistical features are employed based on cyclic features, wavelet transforms and/or high order statistics and cumulants. Inves-

tigations focused on I/Q samples, generally, use I/Q imbalances to distinguish different transmitters. This study is most comparable to the work in [42] as I/Q samples, RSSI samples and image-based spectrograms, based on fast Fourier transform (FFT) algorithms, are all used for signal classification. However, the authors in [42] focus on a different signal set that consists of LTE, WiFi and DVB-T signals. The study proves the usefulness of using different samples and analyzes both manual and deep learning-based automatic feature extraction for deep learning and supervised machine learning approaches. Notably, the study concluded that low-complexity models need to be developed to reduce the operational costs of future intelligent devices.

Here, a low-complexity feature-driven approach, that can generalize to unseen data, for wireless signal classification aimed at edge device operation is produced. It is novel by the exclusive use of raw I/Q samples and the developed low-order feature set, extracted entirely from received over-the-air I/Q samples in a typical wireless operating environment. To the best of the authors' knowledge, this study applies Hjorth parameters [43] and FFT dynamics in a novel method. Additionally, no assumptions are made about the wireless channel and network data are neglected. Data are collected from an active wireless environment that is typically changeable. The raw I/Q data approach to create a concise novel feature set based on time, frequency and space dimensions, is the main contribution. The need for, and importance of, this type of real over-the-air practical research is discussed in detail as a challenge that needs a solution in [44]. This study uses over-the-air experimentation to produce a low-complexity model for making high-level decisions based on low-level data.

3. Materials and Methods

3.1. Experimental Data Collection

Before extracting the features and designing the detection approach, a data strategy accounting for data quality, quantity and source was established. This is necessary as classifiers will, generally, not perform adequately if the training set is too small, or if the data are not representative, is noisy, or is "polluted" with irrelevant features. The strategy, as shown in Figure 3, was built around an Analog Devices Pluto SDR and Raspberry Pi design to maximize the data collection process using low-cost hardware. The Pluto SDR was selected as it is relatively low cost, has Python and Matlab libraries, previous experimental success [45] and encompassed suitable parameters for operating as a data receiver and controlled data transmitter. These parameters include a 12-bit analog to digital converter (ADC) resolution, a 20 MHz maximum attainable bandwidth and frequency and sample rate ranges of 325 MHz–3.8 GHz and 65.2 ksps–61.44 Msps, respectively. The embedded Raspberry Pi device uses a USB connection to power, program and control (both directly and remotely) the Pluto SDR. Python3 can be utilized to develop various signal types for transmission and the "pyadi-iio" library programs the necessary Pluto parameters, such as center frequency, gain, sample rate, etc. The developed wireless data collection system is depicted in Figure 4, where ISM radio frequency (RF) band antennas are utilized. Typically, the chosen RF antenna is a critical element in the wireless data transmission process. Here, a ZigBee Siretta stubby antenna, designed for use in the 2.4–2.5 GHz range, was utilized. This antenna has a 2 dBi gain, vertical polarization, a maximum VSWR of 2.0 and an input impedance of 50 Ω. All experiments utilize the Raspberry Pi/Pluto SDR approach to access received I/Q samples at various center frequencies using a 4 MHz sampling rate.

Wireless signals are received from both commercial and SDR sources. The commercial signals are typical ISM RF band transmissions and include WiFi both with (IEEE802.11ac and IEEE802.11n) and without internet access (IEEE802.11b and IEEE802.11g), Bluetooth, where the advertising channels are targeted, and DIGI XBee ZigBee nodes. SDR sources produce Python3 generated CW and ZigBee signals, that are based on previous Matlab simulations [20], where the Matlab code has been translated to Python3. In Figure 2, the Python3-based ZigBee Pluto SDR transmissions are confirmed to be comparable to commercially transmitted ZigBee signals. This DPX visualization in Figure 2, which is a

digital signal processing software that rasterizes samples into pixel information, allows the presence and operating center frequencies of the necessary signals to be confirmed before commencing a test. The commercial XBee nodes are controlled using the Python3 "digi-xbee" library, powered using a Raspberry Pi and transmit real data by utilizing the Raspberry Pi SenseHat sensor. This operation was validated in an IoT ZigBee testbed in [45]. Hence, the Raspberry Pi embedded device operates both the SDRs and XBee nodes and enables environmental data acquisition using the SenseHat sensor.

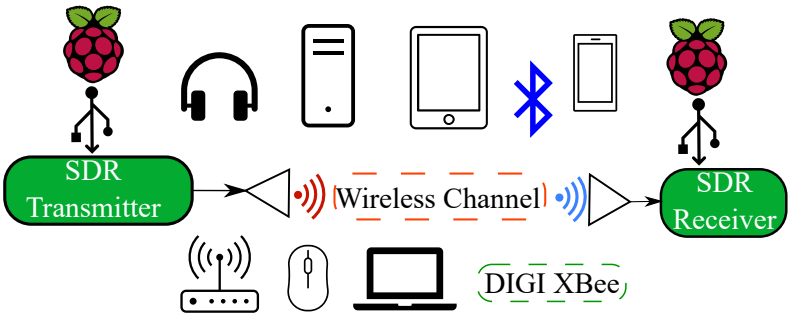


Figure 3. Proposed SDR approach for achieving the desired data strategy for I/Q samples in the ISM RF band between 2.4 and 2.5 GHz.

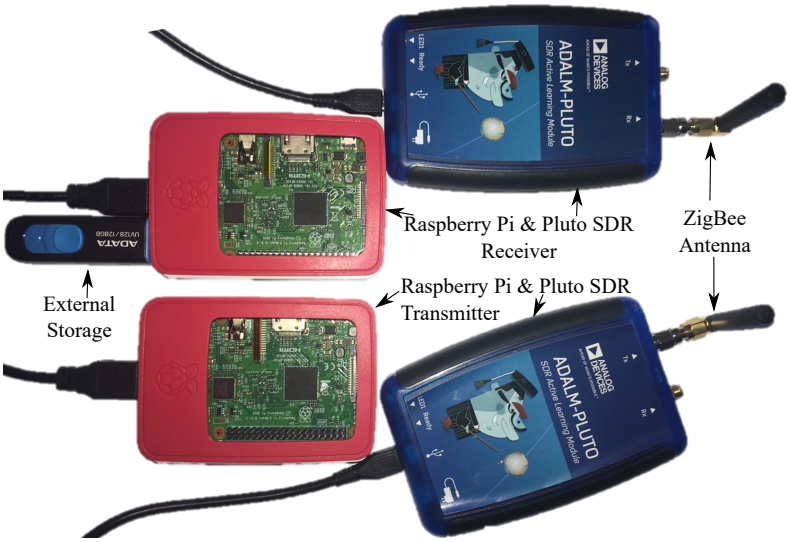


Figure 4. SDR and Raspberry Pi I/Q wireless data collector and signal transmitter for wireless transmissions in the 2.4–2.5 GHz ISM RF band.

The developed data strategy, which incorporates hardware testbeds and commercial devices, produces the necessary I/Q data for off-line analysis and feature investigation. I/Q samples are received, using a 4 MHz sampling rate, in individual data grabs, from which the required I/Q data are identified and extracted. Data collection occurred in a wireless operating environment consisting of changeable service requirements for WiFi and Bluetooth, including the number of connected devices and service load (large download or constant music streaming, for example). A summary of the collected data for training and testing ("unseen") is provided in Tables 1 and 2, respectively, which specifies the center frequency(ies), signal type, approximate number of data grabs and source. Approximately 400 test instances were collected for each of the six signal types and designated as an "unseen" test data set to investigate how the designed classification approach generalizes to new data. Emphasis was given to the XBee ZigBee signals as these are WSN operating signals. However, the number of received signals in each data grab is protocol specific, as data grabs of certain protocols, for example, WiFi, can contain multiple signals for

analysis. The collected data are fully explored in Section 3.2, while the model development results and how each model generalizes to unseen data is described in Section 4.

Table 1. Summary of Collected Data—Model Development.

Signal Type	Center Frequency (MHz)	Total Data Grabs	Source
WiFi	2427	536	Commercial
Router	2447, 2462	270	Commercial
Bluetooth Advertising	2402, 2480	564	Commercial
ZigBee (XBee)	16 ZigBee Frequencies	944	Commercial
ZigBee (SDR)	16 ZigBee Frequencies	1120	SDR
CW	16 ZigBee Frequencies	1120	SDR
Noise	16 ZigBee Frequencies	492	Channel

Table 2. Summary of Collected Data—Unseen Test Data.

Signal Type	Center Frequency (MHz)	Total Data Grabs	Source
WiFi	2427	144	Commercial
Router	2442	107	Commercial
Bluetooth Advertising	2402, 2480	206	Commercial
ZigBee (SDR)	Subset-ZigBee Frequencies	240	Commercial
CW	Subset-ZigBee Frequencies	240	SDR
Noise	Subset-ZigBee Frequencies	210	Channel

Before the data could be analyzed, it needed to be pre-processed. As shown in Figure 3, the wireless channel is open to any accessible wireless transmitter and spurious interference. It is difficult to receive only a specific type of signal when collecting a data grab during a random time period using an unconnected device (SDR). As a result, most data grabs contained multiple signals of interest along with, potentiality, other "unwanted" signals. The data grabs required processing to obtain the correct signal samples before the feature engineering stage. This processing stage was used to reduce the error/noise in the collected data and to discard clear outliers. Without this data processing stage poor-quality, unrepresentative, noisy, or polluted (with irrelevant features) measurements could be employed in the model development process.

The I/Q samples were initially examined in the time domain to visualize the signal patterns (I/Q samples) of interest and to compare them with expected sequences. An example is shown in Figure 5 to illustrate that the signals can be clearly identified compared to the received noise (or operating wireless channel), as signals have an "on-off" nature. Then, the Pluto ADC specifications were used to convert received I/Q samples into the range $[-1, 1]$ from the original Pluto range of $[-2048, 2047]$. The Pluto SDR uses the Analog Devices "AD9363" RF chip, which has a 12-bit ADC, where the 12-bit data from the ADC are stored in the lower 12 bits of the output value and sign-extended to 16 bits. This

conversion supported the development of features having similar ranges, even when the ADC is close to saturation, and produces higher performing machine learning classifiers. The probability density function (PDF), the Tektronix RTSA and the time series plots were utilized to identify any outliers. This analysis permits the removal of outliers and the acquisition of I/Q samples for each received signal type. The chosen sample length for analysis is 1250 I/Q samples and relates to the shortest signal length received, the Bluetooth Advertising channel. Using this designed experimental data strategy, sufficient wirelessly received I/Q data were collected. These data enabled a novel low-order feature extraction investigation, where the feature engineering and developed features are discussed in Section 3.2.

