IEEE INFOCOM 2021 - IEEE Conference on Computer Communications | 978-1-6654-0325-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/INFOCOM42981.2021.9488764

DeepSense: Fast Wideband Spectrum Sensing Through Real-Time In-the-Loop Deep Learning

Daniel Uvaydov*, Salvatore D'Oro*, Francesco Restuccia*†, Tommaso Melodia*
*Institute for the Wireless Internet of Things
† The Roux Institute
Northeastern University, Boston, MA, USA
Email: {uvaydov.d, s.doro, frestuc, t.melodia}@northeastern.edu

***Abstract*—Spectrum sharing will be a key technology to tackle spectrum scarcity in the sub-6 GHz bands. To fairly access the shared bandwidth, wireless users will necessarily need to quickly sense large portions of spectrum and opportunistically access unutilized bands. The key unaddressed challenges of spectrum sensing are that (i) it has to be performed with extremely low latency over large bandwidths to detect tiny spectrum holes and to guarantee strict real-time digital signal processing (DSP) constraints; (ii) its underlying algorithms need to be extremely accurate, and flexible enough to work with different wireless bands and protocols to find application in real-world settings. To the best of our knowledge, the literature lacks spectrum sensing techniques able to accomplish both requirements. In this paper, we propose *DeepSense*, a software/hardware framework for real-time wideband spectrum sensing that relies on real-time deep learning tightly integrated into the transceiver's baseband processing logic to detect and exploit unutilized spectrum bands. DeepSense uses a convolutional neural network (CNN) implemented in the wireless platform's hardware fabric to analyze a small portion of the unprocessed baseband waveform to automatically extract the maximum amount of information with the least amount of I/Q samples. We extensively validate the accuracy, latency and generality performance of *DeepSense* with (i) a 400 GB dataset containing hundreds of thousands of WiFi transmissions collected "in the wild" with different Signal-to-Noise-Ratio (SNR) conditions and over different days; (ii) a dataset of transmissions collected using our own software-defined radio testbed; and (iii) a synthetic dataset of LTE transmissions under controlled SNR conditions. We also measure the real-time latency of the CNNs trained on the three datasets with an FPGA implementation, and compare our approach with a fixed energy threshold mechanism. Results show that our learning-based approach can deliver a precision and recall of 98% and 97% respectively and a latency as low as 0.61ms. For reproducibility and benchmarking purposes, we pledge to share the code and the datasets used in this paper to the community.**

I. Introduction

Our never-ending appetite for multi Gigabit-per-second (Gbps) data rates is pushing today's wireless technologies and infrastructures to their limits [1]. Although millimeter wave (mmWave) technologies are being deployed to alleviate the spectrum crunch issue, 5th generation (5G) technologies are envisioned to heavily utilize sub-6 GHz spectrum bands in the foreseeable future [2]. As these bands become more and more saturated, *spectrum sharing* in licensed bands [3, 4] will become a fundamental component to fuel wireless growth in the years to come. In short, spectrum sharing allows multiple wireless technologies (e.g., LTE, NR, WiFi) to coexist in the same spectrum band, by letting primary users (PUs) access the spectrum with some level of priority, while secondary users (SUs) opportunistically utilize the spectrum whenever available [5, 6]. Realizing the compelling need for spectrum sharing technologies, in April 2020, the US Federal Communication Commission (FCC) has opened up 1.2 GHz of spectrum in the 6 GHz band to new users [7]. Moreover, in January 2020, the FCC has opened the 3.5 GHz band for spectrum usage, also known as the "Innovation Band" and widely recognised as the key to unlocking the full 5G potential [8].

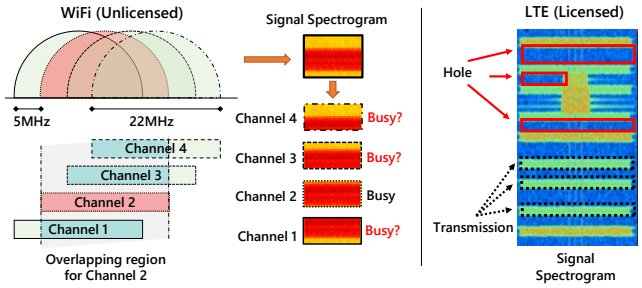

Fig. 1: Comparison of spectrum sensing in WiFi and LTE systems.

Spectrum sharing is critically dependent on the capability of radios to determine unutilized spectrum bands, also referred to as *spectrum sensing*, which has received significant attention in the past few years [9–22]. Effective spectrum sensing is challenging because large spectrum chunks (*i.e.*, several tens of MHz) often need to be sensed instantaneously. To give a practical example, IEEE 802.11 (WiFi) has 14 channels in the 2.4-GHz band, which are spaced 5 MHz apart. This covers a total bandwidth of 94 MHz, meaning that to sense the whole WiFi spectrum a device has to operate at 200 MHz sampling rate. Perhaps even more critically, spectrum sensing has to clearly distinguish transmissions from noise/interference. Figure 1 (left) shows an example of spectrum sensing in a 2.4 MHz WiFi network, where we consider a signal from the dataset described in Section IV-C. For simplicity, we consider only the first 4 channels. In this example, only channel 2 is being occupied by a transmission, while the other 3 channels receive interference. In this scenario, which is common of wireless systems operating in unlicensed bands, identifying whether a whole channel is being occupied is not easy due to interference from neighboring channels. For this reason, an easier—and more efficient—approach is to identify portions of the same channel that are not being used (*e.g.*, the upper sub-band of channel 4). The same problem also arises in licensed applications with completely orthogonal channels—such as LTE—as

This article is based on material supported in part by the US National Science Foundation under grants CNS-1923789 and CCF-1937500.

shown in Figure 1 (right). We show that even licensed bands are usually partially used and present holes with substantially different size.

These examples clearly show that spectrum sensing is a very challenging problem with no one-size-fits-all solutions. Most importantly, it demonstrates the need for spectrum sensing techniques that (i) are fast enough to keep up with the incoming bandwidth; (ii) distinguish transmissions from interference and noise; and (iii) are able to detect the smallest holes in the spectrum with high accuracy. Critically, given the multitude of protocols and spectrum bands today available, spectrum sensing techniques also need to be flexible, and able to work with different wireless technologies and spectrum bands.

To address the existing lack of real-time general-purpose spectrum sensing, in this paper we leverage hardware-based deep learning (DL) – in particular, convolutional neural networks (CNNs) – to sense the spectrum with the desired accuracy and speed directly on the receiver's baseband processing logic. We chose CNNs since they have been proven to be well suited for real-time inference in the wireless domain [23]. Moreover, CNNs have the major advantage to operate on "raw" (*i.e.*, unprocessed) I/Q samples, which allows fast and generalizable spectrum classification. Conversely, legacy spectrum sensing techniques rely on ad hoc, protocol-dependent feature extraction techniques, which (i) do not generalize to different protocols, architectures and spectrum environments; and (ii) are not amenable to be implemented in real time. Energy detection [24] simply compares the energy level in a narrowband channel with a threshold in a short time window of tens of μs, but does not reveal insights on whether the transmission is a packet or noise/interference. In Section V, we show that our approach outperforms energy detection by 63% at its best. Techniques like wavelet analysis [25] and compressed sensing [16, 18, 20] require (i) heavy computational resources, and (ii) exact knowledge of noise conditions, and so far have not been demonstrated for real-time networks.

