A Framework for Wireless Technology Classification using Crowdsensing Platforms

Alessio Scalingi*, Domenico Giustiniano*, Roberto Calvo-Palomino†, Nikolaos Apostolakis* and Gérôme Bovet‡
*IMDEA Networks institute, Madrid, Spain {alessio.scalingi, domenico.giustiniano, nikolaos.apostolakis}@imdea.org
†Universidad Rey Juan Carlos, Madrid, Spain {roberto.calvo}@urjc.es
‡Armasuisse, Thun, Switzerland {gerome.bovet}@armasuisse.ch

***Abstract*—Spectrum crowdsensing systems do not provide labeled data near real-time yet. We propose a framework that addresses this challenge and relies solely on Power Spectrum Density (PSD) data collected by low-cost receivers. A major hurdle is to design a system that is computationally efficient for near real-time operation, yet using only the limited 2 MHz bandwidth of low-cost spectrum sensors. First, we present a method for unsupervised transmission detection that works with PSD data already collected by the backend of the crowdsensing platform, and that provides stable detection of transmission boundaries. Second, we introduce a data-driven deep learning solution to classify the wireless technology used by the transmitter, using transmission features in a compressed space extracted from *single PSD measurements over at most 2 MHz band*. We build an experimental platform, and evaluate our framework with real-world data collected from 47 different sensors deployed across Europe. We show that our framework yields an average classification accuracy close to 94.25% over the testing dataset, with a maximum latency of 3.4 seconds when integrated in the backend of a major crowdsensing network. Code and data have been released for reproducibility and further studies.**

I. Introduction

The need to monitor continuously the spectrum at large scale is becoming more and more prominent with new emerging applications that are increasing its demand. In order to manage the spectrum usage, scalable data-driven techniques are required to swiftly understand the spectrum, its usage, and its evolution over time, supporting near real-time spectrum management policies such as spectrum allocation, spectrum sharing, dynamic spectrum access, and anomaly detection. Crowdsensing spectrum systems based on low-cost Software Defined Radios (SDR)-based sensors such as KiWiSDR [1] and Electrosense [2] have recently received attention by the community to monitor the spectrum in time, frequency and space. Yet, they do not provide so far labeled data.

At high level, there are a couple of potential architectural options to classify the wireless technology using crowdsensing spectrum systems, making labeled data available for spectrum management policies. Streaming IQ data to the backend for technology classification is not viable as IQ data requires a network uplink rate of 153 Mb/s per sensor, even with very low-cost RTL-SDR front-ends [3]. As comparison, the new proposed broadband definition in US by the Federal Communications Commission for uplink data rate is set to 20 Mb/s [4]. Another option could be to embed classification capabilities in the sensors. However, the sensors' front-end and computational capabilities in crowdsensing setups is often limited, and thus performing classification on embedded devices is not feasible with today's deployments.

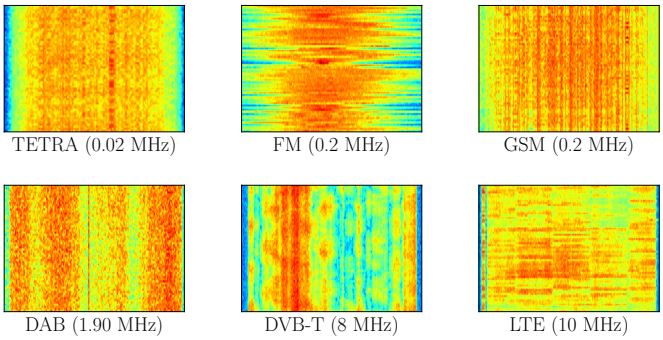


Fig. 1: Wireless technologies classified in this work. A single transmission spectrogram is characterized by a continuously occupied time-frequency block. Time on *y-axis* and frequency on *x-axis*. Some of the technologies use the same modulation schema (OFDM for DAB, DVB-T and LTE) or have similar bandwidth (FM and GSM), others are very narrowband (Tetra).

In this work, we propose a framework to classify wireless technologies using solely Power Spectrum Density (PSD) measurements. The technologies we aim to classify are those shown in Fig. 1, which show different properties in terms of modulation schema and bandwidth. We take benefit from the fact that PSD data is already uploaded through crowdsensing sensors to the backend for waterfall visualization. Besides, PSD data is compatible with private deployments having limited internet connectivity due to the lower network uplink data rate (in the order of 150 kb/s [2]). Yet, it is more challenging to label PSD data because of its limited information [5]. Indeed, PSD performs squared magnitude Fast Fourier transform (FFT) using the Welch method [6], averaging and smoothing IQ spectrum data from a given number of readings. Our framework aims to infer the transmitting signal technology using PSD data while addressing the following two fundamental architectural issues:

- As the spectrum is sparsely used [7], directly performing technology classification on the whole spectrum is not efficient from a storage and computation standpoint. Furthermore, this approach does not scale well with the number

©2023 IEEE. Personal use of this materialis permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

of spectrum sensors connected to the backend. However, prior work focused on the classification task rather than spectrum process, using also empty spectrum to classify signals or synthetic dataset [8]–[10];
- Prior work used Deep Learning (DL) approaches to classify technologies [5] [11] [12]. However, they failed to distinguish technologies, in particular if the modulation schema is the same (e.g., both using OFDM [5]). Besides, the signal bandwidth was used to infer the technology, which could result in wrong decisions, as we show in this work. Other solutions applied DL on large portions of time-frequency spectrogram or image format transformed spectrum (similar to the ones shown in Fig. 1), causing high computational costs and delays in labeling data [9], [13].

Given this background, this paper makes the following contributions:

- We present a light and low-computational cost framework for spectrum transmission detection and classification;
- We present an unsupervised signal detection algorithm that detects single transmissions efficiently, to save computational and storage resources;
- We propose a LSTM-based model as sequence classifier that uses a Neural Network (NN) Encoder for feature reduction, extracted from PSD measurements;
- We integrate our prototype in Electrosense [2] as crowdsensing platform addressing practical challenges;
- For reproducibility and to foster further research, the code and the labeled dataset collected with 47 spectrum sensors deployed across Europe are made freely available.

Our framework for wireless technology classification is based on DL models and it has two distintive properties: i) it uses only features from one-single PSD measurement, meaning only one row of the spectrogram matrix in Fig. 1; and ii) it uses at most the front-end's maximum bandwidth - 2 MHz - as input of our model, which helps the model to learn signals' patterns and reduce the inference time. We achieve an accuracy of 94.25% over the testing dataset, with a maximum latency of 3.4 seconds when running our framework in the Electrosense platform. Our practical design can contribute to develop more sophisticated intelligent radio applications, e.g. spectrum sharing, which require to address challenges in distributed spectrum monitoring and classification.