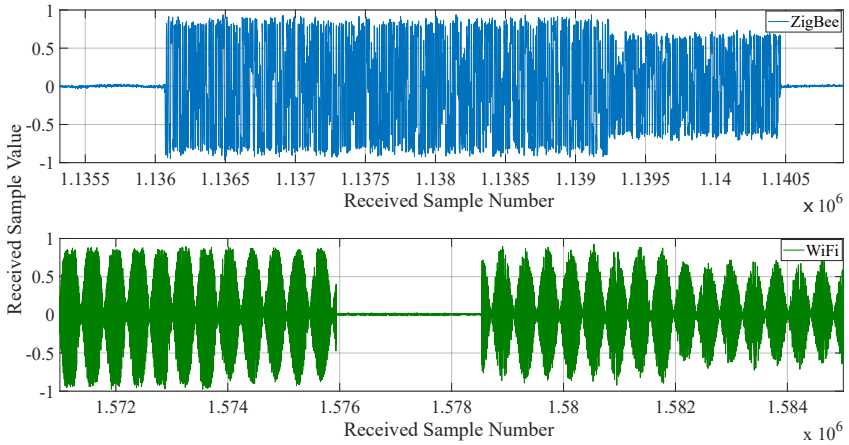


Figure 5. Example time series representations of received wireless signals on the I-channel, which establishes the need for correct sample identification.

3.2. Feature Engineering

The feature engineering in this study initially leveraged previous features extracted from simulated ZigBee data in [20], which focused on identifying error-free ZigBee signals from interference injected ZigBee instances. The wired approach developed in [11] provided additional evidence that a live practical implementation had promise. As per these previous cases, features are entirely based on received I/Q samples and initially extracted from the calculated PDF and statistical analysis of the I/Q data. This wirelessly received I/Q data analysis and feature extraction was implemented in Matlab, while the final implementation targets using available Python3 libraries on a suitable embedded platform.

As per Section 3.1, the time series analysis granted access to the required received I/Q samples and the calculation of associated PDFs. In certain circumstances, it was discovered that the ADC can become saturated and, so, the PDFs can become less distinctive (Figure 6). However, it is envisaged that a specific interference detection approach (using different features) [20] can be used to determine if a jamming signal is present. This topic will be discussed further in Section 4.3, where an interference detection use-case using this paper's features is discussed. Here, wireless devices were non-malicious and, so, the ADC saturation was typical of this operating environment. Here, specific features are developed to identify signals, even when saturation occurs. The automatic gain control (AGC) of the receiver can also affect the produced PDF as two spikes can occur at the limits, since the initial receiver gain changed to a lower value after initial packet reception. However, the calculated PDF is extremely useful in identifying the distinct signals or, at the very least, narrowing the search to a smaller number of possible signals.

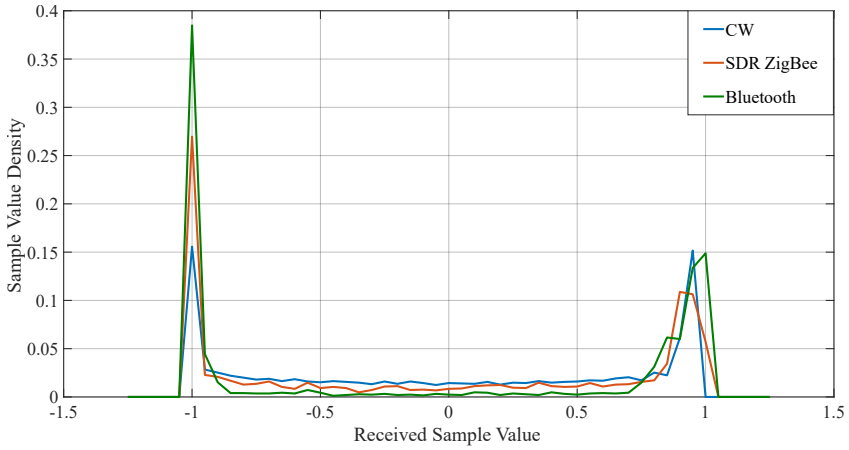


Figure 6. Example PDFs for receiver saturation, emphasizing how PDF become difficult to distinguish and the need for features not dependent on the PDF.

During this study, it was determined that if transmitted signals use comparable modulation schemes (Bluetooth and ZigBee can both use phase-shift keying approaches) or saturate the receiver, further features are required. Thus, the focus diversified to the frequency domain, specifically, the fast Fourier transform (FFT). By utilizing the FFT, it was envisaged that the larger bandwidth signals could be identified from signals with a smaller channel width. For example, ZigBee has a 2 MHz wide channel while Bluetooth channels are 1 MHz in width. This allows, potentially, for a categorization to be made from the FFT of the received I/Q samples. This method exploits the information gained from using the RTSA, and its associated DPX technology (Figure 2), to visualize the spectrum before grabbing the data. Statistical analysis expanded the feature set, to gain additional understanding of the received signals. The time series representation produces features associated with, for example, the number of zero crossings, Hjorth parameters [43], mean value and variations of available Entropy functions.

Figure 7a–d present the calculated averaged signal PDFs for the received signal types including noise, WiFi, router (no internet access), Bluetooth Advertising Channels, CW and ZigBee. Distinct features are identifiable in most cases. The exhibited PDFs are the averaged PDFs for each signal across all available data and ZigBee center frequencies [6]. The individual data grabs, as defined in Section 3.1, can contain more than one signal and, so, multiple extractions are permitted, where possible. As the sample length was chosen to be 1250 I/Q samples, which was associated with the received length of the Bluetooth signals, other received signals were also divided into 1250 sample segments. This process meant that the start, middle and end of the packets were accounted for and that a specific part of a signal packet is not required, only 1250 samples from the channel are required. Calculating the PDFs using this method provided an opportunity to find the PDF distribution for each, even in the presence of spurious wireless transmissions. The PDFs for some of the signals include multiple variations of what the PDF can exhibit, for example, the Bluetooth PDF displays both the receiver saturation and non-saturation cases, which is also reflected in the PDFs for the CW and ZigBee signals.

Based on the above feature investigation, a set of 28 features, which could classify the received I/Q samples, was developed. This initial feature set included some comparable features and some features were included to evaluate their potential. However, by focusing on calculating the early results using this set, the informed decision to progress to model optimization and generalization performance, or remain in the feature engineering stage, could be made. As presented in Section 4.1, these 28 features provided the necessary classification results to warrant model development and optimization. This initial feature set contained features extracted from the PDFs, including the area in specific bins (centre and sides), averaged and combined area of the two side bin ranges, the zero bin value, the number of non-zero entries and the maximum value in the PDF. The feature set was expanded

by extracting statistical features from the time series representation including the Hjorth parameters of activity, complexity and mobility, the absolute mean value, the absolute maximum value, root mean square value, the standard deviation and different variations of entropy functions (four in total). The Hjorth parameters are particularly interesting as they have been useful in medical signals but have not been applied to communication signals before. Other time series features included skewness, kurtosis, number of peaks and number of zero crossings, while additional PDF features included the number of peaks and a function characterizing the PDF's uniformity. The FFT provided features that analyzed the received bandwidth, power and number of peaks over a predefined threshold.

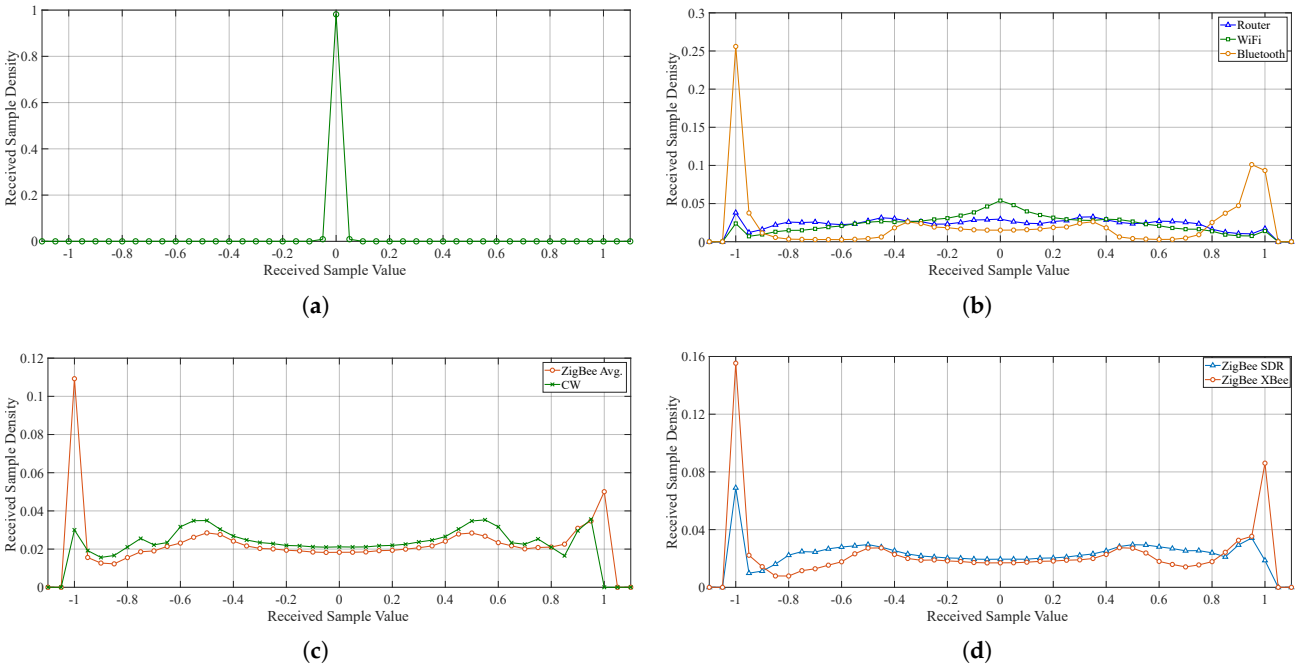


Figure 7. Calculated PDFs of the wirelessly received I/Q samples for a subset of signals operating in the 2.4 GHz ISM band. Signals are split into groups for ease of visualization. (**a**) The calculated PDF for the inactive noise channel in the operating environment, illustrating the relatively quite channel. (**b**) The computed PDFs for received WiFi, router without internet connectivity and Bluetooth advertising signals. (**c**) The PDFs for the received CW and ZigBee signals for various power levels. (**d**) The calculated PDFs of the received legitimate ZigBee-XBee and ZigBee-SDR.