Although using CNNs may seem a panacea, the reality is that the unavoidable non-stationarity of wireless networks makes learning problems in the spectrum domain extremely daunting [26]. Indeed, wireless phenomena (*e.g.*, channel, noise, fading, concurrent transmissions) usually change in a matter of a few milliseconds [27], which provides an indicative timescale in which the real-time inference must operate. Although [26] has proven the feasibility of using DL for classification purposes, it is still unclear if CNNs can be successfully used for real-time spectrum sensing, which justifies our large-scale investigation conducted in this paper.

Summary of Novel Contributions

We briefly summarize the key technical contributions of this paper below:

• We propose *DeepSense*, a software/hardware framework for real-time wideband spectrum sensing that relies on DL in the receiver's baseband processing logic to detect and exploit unutilized spectrum bands. At its core, DeepSense uses a CNN implemented in the wireless platform's hardware fabric and placed in the receiver's baseband demodulation logic to analyze a small portion of the unprocessed baseband waveform to automatically extract the maximum amount of information with the least amount of I/Q samples. We also mathematically derive real-time system constraints to correctly compute *DeepSense*'s memory and latency requirements;

• We extensively evaluate the performance of *DeepSense* in both accuracy and latency performance. Specifically, we utilize (i) a large-scale (400 GB) dataset provided by a government agency (GVT dataset) containing hundreds of thousands of WiFi transmissions with different Signal-to-Noise-Ratio (SNR) conditions and collected over different days; (ii) a dataset of WiFi transmissions collected using our own software-defined radio testbed (SDR dataset); (iii) a synthetic dataset of LTE transmissions under different noise conditions (LTE dataset). We train multiple CNN and evaluate their performance in terms of precision, recall and F1 scores. We also measure the real-time latency of the CNNs trained on the three datasets with a field-programmable gate array (FPGA) implementation. Results show that our learning-based approach can deliver a precision and recall of 98% and 97% respectively and a latency as low as 0.61ms;

• For reproducibility purposes and to stimulate further research in the field, we pledge to release the SDR and LTE datasets to the community[1], as well as the code used to train and test our models. We hope that these datasets can become benchmarks of performance for future research.

The rest of the paper is organized as follows: Section II summarizes the current state of the art, while Section III provides an overview of the *DeepSense* framework and DNN architecture. Section IV describes the utilized datasets and performance metrics utilized, whereas Section V presents our results. We draw conclusions in Section VI.

II. Related Work

Spectrum sensing has received a lot of interest from the research community [9–21, 28–32, 32–36]. For a recent survey on the topic, the reader can refer to [3, 22]. Previous work on spectrum sensing has been divided between narrowband and wideband approaches [22]. Narrowband approaches such as energy detection and matched filtering are hardly adaptable to the diverse and constantly changing spectrum, requiring prior knowledge of the transmission to effectively detect holes [28]. Other narrowband methods such as adaptive thresholds, cyclo-stationary feature detection and even some deep learning/machine learning techniques experience good performance in low SNR but are inefficient as they need to sequentially scan the spectrum to get a sense of occupancy for a wider band, occupying time and potentially missing holes [29–31].

Wideband sensing techniques are more suitable for spectrum sharing applications, as they can monitor multiple holes at once which can be allocated accordingly. These techniques usually fall into Nyquist and sub-Nyquist categories [22]. Nyquist based methods such as multi-band joint detection, wavelets, or utilizing filter banks present high complexity and thus, incur in high latency [25, 32, 33]. For example, Quan *et al.* [32] utilize an energy threshold vector that is found through convex

[1]Due to contract obligations, we cannot release the GVT dataset as part of this paper. We hope this will change in the future.

optimization, and requires recalculations to accommodate fast-changing channels. Current sub-Nyquist wideband techniques attempt to reconstruct the channel through sparse signals and although they utilize less data, the reconstruction process causes increased latency and complexity, and thus do not outperform Nyquist based techniques [34–36].

Deep learning approaches to spectrum sensing have been considered in previous work. However, they are computationally heavy or have not attempted wideband sensing. The method proposed by Liu *et al.* [19] shows good performance, but requires a signal covariance matrix as input to the CNN, whereas we use unprocessed I/Q signals. Chew *et al.* [21] utilize AlexNet, a CNN with over a million parameters that cannot meet real time constraints for spectrum hole detection. Indeed, CNNs need to detect holes at sub-millisecond speeds to be effective for spectrum sharing applications. To the best of our knowledge, all current deep learning approaches do not take latency into account, being the most crucial aspect of realizing a practical device. In addition, almost all existing work has not been tested on extensive real-time datasets and have been mostly based on simulated results. The RadioML2016.10a dataset [37], for example, has been used in [31] but the signals are generated from simulated channels. We offer results with both simulated data and waveforms collected "in the wild" for a full evaluation of wideband spectrum sensing with deep learning.

III. The DeepSense Framework

Figure 2 provides a high-level overview of the main components of DeepSense. In a nutshell, DeepSense can be seen as a software-defined radio (SDR) where (i) a Central Processing Unit (CPU) implements the higher-level functionalities of the network protocol stack, *i.e.*, Medium Access Control (MAC) layer and above, as well as the drivers to control the spectrum sensing module; and (ii) an FPGA holds the main computational burden of the system, *i.e.*, the Transmitter (TX) and Receiver (RX) baseband processing chains and the deep neural network (DNN) responsible for spectrum sensing. In the following, we will use the word "hole" to define an empty spectrum sub-band.

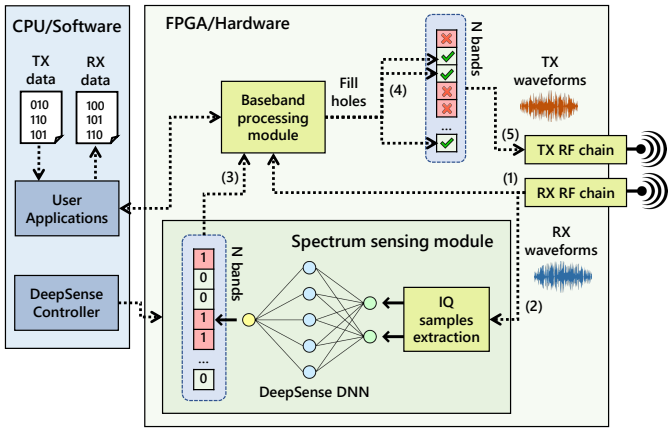

Fig. 2: DeepSense system model.

We chose a hybrid CPU/FPGA architecture since it maps well to off-the-shelf system-on-chip (SoC), which combine both components on the same chip [38]. SoCs have the advantage of low power consumption [39] and allow great flexibility to DeepSense, as the spectrum sensing module in the FPGA can be reprogrammed based on the needed learning design. Moreover, FPGAs experience much lower latency than CPUs, and thus are a better fit to do most of the computations required in the system with the added benefit of being software programmable [26]. Specifically, the FPGA interfaces with the CPU through registers, where the CPU writes the DNN weights on block memory (BRAM). GPUs are not a viable option either, though they can be fast they do not guarantee a fixed latency and furthermore do not benefit the RF chain.

