American University of Sharjah
College of Engineering
Department of Computer Science & Engineering

Cognitive Radio Spectrum Sensing and Allocation: A Low-Complexity Deep Learning Approach

Senior Design Project Report

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Bachelor of Science in Computer Engineering

Presented By

Name	Major	ID
Hamza Ahmed Abushahla	COE	90279
Ghanim Abdulla Alarai Al-Ali	COE	89804
Sultan Khalid Abdalla	COE	89987
Muhammad Ismail Sadaqat	COE	90340

Date of Submission:

04/05/2024

Advised By

Dr. Mohamed AlHajri
Dr. Taha Landolsi

Abstract

In the field of wireless communication, the growing demand for bandwidth is pushing the limits of the conventional radio frequency (RF) spectrum allocation paradigms. The current static allocation methods are not keeping up with the growing demands for more dynamic and effective solutions. This challenge serves as the driving motivation for this project, which seeks to enhance spectrum efficiency through the of an intelligent spectrum sensing (SS) and allocation system utilizing cognitive radio (CR) technology. The primary objective of the project is to develop an advanced algorithm that can identify underutilized spectrum bands, and then dynamically allocate them to secondary users (SUs), all while ensuring minimal interference with primary users (PUs). Methodologically, the project proposes a low-complexity deep learning (DL) based algorithm, aiming to surpass the limitations of traditional SS techniques. In addition, the project will encompass a hardware implementation that will demonstrate and validate the algorithm's usefulness in real-world scenarios. The project is expected to produce significant benefits, as it aims to support the growing demands from mobile and internet of things (IoT) devices in addition to enhancing spectral efficiency. This is especially important for the development of wireless technologies of the next generation. Beyond technical merits, the project is poised to have a substantial impact on global, economic, and societal levels, highlighting the transformative potential of DL in the field of spectrum management.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AGM	Arithmetic to Geometric Mean
AI	Artificial Intelligence
ANN	Artificial Neural Network
APASS	Activity Pattern Aware Spectrum Sensing
ASK	Amplitude Shift Keying
AWGN	Additive White Gaussian Noise
BAQ	Bandwidth Allocation Queue
BPSK	Binary Phase-Shift Keying
CA	Classification Accuracy
CBD	Covariance-Based Detection
CED	Conventional Energy Detection
CFD	Cyclostationary Feature Detection
CNN	Convolutional Neural Network
CR	Cognitive Radio
CSS	Cooperative Spectrum Sensing
DL	Deep Learning
DSA	Dynamic Spectrum Allocation
ED	Energy Detection
EM	Expected Maximum
FCC	Federal Communications Commission
FCFS	First-Come-First-Serve
FDMA	Frequency Division Multiple Access
FFT	Fast Fourier Transform
FHSS	Frequency-Hopping Spread Spectrum
FSK	Frequency Shift Keying
GMM	Gaussian Mixture Model
IoT	Internet of Things
KNN	K-Nearest Neighbors
LSTM	Long Short-Term Memory
MBJD	Multi-band Joint Detection
MDP	Markov Decision Process
MED	Maximum Eigenvalue Detection

MFD	Matched Filter Detection
ML	Machine Learning
N-CNN	Unstandardized Convolutional Neural Network
PAM	Pulse Amplitude Modulation
PSD	Power Spectral Density
PU	Primary User
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
RF	Radio Frequency
RIP	Restricted Isometry Properties
RL	Reinforcement Learning
S-CNN	Standardized Convolutional Neural Network
SARS	State-Action-Reward-State
SDR	Software Defined Radio
SNR	Signal-to-Noise Ratio
SS	Spectrum Sensing
SSE	Signal Subspace Eigenvalues
SU	Secondary User
SUQ	Secondary Users Queue
SVD	Singular Value Decomposition
SVM	Support Vector Machine
UI	User Interface
USRP	Universal Software Radio Peripheral

1. Introduction

The past few years have witnessed remarkable advances in wireless communication technologies, giving rise to a rapid expansion in wireless applications. The exponential growth of mobile devices, the internet of things (IoT), and the ease of accessing wireless devices has unleashed an unprecedented demand for the available radio frequency (RF) spectrum; a fundamental resource for communications [1]. According to [2], the global number of mobile subscriptions will grow from about 8.4 billion in 2022 to around 9.2 billion by the end of 2028. This surge in connectivity is pushing existing 4G and 5G bands to their limits.

Traditionally, spectrum allocation has been static, with a large part of the radio spectrum assigned by regulatory bodies to licensed primary users (PUs) and left a smaller portion of the spectrum for unlicensed secondary users (SUs) [3]. Having most of the available spectrum statically allocated means that the current standard will be unable to support emerging wireless technologies and applications like 6G, internet of everything, and industry 5.0 [4]. On the other hand, spectrum occupancy measurements have shown that some licensed bands are significantly underutilized. For example, the Spectrum Policy Task Force reported that radio channels are typically occupied 15% of the time with a peak occupancy reaching 85% [5]. These findings suggest that the limitations on available spectrum resources are primarily due to inefficient static allocation methods rather than inherent spectrum scarcity. Consequently, the underutilization of these spectrum resources has led to the development of dynamic spectrum allocation paradigms, commonly referred to as cognitive radio (CR) networks [3].

A CR is essentially a software defined radio (SDR) system that achieves a higher spectral efficiency by allowing SUs, to *sense* the radio spectrum continuously and intelligently and then opportunistically utilize the frequency bands which are not used by PUs [6]. If a frequency band is occupied by a PU, it is called an “active band” or a “black space.” Otherwise, it is called a "spectrum hole" or a "white space" [3], [7]. SUs must make use of vacant spectrum gaps without causing harmful interference to PUs since they have a higher priority to access it

over SUs. Hence, effective, and efficient SS techniques must be used in order to minimize the interference between spectrum users [3].

Furthermore, one of the major challenges in CR system design is spectrum management or spectrum allocation, as the decision of whether or not to allocate a channel to a SU is based on a number of factors, including the availability of the channel, the quality of service (QoS) that the SU can achieve on the channel, and the potential for interference to the PU. SUs must also be able to vacate channels quickly if the PU needs to use them [7].

In this senior design project, we propose a deep learning (DL) based SS and allocation system to intelligently manage spectrum resources for mobile and IoT devices. The proposed algorithm will leverage DL techniques to design a SS algorithm that can accurately detect unused spectrum bands and then intelligently allocate available spectrum resources to SUs in a way that optimizes their performance and minimizes interference to PUs. Furthermore, a hardware platform that showcases the optimized operation of the designed algorithm will be implemented.

1.1 Motivation

CR is currently one of the hottest and most active research topics in wireless communication since it offers a promising alternative to the existing fixed spectrum allocation policies, which are becoming increasingly inefficient as the demand for wireless spectrum continues to grow. Currently, wireless communication heavily relies on traditional SS techniques like energy detection and matched filter detection. These conventional techniques result in inefficient use of the radio spectrum due to missed detection of the PU and false alarms in interference-prone settings.

In recent years, and as artificial intelligence (AI) became more and more popular, researchers have been proposing new approaches to overcome the spectrum scarcity challenge and enhance the efficiency of SS techniques using advanced machine learning (ML) and DL techniques due to their remarkable performance compared to conventional algorithms. DL

provides an exciting alternative, empowering systems to learn and adapt to intricate spectral patterns and interference, overcoming the limitations of existing methods. Employing a similar DL approach to those found in literature in designing an efficient SS and allocation algorithm is expected to address the growing demand for radio resources in modern wireless communication systems.

1.2 Project Description

The main requirements of this project involve the development of a low-complexity DL-based SS and allocation system. The system's design aims to facilitate integration with existing mobile devices, IoT hardware, and network infrastructure, with a key emphasis on requiring minimal processing resources. The primary goal is to unlock the full potential of DL in order to maximize spectrum utilization, addressing a critical need in contemporary wireless communication systems.

This project offers several significant benefits. For example, by incorporating DL, it aims to improve spectrum usage efficiency, ensuring that available frequency bands are optimally utilized. This is particularly crucial given the increasing demand from mobile devices and IoT, coupled with the diminishing radio spectrum.

Furthermore, our project distinguishes itself by aiming to contribute not only algorithmically to the SS field but also by providing a practical implementation that translates pioneering DL-based SS research into a tangible, real-world solution. This dual focus on algorithmic advancement and practical implementation positions our project as a comprehensive and impactful contribution to the evolving landscape of spectrum management.

1.3 Organization of the Report

The report is organized as follows. Section 2 describes the problem statement and design objectives while discussing the proposed solution and the limitations of the project. In Section 3, a detailed literature review on SS, its approaches, spectrum allocation, as well as hardware

platforms is presented along with examples of existing models that could contribute to our project's solution. Section 4 includes the specifications, the functional, and the non-functional requirements of our proposed solution. Onwards, section 5 illustrates the proposed solution and its design through block diagrams and flowcharts. Section 6 discusses the project management plan and the schedule of the project across the Fall 2023 and Spring 2024 semesters. Moreover, Section 7 is dedicated to the implementation aspects of the project, encompassing the hardware setup, software development, and the integration of these components to build the system. Section 8 of the report explains the multiple methods of testing, verification, and validation of the project. Furthermore, Section 9 focuses on the main impacts that the project will have globally, economically, and socially. Section 10 presents the standards that are followed by each component of the project. Finally, section 11 concludes the report with an overall summary of the whole project.

2. Problem Statement and Design Objectives

2.1 Problem Statement and Proposed Solution

The exponential growth of mobile devices, the IoT, and other connected technologies has unleashed an unprecedented demand for spectrum resources, pushing existing 4G and 5G bands to their limits [1]. The impending spectrum scarcity poses a significant threat to communication systems, as it can result in congestion, interference, and compromised performance. Moreover, this problem persists despite the fact that many licensed spectrum bands remain underutilized [8].

To address this challenge, our project aims to develop an intelligent spectrum management system which will harness the power of advanced DL algorithms and CR techniques. At its core, the proposed solution aims to create an algorithm that utilizes SS to monitor and dynamically allocate available spectrum resources to mobile and IoT devices, which in turn aims to optimize the use of available frequency bands. Moreover, this project draws upon existing research that has explored DL-based SS such as [8], [9] with the overarching aim of translating such pioneering research into a practical, real-world implementation. Most importantly, our project aims to encompass an algorithmic contribution to the SS field and a practical implementation of such a system.

2.2 Design Objectives

- **Spectrum Optimization:** Our system should implement a SS and spectrum allocation solution that maximizes the use of available frequency bands, addressing spectrum scarcity concerns.
- **Interference Mitigation:** Our system should minimize interference among devices sharing the same spectrum resources to enhance communication performance.
- **Deep Learning Integration:** Our system should implement advanced DL techniques to enhance SS capabilities, improving accuracy and adaptability.

- **Algorithm Design:** Our system should incorporate a low-complexity SS algorithm design to support the core functionality.
- **Practical Implementation:** Our system should include a practical implementation that integrates the algorithm, allowing for testing and showcasing its optimized operation.

2.3 Design Constraints

- **Cost-Effectiveness:** Design the system within the allocated budget, considering hardware, software, and development costs.
- **Hardware Compatibility:** The system must work with existing mobile devices, IoT hardware, and network infrastructure without requiring significant modifications.
- **Computation Power:** The designed system must have sufficient computation power to run DL algorithms efficiently and effectively.

2.4 Design Assumptions

- **Availability of Spectrum Data:** Assume access to real-time or periodically updated spectrum usage data to inform allocation decisions.
- **Deep Learning Model Availability:** Assume access to suitable advanced DL models for SS and prediction.
- **Sufficient Processing Power:** Assume the system has sufficient computational resources to participate in SS and dynamic spectrum allocation.
- **Data Privacy Compliance:** Assume compliance with data privacy regulations and user consent for spectrum usage data collection.

3. Literature Review

In this section, the fundamental concepts of CR and SS are presented, followed by a literature survey of different SS approaches, covering conventional, ML-based, and DL-based sensing techniques. Thereafter, an overview of spectrum allocation and its types is discussed. Finally, a review of various hardware platforms that implement SS algorithms in real time is provided.

3.1 CR Fundamentals

The electromagnetic radio spectrum is a scarce natural resource that is predominantly used for wireless communication applications. The spectrum is divided into frequency bands, and traditionally, some of these bands were allocated to specific licensed users or application, leaving a smaller portion of the spectrum for unlicensed users. However, due to the increasing popularity of various wireless technologies and services, the demand for more bandwidth and higher data rates has increased over the years. On the other hand, this fixed allocation policy has resulted in some frequency bands being underutilized, leading to a phenomenon known as "spectrum holes," which are defined as bands that are not used by its licensed PUs at a specific time and geographical location [7], [10]. Therefore, spectrum holes are important to study as there is a limited amount of frequency spectrum available, and the demand for wireless communications is growing rapidly. In addition, studies suggest that approximately 80% of the electromagnetic spectrum remains unutilized [5].

Hence, CR was introduced as a new paradigm of designing wireless communications systems aiming at maximizing the utilization of the underutilized RF spectrum and use spectrum holes efficiently. The term CR was first introduced by Joseph Mitola III in 1999 [7]. Since then, CR has been defined in many ways by different entities and researchers. For example, the Federal Communications Commission (FCC) defines CR as "a radio or system that senses its operational electromagnetic environment and can dynamically and

autonomously adjust its radio operating parameters to modify system operation, such as maximize throughput, mitigate interference, facilitate interoperability, access secondary markets" [6].

As shown in Fig. 1, there are two networks in a CR system: a primary network and a secondary or CR network. The primary network owns the licensed band and consists of the PUs like TV broadcasters who own the full right of always using the spectrum band, and the radio base station. The secondary network, on the other hand, which consists of the CR base station and SUs, shares the unused spectrum with the primary network in an opportunistic manner [7].

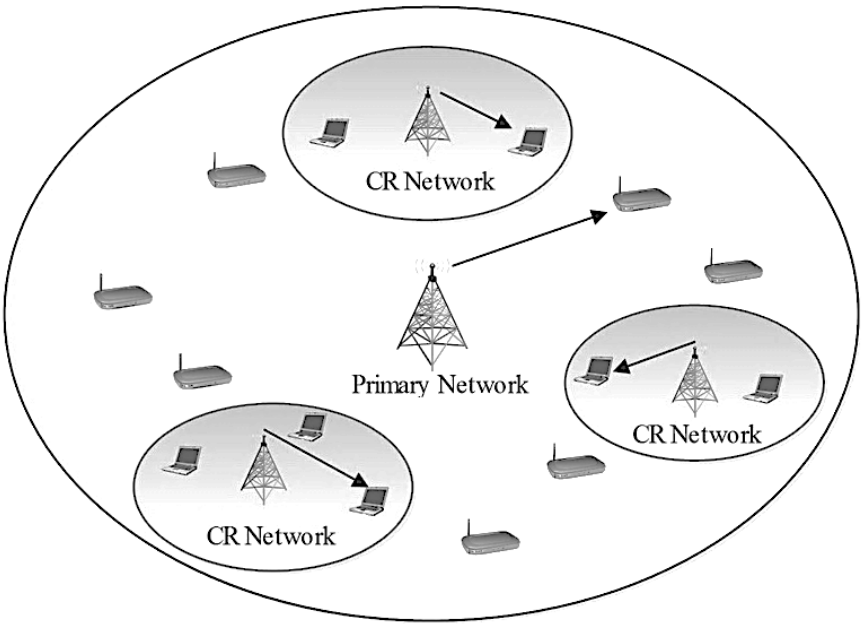


Fig. 1. CR networks within a primary network [10]

The premise of CR lies in its ability to continuously and intelligently *sense* the availability of spectrum holes in terms of duration, frequency, and location, and then opportunistically access these holes for data transmission. When a PU reappears in a spectrum hole that is being used by a CR device, the CR device must vacate the band and find another spectrum hole to use so that the service is not interrupted. This can be illustrated using Fig. 2.

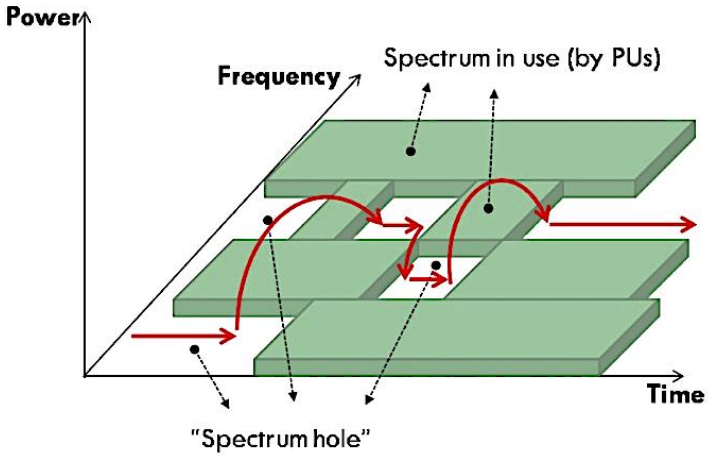


Fig. 2. Time-frequency-power spectrum illustrating spectrum holes [7]

Furthermore, CR networks can be further classified into centralized and distributed networks. In a centralized CR network, the base station is responsible for managing the spectrum resources and coordinating the activities of the SUs. The base station collects SS data from the PUs and uses it to identify and allocate spectrum holes to the SUs. When a PU appears in a spectrum hole that is being used by an SU, the base station must reallocate the SU to another spectrum hole to avoid interference [7]. In this case, the decisions related to spectrum allocations and reallocations are made solely by the base station.

However, Centralized CR networks have two main disadvantages. First, they require additional infrastructure to be deployed in conjunction with the existing PU base stations. Second, the centralized base station is a single point of failure, which means that if the base station fails, the entire network will be disrupted. On the contrary, coordination between SUs is simpler in centralized CR because all decisions are made by a single unit, the base station .

In distributed CR networks, on the other hand, the SUs form a network with each other without the need for a central base station. The SUs themselves make decisions about how to allocate and reallocate spectrum, which requires more complex sharing mechanisms than in centralized CR networks [7]. Fig. 3 shows the two types of CR network architectures.

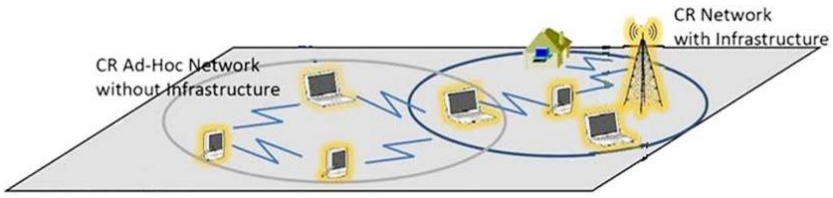


Fig. 3. Centralized and distributed CR networks [7]

The main principle of CR networks, regardless of whether they are centralized or distributed, is dynamic spectrum access, which allows SUs to opportunistically use PU channels when they are not in use. When designing a CR system, two main requirements must be met: (1) The SUs must not interfere with the PUs' use of the spectrum since they have a higher priority to access it. (2) the SUs should try to maximize their channel usage to improve spectrum efficiency [7].

Furthermore, PUs are not obligated to share their spectrum with SUs, and they can change their operating parameters at any time. Therefore, SUs must be able to independently detect spectrum holes without any help from PUs [10]. This ability is called SS, and it is one of the most important components of CR networks. Therefore, it is imperative that SS techniques are not only highly efficient but also designed to minimize interference to the PU while maximizing the utilization of available spectrum resources [3]. To address this critical aspect, the remainder of this section offers comprehensive survey of various SS methods that have been proposed in literature.

3.2 Spectrum Sensing Approaches

In the field of CR, efficient SS is critical for maximizing frequency band utilization. In this section, SS techniques are categorized into conventional, ML-based, and DL-based sensing approaches. Each type has different methods with unique strengths and weaknesses. The best approach depends on the specific requirements and constraints of the wireless communication environment.

Fig. 4. presents a visual overview of the different SS techniques reviewed in this section, which will serve as a reference as each approach is discussed in more detail. The aim is to gain a deeper understanding of the SS strategies proposed in literature, so that more informed decisions about efficient spectrum allocation and utilization can be made.

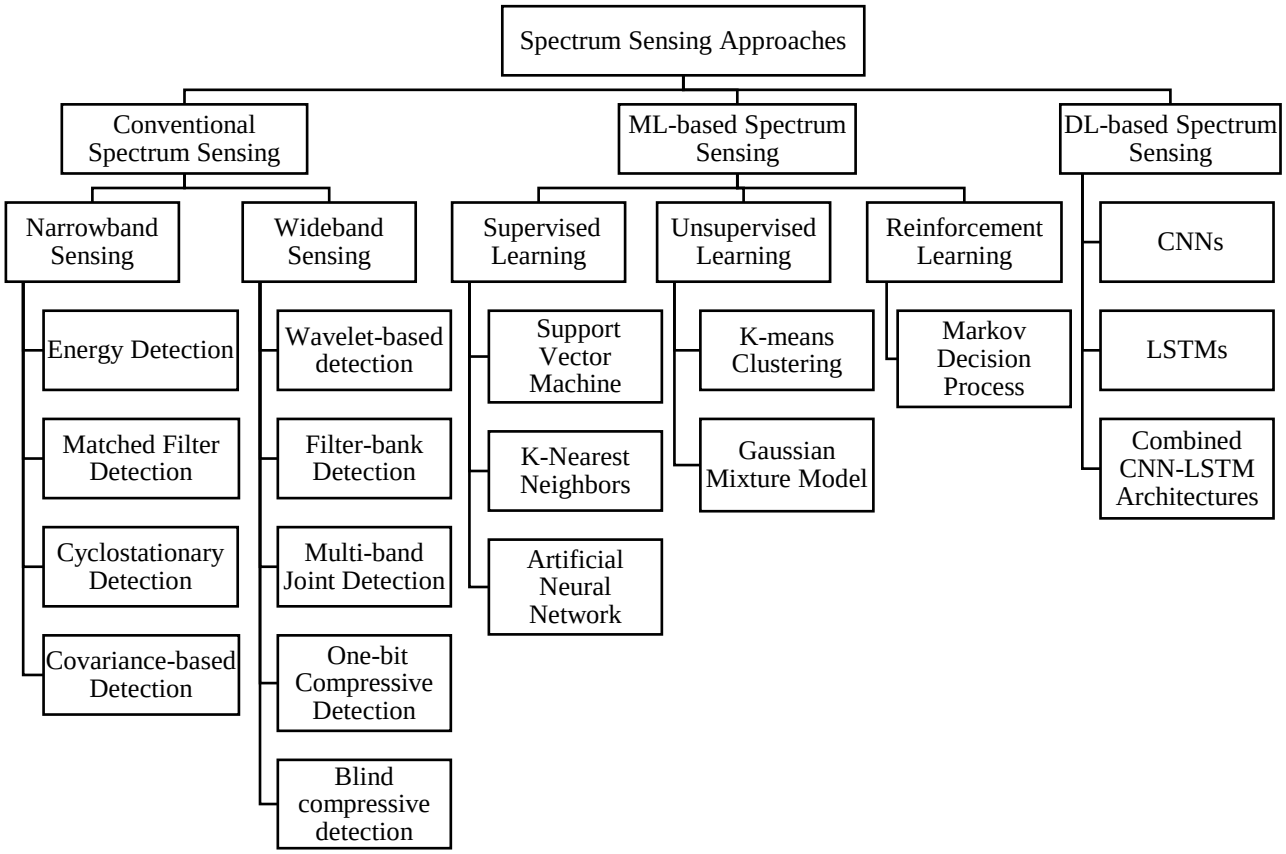


Fig. 4. Classification of SS approaches

3.2.1 Conventional Spectrum Sensing

Conventional SS techniques can be broadly classified based on bandwidth into two main categories: Narrowband Sensing and Wideband Sensing techniques. Narrowband Sensing exploits the spectral opportunities over a narrow range of frequency, while Wideband sensing techniques, on the other hand, focuses on exploiting a broader frequency range [11].