Some features will have more distinctive characteristics and, so, will provide more useful information. Hence, the initial feature set was investigated to determine the most important/useful features. Both the theory underlying what each feature was supposed to highlight and built in Matlab functions for estimating predictor (feature) importance were used to optimize the feature set. For this purpose, the Random Forest supervised machine learning approach was adopted as it is envisaged that its associated ensemble concept will be advantageous in identifying the correct received signal, as is shown in Section 4. Two Matlab functions were used with the Random Forest specifications applied, namely, TreeBagger and fitcensemble. Both functions were investigated using different numbers of trees with the full number of predictors available at each decision point. Hence, the model was trained using the available data and the out-of-bag predictor importance function was applied to each trained model. This was repeated for a number of iterations for each function and the final result combined the averages of the two individual approaches. The results of this feature importance investigation are supplied in Figure 8, where fourteen features were chosen, based on the results and the theory underlying the features, since, to be useful, the model will need to generalize to unseen data. The final feature set encompassed: (1) number of non-zeros entries of PDF (2) area in centre PDF bins (3) area in left PDF side (4) PDF centre bin value (5–7) the Hjorth parameters (activity, mobility

and complexity) (8) the number of zero crossings (9,10) two entropy representations (11) absolute mean of samples (12) RMS value of samples (13) number of unique FFT thresholds, and (14) a unique FFT function that provides a differentiation between signals of different bandwidths. The final set of features is displayed in Table 3, where the initial feature number corresponds to Figure 8 and the final feature number corresponds to the input data, X, in the final models as developed in Section 4.

Table 3. PDF, Time and Frequency Domain-Optimized Features Extracted from the Raw Received I/Q Data.

Domain	Initial Feature Number	Final Feature Number	Feature Description
PDF	1	1	Number of non-zeros entries of PDF
	2	2	The area in the center bins ([−0.1:0.1])
	3	3	The area in the left hand side bins (<−0.1)
	28	14	The center bin (0) value
Time	8	4	Hjorth parameters [43]-Activity (Sample Variance)
	10	5	Absolute mean value
	12	6	The root-mean square (RMS) value
	13	7	Hjorth parameters [43]-Mobility
	14	8	Hjorth parameters [43]-Complexity
	16	9	Shannon Entropy-using use specific approach
	18	10	Matlab's "approximateEntropy" function
	24	12	Number of zero crossings
Frequency	21	11	Number of FFT points over a predefined threshold
	27	13	Unique function that uses the FFT points to estimate signal bandwidth

The final feature set contains elements which have theoretical justifications explaining why the corresponding importance estimates were high. For example, the PDF features classify the received signal into a specific pattern which can be used to identify a specific signal type directly (for example, noise or a group of signals). The FFT-based features provide access to information regarding the bandwidth of the received signal, which can be used to help identify tones or signals with larger bandwidths and similar modulations schemes. Currently, the FFT size is the next power of two above the signal length, meaning an FFT of length 2048 was applied. This will need further investigation to see if this can be reduced, given that the motivation is to apply this methodology to low-power edge devices. Finally, the time series low-order statistical features help to identify the correct samples that correspond to the distinctive signal, as CW differs from IEEE802.11, for example. Section 4 verifies that the features are effective by focusing on the generalization error of the developed machine learning model. The feature count remains at fourteen, as it is envisaged that the extra features will provide additional options in different wireless environments and allow the methodology to be applied with a greater probability of success. This aspect is discussed further in Section 4.3, where the applicability of the fourteen features to different scenarios is identified.

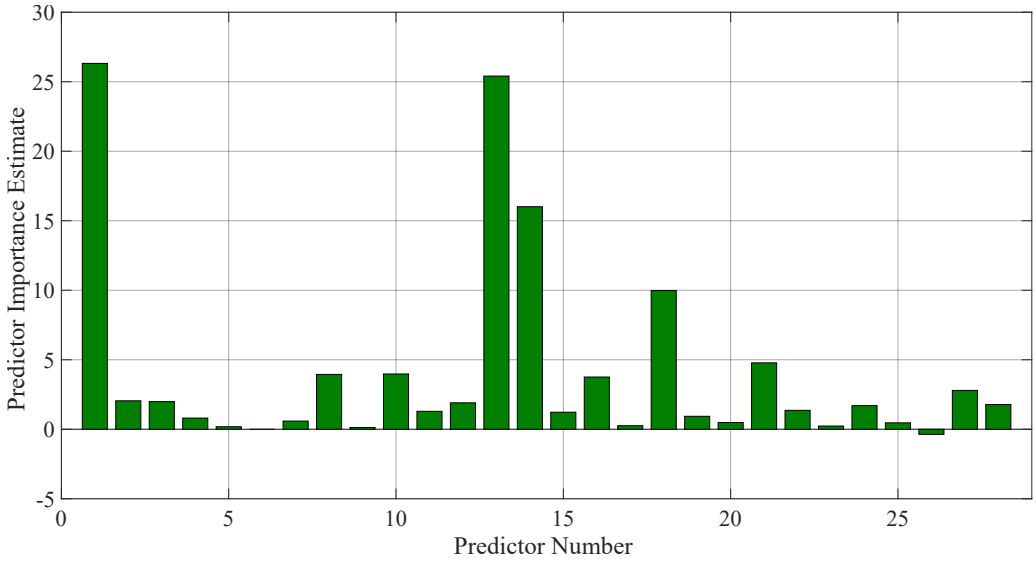


Figure 8. The estimated predictor importance measurements using the built in MatLab functions averaged across multiple iterations, feature depths and number of trees using two distinct functions-TreeBagger and Fitcensemble.

4. Results and Discussion: Signal Classification

The features developed in Section 3.2 are used to classify received samples into one of six signals/channel types—noise, WiFi (IEEE 802.11ac/n), a WiFi router without internet access (IEEE802.11b/g/n), a Bluetooth advertising channel, CW and ZigBee. A feature-based approach was chosen as it can provide a relatively low-complexity solution with near optimal performance [24], as discussed in Section 2. As the data from signals operating in the 2.4 GHz ISM RF band can be clustered relativity close-by, supervised machine learning techniques that incur relatively fast optimization and training times (compared to deep learning, see Section 4.2) are chosen. The fundamental approaches chosen include SVMs, Random Forest, k-NNs, Gaussian Naive Bayes and XGBoost [46], whilst a feature-based DNN is studied to assist the validation of the selected approach. Since an emphasis is applied to developing a single multi-class classifier rather than multiple binary classifiers, the majority of the work focuses on multi-class classification. This study comprises a classification problem since the required output is categorical (discrete). In each supervised learning approach, the algorithm is employed to learn the mapping function (f) from the input variable (x) to the output variable (y); that is $y = f(X)$. Each algorithm attempts to estimate the mapping function (f) from the input variables (x) to discrete or categorical output variables (y).

An important characteristic to identify is the designed machine learning model, which is based on data collected in a specific wireless environment under a unique set of channel fluctuations. The proposed machine learning algorithms are adopted to fully validate the suitability of the developed feature set for classifying wireless ISM band signals. Even though more modern deep learning approaches are available, the results in this study, through a comparative approach, indicate that fundamental approaches are still fit for purpose. They incur relatively fast optimization times, compared to deep learning, have low complexity and are shown to generalize well to new data. Thus, this investigation's overall strategy of data collection, feature extraction and model development can be applied to various deployed communication environments, where extensive time and computational resources are rare [40]. As a result, this section provides results identifying the value in the developed feature set and how well known and extensively studied machine learning approaches continue to be effective, especially when the desired deployments require off-line analysis to be as fast as possible.

The collected data were split into training and testing datasets, where test data included an unseen testing dataset collected after the original. This allowed an estimate of

the error rate of new cases, known as the generalization error (or out-of-sample error), to be found. This generalization error value outlines how the designed model would perform on instances it never encountered before and, so, is used as the main verification of model suitability to the problem and potential real-world operation. For this discussion, it is worth noting that received wireless signals are, typically, unique on reception. The wireless channel varies over time and signals, generally, interact with other signals and obstacles differently on each signal transmission. The concept of signals incurring different interactions once transmitted is acceptable and data instances can be perceived as unique. Hence, testing data are relatively unseen to the training instances used in model development. However, as one packet of data can be divided into multiple instances and used in either training and/or testing, some test points may not be unseen. The test data discussed in Section 3.1 ensure the use of unseen data.

The conventional data split between training and testing data of 80%:20% was applied, respectively. This data split was chosen as the dataset was of a reasonable size for experimentation, meaning that tests could be completed in a suitable time-frame for model development and, so, a validation set was not required. Additionally, an "unseen" test dataset was collected to investigate how the developed model generalizes to completely new data that were not used during model training. The data for each signal type were randomly split in the ratio of 80%:20% by initially creating a column array of unique random positive integers in the range of the number of individual signal instances. This array of random numbers was then split, using the 80:20 ratio, and used to select the training and testing instances. The sampling bias of the data split procedure was examined by calculating the percentage of each classification type in the datasets, on the scale of $0.0 \rightarrow 1.0$. The results are provided in Table 4, where it is clear that the percentage split is maintained across all developed datasets. Approximately 400 instances were added to each test signal type as unseen data, but there is a slight variation which leads to a small deviation in the sampling for test data, including the additional unseen data. These datasets are used in the following Matlab and Python3 investigations to determine whether the extracted features are suitable for wireless signal classification using machine learning approaches.

Table 4. Sampling Bias Comparison in Developed Datasets.

Signal	Model Data	Training Data	Testing Data	Testing + Unseen Data
Noise	0.0938	0.0938	0.0938	0.1335
WiFi	0.1511	0.1511	0.1511	0.1601
Router	0.1077	0.1077	0.1077	0.1405
Bluetooth Advertising	0.1099	0.11	0.1097	0.1372
CW	0.1805	0.1805	0.1805	0.1669
ZigBee	0.3569	0.3569	0.3571	0.2617

4.1. Initial Matlab Investigation

Analysis initially focuses on Matlab machine learning functions, specifically the "fitcsvm" and both the "TreeBagger" and "fitcensemble" functions for SVM and Random Forest models, respectively. Adaptive boosting is also investigated, based on an optimization study, and is applied in the "fitcensemble" case. Matlab operated as a continuation from the feature extraction environment and granted easy access to the I/Q data.

Based on previous work [11] and insights into SVMs, clearly multiple binary classifiers are required to classify the different signal types. This means potentially six classifiers using the one-versus-all method or fifteen using the one-versus-one method. An additional logic decision stage would also be required based on the SVM outputs [47]. Multiple models would, typically, increase the computational load and be time consuming, leading

to potential implementation problems on resource-constrained WSN/IoT edge devices. Hence, a single multi-class classifier was the preferred approach. As a result, one SVM ZigBee versus all other signals binary classifier was developed for a performance comparison. As the RTSA-confirmed SDR-transmitted ZigBee signals sufficiently matched XBee transmitted signals (see Figure 2), both were classified as ZigBee. This combination provided a greater ZigBee operating range, as the SDR transmit power was controllable, whereas the XBee's had limited control.

