Department of Design Engineering and Mathematics

Middlesex University

Activity Recognition via Wi-Fi Sensing

Student name: Tek Bahadur Pun

Student ID**: M00834324**

Supervisor: Dr Tuan Le

PDE3112 Major project

BEng Computer Systems Engineering

May 202

Abstract

Activity recognition through Wi-Fi sensing is being favoured over methods such as device-recognition or sensor recognition methods as it is non-intrusive. As there is an abundance of access points all around us to be used to extract Channel State Information (CSI) via publicly released data interactive framework which is capable of interpreting, processing, and visualising such as Nexmon and CSIkit to be used to analyse and plot data into graph from the extracted CSI. This allows user to be able to generate substantial amount of dataset based on CSI and classify them according to activities. In this paper, readily available hardware such as Raspberry Pi 4+ and 5 which are popular embedded boards are implemented to create a communication link via packet transferring and extracting the CSI from it. This involves the processes of gathering dataset for 7(Clapping, sitting, sitting left hand up, standing, standing up, standing right hand up and empty) activities which are performed over a span of 4 different days in a living environment. These data are then pre-processed to remove noise based on the amplitude power level of the signals. A Convolutional Neural Network (CNN) is then designed and implemented to classify and train a model to be able to predict which activity is being performed. Experiments are performed and recorded in a living environment to make it as realistic as possible. The experimental results yielded a result of 85% on average as accuracy on all 7 different activities.

Keywords: Channel State Information (CSI), Convolutional Neural Network (CNN)

Table of contents

1. Chapter 1 – Introduction

Wireless technology has revolutionized human activity detection by offering an accessible and non-intrusive means of tracking activity. Researchers have developed complex techniques that use readily available Wi-Fi devices and its infrastructure to detect human activity with efficiency. In recent years, using channel state information (CSI) from Wi-Fi signals has become an effective tool for distinguishing human activity. Wi-Fi signal propagation produced by human activity have been shown to have a disturbance on the Wi-Fi signal and studies have demonstrated the effectiveness of machine learning methods, such as Convolutional Neural Networks (CNNs), in obtaining valuable insights from unprocessed raw Channel State Information of Wi-Fi signal. Additionally, preprocessing methods like removing noise, normalization, smoothing and Principal Component Analysis (PCA) have allowed CSI data to be more accurate and reliable. [1] When combined with strong network interface controllers (NICs) and specialized tools such as Nexmon CSI-Tool, gathering large amounts of CSI data pertaining to human behaviours can be made easy. Using this data as the foundation, deep learning architectures such as Convolutional Neural Network (CNN), ResNet and Reoccurring Neural Network (RNN) can be used to create advanced recognition systems with high accuracy prediction. In this report, we are creating a system design that gathers CSI data from an access point, where 7 distinct actions are performed between the transmitter and receiver, and then use the collected data to train a CNN model. This CNN model will then be used to predict activity based only on the CSI data gathered and the CNN model that has been trained.

1.1. Aims

- Provide methods to collect and examine Channel State Information (CSI) data from Wi-Fi signals that are influenced by human activities.
- To extract relevant information from CSI data, such as human-caused variations in frequency, phase, and amplitude response, applying signal processing techniques to filter out unwanted noise.
- By applying machine learning and pattern recognition techniques, establish a robust framework for activity recognition based on CSI data analysis.
- Examine supervised learning approaches to create models that use CSI features to accurately recognize a variety of human activities, including clapping, standing, standing up, no movement, sitting, sitting with left hand up and standing up with the right hand up.

1.2. Objectives

- Collect Raw CSI Data from Access Point.
- Filter out noise and unnecessary signals.
- Design and train a machine learning algorithm architecture capable of processing CSI Data and accurately classify them.
- Collect CSI data from router's mac address, connected devices mac address and perform experiments in a living environment and see if it affects the CSI.
- Use previous data on machine learning model and see how it compares to collected data.
- Perform activities as the CSI data is being collected and see if the machine learning model predicts the output correctly.