II. Challenges

We study the problem of wireless technology classification using data collected with a crowdsensing network of low-cost spectrum sensors. These sensors compute PSD data and send it to the backend, where we envision to infer the transmitting technologies. A *transmission* is defined as the wireless signal observed in the time-frequency spectrogram, characterized by a continuous-occupied time-frequency block like the examples in Fig. 1, which also details the wireless technologies classified in this work. We identify key challenges in this section, and that we address in this work.

Challenge A: Spectrum fragmentation with limited hardware. Spectrum sensors in this work are composed of a Raspberry Pi embedded board for computing and communication, and an RTL-SDR of less than 30 $ as radio front-end. The RTL-SDR has a limited sampling rate of 2.4 MHz, and the edges of the spectrum are removed for further processing because the frequency response is not flat [14]. It results that the sensor uses S=2 MHz of bandwidth. The RTL-SDR stays in a given 2 MHz band only for a fraction of time. Considering FFT of 256 bins and 10 readings of IQ measurements in the band for PSD computation with Welch method, the total time spent in a band before switching frequency is $256*10/(2.4*10^6) \approx 1\ \mu s$. We define this data as *single PSD measurement*. The RTL-SDR sweeps the full spectrum from 24 MHz to 1.7 GHz with a general purpose antenna. Hence, the framework has to classify wireless transmissions with limited and fragmented knowledge of the spectrum.

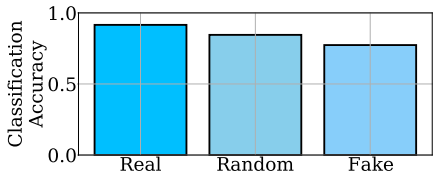


Fig. 2: Accuracy when using same length vectors of frequency bins M, for real, random and fake segments with state-of-the-art methods [5]. Training a DL model directly with PSD segments transmissions leads the classifier to learn the bandwidth instead of more relevant bandwidth-agnostic features.

Challenge B: Transmission detection robust to noisy measurements. As we will show using our real dataset in Section VII, state of the art algorithms for transmission detection (like Airview [15]) are sensitive to noisy measurements such as those provided by low-cost RTL-SDR-based sensors, and may miss or aggregate multiple real transmissions. This is problematic for the classification task, as the bandwidth of the transmission of the DL algorithm becomes dependent of the noise level. Correctly identifying the signal transmission is then essential for the classification task.

Challenge C: Classifying technologies of different bandwidth. We define a *PSD segment* as the row vector $\mathbf{x} \in \mathbb{R}^M$ containing the frequency bins of the power spectrum density. For instance, the vector length of a single PSD measurement is about 2 MHz, and it consists of a vector of $M = 215$ bins. Several transmissions are present in the spectrum, with bandwidth B that varies largely. A well-known solution proposed in the seminal work [5] for making consistent the length of the input vector of DL model for different vector sizes is to add zero-padding until all PSD segments have the same length. For instance, let us consider FM (200 kHz, corresponding to $M \approx 22$) and LTE (10 MHz, $M \approx 1067$). Following this prior approach, PSD segments of FM transmissions would be padded to achieve the same length as those from LTE. Fig. 2 shows the classification accuracy using the aforementioned model, comparing it with random and fake spectrum segments created with the same bandwidth as the real transmissions. Random dataset is created using a seed function to build random PSD segments that were used as input for the inference. Fake dataset is created using the vector length expected for

the target technology (FM at 200 kHz), but adding PSD values corresponding to a different signal technology (i.e. LTE). In all three cases, the accuracy is high, meaning that 0-padding has more weight in learning the bandwidth rather than the PSD values. We then conclude that the state-of-the-art models classify the transmissions mainly by relying on the number of bins, which relates to the bandwidth of the technology.

Challenge D: Near real-time operation. In a practical setting, the designer may decide to speed up the classification process, e.g, by classifying the transmissions using only a single PSD measurement through the data collected with the sensor bandwidth of up to S=2 MHz. One issue of this approach is that the classifier would have limited knowledge of the technology when $B > S$ (e.g., DVB-T or LTE). Another problem is that classifying using a single PSD measurement may result in missed temporal features in the transmissions.

III. Transmission Detection

A high-level overview of the system is shown in Fig. 3. Spectrum sensors measure in-phase (I) and in-quadrature (Q) samples over 2 MHz bands. They convert IQ data to PSD data, which are sent to the backend of the crowdsensing platform for further analysis and visualization, like waterfall plots of the whole 24 MHz-1.7 GHz spectrum. The Transmission Detection System (TDS) is part of our framework that aims to detect active signals by measuring spectrum occupancy (Challenge B), and is further detailed in this section. Data is then classified using only the portions of spectrum where transmissions are detected (Challenge A, C, and D), as detailed in Section IV.

The main task of TDS is to identify transmissions and store their samples along with some metadata. TDS is essential, as it enhances scalability and efficiency in the overall classification process. Indeed, the spectrum is sparsely used [7], and processing empty spectrum wastes computational resources, also risking to overload the system. Furthermore, the more sensors there are in the network the more transmissions need to be classified. More in details, the RTL-SDR device scans the full spectrum, hopping sequentially among different spectrum bands [16]. As example, a complete scanned

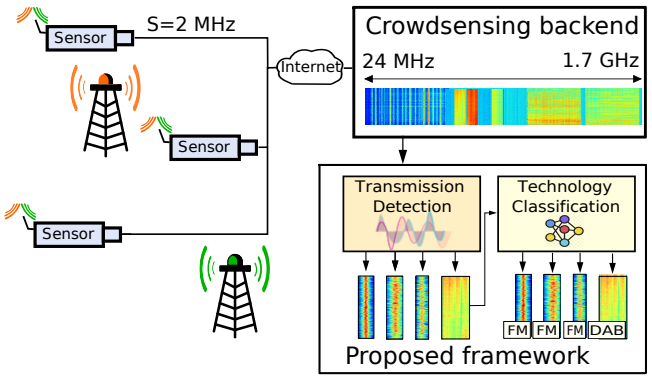


Fig. 3: Data is collected by spectrum sensors and sent to the crowdsensing backend. For each sensor, transmissions are then detected and classified.

spectrum provides 870 *PSD segments* that contain 215 bins each, for a total length of $870 * 215 = 187'050$ frequency bins. We use PSD values sampled through RTL-SDR sensors as input, which are stored in the platform's backend in dB scale. Each sensor computes and sends PSD in dB scale as[1] $X_m = 10 * log_{10}(I^2 + Q^2)$, where X_m represents the squared magnitude values of the frequency bin m. Each bin has a resolution of approximately 10 kHz. In our dataset (Section V) we collect 6 hours of spectrum from each sensor. We group the segments of our dataset as M × K matrix, that represents the full spectrum in the time span of K consecutive time periods, corresponding to $K = 360$ and $M = 187'050$.