DeepSense: A Walk-through. The key innovation of DeepSense is ultimately to integrate spectrum sensing capabilities into the SDR baseband processing. Specifically, at a high level, the FPGA includes two main components, the spectrum sensing module (SSM) and the baseband processing module (BPM). The first task performed by DeepSense is to receive the RX waveform through the RX chain (step 1). Then, the I/Q samples are extracted and fed to the trained DeepSense DNN (step 2). The target of the DNN is to determine which portions of the spectrum are empty and could be utilized for transmission. The training procedure of the DNN will be described in Section III-A. The generation of the ground truth depends on the dataset, and will be detailed in Section IV. Once the spectrum holes are detected out of the N possible spectrum bands, this information is fed to the BPM (step 3). The BPM then fills the holes according to the inference provided by the DNN (step 4). Finally, the TX waveform is transmitted through the TX RF chain (step 5).

Filling the Holes. The system must be able to generate waveforms that can effectively fill the spectrum hole, be decoded properly by the intended receiver, and do not generate interference to already ongoing transmissions. As a consequence, how to fill holes in practice is strongly application- and device-dependent. Specifically, although many protocols and standards operate over the same spectrum frequencies, they generate waveforms requiring completely different bandwidth. For example, protocols operating in the industrial, scientific and medical (ISM) band share the same frequencies, yet require a bandwidth ranging from 20MHz (*i.e.*, WiFi) to 1MHz (*i.e.*, Bluetooth). In this case, a node can simply choose one out of the possible sub-bandwidths available, and then perform clear channel assessment (CCA) or similar techniques before transmission, to avoid collision with other transmitters. If non-standard-compliant waveforms can be used, then more than one sub-bands could be used, which in turn significantly complicates the receiver design, and introduces more coordination between the transmitter and receiver.

Although a naive approach would be to design an SSM for each standard and protocol, it would not generalize properly, as it would apply to specific conditions only. On the contrary, we have designed *DeepSense* to detect holes with high resolution while keeping the complexity of the network as small as possible. As we will demonstrate in Section V-A, *DeepSense* can detect holes that are up to 5.4% smaller than the bandwidth of the monitored spectrum in less than 1ms.

This way, *DeepSense* can rapidly provide the transmitter with a high resolution map of spectrum holes. The receiver can then aggregate multiple neighboring holes into larger ones that are wide enough to fit standard-compliant waveforms. For example, if *DeepSense* detects two holes, 500MHz wide, and close to each other, the transmitter can aggregate these holes and fill them by generating a Bluetooth signal occupying 1MHz (*i.e.*, the size of a Bluetooth channel).

A. DNN Architecture and Training

The strict latency constraints of spectrum sensing imply that our DNN needs to be lightweight, flexible yet accurate. In other words, the smaller input we give to our network to detect holes, the lower the latency, since the DNN (i) will perform less computation, and (ii) will require fewer I/Q samples and therefore less time to produce an output. Furthermore, if the input is smaller, it is less likely that a channel will change from free to busy within the input. For this reason, we choose multi-label CNNs as our framework of choice. Indeed, recent research on modulation classification [23] and radio fingerprinting [40] has proven that CNNs are exceptionally versatile to address complex classification problems in the wireless domain. This is mainly thanks to fact that patterns in the I/Q constellation plane can be learned by the filters in the convolutional layers.

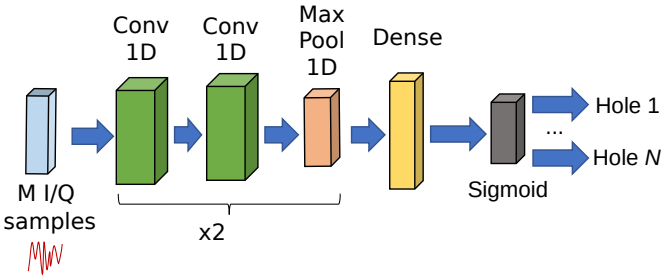

Fig. 3: DeepSense Multi-label CNN Architecture.

In the following, we consider the one-dimensional (1D) CNN architecture in Figure 3. Specifically, the input tensors is of size (M, 2), where M is the number of consecutive I/Q samples taken from the RX waveform. By increasing the number of I/Q samples, the CNN will be more likely to recognize the I/Q patterns in the constellation, at the expense of more computational burden. The input is processed by two 1D convolutional (Conv1D) layers, followed by a 1D maximum pooling (MaxPool1D) layer with filters of size 1x2 and stride 2. This way, the MaxPool1D cuts the output dimension in half. This layer pattern is repeated twice. One dense layer follows, with a Sigmoid layer placed at the end. The exact dimensions for the CNN in each application can be found in Table I.

The input sizes that we use are found through multiple iterations. In short, we experience better performance as the input size increases, up to a point, then we see little to no improvement. We use the smallest input size that the data allows before the CNN performance metrics suffer. This architecture is chosen because with a small input size, a network with many MaxPool1D layers would reduce the tensor size very quickly and minimize the depth of our network. Deep networks have shown to help CNNs develop high levels of abstractions and improve learning [41–43]. Therefore, our network consists of two stacks of two convolutional layers followed by a max pooling layer as inspired by VGGNet [41], rather than a pooling layer after each convolutional layer such as LeNet [44]. We also use VGGNet as the main inspiration for our own network rather than more contemporary networks like ResNet due to its lack of residual blocks [42]. Moreover, the residual blocks in ResNet require memory to store the intermediate feature maps and more computation. Since wideband spectrum sensing can be seen as a multi-label classification problem, the sigmoid function is used as the output activation function with binary cross-entropy as the loss function. This way, each output neuron is independent of the other in its final value, creating a multi-label classifier.

Layer (Activation)	LTE/SDR Dataset (Channels Last)	GVT Dataset (Channels Last)
Input	(32,2)	(128,2)
Conv1 (LeakyRelu)	(30,16)	(124,16)
Conv2 (LeakyRelu)	(28,16)	(120,16)
Pool1	(14,16)	(60,16)
Conv3 (LeakyRelu)	(10,32)	(50,32)
Conv4 (LeakyRelu)	(6,32)	(40,32)
Pool2	(3,32)	(20,32)
FC (LeakyRelu)	64	32
Output (Sigmoid)	16/4	14

TABLE I: Summary of CNN dimensions for all datasets.

To train our CNNs, we used the popular Adam optimizer. We used additional training techniques to help convergence towards a minima. First, dropouts are used after each stack of convolutional/pool layers to help with overfitting and generalization. Next, we reduce the learning rate adaptively whenever there is a plateau in training by a factor of 10, this helps the loss not overshoot minimas by slowing down when coming close to one. Training is stopped adaptively whenever there is no progress for a large number of epochs and the network that experiences the best validation results is saved. We use 80%, 10%, and 10% of the data for training, validation and testing.

