

Chulalongkorn University

Chula Digital Collections

Chulalongkorn University Theses and Dissertations (Chula ETD)

2022

Applications of deep learning framework and localization to intelligent radio spectrum monitoring

Truong Thanh Le
Faculty of Engineering

Follow this and additional works at: <https://digital.car.chula.ac.th/chulaetd>



Part of the [Electrical and Electronics Commons](#)

Recommended Citation

Le, Truong Thanh, "Applications of deep learning framework and localization to intelligent radio spectrum monitoring" (2022). *Chulalongkorn University Theses and Dissertations (Chula ETD)*. 5846.
<https://digital.car.chula.ac.th/chulaetd/5846>

This Thesis is brought to you for free and open access by Chula Digital Collections. It has been accepted for inclusion in Chulalongkorn University Theses and Dissertations (Chula ETD) by an authorized administrator of Chula Digital Collections. For more information, please contact ChulaDC@car.chula.ac.th.

APPLICATIONS OF DEEP LEARNING FRAMEWORK AND LOCALIZATION TO INTELLIGENT
RADIO SPECTRUM MONITORING



Mr. Truong Thanh Le

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering in Electrical Engineering

Department of Electrical Engineering

FACULTY OF ENGINEERING

Chulalongkorn University

Academic Year 2022

Copyright of Chulalongkorn University

การประยุกต์ใช้กรอบการเรียนรู้เชิงลึกและการระบุตำแหน่งกับการตรวจสอบสเปกตรัมวิทยุ



วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2565
ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

Thesis Title	APPLICATIONS OF DEEP LEARNING FRAMEWORK AND LOCALIZATION TO INTELLIGENT RADIO SPECTRUM MONITORING
By	Mr. Truong Thanh Le
Field of Study	Electrical Engineering
Thesis Advisor	Professor WATIT BENJAPOLAKUL, D.Eng.
Thesis Co Advisor	Assistant Professor PANUWAT JANPUGDEE, Ph.D.

Accepted by the FACULTY OF ENGINEERING, Chulalongkorn University in
Partial Fulfillment of the Requirement for the Master of Engineering

----- Dean of the FACULTY OF
ENGINEERING
(Professor SUPOT TEACHAVORASINSKUN, D.Eng.)

THESIS COMMITTEE

----- Chairman
(Associate Professor LUNCHAKORN WUTTISITTIKULKIJ,
Ph.D.)

----- Thesis Advisor
(Professor WATIT BENJAPOLAKUL, D.Eng.)

----- Thesis Co-Advisor
(Assistant Professor PANUWAT JANPUGDEE, Ph.D.)

----- Examiner
(Assistant Professor WIDHYAKORN ASDORNWISED, Ph.D.)

----- External Examiner
(Associate Professor Poompat Saengudomlert, Ph.D.)

ตรง ทัน เล : การประยุกต์ใช้กรอบการเรียนรู้เชิงลึกและการระบุตำแหน่งกับการ
ตรวจสอบสเปกตรัมวิทยุ. (APPLICATIONS OF DEEP LEARNING FRAMEWORK
AND LOCALIZATION TO INTELLIGENT RADIO SPECTRUM MONITORING) อ.ที่
ปรึกษาหลัก : ศ. ดร.วาทีต เบญจพลกุล, อ.ที่ปรึกษาร่วม : ผศ. ดร.ภาณุวัฒน์ จันทร์ภักดี

ในวิทยานิพนธ์นี้ ผู้เขียนได้เสนอระบบตรวจสอบคลื่นความถี่วิทยุอัจฉริยะที่นำเอากรอบ
การเรียนรู้เชิงลึกที่วิทยานิพนธ์นี้พัฒนาขึ้นมาใช้ การเรียนรู้เชิงลึกเป็นวิธีที่ทรงพลังในการจัดการ
งานยากโดยอัตโนมัติ ระบบใช้ RTL-SDR USB Dongle เป็นเซ็นเซอร์เพื่อรวบรวมข้อมูลสเปกตรัม
ต่อเนื่องนี้เป็นอุปกรณ์ราคาประหยัดที่สามารถวัดสัญญาณได้ตั้งแต่ 500kHz ถึง 1700MHz แบนด์
วิดท์สูงสุดของการวัดคือ 2MHz หน้าที่หลักของระบบนี้คือการรวบรวมข้อมูลสเปกตรัม จากนั้น
ตรวจจับสัญญาณการแทน (representation signal) และแยกคุณลักษณะ เช่น แบนด์วิดท์
ความถี่กลาง ประเภทการมอดูเลต และความจุ งานการจำแนกประเภทการมอดูเลตกระทำโดย
แบบจำลองหน่วยความจำระยะสั้นแบบยาวของการเรียนรู้เชิงลึก โมเดลนี้สามารถบรรลุความ
ถูกต้องได้ถึง 92% ในชุดข้อมูลที่ใช้ตรวจสอบความถูกต้อง เมื่อเทียบกับผลลัพธ์ในงานวิจัยของ
Convolutional radio modulation recognition networks O'Shea, T. J., Corgan, J., &
Clancy, T. C. (2016) โมเดลในงานวิจัยนี้มีความแม่นยำสูงกว่า (92% เทียบกับ 87.4%)
นอกจากนี้ แบบจำลองยังสามารถทำงานกับค่าอัตราส่วนสัญญาณต่อสัญญาณรบกวนได้
หลากหลาย ในขณะที่บทความวิจัยอื่นๆ มักจะวิเคราะห์ด้วยค่าอัตราส่วนสัญญาณต่อสัญญาณ
รบกวนค่าเฉพาะค่าใดค่าหนึ่ง สุดท้าย อัลกอริทึมการระบุตำแหน่งถูกสร้างขึ้นเพื่อนำมาใช้ในระบบ
เพื่อค้นหาตำแหน่งของสัญญาณที่ไม่ต้องการหรือผิดกฎหมาย ข้อผิดพลาดเฉลี่ยของอัลกอริทึมการ
ระบุตำแหน่งในการวิจัยนี้คือ 450 เมตร

สาขาวิชา วิศวกรรมไฟฟ้า

ปีการศึกษา 2565

ลายมือชื่อนิสิต

ลายมือชื่อ อ.ที่ปรึกษาหลัก

ลายมือชื่อ อ.ที่ปรึกษาร่วม

6372045621 : MAJOR ELECTRICAL ENGINEERING

KEYWORD: Electrosense, RTL-SDR, Long Short-Term Memory, Intelligent Radio
Spectrum Monitoring, Localization, Deep Learning, Modulation
Classification

Truong Thanh Le : APPLICATIONS OF DEEP LEARNING FRAMEWORK AND
LOCALIZATION TO INTELLIGENT RADIO SPECTRUM MONITORING. Advisor:
Prof. WATIT BENJAPOLAKUL, D.Eng. Co-advisor: Asst. Prof. PANUWAT
JANPUGDEE, Ph.D.

In this thesis, the author proposed an intelligent radio spectrum monitoring system with implemented deep learning framework. Deep learning is a powerful method to handle hard tasks automatically. The system uses the RTL-SDR USB dongle as the sensor to collect the spectrum data. This dongle is a low-cost device that can measure the signal from 500kHz to 1700MHz. The maximum bandwidth of measurement is 2MHz. The main functions of this system are to collect the spectrum data and then detect the representation signals and extract their characteristics, such as bandwidth, center frequency, modulation type, and capacity. The modulation classification task was done with the deep learning Long Short-Term Memory model. The model can achieve up to 92% accuracy in the validated dataset. Compared to the result in the paper Convolutional radio modulation recognition networks O'Shea, T. J., Corgan, J., & Clancy, T. C. (2016), the model shows higher accuracy (92% vs. 87.4%). In addition, the model can run with a wide range of Signal-to-Noise Ratio values, while the other research papers often analyzes with a specific value. Finally, an algorithm of localization was implemented in the system in order to find the position of the unwanted or illegal signals. The average error of the algorithm in this thesis is 450m.

Field of Study: Electrical Engineering

Academic Year: 2022

Student's Signature

Advisor's Signature

Co-advisor's Signature

ACKNOWLEDGEMENTS

The authors would like to thank Chulalongkorn University's Graduate scholarship program for ASEAN or Non-ASEAN countries. In addition, I am very grateful to Professor Watit Benjapolakul and Doctor Le Ngoc Thien, who help me a lot in my study and work. Finally, the author thanks Mr. Le Truong Sinh, Ms. Vo Ai Chi, and Ms. Vo Thuy Nga for supporting me a lot in my spiritual life when I do this thesis.

Truong Thanh Le



TABLE OF CONTENTS

	Page
ABSTRACT (THAI)	iii
ABSTRACT (ENGLISH)	iv
ACKNOWLEDGEMENTS	v
TABLE OF CONTENTS	vi
CHAPTER 1. INTRODUCTION	1
CHAPTER 2. LITERATURE REVIEW.....	3
2.1. Theoretical background:.....	3
2.2. Wireless signal/modulation type classification articles:	4
2.3. Signal detection:.....	7
2.4. Spectrum monitoring software:.....	8
CHAPTER 3. PROPOSED INTELLIGENT RADIO SPECTRUM MONITORING SYSTEM	10
3.1. Hardware overview.....	10
3.1.1. Client-sensor device:.....	10
3.1.2. Central computational server:	12
3.2. Data measurement.....	15
3.3. Signal identification over spectrum.....	19
3.4. Characteristic extraction with deep learning.....	25
3.5. Localizing source of signal.....	32
CHAPTER 4. RESULT AND DISCUSSION.....	39
4.1. Web page for monitoring the spectrum.....	39
4.2. Analyzing the characteristic extraction result.....	42

4.3. Analyzing the signal's source localization.....	45
CHAPTER 5. CONCLUSION	52
REFERENCES.....	55
VITA	58



CHAPTER 1. INTRODUCTION

Nowadays, wireless communication has a large portion of telecommunication technologies. Many new wireless technologies are upcoming, such as 5G, 6G, LoRaWAN, or NB-IoT. Wireless communication is convenient, efficient, and highly portable. With this rapid expansion, one of the most common problems is ensuring the signal source is relatively free from interference. The existence of an illegal or unlicensed signal will lead to a reduction in the capacity, bandwidth, and reliability required for other necessary transmissions. The interference signals could be easily detected by the report of the users (appear time, which application/service is a malfunction, repetition frequency), but detecting the illegal or unlicensed one is a significant challenge because it may not appear continuously or in the same frequency. Therefore, an automatic task that monitors and analyzes the current spectrum is essential.

A spectrum monitoring system will automatically identify, report, and remove illegal or unlicensed interference signals. By monitoring the spectrum continually, problem signals can be identified as they occur in real time. This system can also examine and characterize the patterns of unwanted signals, then locate the transmission source, making the removal task can be done easier. In addition, the spectrum monitoring system can analyze the spectrum occupancy and make a report for government regulators, operators, and administrators to have a plan to utilize, optimize, provide solutions, allow new license registration, re-use, re-establish, and avoid unnecessary violations between broadcasters.

As mentioned above, the signal can be periodic or present at a different frequency over time. Therefore, one particular problem is promptly detecting the illegal or unlicensed frequency and its characteristics/location. In recent days, most of the work of spectrum monitoring has been done manually. The main function of a standard spectrum monitoring system is to measure the signals and broadcast their visualization to the monitor screen for supervision. Then the supervisor will use their acknowledgment and experiment to point out the unwanted ones. This work may

not be efficient and continuous because there will be several distractions when supervising and depending too much on personal decisions. In the Artificial Intelligence era, many tasks can be done automatically without humans. The evolution of computation machine's power could operate complex and huge tasks with incredible speed, accuracy, and reliability. Therefore, implementing AI in the spectrum monitoring system is also a good idea.

The primary goal of this work is to design a spectrum monitoring system that automatically measures signals, detects all frequencies with their characteristics using deep learning, classifies them into legal and unwanted signals, and localizes the illegal or unwanted interference signal. This system will be simply designed to address the problem above. With this basement designation, we can apply it with several applications, which are given below:

- Satellite earth station monitoring.
- Government regulators enforcing spectrum policy.
- Security at military facilities, national borders, utilities, airports, and other sensitive sites where monitors are positioned indoors.
- Monitor jails/prisons for illegal broadcasts
- Spectrum Monitoring usage surveys (white space).
- Airport monitoring for interference.
- Spectrum occupancy and frequency band clearing.
- Sports venue monitoring.

CHAPTER 2. LITERATURE REVIEW

Spectrum monitoring systems usually consist of a variety of functions, such as signal detection over a certain frequency of range, signal demodulation, signal waterfall recording, and feature extraction (bandwidth, center frequency, modulation type). There are several articles that mainly focus on determining the modulation type using Artificial Intelligent/Deep Learning. In addition, a few applications have been released for the user to use the essential functions of a spectrum monitoring system. They will be discussed in the sections below.

2.1.Theoretical background:

All wireless signals have a component path called Carrier Frequency/Signal. When transmitting information through a wireless signal, the information will be modulated with a carrier signal at a designated frequency. Depending on the modulation type, the transmitting signal will be different. When a device receives the signal, $r(t)$, the signal goes through the amplifier path, mixed path, and low-pass filter path, and then the Analog to Digital Conversion (ADC) Module will resample the signal at rate $f_s = 1/T_s$ [2]. The output of this step is the discrete version of the input signal. It includes two components, in-phase r_i and quadrature-phase r_q (IQ).

$$r_n = r_{ni} + jr_{nq} \quad (2.1)$$

From IQ data, the discrete signal can be re-perform as:

$$\begin{aligned} r_n &= A_n \cos \theta_n + j A_n \sin \theta_n \\ \begin{cases} r_{ni} = A_n \cos \theta_n \\ r_{nq} = A_n \sin \theta_n \end{cases} \end{aligned} \quad (2.2)$$

A_n and θ_n are, respectively, the amplitude and phase of the signal at step n .

Fast Fourier Transform (FFT) is an algorithm that computes the Discrete Fourier Transform with less complexity. Applying the FFT to the IQ data will convert the IQ data from time-domain to frequency-domain, and then we can figure out the signal's frequency.

With the frequency representation of a signal, the Signal-to-Noise Ratio is the difference between the signal amplitude and the base noise signal amplitude. The

Signal-to-Noise Ratio is an essential characteristic of the signal. SNR can determine how good the signal is. The data getting by Electrosense open APIs is a graph of SNRs.

2.2. Wireless signal/modulation type classification articles:

The authors of the paper [1] study the possibility of the convolutional neural network to the In-phase and Quadrature-phase (IQ) values of the temporal signal domain. They found that the efficiency and performance of the recent evolutionary artificial intelligent methods are much higher than the classical method, which uses pure and classic theory. Therefore, they demonstrate an approach to radio signal modulation classification using Convolutional Neural Networks (CNNs) and Deep Neural Networks (DNNs), which offer flexibility to extract and learn the features in a wide range of applications and demonstrate proven classification accuracy against the classical approach. The dataset in this paper is available at radioml.com (which is now <https://www.deepsig.ai/datasets>). The dataset consists of 11 modulation types: eight in digital (BPSK, QPSK, 8PSK, 16QAM, BFSK, CPFSK, and PAM4) and three in analog (WB-FM, AM-SSB, and AM-DSB). Data was modulated at a rate of about eight samples per symbol with a normalized average transmit power of 0dB. They consequently extracted the dataset in the length of 128 samples and the shift step of 64 samples. They trained with several candidates for neural network and found that a 4-layer network, which consists of two convolutional layers and two dens fully connected layers, work very well. Figure 2.1 shows their proposed network.

In general, they have approximately 96,000 examples for training and 64,000 examples for testing and validation. After training, they achieved about 87.4% classification accuracy across all signal-to-noise ratios on the test dataset.

The authors in [3] have proposed another model that uses Long-Short Term Memory architecture, which is a particular Recurrent Neural Network and very popular in extracting features from time-series data, to classify the signal's modulation based on the RadioML dataset. In addition, they did not only train the model with In-phase and Quadrature-phase data but also applied the model to the average magnitude FFT data. In summary, they will build two general models with different input sizes. Figure 2.2 gives a view of their proposed model.

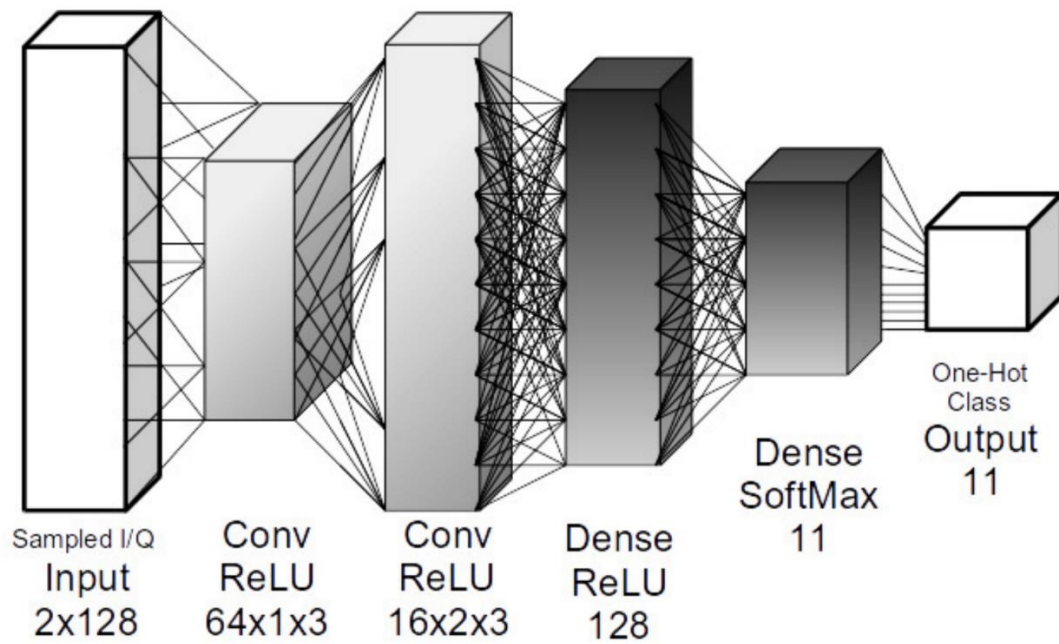


Figure 2.1 The authors in [1] propose the network to classify the signal.

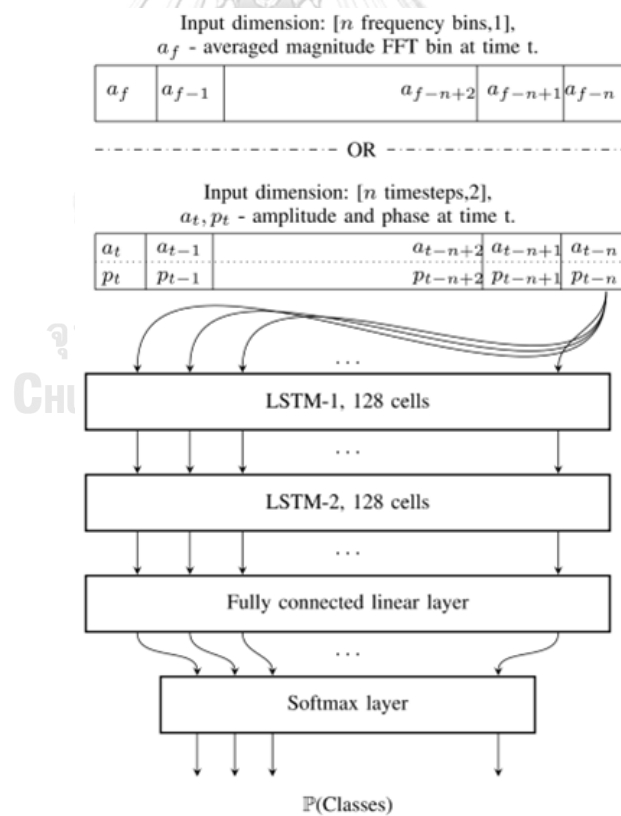


Figure 2.2 The proposed model in [3].