Our TDS is articulated in three main independent blocks as depicted in Fig. 4: *(1) Noise level computation, (2) Energy-detector, (3) Peaks finding and edge detection.* Without TDS, the classification task would need to perform continuously inference over 870 PSD segments, with some segments containing multiple transmissions and complicating the design of the classifier. Instead, our aim is to use at most 2 MHz band for a single transmission. This would result in many less PSD segments to process depending on activities in the wireless spectrum. TDS implements an algorithm that processes multiple time sweeps of PSD segments over the whole spectrum to detect transmissions, and stores the samples and metadata like start/end frequency and SNR of detected signals. TDS then extracts the transmissions addressing issues, like noisy measurements and signal bandwidth detection, that could lead to the inference of a lower number of transmissions than expected in a given spectrum band (or also more than expected) with respect to the national regulation.

Noise level computation. In the first step, we estimate the noise floor η considering the full spectrum. The idea is that there exists a portion of the spectrum at any point in time without transmissions, that can be used to determine the noise level. Furthermore, it is a common practice when studying frequency transmission systems to consider the noise level to be flat across the whole spectrum.

The backend of the crowdsensing platform receives PSD values in dB scale, however due to the fact that the average in dB scale cannot be computed, we convert each bin to an amplitude absolute scale using an inverse transformation $Y_m = 10^{\frac{X_m}{20}}$. Our algorithm averages the bins for every single *PSD segments* of length $M = 215$ and computes which portion of the full scanned spectrum has the minimum value as follows:

$$j = \arg\{\mu_j = min(\mu_m)\}, \quad m = 1, 2, \ldots, 870.$$

As the average could be influenced by the distribution of frequency bins, we compute the standard deviation σ_j on the selected portion of the spectrum j, and apply the 3-sigma rule to infer the noise level η as:

$$\eta = \mu_j + 3 \cdot \sigma_j$$

[1]More in detail, X_m represents the PSD value computed using a given number of IQ readings over a given band, which is computed in each sensor using the Welch method, similarly to [16]. However, we omit the averaging process in the notation for simplicity.

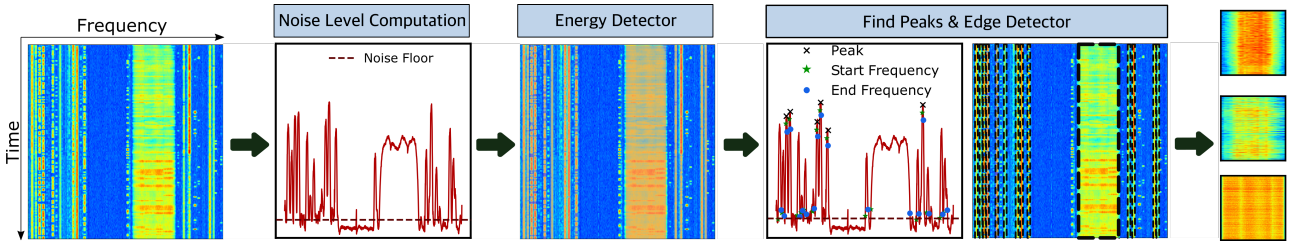


Fig. 4: Block diagram of transmission detection.

In presence of thermal Gaussian noise at the receiver, this corresponds to select 99.7 percentile of noise as threshold, which allows us to discard most of the noise in the process of detecting transmissions. After estimating the noise level η, the system evaluates the spectrum occupancy to identify every single active transmission.

Energy detector. This step operates on multiple sweeps, which means considering a time span of K consecutive time segments. As for the noise level, we convert K time segments to the linear-amplitude scale and compute the average over time per each bin as follows: $AVG = 20 * log_{10}(\frac{\sum_{i=1}^{K} Y_i}{K})$. We then compute the occupancy according to the International Telecommunication Union (ITU) standard that recommends using 3-5 dB threshold for amplitude values above the noise level [17]. We note that the aforementioned ITU standard does not provide details on the algorithm implementation as proposed here. Therefore, starting from the first bin, we detect an active transmission whenever $AVG - \eta \geq 5$ dB. In the same way, the algorithm declares the end of a transmission in the frequency domain if $AVG - \eta$ is below the threshold. The process is repeated sweeping to the end of the spectrum.

Find peaks and edge detection. Multiple transmissions may be seen as a single one in presence of noisy data, e.g. FM transmissions which have small frequency spaces between them. We therefore re-process the detected transmissions to identify the presence of smaller band transmissions, not detected previously. We consider each detected transmission as a matrix of PSD values $X \in \mathbb{R}^{MxK}$, with M rows (frequency bins) and K columns (time segments). We compute the *Coefficient of Variation (CV)* defined as the standard deviation over the mean σ_X/μ_X of the matrix values. CV presents the extent of variability in relation to the mean of the population. For $CV < 1$, the transmission is added to the list of detected transmissions and the matrix X is stored as well as the SNR, start/end frequency; for $CV \geq 1$, bins values are fluctuating significantly with respect to the mean value suggesting sub-transmissions, and the algorithm processes X with *find peaks* and *edge detection* functions. *Find peaks* takes as input 1-dimension arrays and returns the index of the peaks based on the shape of the signal. Then, *edge detection* works on each detected peak, and it defines the bandwidth of the transmission considering a 3-dB threshold below the peak (cf. Fig. 4).

IV. Technology Classification

The output of TDS provides the set of transmissions to be classified. The data representation of a single transmission is the time-frequency spectrum mapped as a matrix $X \in \mathbb{R}^{MxK}$. The input of the Technology Classification System (TCS) for both the learning and inference process is a PSD segment, that is, a single row k of the transmission matrix. We design our framework with a feature extraction along with DL classifier to better discriminate signals' technologies as detailed in the remainder of this section.

A. Feature extraction

A major challenge of our framework is the ability to classify transmissions with different bandwidth (Challenge C). This means working with *PSD segments* of different number of frequency bins M because i) the classifier must classify different technologies, and ii) the same technology may result in slightly different bandwidths depending on the measurement noise in the TDS task. This raises the problem of making the input consistent for the classifier.

In order to address these issues, we adopt a method that transforms *PSD segments* in a fixed vector of features that contains its statistical characteristics. We implement the feature extraction module with the open source library *tsfresh* [18]. The module takes as input a single PSD segment and returns a fixed-length vector with 32 statistical features. These features are mainly used to capture the segments' characteristics specific to each technology, such as mean power, variance, maximum, number of elements above the mean, occurrences of repeated values, and absolute energy (the sum over the squared PSD values) among others. Nevertheless, we observe through Pearson correlation studies that not all the features are equally relevant to represent all the classes while some of them are correlated. In the next section, we detail an architecture that encodes the features to reduce the vector dimensions.

B. Model architecture

The statistical features are used as input of the learning component which is split into two parts.

Dimensional reduction. We resort to an AutoEncoder (AE) architecture to automatically reduce the number of features and work only with a set of independent features. This is because we observed that there is not a unique subset of the features - neither all of them - to well classify all the technologies at the same time[2]. In general terms, the AE is designed to reproduce its input at the output layer by compressing the input

[2]In this evaluation, not reported for limited space, we compared different techniques for feature extraction against AE including Three-based, Recursive feature selection and correlation-based. AE-based method got the best results.