3.2.1.1 Narrowband Sensing

- **Energy Detection**

The energy detection (ED)-based SS also known as conventional energy detection (CED) is a popular SS technique due to its simplicity, reduced computational complexity and

low operating cost [10], [12]. In addition, ED technique does not require any prior knowledge about the PU's signal characteristics [10], [12], [13].

However, ED suffers from deficient performance in environments with low signal-to-noise ratio (SNR), a high false alarm rate, susceptibility to noise uncertainty, and performance degradation in the presence of multipath fading and shadowing [9], [10], [12], [13]. Additionally, it exhibits high power consumption and is unsuitable for spread spectrum methods [12]. ED's inability to distinguish noise from primary signals leads to significant uncertainty, making it impractical in dynamic and noisy spectrum environments [12], [13].

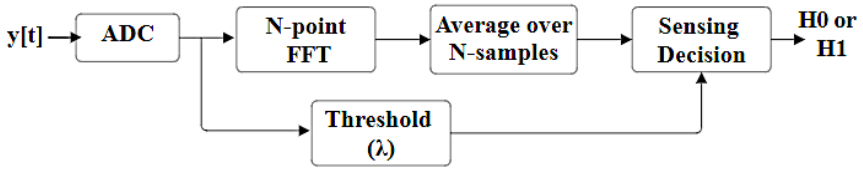


Fig. 5. Block diagram of ED

The way ED works as shown in Fig. 5 is based on the identification of the central frequency and energy of the received signal. The figure represents the process of ED scheme, where the power of samples is computed and compared with the predefined threshold (λ) to decide the status of PUs signal. If the computed energy is greater than λ, the PU signal exists; otherwise, it does not exist [12], [13].

- **Matched Filter Detection**

The matched filter detection (MFD) based SS technique known as the optimal coherent detector represents an optimal detection scheme that maximizes the output SNR in the presence of additive white gaussian noise. The MFD SS technique offers several advantages. It excels in optimal detection with minimal sample requirements, leading to reduced sensing time to achieve high processing gain [10], [12]. Particularly noteworthy is its superior performance in low SNR conditions, making it robust against interference and well-suited for environments with Gaussian noise [10].

However, MFD sensing technique comes with notable disadvantages. Firstly, it requires prior knowledge of the PU's signal, limiting its applicability in scenarios with changing or unknown PU waveforms [10], [12]. Secondly, it necessitates a dedicated detector for synchronization at each SU, which can lead to increased complexity, higher power consumption, and elevated implementation costs [12]. Perhaps the most significant drawback is that a CR would need a dedicated receiver for each PU class, making it less flexible and cost-effective in situations with multiple PUs, each with distinct signal characteristics [10].

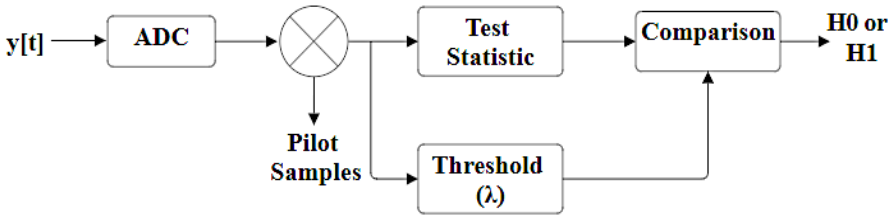


Fig. 6. Block diagram of MFD

The way matched filter technique works as shown in Fig. 6 is that the received signal y[t] is digitized using the analog-to-digital converter (ADC). The received samples are then correlated with the pilot samples to determine the decision metric, then compared with the λ to identify the status of PU signal by conducting hypothesis testing at the receiver end [12], [14]. If the output of the samples is lower than λ, then the PU is absent, otherwise the PU is present [12], [13].

- **Cyclostationary Feature Detection**

Cyclostationary feature detection (CFD) is a SS technique used in CR and wireless communication. It leverages the cyclostationary properties of signals, which are periodic statistical properties that can be observed in several types of modulated signals, including those used by PUs in the radio spectrum [10], [12]. Unlike an ED scheme, CFD scheme converts the time-domain feature into a frequency-domain feature [12]. Cyclostationary detection demonstrates a notable ability in distinguishing between noise and primary signals, contributing to robust noise resilience [10], [12], [13]. Its superior performance in low SNR

environments ensures reliable signal detection [10], [12]. Additionally, it is robust against noise uncertainty and requires fewer samples to converge, making it an efficient choice for SS in challenging and dynamic wireless environments [12].

CFD presents certain drawbacks. It necessitates more detection time, particularly as the received signal's length increases, which can impede real-time applications [12], [13]. It also involves a high computational cost and complexity, demanding substantial processing power and increasing power consumption, potentially unsuitable for energy-efficient devices [10], [12], [13]. Prior knowledge of the PU's signal is essential, limiting its use in scenarios with dynamic or unknown PU signals [10], [12]. Additionally, performance may degrade due to poor cyclic spectral density estimation, impacting the accuracy of signal detection [12]. These factors collectively pose challenges to the practical implementation of the technique.

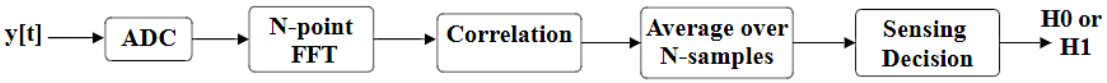


Fig. 7. Block diagram of CFD

The way matched filter technique works as shown in Fig. 7 is that the ADC digitizes the received signal y[t] and computes N-point fast Fourier transform (FFT) of these samples. These samples are correlated with each other and then averaged over the N-samples are computed. This average is then fed to the feature detector to obtain the presence and absence of PU via conducting hypothesis testing at the receiver [12].

- **Covariance-based Detection**

Covariance-based detection (CBD) techniques, as shown in Fig. 8, use sample covariance matrix of the received signal and singular value decomposition (SVD) to detect the presence of the PU signal [10], [12], [13].

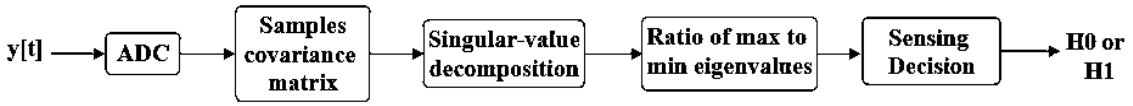


Fig. 8. Block diagram of covariance-based detection

This is determined by evaluating the structure of the covariance matrix of the received signals. The signals from the PU are correlated and can be differentiated from the noise. Using the singular value decomposition method, the singular values of this matrix can be determined. Then, the ratio between the maximum eigenvalue and minimum eigenvalue is calculated and compared with a threshold to decide between the two states, H0 and H1 [13].

CBD technique offers several distinct advantages. It eliminates the need for prior knowledge of the PU's signal and noise characteristics, enabling blind detection in dynamic environments [12], [13]. This method is robust against noise uncertainty, making it reliable in scenarios where noise conditions may vary. Unlike some other techniques, the threshold value in CBD is independent of the SNR, simplifying its implementation and making it suitable for varying signal conditions [12].

CBD exhibits several disadvantages, the most significant being its susceptibility to degraded detection performance when dealing with uncorrelated signals. This limitation can lead to errors in distinguishing PU signals from noise in scenarios with diverse or uncorrelated signal sources [12]. Moreover, the technique is characterized by high computational complexity, necessitating significant processing power, which may not be practical for resource-constrained devices or real-time applications [12], [13]. Furthermore, it requires an extended detection time, which can limit its applicability in situations that demand rapid and efficient SS [12].

3.2.1.2 Wideband Sensing

- **Wavelet-based detection**

The wavelet-based detection technique can be considered an edge detection problem that identifies singularities near the frequency channel's boundaries. The detected singularities provide essential information related to the frequency locations of sub-bands [10], [12], [13]. The way wavelet detection works is shown in Fig. 9, the wideband signal is broken down into sub-bands, each exhibiting distinct frequency variations [12], [13].

Fig. 9. Block diagram of wavelet-based detection

The wavelet transform is then employed to identify localized spectral irregularities, which convey crucial data regarding the sub-bands' frequency positions and power spectral characteristics [10], [12], [13]. Traditional sequential channel-by-channel sensing introduces considerable delays. To mitigate this, a more efficient approach is to simultaneously sense multiple bands using an RF front-end equipped with an array of narrow band-pass filters [13].

Wavelet detection offers several advantages in the context of wideband SS. Notably, it significantly reduces latency, allowing for more efficient and rapid spectrum analysis [12], [13]. The technique excels in identifying the locations of frequency bands, providing detailed information about the spectral characteristics of the sub-bands. Its adaptability to dynamic spectrum conditions makes it a versatile choice, enabling CR systems to adjust to changing frequency environments. Importantly, wavelet detection eliminates the need for the time-consuming one-by-one sensing of individual sub-bands, streamlining the SS process, and enhancing its efficiency [12].

Despite its advantages, wavelet detection also presents several disadvantages. One notable drawback is its demand for a high and often unaffordable sampling rate, which can be

impractical for resource-constrained devices and systems. Additionally, the technique is characterized by high computational complexity, resulting in increased power consumption and latency, which may not be suitable for real-time applications [12], [13]. Moreover, wavelet detection exhibits limitations in detecting signals in SNR conditions, which can impact its reliability in environments with weak or noisy signals [12]. These challenges need to be considered when choosing wavelet detection for SS applications.

- **Filter-bank Detection**

Filter bank detection is a Nyquist-based sensing technique that usually involves estimation of power spectral density (PSD) [12], [13]. As shown in Fig. 10, one way to estimate this function is to use the filter bank approach which can enable an efficient implementation of band-pass filters by using a poly-phase decomposition of the prototype filter. Each filter is the frequency-shifted copy of a low-pass filter [13]. After the output of each filter bank, narrowband sensing is performed to identify the presence and absence of PU for each band [12].

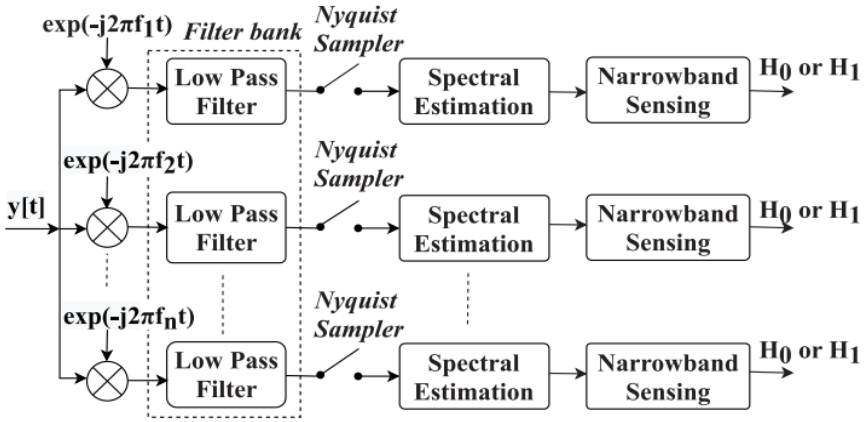


Fig. 10. Block diagram of filter-bank detection [12]

Filter bank detection offers several notable advantages in the context of wideband SS. Notably, it provides reduced latency when compared to wavelet-based sensing schemes, making it suitable for applications that require rapid spectrum analysis [12], [13]. One of its key strengths lies in its ability to sense multiple frequency bands simultaneously, which

enhances the efficiency of spectrum utilization. Additionally, filter bank detection achieves a high spectral dynamic range by using a low sampling rate, contributing to its flexibility in adapting to diverse signal types and environments. Furthermore, it exhibits high detection performance, ensuring accurate identification of PU signals and enabling CR systems to efficiently access available spectrum while avoiding interference [12].

While filter bank detection presents significant advantages, it also comes with certain drawbacks. Notably, the technique is associated with increased hardware costs due to the parallel structure of the filter bank, the implementation of this algorithm requires many RF components [10]. Filter bank detection tends to exhibit higher latency, which may be a limitation in scenarios requiring real-time spectrum analysis [12], [13]. Additionally, it involves high implementation complexity, demanding advanced signal processing techniques [9], [12], [13]. The use of filter banks can lead to elevated energy consumption, which may not be suitable for energy-efficient devices. Moreover, filter bank detection can struggle with signal detection in low SNR conditions, limiting its reliability in noisy or weak signal environments [12]. These considerations need to be weighed when considering filter bank detection for SS applications.

- **Multi-band Joint Detection**

The multi-band joint detection (MBJD) schemes can be performed using serial sensing, parallel sensing, and wideband sensing [12]. This scheme detects the presence and absence of PUs over multiple sub-bands [12], [13]. The process of MBJD is illustrated in Fig. 11, where a received wideband signal y[t] is sampled using a high-speed ADC that operates at a high Nyquist rate. The sampled signal is then fed to the serial-to-parallel converter to split into multiple segments and transform into a frequency domain using FFT [12]. The energy of the received signal for each band is computed in the frequency domain using PSD and then compared with a predefined threshold to decide the absence or presence of PU [12], [13].

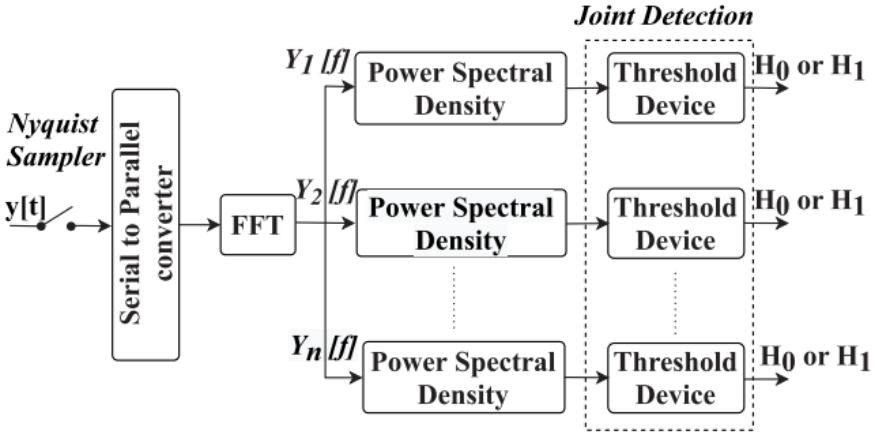


Fig. 11. Block diagram of multi-band joint detection [12]

MBJD, as a SS technique, offers several distinct advantages. It notably reduces latency compared to wavelet-based schemes, making it highly suitable for applications requiring rapid spectrum analysis [12], [13]. This approach excels in acquisition performance, allowing for the efficient identification of signal activity across multiple frequency bands simultaneously. Joint detection enhances the overall reliability of the process, ensuring that detection decisions are made collectively rather than individually for each band. Additionally, it optimizes the selection of detection thresholds for ED, improving the accuracy and effectiveness of the sensing process, which is essential for CR systems in dynamic and crowded spectrum environments [12].

MBJD drawbacks include the large latency caused by the complex optimization processes involved. The need to process and analyze multiple frequency bands simultaneously can result in substantial delays, which can be impractical for real-time applications. Additionally, the high sampling rate required for MBJD can strain computational resources and increase power consumption, making it less energy efficient [12], [13]. Furthermore, the complexity of implementing and maintaining such a system can be a substantial challenge, requiring specialized expertise and resources [9], [12], [13]. Overall, while MBJD can be a powerful tool in various applications, these disadvantages must be carefully considered when deciding whether to implement this technology.

- **One-bit Compressive Detection**

One-bit compressive sensing has been proposed recently as a promising solution to minimize multi-level quantization errors. This technique performs 1-bit quantization using a quantizer most often implemented as a comparator to a level l, which is often zero. As shown in Fig. 12, the framework of one-bit compressive sensing consists of sparse representation, one-bit quantization, and sparse recovery. One-bit compressive sensing preserves the sign information of the measurements, reducing the hardware cost [12], [13].

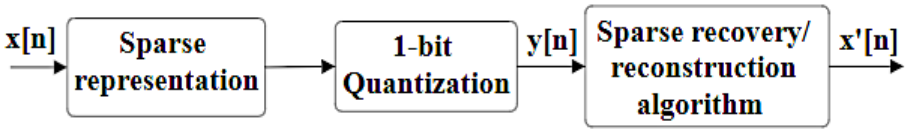


Fig. 12. Block diagram of one-bit compressive detection

One-bit compressive detection offers many advantages that make it a compelling choice for various applications. First, it excels in delivering fast and accurate results while maintaining low hardware complexity, which simplifies the implementation and integration of the detection system. This, in turn, reduces the computational cost and allows for more efficient processing of data. One-bit compressive detection also demonstrates robustness in noisy environments, ensuring reliable performance even in adverse conditions. Moreover, it offers substantial advantages in terms of low storage cost, as compressed data requires less memory capacity [12], [13]. Additionally, it contributes to energy efficiency with small power consumption and the ability to operate at a low sampling rate, making it an ideal choice for resource-constrained environments and applications where efficiency is paramount [12].

One-bit compressive detection offers several advantages but is accompanied by significant drawbacks. It is primarily suited for sparse signals that adhere to mathematical constraints like the restricted isometry properties (RIP), limiting its versatility. Additionally, it retains only the sign information, omitting amplitude data, which can be problematic in applications requiring precise signal amplitudes. Reconstruction quality is compromised,

resulting in potential fidelity loss [12]. This method is more sensitive to noise, making it less suitable for noisy environments or low-quality sensor data. Implementing one-bit compressive detection might necessitate specialized and potentially complex hardware, raising cost considerations for some applications, requiring careful evaluation of its suitability [12], [13].

- **Blind compressive detection**

Blind compressive sensing expands upon traditional compressive sensing by addressing scenarios where estimating key parameters, like sparsity levels and the number of measurements, is necessary to reduce recovery errors and improve detection performance. However, estimation of sparsity adds more complexity to the sensing process because doing so often requires more traffic exchanges between the sensing nodes and an external database or the use of one additional block to estimate that sparsity [13], [15]. Moreover, blind compressive techniques can lead to a reduction in the quality of signal reconstruction due to the absence of prior knowledge about signal characteristics [12], [13], [15]. Additionally, once implemented, the number of branches in the technique cannot be easily altered, potentially limiting its flexibility in adapting to changing requirements [12], [13].

The blind compressive sensing technique offers a range of significant advantages for various applications. Perhaps one of its most compelling features is the fact that it does not necessitate any prior knowledge of the sparsity level of the signal, making it highly versatile and adaptable to scenarios where such information may be unavailable or dynamic. Moreover, it significantly reduces computational complexity, ensuring efficient processing of data and faster detection processes. This approach excels in scenarios where prior knowledge of PU signals is lacking, enabling the blind detection of available frequency channels. In essence, blind compressive sensing empowers users to identify frequency channel locations without prior knowledge, enhancing its applicability in dynamic and complex SS tasks.

3.2.2 ML-based Spectrum Sensing

ML has received increasing attention and has been applied across various fields due to its capability to apply complex mathematical calculations for data analysis and interpretation. These techniques are more adaptive than conventional sensing techniques due to their learning ability and when adopted for cooperative spectrum sensing (CSS) can achieve a better detection performance by effectively defining optimized decision regions on the feature space [13].

ML-based sensing techniques are primarily concerned with identifying the availability of frequency channels by framing the task as a classification problem in which the classifier must decide between two states of each frequency channel: free or occupied. These classifiers use features, such as the energy statistic or probability vector, to determine the availability of RF channels [4]. The ML algorithms can be broadly classified into three classes: Supervised Learning, Unsupervised Learning, and finally Reinforcement Learning.

3.2.2.1 Supervised Learning

Supervised learning techniques involve executing a task by learning from training data provided with corresponding labels or outputs. The primary goal is to establish an objective function based on the labelled training data, which can then be used to predict the correct output for upcoming training samples [16]. Supervised ML algorithms consist of support vector machine (SVM), k-nearest neighbors (KNN), and artificial neural network (ANN).

- **Support Vector Machine**

SVM is primarily used for classification tasks with binary spectrum availability [16]. It employs a linear hyperplane to separate training feature vectors into two classes, aiming for a maximal margin with support vectors. For non-linearly separable data, different kernel methods create linear separability by mapping to a higher-dimensional space. Linear kernel SVM is well-suited for linearly separable input features.

- **K-Nearest Neighbors**

KNN is a classification algorithm that relies on majority voting from nearest neighbors, with 'K' representing the number of neighbors. It classifies test vectors based on similarity to neighbors, finding the K nearest training vectors by calculating distances. The class with the most neighbors determines the test vector's class [16].

- **Artificial Neural Networks**

Artificial neural networks (ANNs) excel in solving complex real-world problems due to their adaptability to non-linear signals [16]. ANNs consist of layers: input, hidden, and output. Hidden layers process input vectors and output layers using non-linear activation functions. ANNs learn network parameters from training data. Modern DL architectures like deep neural networks (DNNs) and convolutional neural networks (CNNs) stem from ANNs.

For example, work [17] proposed a CSS model in CR networks using supervised learning techniques such as SVMs and weighted K-nearest neighbors (KNN) algorithms. They used the energy vector estimated at SUs as a feature vector and fed it to the classifiers to detect whether the channel is available or not. They found that the SVM classifier with a linear kernel for linear inputs provides better detection probability and requires lesser training and classification delay than the weighted KNN algorithm.

Moreover, study [18] proposes a new method for CSS in CR networks using a low-dimensional probability vector as a feature vector for an SVM classifier. Making use of low dimensional probability vector instead of N-dimensional energy vector, this method achieves the same or better SS performance with a lower training duration and a shorter classification time and provides equal classification accuracy (CA).

3.2.2.2 Unsupervised Learning

Unsupervised learning exploits unlabeled training data aimed to extract hidden features in the input data. Unsupervised ML algorithms include K-means clustering and the gaussian mixture model (GMM).

- **K-means clustering**

K-means clustering groups data based on similar features. It is a popular method for classifying unlabeled data into K clusters, where "means" refers to cluster centroids [16]. Clusters minimize the sum of distances from data points to their centroid. K-means is widely used in SS with various features like Energy Vector, Eigen Values, and Probability vector. K-medoids clustering, a variant, is also applied in SS, and genetic algorithm clustering enhances sensing performance for multi-antenna SUs.