The aim of the binary classifier was to identify received samples as either being ZigBee or some other 2.4 → 2.5 GHz ISM band signal. In essence, this is taking the WSN signal focused binary classifiers from the one-versus-all classification method. This approach would be suitable for potential applications of this methodology, which will be discussed later. The results of this binary classifier development are shown in Table 5, where four kernels were investigated using the "fitcsvm" function. The results focus on using all available testing data, including the additional "unseen" data, and datasets containing both 28 and 14 features were applied. Table 5 shows that the reduction in features enhanced the model's performance, which reiterates the idea that only the most useful features were retained during feature optimization. The fourteen features did incur an increase in training time for specific kernels. However, in this study, prediction time is the more important as model training time can be carried out off-line. The associated prediction times for the reduced feature set did not increase and the model error was reduced, resulting in a positive outcome when applying the reduced feature set. The optimal kernel was chosen based on the achieved training and prediction times and model error, as per Table 5. The kernel that produced the highest performance was the third-order polynomial, which provided an area under the curve of approximately 0.9997 when using the 14 element feature set. This means the receiver operating characteristic (ROC) curve is near the optimum for a binary classifier. The confusion matrix and ROC for this kernel are provided in Figure 9a,b respectively, which shows that the errors were false positives. These results indicate that the SVM can identify ZigBee signals with a relatively low generalization error and, so, could be used to identify ZigBee signals in new environments. This is advantageous as the methodology discussed in this study could then be applied to multiple operating environments. The SVM performance will be compared to identifying the same ZigBee signals in the subsequent multi-class approaches.

Table 5. SVM Signal Classification Generalization Error Results: ZigBee versus All.

Kernel	Training Time (ms)	Average Prediction Time (ms)	Test Data Error (%)	10 Fold Cross Validation Error (%)	AUC
28 Features and All Test Data					
Linear	520	1.85	0.219	0.173	0.9969
Gaussian	388	1.89	0.176	0.025	0.9975
Radial Basis Function	383	1.85	0.176	0.025	0.9975
Polynomial (3rd Order)	355	1.83	0.132	0.039	0.9985
14 Features and All Test Data					
Linear	508	1.84	0.329	0.4863	0.9956
Gaussian	390	1.83	0.066	0.00	0.9993
Radial Basis Function	534	1.84	0.066	0.00	0.9993
Polynomial (3rd Order)	384	1.81	0.044	0.00	0.9997

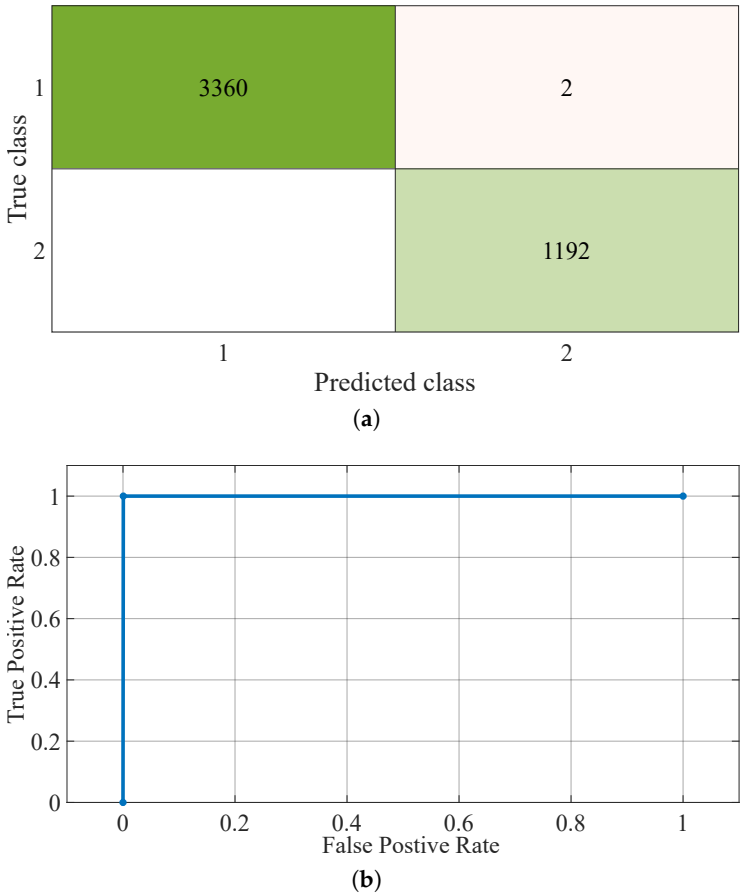


Figure 9. In-depth results for the selected third-order polynomial kernel for the ZigBee vs. all SVM classifier. (**a**) The confusion matrix for testing data. (**b**) The ROC.

However, the requirement for creating multiple binary classifiers, either using the one-versus-all or one-versus-one method, and the associated complexity, which rises as more signal types are added to the dataset, is undesirable. Therefore, a single multi-class model is targeted, where previous success was established based on wired signals and the Random Forest algorithm in [11]. Generally, if other signals are discovered and require classification, additional samples can be included in the dataset and the multi-class model retrained. A similar process is available for expanding the developed model to include the 2.4 GHz ISM RF band signals operating in different environments. In comparison, the binary classification would require additional models to be developed and the associated logic decision approach to be updated. Hence, a multi-class model is advantages, as developed models have the potential for expansion to include 2.4 GHz ISM band signals operating in various operating environments and other signal types.

Initially, the Random Forest [14] classifier was chosen, as it fulfilled the multi-class classification requirement for a subset of wired signals in [11]. The Matlab "TreeBagger" and "fitcensemble" functions were utilized to show that performance is not limited to a specific Matlab approach. Additionally, the fitcensemble approach was used to find an optimal approach for this methodology. As a result of this optimization investigation, an adaptive boosting algorithm was examined in the form of the Matlab "AdaboostM2" to see if performance could be enhanced. AdaBoostM2' is an adaptive boosting dependent ensemble algorithm, where the "M2" is a Matlab specification for multi-class operation. Adaptive Boosting, or "AdaBoost", is a specific method for a predictor (decision tree) to correct its predecessor by focusing on the training instances that the predecessor underfitted, resulting in new predictors concentrating on the hard cases. This is achieved in AdaBoost by initially training a base classifier (a decision tree, for example) and making predictions on the training set. The algorithm then increases the relative weight of misclassified training

instances and trains a second classifier, using the updated weights, and again makes predictions on the training set, updates the instance weights, and so on. These approaches are analyzed across a range of feature depths and number of decision trees, focusing on the generalization error, as available test instances were predominantly comprised of unseen data.

Each Random Forest investigation was iterated over the number of grown trees and the feature depth available at the decision points, where the random seed was set to the same value at the start of each interaction, for reproducibility. The main generalization error results for the Random Forest approaches are provided in Figure 10, where the total available testing data was used in each case ("TreeBagger" and "fitcensemble") for the original 28 features and the optimized set of 14 features. In each model, the achievable generalization error was less than 2.5% and, in most cases, the smaller the feature depth the smaller the error. The training and average analysis (prediction) times were also investigated. It was determined that both the training and average prediction time increase, as the number of grown trees and/or feature depth increase. These trends are maintained across all of the investigated approaches and show that there is a trade-off between reducing the error and minimizing the prediction and training times. Typically, the training time increases with the number of features to consider at each decision node when looking for the best split. These trends are expected for the Random Forest approach. As the training can be completed off-line on a resource heavy device, the main focus is on the average prediction time.

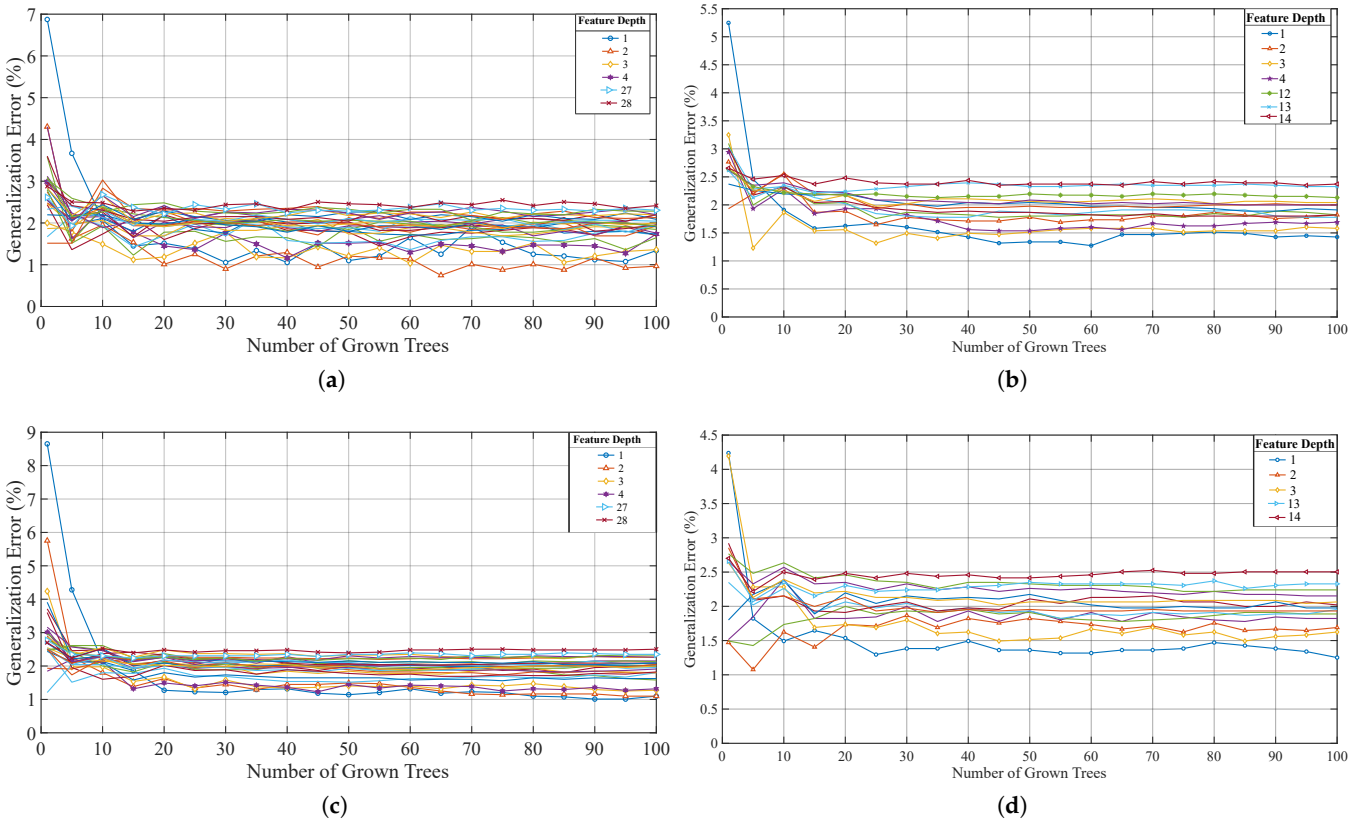


Figure 10. The generalization error results for the developed Random Forest approaches for a range of grown trees and feature depths (only certain depths marked as most follow a similar trend). (**a**) "TreeBagger" function and the original set of 28 features, where the error stabilizes between 1 and 2.15% as the number of trees increases. (**b**) "TreeBagger" function and the optimized set of 14 features, where the error stabilizes between 1.25 and 2.5% as the number of trees increases. (**c**) "fitcensemble" function and the original set of 28 features, where the error stabilizes between 1 and 2.1% as the number of trees increases. (**d**) "fitcensemble" function and the optimized set of 14 features, where the error stabilizes between 1.5 and 2.5% as the number of trees increases.