TABLE I: Model architecture. Line in bold in the AE architecture represents the compressed features given as input of the classifiers. Layers after it are discarded from our model. The compressed vector of 16 elements is used as input of the classifier.

(a) AE architecture.

Layer	Output dim.
Input	1x32
Dense/ReLu	1x64
Dense/ReLu	1x32
Dense/ReLu	**1x16**
Dense/ReLu	1x32
Dense/ReLu	1x64
Dense/ReLu	1x34

(b) Classifier architecture

Layer	Output dim.
Input	1x16
LSTM	1x32
LSTM	1x16
Dense/ReLu	1x16
Dropout	1x16
Dense/SoftMax	1x6

dimension and expand it back to the output. After training the full AE, we take the encoder part so that we can represent the input feature vector with reduced dimensions. AE design is further investigated in Section VII.

Classifier. A Long Short-Term Memory (LSTM) network is used for PSD data classification. It is a type of Recurrent Neural Network (RNN) capable to learn long-term dependencies within sequence data. Due to the internal connections, states and a gating mechanism, the NN learns what information within the sequence is important for its task. The LSTM may capture the relationship of the extracted features from every single row k of the transmission. In this way, the LSTM predicts the $k+1$ *PSD segment* without considering consecutive multiple measurements in time (Challenges A and D). We train first the AE in a semi-supervised manner, its architecture is detailed in Table Ia, then we use the labeled version of the encoded data to train the LSTM classifier in supervised way, with its architecture detailed in Table Ib. Notably, the modular architecture of the spectrum classifier ensures that we can interchange the NN with any different kind of model. We further analyze these aspects in detail in Section VII.

Untrained technologies. In this work we focus on the classification of six technologies, as shown in Figure 1. However, as there are transmissions from many more technologies in the spectrum, we need a mechanism to correctly label the untrained technologies. We use the *abstain class* mechanism based on the *entropy selection* to transform the original m-*class* classification problem into $(m+1)$-*class*, where the $(m+1)$-*th* class represents the model abstaining from making a prediction due to lack of confidence. In particular, we define p_i as the probability that the output belongs to class i. The *Entropy* is defined as follows:

$$H = -\sum_{i=0}^{m} p_i log(p_i) \tag{1}$$

This methodology uses the H as metric to measure an uncertainty score. The TCS task abstains from making a prediction when the measure of the entropy is above a predetermined threshold α. That prediction is then labeled as *Untrained Technology* (UT) (the $(m+1)$-*class*), that is:

$$Class = \begin{cases} \arg\max_{i=0,..,m} p_i & if\ H < \alpha \\ \text{Untrained Technology} & \text{otherwise.} \end{cases} \tag{2}$$

TABLE II: Our main dataset used in this work.

Description	Values
Number of RTL-SDR sensors	47
Sensing hours	282
Technologies (target)	DAB, DVB-T, FM, GSM, LTE, TETRA
Frequencies (MHz)	80-120,180-230,350-700,780-830,930-960
Number of training samples	108.000 segments
Number of testing samples	26.490 segments

With this mechanism, TCS classifies as UT all those transmissions for which it has not been trained for.

V. Dataset

We use Electrosense to collect PSD segments. Each sensor in Electrosense executes digital signal processing tasks: it samples I/Q data, applies the FFT with the Welch method, compresses and sends PSD data to the backend. RTL-SDR takes 24 *ms* to re-tune the central frequency in the frequency hopping strategy due to electronic switching delays. Considering the processing delay, the full scan (24-1766 MHz) is performed in roughly 40 seconds. Overall, the dataset has 60.3 GB of PSD data from 47 different RTL-SDR sensors around Europe. The sensor's owner has full control over the deployment, then we do not control factors like SNR, indoor/outdoor antenna, environmental conditions, antenna gain, and distance from the transmitter. For each sensor, we collect 6 hours of the full spectrum scan, at a resolution of approximately 10 kHz per bin (the highest that can be provided by RTL-SDR front-ends), resulting in a total of 282 hours. During the data collection, we label wireless technologies (see Table II) with information provided by regulatory agencies. We obtain a labeled dataset of the spectrum occupied by these technologies. The dataset has over 134k PSD segments, where each of them is a transmission to be classified. We use 80% of the dataset for training, and the 20% for testing. Class instances are equally balanced. Because of the TDS identifies transmissions 5 dB above the noise floor with the algorithm presented in Section III, PSD segments have good characteristics from the spectrum standpoint.

VI. Experimental prototype

In this section, we explain how we deploy the experimental prototype of our framework employing different software modules. We deploy the prototype in a controlled and separate *test environment* of Electrosense to test the feasibility in a production-like setting with a near real-time application. Fig. 5 depicts the components deployed in the test environment, how they interact with the crowdsensing platform (both sensors and backend), and how the final user is involved in the loop.

A walk-through. We deploy our framework by implementing the backend and the frontend. We use virtualization with Docker containers [19] to deploy and replicate the framework to the cloud. A single container embraces all the necessary components to run the system and facilitates the inference on data centers equipped with or without GPUs. We mainly focus on the backend of our prototype for the evaluation in this work. The backend is implemented in *Flask* and it handles the required functionalities, connecting with the operational

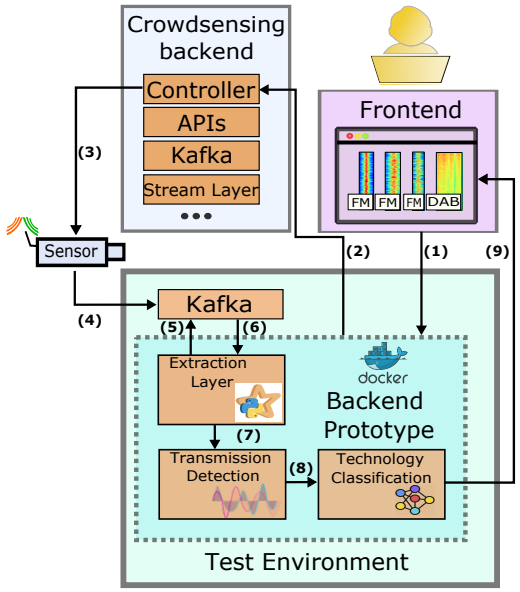


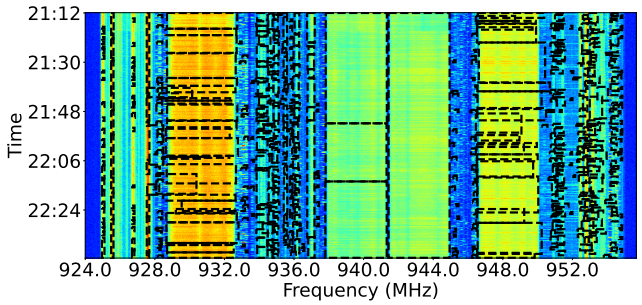
Fig. 5: Implementation and integration in the crowdsensing backend.