- **Gaussian Mixture Model**

The GMM is a popular clustering algorithm that employs multiple Gaussian distributions as parameter models, the number of which corresponds to the clusters [16]. The expected maximum (EM) algorithm determines the best Gaussian distribution parameters based on the training samples. In SS, the presence and absence of the PU signal are treated as distinct Gaussian distributions.

In study [18], a ML method enhances CSS in CR networks. It employs two-dimensional probability vectors, a more efficient alternative to high-dimensional energy vectors for K-means clustering and SVM-based CSS, reducing training time and classification delay. The study compared various techniques, highlighting SVM with probability vectors as the most effective approach in different CR network scenarios, improving over traditional fusion rules and achieving optimal detection accuracy. K-means clustering, though slower to train, offered shorter classification delays due to its compact probability vectors, compared to energy-based ML methods.

3.2.2.3 Reinforcement Learning (RL)

Reinforcement Learning is applied when there's partial knowledge about the environment, and an adaptive agent learns to make decisions. The agent's primary goal is to take actions that maximize rewards through interactions with the environment [16].

- **Markov Decision Process (MDP)**

The MDP is the primary RL framework for modelling the SS issue. MDP involves four components: S (finite states), A (actions), T (state transitions), and R (rewards). An agent (FC) interacts with SUs, seeks local decisions, and employs a majority rule to determine PU presence [16]. The agent receives rewards from neighboring SUs for the next state, influencing its action choice. MDP is used to make sequential decisions, with the agent learning from chosen SU's decisions, exploring unknown states, and receiving rewards. The main MDP tasks involve action selection and reward calculation.

For instance, work [19] proposes a multi-agent RL-based SS framework to discover more available frequency spectrum and achieve the desired diversity gain. The framework is based on the state-action-reward-state (SARS) algorithm and linear function approximation to reduce the dimensionality of the SS state-action space. In this multi-agent learning-based sensing policy, each SU employs a reinforcement learning algorithm to learn how to collaborate with other SUs to sense the spectrum in a way that maximizes the rewards. The rewards are designed to maximize the number of frequency bands discovered simultaneously while minimizing the energy consumption of the SUs. The framework also employs an action selection algorithm to select the number of SUs to collaborate with. This algorithm considers the current state of the environment, such as the number of available frequency bands and the energy levels of the SUs, to select the best course of action.

3.2.3 DL-based Spectrum Sensing

DL is a subset of ML that can automatically capture complex patterns and features from input Data [20]. DL's algorithms are quickly adaptable. They are a key factor to analyze and be used in uncertain radio environments that can be affected majorly by noise [4]. The use of DL in SS has attracted a considerable attention to SS due to its high-performance and speed to complete its given task compared to conventional algorithms [9]. DL has many algorithms that

can be used for SS which include CNN, long-short term memory (LSTM), a combination of both (CNN-LSTM), and many more that will be discussed below.

3.2.3.1 Convolutional Neural Networks

CNNs are widely used in the domains of computer vision and natural language processing. In a basic CNN architecture, apart from the input and output layers, it primarily comprises three essential components: convolutional layers, pooling layers, and fully connected (FC) layers [21]. Features are extracted automatically from the input samples, where CNN uses kernel or filters to extract these features. Pooling layers preform down-sampling to decrease the complexity for further layers and to prevent overfitting [22]. There is a similarity between image processing and signal covariance matrices which makes CNNs widely useful for SS [20].

For example, researchers in [23] proposed CNN-based SS algorithm with a singular convolutional layer, tailored to discern the presence of PU signal in scenarios characterized by low SNRs. By leveraging the convolutional layer, the energy and cyclostationary characteristics of the signals are meticulously extracted and standardized prior to being channeled into the CNN model. This idea is based on the understanding and findings from work [24], where PU signals have cyclostationary features, while noise is stationary, allowing for their separation based on these cyclostationary attributes. The convolutional layer is utilized to extract and standardize the energy and cyclostationary features of the signals before they are fed into the CNN model.

The proposed model was tested using binary phase-shift keying (BPSK) signals in a channel with standard noise, known as an additive white gaussian noise (AWGN) channel. A comparative evaluation was conducted between the CNN with standardized input (referred to as S-CNN), another CNN model without input standardization (referred to as N-CNN), and traditional CFD models, across a range of SNRs from -20 dB to -5 dB. The findings demonstrated that the S-CNN model had better detection performance compared to the N-CNN

model. Moreover, both S-CNN and N-CNN models exhibited superior performance over the conventional CFD algorithm at all tested SNR levels.

In another study [25], a CNN-based architecture that treats SS as a binary classification problem is introduced. The model consisted of two basic convolution layers and six residual blocks in cascade. To account for noise power uncertainty, the received signal power was normalized to enhance robustness. The dataset included simulated signal data for eight modulation techniques, such as 32QAM, 16 quadrature amplitude modulation (QAM), 8PAM, 4 Pulse amplitude modulation (PAM), quadrature phase shift keying (QPSK), 4-frequency shift keying (FSK), 2FSK, and BPSK. Both AWGN and colored noise samples were used for training the DL model.

The model achieved an accuracy of 90.55% on the test data, surpassing the performance of two conventional SS models based on frequency domain entropy and maximum-minimum eigenvalue ratio. To evaluate the model's performance with untrained data, additional test samples with modulation types 8PSK, 8FSK, and 64QAM were simulated. With a fixed P_{fa} at 0.01, the model demonstrated a high probability of correctly sensing the signals. Furthermore, when tested with real-world aircraft communications addressing and reporting system (ACARS) signals, the model, fine-tuned using a transfer learning approach, outperformed the conventional techniques. Notably, the model exhibited robustness to pink noise, unlike the conventional approaches that experienced performance degradation. This finding highlights the ability of DL to automatically extract noise characteristics from data.

Similarly, in work [26], the authors introduced a CNN-based approach called “Deep-CRNet” for opportunistic spectrum access-based SS in a communication network involving IoT and UAVs. The model consists of a total of 85 layers, including 5 convolution blocks and 2 intermediate residual-inception blocks. To generate complex signal frames, the authors artificially generate PU and noise signals using eight different modulation techniques: 64-QAM, 16-QAM, CPFSK, GFSK, BPSK, 8-PSK, QPSK, and 4PAM. Each frame is affected by

an independent Rayleigh multipath fading channel, clock offset, and AWGN. The SNR range is set between -20 dB to +25 dB, with increments of 5 dB. Additionally, an equal number of AWGN samples are generated by varying the noise power from -100 dBm to -5 dBm in steps of 5 dB. The model achieves an impressive accuracy of 99.74% in distinguishing between signal and noise frames.

To assess the performance of Deep-CRNet with over-the-air signals, signal frames from the RadioML dataset with modulation techniques of 64-QAM, 16-QAM, 8-PSK, BPSK, and QPSK are used. Deep-CRNet outperforms other pre-trained DNN architectures like GoogLeNet and MobileNetV2. Furthermore, Deep-CRNet demonstrates superior detection performance compared to other benchmark traditional and DL-based SS schemes.

3.2.3.2 Long Short-Term Memory Networks

LSTMs are extensively utilized in the realms of time series analysis and natural language processing [27]. In a fundamental LSTM architecture, besides the input and output layers, it primarily encompasses three core components: the cell state, the forget gate, and the input gate. These elements work harmoniously to allow LSTMs to capture long-term dependencies and exploit correlations within time series spectrum data, providing a robust framework for tackling sequential and temporal challenges. Which makes LTSM a useful tool and approach for SS [28].

For example, in work [28], the temporal correlation within spectrum data was analyzed using an empirical setup, employing an LSTM network. To ensure the LSTM's unbiased performance, data at extremely low SNR values was included. Two models were introduced: LSTM-based SS (LSTM-SS) and PU activity statistics-based SS (PAS-SS) [29]. LSTM-SS effectively captured the temporal correlation in the input data, while PAS-SS utilized PU activity statistics and occupancy patterns estimated from the sequence of sensing decisions.

To evaluate the performance, LSTM-SS and PAS-SS were compared with other ML techniques such as ANN, Gaussian Naïve Bayes, and Random Forest. LSTM-SS demonstrated

the highest classification accuracy, but it was observed that it required longer training and execution times. The experiments were conducted using two empirical bed setups, one with a universal software radio peripheral (USRP) and the other with a digital spectrum analyzer. Comparing the detection performance using probability of detection (P_D) versus SNR curves, LSTM-SS outperformed CNN and ANN. Additionally, when examining the classification accuracy versus SNR curves, LSTM-SS achieved the best results. However, LSTM-SS had longer training and execution times compared to the other ML/DL techniques.

3.2.3.3 CNN-LSTMs

As discussed above, CNN is a powerful algorithm that can extract spatial features from the input data, while LTSM can capture temporal variations. An innovative algorithm can be created by combining both CNN and LTSM. The combination of both will be able to extract complex features from data. The ability of extracting complex data proved to be very efficient when using this approach for SS.

In a recent study, Xie et al. [8] proposed a model that integrates both CNN and LSTM networks to enhance SS capabilities. In this architecture, CNN layers are employed first to extract energy correlation features from the covariance matrices, which are created using sensing data. These extracted features, corresponding to multiple sensing periods, are then fed into an LSTM network designed to learn the patterns of PU activity. The DNN architecture is comprised of two convolutional layers, an LSTM layer, and a dense layer. One of the notable advantages of this DL-based SS model is its resilience to the assumptions of the signal-noise model. It learns directly from sensing data, eliminating potential biases and errors associated with assumptions.

Moreover, in the experiments, PU signals are modulated using QPSK and have unit energy. Noise signals, on the other hand, are simulated based on Gaussian and Laplace distributions. The PU activity pattern is analyzed using a lognormal state sojourn time model, with state transitions depicted by a semi-Markov process [30], [31]. Real sensing data,

collected using USRP-2922, is also considered to evaluate the real state sojourn time model. The performance of the CNN-LSTM detector is compared with other detectors like Maximum eigenvalue detection (MED) [32], signal subspace eigenvalues (SSE) detector [33], arithmetic to geometric mean (AGM) detector [33], and the DL-based. Activity pattern aware SS (APASS) detector [34]. The results indicate that the proposed model delivers superior performance in various scenarios, including those with noise uncertainty. This underscores the model's robustness and effectiveness in real-world applications, marking a significant advancement in the field of SS.

3.3 Spectrum Allocation

In CR networks, the process of SS is followed by the critical task of spectrum allocation. Once spectrum holes are detected, SUs are assigned channels for data transmission. Unlike in traditional fixed spectrum assignment, CR networks necessitate Dynamic Spectrum Allocation (DSA) to ensure that SUs vacate the spectrum promptly upon the arrival of PUs [1]. The main challenge in DSA lies in allocating available channels to different SUs based on performance metrics, while minimizing interference to both PUs and other SUs.

- **Spectrum Auction and Leasing**

Among the various methods proposed for DSA in literature, spectrum auctioning has emerged as a particularly effective strategy. Benedetto et al. in [35] discuss the effectiveness of auction-based methods in CR networks, highlighting the use of different auction models like single-sided, double-sided, and combinatorial auctions. There, the primary users play the role of spectrum sellers, and the secondary users play the role of spectrum buyers. Spectrum auctioning models offer a flexible approach to spectrum management, balancing efficiency and fairness while adhering to regulatory policies. Moreover, technological challenges such as real-time spectrum monitoring and secure bidding processes are pivotal in implementing these auctions.

- **Cooperative Spectrum Sharing**

While spectrum auctioning offers a dynamic solution for spectrum allocation in CR networks, another notable method is cooperative spectrum sharing. This approach, as explored in [36], involves SUs cooperating to gain access to the spectrum in a distributed CR network architecture. Unlike auction-based methods where SUs compete for spectrum access, cooperative spectrum sharing fosters a collaborative environment where SUs share the spectrum resources. This method is particularly beneficial in scenarios where spectrum availability is limited, and cooperation can lead to more efficient use of resources. In cooperative spectrum sharing, SUs can share their spectrum access rights or collaborate in sensing and accessing the spectrum, thereby reducing the sensing burden, and improving the detection of spectrum holes [7]. This method not only enhances spectrum efficiency but also promotes fairness among SUs. However, it requires robust mechanisms for coordination and trust among users to prevent misuse of shared resources.

- **Game Theory-Based Spectrum Allocation**

Another pivotal method in the realm of DSA within CR networks is the Game Theory-Based Spectrum Allocation. This approach leverages the principles of game theory to model the interactions and strategies among network users in a competitive spectrum environment. For example, researches in [37] discuss the application of auction mechanisms in conjunction with game theory to enhance spectral efficiency in CR networks. Their work highlights how cognitive users can access unused licensed bands, optimizing the spectrum's utility. Similarly, Ni et al. [38] provide a comprehensive overview of spectrum allocation models in CR networks based on both cooperative and non-cooperative game theory, offering insights into flexible and efficient spectrum management. Their work highlights the flexibility of these models in managing spectrum resources, catering to the dynamic nature of CR networks. The study emphasizes how game theory can be applied to devise strategies that ensure efficient and equitable spectrum distribution among users, a critical aspect in maintaining network integrity

and performance. Furthermore, Garhwal and Bhattacharya in [39] investigate a static game model for spectrum sharing in CRNs. Their research focuses on the intricate balance between PUs and SUs, exploring the utility functions and Nash equilibrium concepts. This study is pivotal in understanding how game theory can be leveraged to create a harmonious coexistence between distinct types of network users, ensuring fair access to spectrum resources while maintaining the priority rights of PUs.

- **Markovian Queuing Model for DSA in Centralized Architectures**

Yet another significant method for DSA that is especially useful in centralized CR network architectures is done by applying the Markovian queuing model as proposed in [40]. This model is particularly adept at managing the allocation of bandwidth in environments where a central controller oversees the distribution of resources. In this setup, the central controller is tasked with the critical role of allocating bandwidth to the various SUs based on the real-time availability of spectrum holes. The essence of the Markovian queuing model lies in its ability to efficiently manage a network of queues, each representing a set of service requests from SUs. These requests are processed based on a first-come-first-serve (FCFS) principle, ensuring a fair and orderly allocation of resources.

This model as illustrated in Fig. 13 below, showcases the process of bandwidth allocation managed by the central controller. Queues in the figure are represented as special cases of stochastic processes, characterized by an arrival process of service requests and a waiting list for processing these requests.

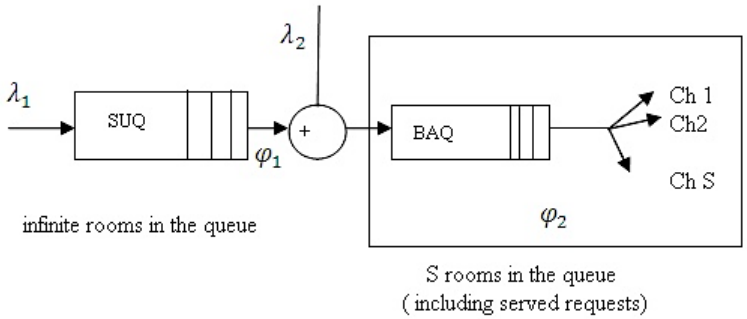


Fig. 13. Queuing model for DSA in CR network [37]

A prominent feature is the secondary users Queue (SUQ), where entries from SUs are accumulated. This queue operates on a FCFS basis, ensuring a systematic approach to handling service requests. At a given time when bandwidth needs to be allocated to the SU, the central controller, referred to as the Head, considers requests from both SUs and PUs who require access to their licensed channels. The arrival rates of both user types are summed at the Head to form the bandwidth allocation queue (BAQ), which is pivotal in allocating frequencies to both user types [37].

This model assumes that the message arrival rates for both SUQ and BAQ are Poisson and independent with mean rates λ_1 and λ_2 respectively. In addition, SUQ is assumed to have infinite capacity. Thus, SUQ is modeled as an M/M/1 queue where the M means Markovian distribution and the '1' denotes a single server. Moreover, A key aspect of this model is its use of the Erlang-B formula, which is instrumental in calculating the blocking probability of bandwidth requests (P_b). This probability is particularly important in scenarios where the demand for bandwidth from SUs is high, as it helps in understanding the likelihood of requests being denied due to all channels being occupied.

3.4 Hardware Platforms

The development and implementation of efficient hardware for SS are essential for the realization of truly dynamic spectrum access. These range from custom-designed integrated circuits to SDRs that offer flexibility and adaptability across different wireless standards. The evolution of SS hardware continues to be a vibrant area of research, with studies exploring the trade-offs between performance, power consumption, and real-time processing capabilities.

In work [41], the authors propose a low-cost and low-power consumption SS implementation based on real signals. The signals are generated by an Arduino Uno card and a 433 MHz Wireless transmitter (amplitude - shift keying ASK and FSK modulation type). The reception interface is constructed using an RTL-SDR dongle connected to MATLAB software.

The signal detection or SS is done by three methods: SVM, decision trees, and KNN. The main objective is to identify the best method for SS between the three methods. The performance evaluation of the proposed model is the P_D and the false alarm probability (P_{fa}). The comparative work has shown that the SS operation by SVM and KNN can be more accurate than decision trees and some other classical detectors. The proposed implementation addresses some of these challenges, such as cost and power consumption, by using low-cost hardware and real signals.

In a similar study [42], the authors embarked on an investigation into the deployment of SS techniques within CR systems, leveraging the capabilities of a Raspberry Pi and RTL-SDR for the detection of FM signals. The cornerstone of their approach was the implementation of an ED method, which is particularly advantageous due to its simplicity and the fact that it does not require previous information about the primary user's signal. The system transmitted FM signals using the Raspberry Pi 3 card through frequency-hopping spread spectrum (FHSS) and detected them via RTL-SDR hardware interfaced with MATLAB-Simulink. The performance of this system was gauged based on its proficiency in identifying signal presence in the context of FHSS. The results demonstrated the efficiency of the ED method, affirming its suitability for SS in CR applications, even amidst frequency variations.

In another work [43], the authors introduce DeepSense, a hardware framework that integrates real-time DL within field-programmable gate arrays (FPGAs)-based SS systems. This architecture embeds CNNs directly into the FPGA, minimizing latency in processing unprocessed I/Q samples for efficient spectrum utilization. The hybrid CPU/FPGA setup distributes processing tasks to enhance real-time capabilities vital for dynamic spectrum access. Notably, the implementation achieves very low latency, with CNN processing times as short as 0.61 milliseconds, essential for real-time spectrum decision-making. This advancement significantly boosts the responsiveness and efficiency of SS in CR networks, advancing real-time wireless communication technologies.

3.5 Conclusion

The efficient management of spectrum resources has emerged as a critical challenge in the rapidly evolving field of wireless communications. With the growing demand for bandwidth and the inherent limitations of available spectrum, CR technologies provide a promising solution for optimizing spectrum utilization. This literature review section explores CR technologies, sensing approaches, allocation methods, and hardware platforms in depth, shedding light on the evolution from traditional techniques to advanced ML and DL methodologies, each offering distinct advantages in detecting spectrum opportunities.

The transition from conventional narrowband and wideband SS methods to more sophisticated ML and DL approaches, as discussed in the literature [4], [13], [16], highlights a significant shift towards more accurate and adaptable SS capabilities. These advanced techniques, particularly DL methods like CNNs and LSTMs, have shown remarkable proficiency in detecting spectrum opportunities with higher accuracy and efficiency [8], [9], [23], [25], [28].

Moreover, when it comes to spectrum allocation, this section discusses various followed methodologies found in literature such as cooperative spectrum sharing, and spectrum auction [35], [36]. Most notably, the review discusses the Markovian queuing model for DSA in centralized CR architectures as proposed in [40], which stands out for its methodical approach in managing bandwidth allocation. This model ensures a fair and systematic distribution of spectrum resources, a critical aspect in maintaining network integrity and performance.

Building on the discussion of spectrum allocation methods CR networks, it is equally important to consider the advancements in the hardware that enables these technologies. Recent developments in hardware for SS have emphasized efficient, cost-effective solutions. Key studies, such as those in [41] and [42], demonstrate the use of accessible platforms like Arduino Uno and Raspberry Pi microcontrollers, coupled with RTL-SDR dongles. These studies

explore various ML-based SS methods, including SVMs, decision trees, and KNN, highlighting the feasibility of deploying advanced SS techniques on cost-effective hardware.

The proposed solution, drawing inspiration from these insights, aims to combine the precision of DL techniques in SS with the methodical allocation capabilities of a modified Markovian queuing model. Unlike the model discussed in [40], our approach will introduce a nuanced queuing system that will incorporate priority classes for PUs and SUs. This distinction is crucial in addressing the unique requirements and rights of PUs and SUs in CR networks, ensuring a more equitable and efficient spectrum allocation. Furthermore, this hybrid approach is not only innovative but also practical, as it addresses the real-world complexities of spectrum management in CR networks. The integration of DL for SS, as evidenced by the successes in [8], [9], combined with the efficiency of the queuing model for DSA, forms the backbone of this project. Moreover, the hardware implementation of this system marks a significant departure from previous works that predominantly utilized ML algorithms. This hardware implementation, designed to operationalize the DL model, is aimed to enhance the practical applicability and effectiveness of CR technologies in real-world scenarios.

To sum up, the proposed system is an innovative advancement in the area of wireless communications. Not only does it address the critical need for intelligent and efficient spectrum management, but it also establishes a new standard for future technological developments. By harnessing the power of DL for SS and combining it with an effective DSA model, all implemented on a robust hardware platform, this system exemplifies the potential of integrating advanced computational methods with practical hardware solutions. It sets the stage for wireless networks to become more equitable, efficient, and adaptable in order to meet the needs of a world that is becoming more and more connected.

4. System Specification

In this section of the report, we will discuss the system's functional and nonfunctional requirements, as well as the use case diagram which will showcase the use cases and how they interact with each other.

4.1 Functional Requirements

1. Spectrum Sensing

1.1. The system shall continuously monitor the RF spectrum to identify both occupied and unoccupied bands, specifically catering to IoT and mobile devices.

1.2. The system should be capable of detecting signals and activities of IoT devices operating on cellular networks.

2. Dynamic Spectrum Allocation

2.1. The system shall dynamically allocate spectrum, considering the specific communication needs and patterns of IoT devices.

2.2. The system should provide mechanisms for prioritizing spectrum allocation based on device type, usage, and requirements.

3. IoT Device Communication

3.1. The system shall facilitate efficient communication protocols for IoT devices operating on radio networks.

3.2. The system should adapt allocation strategies to optimize the performance and energy efficiency of IoT devices.

4. Interference Management

4.1. The system shall include specialized algorithms to mitigate interference issues peculiar to IoT devices on cellular networks.

5. Data Handling

5.1. The system should be capable of handling the diverse data types and communication patterns associated with IoT devices.