The Random Forest results indicate that a certain small number of features can be used to produce the prediction. However, that approach has a high probability of overfitting the model to the available data and reducing the probability of successfully generalizing to new data or new wireless operating environments. Therefore, the optimized set of 14 features has, even if it incurs a slightly reduced performance, the best potential to enable the developed methodology to generalize and be useful in new operating environments. Suitable optimization and performance enhancing model development techniques such as boosting, are also options to help the overall designed approach to adapt to new signals and help reduce the possibility of overfitting.

Based on the "HyperparameterOptimizationOptions" input into the fitcensemble function, the available training data were used to obtain an optimized set of parameters. The results indicated using the "AdaBoostM2" function with a learning rate of 0.61992 and a minimum leaf size of 3. This setup was investigated for different numbers of decision tree learners in the range [1, 5, 10, 15, ..., 100] and feature depths ([1:14]) available at the decision points. The results are visualized in Figure 11, where the lowest error (1.2956%) occurred for the combination of sixty learners and a feature depth of seven and seventy learners and a feature depth of eight. These results indicate the importance of using the 14 element feature set and not reducing the features based on the Random Forest results using basic hyper-parameter optimization. The sixty learners incurred a training time of 749.1 ms and an average prediction time of 20.94 ms. In contrast, the seventy learners incurred a training time of 993.58 ms and an average prediction time of 24.46 ms. As the training and analysis times were shorter for the smaller number of learners, it was chosen as the optimal "AdaBoostM2" Matlab combination. A more focused set of results was obtained, specifically the confusion matrix for this combination is depicted in Figure 12a. The majority of the errors occur when classifying between the two different IEEE802.11 signals. Thus, the model has generalized well to signals from different sources and has high accuracy in classifying the wireless signals being received and generalizes well to "unseen" data. This result further validates the feature set and methodology that has been developed in this study. Additionally, as this approach focuses on the hard instances and aims to improve on predecessors, it has a higher likelihood of aiding this methodology in adapting to different wireless operating environments.

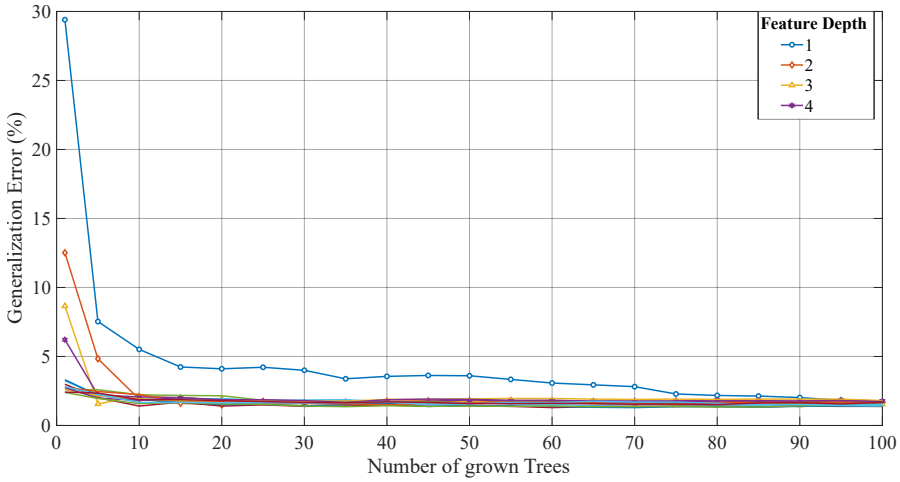


Figure 11. The generalization error results for the AdaBoostM2 ensemble method with a minimum leaf size of 3 and a learning rate of 0.61992, for a range of decision tree learners and feature depths. The error stabilizes between 1.2 and 1.9% and the unique feature depth trends are marked.

This initial Matlab investigation section focused on determining the usefulness of the extracted features (Table 3) and if applying machine learning techniques was applicable to this classification problem. The results indicate that ensemble methods are suitable to this classification problem and dependent methods are optimal. However, for WSN/IoT deploy-

ments, a more lightweight machine learning programming language is required. As a result, the leading machine learning programming language, Python, needs to be investigated.

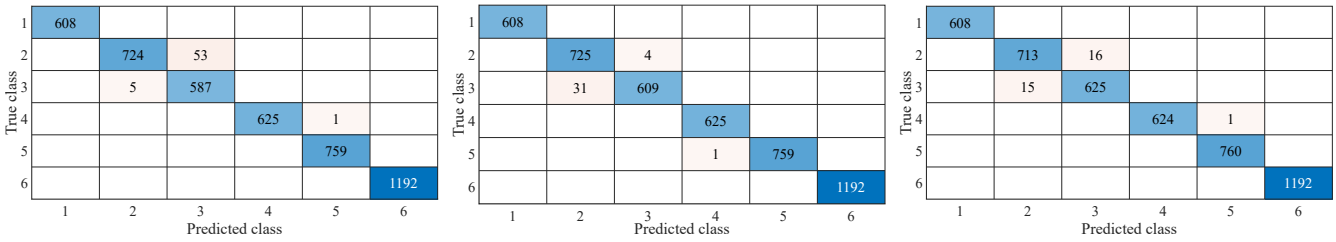


Figure 12. Confusion matrix for the (**a**) designed Matlab AdaBoostM2 method using sixty decision tree learners, a feature depth of seven, a minimum leaf size of three and a learning rate of 0.61992; (**b**) designed XGBoost method, which produced a generalization error of 0.7905%; (**c**) designed DNN, which produced a generalization error of 0.7027%. The predictors are as follows: (1) noise (2) WiFi (3) router (4) Bluetooth advertising (5) CW and (6) ZigBee.

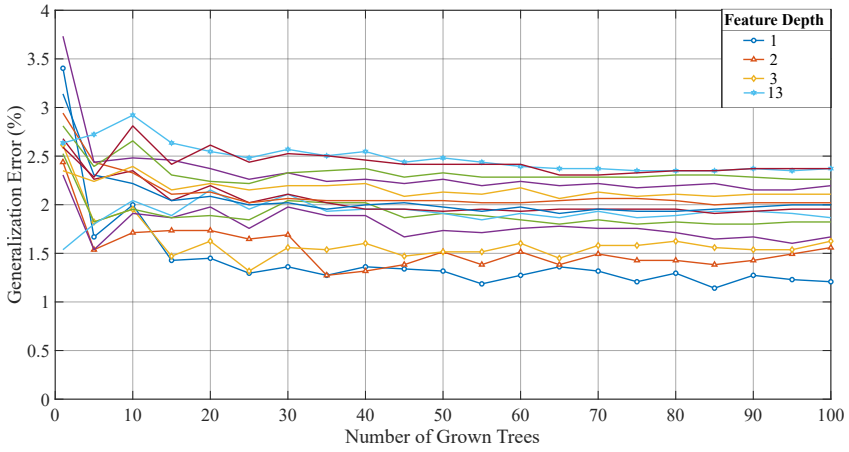
4.2. Python Investigation

The main objective of this work is to permit decentralized computation and allow specific embedded devices to identify signals present in their surroundings. This approach can lead to methods to increase security by monitoring the wireless channel and adapting to changes in real time, such as frequency hopping when a CW jamming wave is detected, for example. To initially investigate this embedded implementation and to employ the leading machine learning programming language, the developed classification approach was translated to Python3 and computed on a RaspberryPi embedded device. This implementation focused on the optimized 14 features and the same training and testing datasets as the Matlab approach. As a result, this section identifies the optimal machine learning approach for this study's wireless signal classification problem.

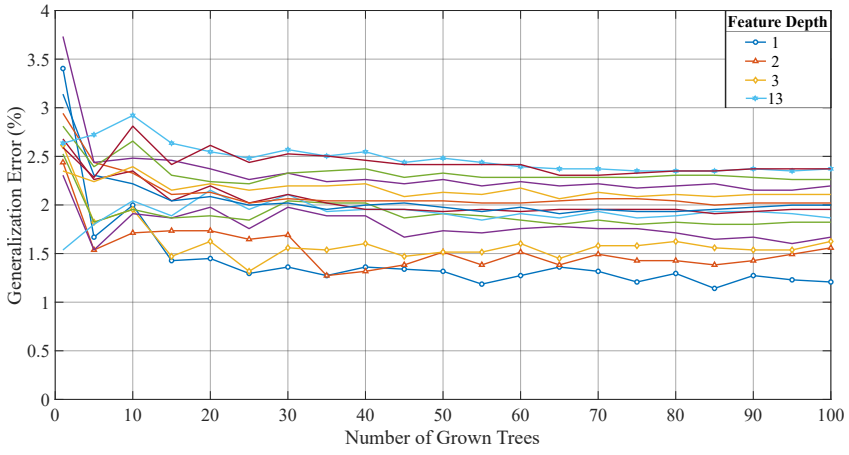
For this initial embedded device investigation, the "scikit-learn" machine learning library was utilized, specifically the "RandomForestClassifier" function. A summary of the results of this approach is shown in Table 6 and in Figure 13, where the Raspberry Pi results (Figure 13b) are compared with an implementation on the same Desktop PC that produced the Matlab results (Figure 13a). A Dell XPS8930 was used as the PC to compute the results, where 16 GB of RAM and an Intel i7 processor (3 GHz) were available, while the Raspberry Pi (as shown in Figure 4) has 1 GB of RAM and a quad-core Broadcom Arm Cortex A53 processor (1.4 GHz). Table 6 shows that the embedded device had a larger training and prediction time compared to the PC. However, the prediction time of the embedded device is acceptable for real-time operation, as the average prediction time is 0.0679 ms. The achieved model error is the same for both devices and the training time increase can be rectified by training on the resource extensive device and transferring the trained model to the embedded Raspberry Pi. These results show that the Python3 approach aligns with the Matlab results, while providing a faster implementation time and further validating the extracted feature set. The Python3 "RandomForestClassifier" function was analyzed further by investigating additional metrics using a specific random state including the number of trees, maximum number of predictors, maximum tree depth, the maximum number of samples, minimum number of samples required to split an internal node and the minimum number of samples required to be at a leaf node. In total, 197,568 iterations were completed and the optimum generalization error was 1.098%, which shows that the Python3 investigation benefits from delving deeper into the model development. As the Raspberry Pi achieved the same model error as the PC, the remaining aspects of this investigation can be processed on the PC device to provide efficient model development, as the Python3 models will achieve similar model error results on the Raspberry Pi device.