Electrosense backend, and interacting with the user through the frontend. The process starts with the user who selects the sensor and defines the spectrum frequencies to scan (1). Our prototype framework interacts with the Electrosense controller (2) to start data collection. The controller initiates a timed campaign on a specific sensor (3) that, in turn, sends PSD measurements to the framework's backend (4). Like in the production environment, the messages are delivered to *Kafka*, which is a publish-subscribe streaming message queue that collects the messages from the sensor. A single message contains *PSD segment* of S=2 MHz of bandwidth.

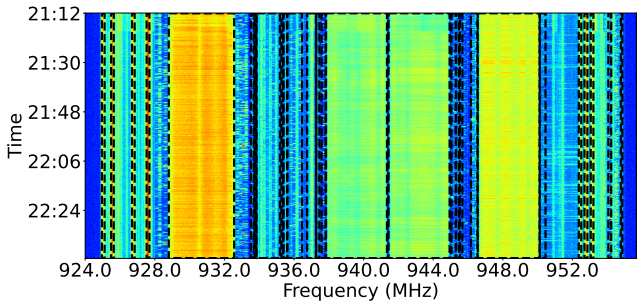
Overall, the algorithmic design and implementation of the proposed framework must guarantee that the latency in the Extraction Layer (EL), TDS and TCS allows to run near real-time applications. First, the EL is a module implemented in *PySpark* that fetches (5) the messages from *Kafka* and group the segments as M × K matrix that represent the spectrum (6) (cf. Section III). The output of the EL is the portion of the spectrum measured in the total period of the campaign. The data is sent to the TDS, where the single transmissions are detected (7). Every single transmission is processed by the TCS that performs the inference with single PSD segments (8). The labeled transmissions are sent to the frontend (9) of and displayed to the final user.

VII. Experimental evaluation

This section is organized in three parts. First, we present the experimental results of TDS showing a comparison with the state of the art. In the second part, we evaluate the TCS with different parameters of our architecture, comparing our solution with different ML and DL algorithms. Then, we test its robustness in presence of other types of technologies. Finally, we evaluate the performance of our prototype integrated in the crowdsensing platform.



(a) Baseline



(b) Proposed transmission detection algorithm.

Fig. 6: Example of Airview [15] (a) compared to our TDS mechanism (b). The latter shows stable detection of transmission boundaries, both in frequency and time domains, which eases the task of classification. Our algorithm compared to the baseline also avoids fragmentation of the same transmission in time for all the trace lengths.

A. Transmission detection evaluation

We evaluate the framework using our real spectrum dataset presented in Section V. We evaluate the performance with DAB, DVBT, FM, GSM, LTE and TETRA technologies. For the sake of simplicity, we report two analysis evaluated in FM, TETRA and GSM frequency bands.

1) Qualitative evaluation with AirView benchmark: In this analysis we use as baseline the AirView algorithm for transmission detection because it was designed for fast detection of transmissions [15]. We show a comparison in the GSM Band in Fig. 6. The picture shows the ability of our TDS algorithm to identify and localize small and nearby transmissions even in noisy conditions, addressing *Challenge B*. From this figure, we can also observe that the time-frequency boundaries for *PSD segments* are clearly defined. Applying highly varying boundaries as we can observe with AirView would lead to in a higher complexity for the classification task.

2) Quantitative evaluation: In this study, we evaluate the bandwidth distribution of the detected transmissions for the peaks finding and edge detection steps while varying the threshold below the peak. As explained in Section III, in order to detect transmission boundaries and avoid to merge different transmissions, the algorithm interpolates the signal with a threshold below the peak. We evaluate the performance using 3 or 6 dB below the peak, comparing the bandwidth of FM, GSM, TETRA signals in Fig. 7. Using 3 dB threshold with respect 6 dB, we achieve a bandwidth closer to the real one, for example 200 kHz for FM, and GSM. Conversely, 6 dB

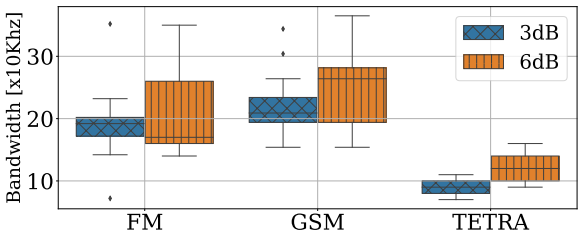
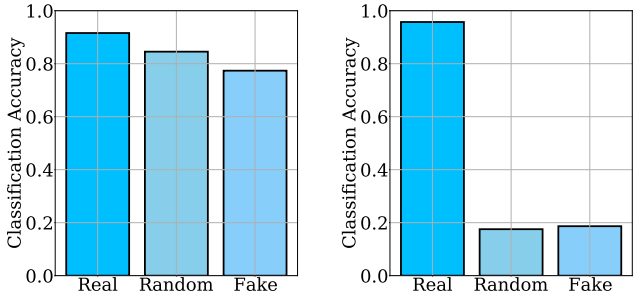


Fig. 7: Distribution of the transmission bandwidths detected by the algorithm: comparison of two thresholds. 3 dB threshold provides the most accurate results.



(a) Previous approach [5]. (b) Our approach.

Fig. 8: Accuracy of DL approaches using PSD as input vectors compared to our approach. We use real, random and fake transmissions of same bandwidth to show that our algorithm is bandwidth-agnostic.

merges small-bands and nearby transmissions as a single one, leading to wrongly detected boundaries with respect to the spectrum regulations in the observed bands. Therefore, we consider 3 dB as the threshold of our TDS task.

B. Technology classification evaluation

We first present the justification for a new model for classifying technologies using PSD data, then introduce the proposed methods for feature selection and inference with single PSD measurements. We also study the impact of untrained technologies on the model and show the experimental tests for latency for our prototype integrated in Electrosense.

1) Learning from PSD data: Prior work used DL algorithms directly on PSD data [5]. In Fig. 8a, we show predictions with random and fake spectrum segments, with the same bandwidth B as the actual transmission. The high score achieved with a model trained directly with PSD data demonstrates that a model is learning the signals' bandwidth B instead of the PSD data itself. We justify it with the following two arguments. First, large-bandwidth signals like LTE define long sequences as input, which may be difficult for NN like LSTM to remember. Second, due the padding, the model learns just the length of the non-zero part of the input sequence. We instead extract statistical features from PSD segments, and the prediction is not influenced by the bandwidth B, as shown in Fig. 8b, addressing *Challenge C*.

2) AE for features selection: We adopt an AE for dimensional reduction, the compressed dimension contains the important information for the inference. We evaluate two different architectures for its implementation, Fully-Connected Neural Network (FCNN)-based and RNN based, respectively.