B. Real-Time Hardware Constraints

Long computation times will inevitably result in missed opportunities as detected holes would be too far in the past to be used. In addition, the sensor should be easily implemented in user devices, *e.g.*, mobile nodes, meaning that the sensing logic should use as low memory as possible. At the receiver side, let us consider a spectrum sensor sampling the spectrum at S samples/second, and taking as input N I/Q samples. This makes the time resolution of a hole, *i.e.*, the minimum detectable hole size in time, to be $T = N/S$ seconds which is also the time to gather an input for the sensor. If the system demands for a frequency resolution, *i.e.*, the hole bandwidth, of B Hz then the hole resolution becomes $R = B/T$ Hz/s, which indicates the amount of empty bandwidth the sensor can detect per unit of time. Let us now assume that the CNN produces an output with a total latency of L seconds. To be effective, the system must be able to detect holes whose duration in time H satisfies the relationship $H > L + T$, We can therefore see how crucial the input size and system latency is for the functionality and usability of a spectrum sensor.

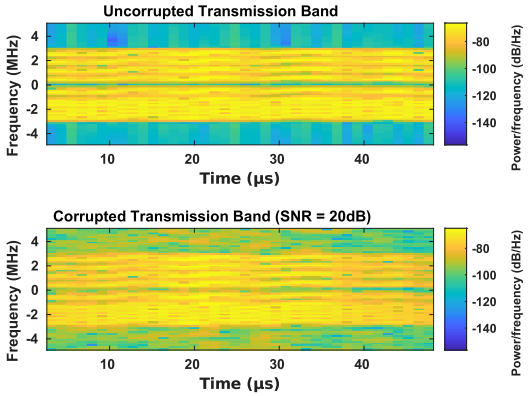

Fig. 4: Spectrogram of LTE-M uplink transmission bands, with and without channel effects.

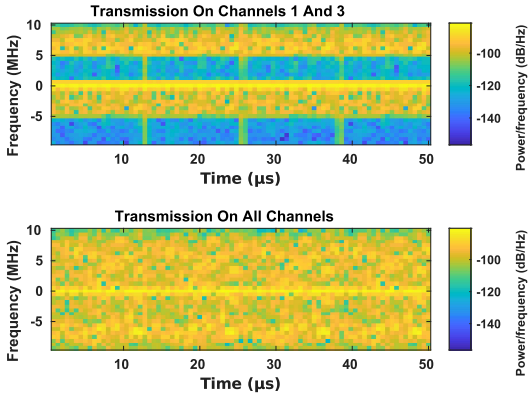

Fig. 5: Spectrogram at receiver on day 1 of SDR dataset collection in two different scenarios.

Another important relationship is that between the accuracy of the sensor and the physical constraints of the transmitter node. For example, any wireless device has limited transmission buffers. If the transmitter is allowed to opportunistically fill holes whenever the sensor detects one, the aforementioned buffers will be emptied at a frequency that depends on how fast holes appear in the spectrum, and how good and fast is the sensor in detecting these holes. To better understand how these elements are tightly intertwined to one another, let us consider a $M/M/1/k$ queueing system. In this case, the queue is represented by the transmission buffer with limited capacity k. The queue is filled at rate λ which defines the rate at which packets to be transmitted are generated. The sensor represents the server of the queue, and μ' denotes the rate at which spectrum holes are detected by the sensor, which in this case corresponds to the rate at which packets can be transmitted over each detected hole. Let μ be the rate at which spectrum holes appear in the spectrum. Moreover, let p^D be the probability that the sensor (a CNN in our case) detects a hole (this includes both true positives and false positives), and let p^F denote the probability that the CNN detects a hole but the prediction is wrong (i.e., false positive). It is straightforward to notice that the rate at which packets are transmitted is $\mu' = \mu * p^D$, and the traffic intensity is $\rho = \lambda/\mu'$. This implies that the queue is stable if and only if $\rho = \lambda/\mu\frac{1}{p^D} < 1$. At a first glance, one might think that designing a sensor that detects holes with $p^D = 1$ might be the best solution. However, in this case the sensor might result in high false positives which indeed result in increasing the number of transmitted packets, but also results in high number of collisions that would eventually decrease the actual throughput of the network. Another naive solution might be to increase the size k of the buffer, reduce the packet generation rate λ and prevent overflow. Indeed, this solution is inefficient as it impacts the throughput of the system and require larger memories to store more packets, which is unpractical for most wireless applications.

The most efficient solution is instead to design a sensor that detects holes with extremely high probability p^D while providing a close-to-zero false positive rate p^F. As we will show in this Section V, our solution succeeds in both tasks by providing an extremely high detection accuracy with sub-millisecond reaction time.

IV. Experimental Datasets

We extensively evaluate DeepSense on three datasets, each with different channel and transmission properties. We have collected two of our own datasets, one from a synthetic channel (LTE) and the other "in the wild" (SDR). The third dataset (GVT) is a more heterogeneous and extensive dataset also collected "in the wild" but spans a much larger bandwidth than the previous two. By fabricating our own dataset we are also able to control for label imbalances that can hinder training and performance.

A. LTE Dataset

The first dataset we utilize contains about 500,000 transmissions, and contains 16 sub-bands, the highest amount to classify out of all three datasets. This dataset is created utilizing MATLABs LTE Toolbox to simulate LTE-M uplink transmissions in the Physical Uplink Shared Channel (PUSCH) over a 10-MHz-wide band. We split the spectrum band into 16 non-overlapping sub-bands, each acting as their own channel allocated to a user. We chose this wireless scenario since it allows us to evaluate our framework over more controlled channel conditions and to vary the SNR precisely. Furthermore, we wanted to evaluate our CNNs performance with a protocol operating in a licensed band.

Figure 4 depicts the spectrogram of the simulated transmission band without and with the channel effects in a high SNR scenario of 20dB for the latter. Additive White Gaussian Noise and Rayleigh fading have been added to each transmission to simulate a typical non-line-of-sight channel in addition to adding difficulty to the classification process. The CNN is also trained and tested on different SNRs. There are a total of 2^{16} possible training labels that exist with 16 sub-bands to classify. Each sub-band is 3 resource blocks wide which is 540 kHz in bandwidth. The first and last resource blocks of the whole 10MHz band have been omitted. We chose 3 resource blocks as our hole resolution as it can fit many wireless communication protocols (Zigbee, LoRa, etc.) with room for guard bands. All other networks that require more bandwidth can aggregate multiple sub-bands.

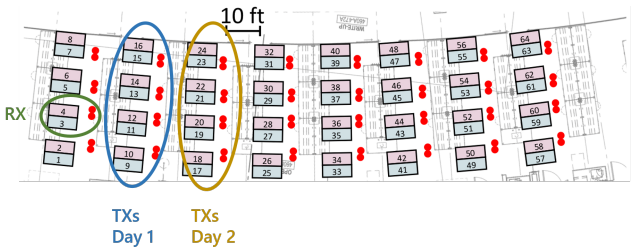

Fig. 6: Experimental setup for small-scale, SDR dataset collection

B. SDR Dataset

The second dataset we collect contains about 500,000 transmissions of four 5-MHz-wide non-overlapping channels, occupying a total of 20 MHz bandwidth. We collected this dataset using 5 USRP N210s SDRs running GNU Radio. Four USRPs acted as WiFi transmitters, each with 64 sub-carriers, and one as the receiver sampling at 20MS/s to obtain the whole bandwidth. Data was collected from two separate days with two different transmitter orientations to give the dataset diversity in the SNR and channel effects it contained. The experimental setup is shown in Figure 6.