Table 6. Random Forest Classification Results: Device Comparison for Wireless Data.

Device	Predictor Depth	No. of Trees	Training Time (ms)	Avg. Prediction Time (ms)	Test Data Error
Raspberry Pi 3-B	1	85	1564	0.0679	1.142
Specifications:	1 GB of RAM and Quad-core Broadcom BCM2837B0, Cortex-A53 CPU @ 1.4 GHz				
PC	1	85	139.6	0.005	1.142
Specifications:	Dell XPS8930, 16 GB of RAM and an Intel i7-9700 CPU @ 3 GHz, 8 Cores				



(**a**)



(**b**)

Figure 13. The generalization error for the designed scikit-learn Python Random Forest method using variable number of grown decision trees, variable feature depths and an unspecified maximum depth for each tree, where the error stabilizes between 1.14% and 2.57%. The same python scripts were implemented on (**a**) a Desktop PC and (**b**) a Raspberry Pi 3 Model B.

To further aid in minimizing the effect of overfitting, the dependent decision tree-based "XGBoost" [46] algorithm was investigated across typical parameters that enable the training of a decision tree boosted model. This algorithm stands for "eXtreme Gradient Boosting" and, in general, produces strong learners based on the correction of errors produced from weak learners. Based on the improvement seen in the Matlab investigation when applying AdaBoost, which increased model performance, boosting (dependent ensemble approaches) is seen as an appropriate method for creating an accurate and

strong classifier from a set of weak classifiers. XGBoost, which applies a gradient boosting approach, was the chosen Python3 boosted tree approach. For this investigation, several parameters were optimized, including (1) the number of decision trees, (2) the applied learning rate, (3) the maximum tree depth, (4) the minimum child weight, (5) the percentage of available data used per decision tree, (6) the applied booster algorithm, (7) the sub-sampling percentage of the feature columns and (8) the applied minimum loss reduction when determining if a further partition is required. The XGBoost approach produced the highest accuracy (and lowest generalization error of 0.7905%) in this study before developing deep neural networks. The confusion matrix for the designed approach is seen in Figure 12b, where the majority of errors occur between classifications of different IEEE802.11 protocols. The final algorithm contained five trees, a learning rate of 0.8, a maximum tree depth of 10, a minimum child weight of 2, used 95% of available data per tree, used the "gbtree" booster, sub-sampled 75% of the feature columns and applied a minimum loss reduction of 0.5 when determining if a further partition is required. This produced a smaller error and training time, when compared to the AdaBoost approach in the Matlab investigation (see Table 7), using the same data. By using XGBoost, an error reduction of 38.98% was achieved (1.2956% $\rightarrow$ 0.7905%), where the improvement occurred in the classification between the separate IEEE802.11 signals. This result indicates that gradient boosting is more beneficial for this classification problem and that using the decision tree approach is applicable.

Table 7. Random Forest Selection Evidence-Supervised Approaches.

Algorithm	Lowest Achieved Error (%)	Training Time	Iterations
XGBoost/SVM	0.527	122.54 ms	n/a
DNN	0.7027	1937.04 s	400
XGBoost	0.7905	78.1 ms	400,000
Random Forest	1.098	265.57 ms	197,568
AdaBoost (Matlab)	1.2956	749.1 ms	n/a
K Nearest Neighbors (17 Neighbors)	3.3816	26.93 ms	n/a
K Nearest Neighbors (20) (20 Neighbors)	3.4695	15.6 ms	1152
K Nearest Neighbors (within Radius)	3.6232	35.934 ms	7166
Gaussian Naive Bayes	5.907	3 ms	17
Nearest Centroid	9.222	2 ms	8

To further validate choosing the XGBoost approach for this multi-class classification problem, other distinct machine learning approaches were briefly examined. Each model is investigated across a range of suitable available parameters, specific to each machine learning approach. Analyzed algorithms included the "scikit-learn" neighbors-based classification functions, namely "KNeighborsClassifier", "RadiusNeighborsClassifier" and "NearestCentroid", and the "scikit-learn" Gaussian Naive Bayes ("GaussianNB") function.

For both the "KNeighborsClassifier" and "RadiusNeighborsClassifier" investigations, the examined algorithms were "ball_tree", "kd_tree", and "brute", the associated leaf size, where applicable, was an element of the set [5, 10, 20, 30, 40, 60, 80, 100], the weight function used in prediction was either "uniform" or "distance" and the exponent for the Minkowski metric was either 1 (equivalent to using Manhattan distance) or 2 (equivalent to using Euclidean distance). For the weights, "uniform" results in all points in each neighborhood being weighted equally and "distance" means points are weighted by the inverse of their

distance, meaning closer neighbors of a query point will have a greater influence than neighbors which are further away. For the "KNeighborsClassifier" the number of neighbors was an element of the set [1, 2, 5, 10, 20, 40, 60, 80, 100]. For the "RadiusNeighborsClassifier" approach, the range of parameter space to use was an element of the set [85, 90, 95, 100, 110, 120, 140, 150], where 85 was the lowest radius that ensured each instance had at least one neighbor for each of the other parameters setups. The "NearestCentroid" approach was investigated, where each class is represented by its centroid, with test samples assigned to the class with the nearest centroid. The examined metric for calculating the distance between instances in a features array included: {*cityblock, cosine, euclidean, haversine, l1, l2, manhattan, nan-euclidean*}, which were all of the available distance metrics provided in the "metrics.pairwise.distance_metrics()" section of the "sklearn" library. For this approach, the centroids for the samples corresponding to each class are the points from which the sum of the distances (according to the metric applied) of all samples that belong to that particular class are minimized. If the "manhattan" metric is provided, this centroid is the median and for all other metrics, the centroid is set to be the mean.

The optimal parameters for the "KNeighborsClassifier", based on the parameter sets above, corresponded to using twenty neighbors, any of the three algorithms, a leaf size of 5, the uniform weight function and the Manhattan distance. This optimized parameter set produced an error of 3.4695%. However, as an odd number of neighbors negates the condition of multiple classes attaining the same number of maximum votes, this optimization of 20 neighbors allowed for further investigation. The odd numbers around twenty were investigated, while maintaining all other optimal parameters. The results indicated that seventeen neighbors was the most optimal result and produced an error of 3.38%, which indicates better performance compared to the even number of neighbors, but for an increase in training time (Table 7). For the "RadiusNeighborsClassifier" approach, the optimal parameters are a radius of eighty-five, any of the three algorithms, a leaf size of 5, the distance weight function and the Manhattan distance. These parameters resulted in a calculated generalization error of 3.6232%. The "GaussianNB" was briefly investigated across an array of values for the portion of the largest variance of all features that is added to variances for calculation stability. The examined values were [1×10^{-15}, 1×10^{-14}, ..., 1×10^{-10}, 2×10^{-10}, 5×10^{-10}, 1×10^{-9}, 2×10^{-9}, 5×10^{-9}, 1×10^{-8}, 2×10^{-8}, 5×10^{-8}, 1×10^{-7}, 1×10^{-6}, 1×10^{-5}]. The optimal approach applied a portion of 1×10^{-10} or lower, and produced an error of 5.907%. These results are provided and compared to the other investigated algorithms in Table 7, where the training times are provided as a complexity metric. As shown in Table 7. the smaller training times, typically, lead to less accurate models which are less complex to deploy.

A valuable insight is gained from comparing the XGBoost approach to the SVM binary classifier. When compared to the ZigBee-versus-all case, this multi-class classifier achieved similar, if not better, performance to the SVM. In fact, all ZigBee signals were correctly classified in each case, but no signals were misclassified as ZigBee when XGBoost was applied (Figure 12b). The SVM classifier misclassified two instances as ZigBee (Figure 9a). Yet, the multi-class approach can be assisted by a binary classifier. The majority of the XGBoost errors occurred when classifying between the different IEEE802.11 signals. A SVM for this one-versus-one case was developed to examine if a higher performance was achievable. The designed SVM produced 53 errors (3.8714%) when using the same third-order polynomial kernel as the ZigBee case. By adopting the radial basis function the error was reduced to 23 misclassifications (or 1.68%). The SVM obtained a 34% error reduction in the classification of the IEEE802.11 signals compared to XGBoost.

This comparison discovered the optimal approach for using these fundamental algorithms and the developed feature set. The Random Forest XGBoost method is optimal for all but the classification between IEEE802.11 signals. Hence, for maximal performance, if an IEEE802.11 signal is detected, the data instance is passed to a separate binary SVM classifier focused on IEEE802.11 signals. This achieves optimal performance for the mini-

mum amount of designed classifiers that generalize well to unseen data, when resource management is key.

However, by focusing entirely on a resource-constrained operation, the most optimal approach may be neglected. Thus, deep learning was investigated to explore how traditional techniques compare to a fully connected neural network in terms of generalization error, computation load, training times, parameter optimization and hardware resources. The DNN was optimized, using TensforFlow and the "KerasClassifier" function, over the number of epochs, batch size, optimizer, number of hidden layers and neurons. When selecting the optimal parameters, consideration was given to the training time, average prediction time and required resources. The batch size was examined using the values $[5, 10, 20, 40]$, the number of epochs were $[100, 200, 500, 1000, 2000]$, the investigated optimizers were [adam, rmsprop, Adadelta, Adagrad, Adamax], the initial hidden layer sizes were [50, 100, 200, 300] and the number of hidden layers were [1, 3, 5, 7, 9]. The hidden layers were calculated by decreasing the number of neurons in each layer based on the ratio of the number of neurons in the first hidden layer to the required number of hidden layers. The input layer was fourteen and the output layer was six, which correspond to the number of input features and output classes, respectively. The process was implemented using the Python "GridSearchCV" function and 5-fold cross-validation, where the K-fold accuracy was the chosen metric. The developed DNN architecture for this study is provided in Table 8 and it was optimized over the number of epochs, batch size, optimizer, number of hidden layers and neurons. The developed DNN classifier consists of three hidden layers, the "Adamax" optimizer, 2000 epochs and a batch size of five. The DNN approach achieved high 5-fold cross-validation accuracy results and generalized well to the unseen data. The DNN requires a training time of 1937.04 seconds and achieves an average prediction time of 23.227 ms. These results correspond to being computed using Keras on a NVIDIA GeForce RTX 2060 with 6GB of RAM. In contrast, the average prediction times for the XGBoost and SVM approaches are 0.05 ms and 1.242 ms, respectively, while the training times are 78.1 ms and 44.44 ms. The confusion matrix for the developed Adaboost (Figure 12a), XGBoost (Figure 12b) and DNN (Figure 12c) models are compared in Figure 12, where the results are near equivalent. Notably, if a DNN/SVM approach is applied in a similar approach as the XGBoost/SVM method, the results will be equivalent. This is due to the SVM being the optimal approach but limited to reducing the number of errors in IEEE802.11 classification to 23, which is lower than the XGBoost and DNN models. Table 7 provides a summary of the results, where it is evident that the XGBoost approach achieves the lowest generalization error for the traditional supervised machine learning approaches, while the DNN is the optimal approach.