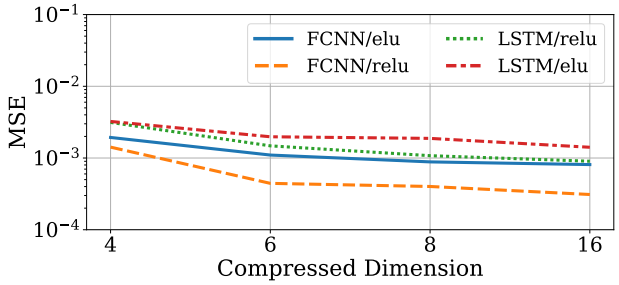


Fig. 9: Comparison of Mean Square Error (MSE) of autoencoders for compressing features of PSD segments.

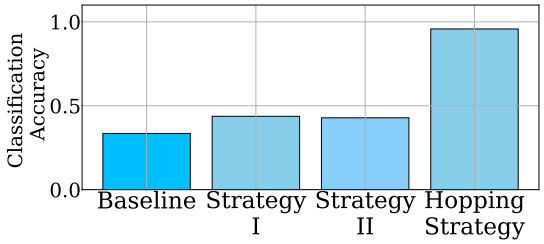


Fig. 10: Overall accuracy (test-set) of different strategies.

We show the reconstruction error of these two different architectures with the training set and the validation set in Fig. 9, varying the compressed dimensions from 4 to 16. For this study, the network's target is the input itself. Hence, the smaller the error, the more the network compresses and reconstructs the inputs. As expected, reducing the compressed dimension increases the reconstruction error. We also evaluate different activation functions. The best performance is achieved with FCNN architectures with *ReLu* as activation function. As presented in Section IV-B, we use the encoder part of the latter architecture to compress the features in a dimension of 16, which provides the minimum error.

3) Inference with single PSD measurements: Wireless technologies have bandwidth B that varies largely with respect to the sensor bandwidth $S = 2$ MHz. As discussed in Section II, we aim to classify technologies using at most the sensor bandwidth S. The model training is the crucial step to face this challenge and, for a consistent evaluation, in the test set we use segments of 2 MHz for transmissions with $S \leq B$. We study different versions of the training set, experimenting with different strategies to extract the features vectors from PSD segments. We train the model with training and validation set for 550 epochs, using the Adam optimizer [20], a learning rate of 0.001, early stopping and patience equal to 10 epochs to prevent overfitting. We take the accuracy score as a metric for our evaluation. Fig. 10 describes the classification accuracy on the test set of different strategies.

As *baseline*, the feature extractor (cf. Section IV-A) uses the entire B of the transmission, where all the frequency bins are used to create the feature vector. Low classification accuracy on test set in Fig. 10 shows poor performance to infer larger signals with segments of 2 MHz.

In *strategies I/II*, we extract the features as a function of B. From a single PSD segment $\mathbf{x} \in \mathbb{R}^M$ in a transmission, we extract 4 vectors of features, using a fixed portion of the bandwidth, 25%, 50%, 75% and 100% of B, respectively.

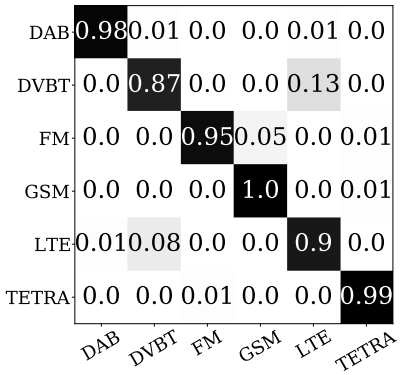


Fig. 11: 2-Layer LSTM confusion matrix. The features are extracted with the hopping strategy.

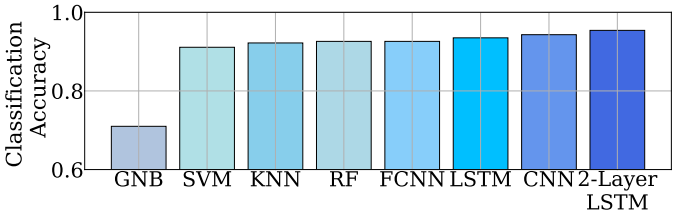


Fig. 12: Benchmark of different models: gaussian naive bayes (GNB), support-vector machines (SVM), k-nearest neighbors (KNN), random forest (RF), Fully-Connected Neural networks (FCNN), 1-layer LSTM, convolutional neural networks (CNN), 2-layer LSTM (our proposed model).

In *Strategy I*, we extract portions of the bandwidth from the center to the edge of the PSD segment. In *Strategy II*, we extract features starting from the edge (e.g. from the first-left frequency bin we use the 25%, 50%, 75%, 100% of B). The overall accuracy increases on the test set with respect to the baseline, but we still have unsatisfactory results.

The *hopping strategy* extracts features in chunks of 2 MHz from the first frequency bin. As shown in Fig. 10, this approach achieves the best accuracy of 94.25%, addressing *Challenge D*, because data preparation is fundamental to training the model. The confusion matrix in Fig. 11 shows the class inference: different modulations change into different spectrum occupancy that can be easily learned. Moreover, signals of similar B like DVB-T and LTE are well classified, addressing *Challenge C*.

4) Models' benchmark: We choose the hopping strategy to process our dataset with the feature extractor (Section IV-A), and compare the classification accuracy of several machine learning and DL models in Fig. 12. Extracting the features with the hopping strategy works well with several classifiers but the 2-Layer-LSTM outperforms other models.

5) Entropy study: After selecting the NN and training the models, we tune the α threshold. Specifically, we use another dataset, called α-*test* set that contains transmissions in the full spectrum (also in the untrained frequencies, cf. Table II). We observe in Fig. 13 that technologies in other bands than those with trained technologies, denoted as *Untrained Technology* (UT), have much higher Entropy. Therefore, selecting a threshold α around 0.6-0.7 allows to correctly label UTs.

After this high-level evaluation, we aim to more formally find the best α trading-off the *Coverage*($Covr$) and the *Accuracy*(Acc). The *covered dataset* is the portion of data where predictions are below the threshold, and the $Covr$ is the ratio of the size of the covered dataset to the size of the original test dataset. For this evaluation, we discard the predictions with $H \geq \alpha$, that is, we do not consider them for the final accuracy score. In Fig. 14 (left), we show the Pareto-fronts for different α. We observe that (i) larger is α the higher is the $Covr$, (ii) as α increases the Acc decreases. We frame our problem as an optimization of two joined objective functions that should be maximized, Acc and $Covr$, respectively. To select α, we convert the problem into a unique objective function to be maximized by using the weighted sum method as follows:

$$f_1 = \gamma * Acc(\alpha) + (1-\gamma) * Covr(\alpha) \ . \quad (3)$$

Fig. 14 (right) represents the values of f_1 considering $\gamma = 0.5$, equal weights for Acc and $Covr$. The system provides the highest values of f_1 with $\alpha = 0.7$. This ensures the best accuracy considering as many number of samples as possible and it is in agreement with the results in Fig. 13.