In detecting four channels, there are a total of 2^4 possible outcomes at the last layer of the CNN. Because we have personally collected this dataset, each outcome has a close to equal number of training samples. Figure 5 shows the spectrogram that the receiver experiences on the first day of data collection when only 2 of the USRPs are transmitting and all of the USRPs are transmitting. These are represented at the output of CNN as [0 1 0 1] and [1 1 1 1] respectively. Here, "1" is an occupied channel.

C. GVT Dataset

This dataset contains 675,000 IEEE 802.11a transmissions stored in Signal Metadata Format (SigMF) format. Each entry is associated with two files, *i.e.*, sample and metadata files. The sample file contains unprocessed I/Q samples corresponding to a full WiFi packet transmission in one of the 14 available WiFi channels. The metadata file, instead, provides useful information on the transmission such as transmitting device identifier, duration of the transmission, guard times before and after each transmission, sampling frequency and occupied WiFi channel (out of the 14 available channels), among others. The dataset is extremely heterogeneous in terms of transmitting devices (*e.g.*, laptops, smartphones, as well as software-defined radios), packet length and sampling frequency ranging from 20MS/s to 200MS/s. Although each entry in the metadata file is related to a specific devices transmitting a single WiFi packet on a single WiFi channel, the sampling frequency is high enough that the corresponding IQ samples cover the whole ISM band, thus providing an accurate snapshot of the 2.4GHz WiFi frequency band. As an example, Fig. 8 shows the spectrogram of an entry in the dataset. For each entry, the metadata file specifies the length and occupied channel of the corresponding WiFi transmission (i.e., the packet highlighted with blue dashed lines in Fig. 8). However, it is straightforward to notice that the spectrogram also provides information on other WiFi transmissions happening on other channels as well as spectrum holes in the same frequency band.

Generating Ground Truth for GVT. Let us now present an automated procedure to generate a spectrum hole ground truth to train and test DeepSense. As mentioned before, each entry provides information on a single transmission only disregarding other transmissions happening on other channels at the same time. However, IQ samples give an accurate snapshot of the WiFi spectrum where it is possible to clearly identify packet transmissions and spectrum holes. We leverage guard times specified in the metadata files to design an automated ground truth generation pipeline that makes it possible to determine which channels are occupied—and which are idle—at each time instant. As shown in Fig. 8, all entries in the dataset are associated to two guard times at the beginning and at the end of each transmission. Since those guard times never contain any transmission, we can use them to compute the noise power which will be used as a threshold to determine which channels are occupied, and which are idle. We extract the spectrogram of each WiFi channel individually and compare the power level on each frequency component to the measured guard time noise power. Finally, we generate a binary matrix showing which frequency components exceed the noise power (*e.g.*, busy) and which do not (*e.g.*, idle).

We have compared the noise power of guard times with that of other idle channels in the same frequency bands and we have found only negligible differences in the measured noise levels, thus making the guard time noise power a reliable threshold to generate our ground truth. The complete spectrum hole detection procedure we have designed is summarized in Fig. 7. We first extract each entry from the SigMF dataset (Step 1) and compute the corresponding spectrogram (Step 2). Then we extract the WiFi 2.4GHz spectrum bands (Step 3) and execute the power detection procedure described in Fig. 8 (Step 4). Finally we generate the per-channel occupancy matrix specifying which channels are to be considered idle/busy at each time instant (Step 5). This is achieved by measuring the percentage p_s of frequency components in each WiFi channel that have been detected as busy. If $p_s > p_{th}$, with $p_{th} \in (0, 1]$, we label the channel as busy, or idle otherwise. The p_{th} parameter is tunable and can be used to generate ground truth for a variety of applications. $p_{th} = 1$ means that the channel is labeled as busy only if all frequency components of a specific channel are utilized. Any other $p_{th} < 1$ make it possible to mark as partially idle those channels where only p_{th} percent of channel bandwidth is used, thus leaving room for transmission of smaller bandwidth signals to fill the available holes. An example with $p_{th} = 0.7$ (*i.e.*, channel is labeled as busy if more than 70% of available bandwidth is utilized) is shown in Step 5.

D. Performance Metrics

A multi-label classification problem means that each input sample can be a part of more than one class, which in our case translates to a whole band having multiple sub-bands that are unoccupied. Precision, recall, and f1-score are the general metrics used to assess the performance of a classifier:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives + False Positives}}, \qquad (1)$$

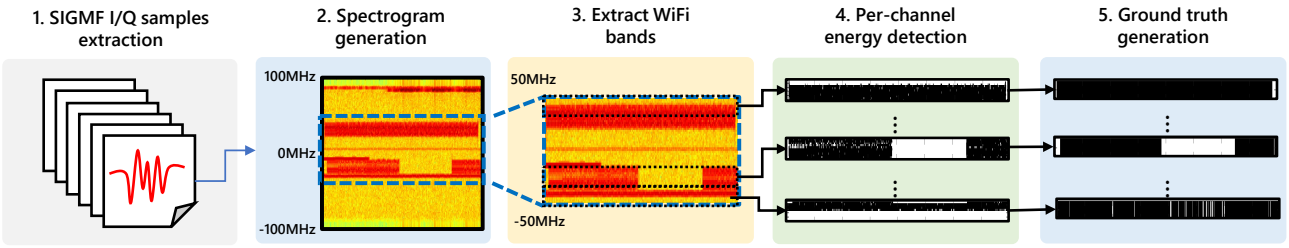

Fig. 7: Ground truth generation pipeline for GVT dataset.

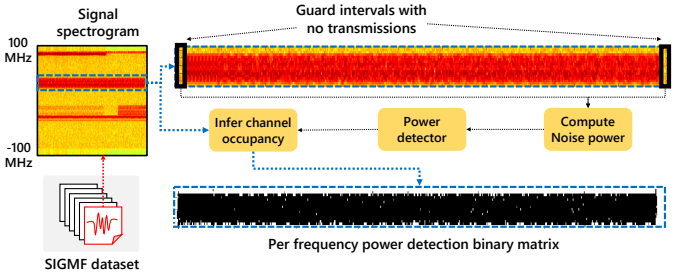

Fig. 8: Power-based spectrum hole detection used in Step 4 of Fig. 7.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}, \tag{2}$$

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision + Recall}}. \tag{3}$$

Precision is essentially a measure of false positives, recall is a measure of false negatives, and F1-score is the harmonic mean of the precision and recall. Since this is a multi-label problem, each class has its own value for these metrics. Thus, to get a sense of the overall classifier performance, these values need to be averaged amongst classes. Macro-averaging and micro-averaging are the standard techniques for averaging performance metrics in multi-label classification problems. Macro-averaging is the more traditional form of averaging and gives a better sense of the classifier performance when the data isn't biased. Micro-averaging is better suited for multi-label classification especially if the data is unbalanced. The macro-average and micro-average for precision are:

$$P_{mac} = \frac{\sum_{c=1}^{C} (\text{Precision})_c}{C}, \tag{4}$$

$$P_{mic} = \frac{\sum_{c=1}^{C} (\text{True Positives})_c}{\sum_{c=1}^{C} (\text{True Positives})_c + (\text{False Positives})_c}, \tag{5}$$

where C is the total number of classes or channels for our application and P_{mac} and P_{mic} are the macro-average and micro-average of precision respectively. These average would be calculated the same way for recall except false negatives are used in the micro-average calculation. Finally the micro-average of the f1-score is the harmonic mean of the recall and precision micro-average. When the different classes have an equal distribution or the same number of occurrences the micro-average and macro-average become the same.