4.2 Use Case Diagram

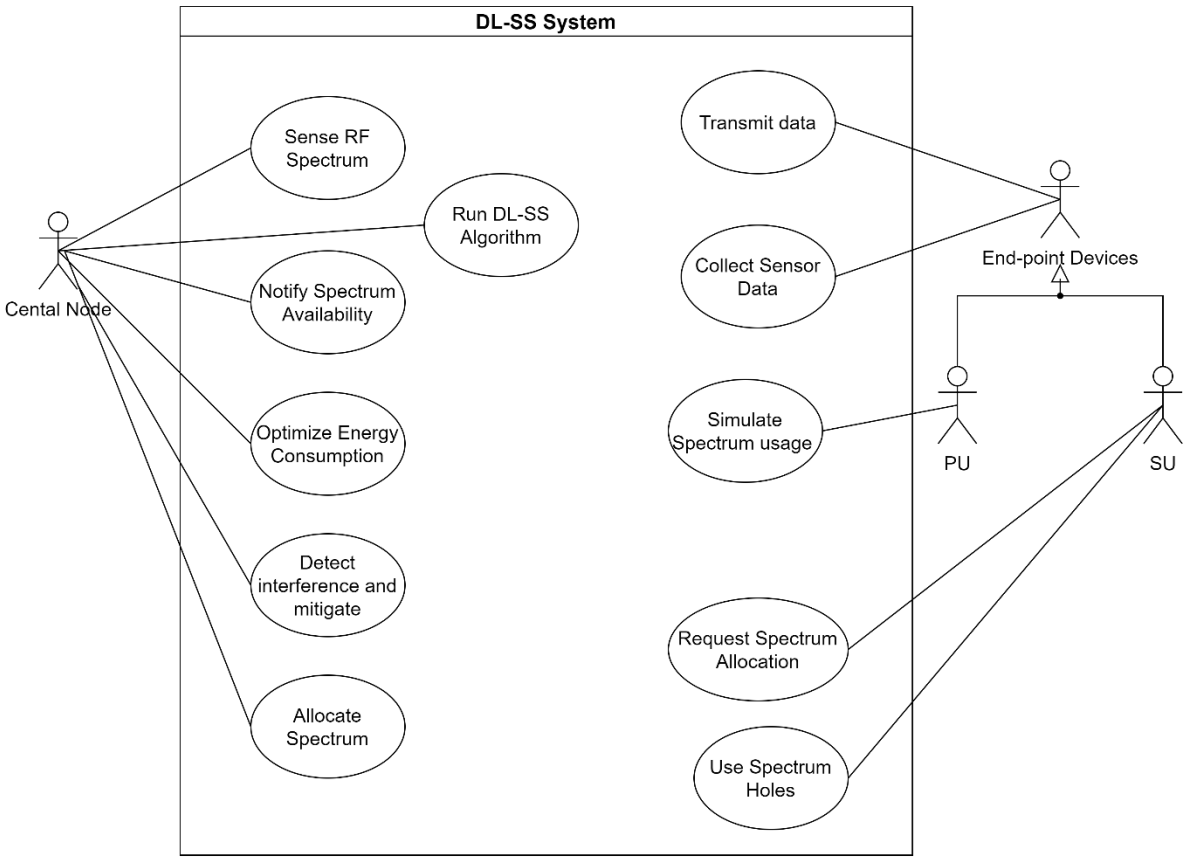


Fig. 14. Use case diagram of the proposed system

4.3 Non-Functional Requirements

1. Performance

1.1. The system should ensure rapid spectrum allocation, especially catering to the real-time data needs of IoT devices.

1.2. The system should handle the diverse and dynamic communication patterns of IoT devices efficiently.

2. Reliability

2.1. The system should be reliable in ensuring that IoT devices maintain consistent connectivity and performance.

2.2. The system should offer mechanisms to quickly recover and restore connectivity for IoT devices in case of failures.

3. Scalability

3.1. The system should be scalable to support the integration and communication needs of a growing number of IoT devices.

3.2. The system should facilitate easy integration of new IoT devices without compromising performance.

4. Adaptability

4.1. The system should adapt to the diverse and dynamic operational needs of various types of IoT devices.

4.2. The system should adapt to different types and levels of signal noise, including variations in SNR to maintain effective communication.

5. Technical Approach and Design Alternatives

5.1 Problem Statement and Proposed Solution

As previously stated, the exponential growth of mobile devices, the IoT, and other connected technologies has unleashed an unprecedented demand for spectrum resources, pushing existing 4G and 5G bands to their limits [1]. The impending spectrum scarcity poses a significant threat to communication systems, as it can result in congestion, interference, and compromised performance. Moreover, this problem persists despite the fact that many licensed spectrum bands remain underutilized [8].

To address this challenge, our project proposes the development of an intelligent spectrum management system. Central to this system is an advanced DL algorithm, specifically a CNN model, designed to enhance SS capabilities. This model will monitor and dynamically allocate available spectrum resources to both mobile and IoT devices, thereby improving the efficiency of frequency band usage. Moreover, this project draws upon existing research that has explored DL-based SS such as [8], [9] with the overarching aim of translating such pioneering research into a practical, real-world implementation. Most importantly, our project aims to encompass an algorithmic contribution to the SS field and a practical implementation of such a system.

5.2 Design of the Solution

5.2.1 Block Diagrams

Our proposed system's block diagrams are shown in the figures below. A block diagram is a representation of a system in which the main components are integrated. Our proposed SS and allocation system consist of an algorithmic contribution on the software side and a practical implementation on the hardware side that runs the SS algorithm and allocates spectrum resources to participating PUs and SUs.

As seen in Fig. 15 below, the system is based on a centralized CR network architecture, where a base station, as a central node in our case, plays a pivotal role in managing spectrum resources and coordinating the activities of the SUs. Both PUs and SUs in this system are modeled using Arduino Uno R3 microcontrollers. The PUs are allocated frequency bands using Frequency Division Multiple Access (FDMA), and when a band becomes available, it is promptly allocated and occupied by the SUs. Moreover, the central node, powered by a Raspberry Pi 4, is at the heart of this architecture. Utilizing its sensing antenna, the central node runs the SS algorithm to continuously monitor the radio environment and identify spectrum holes. These identified spectrum holes are subsequently allocated to the SUs, ensuring efficient utilization of the available spectrum. Moreover, all decisions related to spectrum allocations and reallocations are made solely by the central node. Further details about the system specifics on a block level is discussed later in this section.

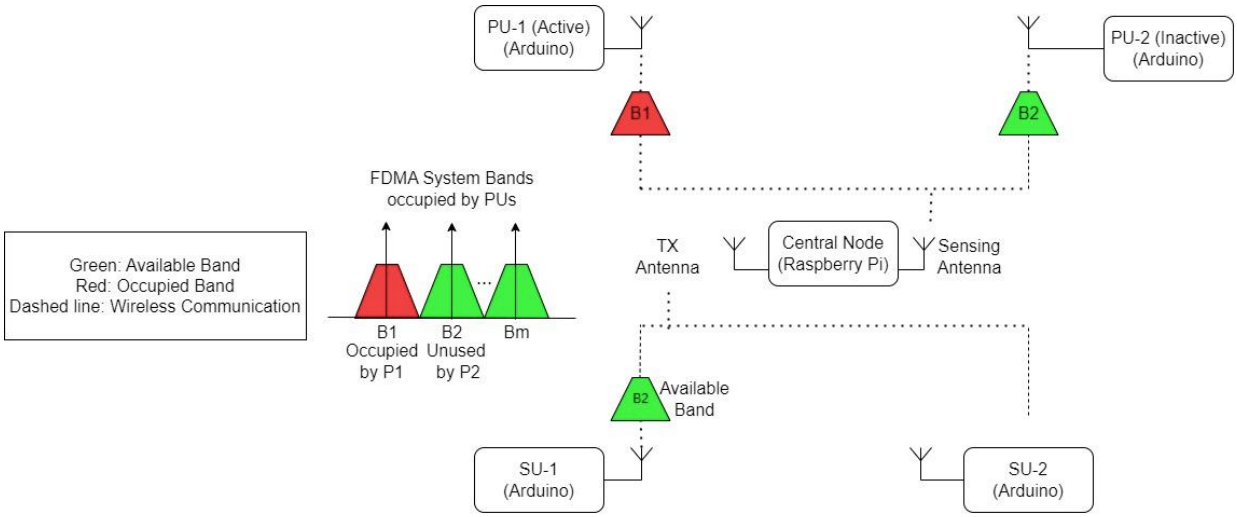


Fig. 15. Block diagram of the proposed system

Building on the high-level block diagram provided in the figure above, the subsequent figures provide better detail into the specific components within our proposed system. Figure 16 presents a detailed block diagram of the central node, showcasing the Raspberry Pi 4 and all connected devices. This highlights how the Raspberry Pi interfaces with the RTL-SDR antenna to perform continuous SS and how it communicates with SUs using the LoRa RF

transceiver to allocate spectrum resources dynamically. Additionally, Figure 17 illustrates a sensor node, either a PU or a SU, modeled using Arduino Uno R3 microcontrollers. Each node is equipped with a sensor and a LoRa RF transceiver module, enabling it to participate actively in the network by sending data and receiving spectrum allocation commands.

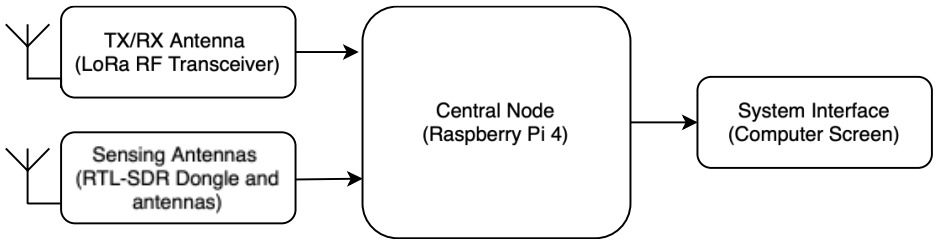


Fig. 16. Detailed block diagram of central node

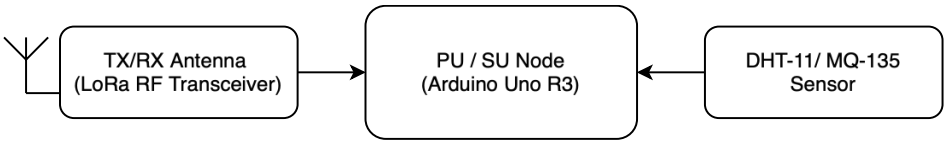


Fig. 17. Detailed block diagram of PU/SU nodes

5.2.2 Main Resources Needed on a Block Level

Raspberry Pi 4 Model B 8GB RAM (Central Node):

The Raspberry Pi acts as the central node in this system, which responsible for the SS and DSA. It uses its RTL-SDR antenna to continuously monitor the RF spectrum and identify spectrum holes, this is done through the compressed pre-trained DL model that running as part of the main program. The model takes the raw input data that is coming from the RTL-SDR, runs some preprocessing steps and then uses its trained neural network to infer weather the channel is occupied by a PU, or free, indicating a spectrum hole. On the other hand, the Raspberry Pi uses its LoRa RF transceiver module to receive transmission request from SUs and allocate channels to them dynamically based on availability of holes. The request collected in a queue and are processed on a FCFS basis.

Arduino Uno R3 Microcontrollers (PU and SU Nodes):

The Arduino Uno R3 serves as both PU and SU nodes within the system. Each Arduino is equipped with sensors that collect environmental data and a LoRa RF transceiver module, enabling it to actively participate in the network. These microcontrollers manage tasks such as transmitting collected data and receiving commands for spectrum allocation from the central node. They also handle occupying or vacating frequency bands based on these commands. This setup allows the Arduinos to execute simple control tasks efficiently and respond dynamically to the spectrum environment, ensuring effective communication and coordination within the CR network.

Adafruit RFM9x LoRa transceiver module (Attached to all nodes):

The Adafruit RFM9x LoRa transceiver module is essential for communication in the network. This module supports long-range, low-power wireless communication, enabling the nodes to send and receive data, including spectrum allocation commands. It facilitates continuous interaction between the PUs, SUs, and the central node, ensuring robust network operations. Its capacity to operate in high interference environments and its efficiency in managing power consumption are crucial for maintaining reliable and effective communications within the system.

RTL-SDR dongle and Antenna (Attached to Central Node)

The RTL-SDR is an inexpensive small and compact SDR USB dongle housing the Realtek RTL2832U integrated circuit. In our setup, it is attached to the central node, enabling the system to continuously monitor the RF spectrum. This capability is essential for real-time SS, as the RTL-SDR collects raw RF input data and performs signal processing to output I/Q data. This processed data is crucial for the system to detect available spectrum and identify spectrum holes, enabling dynamic spectrum allocation based on real-time environmental conditions.

DHT-11 and MQ-135 Sensors (Attached to PU and SU Nodes):

The DHT-11 and MQ-135 sensors are attached to the Arduino nodes to collect basic environmental data. The DHT-11 measures temperature and humidity, and the MQ-135 detects gases. These sensors help simulate real-world conditions in the network but are not key to its main functions. They simply provide extra data to test how the system handles different situations.

5.2.3 Flow-Charts

The figures below describe the behavior of the system. Fig 18 shows the steps involved in training the DL model. Fig. 19 shows the activity of the system's central node. The flowchart demonstrates the sequence of events that occur depending on what the DL model decides to do when allocating spectrum to SU, as well as the actions taken in response to primary users PU attempting to transmit data while the SU is currently occupying the frequency band of the PU. Fig. 20 shows the activities of SUs based on the central node approval. it shows when an SU begins sending data. It also demonstrates what happens when the central node interrupts while the SU is transmitting data.

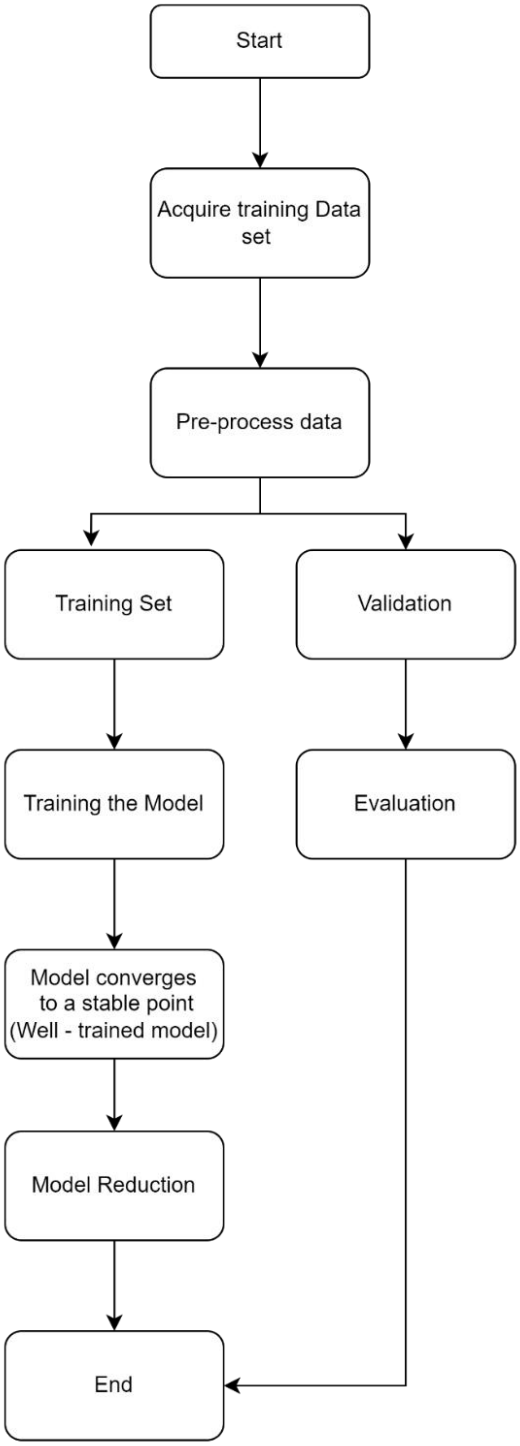


Fig. 18. Flow chart of training the DL model

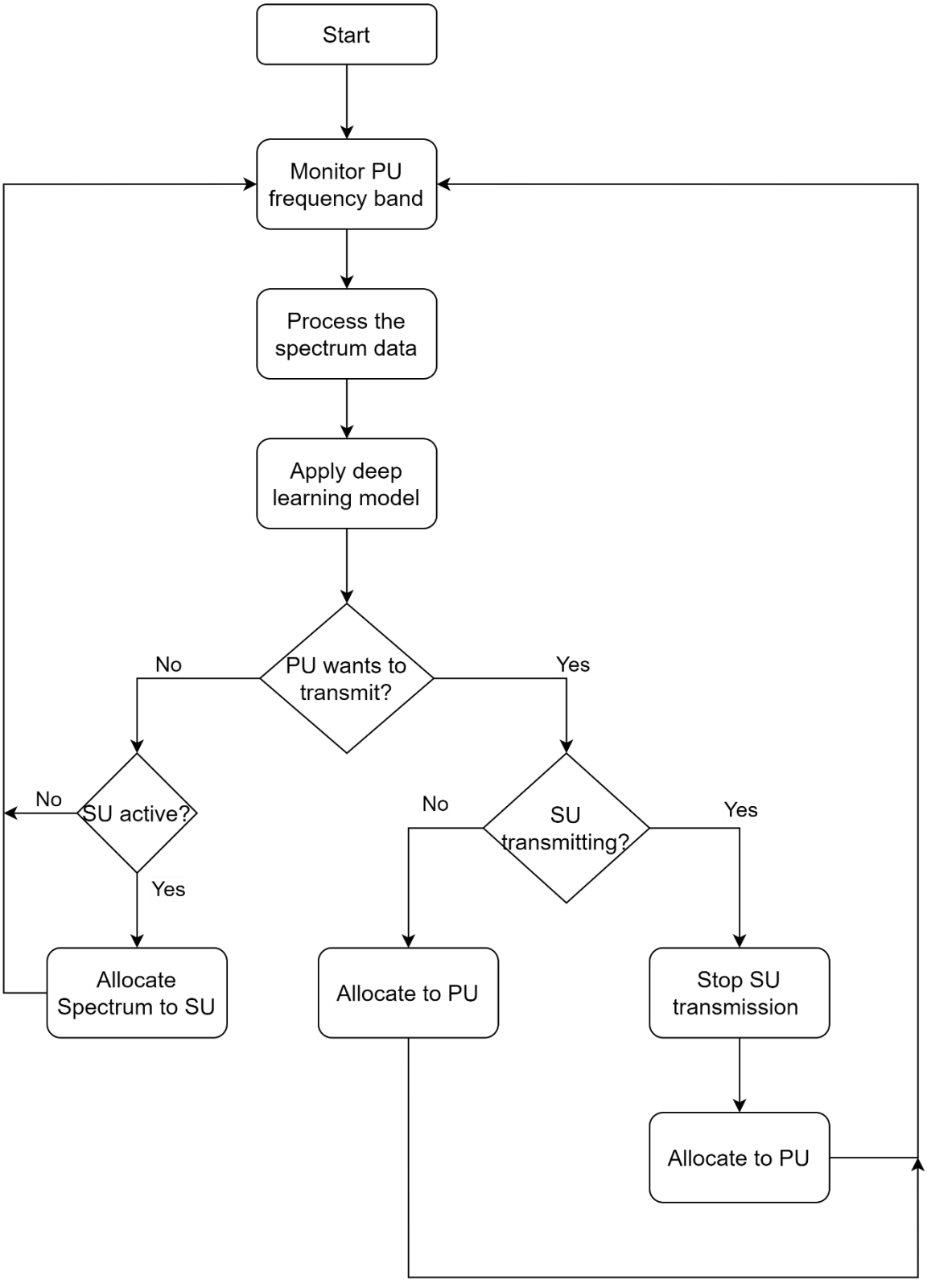


Fig. 19. Flow chart of central node activity

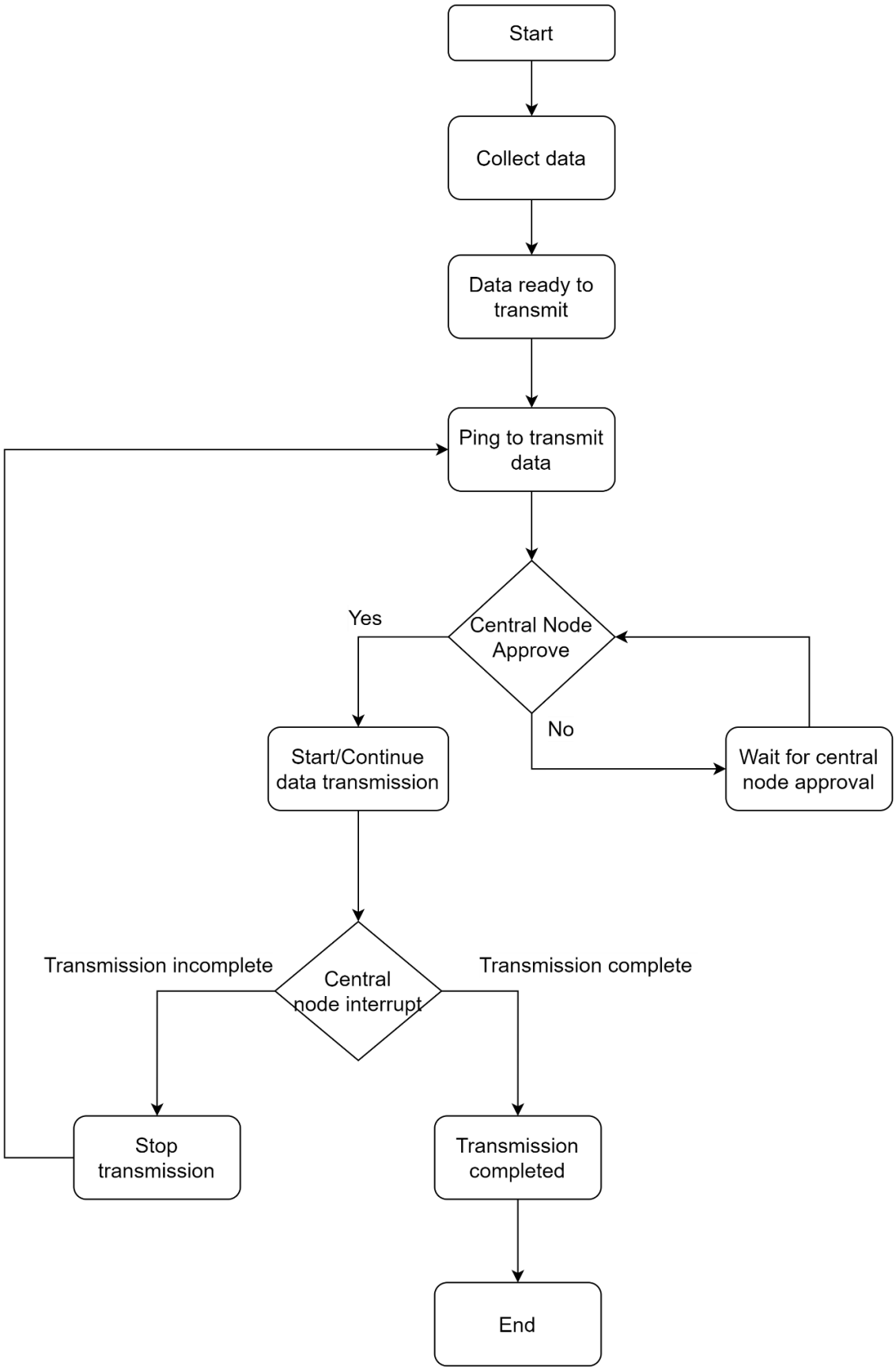


Fig. 20. Flow chart of SU activity based on central node approval

5.3 Alternative Designs

In terms of the hardware implementation, an alternative design would be to implement the distributed CR network architecture instead. In a distributed CR network, the SUs forms a network with each other without the need for a central base station. The SUs themselves make decisions about how to allocate and reallocate spectrum, which requires more complex sharing mechanisms than in centralized CR networks. This means that each SU node would need to run and implement the SS algorithm on its microcontroller, and proper communication between the SU nodes and between them and the PU nodes would need to be facilitated. However, this also means increased cost and complexity as doing so require each sensor node to house a powerful microcontroller to run the DL SS algorithm. On the other hand, in terms of the algorithm choice, while a DL-based algorithm is proposed, alternative ML approaches such as SVMs or other conventional methods could be considered for SS, each with its own trade-offs in terms of complexity and performance.