1.3. Potential Issues

- Environmental factors that affect data quality and noise from interference, multipath effects and signal attenuation which might make it unreliable
- Hardware compatibility
    - Some background comparison of the two raspberry pi that was considered for use in this project.

	Raspberry Pi5	Raspberry Pi 4+
Processor	Broadcom BCM2711, Quad-core Cortex-A72 (ARMv8) at 1.5GHz	Broadcom BCM2711, Quad-core Cortex-A72 (ARMv8) at 1.8GHz
Memory	2GB, 4GB, or 8GB LPDDR4-3200 SDRAM	4GB, 8GB, or 16GB LPDDR4-3200 SDRAM
Wireless Connectivity	IEEE 802.11ac (2.4 GHz and 5.0 GHz) wireless, Bluetooth 5.0	IEEE 802.11ac (2.4 GHz and 5.0 GHz) wireless, Bluetooth 5.0
Network Card	BCM43455C0 Wi-Fi chip	BCM43455C0 Wi-Fi chip
Nexmon Compatibility	FP = Flash Patching UC = Ucode Compression	M = Monitor Mode RT = Monitor Mode with Radio-Tap headers I = Frame Injection

		FP = Flash Patching UC = Ucode Compression
Power Consumption	Similar	Similar
Nexmon Support	No	Yes

While both RPi's does have the same network card, the nexmon patch, which is only compatible with kernel version 5.9.10. Nexmon compatibility to allow RPi4+ to go into monitor mode was the deciding factor for it to be the receiver and for RPi5 to be used as a transmitter.[4]

- Software compatibility
    - Nexmon-CSI only supports up RPi kernel version up to 5.10.92 and the latest version for both RPi's are the Debian Bookworm which is a version 6.6+. However, the RPi 4+ can be downgraded to Debian Bullseye OS which is a kernel version variant of 5.10.92.
- Computational Resources
    - RPi's typically uses ARM-based processes with lower clock speed and fewer cores as compared to desktop or laptops, this might affect intensive operations such as training a CNN model.

2. Literature Review

Title: Activity Recognition using Wi-Fi Sensing

Rationale: The three primary approaches to human activity recognition are radio frequency-based behaviour recognition, vision-based behaviour recognition, and wearable sensor-based behaviour recognition. Among these, there are still certain serious issues that need to be resolved for the system to function properly. For example, even though wearable sensors can achieve highly accurate fine-grained motion recognition by gathering data on environmental changes based on sensors attached on the body, this comes with a negative that limits their freedom and causes them to feel uncomfortable. For vision-based behaviour recognition, the problem of privacy invasion comes into play. However, for radio frequency-based such as Wi-Fi sensing, there are no such problems as Wi-Fi signals are all around us and are also susceptible to changes based on human actions when performed in-between a receiver and a transmitter. [15] One commonly used signal for Wi-Fi sensing is Received Signal Strength (RSSI) but its effectiveness is restricted by noisy and inconsistent readings. Due to the wealth and consistency of information included in the Channel state information (CSI), a more informative feature of Wi-Fi than RSSI has gained increasing attention by offering detailed on a per-orthogonal frequency division multiplexing (OFDM) basis for each packet while RSSI only offers a single but broad measurement value per packet. [16] In general, Wi-Fi sensing has a lot of promise as a discrete and private method of identifying human activity.