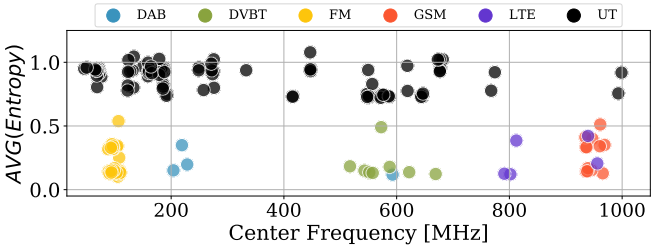


Fig. 13: Entropy (H) for the six different classes plus transmissions in frequency bands not used in the training set as Untrained Technology (UT). We observe that entropy in these latter bands is much higher.

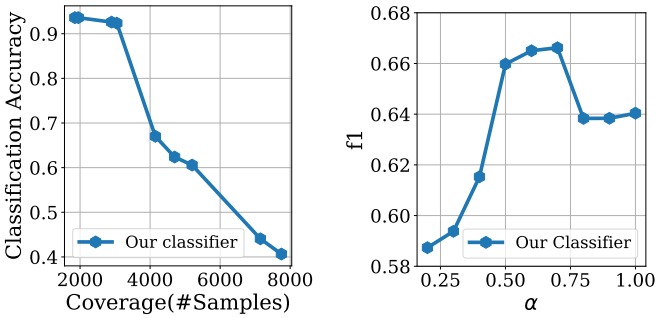


Fig. 14: Pareto-Fronts (left) of the Accuracy and Coverage and f_1 optimization (right) for each value of the threshold α.

C. Prototype Evaluation

We deploy our prototype on three different machines. The first one is only equipped with Intel(R) Core(TM) i5 CPU, the second and the third have also GPUs, GeForce RTX 2080 Ti and 2xNVIDIA A100-PCIE, respectively.

Speeding up the classification. We demonstrate the advantage of using at most 2 MHz of B to classify the wireless technologies. Fig. 15 shows TCS latency for a single PSD segment $\mathbf{x} \in \mathbb{R}^M$. We compare two techniques: full bandwidth B of the transmission and up to 2 MHz. For narrowband signals, predicting the technology is constant because the features are extracted (cf. Section IV-A) from PSD segments

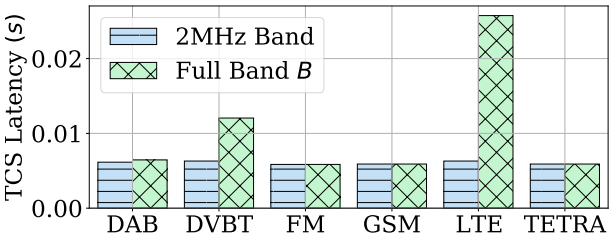


Fig. 15: TCS latency for each technology. For large B, our framework predicts technology's label with PSD segments of 2 MHz, reducing the latency time respect to using the full band B.

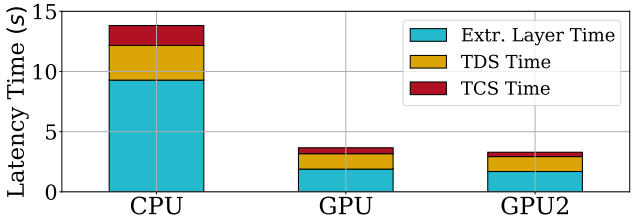


Fig. 16: Latency for executing the entire framework on one PSD segment covering the whole spectrum, with transmissions with both trained technologies and untrained ones.

with a smaller length than B. Instead, TCS latency for signals with bandwidth $S \leq B$ is greater using the full B to extract the features. Our framework predicts these technologies correctly, and it reduces the latency as we only use part of the band B.

Latency of executing the framework. We study the latency to label the transmissions in the spectrum from 20 MHz to 1.7 GHz. The frontend interacts with the crowdsensing platform: we select one of the available sensors and request to label a single PSD segment of the full spectrum, $\mathbf{x} \in \mathbb{R}^M$, with $M = 870$. Fig. 16 shows the time spent by the EL, TDS and TCS. The EL orders all PSD segments of 2 MHz, and its latency is significant on the CPU, relatively less on the GPU. In particular, it takes 3.4s on the GPU2 to classify all transmissions in the whole spectrum.

VIII. Related Work

In this section, we present work related to the two main areas of contribution of this paper, transmission detection and technology classification. Prior work in both areas lacks often real and large-scale deployment testbeds. The few solutions tested on real data required high-end spectrum sensing devices.

Wireless transmission detection. Traditional methods for the identification of wireless signals rely on extensive engineering of specialized features to identify active radio transmitters, such as higher-order signal statistics like for example cumulants [21], moments [22] or cyclo-stationary [23]–[25]. Instead, energy-based detection [26]–[28] have shown so far limited capabilities in terms of identifying active transmissions. An unsupervised algorithm for rapid transmission detection - based on wavelet decomposition - has been presented in [15] but it is not able to provide clear boundaries of transmissions with data extracted with low-cost sensors. [29] uses Rayleigh-Gaussian mixture models on batch data and cannot be applied to near-real time detection of transmissions.

Wireless technology classification. A large body of literature exists in the field of technology classification. The scientific community has followed three main approaches: Likelihood-Based (LB), Feature-Based (FB) and Deep-learning (DL). The major limitation in LB [30], [31] approaches is the requirement of prior knowledge about all signals and channel parameters. In FB approaches [32], [33] the presence of domain experts is fundamental to designing a satisfactory classification system, which is a limitation too. DL approach has been studied from two different perspectives related to the type of data used and the target of the classification task. A large body of works used IQ data with the objective to classify the modulation schema which can not be sent to the backend using crowdsensing platforms neither process them locally. In [10] a CNN-based model is used for wireless interference identification with synthetic-generated data, while in our work we designed a framework that can work with real, noisy and fragmented spectrum data. Concerning PSD data, although [12] uses real data to infer technologies such as WiFi, Bluetooth and ZigBee, their experiments in a semi-anechoic chamber exclude the possibility to adopt their model to the real world. Authors in [34] use time-frequency power spectrogram and a CNN classifier but approach requires capturing wideband spectrum with high-end equipment. [35] used a fuzzy logic technology classifier including the bandwidth information, which we avoid to learn as discussed in this work. According to the results of [5], technologies that shares the same modulation schema are difficult to discriminate, especially if the have similar spectrogram, one of the challenges addressed in this work.

IX. Conclusion

We have proposed a framework for transmission detection and wireless technology classification in a crowdsourcing network using only already collected PSD data. We have addressed several practical challenges, such as spectrum fragmentation, transmission detection with noisy measurements, classification of transmissions with different estimated bandwidths and support for near real-time operation. We have provided an evaluation with real signals captured by RTL-SDR sensors deployed across Europe. Our experimental study has shown that we can classify wireless technologies with an accuracy of 94.25% using only one single PSD measurement over at most 2 MHz band. Finally, we proposed and evaluated the design of an experimental prototype integrated in the crowdsensing platform for testing the system latency for near real-time application scenarios. We have provided public access to code and data here https://doi.org/10.5281/zenodo.7504334.