The input of the model can be the average magnitude FFT bin at time t (with dimension n -by-1) or the amplitude (L2 normalized) and phase (normalized between -1 and 1) at time t (with dimension n -by-2). There are two LSTM layers with 128 cells connected to the input layer. The second LSTM layer would connect to a fully connected linear layer. Finally, a dense layer with softmax activation mapping with 11 types of modulations. The authors gave several experiments to demonstrate the difference in performance with various parameters. Figure 2.3 show the confusion matrix for 2-layer amplitude-phase LSTM model on RadioML dataset at 0dB SNR. Figure 2.4 gives a result when changing the number of samples per symbol and sample length. Figure 2.5 demonstrates the result when changing the LSTM depth and number of cells in each LSTM layer.

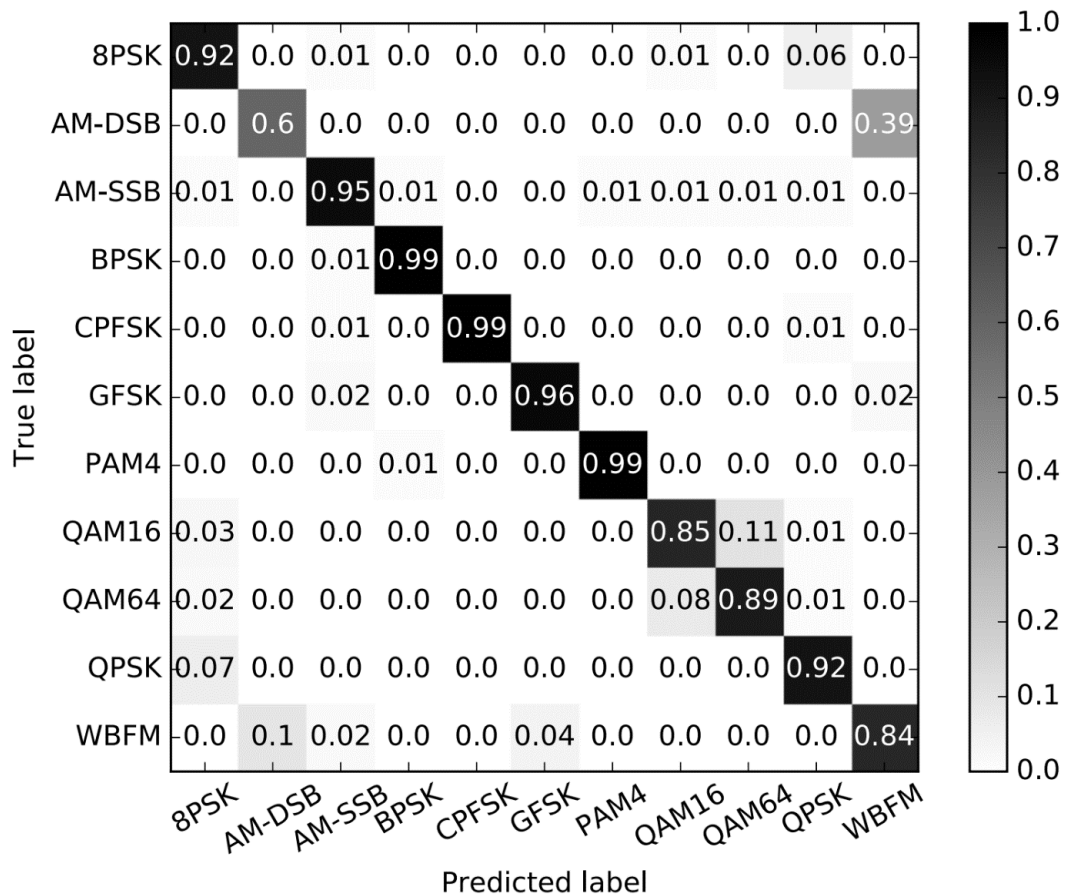


Figure 2.3 The confusion matrix for 2-layer amplitude-phase LSTM model on RadioML dataset at 0dB SNR.

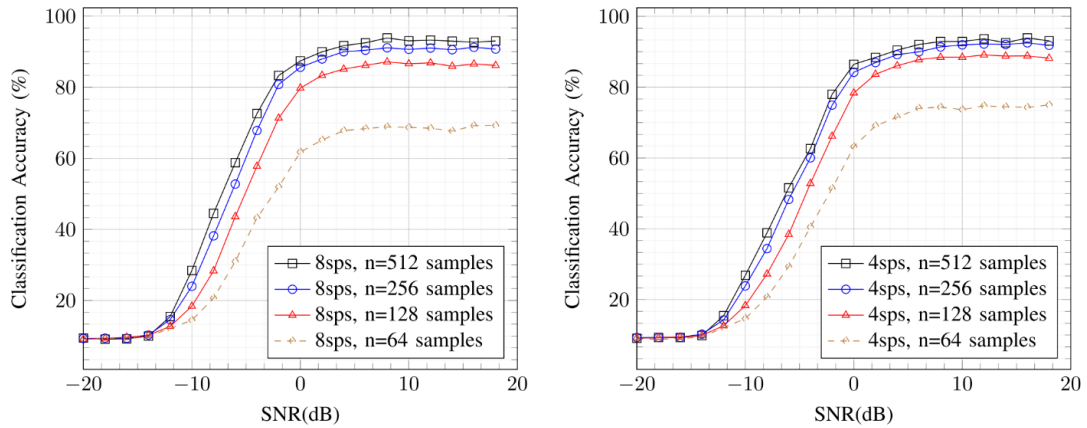


Figure 2.4 The corresponding result when changing the number of samples per symbol and sample length.

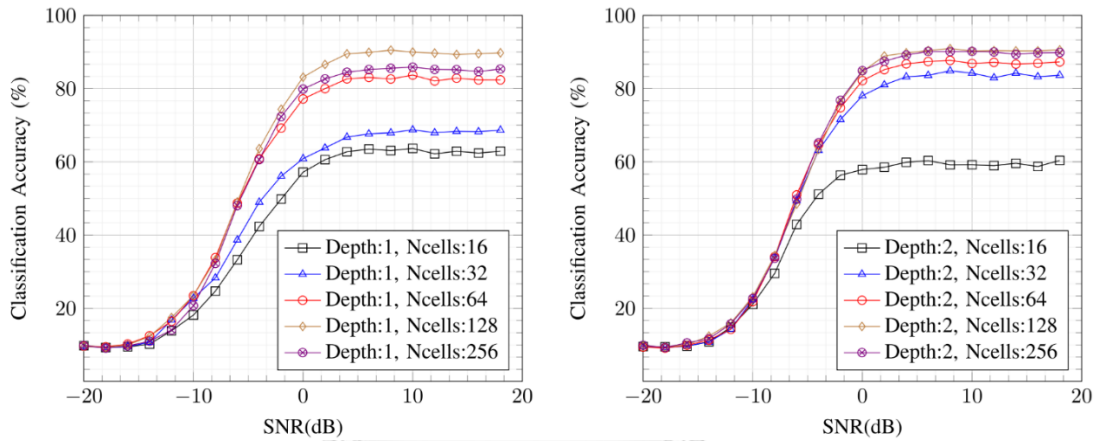


Figure 2.5 The corresponding result when changing the LSTM depth and number of cells in each LSTM layer

2.3.Signal detection:

The authors in [4] proposed a model with Convolutional Neural Network in order to detect the representation of radar band signal through spectrogram/waterfall image. They considered a radio environment where three types of wireless technology may coexist: radars as incumbents, WLAN, and commercial downlink LTE. They performed this sample collection over three different bands: LTE 906 MHz, ISM 2462 MHz, and the 2300 MHz band, where the latter represented a radio medium that is free of interference. Their proposed model is given in Figure 2.6. The result of the research is demonstrated in Figure 2.7.

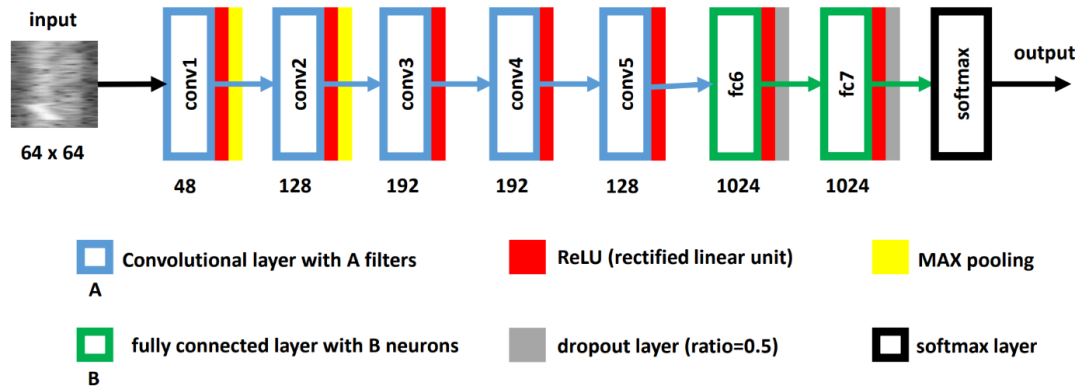


Figure 2.6 The proposed model of paper [4].

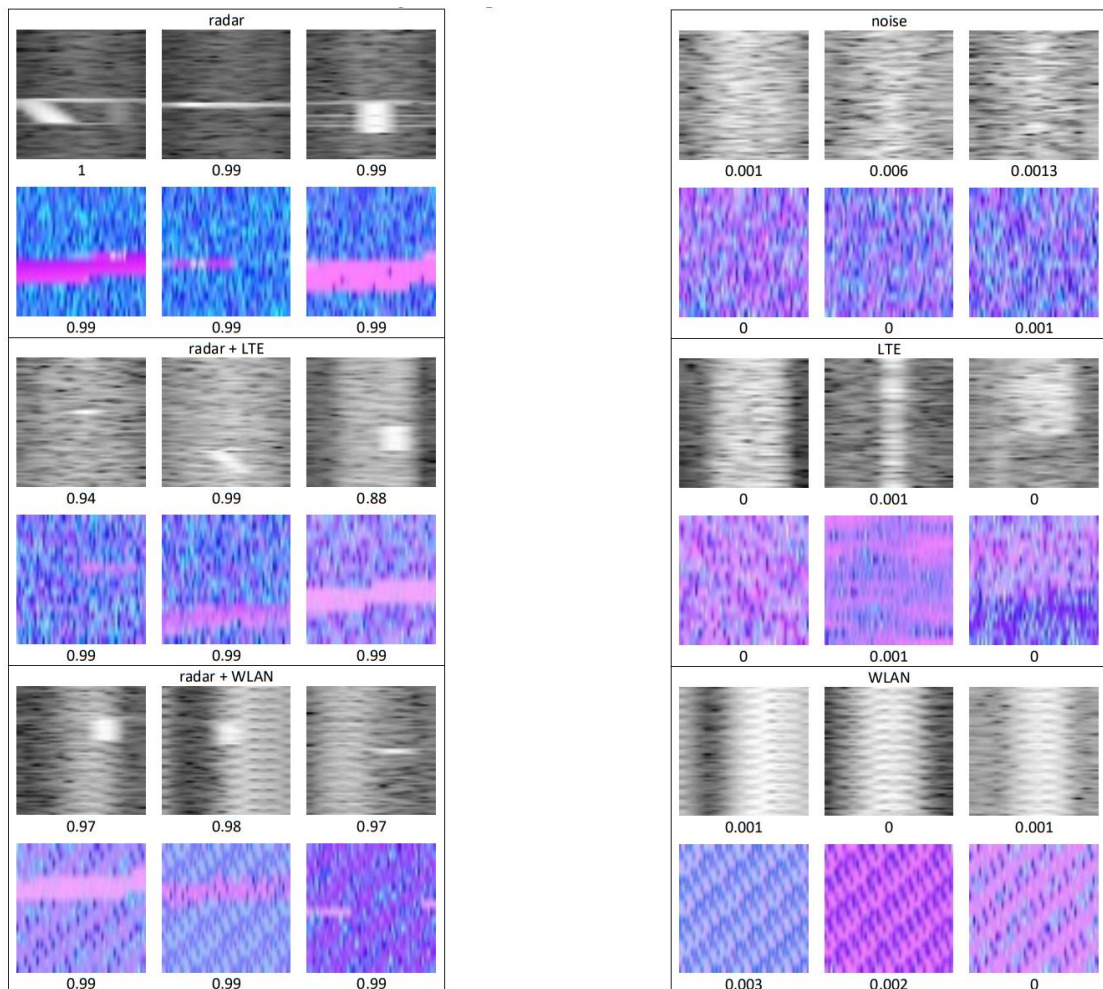


Figure 2.7 The detection result of paper [4].

2.4. Spectrum monitoring software:

There are several released software for spectrum monitoring systems such as SDR-Sharp, HSDR, SDR-RADIO.COM V2/V3, SDR++, Linrad, GQRX, CubicSDR, Studio1,

SDRUno, SigDigger, ShinySDR, WebRadio,... This software has various common functions as below:

- Communicate with the hardware device.
- Monitoring the spectrum at a certain frequency (normally with a range of 2 MHz)
- Adjust antenna gain.
- Demodulate and decode the signal.
- Capture signal.

Most of this software will not automatically detect the representation of a signal. Figure 2.8 gives a look at one of the spectrum monitoring software (SDR-Sharp).

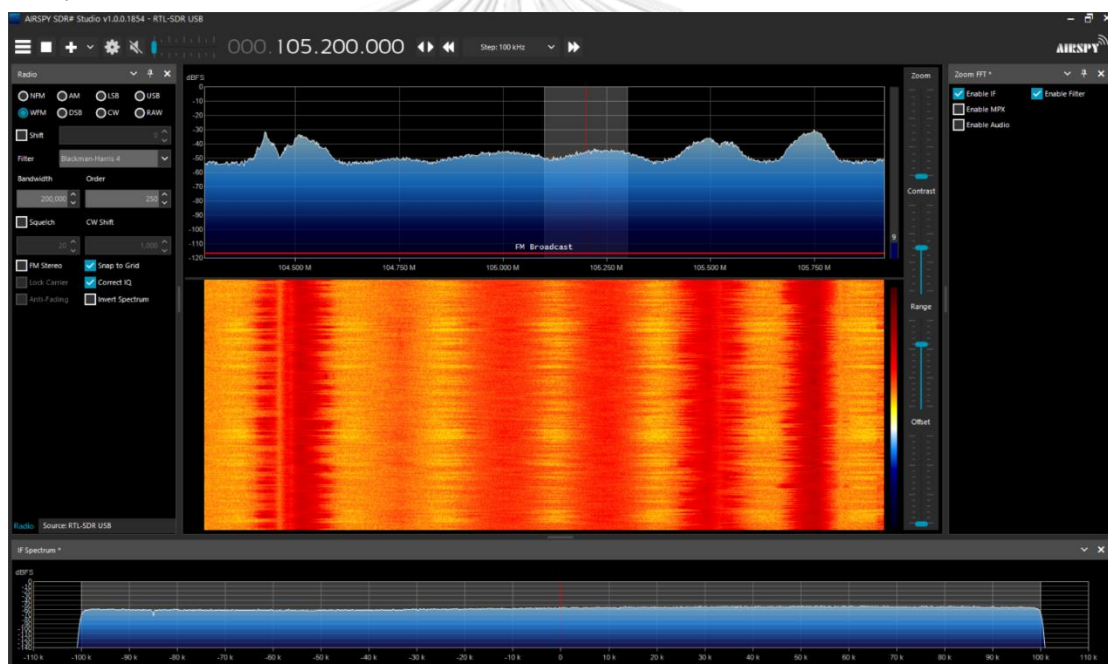


Figure 2.8 SDR-Sharp's graphical user interface.

CHAPTER 3. PROPOSED INTELLIGENT RADIO SPECTRUM MONITORING SYSTEM

3.1. Hardware overview

3.1.1. Client-sensor device:

The remote sensing node used in this project is Electrosense, which consists of small-sized, low-cost, software-defined embedded computing devices connected to a simplistic RF frontend and a general-purpose antenna. The sensors can measure the spectrum ranging from 20 MHz up to 1.3 GHz. If an optional down-converter is applied, the measured range could expand to 6GHz. In general, the Electrosense sensor only consists of an embedded system, a radio frontend, and an antenna. The price of an Electrosense sensor varies from 184.45 to 238 Euros. Normally, the Electrosense sensor hardware will come along with the Electrosense software. The primary function of Electrosense is to monitor the radio spectrum over time. Besides, it can analyze the spectrum occupancy and decode the raw data (FM Radio, AM Radio, ADS-B, ACARS, AIS, LTE). In addition, the default Electrosense software could connect to the public Electrosense server to provide a visual of every function. Figure 3.1, Figure 3.2, and Figure 3.3 are, respectively, the homepage, monitoring page, and decoder page.

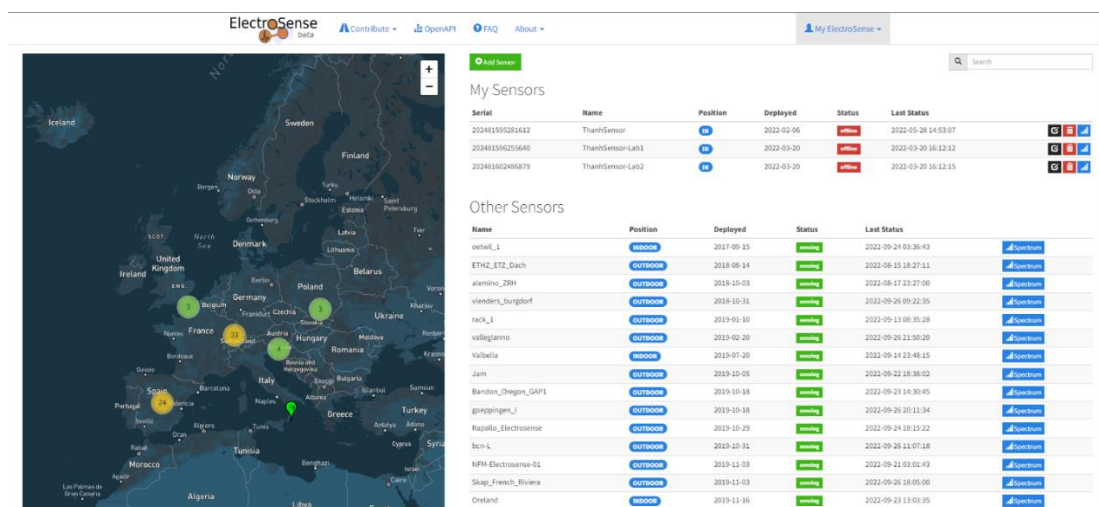


Figure 3.1 Homepage of Electrosense server.