Literature Review: While there are many different network cards and machine learning models that have been developed so far to extract raw CSI from a Wi-Fi signal and pre-process it before using different machine learning models to train with the pre-processed data and predict a human activity but finding the most efficient and optimal overall system is the goal. C.tian proposed that existing strategies, like vision-based and specialized sensor-based techniques, have drawbacks in terms of cost, convenience, and privacy and stated that the advantages of contactless operation, affordability, and availability of Wi-Fi-based human activity identification make it a strong substitute. Using 2 Raspberry pi with Nexmon Firmware and a router where CSI was extracted in a lab environment and noisy data that caused phase error after single value decomposition was filtered out by finding the optimal phase value through Dynamic Time Wrapping (DTW) algorithm which has the property of being transverse between phase and amplitude. This allowed huge improvements in the accuracy of recognising human activity. Even when holding experiments in 2 different scenarios, two scenarios: a conference room and an office environment, considering four common activities (standing, bending over, squatting, and holding up hands), yielded an overall accuracy of 95%.[17] J. Xi proposed another method by employing the ResNet deep learning model to extract the classification's representation features and the phase difference correction techniques to remove original phase difference's skipping on a design system of two desktop computers with intel 5300 as the network card and an open sourced CSI-tool in 5 GHz band which yielded prediction accuracy for human activity at 95.5% of activities consisting of 7 actions, one person activity of waving, walking and sitting down, two person activity of one person waving with one person walking, one person waving with one person sitting down, one person walking with one person sitting down and two persons waving. [18] X. Xie proposed a design system using Convolutional Neural Network (CNN) as the deep learning algorithm and collected CSI using intel 5300 as a receiver and a NetGear R7000 wireless router as the transmitter at 5Ghz frequency band with a total of 16 activities performed. Using Principal Component Analysis as the pre-processing method which yielded a accuracy of 90.43% - 91.07%[20] D.Khan stated that completely classifying one complete series into an activity is what is done in most human activity recognition which requires massive amounts of dataset to give a high prediction accuracy in real-time and instead proposes using a special type of Convolutional Neural Network(CNN), U-Net, utilizing a concept of multi-layer architecture to learn and classify features of CSI on a pixel level. The Experiment was carried out inside a research office and the transmitter was configured on a 2.4GHz frequency band with a channel width of 40 band which had 128 subcarrier count. Two humming-board pro devices were used as receiver to ping the transmitter to create a communication link and collected a total of 4032 training samples and 2304 testing samples. These samples were then pre-processed for outliers to which were feature-normalized according to the subcarrier count then passing it onto the CNN U-Net architecture for training and yielded a accuracy of 98.57%.[19] while [17], [18], [20], [19] all yielded a high accuracy for

prediction based on human activity, there are still propagation challenges in Wi-Fi sensing as stated by X. B. Maxama, like path loss as there is a signal loss as the distance increases from the transmitter and receiver for any propagation path environment and only grows bigger in non-line of sight areas. Co-channel and adjacent channel interference would be another issue with CSI when channels close to each other using the same frequency creeps into one another and gives off unwanted signals, but this would be easily fixed by increasing the distance between receiver and transmitter. Electromagnetic Interference would also cause reliability issues in indoor as metallic structures, electric motors which are all source of electromagnetic interference can easily cause signals to degrade. [21] When comparing to use either the 5Ghz or 2.4 GHz frequency band, I. Dolnska listed the advantages of 5GHz band like the possibility to use ultra-wide channels which significantly increases bit rate which is rarely achieved in 2.4GHz. The amount of non-overlapping channels is also higher in 5Ghz and is significant when coverage of signal need not be large. However, 5GHz does not have a lower level of interference than 2.4GHz band if three or less access points were simultaneously available in the same area, the interference power distribution could have been more or less the same in both frequency bands but in an environment with high access point density, 5Ghz were found to have a lower level of interference than 2.4GHz band.[6] Since there might be limiting factors and potential issues like having access to a controlled environment affecting the collection of CSI namely placement of router, surrounding obstruction, G. Forbers suggested that using access point with Nexmon integrated into raspberry pi is an efficient and effective way to implement a classification system and activity recognition system. The capabilities provided by the raspberry pi board using publicly available released data interaction frameworks like CSIkit and Nexmon allows for interpreting, processing, and visualizing. It was then trained on a Long-Term-Short-Term deep learning model with 11 different activities without any pre-processing as real-time capabilities could be impacted in a way that lowers overhead for further simulation applications and yielded an average accuracy of 92% across prediction of all activities. [4] J. Choi however implemented sensor-aided learning for Wi-Fi positioning together with beacon frames from Wi-Fi access point which proved to be effective as CSI data helped to identify channel conditions and produced an accurate standard deviation output of each distance estimate which allow the positioning performance to improve ranging and positioning in more complicated indoor environments. Experimental results from an indoor office equipped with 59 Wi-Fi access points and Nexmon CSI tool was used to collected training data at 2.4Mhz with each SSID beacon broadcasted at 100ms interval. 6 participants collected unlabelled data by randomly walking. These data were trained using a CNN model which helped to increase average positing accuracy by 25-60cm. [22]