Energy Detector. To compare our results with the a more traditional method of spectrum sensing, we employed an energy detector. The energy detector works by first performing an N-point FFT on N IQ samples. The energy for each channel or sub-band is

$$E_c = \frac{1}{N_c} \sum_{n=1}^{N_c} |R[n]|^2, \tag{6}$$

where E_c is the energy of channel c, $R[n]$ is the received signal in frequency, and N_c is the number of IQ samples in channel c. For the energy detector to classify a channel as occupied then $E_c > \tau$, where τ is the static energy threshold. We can see that the smaller the input sample is of the received signal, the less data the energy detector has to work with to make its decision. While this technique may work in narrowband sensing of a single channel, it will require prior knowledge of the channel and that the SNR remain relatively the same.

V. Experimental Results

A. LTE Dataset Results

1) CNN Performance: Different testing data is generated with varying SNRs. Figure 9 shows the micro-average of the performance metrics with varying SNR. Because the data is balanced the macro and micro averages are the same therefore only the micro-average is plotted. It can be seen that under fair to great SNR conditions such as would be experienced at 10dB and 20dB respectively, the CNN performs in the 90th percentile with near perfect performance at 20dB, only utilizing 32 IQ samples.

Figure 11 shows the performance metrics at 10dB SNR as they apply to each channel or sub-band. These results are more notable as this SNR can be considered at the mid-cell level in LTE. The edge channels experience slightly better performance than those in the middle in terms of f1-score. This is most likely due to them only experiencing side bands from one channel and therefore not getting corrupted by the side bands of multiple channels.

2) Energy Detector Performance: To give a fair comparison with our method and the energy detector, we utilized the same testing data. We varied the input sizes, n_i, and energy threshold of the detector. The range of energy thresholds used are the ones that experience the best classification performance for the given dataset.

Figure 10 shows the micro average performance metrics as a function of the energy threshold when tested on the 20dB SNR data. The figure shows that even with great SNR conditions the energy detector experiences overall worse performance than the CNN in addition to an inverse relationship between recall and precision. The overall performance is hindered because with

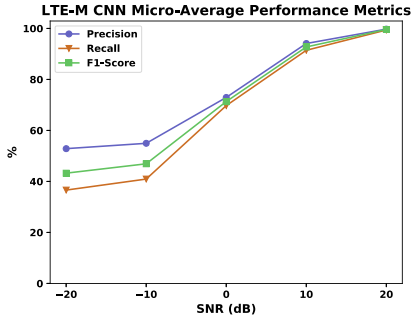

Fig. 9: Micro-average performance metrics of the CNN as a result of varying SNR on simulated LTE-M dataset

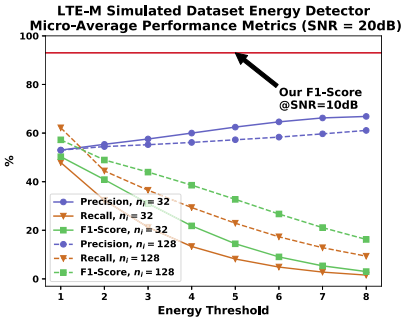

Fig. 10: Micro-average performance metrics of the energy detector as a result of varying energy thresholds on the simulated LTE dataset with an SNR of 20dB, our f1-score at 10dB is plotted for comparison

such a small input size and such a large number of channels to classify, the energy detector has to work with data that is low in resolution in the frequency domain. For perspective, when taking 32 IQ samples, converting them to frequency and splitting them up into 16 groups, each sub-band only has 2 IQ samples in frequency for deducing sub-band occupancy. The energy detector therefore requires more IQ samples to collect and a larger FFT to compute.

The inverse relationship between recall and precision occurs because as the energy threshold becomes stricter or higher, there are less false positives and therefore more precision. However, because of that same strictness, many channels that are occupied but have low energy do not make the cut off. This results in more false negatives and a decrease in recall.

We can also see that increasing n_i by a factor of 4 compared to the input size used at the CNN still does not match the energy detectors overall performance to that of the CNN, though it does improve the recall and f1-score.

B. SDR Dataset Results

1) CNN Performance: The classification task for this dataset is less difficult compared to the previous dataset as the number of IQ samples stay the same but the number of output classes are reduced to 4. This makes an easier input to output mapping for the CNN as well as more IQ samples per sub-band for the energy detector. But where difficulty in mapping inputs to outputs decreases, difficulty in utilizing less controlled data increases due to the nature of "in the wild" signals. Therefore SNR could not be controlled as this is a real world dataset and the same applies for the GVT dataset.

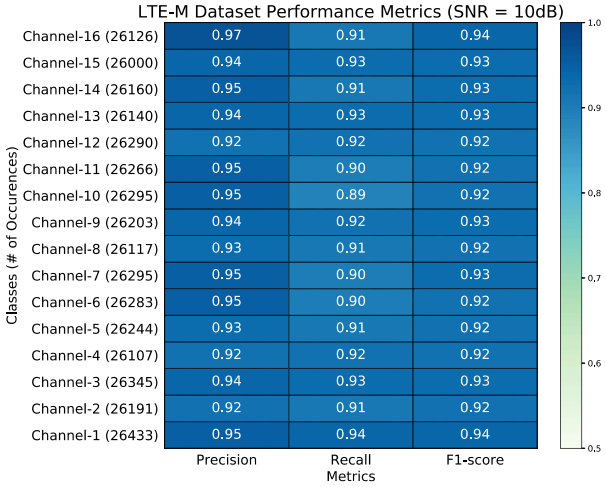

Fig. 11: Performance metrics per channel on simulated LTE-M, non-overlapping, 16 channel/sub-band dataset

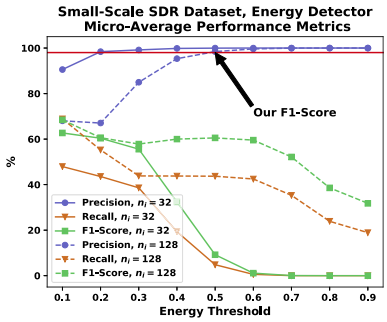

Fig. 12: Micro-average performance metrics of the energy detector as a result of varying energy thresholds on the small-scale SDR dataset, our f1-score is plotted for comparison

We can see from Figure 13 that the each channel experiences near perfect performance. The transmitter on the last channel experienced slightly worse transmission and therefore poorer performance relative to the others. These results show that the CNN takes advantage of the smaller hole resolution and is able to optimize precision and recall highly without affecting one another. The CNN experienced micro-averages of 98%, 97%, 98% for precision, recall, and f1-score respectively.