Table 3.1 Electrosense open-APIs information.

Path	Operation	Description
/sensor/list	GET	Get a list of all sensors
/sensor/details/{serial}	GET	Get information about a single sensor
/spectrum/aggregated	GET	Retrieve (aggregated) spectrum measurements of a particular sensor
/spectrum/fft	GET	Retrieve FFT measurements as received by the sensor

The format to use these APIs is shown below:

<https://username:password@electrosense.org/api/> + Path

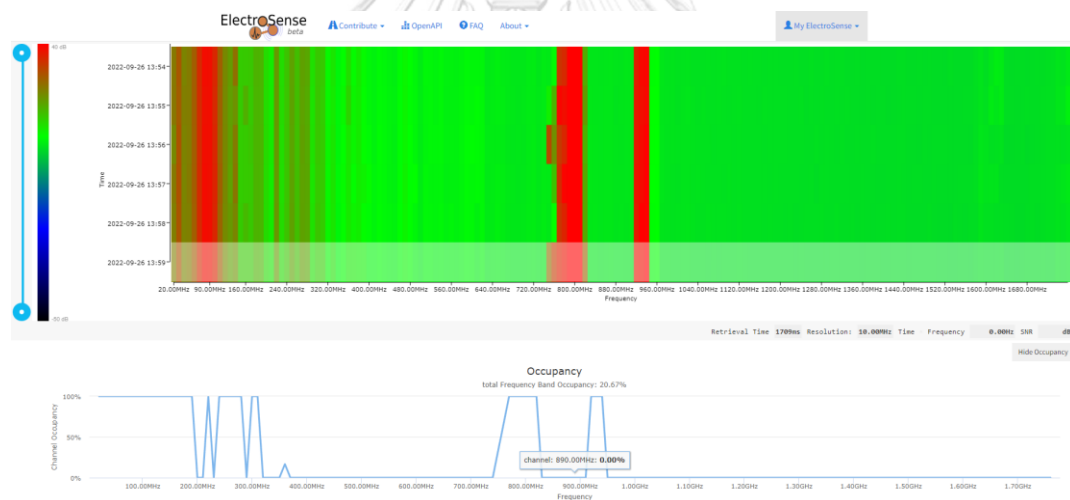


Figure 3.2 Monitoring page of Electrosense server.

Furthermore, the Electrosense server provides several open APIs to retrieve the data. Table 3.1 gives an overview of these APIs. An example of a spectrum image is demonstrated in Figure 3.4.

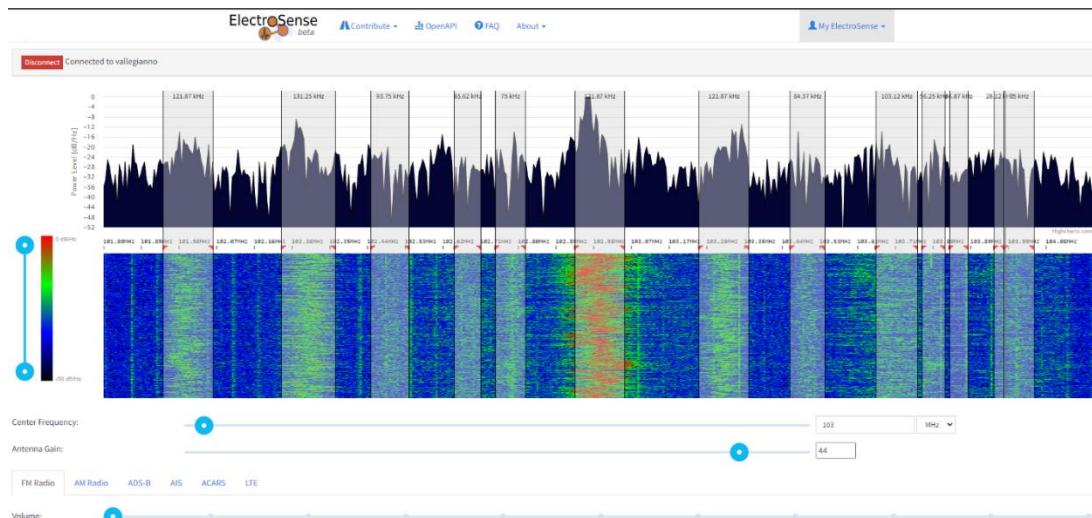


Figure 3.3 Decoding page of Electrosense server.

Although almost necessary functions are provided in this server, it is still lacking for the most significant feature, which is raw IQ data retrieval. In the intelligent radio spectrum monitoring system, raw IQ is the most important data that would contain a lot of important information, such as modulation type and time shifting between receivers. Moreover, the Electrosense server will update the spectrum image of each sensor at a specific time, not in real time. Therefore, the system can not timely detect illegal or unlicensed interference signals, and then we miss our best chance to analyze and take action to remove them. Hence, the new software would be developed and implemented in a new fresh OS on Raspberry Pi 3B+ to avoid the limitation of the Electrosense server. This software will have the basic functions given below:

- Measuring and recording raw IQ data in real time.
- Take the Fast-Fourier Transform on raw IQ data.
- Send the required data to the server.
- Receive command from the server.

3.1.2. Central computational server:

Because the sensor devices run with self-developed software, we could not use the Electrosense server. Therefore, another server should be deployed to fit with

the sensor's functionality. There are two types of servers: cloud and local. With a cloud server, we do not need to maintain the machine periodically. Moreover, the cloud server can be upgraded and expanded (CPU, memory, storage). The data stored on the cloud server would not damage. The cloud server needs high bandwidth and internet speed for accessing, especially when we need to retrieve the data to the local machine for further processing. On the other hand, the local server has a high upload/download speed with the local area network or physical connection, while the price for the hardware and maintenance could be a significant problem.

The local server will be used in this project. The local server machine's specification is given in Table 3.2. There are no GPUs in this machine, so the computation speed will not be considered in the result and discussion.

Table 3.2 Local server machine's specification.

CPU	Intel Xeon Silver 4214
RAM	32GB Error-Correction Coding
Storage	2TB – RAID
GPU	NO
Operating System	Window Server 2019

The local server uses the MQTT protocol based on the Mosquitto platform for communicating with the sensor/client device. In addition, there is a web application on this server for the administrator to monitor and control all the devices in the network. This server has a few functions listed below:

- Receiving and storing data from sensor/client devices.
- Visualizing data and sensor information on the web application.
- Controlling the sensor/client devices.
- Detecting signals from spectrum data.
- Extracting signal characteristics with deep learning.
- Localizing the target signal source.

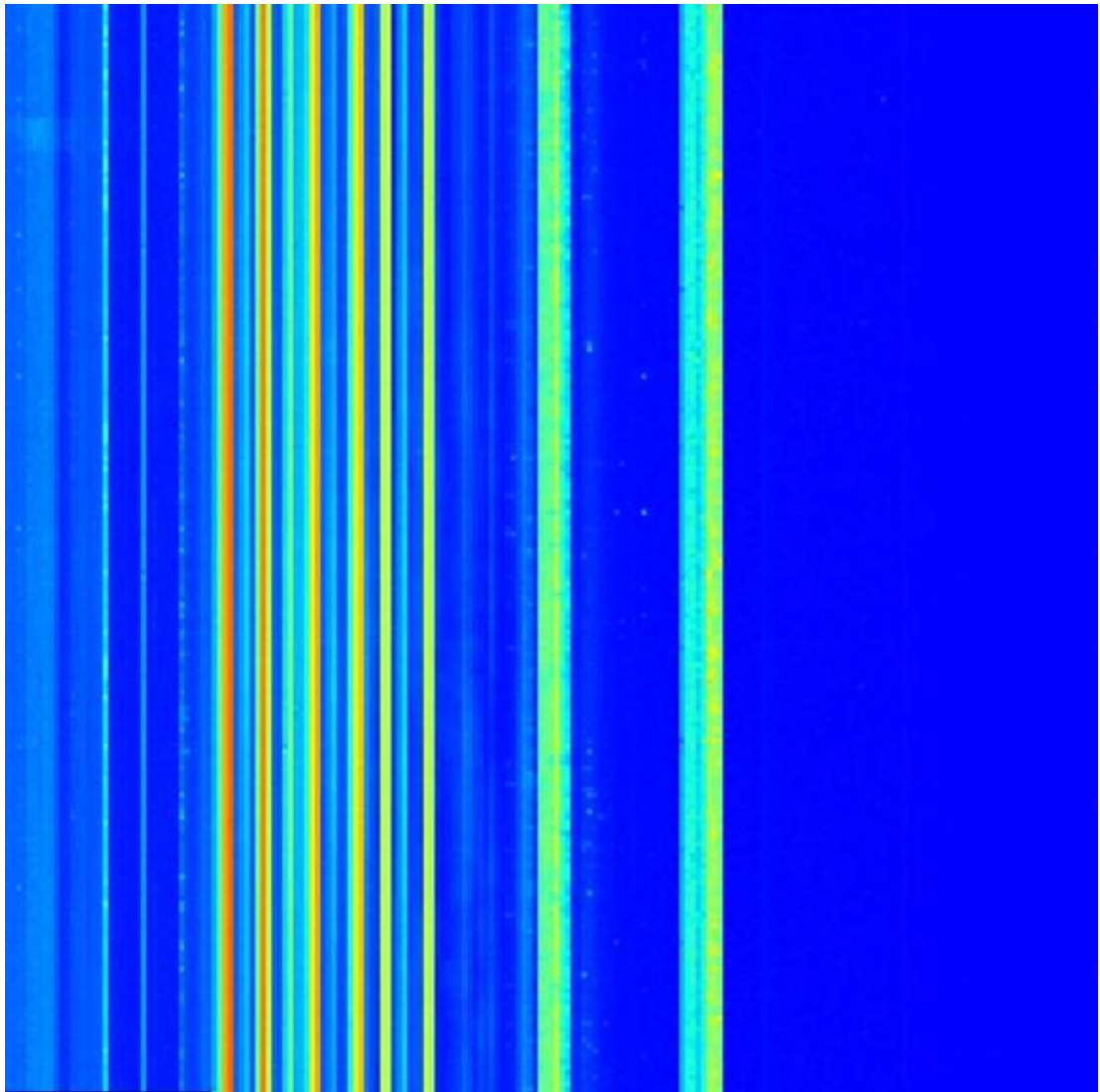


Figure 3.4 Example of spectrum image retrieved from Electrosense server.

Furthermore, to enhance cyber security, a virtual private network (VPN) is a good idea to secure the connection between the clients and the server. All the data flow will be encrypted and routed only inside the VPN. Window Server 2019 has a VPN as a service. It contains a lot of protocols, such as PPTP, L2TP, and SSTP. Unfortunately, all of these protocols has fixed opened port on Window Server 2019 and can not be modified, and the local server connects to the ethernet that only allow port 80 and 443 to be accessed. Therefore, another powerful VPN Server named Open VPN is a better choice because it is easily set up and has an editable port.

3.2.Data measurement

RTL-SDR is a cheap USB dongle that provides a method for scanning the current presented wireless signal (FM, AM, DVB) in the designated set-up area. The received frequency range of RTL-SDR could be from 500kHz to 1.75 GHz, depending on the model. This thesis's intelligent spectrum monitoring system uses the RTL-SDR V3 with RT820T and RTL2832U inside it, given in Figure 3.5. This device could measure the signal frequency from 20MHz to 1766MHz in normal mode and from 500kHz to 24MHz in direct resampling mode. The community developed lots of free software that supports the RTL-SDR, such as SDR-Sharp, HDSDR, and GQRX. The RTL-SDR originates from the mass procedure DVB-T HDTV reception tuner dongle, which is used for receiving and demodulating the digital television signal. Later, with the effort of several contributors, the RTL-SDR has a new function that can directly access the raw IQ data from the built-in RTL2832U chipset. Consequently, the DVB-T could turn into a wideband SDR with a custom software driver.



Figure 3.5 RTL-SDR USB dongle.

The output of RTL-SDR is raw IQ data, a combination of In-phase and Quadrature-phase.

The RTL-SDR provides several resampling rates, where 2.048msps, 2.48msps, and 2.88msps are the most used resampling rate. For example, let's assume that RTL-SDR is now resampling a sine wave $10\cos(2\pi * 100 * 10^6 * t + \pi/6)$ in 100 nanoseconds with the three mentioned resampling rates above. Figure 3.6, Figure 3.7, and Figure 3.8 are the corresponding result. The higher the resampling rate, the more details the output is, but there could be a few missing samples. Figure 3.9 shows a demonstration of a real resampling case when plugging RTL-SDR into the PC and measuring the frequency of 101.5MHz and sampling rate of 2.048msps.

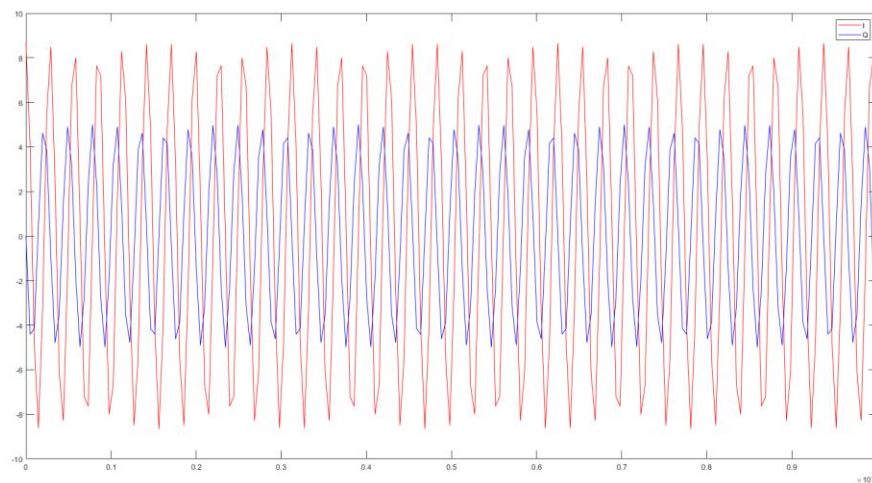


Figure 3.6 Simulating the resampling example sine wave with sampling rate 2.048msps

In conclusion, the sensor device will measure the raw IQ data, take FFT on this data and then return both to the server. The simple code to retrieve raw IQ data and take FFT is demonstrated as Code 3.1. In addition, the RTL-SDR has a resampling bandwidth of 2.4MHz, and when coming far from the center, the FFT spectrum/power level seems to go down. Therefore, we need to combine two adjacent intervals with an overlapped value to avoid this issue. There is a library in python called "Soapy Power" that provide the function mentioned above. Hence, the sensor device will implement both "PyRtlSdr" and "Soapy Power" to retrieve the raw IQ data and FFT spectrum in real-time. Figure 3.11 gives an example of the spectrum image with the range of 300MHz-1000MHz.

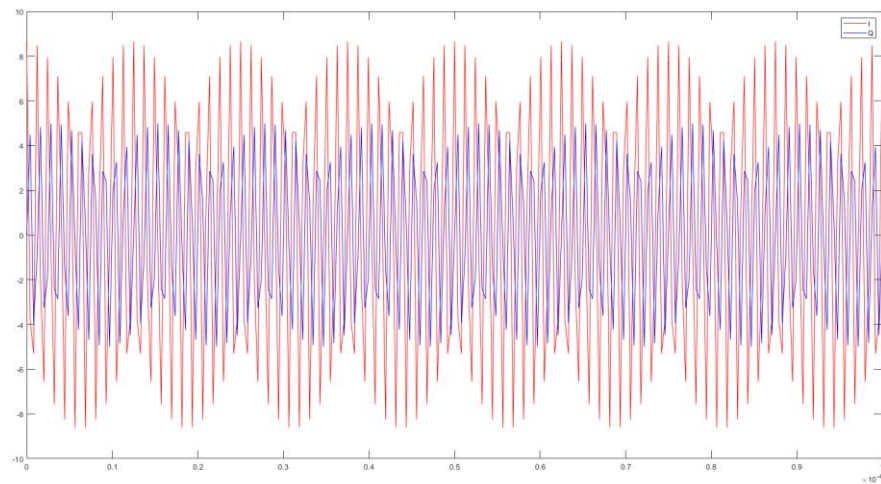


Figure 3.7 Simulating the resampling example sine wave with sampling rate
2.480mps

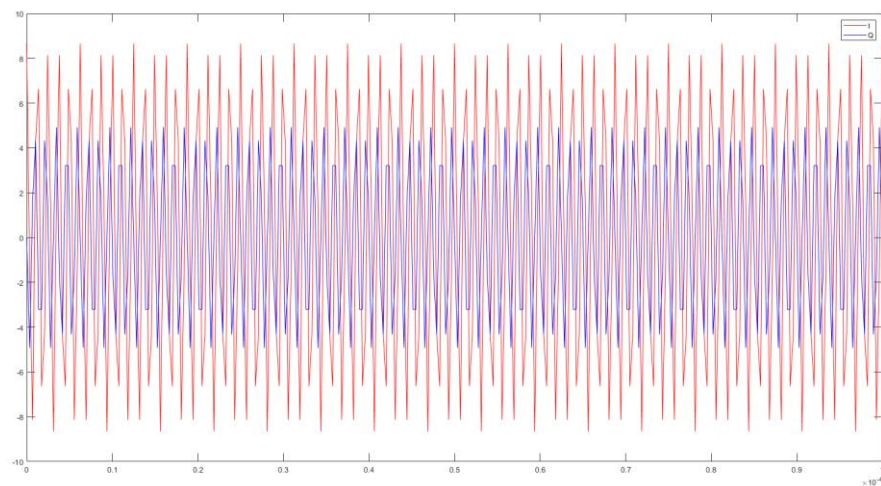


Figure 3.8 Simulating the resampling example sine wave with sampling rate
2.880mps

```

from rtlsdr import RtlSdr
from pylab import *

number_sample = 32*1024

sdr = RtlSdr()
sdr.center_freq = 101.5e6 # change frequency here
sdr.sample_rate = 2.048e6 # change sample rate here
sdr.gain = 37.2 # change antenna gain here
sdr.read_samples(number_sample)
data = sdr.read_samples(number_sample)
magnitude_spectrum(data, Fc=sdr.center_freq,
                   Fs=sdr.sample_rate, scale="dB") # FFT
show()

```

Code 3.1 Example code for retrieving raw IQ data and take FFT.