Table 8. DNN Structure: TensorFlow/Keras.

Layer Type	Layer Size	Activation Function
Input	14 neurons	relu
Fully Connected	50 neurons	relu
Fully Connected	34 neurons	relu
Fully Connected	17 neurons	relu
Output	6 neurons	softmax

The main takeaways are the training (Table 7) and prediction times, which are much lower for the supervised traditional XGBoost/SVM approach. As a result, the parameter optimization for the non-deep learning approaches is orders of magnitude faster, which pairs well with deployed wireless communication systems where extensive computational resources and time are rarely found [40]. The desired deployment of edge devices requires low complexity and fast optimization times for new environments. This requirement validates the selection of the developed XGBoost/SVM machine learning approach, as the XGBoost/SVM design achieves equivalent accuracy and generalization error results as the

developed fully connected neural network, for a fraction of the computation requirements and in a vastly reduced timescale.

The final implementation result from the Python experimentation indicates that the same prediction accuracy can be achieved on a much more lightweight device but the training and prediction times increase. This increase in training and prediction times would be exceedingly larger for the deep learning approach. However, the increased training time is generally not a concern as it can be rectified by simply training and optimizing the model on a much more advanced machine. Only the optimally trained model is required to be uploaded and used on the lightweight embedded device. Thus, the feature set and optimal models developed in this paper can, potentially, be used on IoT edge devices. The Raspberry Pi is suitable for this evaluation, as it is an example of how hardware has advanced over the past decade or so. As we look to the future, it is not unreasonable to suggest that edge devices will have similar specifications.

4.3. Summary and Discussion

This study of live wireless signals in a typical operating environment, which contained different signal sources, devices, obstacles and service usage, incorporates some limiting factors. The developed two-stage model approach would be heightened if data were captured across multiple industrial environments, as more data would be available and applied to the existing dataset developed in this study. The hardware specifications are also a limiting factor since performance is linked to the ADC resolution (12-bit) and reference voltage (1.3 V). A higher resolution would allow for received signals to be extracted in greater detail from the channel. The reference voltage, which is the maximum voltage available to the ADC, determines the ADC conversion ceiling for received analog inputs. Essentially, a higher reference voltage allows for higher powered signals to be received before saturation occurs. However, the novel feature set developed in this study has proven its ability to differentiate between signals when receiver saturation occurs.

Despite these limitations, the wireless approach depicted in this study obtained high performance in both classification accuracy and in generalizing to unseen data. This study, which employs fundamental feature-based machine learning algorithms, are in contrast to [22], which states that traditional feature-based approaches lack generalization. The results prove the effectiveness of the designed methods, which differ from the literature by only requiring access to raw received I/Q samples, permitting independent device decisions and using low-order statistical features. Typically, the literature uses high-order statistics [27] and/or cumulants [28–30] when applying traditional techniques. The achieved generalization error of below 1% is comparable to performance levels of other developed systems focused on the applied modulation scheme. However, unlike image classification, unified datasets for wireless signal classification are, generally, not yet available. So, the authors in [22] compared their system against various other feature-based schemes. The results, at sufficient signal-to-noise ratios (SNR), vary from classification accuracies of 80% up to almost 100%. In [42], similar results are achieved for a sufficient SNR when classifying wireless signals. Hence, the achieved results in this study compete with the literature and do so by using a low-complexity novel feature set and without focusing on modulation schemes, using spectrograms or RSSI samples. Overall, despite the trend to use deep learning approaches, as specified in Section 2, this study proves that potent data analysis and signal processing permit traditional techniques to still be effective (Table 7) when paired with sufficiently descriptive feature sets based on time, frequency and space (PDF).

A use case for this investigation can apply to developing interference detection systems or edge device decentralized decision making. An interference detection approach is, typically, developed based on data that are available to the system designer. This data, as shown in this paper, can be leveraged from a known dataset, commercial nodes, designed testbed, etc. However, the training data may be collected from a wireless environment that differs significantly, or marginally, from the proposed deployment. Hence, having another

approach that can generalize well and identify specific legitimate signals can leverage available optimized models and approaches to efficiently develop new interference detection models in new operating environments. Additionally, edge devices making independent real-time decisions based on the operating environment is the key development rationale for this study. In any use case that implements this designed methodology, especially edge devices, energy usage will be a key performance metric. Thus, it is envisaged that the designed approach would not operate continuously and, instead, only operate on a certain duty cycle, or when called upon. To demonstrate how the features and approach identified in this paper can be used in other developments, the features and optimal models in this study were leveraged in a domain adaption transfer learning study to develop a novel WSN interference diagnostic framework in [48]. The work in [48] produced and tested various interference detection scenarios which on average achieved a model accuracy $\geq$ 98%. These scenarios included interference detection, interference classification and legitimate node identification. In each model, the features and optimized XGBoost and DNN models developed in this paper were utilized. This utilization provides additional validation of the work in this paper as the features can be transferred to different use cases and data distributions.

5. Future Work and Conclusions

This study dealt with exclusively using raw received I/Q samples to develop a low-order statistical feature set for typical WSN and ISM band wireless signal classification. The signals included noise, IEEE802.11, Bluetooth advertising, CW and ZigBee (IEEE802.15.4) signals, which were transmitted from commercial devices and SDRs, where applicable. Features were extracted from the calculated PDF of received samples, statistical analysis of the time domain and from the frequency domain by implementing a FFT. The feature set differs from previous approaches due to the use of low-order statistics and novel uses of FFT samples and Hjorth parameters. Analog Pluto SDRs and Raspberry Pi embedded devices were exploited as a low-cost yet high-performing analysis approach for obtaining the required I/Q samples. The designed novel feature set was validated by intensively investigating the Random Forest approach, briefly analyzing k-NNs and SVMs and by developing a fully connected DNN. Test data included unseen data that were used to examine how the developed models generalized to new data.

The optimal discovered approach was an XGBoost/SVM adaption, which achieved an error of 0.527%. Most errors in the XGBoost model occurred between different IEEE802.11 signals and, so, the separate binary SVM classifier was developed to reduce the error if an IEEE802.11 signal was detected. A DNN was developed to provide a comparison between deep learning and supervised traditional approaches. The XGBoost/SVM model achieved the same accuracy as the DNN, but for reduced computational and time requirements. This proved that traditional feature-based approaches are still fit for purpose, particularly for low-complexity solutions, and achieve high performance when potent data analysis and novel descriptive feature sets are applied. A Raspberry Pi demonstrated that the designed models achieve the same results on an embedded device. Overall, this study showed that the lowest level of receivable data, I/Q samples, can be leveraged to make higher-level decisions.

Future work encompasses adopting the developed models and techniques for use on low-power edge devices, including power usage and hardware metrics, and investigating the usefulness of the designed approaches in making real-time edge decisions. Additionally, potential collaborations with other applications can be explored based on the successful transfer learning of the features and optimized models to interference detection mechanisms. The machine learning models and features can be optimized further by focusing on reducing the size of the applied FFT, if possible. More data will, typically, produce a model which can generalize to new instances in new environments with a reduced error. Thus, relevant data from various industrial environments and/or WSN/IoT deployments can be acquired, if possible.

Author Contributions: Conceptualization, G.D.O.; data curation, G.D.O.; formal analysis, G.D.O.; funding acquisition, G.D.O., P.J.H. and C.C.M.; investigation, G.D.O.; methodology, G.D.O.; project administration, G.D.O., P.J.H. and C.C.M.; resources, G.D.O.; software, G.D.O.; supervision, K.G.M., P.J.H. and C.C.M.; validation, G.D.O.; visualization, G.D.O., K.G.M., P.J.H. and C.C.M.; writing—original draft, G.D.O.; writing—review and editing, G.D.O., K.G.M., P.J.H. and C.C.M. All authors have read and agreed to the published version of the manuscript.