Acknowledgement

Part of this work was supported by the project MAP-6G, reference TSI-063000-2021-63, granted by the Ministry of Economic Affairs and Digital Transformation and the European Union-NextGenerationEU through the UNICO-5G R&D Program of the Spanish Recovery, Transformation and Resilience Plan.

REFERENCES

[1] "Kiwisdr," http://www.kiwisdr.com/, Last seen: July 2022.

[2] R. Calvo-Palomino, H. Cordobés, M. Engel, M. Fuchs, P. Jain, M. Liechti, S. Rajendran, M. Schäfer, B. Van den Bergh, S. Pollin *et al.*, "Electrosense: Open and big spectrum data," *Computer Networks*, vol. 56, no. 1, pp. 210–217, 2018.

[3] Y. Zeng, R. Calvo-Palomino, D. Giustiniano, G. Bovet, and S. Banerjee, "Adaptive uplink data compression in spectrum crowdsensing systems," in *2021 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, 2021, pp. 114–122.

[4] "FCC," https://www.fcc.gov/document/chairwoman-rosenworcel-proposes-increase-minimum-broadband-speeds, July 2022, Last seen: July 2022.

[5] S. Rajendran, W. Meert, D. Giustiniano, V. Lenders, and S. Pollin, "Deep learning models for wireless signal classification with distributed low-cost spectrum sensors," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 3, pp. 433–445, 2018.

[6] J. G. Proakis, *Digital signal processing: principles algorithms and applications*. Pearson Education India, 2001.

[7] H. Hassanieh, L. Shi, O. Abari, E. Hamed, and D. Katabi, "Ghz-wide sensing and decoding using the sparse fourier transform," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, 2014, pp. 2256–2264.

[8] Z. Ke and H. Vikalo, "Real-time radio technology and modulation classification via an lstm auto-encoder," *IEEE Transactions on Wireless Communications*, 2021.

[9] S. Chen, K. Qiu, S. Zheng, Q. Xuan, and X. Yang, "Radio–image transformer: Bridging radio modulation classification and imagenet classification," *Electronics*, vol. 9, no. 10, p. 1646, 2020.

[10] M. Schmidt, D. Block, and U. Meier, "Wireless interference identification with convolutional neural networks," in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*. IEEE, 2017, pp. 180–185.

[11] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168–179, 2018.

[12] N. Bitar, S. Muhammad, and H. H. Refai, "Wireless technology identification using deep convolutional neural networks," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE, 2017, pp. 1–6.

[13] H. N. Nguyen, M. Vomvas, T. Vo-Huu, and G. Noubir, "Spectro-temporal rf identification using deep learning," *arXiv preprint arXiv:2107.05114*, 2021.

[14] R. Calvo-Palomino, D. Giustiniano, and V. Lenders, "Measuring spectrum similarity in distributed radio monitoring systems," in *International Tyrrhenian Workshop on Digital Communication*. Springer, 2017, pp. 215–229.

[15] M. Zheleva, P. Bogdanov, T. Larock, and P. Schmitt, "Airview: Unsupervised transmitter detection for next generation spectrum sensing," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 1673–1681.

[16] D. Pfammatter, D. Giustiniano, and V. Lenders, "A software-defined sensor architecture for large-scale wideband spectrum monitoring," in *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, 2015, pp. 71–82.

[17] S. Series, "Spectrum occupancy measurements and evaluation - report itu-r sm.2256-1," 2016.

[18] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package)," *Neurocomputing*, vol. 307, pp. 72–77, 2018.

[19] I. Miell and A. Sayers, *Docker in practice*. Simon and Schuster, 2019.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[21] O. A. Dobre, Y. Bar-Ness, and W. Su, "Higher-order cyclic cumulants for high order modulation classification," in *IEEE Military Communications Conference, 2003. MILCOM 2003.*, vol. 1. IEEE, 2003, pp. 112–117.

[22] S. S. Soliman and S.-Z. Hsue, "Signal classification using statistical moments," *IEEE Transactions on Communications*, vol. 40, no. 5, pp. 908–916, 1992.

[23] P. D. Sutton, K. E. Nolan, and L. E. Doyle, "Cyclostationary signatures in practical cognitive radio applications," *IEEE Journal on selected areas in Communications*, vol. 26, no. 1, pp. 13–24, 2008.

[24] P. Schmitt, D. Iland, M. Zheleva, and E. Belding, "Hybridcell: Cellular connectivity on the fringes with demand-driven local cells," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[25] S. S. Hong and S. R. Katti, "Dof: A local wireless information plane," in *Proceedings of the ACM SIGCOMM 2011 Conference*, 2011, pp. 230–241.

[26] Y. Yuan, P. Bahl, R. Chandra, P. A. Chou, J. I. Ferrell, T. Moscibroda, S. Narlanka, and Y. Wu, "Knows: Cognitive radio networks over white spaces," in *2007 2nd IEEE international symposium on new frontiers in dynamic spectrum access networks*. IEEE, 2007, pp. 416–427.

[27] M. McHenry, E. Livsics, T. Nguyen, and N. Majumdar, "Xg dynamic spectrum access field test results [topics in radio communications]," *IEEE Communications Magazine*, vol. 45, no. 6, pp. 51–57, 2007.

[28] F. F. Digham, M.-S. Alouini, and M. K. Simon, "On the energy detection of unknown signals over fading channels," in *IEEE International Conference on Communications, 2003. ICC'03.*, vol. 5. Ieee, 2003, pp. 3575–3579.

[29] M. Zheleva, R. Chandra, A. Chowdhery, A. Kapoor, and P. Garnett, "Txminer: Identifying transmitters in real-world spectrum measurements," in *2015 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*. IEEE, 2015, pp. 94–105.

[30] J. L. Xu, W. Su, and M. Zhou, "Likelihood-ratio approaches to automatic modulation classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 4, pp. 455–469, 2010.

[31] O. A. Dobre, A. Abdi, Y. Bar-Ness, and W. Su, "Survey of automatic modulation classification techniques: classical approaches and new trends," *IET communications*, vol. 1, no. 2, pp. 137–156, 2007.

[32] A. Ebrahimzadeh and S. A. Seyedin, "Automatic digital modulation identification in dispersive channels," in *Proceedings of the 5th WSEAS International Conference on Telecommunications and Informatics*, 2006, pp. 409–414.

[33] J. Li, Q. Meng, G. Zhang, Y. Sun, L. Qiu, and W. Ma, "Automatic modulation classification using support vector machines and error correcting output codes," in *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*. IEEE, 2017, pp. 60–63.

[34] T. J. O'Shea, T. Roy, and T. Erpek, "Spectral detection and localization of radio events with learned convolutional neural features," in *2007 2nd IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks*, 2017, pp. 331–335.

[35] K. Ahmad, G. Shresta, U. Meier, and H. Kwasnicka, "Neuro-fuzzy signal classifier (nfsc) for standard wireless technologies," in *2010 7th International Symposium on Wireless Communication Systems*. IEEE, 2010, pp. 616–620.