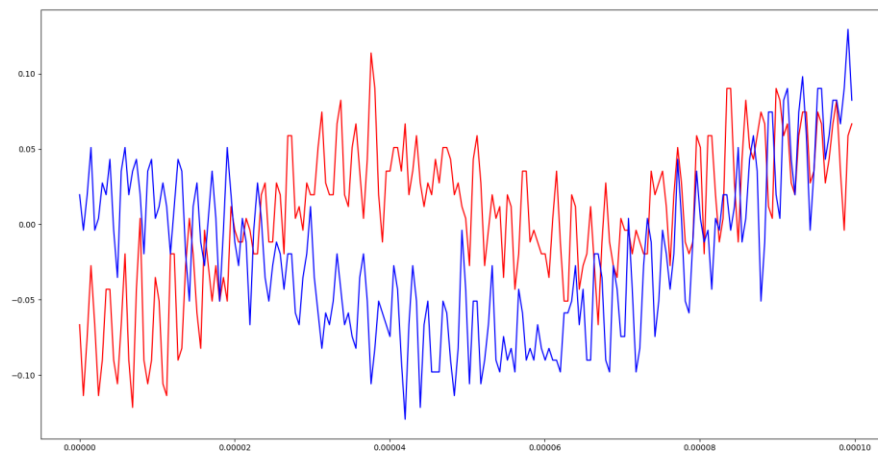


Figure 3.9 Example of IQ data retrieved from RTL-SDR

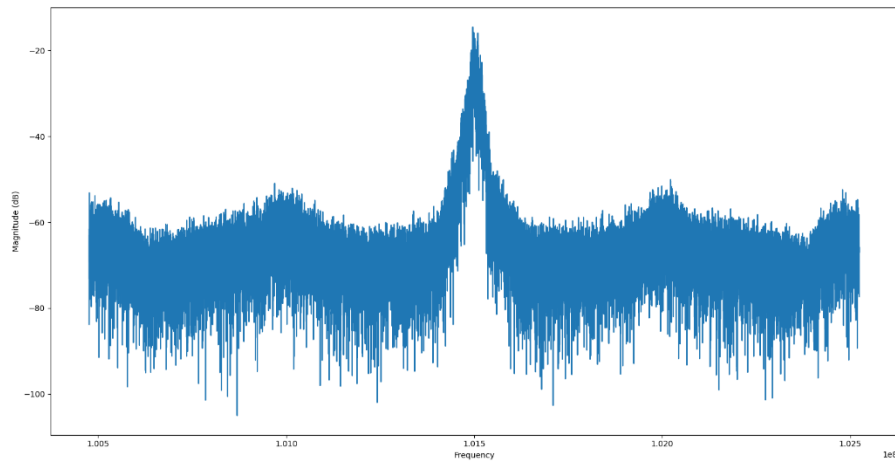


Figure 3.10 FFT magnitude of real raw IQ data.

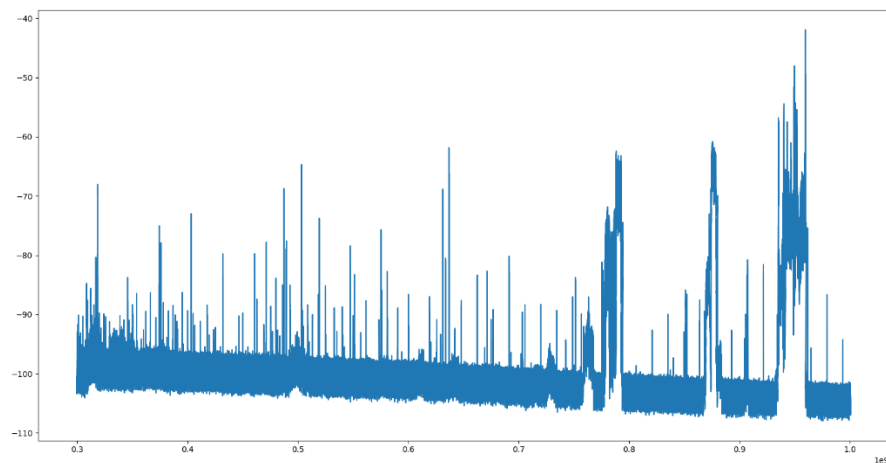


Figure 3.11 An example of spectrum image with the range of 300MHz-700MHz

3.3.Signal identification over spectrum

Once the system receives the spectrum data from the sensor device, it will process to the signal identification function. SNR value is one of the best criteria to determine whether there are signals represented or not. SNR is a value that describes the comparison between the power level of the existing signal and the background noise; in detail, it is a ratio of signal power over the noise power. The unit of SNR is decibels (dB), sometime decibels-miliWatt (dBm). An SNR value higher than 0dB

indicates that the signal power is stronger than the noise power. In general, the formula of SNR is given as equation (3.1):

$$SNR = \frac{P_{signal}}{P_{noise}} \quad (3.1)$$

And equivalents to:

$$\begin{aligned} SNR(dB) &= 10 * \log\left(\frac{P_{signal}}{P_{noise}}\right) \\ &= 10 * \log(P_{signal}) - 10 * \log(P_{noise}) \\ \Rightarrow SNR(dB) &= P_{signal}(dB) - P_{noise}(dB) \end{aligned} \quad (3.2)$$

Where $SNR(dB)$, P_{signal} , $P_{signal}(dB)$, P_{noise} , $P_{noise}(dB)$ are, respectively, SNR values in dB, power of the signal, power of the signal in dB, power of noise, and power of noise in dB.

Furthermore, the power is a proportion of the square of voltage with the value of a resistor as a scaler:

$$P = \frac{V^2}{R} \quad (3.3)$$

Hence,

$$\begin{aligned} SNR(dB) &= 10 * \log\left(\frac{V_{signal}^2}{R} / \frac{V_{noise}^2}{R}\right) = 10 * \log\left(\left(\frac{V_{signal}}{V_{noise}}\right)^2\right) \\ &= 20 * \log\left(\frac{V_{signal}}{V_{noise}}\right) \end{aligned} \quad (3.4)$$

In the equation (3.4), both power of signal and noise use the same resistor value because they use the same antenna to measure.

Looking at the spectrum in Figure 3.12, for example, we can manually find out the presented signal by determining the range that has a greater value than the surrounding.

The SNR value is also a criterion for rating signal strength. Table 3.3 explains this criterion:

Table 3.3 Criteria for rating the signal strength.

SNR	Description
>40	Excellent signal. Lightning fast, always associated.
25-40dB	Very good signal. Very fast, always associated

15-25dB	Low signal. Usually fast, always associated
10-15dB	Very low signal. Mostly slow, usually associated
0-10dB	No signal, almost never associated, agonizingly slow
<0dB	Impossible to associate. In reality, CDMA and WCDMA technology can make connection.

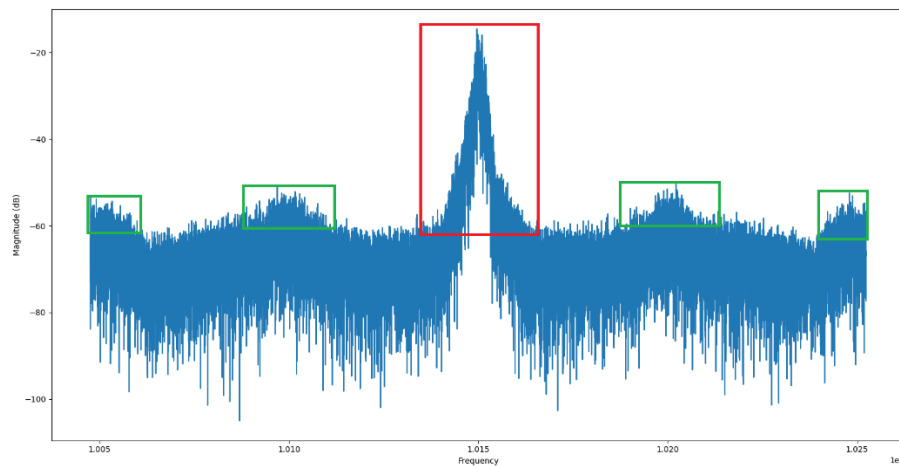


Figure 3.12 Manually detecting the signal over spectrum image.

Because the noise floor value is not an absolute value, the system will use the threshold of 10dB to indicate that there is a signal.

Defining the noise power (noise floor) value is the most important step in finding out the SNR values from the FFT spectrum. From Figure 3.12, we can define the noise floor manually in Figure 3.13. Unfortunately, the intelligent radio spectrum monitoring system needs to do this step automatically.

The noise spectra are often displayed in three formats: power spectral density (PSD), amplitude spectral density (ASD), or power spectrum. The equation (3.5) shows the relation between these formats:

$$ASD = \sqrt{PSD} = \frac{Spectrum}{\sqrt{\Delta f * NoisePowerBandwidth}} \quad (3.5)$$

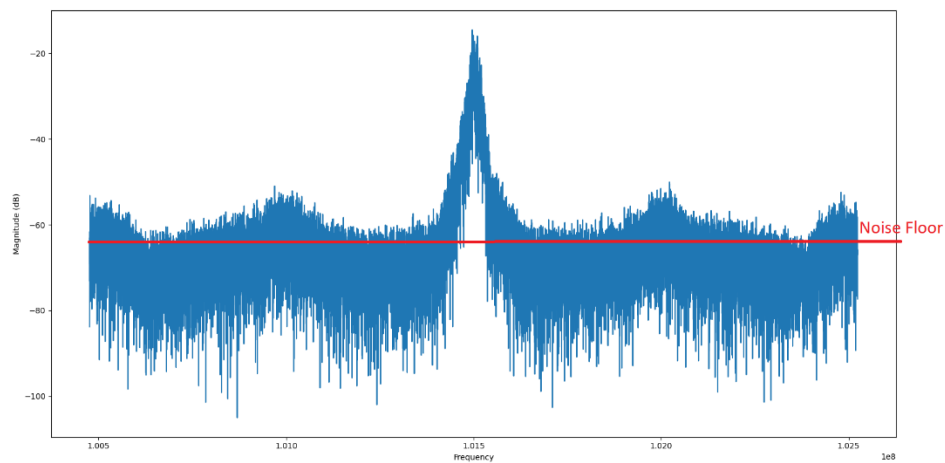


Figure 3.13 Manually defining the noise floor.

Where ***Spectrum*** is the FFT level spectrum (dB or W or V), Δf is the FFT bin resolution (Hz), and ***NoisePowerBandwidth*** can be found in Table 3.4. This project uses the default window in the Soapy Power library, which is the Hann window.

Table 3.4 Window Scaling Factors.

Window	Noise Power Bandwidth
Blackman Harris 3-term	1.73
Blackman Harris 4-term	2.00
Dolph-Chebyshev 150 dB	2.37
Dolph-Chebyshev 200 dB	2.73
Dolph-Chebyshev 250 dB	3.05
Equiripple	2.63
Flat-top	3.83
Gaussian	2.21
Hamming	1.36
Hann	1.50
None (rectangular)	1.00
None, move to bin center	1.00
Rife-Vincent 4-term	2.31

Rife-Vincent 5-term	2.62
---------------------	------

The RMS noise level can be found by:

$$Noise = \sqrt{\sum_{i=1}^{NumBins} PSD_i * \Delta f} = \sqrt{\frac{\sum (Bin_i)^2}{NoisePowerBandwidth}} \quad (3.6)$$

Where **Noise** is the RMS noise level (W or V), and **Bin_i** is the FFT level spectrum (W or V). Then, the noise power in dB is:

$$Noise(dB) = 20 * \log(Noise) \quad (3.7)$$

Furthermore, whenever we double the number of FFT bins, the bin size reduces by half, as well as the noise power in each bin. This is equivalent to 3dB lower in the RMS noise value. The **Noise(dB)** in equation (3.7) is the general noise power, then the **Noise(dB, n)**, the noise power corresponding to **n** FFT bins/number samples, is:

$$Noise(dB, n) = Noise(dB) - 3 * \log_2(n) \quad (3.8)$$

Code 3.2 demonstrates how we compute the noise value, and the result shows in Figure 3.14.

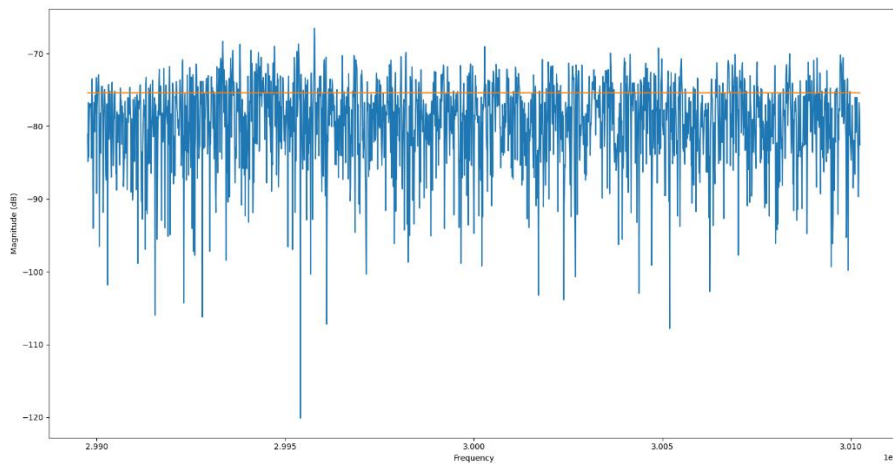


Figure 3.14 Noise floor calculation from simulation data without any signals.


```

import numpy as np
import math
from pylab import *
from rtlsdr import RtlSdr
sdr = RtlSdr()
sdr.center_freq = 101.5e6
sdr.sample_rate = 2.048e6
sdr.gain = 0
sdr.read_samples(1000000)
num_sample = 256*1024
data = sdr.read_samples(num_sample)
[bin_mag, freq, _] = plt.magnitude_spectrum(
    data, scale="dB", Fs=sdr.sample_rate, Fc=sdr.center_freq)
bin_mag = 20*np.log10(np.array(bin_mag))# simulate the data sent from sensor
device
s = 0
for i in bin_mag:
    s += (10**(i/20))**2
noise = 20*math.log10(math.sqrt(s/1.5))-3*(math.log2(num_sample))
print(noise)
plt.plot([freq[0], freq[-1]], [noise, noise])
plt.show()

```

Code 3.2 Noise floor calculation code.

When changing to the frequency range with signal representation, the noise floor becomes like Figure 3.15.

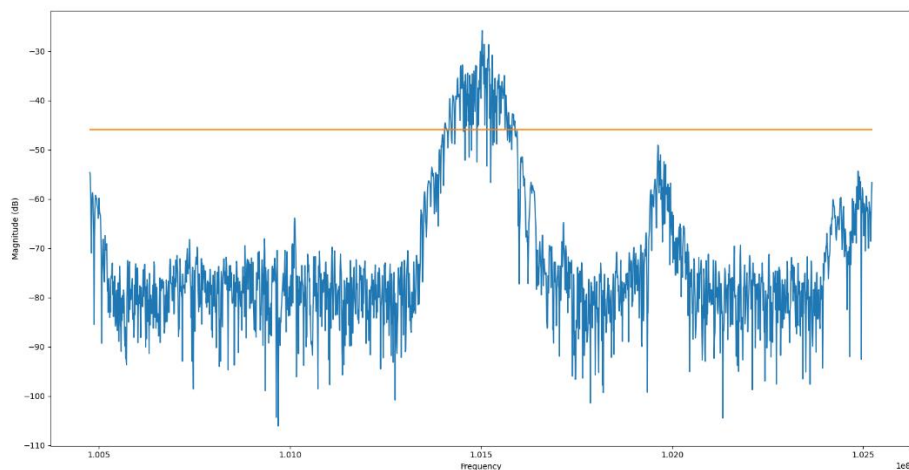


Figure 3.15 Noise floor calculation code from simulation data with signal representation.

Comparing Figure 3.14 and Figure 3.15, we see that the noise floor from Figure 3.14 can be applied to Figure 3.15. Therefore, we define a new critical, using only the noise floor where there is no signal that have $SNR > 10$. Figure 3.16 demonstrates determining the noise floor over a wide range of frequencies.

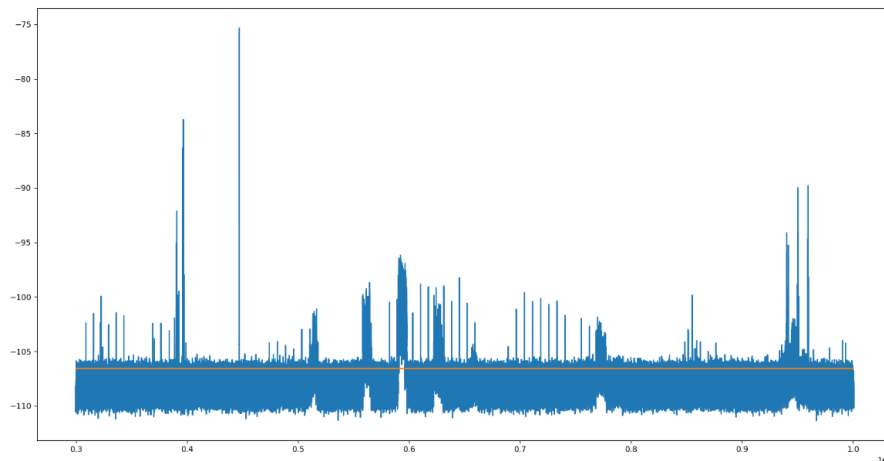


Figure 3.16 Noise floor calculation from real data over wide range frequency

3.4.Characteristic extraction with deep learning

In information theory, we have several valuable characteristics to extract from the wireless signal, such as bandwidth, SNR value, bit rate, channel capacity, and modulation type. All of this information except modulation type can be determined through several formulas. This section will describe how to achieve these characteristics.

First of all, is the SNR value. As mentioned in the previous section, we can find the noise power value. With this noise power value, we iterate all the bins in the spectrum curve and compare each bin with this value. Each bin's SNR value is easy to calculate by using equation (3.4). In addition, there may be several signal bins that have SNR values larger than the pre-defined threshold. Moreover, all types of wireless signals rarely have a bandwidth of a few kHz. Therefore, we will eliminate all the continuous bins with less than pre-defined bandwidth. In detail, the bandwidth threshold is 3kHz. When all the continuously satisfied SNR bins have been found, the general SNR value is their average or maximum. Furthermore, the bandwidth of the

signal is the combination of the beginning and the ending of these continuously satisfied SNR bins. Figure 3.17 gives an example of this step.

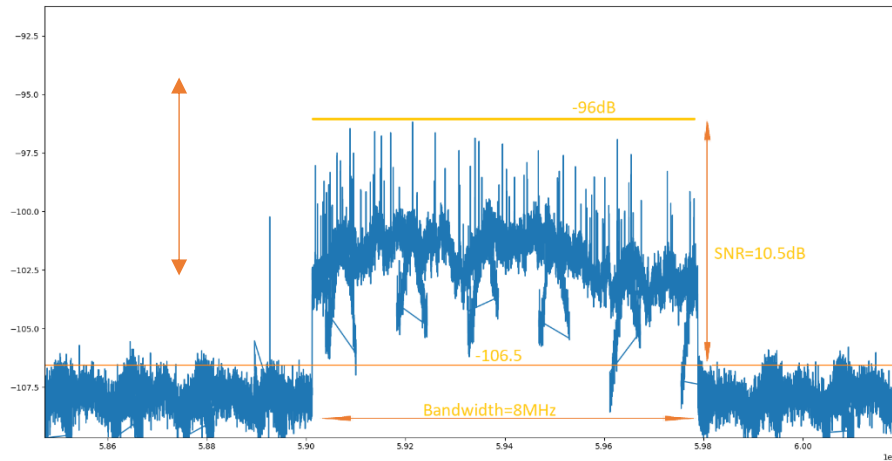


Figure 3.17 Signal to Noise Ratio example

The second is the bit rate and channel capacity. In the noiseless channel, bit rate, or Nyquist bit rate, is a formula that determines the maximum bit rate in theory or ideal. When a signal goes through a low-pass bandwidth filter, the system can reconstruct the signal with a sampling rate of about two times the bandwidth. It is pointless when the sampling rate exceeds two times the bandwidth because the signal's high-frequency components could be filtered out. The formula for the Nyquist theorem is:

$$\text{BitRate (bits/second)} = 2 * \text{Bandwidth} * \log_2(L) \quad (3.9)$$

With L is the number of discrete levels of the signal.