6. Project Management Plan

ID	Task Name	Status	Assigned To	Start Date	End Date	Duration
Phase I (Fall 2023)						
1	Project Initiation	Completed		Sep 04, 2023	Oct 05, 2023	32 days
1.1	Problem Identification & Idea Development	Completed	Ghanim,Hamza,Muhammad,Sultan	Sep 04, 2023	Sep 16, 2023	13 days
1.2	Problem Statement & Proposed Solution	Completed	Ghanim,Hamza,Muhammad,Sultan	Sep 17, 2023	Sep 17, 2023	0 days
1.2	Design Objectives & Limitations	Completed	Ghanim,Hamza,Muhammad,Sultan	Sep 21, 2023	Sep 26, 2023	6 days
1.2	Problem Statement, Design Objectives, and Limitations	Completed	Hamza	Sep 27, 2023	Sep 27, 2023	0 days
1.2	Project Plan	Completed	Hamza	Oct 03, 2023	Oct 05, 2023	3 days
2	Literature Review	Completed	Ghanim,Hamza,Muhammad,Sultan	Sep 28, 2023	Oct 26, 2023	29 days
3	Report Part I	Completed		Oct 09, 2023	Oct 12, 2023	4 days
3.1	Introduction	Completed	Ghanim,Hamza	Oct 09, 2023	Oct 12, 2023	4 days
3.2	Problem Statement, Design Objectives, and Limitations	Completed	Hamza	Oct 09, 2023	Oct 09, 2023	1 day
3.3	Literature Review (Writing and Organization)	Completed	Ghanim,Hamza,Muhammad,Sultan	Oct 09, 2023	Oct 12, 2023	4 days
4	Report Part II	Completed				7 days
4.1	Literature Review (Writing and Organization)	Completed	Ghanim,Hamza,Muhammad,Sultan	Oct 13, 2023	Oct 19, 2023	7 days
4.2	Requirements	Completed	Ghanim	Oct 13, 2023	Oct 19, 2023	7 days
4.3	Cost Estimate	Completed	Hamza,Ghanim	Oct 13, 2023	Oct 19, 2023	7 days
4.4	Updated Project Plan	Completed	Hamza	Oct 16, 2023	Oct 19, 2023	4 days
4.5	Project Impact	Completed	Muhammad,Sultan	Oct 17, 2023	Oct 19, 2023	3 days
5	Report Part III	Completed		Oct 20, 2023	Nov 02, 2023	14 days
5.1	System HW/SW	Completed	Ghanim,Hamza,Muhammad,Sultan	Oct 20, 2023	Nov 02, 2023	14 days
5.2	Architecture & Detailed Design	Completed	Ghanim,Hamza,Muhammad,Sultan	Oct 25, 2023	Nov 02, 2023	9 days
6	Report Part IV	Completed		Nov 03, 2023	Nov 12, 2023	10 days
6.1	Verification & Validation	Completed	Ghanim,Muhammad	Nov 03, 2023	Nov 12, 2023	10 days
6.2	Testing Plan	Completed	Sultan,Muhammad	Nov 05, 2023	Nov 07, 2023	3 days
6.3	Result Analysis & Standards	Completed	Ghanim,Hamza	Nov 08, 2023	Nov 12, 2023	5 days
7	Draft Final Report Submission (Advisor)	Completed	Hamza,Ghanim	Nov 13, 2023	Nov 13, 2023	0 days
8	Final Report Submission	Completed	Hamza	Nov 15, 2023	Nov 15, 2023	0 days
9	Presentation Preparation and Practice	Completed	Sultan,Muhammad,Hamza,Ghanim	Nov 16, 2023	Nov 21, 2023	6 days
10	Final Presentations and Demo	Completed		Nov 21, 2023	Nov 21, 2023	0 days

ID	Task Name	Status	Assigned To	Start Date	End Date	Duration
Phase II (Spring 2024)						
1	Literature Review (Part II)	Completed	Ghanim,Hamza,Muhammad,Sultan	Jan 22, 2024	Mar 21, 2024	60 days
2	Hardware Procurement	Completed	Hamza	Jan 29, 2024	Feb 14, 2024	17 days
3	Refined Problem Statement & Design Objectives	Completed	Hamza,Ghanim	Feb 13, 2024	Feb 13, 2024	0 days
4	Updated Project Management Plan & Standards	Completed	Hamza	Feb 21, 2024	Feb 23, 2024	3 days
5	Hardware Assembly & Testing	Completed	Ghanim,Hamza,Muhammad,Sultan	Feb 25, 2024	Mar 09, 2024	14 days
6	Simulations & Dataset Generation/Acquisition	Completed	Ghanim,Hamza,Muhammad,Sultan	Feb 25, 2024	Mar 09, 2024	14 days
7	Updated System SW/HW Architecture & Design	Completed	Ghanim,Hamza,Muhammad,Sultan	Mar 06, 2024	Mar 09, 2024	4 days
8	Algorithm Design & Review	Completed	Ghanim,Hamza,Muhammad,Sultan	Mar 10, 2024	Mar 30, 2024	21 days
10	System Integration & Testing	Completed	Ghanim,Hamza,Muhammad,Sultan	Mar 26, 2024	Apr 20, 2024	26 days
11	Implementation Update	Completed	Ghanim,Hamza,Muhammad,Sultan	Mar 26, 2024	Mar 30, 2024	5 days
12	Verification, Validation, & Testing Results	Completed	Ghanim,Hamza,Muhammad,Sultan	Apr 13, 2024	Apr 20, 2024	8 days
13	Final Report Writing & Reviewal	Completed	Ghanim,Hamza,Muhammad,Sultan	Apr 21, 2024	May 01, 2024	11 days
14	Final Report Draft	Completed	Hamza	Apr 28, 2024	Apr 28, 2024	0 days
15	Final Report Submission	Completed	Hamza	May 01, 2024	May 01, 2024	0 days
16	Presentation Preparation and Practice	Completed	Sultan,Muhammad,Hamza,Ghanim	May 02, 2024	May 05, 2024	4 days
17	Final Presentations and Demo	Completed	Sultan,Hamza,Muhammad,Ghanim	May 06, 2024	May 06, 2024	0 days

6.1 Gantt Chart

Name	Start Date	End Date	Duration	Resources
Project Initiation	Sep 04, 2023	Oct 05, 2023	32 days	
Problem Identification & Idea Development	Sep 04, 2023	Sep 16, 2023	13 days	Ghanim,Hamza,Muhammad,Sultan
Problem Statement & Proposed Solution	Sep 17, 2023	Sep 17, 2023	0 days	Ghanim,Hamza,Muhammad,Sultan
Design Objectives & Limitations	Sep 21, 2023	Sep 26, 2023	6 days	Ghanim,Hamza,Muhammad,Sultan
Problem Statement, Design Objectives, and Li…	Sep 27, 2023	Sep 27, 2023	0 days	Hamza
Project Plan	Oct 03, 2023	Oct 05, 2023	3 days	Hamza
Literature Review	Sep 28, 2023	Oct 26, 2023	29 days	Ghanim,Hamza,Muhammad,Sultan
Report Part I	Oct 09, 2023	Oct 12, 2023	4 days	
Introduction	Oct 09, 2023	Oct 12, 2023	4 days	Hamza
Problem Statement, Design Objectives, and Li…	Oct 09, 2023	Oct 09, 2023	1 day	Hamza
Literature Review (Writing and Organization)	Oct 09, 2023	Oct 12, 2023	4 days	Ghanim,Hamza,Muhammad,Sultan
Report Part II	Oct 13, 2023	Oct 19, 2023	7 days	
Literature Review (Writing and Organization)	Oct 13, 2023	Oct 19, 2023	7 days	Ghanim,Hamza,Muhammad,Sultan
Requirements	Oct 13, 2023	Oct 19, 2023	7 days	Ghanim
Cost Estimate	Oct 13, 2023	Oct 19, 2023	7 days	Hamza,Ghanim
Updated Project Plan	Oct 16, 2023	Oct 19, 2023	4 days	Hamza
Project Impact	Oct 17, 2023	Oct 19, 2023	3 days	Muhammad,Sultan
Report Part III	Oct 20, 2023	Nov 02, 2023	14 days	
System HW/SW	Oct 20, 2023	Nov 02, 2023	14 days	Hamza,Muhammad
System Architecture	Oct 25, 2023	Nov 02, 2023	9 days	Ghanim,Sultan
System Detailed Design	Oct 25, 2023	Nov 02, 2023	9 days	Hamza,Ghanim,Muhammad
Report Part IV	Nov 03, 2023	Nov 12, 2023	10 days	
Verification & Validation	Nov 03, 2023	Nov 12, 2023	10 days	Ghanim,Muhammad
Testing Plan	Nov 05, 2023	Nov 07, 2023	3 days	Sultan,Muhammad
Result Analysis & Standards	Nov 08, 2023	Nov 12, 2023	5 days	Ghanim,Hamza
Draft Final Report Submission (Advisor)	Nov 13, 2023	Nov 13, 2023	0 days	Hamza,Ghanim
Final Report Submission	Nov 15, 2023	Nov 15, 2023	0 days	Hamza
Presentation Preparation and Practice	Nov 16, 2023	Nov 21, 2023	6 days	Sultan,Muhammad,Hamza,Ghanim
Final Presentations and Demo	Nov 21, 2023	Nov 21, 2023	0 days	

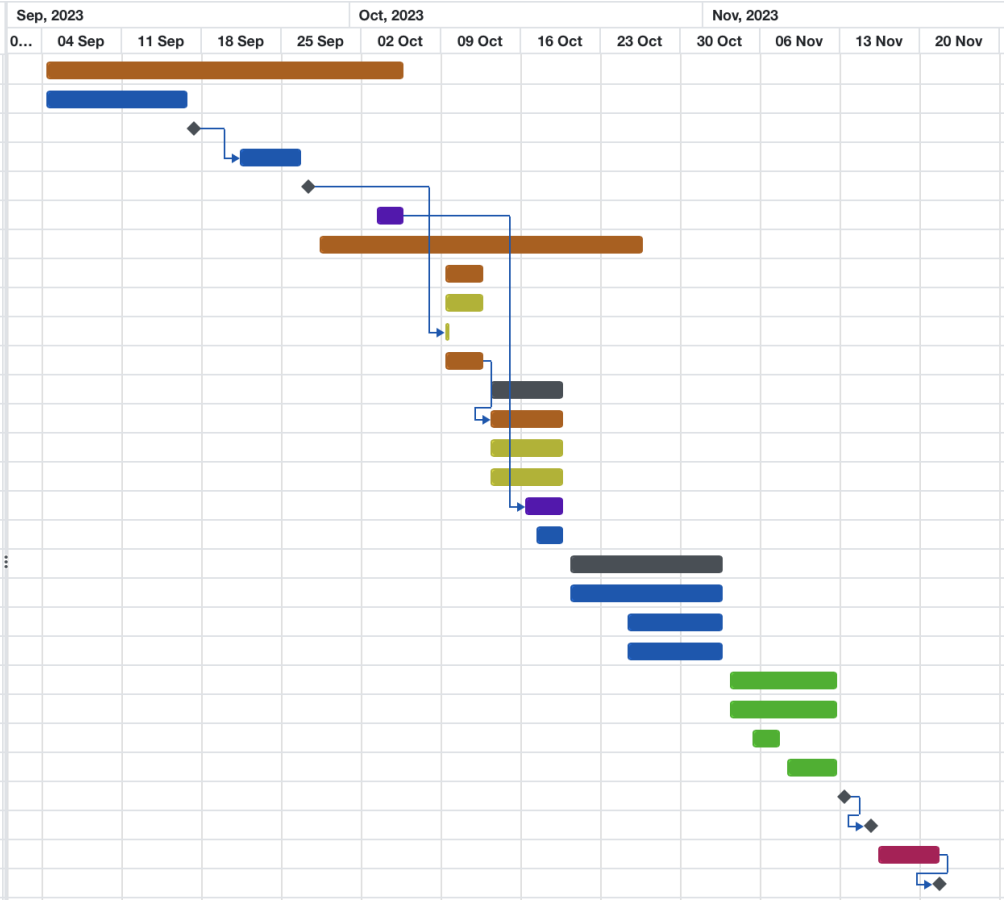
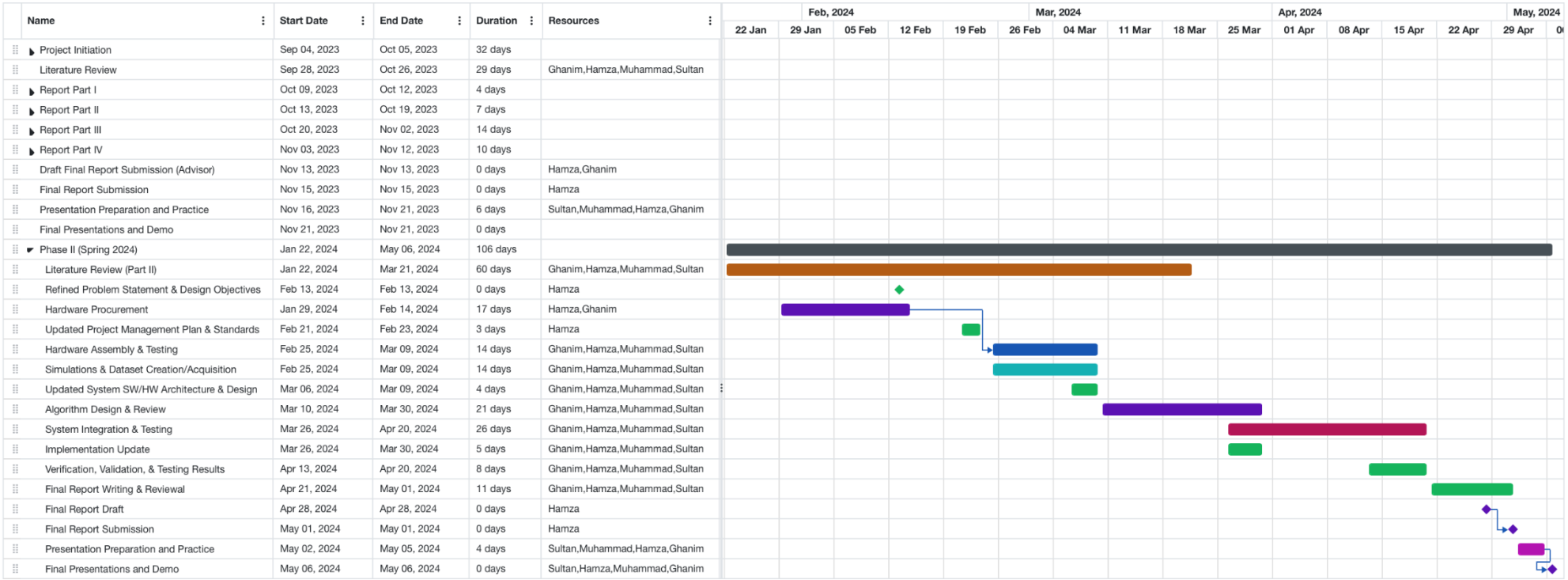


Fig. 21. Phase I of the project (Fall 2023)



Name	Start Date	End Date	Duration	Resources
Project Initiation	Sep 04, 2023	Oct 05, 2023	32 days	
Literature Review	Sep 28, 2023	Oct 26, 2023	29 days	Ghanim,Hamza,Muhammad,Sultan
Report Part I	Oct 09, 2023	Oct 12, 2023	4 days	
Report Part II	Oct 13, 2023	Oct 19, 2023	7 days	
Report Part III	Oct 20, 2023	Nov 02, 2023	14 days	
Report Part IV	Nov 03, 2023	Nov 12, 2023	10 days	
Draft Final Report Submission (Advisor)	Nov 13, 2023	Nov 13, 2023	0 days	Hamza,Ghanim
Final Report Submission	Nov 15, 2023	Nov 15, 2023	0 days	Hamza
Presentation Preparation and Practice	Nov 16, 2023	Nov 21, 2023	6 days	Sultan,Muhammad,Hamza,Ghanim
Final Presentations and Demo	Nov 21, 2023	Nov 21, 2023	0 days	
Phase II (Spring 2024)	Jan 22, 2024	May 06, 2024	106 days	
Literature Review (Part II)	Jan 22, 2024	Mar 21, 2024	60 days	Ghanim,Hamza,Muhammad,Sultan
Refined Problem Statement & Design Objectives	Feb 13, 2024	Feb 13, 2024	0 days	Hamza
Hardware Procurement	Jan 29, 2024	Feb 14, 2024	17 days	Hamza,Ghanim
Updated Project Management Plan & Standards	Feb 21, 2024	Feb 23, 2024	3 days	Hamza
Hardware Assembly & Testing	Feb 25, 2024	Mar 09, 2024	14 days	Ghanim,Hamza,Muhammad,Sultan
Simulations & Dataset Creation/Acquisition	Feb 25, 2024	Mar 09, 2024	14 days	Ghanim,Hamza,Muhammad,Sultan
Updated System SW/HW Architecture & Design	Mar 06, 2024	Mar 09, 2024	4 days	Ghanim,Hamza,Muhammad,Sultan
Algorithm Design & Review	Mar 10, 2024	Mar 30, 2024	21 days	Ghanim,Hamza,Muhammad,Sultan
System Integration & Testing	Mar 26, 2024	Apr 20, 2024	26 days	Ghanim,Hamza,Muhammad,Sultan
Implementation Update	Mar 26, 2024	Mar 30, 2024	5 days	Ghanim,Hamza,Muhammad,Sultan
Verification, Validation, & Testing Results	Apr 13, 2024	Apr 20, 2024	8 days	Ghanim,Hamza,Muhammad,Sultan
Final Report Writing & Reviewal	Apr 21, 2024	May 01, 2024	11 days	Ghanim,Hamza,Muhammad,Sultan
Final Report Draft	Apr 28, 2024	Apr 28, 2024	0 days	Hamza
Final Report Submission	May 01, 2024	May 01, 2024	0 days	Hamza
Presentation Preparation and Practice	May 02, 2024	May 05, 2024	4 days	Sultan,Muhammad,Hamza,Ghanim
Final Presentations and Demo	May 06, 2024	May 06, 2024	0 days	Sultan,Hamza,Muhammad,Ghanim

Fig. 22. Phase II of the project (Spring 2024)

6.2 Cost Breakdown

Table 1. Cost breakdown of main system components

Item Description	Unit Price (AED)	Quantity	Total Price (AED)
Raspberry Pi 4 Model B 8GB RAM	500	1	500
Arduino Uno R3	45	4	180
USB RTL-SDR Dongle and Antenna	118	2	236
Adafruit RFM9x LoRa Transceiver	70	5	350
MQ-135 Air Quality Sensor Module	23	4	92
DHT-11 Temperature Sensor Module	20	4	80
Total Cost			**1,438**

7. Implementation

This section details the implementation of our SS and allocation system, including hardware setups and software configurations. We outline the system model and problem formulation, explore the architecture of the CNN-based DL model, and describe the DSA algorithm. The experimental setup and system interface are also discussed, showcasing how they support the system's functionality and operation.

7.1 System Model / Problem Formulation

In CR networks, efficient SS is critical to detect the presence or absence of PUs and to manage the dynamic allocation of the spectrum. This process can be framed as a binary hypothesis testing problem, where the state of the PU spectrum (free or occupied) is inferred from the received signals. The received signal $x(t)$ is commonly expressed in its in-phase (I) and quadrature (Q) components, which are essential for digital signal processing. Formally, the received signal is represented as:

$$x(t) = I(t) + jQ(t) = A(t)\,cos\big(\phi(t)\big) + jA(t)\,sin(\phi(t))$$

where $A(t)$ and $\phi(t)$ denote the instantaneous amplitude and phase of the signal, respectively. The hypothesis testing for SS can be defined under two mutually exclusive hypotheses:

$$H_0: X(n) = U(n)$$

$$H_1: X(n) = h \cdot S(n) + U(n)$$

where $X(n)$ represents the n-th received sample, $U(n)$ is the noise component, $S(n)$ the PU's signal, and h the channel gain assumed to be constant over the sensing duration. Hypothesis H_0 implies $S(n)=0$, indicating the PU is not transmitting, while H_1 is the hypothesis that the PU is using the channel for transmission.

The aim of this project is to develop a classifier capable of accurately determining the occupancy status of the spectrum. Energy detection is utilized as the primary technique for SS due to its simplicity and effectiveness. It is particularly favored because it does not require prior

knowledge about the PU's signal [12]. The process involves measuring the energy of the received signal and comparing it against a threshold λ which characterizes the energy due to noise alone. Thus, the decision rule for the energy detector can be mathematically stated as:

$$D(E) = \begin{cases} H_1, & if\ E > \lambda; \\ H_0, & otherwise. \end{cases}$$

where E represents the estimated energy of the received samples. If E exceeds λ, the spectrum is inferred as occupied (H_1); otherwise, it is considered unoccupied (H_0).

7.2 CNN-Based DL Model

Building on the problem formulation described above, the solution to detecting the presence or absence of a PU in the spectrum can be effectively addressed by developing a CNN-based DL binary classifier. This classifier is designed to differentiate between signal and noise data, thereby determining the occupancy status of the channel or spectrum. It operates on the principle of binary classification, using a Softmax activation function to compute the probability of signal presence, which in this context, corresponds to the activity of a PU.

The input to this model consists of n I/Q samples captured at any instance of time, which are then structured as a matrix with n rows and 2 columns, representing the real and imaginary parts of the received signal. These samples are processed through a CNN composed of several layers, where each performs calculations that involve layer-specific weights and activation functions, culminating in the final layer where it employs a Softmax function to produce probabilities reflective of two potential states: the presence or absence of the PU. The output of the DL model is a probabilistic occupancy vector $[p_1, p_2, \ldots, p_n]$ for n channels, where $p_i = 0$ denotes an unoccupied channel suitable for SU allocation. For instance, in a scenario where there are two PU channels, the output of the DL model might look like [0,1], where "1" indicates an occupied channel. This probabilistic output is then used to guide the allocation process. The channels identified as "0" or unoccupied are considered available for

dynamic allocation to SUs. This allocation remains in effect until the channel is detected to be occupied by the PU again, at which point the SU must vacate the channel.