2) Energy Detector Performance: The energy detector experiences better performance with this dataset than the LTE-M simulation as can be seen in Figure 12. This is due to the lower hole resolution. In terms of precision, the energy detector slightly outperforms the CNN, however the other metrics still suffer tremendously. Due to the nature of threshold based energy detection this inverse relationship between precision and recall will always exist. Increasing the number of input samples helps the overall performance of the energy detector but it needs more input samples.

C. GVT Dataset Results

Figure 14 and Table II summarize the performance metric results on the large scale GVT dataset. The overlapping channels and heterogeneity of this dataset increase the classification task considerably. The CNN could not be compared to an energy detector in this case as an energy detector was used to generate the ground truth of this dataset. Nonetheless, the performance

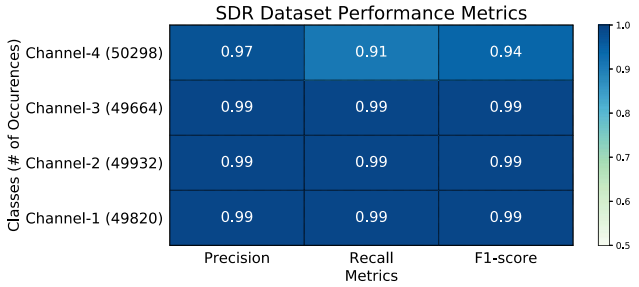

Fig. 13: Performance metrics per channel on non-overlapping, 4 channel dataset collected with SDRs

is still promising with precision and recall maintaining a proportional relationship. Figure 14 shows a similar pattern in the edge channels to that of the simulated dataset. However, the disparity in channel performance is a little more prominent here, this can be seen in the larger difference between performance metrics between channels. The overlapping channels cause this decrease in performance.

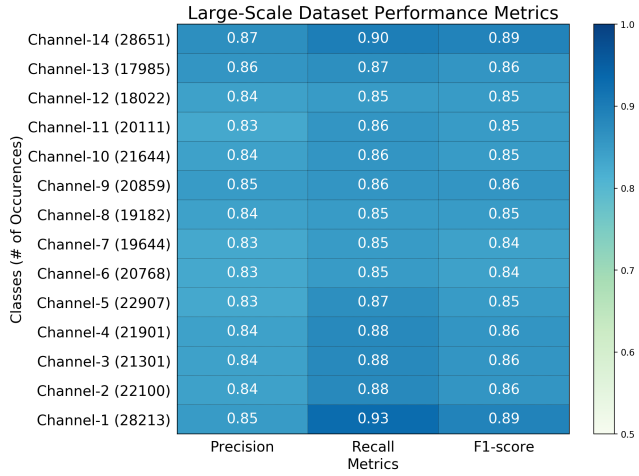

Fig. 14: Performance metrics per channel on large-scale GVT, overlapping, 14 channel dataset

TABLE II: CNN performance metric averages for large-scale, GVT dataset

Averages	Precision	Recall	F1-Score
Micro Avg	0.84	0.87	0.86
Macro Avg	0.84	0.87	0.86

D. Real-Time Hardware Performance

The CNNs are implemented on an FPGA through High Level Synthesis then run on a Xilinx Zynq-7000 SoC ZC706 to test latency. Table III shows the size of the CNNs and performance on the FPGA for the three datasets. The CNN for the LTE-M and small-scale datasets were very similar in size and therefore experience similar latency. Both of their times are faster than the length of one LTE sub-frame with high performance. The last dataset required a larger number of input samples to classify due to its vast heterogeneity when compared to the previous datasets. This caused the CNN to be larger than previous and therefore incurred more latency. The latency for this CNN was increased to 5.1ms on the FPGA. However the classification performance in the midst of such a diverse dataset, especially one that was not fabricated for the purpose of this classification task is notable.

All CNNs occupied minimal space on the FPGA. Our largest CNN is multiple orders of magnitude smaller than conventional CNNs used for image processing and utilizes a fewer amount of filters and smaller input sizes than networks in [19, 21, 31]. This is crucial to a real time realization of a high performing wideband spectrum sensor, because long processing means missed opportunities. In addition the small CNN size makes *DeepSense* easily implemented on user devices or mobile nodes. However, the trade-off of low latency is decreased generalization and performance as low SNR regimes prove difficult for our classifier.

TABLE III: DeepSense FPGA performance on different datasets

Dataset	# of CNN Parameters	Latency (ms)	FPGA Utilization
LTE-M	12,272	0.62	4.25%
Small-Scale	15,108	0.67	4.25%
Large-Scale	39,406	5.1	7%

VI. Conclusions

In this paper, we have proposed *DeepSense*, a software/hardware framework for real-time wideband spectrum sensing. In short, *DeepSense* relies on real-time deep learning tightly integrated into the transceiver's baseband processing logic to detect and exploit unutilized spectrum bands with the least amount of I/Q samples. We have extensively validated *DeepSense* through three different datasets under a wide variety of SNR conditions. We have also measured the real-time latency of the CNNs trained on the three datasets with an FPGA implementation. Finally, we have compared our approach with a fixed energy threshold mechanism. Our experimental results have shown that our learning-based approach can deliver a precision and recall of 98% and 97% respectively and a latency as low as 0.61ms. An an additional contribution, we have pledged to share the code and the datasets used in this paper to the community, for both reproducibility and benchmarking purposes. *DeepSense*'s fast detection capabilities paves the way for concrete opportunistic access in licensed and unlicensed bands by primary or secondary users. The low computational complexity of the system furthermore makes *DeepSense* easy to implement on mobile nodes. However, *DeepSense*'s spectrum hole filling component has not been addressed in this paper, since it is a complex endeavor that deserves a separate investigation. We are currently working on a system-on-chip implementation of *DeepSense* that will close the loop between spectrum sensing and actuation.

REFERENCES

[1] Federal Communications Commission (FCC), "Spectrum Crunch," https://www.fcc.gov/general/spectrum-crunch.

[2] Jeremy Horwitz, Venture Beat, "Wi-Fi 6E and 5G Will Share 6GHz Spectrum to Supercharge Wireless Data," https://tinyurl.com/wyvmn5c, 2020.

[3] L. Zhang, M. Xiao, G. Wu, M. Alam, Y.-C. Liang, and S. Li, "A Survey of Advanced Techniques for Spectrum Sharing in 5G Networks," *IEEE Wireless Communications*, vol. 24, no. 5, pp. 44–51, 2017.

[4] F. Hu, B. Chen, and K. Zhu, "Full Spectrum Sharing in Cognitive Radio Networks Toward 5G: A Survey," *IEEE Access*, vol. 6, pp. 15 754–15 776, 2018.

[5] H. Shokri-Ghadikolaei, F. Boccardi, C. Fischione, G. Fodor, and M. Zorzi, "Spectrum sharing in mmwave cellular networks via cell association, coordination, and beamforming," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 11, pp. 2902–2917, 2016.

[6] L. Lv, J. Chen, Q. Ni, Z. Ding, and H. Jiang, "Cognitive non-orthogonal multiple access with cooperative relaying: A new wireless frontier for 5g spectrum sharing," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 188–195, 2018.

[7] Federal Communications Commission (FCC), "FCC Opens 6 GHz Band to Wi-Fi and Other Unlicensed Uses," https://www.fcc.gov/document/fcc-opens-6-ghz-band-wi-fi-and-other-unlicensed-uses, 2020.