Unfortunately, we cannot know the exact number of discrete levels the signal has. On the other hand, the Digital-to-Analog Converter of RTL-SDR quantized the signal at the 255 levels. Hence, we can assume that L is equal to 255. In reality, the transmitted signals do not have the ideal condition, and there is always a noise effect to the signal. Hence, the Shannon capacity formula replaces the Nyquist to determine the highest possible data rate for a noisy channel. Equation (3.10) is the Shannon capacity formula:

$$\text{Capacity} \left(\frac{\text{bits}}{\text{second}} \right) = \text{bandwidth} * \log_2(1 + \text{SNR}) \quad (3.10)$$

Because the bandwidth is fixed, the channel capacity is the variable that depends on the SNR value.

Last, modulation recognition is the hardest step in characteristic extraction. There are no general formulas or guidelines to do this. Therefore, deep learning is a good method to handle this step. Deep learning is a branch of machine learning, which is totally a neural network. Deep learning applications try to reproduce and simulate the behavior and action of the human brain on an actual task/work. These works may be so complicated that the machine cannot work perfectly, but it can maximize the ability to do that. In addition, the deep learning application can learn from a large amount of data and process them with high computation speed, which takes very much time for a human. Deep learning leverages various AI applications and services to improve accuracy, working speed, and reliability, promote automatic tasks, perform analysis, and learn patterns without human intervention. The intelligent radio spectrum monitoring takes action on modulation classification based on deep learning architecture, such as CNN and LSTN, as in the literature review chapter.

The modulation classification task based on deep learning will use the training dataset from RadioML 2018. The dataset contains 24 modulation types: OOK, 4ASK, 8ASK, BPSK, 8PSK, 16PSK, 32PSK, 16APSK, 32APSK, 64APSK, 128APSK, 16QAM, 32QAM, 64QAM, 128QAM, 256QAM, AM-SSB-WC, AM-SSB-SC, AM-DSB-WC, AM-DSB-SC, FM, GMSK, OQPSK. There are more than two million samples in the dataset, with an equal rate between each class. The data were recorded at the sampling rate of 2.048msps. The sample length is 1024 points of raw IQ data, and the SNR is from -20dB to 30dB. Because the threshold of SNR is 10dB, we will take only the sample that has SNR over this threshold. First, the modulation classification task uses the CNN architecture; then, it will use the LSTM. Because the dataset is too large, the model will train with the incremental/online learning method to reduce average VRAM usage. In addition, online learning is very suitable for further data from RTL-SDR. Figure 3.18 gives an overview of the model using CNN, and Table 3.6 summarizes the model and the total number of parameters in it.

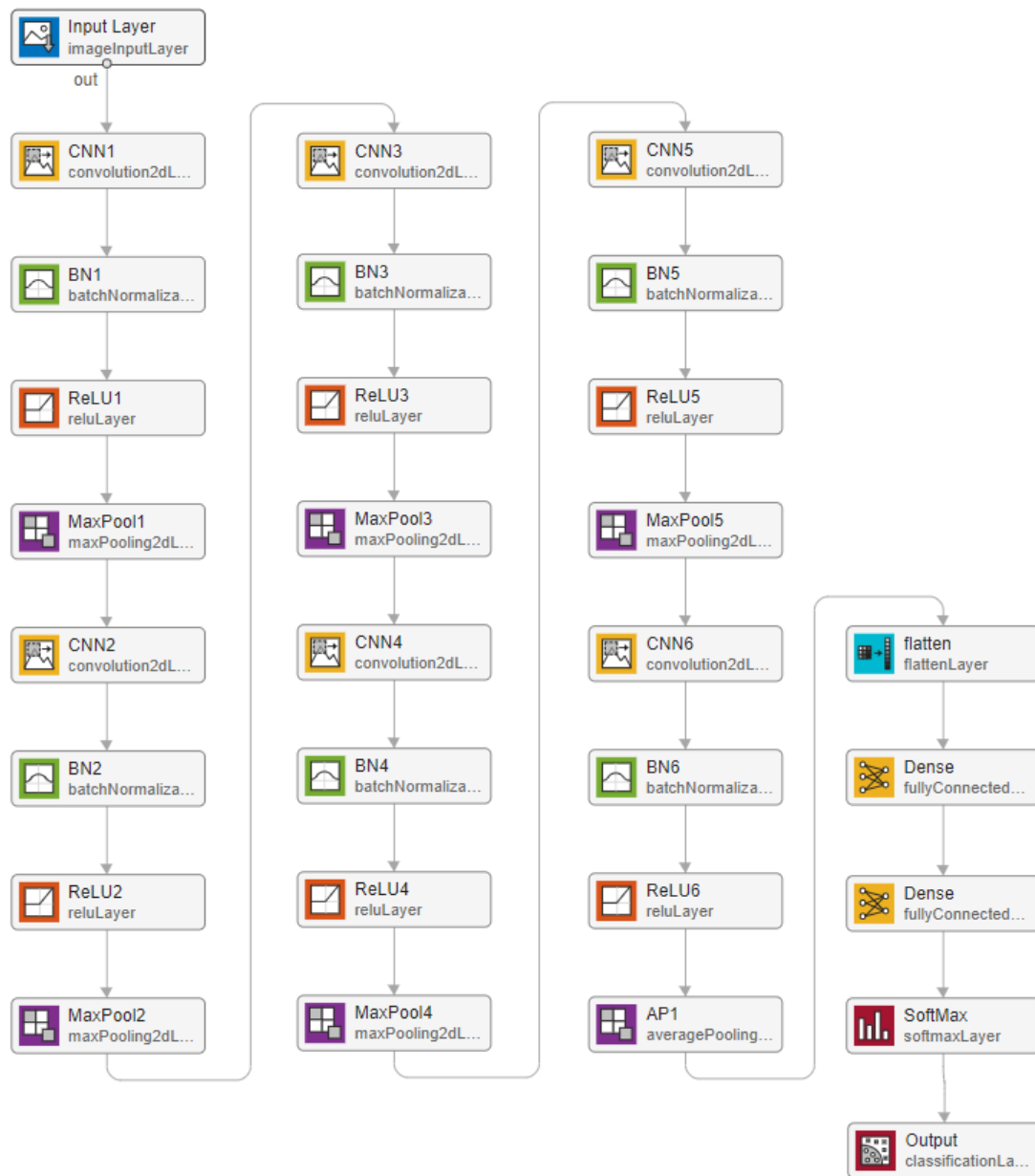


Figure 3.18 CNN model.

Table 3.5 CNN model summary.

Layer	Output Shape	Param
Conv1D	(1024, 16)	272
Batch_Normalization	(1024, 16)	64
Max_Pooling1D	(512, 16)	0
Conv1D	(512, 32)	4128
Batch_Normalization	(512, 32)	128

Max_Pooling1D	(256, 32)	0
Conv1D	(256, 32)	8224
Batch_Normalization	(256, 32)	128
Max_Pooling1D	(128, 32)	0
Conv1D	(128, 64)	16448
Batch_Normalization	(128, 64)	256
Max_Pooling1D	(64, 64)	0
Conv1D	(64, 128)	65664
Batch_Normalization	(64, 128)	512
Max_Pooling1D	(32, 128)	0
Conv1D	(32, 128)	131200
Batch_Normalization	(32, 128)	512
Average_Pooling1D	(1, 128)	0
Flatten	128	0
Dense	128	16512
Dense	24	3096
Total params		247144

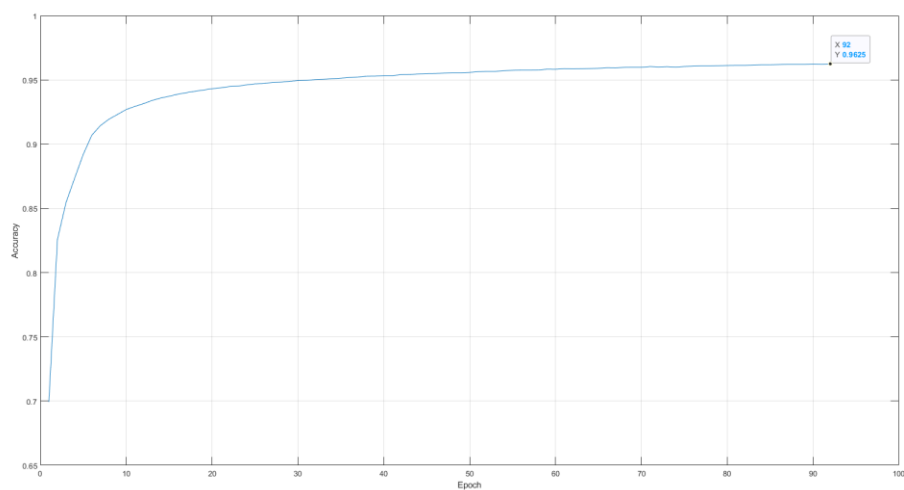


Figure 3.19 CNN model's training accuracy log.

With the LSTM model, it can achieve 96.25% accuracy in the training dataset and 87.51% in the validating dataset. It ran in a total of 92 epochs (2.8 hours). Figure 3.19 and Figure 3.20 shows the log of training and testing accuracy.

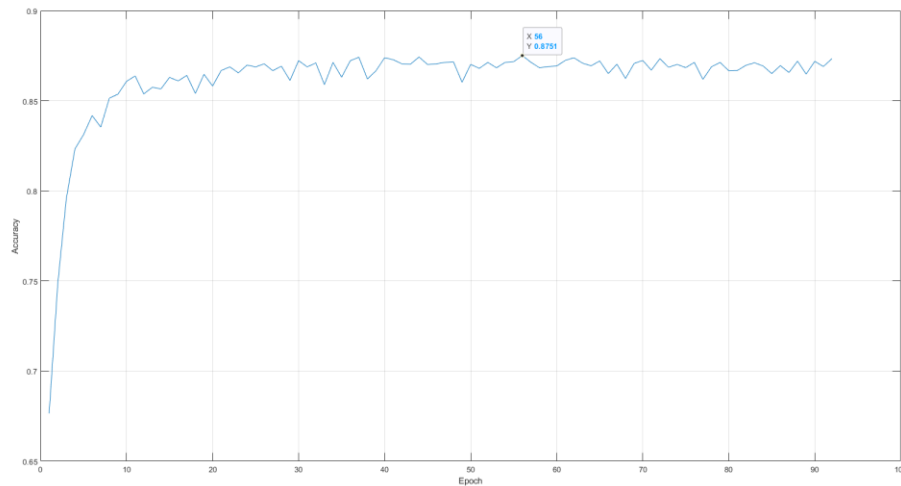


Figure 3.20 CNN model's validating accuracy log.

Figure 3.21 gives an overview of the model using LSTM, and Table 3.7 summarizes the model and the total number of parameters in it.

Table 3.6 LSTM model summary.

Layer	Output Shape	Param
BiLSTM	(1024, 128)	34304
Dense	(1024, 64)	8256
Conv1D	(1024, 64)	4160
Max_Pooling1D	(512, 64)	0
Conv1D	(512, 64)	4160
Max_Pooling1D	(256, 64)	0
Conv1D	(256, 64)	4160
Max_Pooling1D	(128, 64)	0
Conv1D	(128, 64)	4160
Max_Pooling1D	(64, 64)	0
Conv1D	(64, 64)	4160

Max_Pooling1D	(32, 64)	0
Flatten	(2048)	0
Dense	(32)	65568
Dense	(24)	792
Total params		129720

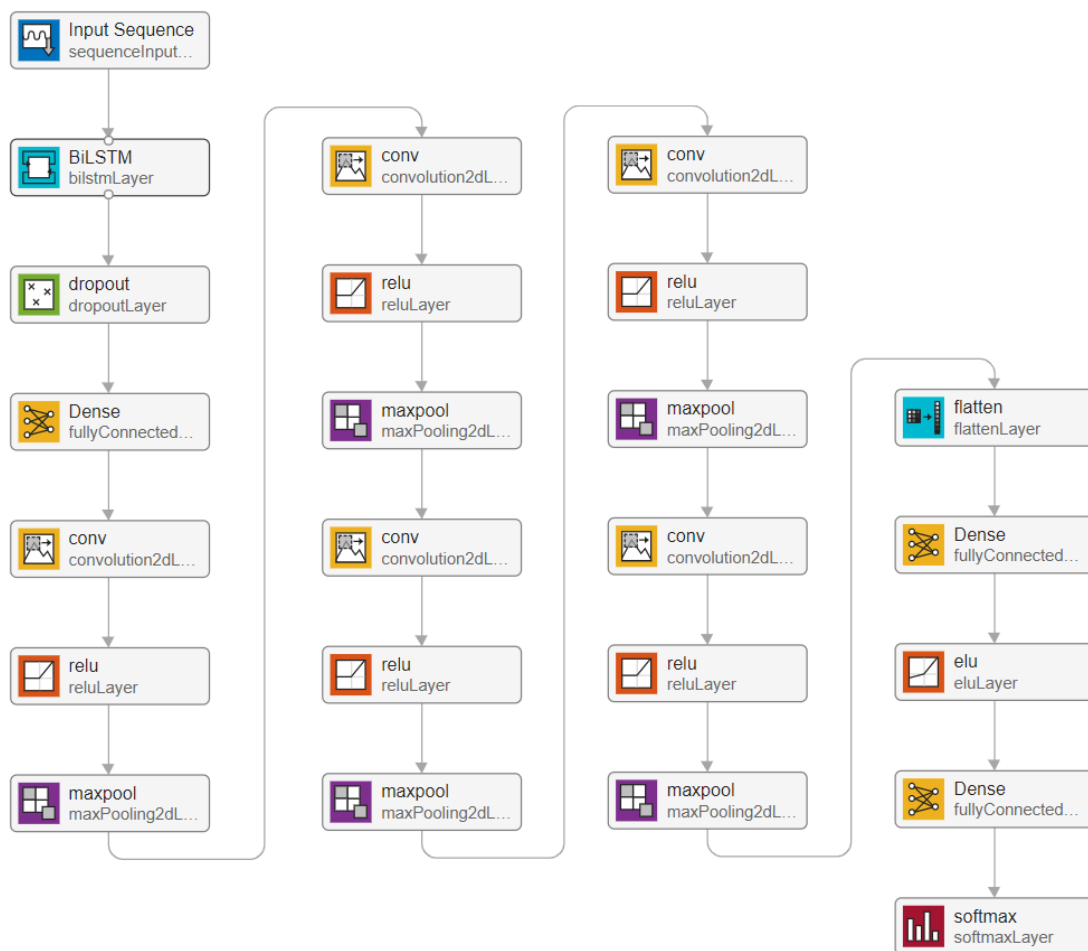


Figure 3.21 LSTM model.

With the LSTM model, it can achieve 97% accuracy in the training dataset and 91.28% in the validating dataset. It ran in a total of 36 epochs (8 hours). Figure 3.22 and Figure 3.23 shows the log of training and testing accuracy.

In conclusion, the training accuracy of CNN model and LSTM model is nearly equal, but the validating accuracy of CNN model is less than LSTM model. In addition, the validation loss of CNN method is much more higher; hence, we will apply LSTM to the intelligent radio spectrum monitoring system.

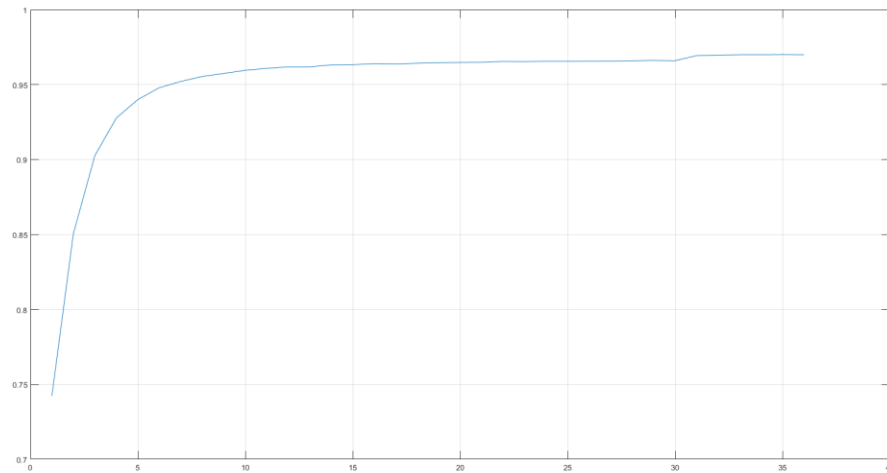


Figure 3.22 LSTM model's training accuracy log.

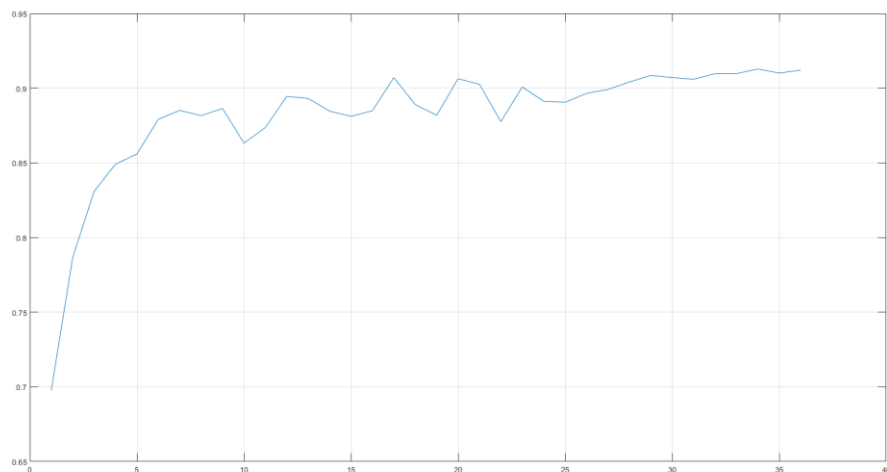


Figure 3.23 LSTM model's validating accuracy log.

3.5. Localizing source of signal

When the abnormal signal is detected, we have to figure out the source location of the signal and prevent it from continuing to broadcast. Several algorithms can be implemented to detect the signal, such as Time of Arrival (TOA), Time Difference of Arrival (TDOA), Angle of Arrival (AOA), Received Signal Strength Indicator (RSSI), Two-Way Ranging (TWR),... Because the source will broadcast continuously, we cannot identify the time to start transmitting a package from the source. Therefore,

the time-on-air, or distance of arrival, is very hard to determine. The angle of arrival technique has the same situation. The received signal strength indicator technique is an easier way to determine the distance of arrival. However, the reliability and stability of this technique are not high because it depends too much on the environment. The Two-Way Ranging is an asynchronous technique that does not need to use synchronization in time, which is very difficult to handle. In contrast, the TWR needs communication between the target and anchor, which is not applicable because we can only receive the package from the target. Finally, the Time Difference of Arrival would calculate the difference in time where the same package arrived, and then a hyperbola of possible locations would be drawn. This technique's only mission is to locate the same packages between a pair of sensors and the timestamp of arrival. In addition, this technique is a synchronization one. Hence, we need to handle this complex problem as well. In conclusion, we will use TDOA as our proposed technique to find out the location of the interference signal.