7.2.1 Dataset Acquisition and Preprocessing

For the development and validation of our CNN-based SS model, it was paramount to utilize a dataset that closely mirrors real-world communication scenarios. The "SDR 802.11 a/g" dataset introduced in [43] was selected for its non-synthetic nature and its emulation of actual Wi-Fi communications. This dataset consists of I/Q samples captured across four non-overlapping 5-MHz-wide channels, resulting in a total bandwidth of 20 MHz. The data was collected using five USRP N210 SDRs, configured through the GNU Radio software. Four of these SDRs functioned as Wi-Fi transmitters, each utilizing 64 sub-carriers, while one acted as the receiver with a sampling rate of 20MS/s. This configuration enabled comprehensive capture of the entire bandwidth. Data collection occurred over two distinct days, incorporating varied orientations and positions of the transmitters to introduce a range of SNR and channel effects.

The dataset comprises 32 binary (.bin) files, evenly split over the two days of data collection. Each file represents different possible outcomes of channel occupancy, which are a total of 2^4 or 16 outcomes for four channels. For example, '1101_day2.bin' indicates that on the second day, the first, second, and fourth channels were actively transmitting [43]. These files contain complex I/Q data, which were manipulated using Python to load the complex time-series signals for further analysis. To streamline the model training and evaluation process, the binary files are first converted into a more manageable HDF5 (.h5) format using a Python script, which then processes these .h5 files to prepare the training and testing datasets.

One-hot encoding is employed in this code to tackle the challenge of converting labels representing channel availability into a format usable by the DL model. The labels themselves are four-element arrays indicating the number of free channels. This format, however, proves inconvenient for the model to handle. To address this, the code implements one-hot encoding. A dictionary, is first created to serves as a reference, mapping each unique channel availability

pattern (e.g., [0, 1, 0, 1]) to a corresponding number. For each labels combination encountered (e.g., [0, 1, 0, 1]), the code locates the indices of data points in the original label dataset that match this pattern. Once these indices are found, the elements at those positions are replaced with a unique numerical value. This value represents the one-hot encoded version of the specific channel availability pattern.

7.2.2 Network Architecture

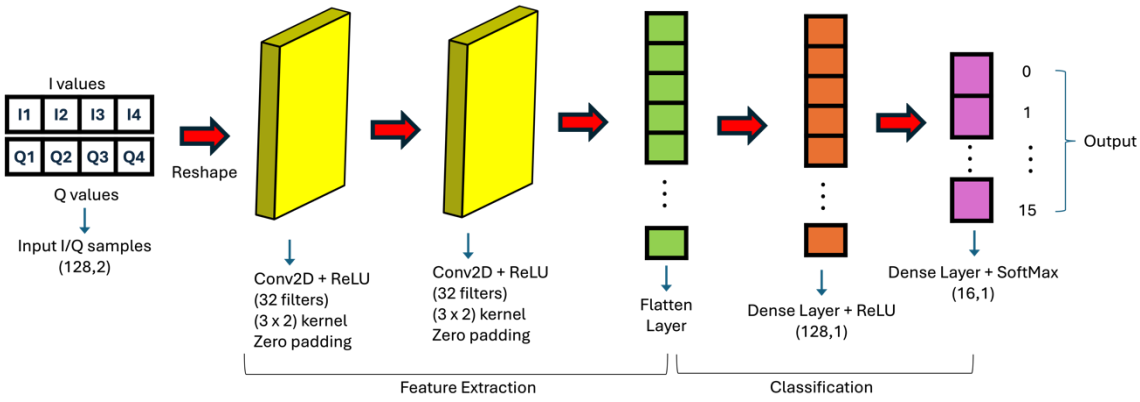


Fig. 23. Network architecture of the CNN model

The proposed CNN model, shown in the figure above, which was used for training on the SDR dataset, is divided into 5 stages which are described as follows:

- **Stage 1:** The first stage utilizes the Input Layer which defines the shape of the input data, (128, 2). This layer specifies the dimensions of the input array, indicating that there are 128 I/Q samples fed to the model at a time, each with a 2x1 feature.
- **Stage 2:** The second stage utilizes a reshape layer that converts the input data into a format appropriate for convolutional operations. It reshapes the input from (128, 2) to (128, 2, 1), adding a third dimension for channel depth. This step is crucial for compatibility with the Conv2D layers.
- **Stage 3:** The third stage utilizes convolutional layers for feature extraction from the input data. The input array goes through Zero-padding before feeding it to the first convolutional layer. This padding helps to prevent information loss from the edges of the data during convolution. First Convolutional Layer applies 32 filters with a kernel

size of (3, 2) to the input. The 'ReLU' activation function is used to introduce non-linearity and improve model performance.

- **Stage 4:** The fourth stage consists of a convolutional layer to further extract features from the outputs of Stage 1. Second Convolutional Layer uses the same number of filters (32) and kernel size of (3, 2) as the previous layer. The 'ReLU' activation is again applied for non-linearity.
- **Stage 5:** The fifth stage consists of a flatten layer to transforms the three-dimensional output of the previous convolutional layer into a single long vector suitable for feeding into fully connected layers.
- **Stage 6:** The sixth stage consists of a dense layer that takes the flattened feature vector and projects it into a higher-dimensional space with 128 neurons. The 'ReLU' activation function is used for non-linearity.
- **Stage 7:** The seventh stage consists of a dense layer that performs the classification task. It has 16 neurons, corresponding to the 16 output class probabilities. The 'Softmax' activation function is used to ensure the output values sum to 1 and can be interpreted as valid probabilities.

The architecture can be summarized in the following table:

Table 2. Summary of CNN model layer characteristics

Layer (Activation)	Output Shape
Input Layer	(128, 2)
Reshape layer	(128, 2, 1)
Conv2D (ReLU)	(128, 2, 32)
Conv2D (ReLU)	(128, 2, 32)
Flatten	(8192)
Dense (ReLU)	128
Output (Softmax)	16

7.2.3 Network Training

During training, the dataset was divided into three sections: 80% for model training, 10% for validation, and the remaining 10% for testing. The model was trained over 15 epochs using the ADAM optimization technique and the sparse_categorical_crossentropy loss function, suitable for multi-class classification problems with Softmax activation. A summary of the network training parameters is given in Table 3 below.

Table 3. Model training parameters on the acquired SDR dataset

Hyperparameters	Values
Model Architecture	Sequential
Input shape	[128, 2, 1]
Filter size	(3, 2)
Optimizer	Adam
Loss Function	Sparse Categorical Crossentropy
Metrics	Accuracy
Epochs	15
Validation Split	0.1
Activation Function	ReLU /Sigmoid

7.2.4 Model Compression

Model compression, particularly through quantization, plays a critical role in deploying ML and DL models on devices with limited computational resources such as microcontrollers. Quantization involves reducing the number of bits required to represent each weight in the model, which decreases the model's size and enhances its operational efficiency. This method significantly lowers the memory usage and power consumption, allowing the model to be executed on hardware platforms that might not support high-precision calculations. However, it is important to note that quantization can sometimes lead to a minor trade-off in model accuracy due to the reduced numerical precision.

In our project, we explored different levels of post-training quantization to compress the model after its initial training phase. This approach was chosen to strike a balance between

reducing the model's size and maintaining high accuracy. The effects of this compression were substantial, with the original model size of 12.14 MB being reduced to as low as 2.021 MB through TensorFlow Lite Float 16 quantization. Despite the significant size reduction, the model's accuracy saw minimal impact, initially at 93.73% and slightly improving to 93.77% after quantization. Below are the comparative results before and after applying different levels of quantization:

Table 4. Model performance on the acquired SDR dataset

Model	Accuracy (%)	Size (MB)
Baseline model (Float 32)	93.73	12.14
Float 16 Quantization	93.74	4.058
Float 8 Quantization	91.74	4.058
TensorFlow lite Float 16 Quantization	93.77	2.021

7.2.5 Complexity Analysis

Quantization successfully reduced the size of the DL model by 83%, decreasing the bit representation of the model’s weights from 32 bits to 16 and 8 bits. This significant reduction not only minimized the model’s memory requirements but also effectively reduced the memory complexity of the DL model, enabling efficient operation on platforms with limited computational resources, such as the Raspberry Pi.

7.3 Spectrum Allocation Algorithm

Following the SS stage, the subsequent phase involves the allocation of identified unoccupied channels, which is predicated on the output of the previously detailed CNN model. The DSA algorithm is thus designed to integrate with the occupancy vector given by DL model output. The algorithm employs a Markovian queuing model with priority classes, with PUs naturally positioned above SUs due to their licensed status. SU requests for channel access are placed in a queue *Q*, processed on a FCFS basis but subject to interruption by active PUs.

Formulating this system mathematically, the allocation function *A* for a given channel *i* can be expressed as:

$$A(p_i) = \begin{cases} PU, & if\ PU\ activity\ is\ detected; \\ SU\ from\ Q, & if\ p_i = 0\ and\ Q\ is\ not\ empty; \\ None, & if p_i = 1\ or\ Q\ is\ empty. \end{cases}$$

The function *A* serves to allocate channels to PUs first, and then to SUs when channels are free, ensuring no interference with PU operations. As channel conditions change, the algorithm quickly adjusts, allowing SUs to use the spectrum while it's not in use by any PU. This approach ensures an effective use of the available spectrum, giving priority to PUs while optimizing the chances for SUs to access unused frequencies.

7.4 Experimental Setup

An experimental testbed featuring two PU nodes and two SU nodes was established to test and validate the SS and allocation algorithms in a real-world scenario. This setup allows for testing and demonstration of the system's ability to detect PU presence and dynamically allocate spectrum holes to SUs.

7.4.1 Hardware

The hardware setup of our system is designed to simulate a real-world centralized CR network. The system operates under FDMA with two dedicated frequencies, 433 MHz, and 500 MHz, specifically assigned for PU activities. Each frequency is continuously monitored by one of the two RTL-SDR antennas connected to the central node, a Raspberry Pi 4 Model B. This configuration ensures precise and uninterrupted surveillance of the RF spectrum at both frequencies. The central node is pivotal, running the SS and DSA algorithms. It processes incoming RF data to detect spectrum holes and manages a FCFS queue that handles transmission requests from SUs. Communication between the central node, PUs, and SUs is facilitated by Adafruit RFM9x LoRa transceiver modules attached to each node, including the four Arduino Uno R3 microcontrollers. These microcontrollers serve as sensor nodes for both

PUs and SUs, simulating user activity within the network. Additionally, these sensor nodes are equipped with DHT-11 and MQ-135 sensors, which collect environmental data that simulate what might be transmitted by PUs or SUs, adding another layer of realism and utility to the testbed. The implemented system is presented in Fig. 24 below.

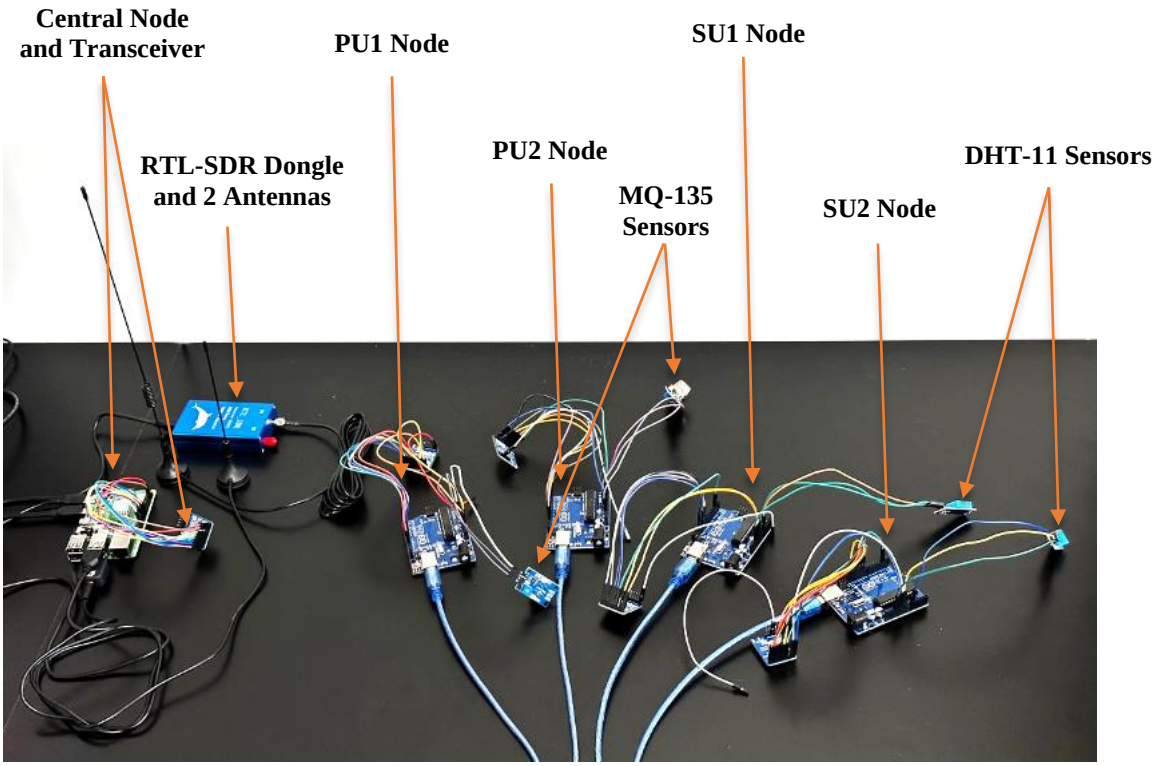


Fig. 24. Hardware experimental setup

7.4.2 Software

The software framework of the system is designed to facilitate efficient SS and DSA operations, crucial for testing the system's responsiveness and efficacy in a real-world scenario. The system utilizes a combination of C++ and Python to manage interactions between the hardware components and to execute the core algorithms effectively.

Arduinos, functioning as both PU and SU nodes, are programmed in C++. PU nodes do not actively communicate with the central node except through their transmissions at set frequencies, which the central node detects. SU nodes, on the other hand, actively collect sensor data and make transmission requests to the central node. These requests are transmitted over a

dedicated control channel at 440 MHz, facilitated by the RadioHead library, which enables reliable communication using the Adafruit LoRa RF Transceiver module. The central node receives these requests and processes them, placing them into a FIFO queue for orderly handling.

The central node's software, running on a Raspberry Pi, is developed using Python. It employs the CircuitPython library to interact with the Adafruit LoRa RF Transceiver, which handles incoming and outgoing communication with Sus over the 440 MHz control channel. Additionally, the central node utilizes the RtlSdr library for real-time spectrum monitoring through the two RTL-SDR antennas. This setup enables continuous monitoring of the RF environment, with each antenna tuned to a distinct frequency used by the PUs.

To manage these dual functionalities, communication handling and spectrum monitoring, the Raspberry Pi runs two separate threads. One thread is dedicated to operating the transceiver module for processing SU requests and responding accordingly. Whereas the other thread focuses on capturing live spectrum data using the two RTL-SDR antennas, feeding this data into the deployed DL model (as described later in section 8.4.4) to determine spectral occupancy. Outputs from this model inform the spectrum allocation algorithm, which then dynamically assigns available spectrum to SUs based on the detected availability.

For visualization and analysis, Matplotlib is used on the Raspberry Pi to plot spectrum usage on the frequencies monitored by the RTL-SDR antennas, offering visual feedback on both PU and SU activities within the network. This visualization is shown in the Fig. 25 below, where it clearly shows the instance when both PUs are actively transmitting over the 433 MHz and 500 MHz frequencies respectively.

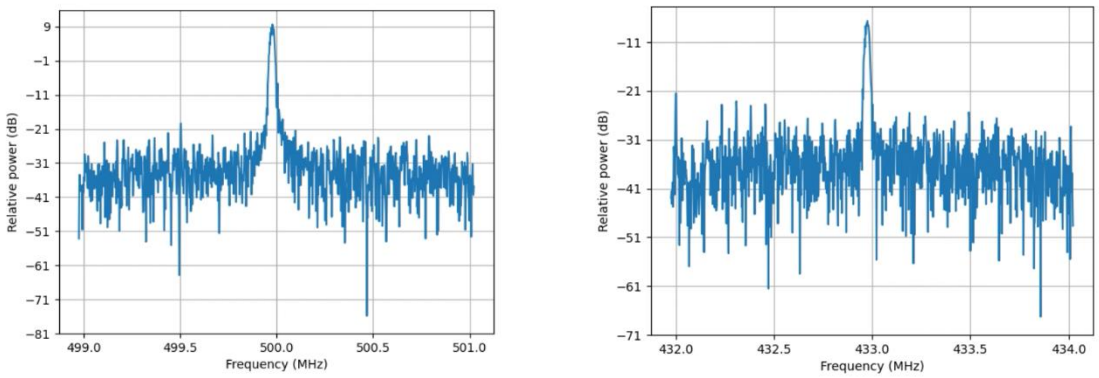


Fig. 25. Spectrum analysis on the Raspberry Pi

To facilitate structured communication and efficient management within the network, each component is assigned a unique identifier. The central node is identified as ID 1, PUs as ID 2 and 3, and SUs as ID 4 and 5. These identifiers help in precisely targeting communications and managing spectrum allocations effectively, ensuring that the system operates cohesively. This software setup ensures that the experimental testbed can replicate realistic operational scenarios, demonstrating the capabilities of the SS and DSA algorithms and providing a robust platform for their evaluation.

7.4.3 Experimental Dataset

In addition to utilizing the pre-acquired dataset for initial testing, our experimental setup was designed to generate its own dataset, reflecting simple real-world scenarios with two PUs and two SUs. This dataset was specifically created to train and subsequently deploy the DL model on the Raspberry Pi, ensuring the system's efficacy in dynamically sensing and allocating spectrum in practical conditions.

To generate this dataset, we used GNU Radio, a popular software tool widely used in communications and SDR projects. The data collection process involved two RTL-SDR source blocks tuned to distinct, non-overlapping frequencies of 433 MHz and 500 MHz, with a sampling rate set to 2.4×10^6 samples per second for both I and Q channels, resulting in a total RTL-SDR bandwidth of 8 MHz. The RF gain was set at 1×10^3 dB, while the IF and BB gains

were each set at 20 dB within the RTL-SDR Source block. Each source block was connected to a file sink, which was responsible for storing the raw I/Q data samples captured by the RTL-SDRs antennas. This setup is depicted in Fig. 26 below.

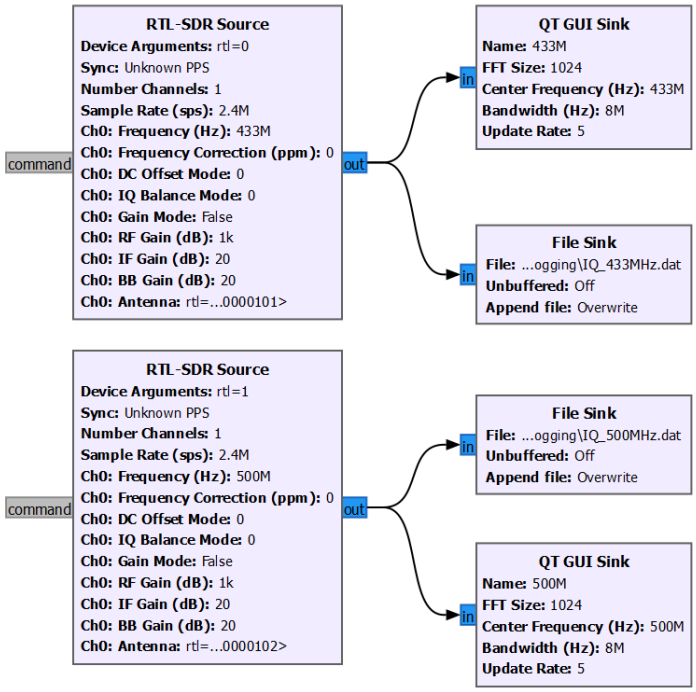


Fig. 26. GNU Radio block diagram of receiver implemented on RTL SDR

Data collection was facilitated by two Arduino devices, each programmed to act as a PU transmitting messages at one of the specified frequencies. We designed the experiment to cover four different scenarios: both frequencies unoccupied, only 433 MHz occupied, only 500 MHz occupied, and both frequencies occupied. For each scenario, we conducted a 30-second run, which proved sufficient to gather a substantial dataset of I/Q samples.

To determine the occupancy status of each frequency band, we employed an energy detection method. This involved setting a specific threshold level to distinguish between occupied and unoccupied states. We established this threshold by analyzing the active signals to determine the decibel level reached during transmission. This method enabled us to accurately assess the status of each frequency band under different experimental conditions. Screenshots of the spectrum of the received signal, in the absence and presence of signal from PU, are shown in Fig. 27 and Fig. 28, respectively.

Fig. 27. Spectrum of the received signal in the absence of signal from PU

Fig. 28. Spectrum of the received signal in the presence of signal from PU

Following the data collection process detailed above, the dataset comprised 300,000 I/Q samples representing various PU spectrum occupancy scenarios. To prepare this dataset for training the CNN model, the raw data captured by GNU Radio was stored in binary files. Then, a Python script labelled the data, assigning binary values (1 or 0) to indicate the presence or absence of PU signals. The labelled data was then compressed into a npz file format. Each input sample within the npz file consisted of two columns, representing the real and imaginary parts of the I/Q samples, with dimensions (300,000, 1024, 2).

7.4.4 Model Training and Deployment

Following the collection of the experimental dataset, the next steps entailed dividing the dataset into three sections: 80% for model training, 10% for validation, and the remaining 10% for testing. The CNN model described in section 7.2.1 above was trained using the preprocessed dataset. The model was trained over 15 epochs using the ADAM optimization technique and the sparse_categorical_crossentropy loss function, suitable for multi-class classification problems with Softmax activation. A summary of the network training parameters is given in the table below:

Table 5: Model training parameters on the generated dataset

Hyperparameters	Values
Model Architecture	Sequential
Input shape	[1024, 2, 1]
Filter size	(3, 2)
Optimizer	Adam
Loss Function	Sparse Categorical Crossentropy
Metrics	Accuracy
Epochs	15
Validation Split	0.1
Activation Function	ReLU /Sigmoid

After training, the model was quantized using TensorFlow Lite into Float 16 to reduce its size while maintaining accuracy. Originally sized at 48.12 Megabytes, the quantization process reduced the model's size to 8.02 Megabytes. This effectively preserved the model’s predictive capabilities while significantly decreasing its size.