Funding: The Irish Research Council (IRC) and Raytheon Technologies Research Center, Ireland, support this work under the IRC Postgraduate Enterprise Partnership Scheme, EPSPG/2016/66.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The generated data presented in this study can be obtained using the experimental method and hardware as discussed in this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Katopodis, P.; Katsis, G.; Walker, O.; Tummala, M.; Michael, J.B. A Hybrid, Large-scale Wireless Sensor Network for Missile Defense. In Proceedings of the 2007 IEEE International Conference on System of Systems Engineering, San Antonio, TX, USA, 16–18 April 2007; pp. 1–5.
2. Förster, A. *Introduction to Wireless Sensor Networks*; Wiley: Hoboken, NJ, USA, 2016.
3. Tennina, S.; Santos, M.; Mesodiakaki, A.; Al, E. WSN4QoL: WSNs for remote patient monitoring in e-Health applications. In Proceedings of the 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 23–27 May 2016; pp. 1–6.
4. Beestermöller, H.J.; Sebald, J.; Sinnreich, M.C.; Borchers, H.J.; Schneider, M.; Luttmann, H.; Schmid, V. Wireless-Sensor Networks in Space Technology Demonstration on ISS. In Proceedings of the Dresdner Sensor-Symposium 2015, Dresden, Germany, 7–9 December 2015; pp. 99–102.
5. Vladimirova, T.; Bridges, C.P.; Paul, J.R.; Malik, S.A.; Sweeting, M.N. Space-based wireless sensor networks: Design issues. In Proceedings of the 2010 IEEE Aerospace Conference, Big Sky, MT, USA, 6–13 March 2010; pp. 1–14.
6. O'Mahony, G.D.; Harris, P.J.; Murphy, C.C. Investigating the Prevalent Security Techniques in Wireless Sensor Network Protocols. In Proceedings of the 2019 30th Irish Signals and Systems Conference (ISSC), Maynooth, Ireland, 17–18 June 2019; pp. 1–6.
7. Atzori, L.; Iera, A.; Morabito, G. The Internet of Things: A survey. *Comput. Netw.* **2010**, *54*, 2787–2805. . [CrossRef]
8. Zanella, A.; Bui, N.; Castellani, A.; Vangelista, L.; Zorzi, M. Internet of things for smart cities. *IEEE Internet Things J.* **2014**, *1*, 22–32. [CrossRef]
9. Cisco. *Cisco Annual Internet Report (2018–2023)*; Tech. Rep.; Cisco: San Jose, CA, USA, 2020.
10. O'Mahony, G.D.; Curran, J.T.; Harris, P.J.; Murphy, C.C. Interference and Intrusion in Wireless Sensor Networks. *IEEE Aerosp. Electron. Syst. Mag.* **2020**, *35*, 4–16. [CrossRef]
11. O'Mahony, G.D.; Harris, P.J.; Murphy, C.C. Investigating Supervised Machine Learning Techniques for Channel Identification in Wireless Sensor Networks. In Proceedings of the 2020 31st Irish Signals and Systems Conference (ISSC), Letterkenny, Ireland, 11–12 June 2020; pp. 1–6.
12. Tektronix. DPX Overview. 2019. Available online: https://www.tek.com/dpx-overview (accessed on 1 July 2021)
13. ZigBee Alliance. *ZigBee Specification. ZigBee Document 053474r20*; Tech. Rep.; The ZigBee Alliance: Davis, CA, USA, 2012.
14. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
15. Shanthakumar, V.A.; Banerjee, C.; Mukherjee, T.; Pasiliao, E. Uncooperative RF Direction Finding with I/Q Data. In Proceedings of the 2020 the 4th International Conference on Information System and Data Mining, Hawaii, HI, USA, 15–17 May 2020; pp. 6–13.
16. Roy, D.; Mukherjee, T.; Chatterjee, M.; Pasiliao, E. Detection of Rogue RF Transmitters using Generative Adversarial Nets. In Proceedings of the 2019 IEEE Wireless Communications and Networking Conference (WCNC), Marrakesh, Morocco, 15–18 April 2019; pp. 1–7.
17. Roy, D.; Mukherjee, T.; Chatterjee, M.; Pasiliao, E. Primary User Activity Prediction in DSA Networks using Recurrent Structures. In Proceedings of the 2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN), Newark, NJ, USA, 11–14 November 2019; pp. 1–10.
18. Roy, D.; Mukherjee, T.; Chatterjee, M.; Pasiliao, E. RF Transmitter Fingerprinting Exploiting Spatio-Temporal Properties in Raw Signal Data. In Proceedings of the 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), Boca Raton, FL, USA, 16–19 December 2019; pp. 89–96.
19. Jian, T.; Rendon, B.C.; Ojuba, E.; Soltani, N.; Wang, Z.; Sankhe, K.; Gritsenko, A.; Dy, J.; Chowdhury, K.; Ioannidis, S. Deep Learning for RF Fingerprinting: A Massive Experimental Study. *IEEE Internet Things Mag.* **2020**, *3*, 50–57. [CrossRef]

20. O'Mahony, G.D.; Harris, P.J.; Murphy, C.C. Identifying Distinct Features based on Received Samples for Interference Detection in Wireless Sensor Network Edge Devices. In Proceedings of the 2020 Wireless Telecommunications Symposium (WTS), Washington, DC, USA, 22–24 April 2020; pp. 1–7.
21. Wu, K.; Tan, H.; Ngan, H.L.; Liu, Y.; Ni, L.M. Chip error pattern analysis in IEEE 802.15.4. *IEEE Trans. Mob. Comput.* **2012**, *11*, 543–552.
22. Wahla, A.H.; Chen, L.; Wang, Y.; Chen, R.; Wu, F. Automatic Wireless Signal Classification in Multimedia Internet of Things: An Adaptive Boosting Enabled Approach. *IEEE Access* **2019**, *7*, 160334. [CrossRef]
23. Xu, J.L.; Su, W.; Zhou, M. Likelihood-Ratio Approaches to Automatic Modulation Classification. *IEEE Trans. Syst. Man Cybern. C Aool. Rev.* **2011**, *41*, 455–469. [CrossRef]
24. Hazza, A.; Shoaib, M.; Alshebeili, S.A.; Fahad, A. An overview of feature-based methods for digital modulation classification. In Proceedings of the 2013 1st International Conference on Communications, Signal Processing, and Their Applications (ICCSPA), Sharjah, United Arab Emirates, 12–14 February 2013; pp. 1–6.
25. Xie, L.; Wan, Q. Cyclic Feature-Based Modulation Recognition Using Compressive Sensing. *IEEE Wirel. Commun. Lett.* **2017**, *6*, 402–405. [CrossRef]
26. Park, C.; Choi, J.; Nah, S.; Jang, W.; Kim, D.Y. Automatic Modulation Recognition of Digital Signals using Wavelet Features and SVM. In Proceedings of the 2008 10th International Conference on Advanced Communication Technology, Gangwon, Korea, 17–20 February 2008; pp. 387–390.
27. Lee, S.H.; Kim, K.-Y.; Shin, Y. Effective Feature Selection Method for Deep Learning-Based Automatic Modulation Classification Scheme Using Higher-Order Statistics. *Appl. Sci.* **2020**, *10*, 5882. [CrossRef]
28. Smith, A.; Evans, M.; Downey, J. Modulation classification of satellite communication signals using cumulants and neural networks. In Proceedings of the 2017 Cognitive Communications for Aerospace Applications Workshop (CCAA), Cleveland, OH, USA, 27–28 June 2017; pp. 1–8.
29. Xie, W.; Hu, S.; Yu, C.; Zhu, P.; Peng, X.; Ouyang, J. Deep Learning in Digital Modulation Recognition Using High Order Cumulants. *IEEE Access* **2019**, *7*, 63760–63766. [CrossRef]
30. Pajic, M.S.; Veinovic, M.; Peric, M.; Orlic, V.D. Modulation Order Reduction Method for Improving the Performance of AMC Algorithm Based on Sixth–Order Cumulants. *IEEE Access* **2020**, *8*, 106386–106394. [CrossRef]
31. Hazar, M.A.; Odabasioglu, N.; Ensari, T.; Kavurucu, Y.; Sayan, O.F. Performance analysis and improvement of machine learning algorithms for automatic modulation recognition over Rayleigh fading channels. *Neural Comput. Appl.* **2018**, *29*, 351–360. [CrossRef]
32. Zhang, Y.; Wu, G.; Wang, J.; Tang, Q. Wireless signal classification based on high-order cumulants and machine learning. In Proceedings of the 2017 International Conference on Computer Technology, Electronics and Communication (ICCTEC), Chongqing, China, 25–26 March 2017; pp. 246–250.
33. Hu, H.; Wang, Y.; Song, J. Signal classification based on spectral correlation analysis and SVM in cognitive radio. In Proceedings of the 22nd International Conference on Advanced Information Networking and Applications (AINA 2008), Gino-wan, Japan, 25–28 March 2008; pp. 883–887.
34. Zhang, Z.; Li, Y.; Zhu, X.; Lin, Y. A Method for Modulation Recognition Based on Entropy Features and Random Forest. In Proceedings of the 2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), Prague, Czech Republic, 25–29 July 2017; pp. 243–246.
35. O'Shea, T.J.; Roy, T.; Clancy, T.C. Over-the-Air Deep Learning Based Radio Signal Classification. *IEEE J. Sel. Top. Signal Process.* **2018**, *12*, 168–179. [CrossRef]
36. Gravelle, C.; Zhou, R. SDR demonstration of signal classification in real-time using deep learning. In Proceedings of the 2019 IEEE Globecom Workshops (GC Wkshps), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–5.
37. Zheng, S.; Chen, S.; Qi, P.; Zhou, H.; Yang, X. Spectrum sensing based on deep learning classification for cognitive radios. *China Commun.* **2020**, *17*, 138–148. [CrossRef]
38. Zheng, S.; Qi, P.; Chen, S.; Yang, X. Fusion Methods for CNN-Based Automatic Modulation Classification. *IEEE Access* **2019**, *7*, 66496–66504. [CrossRef]
39. Zhang, Z.; Wang, C.; Gan, C.; Sun, S.; Wang, M. Automatic Modulation Classification Using Convolutional Neural Network With Features Fusion of SPWVD and BJD. *IEEE Trans. Signal Inf. Process. Netw.* **2019**, *5*, 469–478. [CrossRef]
40. He, H.; Jin, S.; Wen, C.; Gao, F.; Li, G.Y.; Xu, Z. Model-Driven Deep Learning for Physical Layer Communications. *IEEE Wirel. Commun.* **2019**, *26*, 77–83. [CrossRef]
41. Rajendran, S.; Meert, W.; Giustiniano, D.; Lenders, V.; Pollin, S. Deep Learning Models for Wireless Signal Classification with Distributed Low-Cost Spectrum Sensors. *IEEE Trans. Cogn. Commun. Netw.* **2018**, *4*, 433–445. [CrossRef]
42. Fontaine, J.; Fonseca, E.; Shahid, A.; Kist, M.; DaSilva, L.A.; Moerman, I.; Poorter, E.D. Towards low-complexity wireless technology classification across multiple environments. *Ad Hoc Netw.* **2019**, *91*, 101881. [CrossRef]
43. Hjorth, B. EEG analysis based on time domain properties. *Electroencephalogr. Clin. Neurophysiol.* **1970**, *29*, 306–310. [CrossRef]
44. Sagduyu, Y.E.; Shi, Y.; Erpek, T.; Headley, W.; Flowers, B.; Stantchev, G.; Lu, Z. When Wireless Security Meets Machine Learning: Motivation, Challenges, and Research Directions. *arXiv* **2020**, arXiv:2001.08883.

45. O'Mahony, G.D.; Harris, P.J.; Murphy, C.C. Analyzing using Software Defined Radios as Wireless Sensor Network Inspection and Testing Devices: An Internet of Things Penetration Testing Perspective. In Proceedings of the 2020 Global Internet of Things Summit (GIoTS), Dublin, Ireland, 3–5 June 2020; pp. 1–6.
46. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794.
47. Grimaldi, S.; Mahmood, A.; Gidlund, M.; Alves, M. An SVM-based method for classification of external interference in industrial wireless sensor and actuator networks. *J. Sens. Actuator Netw.* **2017**, *6*, 9. [CrossRef]
48. O'Mahony, G.D.; McCarthy, K.G.; Harris, P.J.; Murphy, C.C. Developing novel low complexity models using received in-phase and quadrature-phase samples for interference detection and classification in Wireless Sensor Network and GPS edge devices. *Ad Hoc Netw.* **2021**, *120*, 102562. [CrossRef]