Firstly, we need to define the same package between a pair of sensors and the difference in time of arrival. It will cost more time and effort if we demodulate the signal to the original data. In addition, we have to know the signal's modulation type, which means we have to implement one more layer to detect it. If we do not demodulate it, there will be only raw IQ data to be processed. Thanks to the statistics, if we calculate the cross-correlation between two discrete time series data, we will know the lagging/delay between these two series. That means, in case the series start at the same time, we would know the time difference of arrival between a pair of sensors.

For example:

$$\begin{aligned} a &= [-7, \quad 6, \quad -4, \quad 1, \quad -7, \quad 2, \quad -5, \quad 3, \quad 4, \quad 5] \\ b &= [0, \quad -1, \quad -7, \quad 6, \quad -4, \quad 1, \quad -7, \quad 2, \quad -5, \quad 3] \end{aligned}$$

Hence, the normalized cross-correlation between a and b is:

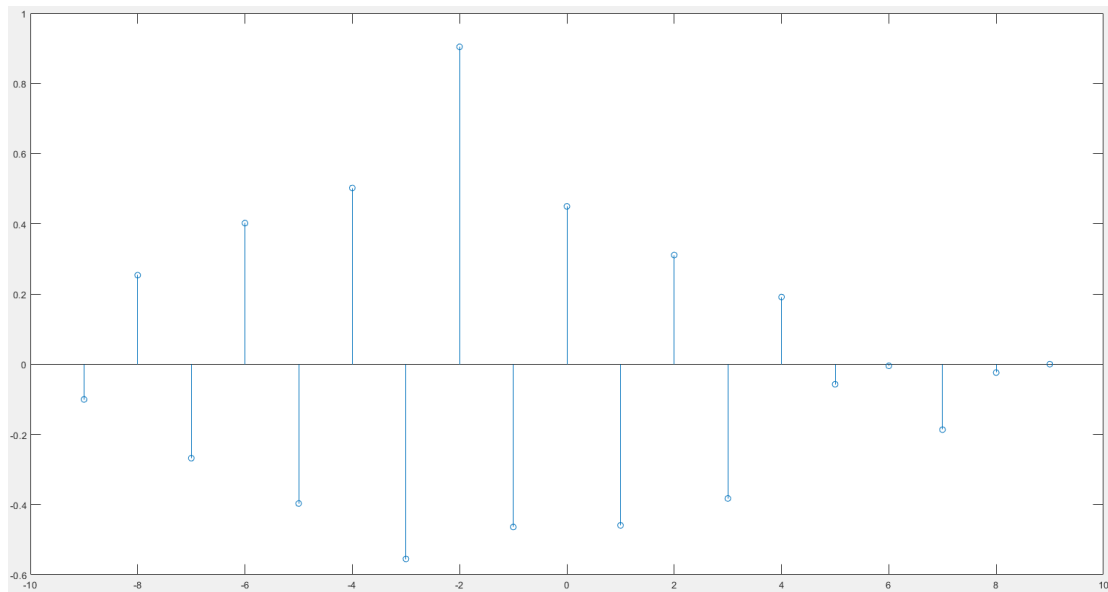


Figure 3.24 An example of cross-correlation between two discrete time series data.

As we can see, the maximum value lies on $x = -2$; therefore, a lag -2 steps when comparing with b , or we can say that a is earlier than b 2 steps. Try with a and $c = b + \text{bias}$, and the result would not change.

Hence, we can conclude that the signal comes to receiver a and receiver b has the Time Different of Arrival equal to -2.

Then the principle of TDOA is demonstrated as Figure 3.25:

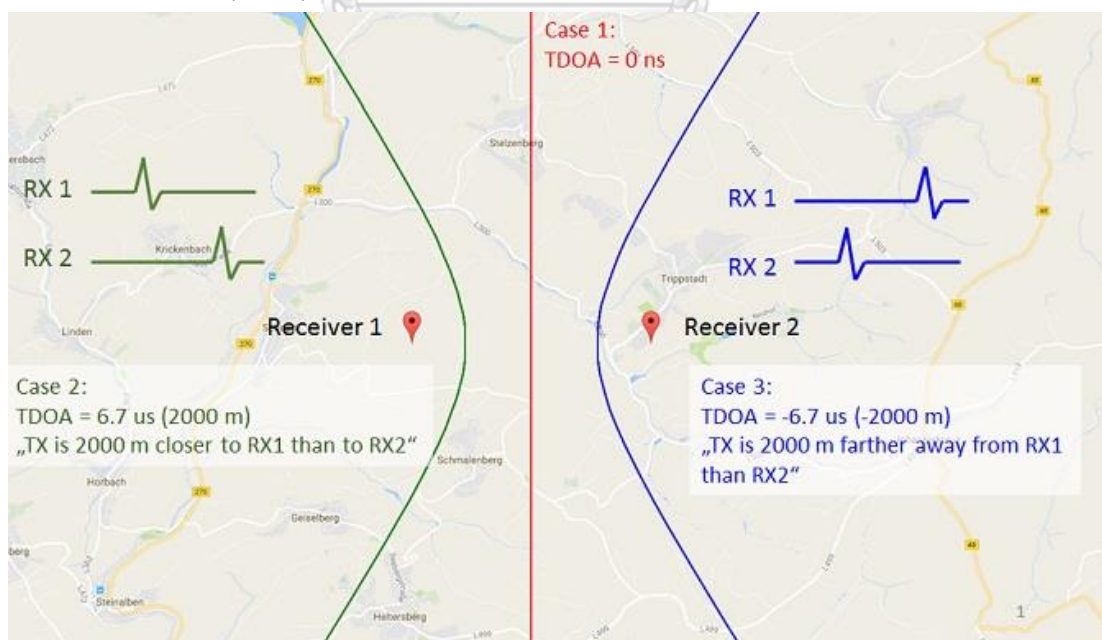


Figure 3.25 The possible location of TX when the TDOA is determined [13].

Secondly, the start time between a pair of sensors needs to be synchronized, which means both sensors must start to measure the signal at the same time. This is very hard for us to handle. Another transmitter signal source would be used as a reference to solve this issue. In detail, with the known location receivers and one known location and transmitted frequency transmitter, we can use the equation (3.11) and (3.12) to calibrate the synchronization. We assume that the difference in the distance from the reference transmitter to receivers 1 and 2 is dl (positive means receiver-1 is farther than receiver-2 and vice versa), the delay in the start is dt , and the light speed is c . Hence, we got:

$$\begin{aligned} k_{ref_async} &= k_{ref_sync} + dt \\ \Rightarrow dt &= k_{ref_async} - k_{ref_sync} \end{aligned} \quad (3.11)$$

$$\begin{aligned} \Rightarrow k_{target_sync} &= k_{target_async} - dt \\ &= k_{target_async} - k_{ref_async} + k_{ref_sync} \end{aligned} \quad (3.12)$$

Where $k_{X_sync} = \frac{dl}{c}$ is the delay received X time series without delay in start time, k_{X_async} is the delay received X time series with the delay in start time, and X is *ref* (reference) or *target*.

The RTLSDR will resample the signal at 2,048,000 samples per second. Therefore, we can use the difference in samples to measure the time. In this case, the error would be ± 0.5 sample, equal to ± 75 meters. Assume that we will retrieve 2.048 million samples. Therefore the k_{ref_async} and k_{target_sync} must not be larger than 1 second. As a consequence, $dt \leq 1$; the equal happens when $dl = 0$. Let's say in a different way, with known dt , the number of samples to be retrieved is $samples > dt * 2.048 * 10^6$. Let $samples = 2 * dt * 2.048 * 10^6$:

Through the two approaches above, we solve the synchronization and time difference issue in the TDOA technique.

A TDOA system needs at least three anchors to determine the target, and the distance between a pair of sensors must be high enough (we choose at least 2km faraway each other). Because the three sensors were placed in three different locations, they would be in three different networks. Therefore, we need a communication technique that can control all three sensors. Therefore, the VPN

protocol described in the previous section may help. The sensor and all the sensors will now be in the same network and subnet. Hence, the SSH connection protocol is a good method to control the sensors to measure the data at the time.

As mentioned above, we need to measure the reference signal and target signal. In order to increase the reliability and stability of the system, there should be no gap between the two measurements, which means we have to measure the two signals *seamlessly*. If not, we need to minimize the lost samples when changing the frequency. To fulfill this, we need a library named "[librtlsdr-2freq](#)"; this library will enable seamless switching of frequency during the reception. To reduce the error of lost samples when changing frequency, we would measure the reference signal again and then average the delay. Hence,

$$\begin{aligned} k_{target_sync} &= k_{target_async} - dt \\ &= k_{target_async} - k_{ref_async_average} + k_{ref_sync} \end{aligned} \quad (3.13)$$

Finally, the order of measurement is Ref -> Target -> Ref. When we finish setting up the Server and Clients (Anchor/Receiver), the procedure for TDOA begins. Figure 3.26 is the process for the whole function.

- The server commands the sensor to measure the data and transfer them back.
- Synchronization: calculating the cross-correlation of the reference signal and computing the measured delay when changing the frequency.
- Determining TDOA: calculating the delay of the signal between two receivers with the measured delay above.
- Generating the TDOA hyperbola using geometry.
- Using optimization method (gradient descent) to compute an optimal result.

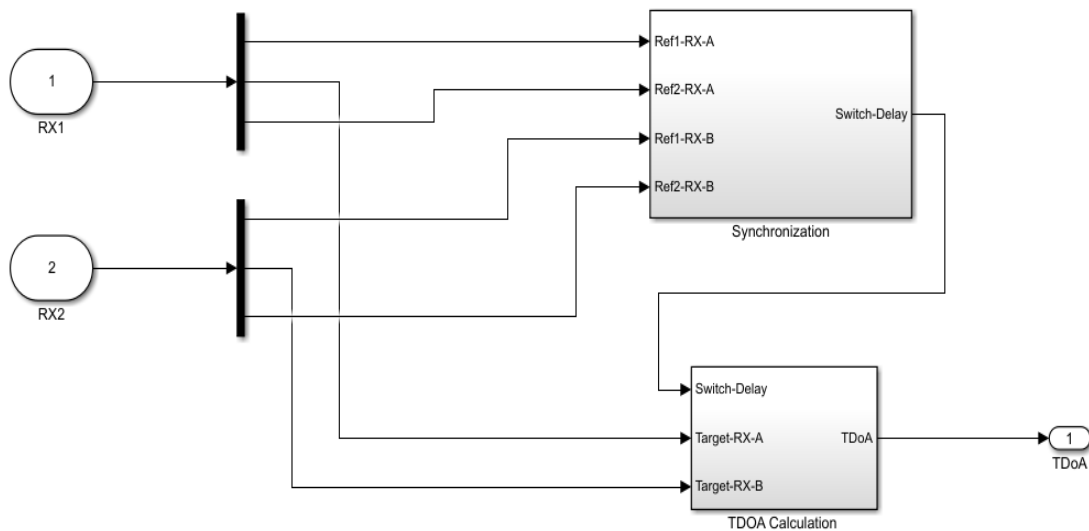


Figure 3.26 Signal processing for the calculation of a time-difference-of-arrival value for two receivers.

```

echo "-----"
if [ $# != 9 ]
then
    echo "parameters missing"
    exit 1
fi

ip1=$1
ip2=$2
ip3=$3
freq1=$4
freq2=$5
gain=$6
num_samples=$7
dev=$8
location=$9

echo "Specified parameters: reference frequency:" $freq1 ", measure frequency:"
$freq2 ", samples_per_freq:" $num_samples
echo "-----"
echo "Login to PI Radios and capture data simultaneously"
ssh pi@$ip1 "echo LD_LIBRARY_PATH=/home/pi/librtlsdr-2freq/build/src;
/home/pi/librtlsdr-2freq/build/src/rtl_sdr -f $freq1 -h $freq2 -n $num_samples -g
$gain -d $dev -s 2.048e6 1_raspi.dat" &\
ssh pi@$ip2 "echo LD_LIBRARY_PATH=/home/pi/librtlsdr-2freq/build/src;
/home/pi/librtlsdr-2freq/build/src/rtl_sdr -f $freq1 -h $freq2 -n $num_samples -g
$gain -d $dev -s 2.048e6 2_raspi.dat" &\
ssh pi@$ip3 "echo LD_LIBRARY_PATH=/home/pi/librtlsdr-2freq/build/src;
/home/pi/librtlsdr-2freq/build/src/rtl_sdr -f $freq1 -h $freq2 -n $num_samples -g
$gain -d $dev -s 2.048e6 3_raspi.dat"

echo "Copy received data to the master"
scp pi@$ip1:/home/pi/1_raspi.dat $location/1_raspi.dat &\
scp pi@$ip2:/home/pi/2_raspi.dat $location/2_raspi.dat &\
scp pi@$ip3:/home/pi/3_raspi.dat $location/3_raspi.dat

echo "done"

```

Code 3.3 Script to command and retrieve data simultaneously.

Code 3.3 is the script to command the sensors to measure the data and send them back.

Because the clients connect to the internet through 4G, the ping of each client to the server is different. In addition, the time to set up the SSH connection should be considered. The overall time would vary between 0.5-1.5s. Therefore, we can assume that $dt < 1.5$. Let $dt = 1.5$; we should retrieve 7.2 million samples.



CHAPTER 4. RESULT AND DISCUSSION

4.1. Web page for monitoring the spectrum

An indispensable vital part of the intelligent spectrum monitoring system is the graphical user interface (GUI). It is responsible for displaying the information of each sensor, visualizing the radio spectrum, summarizing the detail of the detected signals, and connecting to the backend for localization function. The GUI can be designed on several platforms, such as smartphones, computers, or web pages. A GUI based on the web application is good for more convenience and easier maintenance, update, and management.

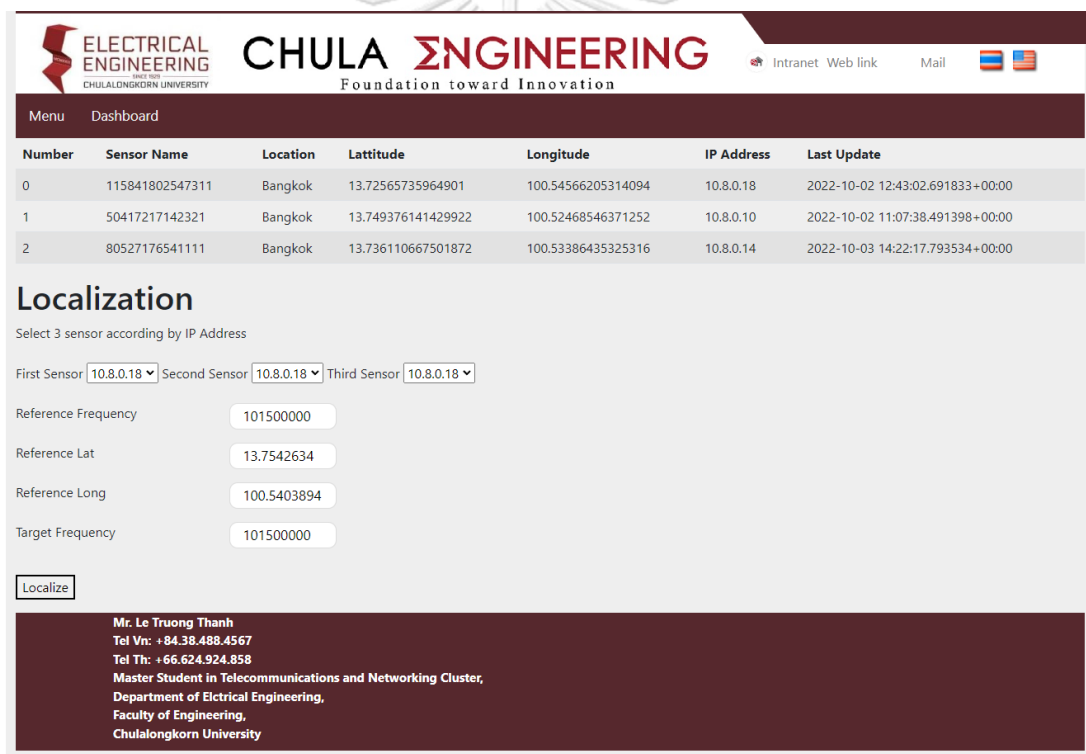


Figure 4.1 The intelligent radio spectrum monitoring dashboard.

The general web application contains two components, the backend, and the front end. The intelligent radio spectrum monitoring uses the Flask framework on Python, which is more familiar to non-IT researchers than the other languages. NodeJS, HTML, and CSS are the primary language for the frontend GUI. The GUI was built with several basic functions in order to provide just enough features of the

system. Figure 4.1 gives an overview of the dashboard when entering the GUI web page.

In general, the dashboard contains four sections. The header section is the logo of the Department of Electrical Engineering. The second section is a table that provides details information about each sensor. From this table, we can know the ID, assigned IP, location, and the last update event. When clicking on each row, the web directs us to the spectrum monitoring page, as shown in Figure 4.2 and Figure 4.3. The location on a map, the last spectrum image, and the details of each detected signal that corresponds to the current sensor will be displayed on this page. The third section is for the localization function. The footer section is the contact details.



Figure 4.2 Spectrum monitoring page.

In the localization section, there are several fields that the user has to input: three IPs corresponding to three sensors for running localizing, the details of the

reference station (frequency and location), and the target frequency to be determined the position. While the localization algorithm is running, the page displays the log below the third section. After that, a map with an estimated position will appear between the third section and the log section, like in Figure 4.4.

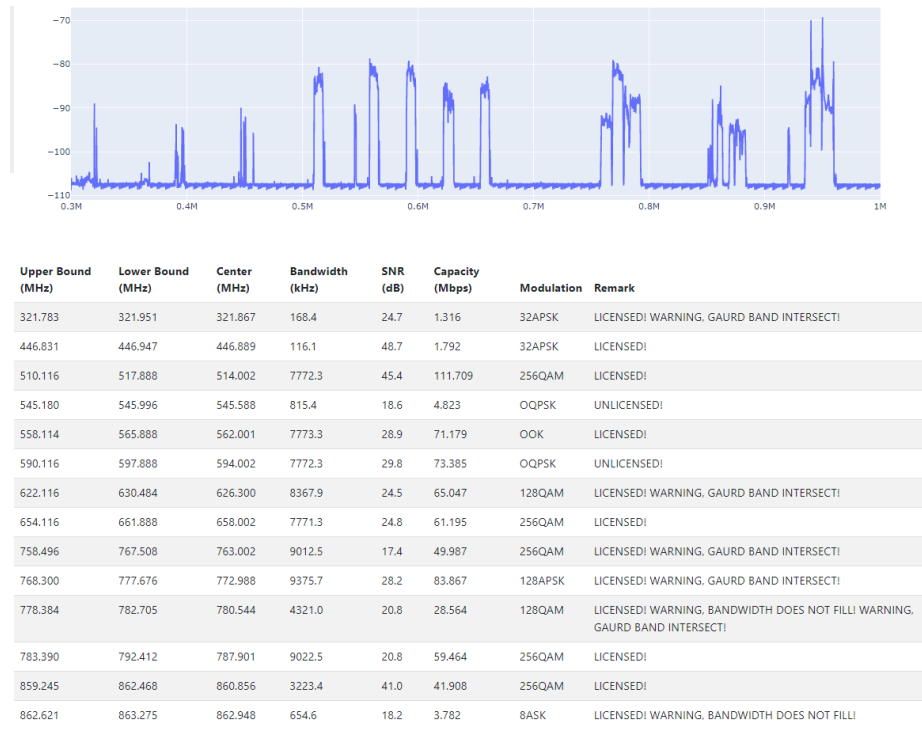


Figure 4.3 Spectrum monitoring page.