The final model accurately distinguishes between occupied and unoccupied spectral states based on the energy levels derived from the I/Q samples. For practical deployment on the Raspberry Pi, the model was converted into TensorFlow Lite format, incorporating the described quantization to minimize both the model size and computational requirements. The quantized model was then integrated into the main program running on the Raspberry Pi. This setup allows continuous analysis of the RF spectrum using the RTL-SDR, execution of the SS

algorithm, and dynamic allocation of spectrum holes to SUs based on the model's predictions. The table below summarizes the model's size and accuracy before and after quantization:

Table 6. Model performance on the generated dataset

Model	Accuracy %	Size (MB)
Baseline model (Float 32)	96.43	48.12
Float 16 Quantization	96.43	16.05
Float 8 Quantization	96.40	16.05
TensorFlow lite Float 16 Quantization	96.18	8.02

7.4.5 System Interface

A screen connected to the Raspberry Pi serves as the system's interface, providing real-time insights into the spectrum's status. It displays occupied and available frequency bands, along with ongoing SU transmissions and PU activities. During SS processes, the interface shows which bands are actively used by PUs and which are idle. After SS, the interface updates to show the newly allocated bands to SUs, providing a clear view of changes in spectrum usage. A screenshot illustrating the system interface, showing PU and SU activity, is presented in the figure below:

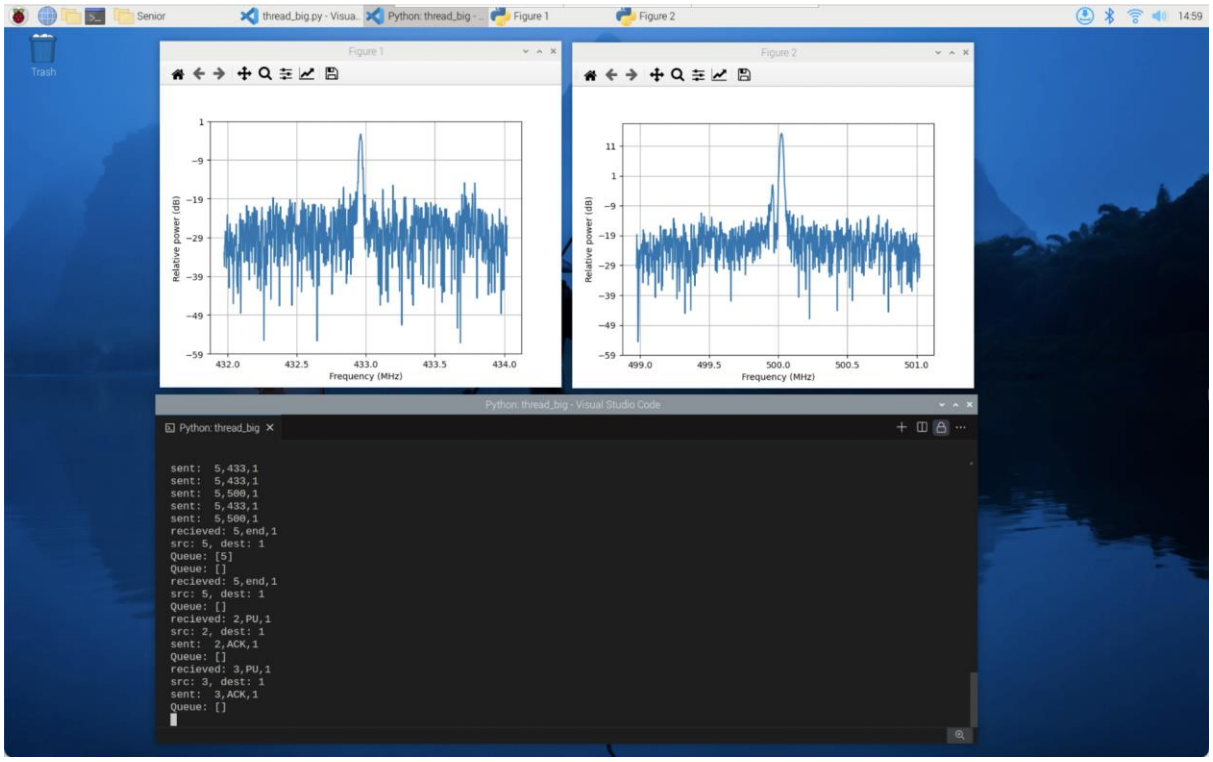


Fig. 29. System interface showing PU and SU activity

7.5 Integration of System Components

The integration of hardware and software is essential for the effective operation of our SS and allocation system. The core of this integration is the Raspberry Pi 4, which serves as the central node. This node runs a compressed CNN model using TensorFlow Lite, which is designed to process real-time RF data captured from RTL-SDR antennas. This setup enables the central node to monitor specific frequency bands, 433 MHz, and 500 MHz, and make informed decisions about spectrum allocation based on the signals received.

In a specific operational scenario illustrated in the figure below, the central node handles transmission requests from SUs via a dedicated 440 MHz channel. For instance, when SU-1 requests a transmission, the request is placed first in a FIFO queue. Meanwhile, the RTL-SDR captures I/Q samples, and the CNN-based SS model analyzes these, outputting [1,0] to indicate that 433 MHz is occupied while 500 MHz is available. Consequently, the central node allocates the free 500 MHz band to SU-1, effectively managing the spectrum resources.

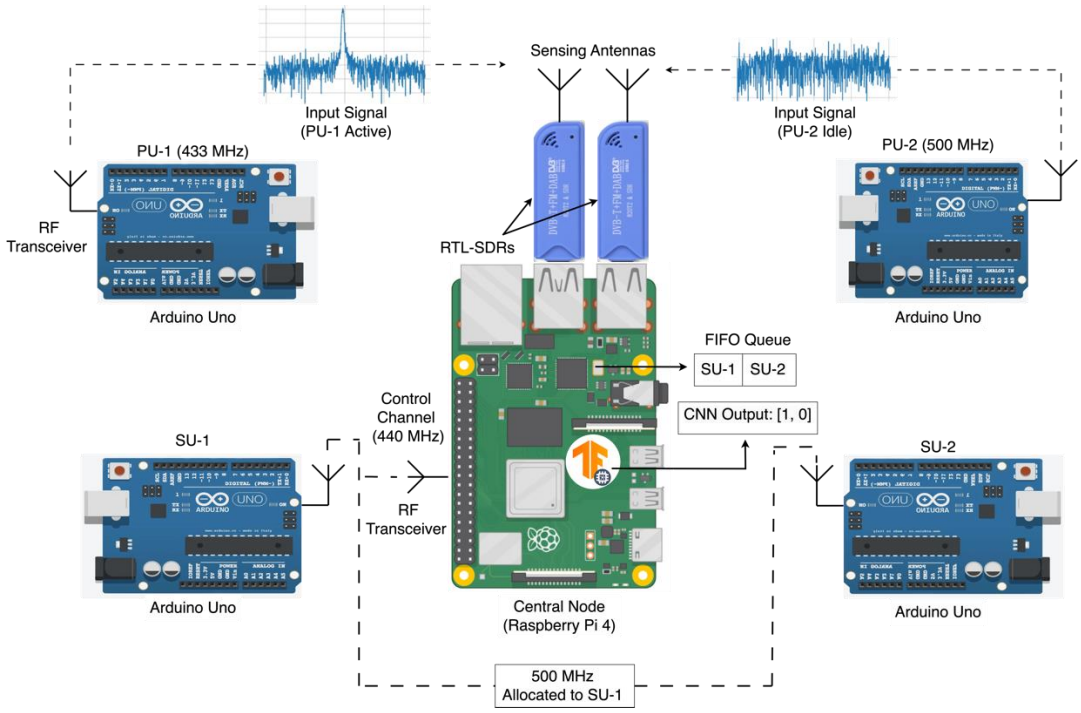


Fig. 30. Detailed system integration and operational scenario

8. Verification, Validation and Testing Results Analysis

This section details the results of our verification, validation, and testing processes, which confirm the effectiveness and reliability of our SS and allocation system. These evaluations are crucial for demonstrating the system's capability to accurately detect spectrum occupancy and effectively manage spectrum allocation dynamically.

8.1 System Testing Approach

The system testing approach adopted for evaluating the system is primarily based on black-box testing. This method allows for comprehensive assessment without the need to examine the internal workings of the code. This method focuses on the outputs in response to given inputs and actions, testing the system's individual components and their interactions holistically. This approach is particularly suitable for our SS and allocation project, as it evaluates the functionality of each system component both independently and in combination.

8.1.1 DL Model Training, Validation, and Testing

The DL model was trained and validated using both an acquired dataset and a dataset collected from our experimental setup. This dual-dataset approach helped ensure that the model was robust and reliable, capable of handling a variety of real-world scenarios.

The initial training of the model was conducted on the acquired SDR dataset using 287,971 data samples over 15 epochs. During this phase, the model demonstrated excellent performance, achieving a high accuracy of 98.54% with a correspondingly low loss of 0.0473. To visually represent the model's training progress and its validation performance, Fig. 31 illustrates the accuracy versus epochs, while Fig. 32 details the loss versus epochs for both the training and validation phases.

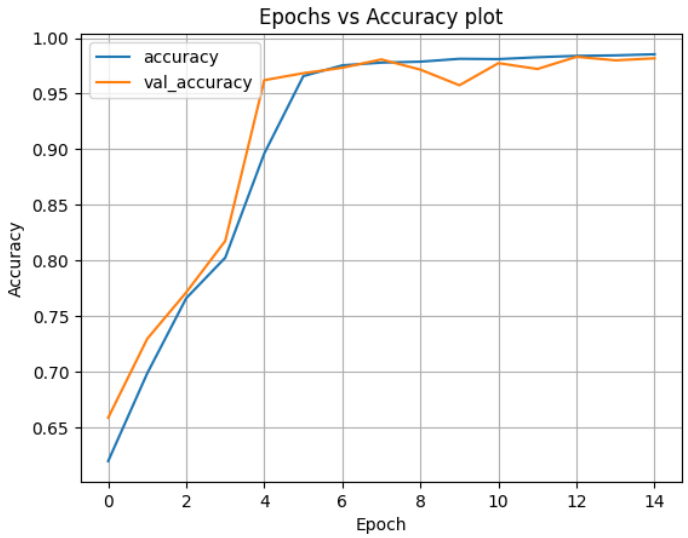


Fig. 31. Accuracy plot for the CNN model on the acquired SDR dataset

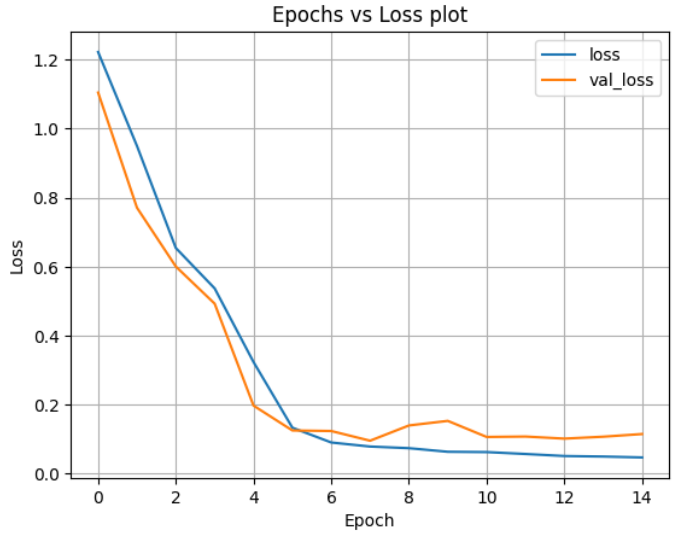


Fig. 32. Loss plot for the CNN model on the acquired SDR dataset

Following the training, the model was subjected to testing using a separate set of 31,997 data samples. The testing results were also impressive, with the model attaining an accuracy of 96.51% and a loss of 0.1404, confirming its effectiveness in handling unseen data and its potential for real-world applications. To provide further insights into the model's performance during the validation phase, a confusion matrix was created. This matrix, shown in Fig. 33, offers detailed insights into the accuracy of the model's predictions across different classes.

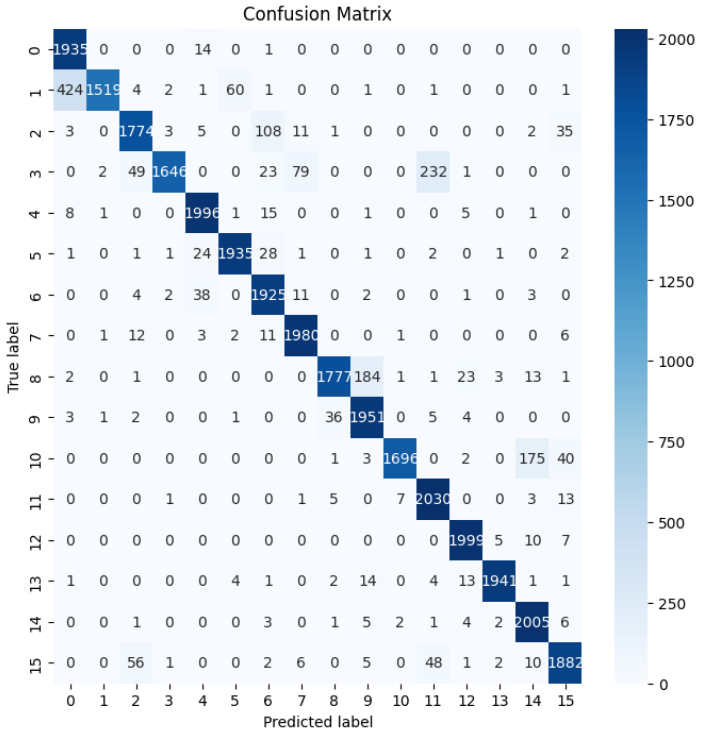


Fig. 33. Confusion matrix for acquired SDR dataset

Next, the model was further evaluated using data collected from our experimental setup. This step was crucial for confirming the model's practical applicability and its ability to function effectively within the parameters of an operational CR network. The model was initially trained on the collected experimental dataset, using 240,000 data samples across 15 epochs. Throughout this stage, the model displayed exceptional performance, reaching an accuracy of 98.98% with a low loss of 0.0190. Fig. 34 and Fig. 35 visually depict the model's training and validation performance, with accuracy and loss plotted against epochs for both the training and validation phases.

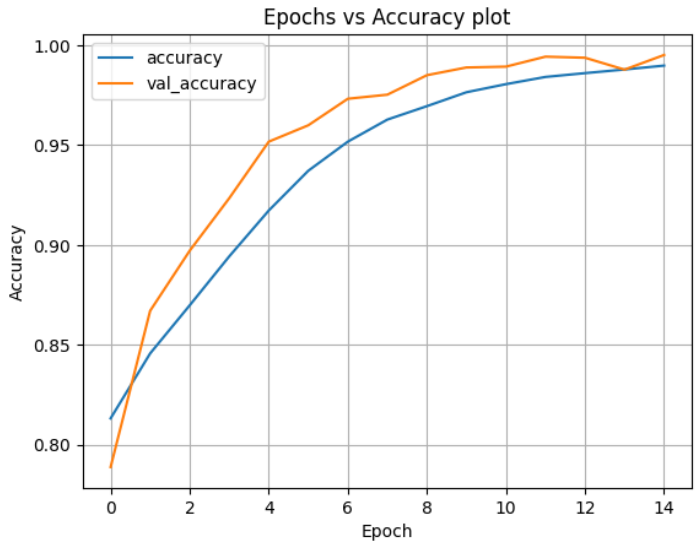


Fig. 34. Accuracy plot for the CNN model on the collected dataset

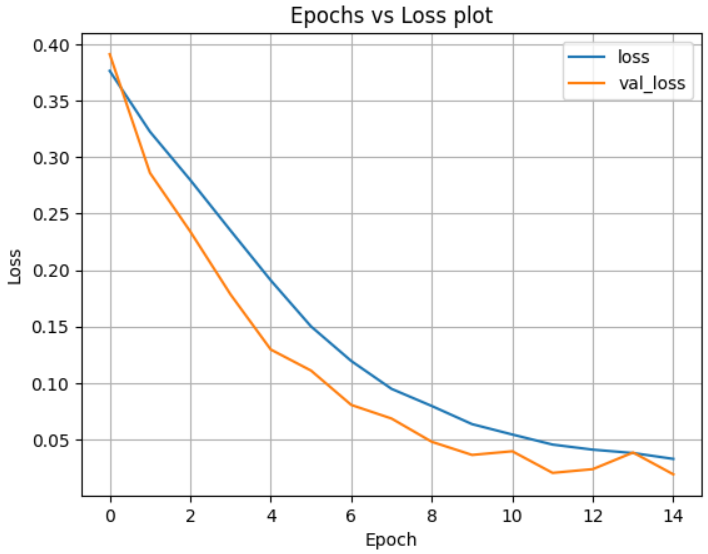


Fig. 35. Loss plot for the CNN model on the collected dataset

Following the training phase, the model was tested on a distinct set of 60,000 data samples. The testing results were equally impressive, with the model achieving an accuracy of 96.43% and a loss of 0.3057. These findings confirm its ability to handle new data effectively and its potential for practical application in real-world scenarios. A confusion matrix was created to gain a better understanding of the model's performance during validation. Fig. 36 depicts a matrix that provides comprehensive insights into the accuracy of the model's predictions across various classes.

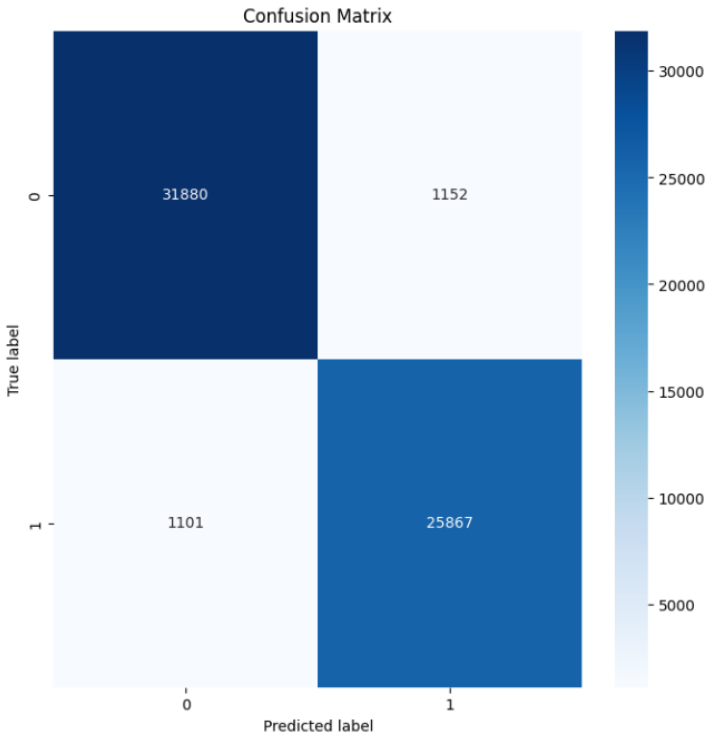


Fig. 36. Confusion matrix for collected dataset

8.1.2 System Hardware and Integration Testing

Integration testing of the system hardware was conducted to ensure that all components functioned seamlessly together, meeting the operational demands of the project. This phase focused on verifying the interactions between the Raspberry Pi central node, Arduino Uno nodes, RTL-SDR dongles and antennas, and LoRa RF transceiver modules.

The testing process began with a detailed verification of the correct wiring and configuration of each hardware component. Individual tests were performed on each component to confirm operation within specified parameters. Following these tests, the integration of components was evaluated, particularly the communication effectiveness between the central node and the Arduino microcontrollers via the LoRa transceivers. The functionality of the RTL-SDR setup was also examined to ensure it could accurately capture and relay RF spectrum data to the Raspberry Pi for processing.

Additionally, the capability of the RTL-SDR setup to continuously monitor the spectrum was assessed. It was important to ensure that the RTL-SDR modules could reliably capture

new I/Q samples and feed them into the model for processing. This setup allows the Raspberry Pi to use these inputs for real-time spectrum analysis and subsequent decision-making regarding spectrum allocation. The effectiveness of this data flow was tested by observing the system's ability to dynamically update and visualize these data on the system interface under various operational scenarios.

8.1.3 Test Case Derivation

Test cases were systematically derived to validate the functionality of our SS and DSA system under various operational scenarios. These scenarios focused on testing the system's performance when spectrum bands are occupied by PUs and when they are unoccupied. This structured approach ensures that the system accurately detects spectrum holes and effectively manages spectrum allocation, providing comprehensive validation of its operational capabilities in real-world conditions.

Occupied Spectrum Scenarios:

1. **Single PU Transmission (433 MHz)**:
    - **Procedure:** A simulation of a PU actively transmitting on the 433 MHz band.
    - **Outcome:** A peak was observed in the 433 MHz waveform due to a PU transmitting on that channel. The system successfully detected the PU transmission, the DL model output was [1, 0], indicating the 433 MHz channel is occupied and the 500 MHz channel is free.
2. **Single PU Transmission (500 MHz):**
    - **Procedure:** Simulate a PU actively transmitting on the 500 MHz band.
    - **Outcome:** A peak was observed in the 500 MHz waveform due to the PU transmission. The system successfully detected PU activity, and the DL model output was [0, 1], indicating the 433 MHz channel is free and the 500 MHz channel is occupied.

3. **Simultaneous PU Transmission on Both Bands (433 MHz and 500 MHz):**
    - **Procedure:** Both PUs transmit simultaneously on 433 MHz and 500 MHz.
    - **Outcome:** Peaks are present in both the 433 MHz and 500 MHz waveforms as both PUs transmit simultaneously. The system accurately recognized both bands as occupied, and the DL model output was [1, 1], effectively preventing any SU allocations.
4. **Simultaneous PU Transmission with a Single SU Request:**
    - **Procedure:** Both PUs transmit simultaneously on 433 MHz and 500 MHz, while an SU sends a transmission request.
    - **Outcome:** Peaks were observed in both the 433 MHz and 500 MHz waveforms due to simultaneous PU transmissions. The DL model output was [1, 1] indicated that both channels were occupied. With no channels available, the central node could not allocate a channel to the SU, and the SU was placed in a waiting FIFO queue until a channel became free.
5. **Simultaneous PU Transmission with Multiple Secondary User Requests:**
    - **Procedure:** Both PUs are transmitting simultaneously on 433 MHz and 500 MHz, while multiple SUs request transmission.
    - **Outcome:** As with the single SU request scenario, peaks in both waveforms indicated active transmissions on both channels. The DL model confirmed both channels as occupied [1, 1]. Initially, no channels were available for the SUs. Once a PU finished transmission and a channel was freed, the central node allocated the channel to the first SU in the FIFO queue.

Unoccupied Spectrum Scenarios:

1. **No PU Transmission with a Single SU Request**:
    - **Procedure:** Ensure no PUs are transmitting, and an SU requests a channel.
    - **Outcome:** No activity was detected on either channel, indicating no PU presence. The DL model output was [0, 0] showed both channels as unoccupied. The central node then allocated the 433 MHz channel to the requesting SU, efficiently utilizing the spectrum when PUs are inactive.
2. **No PU Transmission with Multiple Secondary User Requests**:
    - **Procedure:** No PUs are active and two SUs request channels.
    - **Outcome:** Both channels were confirmed as free by the absence of PU activity in the waveforms. The DL model output was [0, 0] allowed the central node to allocate the 433 MHz channel to the first SU and the 500 MHz to the second SU in the queue, enabling simultaneous transmissions.
3. **Single PU Transmission (433 MHz) with a SU Request:**
    - **Procedure:** A PU transmits on the 433 MHz channel while an SU simultaneously requests to transmit.
    - **Outcome:** A peak was detected in the 433 MHz waveform, indicating PU activity. The DL model correctly outputted [1, 0], showing the 433 MHz channel is occupied while the 500 MHz channel is free. The central node allocated the free 500 MHz channel to the SU, effectively managing available spectrum resources.
4. **Single PU Transmission (500 MHz) with a SU Request**:
    - **Procedure:** A PU transmits on the 500 MHz channel while an SU requests to transmit.
    - **Outcome:** A peak appeared in the 500 MHz waveform due to the PU transmission. The DL model accurately outputted [0, 1], indicating the 433 MHz channel is free.