[8] Jamie Davies, Telecoms.com, "FCC finally opens up 3.5 GHz for US telcos," https://telecoms.com/502070/fcc-finally-opens-up-3-5-ghz-for-us-telcos/, 2020.

[9] J. Kim and J. G. Andrews, "Sensitive White Space Detection With Spectral Covariance Sensing," *IEEE Transactions on Wireless Communications*, vol. 9, no. 9, pp. 2945–2955, 2010.

[10] K. W. Choi and E. Hossain, "Opportunistic Access to Spectrum Holes Between Packet Bursts: A Learning-based Approach," *IEEE Transactions on Wireless Communications*, vol. 10, no. 8, pp. 2497–2509, 2011.

[11] I. F. Akyildiz, B. F. Lo, and R. Balakrishnan, "Cooperative Spectrum Sensing in Cognitive Radio Networks: A Survey," *Physical Communication*, vol. 4, no. 1, pp. 40–62, 2011.

[12] H. Sun, A. Nallanathan, C.-X. Wang, and Y. Chen, "Wideband Spectrum Sensing for Cognitive Radio Networks: a Survey," *IEEE Wireless Communications*, vol. 20, no. 2, pp. 74–81, 2013.

[13] K. Cichoń, A. Kliks, and H. Bogucka, "Energy-Efficient Cooperative Spectrum Sensing: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1861–1886, 2016.

[14] A. Ali and W. Hamouda, "Advances on Spectrum Sensing for Cognitive Radio Networks: Theory and Applications," *IEEE communications surveys & tutorials*, vol. 19, no. 2, pp. 1277–1304, 2016.

[15] Y. Ma, Y. Gao, Y.-C. Liang, and S. Cui, "Reliable and Efficient Sub-Nyquist Wideband Spectrum Sensing in Cooperative Cognitive Radio Networks," pp. 2750–2762, 2016.

[16] B. Hamdaoui, B. Khalfi, and M. Guizani, "Compressed Wideband Spectrum Sensing: Concept, Challenges, and Enablers," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 136–141, 2018.

[17] X. Jin and Y. Zhang, "Privacy-preserving Crowdsourced Spectrum Sensing," *IEEE/ACM Transactions on Networking*, vol. 26, no. 3, pp. 1236–1249, 2018.

[18] H. Qi, X. Zhang, and Y. Gao, "Channel Energy Statistics Learning in Compressive Spectrum Sensing," *IEEE Transactions on Wireless Communications*, vol. 17, no. 12, pp. 7910–7921, 2018.

[19] C. Liu, J. Wang, X. Liu, and Y.-C. Liang, "Deep CM-CNN for Spectrum Sensing in Cognitive Radio," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2306–2321, 2019.

[20] H. Qi, X. Zhang, and Y. Gao, "Low-Complexity Subspace-Aided Compressive Spectrum Sensing Over Wideband Whitespace," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 12, pp. 11 762–11 777, 2019.

[21] D. Chew and A. B. Cooper, "Spectrum Sensing in Interference and Noise Using Deep Learning," in *Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2020, pp. 1–6.

[22] Y. Arjoune and N. Kaabouch, "A comprehensive survey on spectrum sensing in cognitive radio networks: Recent advances, new challenges, and future research directions," *Sensors*, vol. 19, no. 1, p. 126, 2019.

[23] 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, Feb 2018.

[24] W. Zhang, R. K. Mallik, and K. B. Letaief, "Optimization of cooperative spectrum sensing with energy detection in cognitive radio networks," *IEEE transactions on wireless communications*, vol. 8, no. 12, pp. 5761–5766, 2009.

[25] Z. Tian and G. B. Giannakis, "A wavelet approach to wideband spectrum sensing for cognitive radios," in *2006 1st international conference on cognitive radio oriented wireless networks and communications*. IEEE, 2006, pp. 1–5.

[26] F. Restuccia and T. Melodia, "Big Data Goes Small: Real-Time Spectrum-Driven Embedded Wireless Networking Through Deep Learning in the RF Loop," *Proc. of IEEE Conference on Computer Communications (INFOCOM)*, 2019.

[27] I. E. Telatar and D. N. C. Tse, "Capacity and Mutual Information of Wideband Multipath Fading Channels," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1384–1400, 2000.

[28] A. Ranjan, B. Singh *et al.*, "Design and analysis of spectrum sensing in cognitive radio based on energy detection," in *2016 International Conference on Signal and Information Processing (IConSIP)*. IEEE, 2016, pp. 1–5.

[29] M. Z. Alom, T. K. Godder, M. N. Morshed, and A. Maali, "Enhanced spectrum sensing based on energy detection in cognitive radio network using adaptive threshold," in *2017 International Conference on Networking, Systems and Security (NSysS)*. IEEE, 2017, pp. 138–143.

[30] P. S. Yawada and A. J. Wei, "Cyclostationary detection based on non-cooperative spectrum sensing in cognitive radio network," in *2016 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*. IEEE, 2016, pp. 184–187.

[31] J. Gao, X. Yi, C. Zhong, X. Chen, and Z. Zhang, "Deep learning for spectrum sensing," *IEEE Wireless Communications Letters*, vol. 8, no. 6, pp. 1727–1730, 2019.

[32] Z. Quan, S. Cui, A. H. Sayed, and H. V. Poor, "Wideband spectrum sensing in cognitive radio networks," in *2008 IEEE international conference on communications*. IEEE, 2008, pp. 901–906.

[33] I. Raghu, S. S. Chowdary, and E. Elias, "Efficient spectrum sensing for cognitive radio using cosine modulated filter banks," in *2016 IEEE region 10 conference (TENCON)*. IEEE, 2016, pp. 2086–2089.

[34] Y. Wang and G. Zhang, "Compressed wideband spectrum sensing based on discrete cosine transform," *The Scientific World Journal*, vol. 2014, 2014.

[35] B. Farhang-Boroujeny, "Filter bank spectrum sensing for cognitive radios," *IEEE Transactions on signal processing*, vol. 56, no. 5, pp. 1801–1811, 2008.

[36] H. Sun, W.-Y. Chiu, and A. Nallanathan, "Adaptive compressive spectrum sensing for wideband cognitive radios," *IEEE Communications Letters*, vol. 16, no. 11, pp. 1812–1815, 2012.

[37] T. J. O'shea and N. West, "Radio machine learning dataset generation with gnu radio," in *Proceedings of the GNU Radio Conference*, vol. 1, no. 1, 2016.

[38] R. F. Molanes, J. J. Rodríguez-Andina, and J. Fariña, "Performance characterization and design guidelines for efficient processor - FPGA communication in Cyclone V FPSoCs," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 5, pp. 4368–4377, May 2018.

[39] Pete Bennett (EE Times), "The Why, Where and What of Low-Power SoC Design," https://www.eetimes.com/document.asp?doc_id=1276973, 2004.

[40] S. Riyaz, K. Sankhe, S. Ioannidis, and K. Chowdhury, "Deep Learning Convolutional Neural Networks for Radio Identification," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 146–152, 2018.

[41] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[43] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[44] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.