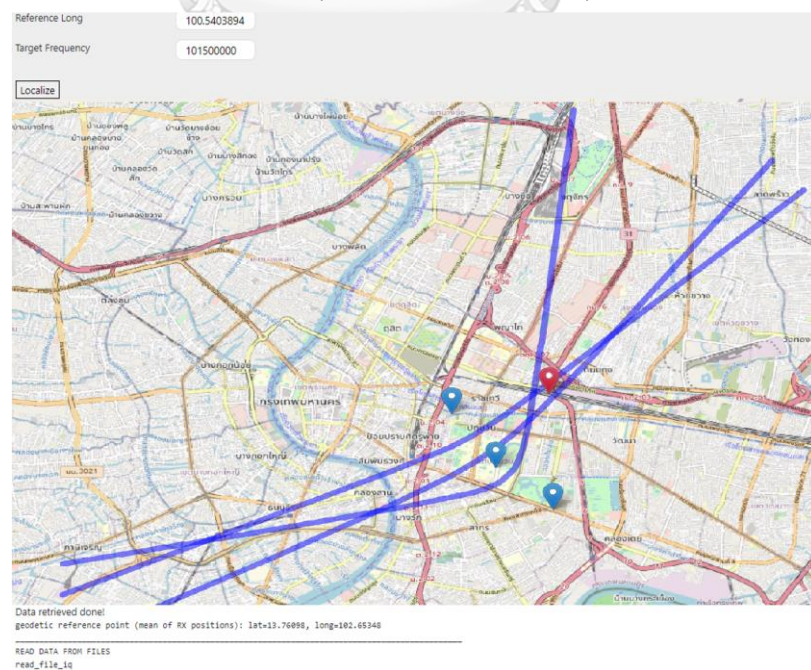


Figure 4.4 Localization result shows on map.

4.2. Analyzing the characteristic extraction result

As described in the previous section, the LSTM model achieves higher validating accuracy and lower validating loss. Hence, the system will use the LSTM model to evaluate and implement in real-time processing. Figure 4.5 gives an overview of the confusion matrix when applying the LSTM model to validate data. The confusion matrix is a particular performance measurement for a machine-learning classification problem. It forms a table/matrix with the row elements as actual labels and the column elements as predicted labels.

There are 17 classes that have more than 99% accuracy. The model usually gets confused between AM-DSB-SC and AM-DSB-WC; and between AM-SSB-WC and AM-SSB-SC. In addition, the model AM-SSB-WC and AM-SSB-SC class can be predicted as 8ASK and OOK. Finally, the 8ASK class has a few percent that is predicted as 4ASK and OOK, and in the opposite situation, the 4ASK and OOK class are usually predicted as 8ASK. Compared to the result in the paper [1], the performance of the proposed model is higher, 91.28% vs. 87.4%. In detail, the FM class in [1] is usually confused with the AM-DSB, while the proposed model has nearly 100% accuracy. Compared to the LSTM model in the paper [3], The overall accuracy of the proposed model is higher. The QAM16 and QAM64 in [3] are confused with each other. In addition, both AM-DSB types in the proposed LSTM model are not confused with WBFM, while the [3] are. Furthermore, in each paper, the authors only run the model on the specific SNR value, not combine all of them. Therefore, the proposed model is more general for all cases.

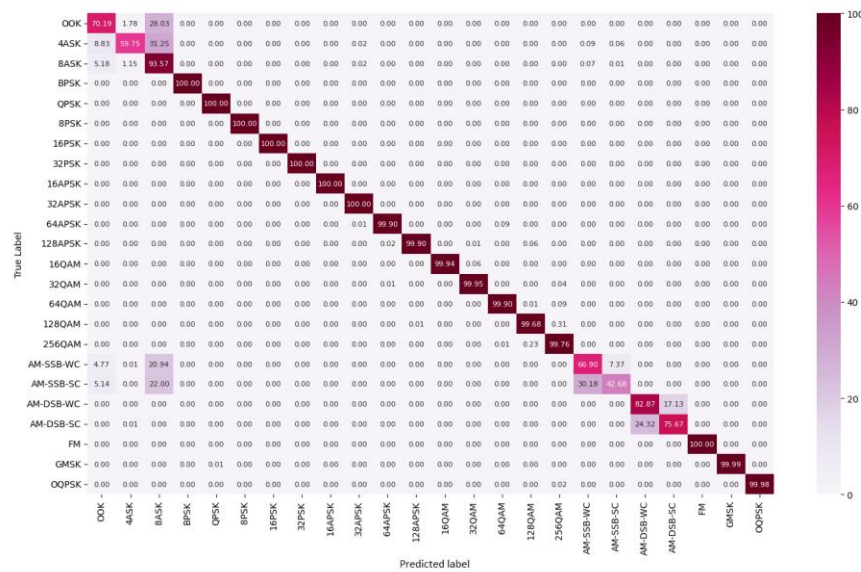


Figure 4.5 Confusion matrix on validate dataset.

From the confusion matrix, there are four values has to be concerned:

- True Positive: the number of cases where the actual value is positive and the predicted value is positive.
- False Positive: the number of cases where the actual value is negative and the predicted value is positive.
- False Negative: the number of cases where the actual value is positive and the predicted value is negative.
- True Negative: the number of cases where the actual value is negative and the predicted value is negative.

The term "positive" and "negative" above are, respectively, the currently considered class and the rest. In the multi-class (more than two) classification problem, the False Negative value is the sum of all values in the same row except the True Positive value, and the False Positive is the sum of all values in the same column except the True Positive value. The True Negative is the sum of all remaining values. In addition, the confusion matrix values can be represented as a ratio. When the model results in the four values above, we can evaluate the model by two criteria: Precision and Recall metric. The high Precision value means the accuracy of the found case is high, and the High Recall value means the missing rate when

finding the positive case is low. Finally, the F1 score is a method that combines both Precision and Recall metrics. In multi-class classification problems, the overall accuracy of the model is usually the average of the F1 score of each class. Table 4.1 describes all the mentioned metrics of the LSTM model.

Table 4.1 Evaluation on validate dataset.

	Precision	Recall	F1-Score	Samples
OOK	0.75	0.70	0.72	8191
4ASK	0.95	0.60	0.73	8191
8ASK	0.48	0.94	0.63	8191
BPSK	1.00	1.00	1.00	8191
QPSK	1.00	1.00	1.00	8191
8PSK	1.00	1.00	1.00	8191
16PSK	1.00	1.00	1.00	8191
32APSK	1.00	1.00	1.00	8191
64APSK	1.00	1.00	1.00	8191
128APSK	1.00	1.00	1.00	8191
16QAM	1.00	1.00	1.00	8191
32QAM	1.00	1.00	1.00	8191
64QAM	1.00	1.00	1.00	8191
128QAM	1.00	1.00	1.00	8191
256QAM	1.00	1.00	1.00	8191
AM-SSB-WC	0.69	0.67	0.68	8191
AM-SSB-SC	0.85	0.43	0.57	8191
AM-DSB-WC	0.77	0.83	0.80	8191
AM-DSB-SC	0.82	0.76	0.78	8191
FM	1.00	1.00	1.00	8191
GMSK	1.00	1.00	1.00	8191
OQPSK	1.00	1.00	1.00	8191
Accuracy			0.91	196584
Macro avg	0.93	0.91	0.91	196584

Weighted avg	0.93	0.91	0.91	196584
--------------	------	------	------	--------

Implementing the LSTM model to the IQ data of the detected signal, we have the result like Figure 4.6.

591.417	599.317	595.367	7900.4	10.0	26.100	256QAM
598.992	599.317	599.154	324.9	7.9	0.884	256QAM
622.117	622.499	622.308	382.2	7.2	0.960	256QAM
622.507	631.317	626.912	8810.0	12.7	36.152	128QAM
647.295	647.298	647.297	3.0	3.4	0.005	256QAM
654.430	654.537	654.483	107.1	4.9	0.207	8ASK
654.886	654.927	654.906	40.7	4.8	0.078	256QAM
654.555	663.317	658.936	8762.1	15.7	43.839	256QAM

Figure 4.6 Result of the implemented model.

As we know that the DVB-T2 support the QPSK, 16QAM, 64QAM, 256QAM, the model can detect that the signal from the station 1.00 in Bangkok as 256QAM. Hence, in conclusion, the model can work well.

4.3. Analyzing the signal's source localization

The cross-correlation method can be perfectly applicable to digital time series data. The synchronization procedure is the most critical step in the TDOA algorithm because if there is no synchronization between the devices, all the following formulas will result in the wrong value. So, a digital broadcaster as a reference is a pre-requisite. In addition, the measured delay calculation also needs a good cross-correlation value to enhance the accuracy. Hence, the system could find the optimal position with high precision if the reference and target signal are digital broadcasts. There are several signal types in the range that the RTL-SDR can measure. The most popular types that can be considered are Analog FM Radio, Digital FM Radio (Digital Audio Broadcasting - DAB/DAB+), Digital Video Broadcasting (DVB), ISM, and LoRa. Furthermore, the ISM and LoRa signals have a low transmitting range that could not cover all three sensors in most situations, and the Analog FM Radio signal could lead to a low accuracy case. Unfortunately, there are no Digital Audio Broadcasting stations in Bangkok. Hence, Digital Video Broadcasting will be used.

Table 4.2 and Table 4.3 describe the detail of the Digital Video Broadcasting station in Bangkok:

Table 4.2 DVB-T2 stations in Bangkok.

Index	Lat	Long	Height (m)	#1	#2	#3	#4	#5	Power (kW)
1.00	100.540270	13.754300	328	26	36	40	44	32	100.0
1.01	100.949558	13.190653	40	45	48	29	25	43	1.00
1.02	100.866450	12.921333	60	45	48	29	25	43	1.00
1.03	99.613515	13.627185	80	33	37	41	30	27	5.00
1.04	101.441810	13.291820	70	45	48	29	25	43	5.00
1.05	99.994444	13.385428	70	26	36	40	44	32	0.50
1.06	101.625020	13.473830	55	24	42	66	38	34	2.00

Table 4.3 Channels' descriptions.

Channel			Center Frequency
	Lower bound	Upper bound	
21	470	478	474
22	478	486	482
23	486	494	490
24	494	502	498
25	502	510	506
26	510	518	514
27	518	526	522
28	526	534	530
29	534	542	538
30	542	550	546
31	550	558	554
32	558	566	562
33	566	574	570
34	574	582	578
35	582	590	586

36	590	598	594
37	598	606	602
38	606	614	610
39	614	622	618
40	622	630	626
41	630	638	634
42	638	646	642
43	646	654	650
44	654	662	658
45	662	670	666
46	670	678	674
47	678	686	682
48	686	694	690

Because the station's transmit power from 1.01 to 1.06 is too low, these stations could not cover all setting up regions. Therefore, we would use station 1.00 as the reference station. After a few surveys on the spectrum from 450 MHz to 700 Mhz, we found that in our setup region, the sensor could only receive the signal of channel 34 (574-582 MHz). Table 4.4 describes the setup locations. Figure 4.8, 4.9, 4.10, 4.11, and 4.12 give a view of the spectrum in each location.

Table 4.4 Setup locations.

Station Name	Latitude	Longitude
Lotus Rama I	13.749389149553245	100.52468698495306
Engineering Building 4	13.736110644939403	100.53393449613192
Lumphini Park View	13.725036661833437	100.54651079537058

Because only one station can cover our setup place, we can not choose two stations with known locations to estimate the stability, reliability, and error of the reference signal. Hence, we could only use the same station as a reference and target to estimate the condition.

Now, we start to measure the signal several times and set it as the input of our system. The error is shown Table 4.5.

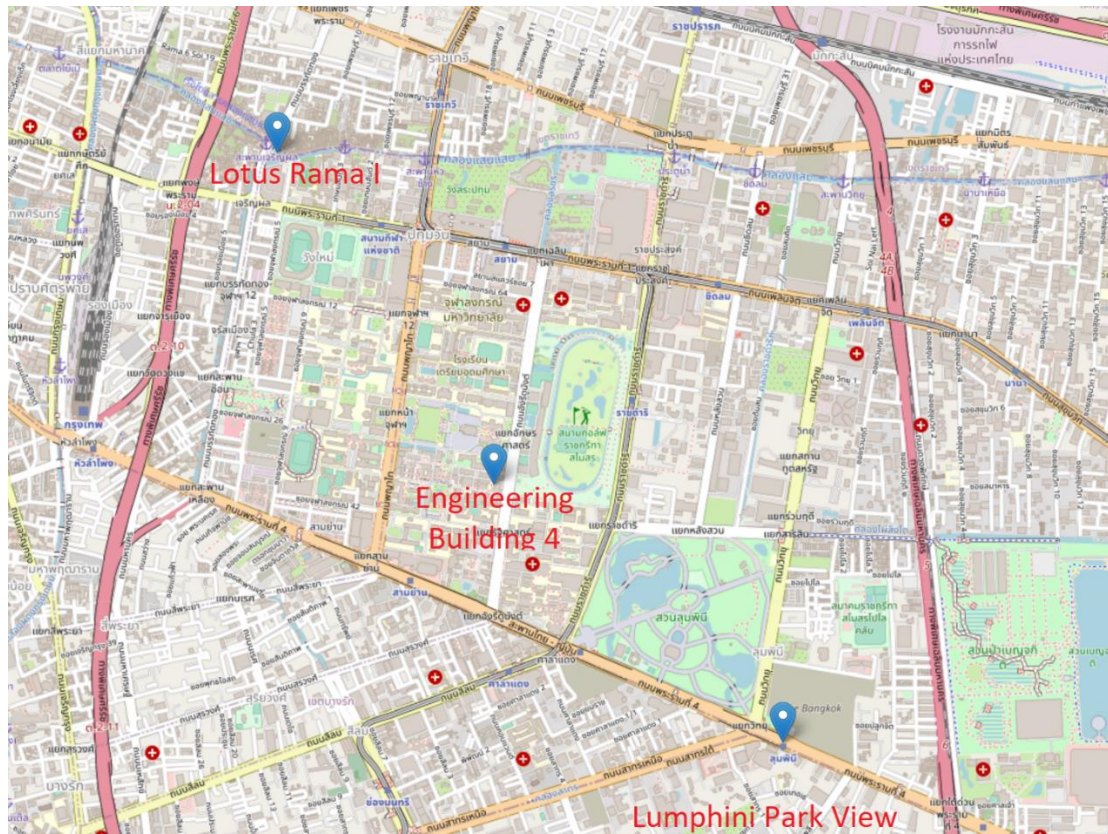


Figure 4.7 Sensors' setup locations.

Table 4.5 The error of multiple running times.

	Sample Rate	Latitude	Longitude	Error
1	2048000	13.75295	100.539	210.7344
2		13.7496	100.5407	520.4556
3		13.75132	100.5405	328.427
4		13.75033	100.5385	482.1158
5		13.74891	100.5392	609.2635
6	2480000	13.75353	100.5424	234.7248
7		13.75338	100.5403	99.21007
8		13.74943	100.5372	640.6192
9		13.74963	100.5385	557.1086
10	2880000	13.74846	100.5396	651.6951
11		13.75022	100.5441	604.3491

The average error of the sampling rate 2.048msps cases is 430m and the average error of the sampling rate 2.48msps is 450m. This is an average of more than 20 times in each case. The cases of 2.88msps mostly fail and are unstable (there is only one good case) because the RTLSDR could lose several samples with a 2.88msps sampling rate. One lost sample could lead to a considerable error. In addition, the case sample rate of 2.48msps is more stable than 2.88, but the error resolution is also higher (can reach 100m of error). The cases with a sample rate of 2.048msps have the highest error resolution and are the most stable.

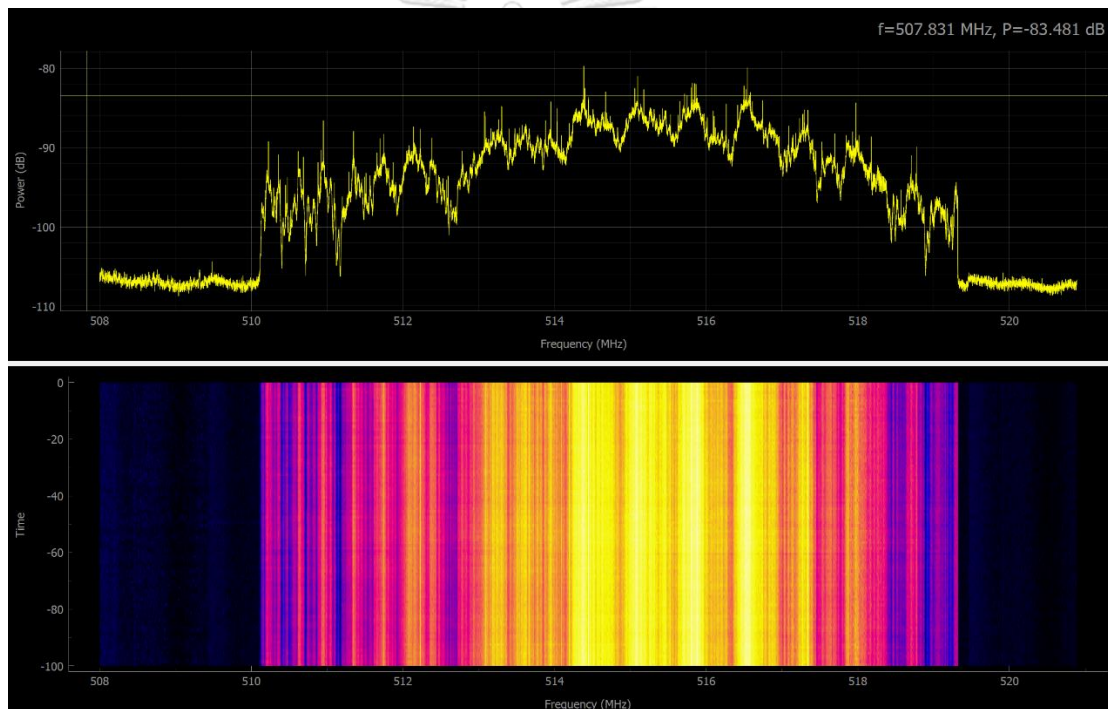


Figure 4.8 Spectrum over DVB-T channel 26.