The central node then allocated the unoccupied 433 MHz channel to the SU, optimizing spectrum use.

5. **Single PU Transmission (433 MHz) with Multiple SU Requests**:
    - **Procedure:** A PU transmits on the 433 MHz channel while two SUs request permission to transmit.
    - **Outcome:** A peak in the 433 MHz waveform signaled PU activity. The DL model output was [1, 0], showing the 433 MHz channel occupied but the 500 MHz channel free. The central node allocated the free 500 MHz channel to the first SU in the FIFO queue. The second SU was remained queue until another channel became available.

Dynamic Changes in Spectrum Occupation:

Wireless communication environments are inherently dynamic, with frequent changes in spectrum usage by PUs. To evaluate how well our system adapts to these changes, we simulated scenarios where PU behavior changes dynamically during operation. Successfully passing these tests confirms the robustness of the SS and allocation system.

1. **SU Transmission on 433 MHz with 433 MHz PU Arrival**:
    - **Procedure:** An SU is transmitting on the 433 MHz channel when suddenly, a PU starts transmitting on the same channel.
    - **Outcome:** The central node immediately detected the PU's transmission, evident from a new peak in the 433 MHz waveform. The DL model updated its output to [1, 0], indicating the 433 MHz channel was now occupied and the 500 MHz channel was free. Consequently, the central stopped the SU transmission on the 433 MHz channel and reallocated it to the free 500 MHz channel for continued transmission. This test confirms the system's effectiveness in adapting to dynamic changes in spectrum occupancy.

2. **SU Transmission on 500 MHz with 500 MHz PU Arrival**:
    - **Procedure:** An SU is transmitting on the 500 MHz channel when suddenly, a PU starts transmitting on the same channel.
    - **Outcome:** The central node promptly recognized the PU's transmission, as indicated by a new peak in the 500 MHz waveform. The DL model updated its output to [0, 1], showing that the 500 MHz channel was now occupied while the 433 MHz channel was free. Accordingly, the central node stopped the SU transmission on the 500 MHz channel and reallocated it to the free 433 MHz channel for continued transmission. This test verifies the system's capability to dynamically manage spectrum resources in response to changes in channel occupancy.
3. **Both PUs Arrive During SU Transmissions**:
    - **Procedure:** One or more SUs are actively transmitting on both the 433 MHz and 500 MHz channels when suddenly, PUs arrive and begin transmitting on both channels.
    - **Outcome:** The central node quickly detected the PUs' transmissions, noted by peaks in both the 433 MHz and 500 MHz waveforms. The DL model immediately updated its output to [1, 1], indicating that both channels were now occupied. As a result, the central node instructed all active SUs to cease their transmissions on both channels. The SUs were then placed back in the queue until the channels became available again. This test confirms the system's ability to dynamically adjust and manage multiple spectrum allocations efficiently in response to simultaneous changes across all utilized channels.

8.2 Simulation and Performance Analysis

Continuing from the test case derivation section, the SS and dynamic spectrum DSA system underwent extensive simulations to validate its performance under various controlled scenarios. These simulations were crucial for testing the system's response to different configurations of PUs and SUs, as well as variations in spectrum occupancy.

The simulations were carried out by adjusting the C++ code on the nodes to vary key operational parameters such as the frequency of spectrum holes, the duration of PU transmissions, and the transmission times and data sizes for SUs. This method allowed for a precise control over the testing environment, enabling the assessment of the system's ability to manage spectrum allocations effectively and detect available spectrum bands accurately.

The system demonstrated robust performance across these scenarios, efficiently identifying spectrum holes and allocating them to SUs without causing interference to PUs. It also showed strong adaptability to dynamic changes in spectrum usage, such as varying the duration of PU transmissions or increasing SU transmission demands, by reallocating resources to maintain effective communication.

The controlled simulation environment facilitated a detailed analysis of the system's responses to a wide range of operational demands, ensuring that the spectrum management system was well-prepared for actual deployment. This comprehensive evaluation affirmed that the system could reliably handle diverse and dynamic operational conditions, making it a robust solution for spectrum management.

9. Project Global, Economic, Societal Impact

9.1 Global Impact

Our project has the potential to make a substantial worldwide difference, especially in the realm of telecommunications and wireless communication. To begin with, using DL algorithms can better utilize the available RF spectrum. This results in improved spectrum efficiency, allowing more devices and applications to coexist and communicate simultaneously, which is especially important as the demand for wireless data continues to grow globally.

Moreover, by dynamically reallocating spectrum resources based on real-time sensing and analysis, CR can enhance the reliability and performance of wireless communication. This can have a profound impact on various sectors, including public safety, critical infrastructure, and mobile networks. Moreover, CR technology promotes the coexistence of multiple wireless systems, including government, military, commercial, and unlicensed users. This supports spectrum sharing, reduces interference, and fosters collaboration between different stakeholders.

9.2 Economic Impact

CR technology has the potential to foster innovation in the telecommunications and wireless communication sectors. It can lead to the development of new products, services, and applications, which, in turn, can stimulate economic growth. In addition, successful deployment of CR systems and the associated increase in wireless connectivity can lead to job creation in various sectors, including telecommunications, software development, hardware manufacturing, and system integration.

When it comes to market expansion, companies that successfully integrate DL into their CR systems can gain a competitive edge in the market. This competitive advantage can translate into increased market share and revenue. Companies specializing in DL for wireless

communication can access global markets, potentially leading to international trade and export opportunities.

9.3 Societal Impact

CR technology has improved wireless communication by intelligently managing the radio spectrum. CRs can improve the quality and reliability of voice, video, and data services by choosing the best frequency bands and reducing interference. This results in better user experiences and more dependable connectivity in our increasingly connected world. CR can also play an important role in public safety and emergency communications. In times of emergency or natural disaster, where reliable communication is critical, CRs can quickly adapt to the available spectrum, establishing resilient communication links. This ensures that emergency responders can always stay connected, making CR a critical asset in ensuring public safety.

Furthermore, CR can also be used to improve rural wireless broadband access. CR can provide high-speed internet connectivity to areas that lack traditional infrastructure by intelligently detecting and utilizing unused spectrum. This can aid in bridging the digital divide and promoting socioeconomic development. In addition, CR has the potential to improve connectivity for the IoT. CR can improve efficiency and scalability by managing and optimizing the spectrum for a wide range of IoT devices. By optimizing spectrum usage, CR can also reduce interference in wireless networks and reduce environmental impact. As a result, energy is saved, and the carbon footprint is reduced.

10. Standards

10.1 Communication

Organization	Standard Number	Description/Relevance
IEEE	802.15.4g	This standard is utilized by the LoRa transceiver module connected to both the sensor nodes and the central node for communication. It allows for reliable data transmission with minimal power consumption.
IEEE	802.22	This standard, employed through the RTL-SDR antenna connected to the central node, is foundational for cognitive radio spectrum sensing and allocation. It specifies the operation bands suitable for opportunistic spectrum usage such as bands between 54 MHz to 862 MHz, and the 1300 MHz to 1750 MHz and 2700 MHz to 3700 MHz bands.
IEC	62196	This standard, used by the I²C protocol is employed for interfacing with the DHT-11 and MQ-135 sensor with Arduino nodes. It allows for a reliable data exchange between the sensors and microcontrollers.

10.2 Coding

Organization	Standard Number	Description/Relevance
ISO/IEC	14882	This standard is applied in programming Arduino nodes for sensor data collection and node communication using C++, ensuring efficient and reliable code compatible with the hardware components of the system.
Python Software Foundation	PEP 8	This standard ensures that Python code, especially for developing DL algorithms, adheres to best practices in formatting, naming, and structuring, promoting readability and maintainability.
Python Software Foundation	PEP 257	This standard is crucial for ensuring that Python functions and modules, especially those for data processing, are well-documented. Clear documentation is key to team collaboration, making it easier for team members to understand, maintain, and extend the codebase.

10.3 Quality and Software Reliability

Organization	Standard Number	Description/Relevance
ISO/IEC	25012	This standard provides a framework for data quality, ensuring the accuracy and reliability of sensor data within the system. It defines data quality requirements that are critical for the precision of spectrum sensing and the effectiveness of allocation decisions.
IEEE	1633	This standard offers recommended practices for enhancing software reliability, which is essential for the system's overall reliability and robust error handling. It guides the development of fault-tolerant software that is capable of managing errors efficiently, thereby minimizing system downtime and ensuring continuous operation.

10.4 Power

Ensuring uninterrupted system operation hinges on a reliable power supply. Optimization through energy-efficient hardware and sleep modes is paramount. These measures not only reduce power consumption but also guarantee the consistent delivery of precise spectrum information. By prioritizing power efficiency, the system remains resilient, minimizing disruptions and aligning with sustainability objectives. This approach integrates power reliability as a fundamental aspect of the system's design framework.

10.5 Accuracy

The accuracy of SS and allocation is pivotal. High accuracy in SS is essential to correctly identify the presence or absence of PUs and the availability of spectrum "holes" for SUs. The DL algorithm, being the core of the sensing process, must be finely tuned to minimize false positives and false negatives in spectrum occupancy detection.

11. Conclusion

11.1 Summary

In conclusion, our project aims to address the challenges communication systems are facing due to the increased demand for spectrum resources, driven by the rapid growth of mobile devices, the IoT, and other connected technologies. This rise in demand puts existing 4G and 5G bands under strain, potentially resulting in congestion, interference, and poor communication performance.

To address the problem of spectrum scarcity, our project proposes an intelligent spectrum management system that utilizes advanced low-complexity DL algorithm and CR techniques. The system's DL algorithm continuously monitors and dynamically allocates available spectrum resources using SS. The workflow of the algorithm begins by capturing the RF signals of the PU and preprocessing the digital signal data to enhance the quality of the data and prepare it for analysis. Then the algorithm uses a DL model to decide regarding the occupancy status of the PU frequency band. If the PU frequency band is free, then the SU is allocated the frequency band for communication. If the PU needs to transmit data while an SU is using the band, the algorithm interrupts the SU to allow the PU priority.

Our spectrum management system integrates hardware and software components for efficient real-time spectrum allocation in wireless networks. Utilizing a Raspberry Pi 4 model B, Arduino Uno, antennas, and sensors, creating a compact network. The Raspberry Pi acts as the central node, executing a DL algorithm to manage spectrum resources for PUs and SUs. The central node, equipped with two antennas, one monitors the RF spectrum continuously and another transmits messages to SUs, and Arduino Uno devices serve as PU and SU nodes. In this system, PUs simulate the exclusive frequency band use of licensed users, and SU request guidance for data transmission from the central node, ensuring effective spectrum utilization.

11.2 Future work

DL-based SS is a promising new approach to spectrum management with the potential to revolutionize spectrum utilization. Moving forward, there are numerous main directions that can be investigated to further refine and improve DL-SS systems. One important area of focus is the development of more efficient and accurate DL models for SS, using advances in neural network architecture design, optimization techniques, and training data preparation. Another promising area of research is the development of DL-SS systems to be adapted to the unique characteristics of 6G networks to ensure their continuous relevance and efficacy. Finally, real-world deployment and testing of DL-SS systems must be considered. This is critical for validating these technologies performance in real-world contexts and identifying any potential problems or constraints.

12. References

[1] A. Upadhye, P. Saravanan, S. S. Chandra, and S. Gurugopinath, "A Survey on Machine Learning Algorithms for Applications in Cognitive Radio Networks," in *2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT),* Jul. 2021, pp. 01–06. doi: 10.1109/CONECCT52877.2021.9622610.

[2] "Ericsson Mobility Report," Nov. 2022. Accessed: Oct. 14, 2023. [Online]. Available: https://www.ericsson.com/4ae28d/assets/local/reports-papers/mobility-report/documents/2022/ericsson-mobility-report-november-2022.pdf

[3] Y. A. Hassan, "Spectrum sensing algorithms for cooperative cognitive radio networks," M.S. thesis, American University of Sharjah, Sharjah, UAE, 2010.

[4] S. N. Syed *et al.*, "Deep Neural Networks for Spectrum Sensing: A Review," *IEEE Access*, vol. 11, pp. 89591–89615, 2023, doi: 10.1109/ACCESS.2023.3305388.

[5] P. Chauhan, S. K. Deka, and N. Sarma, "LSTM-enabled prediction-based channel switching scheduling for multi-channel cognitive radio networks," *Phys. Commun.*, vol. 60, pp. 102–136, Oct. 2023, doi: 10.1016/j.phycom.2023.102136.

[6] N. Kassri, A. Ennouaary, S. Bah, and H. Baghdadi, "A Review on SDR, Spectrum Sensing, and CR-based IoT in Cognitive Radio Networks," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 6, 2021, doi: 10.14569/IJACSA.2021.0120613.

[7] M. D. Shehabeldin, "Learning-based spectrum sensing and access for cognitive radio systems," M.S. thesis, American University of Sharjah, Sharjah, UAE, 2015.

[8] J. Xie, J. Fang, C. Liu, and X. Li, "Deep Learning-Based Spectrum Sensing in Cognitive Radio: A CNN-LSTM Approach," *IEEE Commun. Lett.*, vol. 24, no. 10, pp. 2196–2200, Oct. 2020, doi: 10.1109/LCOMM.2020.3002073.

[9] R. Mei and Z. Wang, "Deep Learning-Based Wideband Spectrum Sensing: A Low Computational Complexity Approach," *IEEE Commun. Lett.*, vol. 27, no. 10, pp. 2633–2637, Oct. 2023, doi: 10.1109/LCOMM.2023.3310715.

[10] K. M. Captain and M. V. Joshi, *Spectrum Sensing for Cognitive Radio: Fundamentals and Applications*, 1st ed. Boca Raton: CRC Press, 2021. doi: 10.1201/9781003088554.

[11] V. Sharma and S. Joshi, "A Literature Review on Spectrum Sensing in Cognitive Radio Applications," in *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS),* Madurai, India: IEEE, Jun. 2018, pp. 883–893. doi: 10.1109/ICCONS.2018.8663089.

[12] S. K. Agrawal, A. Samant, and S. K. Yadav, "Spectrum sensing in cognitive radio networks and metacognition for dynamic spectrum sharing between radar and communication system: A review," *Phys. Commun.*, vol. 52, pp. 101–673, Jun. 2022, doi: 10.1016/j.phycom.2022.101673.

[13] 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, Jan. 2019, doi: 10.3390/s19010126.

[14] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Gener. Comput. Syst.*, vol. 82, pp. 395–411, May 2018, doi: 10.1016/j.future.2017.11.022.

[15] S. Benazzouza, M. Ridouani, F. Salahdine, and A. Hayar, "A Survey on Compressive Spectrum Sensing for Cognitive Radio Networks," in *2019 IEEE International Smart Cities Conference (ISC2)*, Casablanca, Morocco: IEEE, Oct. 2019, pp. 535–541. doi: 10.1109/ISC246665.2019.9071710.

[16] D. Janu, K. Singh, and S. Kumar, "Machine learning for cooperative spectrum sensing and sharing: A survey," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 1, p. e4352, 2022, doi: 10.1002/ett.4352.

[17] K. M. Thilina, Kae Won Choi, N. Saquib, and E. Hossain, "Machine Learning Techniques for Cooperative Spectrum Sensing in Cognitive Radio Networks," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 11, pp. 2209–2221, Nov. 2013, doi: 10.1109/JSAC.2013.131120.

[18] Y. Lu, P. Zhu, D. Wang, and M. Fattouche, "Machine learning techniques with probability vector for cooperative spectrum sensing in cognitive radio networks," in *2016 IEEE Wireless Communications and Networking Conference*, Doha, Qatar: IEEE, Apr. 2016, pp. 1–6. doi: 10.1109/WCNC.2016.7564840.

[19] J. Lunden, V. Koivunen, S. R. Kulkarni, and H. V. Poor, "Reinforcement learning based distributed multiagent sensing policy for cognitive radio networks," in *2011 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, Aachen, Germany: IEEE, May 2011, pp. 642–646. doi: 10.1109/DYSPAN.2011.5936261.

[20] H. Xing, H. Qin, S. Luo, P. Dai, L. Xu, and X. Cheng, "Spectrum sensing in cognitive radio: A deep learning based model," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 1, Nov. 2021, doi: 10.1002/ett.4388.

[21] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, Aug. 2017, pp. 1–6. doi: 10.1109/ICEngTechnol.2017.8308186.

[22] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks." arXiv preprint, Dec. 02, 2015. doi: arXiv:1511.08458.

[23] D. Han *et al.*, "Spectrum sensing for cognitive radio based on convolution neural network," in *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, Oct. 2017, pp. 1–6. doi: 10.1109/CISP-BMEI.2017.8302117.

[24] V. Prithiviraj, B. Sarankumar, A. Kalaiyarasan, P. P. Chandru, and N. N. Singh, "Cyclostationary analysis method of spectrum sensing for Cognitive radio," in *2011 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE)*, Feb. 2011, pp. 1–5. doi: 10.1109/WIRELESSVITAE.2011.5940821.

[25] S. Zheng, S. Chen, P. Qi, H. Zhou, and X. Yang, "Spectrum sensing based on deep learning classification for cognitive radios," *China Commun.*, vol. 17, no. 2, pp. 138–148, Feb. 2020, doi: 10.23919/JCC.2020.02.012.

[26] R. Ahmed, Y. Chen, and B. Hassan, "Deep learning-driven opportunistic spectrum access (OSA) framework for cognitive 5G and beyond 5G (B5G) networks," *Ad Hoc Netw.*, vol. 123, pp. 102–632, Dec. 2021, doi: 10.1016/j.adhoc.2021.102632.

[27] G. Van Houdt, C. Mosquera, and G. Nápoles, "A review on the long short-term memory model," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5929–5955, Dec. 2020, doi: 10.1007/s10462-020-09838-1.

[28] B. Soni, D. K. Patel, and M. Lopez-Benitez, "Long Short-Term Memory Based Spectrum Sensing Scheme for Cognitive Radio Using Primary Activity Statistics," *IEEE Access*, vol. 8, pp. 97437–97451, 2020, doi: 10.1109/ACCESS.2020.2995633.

[29] S. Kyperountas, N. Correal, and Q. Shi, "A Comparison of Fusion Rules for Cooperative Spectrum Sensing in Fading Channels," *EMS Res. Mot.*, Jun. 2010.

[30] Y. Saleem and M. H. Rehmani, "Primary radio user activity models for cognitive radio networks: A survey," *J. Netw. Comput. Appl.*, vol. 43, pp. 1–16, Aug. 2014, doi: 10.1016/j.jnca.2014.04.001.

[31] T. Nguyen, B. L. Mark, and Y. Ephraim, "Spectrum Sensing Using a Hidden Bivariate Markov Model," *IEEE Trans. Wirel. Commun.*, vol. 12, no. 9, pp. 4582–4591, Sep. 2013, doi: 10.1109/TWC.2013.072513.121864.

[32] Y. Zeng, C. L. Koh, and Y.-C. Liang, "Maximum Eigenvalue Detection: Theory and Application," in *2008 IEEE International Conference on Communications*, Beijing, China: IEEE, 2008, pp. 4160–4164. doi: 10.1109/ICC.2008.781.

[33] R. Zhang, T. Lim, Y.-C. Liang, and Y. Zeng, "Multi-antenna based spectrum sensing for cognitive radios: A GLRT approach," *IEEE Trans. Commun.*, vol. 58, no. 1, pp. 84–88, Jan. 2010, doi: 10.1109/TCOMM.2010.01.080158.

[34] J. Xie, C. Liu, Y.-C. Liang, and J. Fang, "Activity Pattern Aware Spectrum Sensing: A CNN-Based Deep Learning Approach," *IEEE Commun. Lett.*, vol. 23, no. 6, pp. 1025–1028, Jun. 2019, doi: 10.1109/LCOMM.2019.2910176.

[35] F. Benedetto, L. Mastroeni, and G. Quaresima, "Auction-based Theory for Dynamic Spectrum Access: a Review," in *2021 44th International Conference on Telecommunications and Signal Processing (TSP)*, Jul. 2021, pp. 146–151. doi: 10.1109/TSP52935.2021.9522600.

[36] S. Li, H. Li, Q. Yang, and J. Gaber, "Hybrid Cooperative Spectrum Sharing Protocols Based on OMA and NOMA in Cognitive Radio Networks," *Appl. Sci.*, vol. 12, no. 24, Art. no. 24, Jan. 2022, doi: 10.3390/app122412683.

[37] D. Sumithra Sofia and A. Shirly Edward, "Auction based game theory in cognitive radio networks for dynamic spectrum allocation," *Comput. Electr. Eng.*, vol. 86, p. 106734, Sep. 2020, doi: 10.1016/j.compeleceng.2020.106734.

[38] Q. Ni, R. Zhu, Z. Wu, Y. Sun, L. Zhou, and B. Zhou, "Spectrum Allocation Based on Game Theory in Cognitive Radio Networks," *J. Netw.*, vol. 8, no. 3, pp. 712–722, Mar. 2013, doi: 10.4304/jnw.8.3.712-722.

[39] A. Garhwal and P. P. Bhattacharya, "A Survey on Dynamic Spectrum Access Techniques for Cognitive Radio," *Int. J. -Gener. Netw. IJNGN*, vol. 3, no. 4, pp. 15–32, Jan. 2012, doi: 10.5121/ijngn.2011.3402.

[40] P. Kaur, A. Khosla, and K. Uddin, "Markovian Queuing Model for Dynamic Spectrum Allocation in Centralized Architecture for Cognitive Radios," *Int. J. Eng. Technol.*, vol. 3, no. 1, pp. 96–101, 2011, doi: 10.7763/IJET.2011.V3.206.

[41] M. Saber, A. El Rharras, R. Saadane, A. H. Kharraz, and A. Chehri, "An Optimized Spectrum Sensing Implementation Based on SVM, KNN and TREE Algorithms," in *2019 15th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, Sorrento, Italy: IEEE, Nov. 2019, pp. 383–389. doi: 10.1109/SITIS.2019.00068.

[42] M. Saber, H. K. Aroussi, A. E. Rharras, and R. Saadane, "Raspberry Pi and RTL-SDR for Spectrum Sensing based on FM Real Signals," in *2018 6th International Conference on Multimedia Computing and Systems (ICMCS)*, Rabat: IEEE, May 2018, pp. 1–6. doi: 10.1109/ICMCS.2018.8525867.

[43] D. Uvaydov, S. D'Oro, F. Restuccia, and T. Melodia, "DeepSense: Fast Wideband Spectrum Sensing Through Real-Time In-the-Loop Deep Learning," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, Vancouver, BC, Canada: IEEE, May 2021, pp. 1–10. doi: 10.1109/INFOCOM42981.2021.9488764.