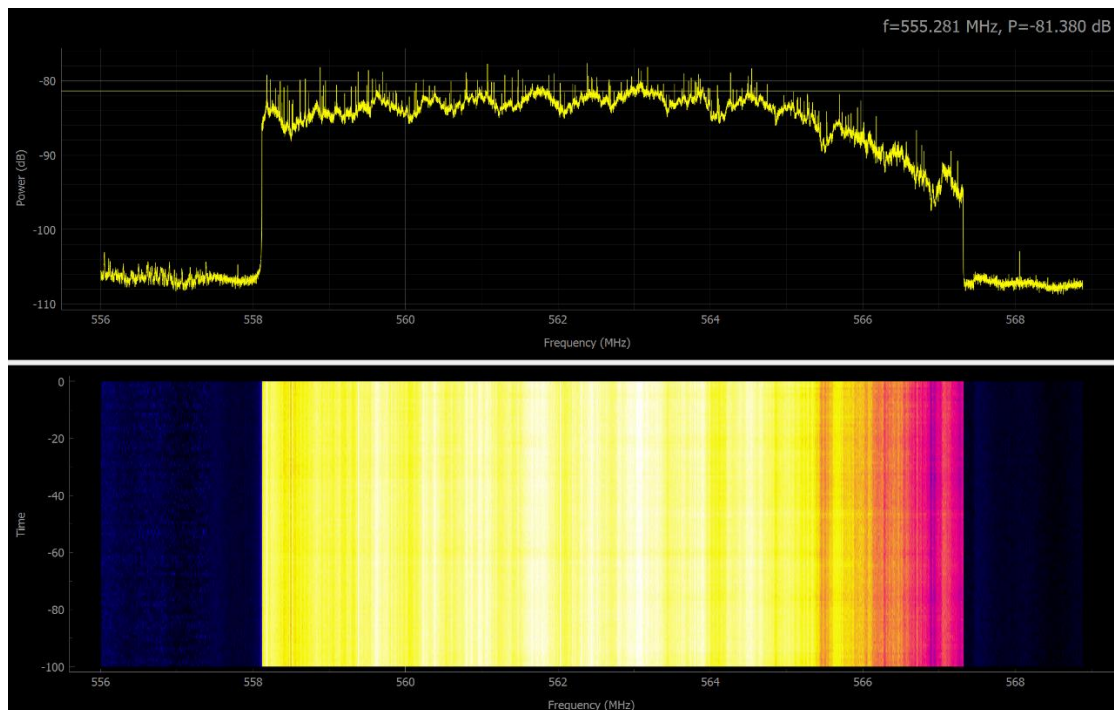


Figure 4.9 Spectrum over DVB-T channel 32.

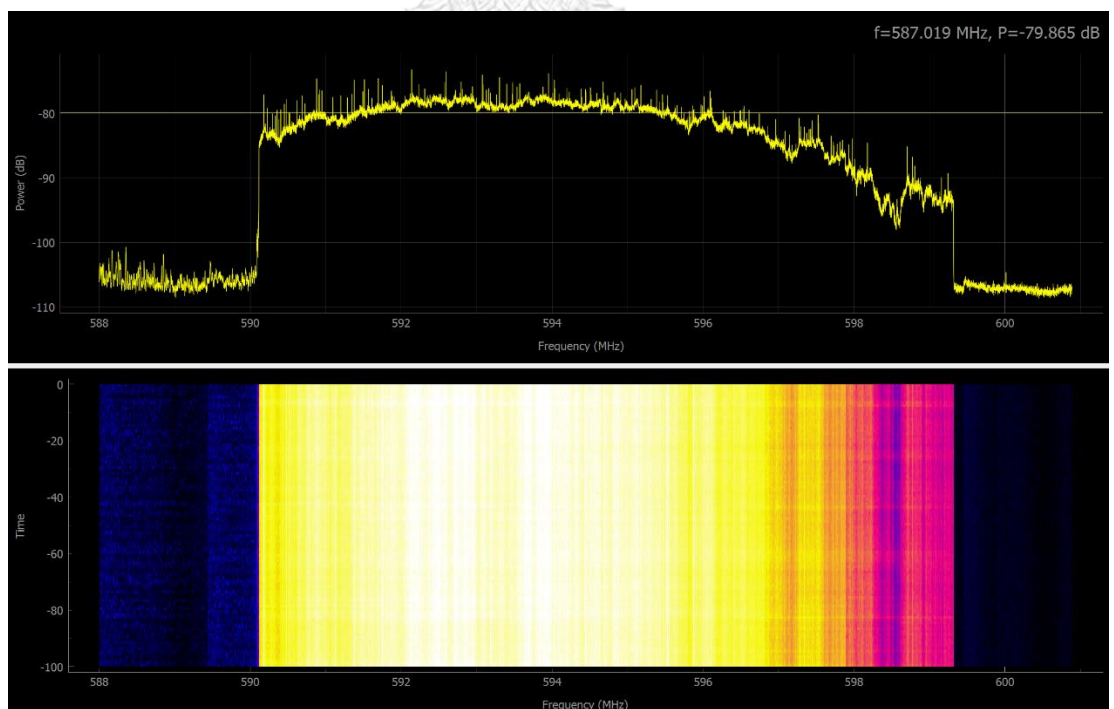
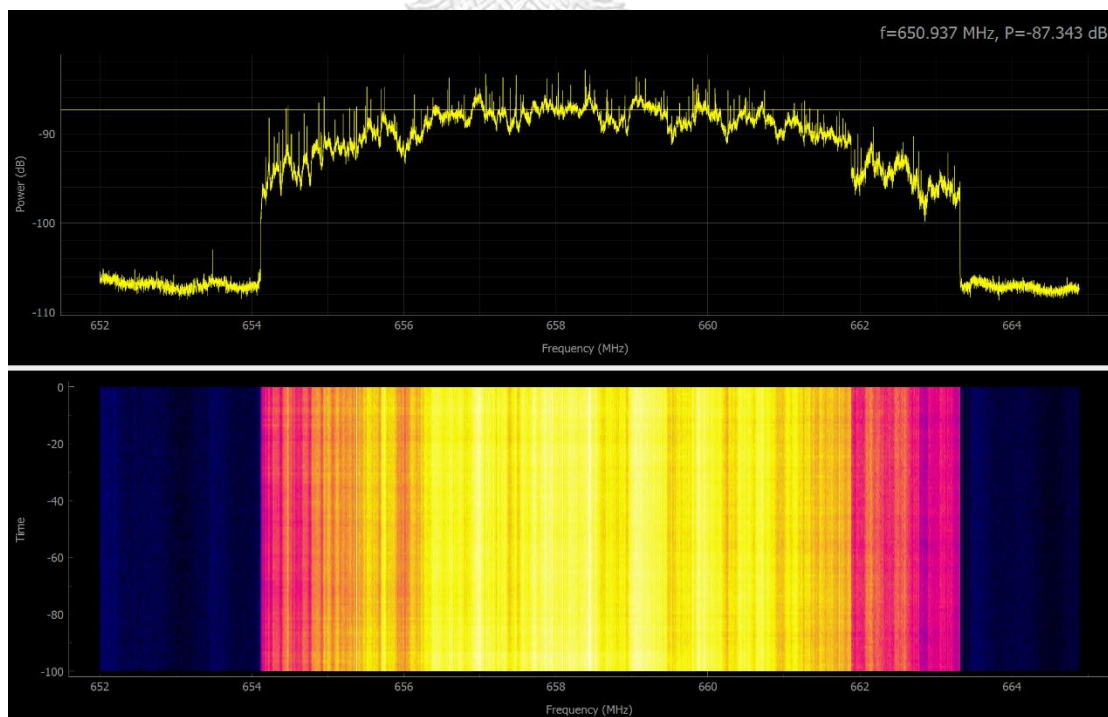
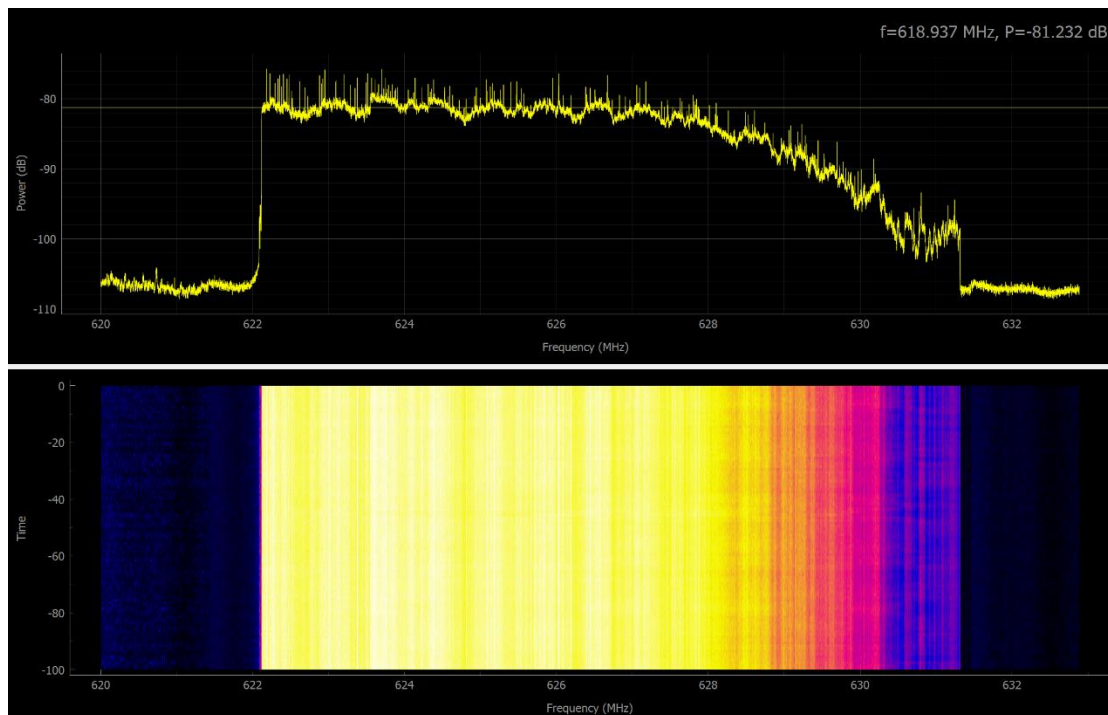


Figure 4.10 Spectrum over DVB-T channel 36.



CHAPTER 5. CONCLUSION

The intelligent radio spectrum monitoring system was successfully built. In this system, there are several skills needed, such as handling embedded systems, familiarity with IoT architecture and communication, designing backend server and frontend GUI, training and testing deep learning models, and mathematic skills with the localization function. In the thesis proposal, there are two objectives:

- Design an autonomous system that detects active frequency signals with low-cost sensors and deep learning.
- Analyze the active signal in terms of center frequency and bandwidth and then determine the condition of the wireless signal spectrum.

The first objective was fulfilled with the RTL-SDR. Figure 5.1 gives a result that the second objective was done.

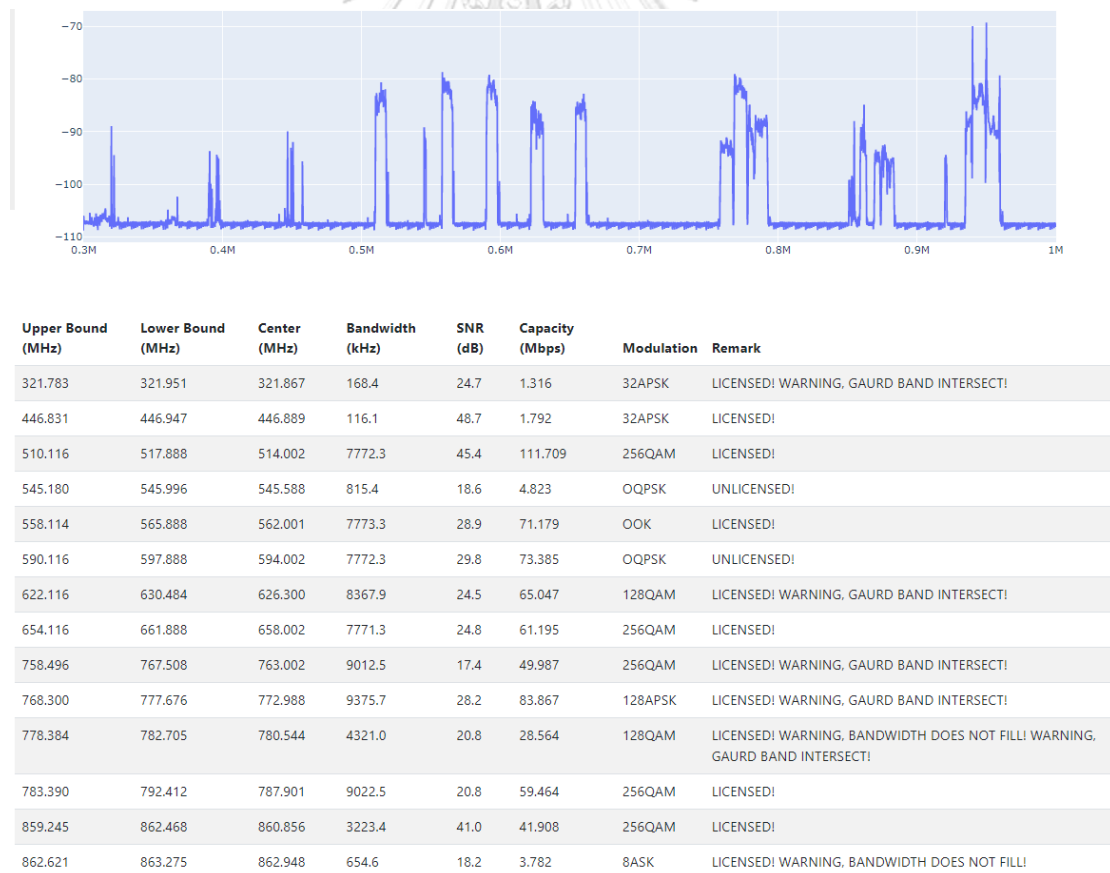


Figure 5.1 Signals' information.

In addition, the Deep Learning architecture was implemented in the system and was very effective. The accuracy on the validated dataset can reach up to 92%, which is higher than 87.4% in [1]. The localization function works very well with the average error is 450m.

With this system, the shortcomings of ordinary supervisors, such as distraction, illness, and personal decisions, will be compensated. The system could automatically self-operate continuously; therefore, the radio spectrum's current information could be updated immediately and stored as historical data for the supervisors to report and revise. In addition, the system could extract several characteristics, for example, SNR, center frequency, capacity, and especially modulation type of the signal. Signal modulation recognition is the most challenging task in the system and is very hard for a human to do it. Furthermore, the system can localize the signal transmission source with reasonable accuracy if it is a digital broadcaster. In conclusion, the contribution of this research work is to show the possibility of our proposed method, i.e. using Deep Learning, to help human-being in radio spectrum monitoring with reasonable high reliability and accuracy.

The system still has several limitations that need to be investigated in the future. First, the system needs a wide bandwidth of internet to transmit the data. In detail, with the range of frequency from 300MHz to 1000MHz and the resolution of 1kHz per each bin of FFT, it needs about 6MB for each message. This could lead to a lot of data storage and latency in transmitting. Moreover, when classifying the modulation type, the system needs 1024 samples per represented signal; hence, if there is a huge number of signals, the modulation classification task would take a long time to handle (retrieving and classification). Therefore, a better compression technique is required to reduce the storage and handling time. Second, the training dataset for the modulation classification task is for the USRP device, which has an ADC resolution of 14 bits, while the RTL-SDR has only 8 bits and is a low-cost device. As a consequence, there may be a reduction in accuracy. In addition, the overall accuracy when applying to RTL-SDR device cannot be examined because there is no information on the current signals in Bangkok. Hence, if we know the signal characteristics in Bangkok, we can build an online learning model to enhance the

accuracy and avoid the impacts of resolution reduction. Finally, the localization module is now using the Gradient Descent method to find out the optimal position of the target. The limitation of this technique is that it needs a good initial position, and the found point could be an optimal local one. For that reason, a new optimization technique could be implemented to find an optimal global position and reduce the operating time.



REFERENCES



จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

1. O'Shea, T. J., Corgan, J., & Clancy, T. C. (2016). Convolutional radio modulation recognition networks. International conference on engineering applications of neural networks.
2. Kulin, M., Kazaz, T., Moerman, I., & De Poorter, E. (2018). End-to-end learning from spectrum data: A deep learning approach for wireless signal identification in spectrum monitoring applications. *IEEE access*, 6, 18484-18501.
3. Rajendran, S., Meert, W., Giustiniano, D., Lenders, V., & Pollin, S. (2018). Deep learning models for wireless signal classification with distributed low-cost spectrum sensors. *IEEE Transactions on Cognitive Communications and Networking*, 4(3), 433-445.
4. Scholl, S. (2019). Classification of radio signals and hf transmission modes with deep learning. *arXiv preprint arXiv:1906.04459*.
5. Perenda, E., Rajendran, S., & Pollin, S. (2019). *Automatic Modulation Classification Using Parallel Fusion of Convolutional Neural Networks*
6. Rajendran, S., Lenders, V., Meert, W., & Pollin, S. (2019). Crowdsourced wireless spectrum anomaly detection. *IEEE Transactions on Cognitive Communications and Networking*, 6(2), 694-703.
7. Li, R., & Li, J. (2014). A novel clouds based spectrum monitoring approach for future monitoring network. The 2014 2nd International Conference on Systems and Informatics (ICSAI 2014).
8. Calvo-Palomino, R., Cordobés, H., Engel, M., Fuchs, M., Jain, P., Liechti, M., Rajendran, S., Schäfer, M., Van den Bergh, B., & Pollin, S. (2020). Electrosense+: Crowdsourcing radio spectrum decoding using IoT receivers. *Computer Networks*, 174, 107231.
9. Selim, A., Paisana, F., Arokiam, J. A., Zhang, Y., Doyle, L., & DaSilva, L. A. (2017). Spectrum monitoring for radar bands using deep convolutional neural networks. GLOBECOM 2017-2017 IEEE Global Communications Conference.
10. Rajendran, S., Calvo-Palomino, R., Fuchs, M., Van den Bergh, B., Cordobés, H., Giustiniano, D., Pollin, S., & Lenders, V. (2017). Electrosense: Open and big spectrum data. *IEEE Communications Magazine*, 56(1), 210-217.

11. Audio Precision Inc. FFT Scaling for Noise. <https://www.ap.com/technical-library/fft-scaling-for-noise/>
12. Harris, F. J. (1978). On the use of windows for harmonic analysis with the discrete Fourier transform. *Proceedings of the IEEE*, 66(1), 51-83.
13. Scholl, S. (2017). Introduction and Experiments on Transmitter Localization with TDOA. *Software Defined Radio Academy, Friedrichshafen, Germany*.
14. Le Truong, T., Le, N. T., & Benjapolakul, W. (2022). An Application of Deep Learning YOLOv5 Framework to Intelligent Radio Spectrum Monitoring. *2022 37th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*



VITA

NAME	LE TRUONG THANH
DATE OF BIRTH	03 June 1998
PLACE OF BIRTH	Viet Nam
HOME ADDRESS	77 Le Hong Phong, Quang Tri Town, Quang Tri Province, Viet Nam
PUBLICATION	Le Truong, T., Le, N. T., & Benjapolakul, W. (2022). An Application of Deep Learning YOLOv5 Framework to Intelligent Radio Spectrum Monitoring. 2022 37th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)