Description: The aim of this project is to be able to find the most efficient, effective, and non-invasive way to detect and classify various human activities. The main objective of this project is to:

- Analyse the different possibly ways to predict human activity based on extracted CSI data.
- Design an activity detection system that is cost effective and efficient using transmitter and receiver where communication between them is available to extract CSI.
- Collect a diverse dataset of CSI data from the designed system while performing activities in a controlled environment.
- Explore and compare the performance of various modelling techniques, such as support vector machines or convolutional neural networks. Design and implement machine learning or deep learning models for the classification of human activities based on the extracted CSI features. Optimize the model architecture and use data pre-processing techniques to optimise the model's architecture.
- Analyse predictions and share insights on limitations and sources of errors.

Data from the experimental setup will be used throughout the project for data gathering and will be mainly done by user, thus ethical approval is not needed for this project.

Deliverables:

- Dataset of specific activities (Clapping, standing, standup, standing right hand up, sitting left hand up and empty)
- Design system that can classify and predicting this activity.
- Recording, reading and analyse of proposed design system and detailed reviews of design, development, and implementation.

Hardware Resources required:

- Raspberry Pi4+
- Raspberry Pi5
- Laptop
- Monitor

Software Resources:

- Nexmon CSI tool
- CSIkit Tool
- Machine learning algorithm

Since the objective of this project is to be cost effective and efficient while also understanding that having a controlled environment is important, raspberry pi5 was used as a transmitter to provide wireless access point and using the laptop as the receiver to create a communication link between them by using ping packets. The Raspberry 4+ will be patched with the Nexmon firmware to enable CSI extraction from the access point.

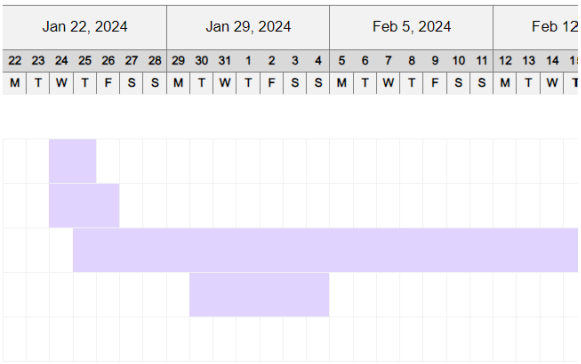
Project Planning including work break down structure & Gann Chart

Link : [Gann Chart Tek.xlsx]

Project start: **Wed, 1/24/2024**

Display week: **1**

TASK	ASSIGNED TO	PROGRESS	START	END
Research and planning				
Project goals & objective	Tek	100%	1/24/24	1/25/24
Blog creation	Tek	100%	1/24/24	1/26/24
Literature Review	Tek	100%	1/25/24	5/1/24
User Requirements	Tek	100%	1/30/24	2/4/24



3. System Design & Implementation

3.1. BackGround

3.1.1. What is Channel State Information (CSI)

In wireless communication systems, Channel State Information (CSI) is an invaluable tool that sheds light on the properties of the radio channel that separates a transmitter and a receiver. It includes details regarding the effects of multipath propagation, attenuation, and interference on a broadcast signal's path across the wireless medium. A matrix or tensor of complex numbers is commonly used to describe CSI, with each element representing a distinct subcarrier (or frequency channel) inside the transmission bandwidth. [2] CSI is chosen of RSSI which is more stable and contains rich contextual information on the transmission link. Due to these CSI characteristics, multi-class classification accuracy is higher than RSSI, which is unreliable. [13] The number of antennas at the transmitter and receiver, the quantity of subcarriers, and the modulation technique in use are some of the variables that affect the CSI matrix's dimensions. An example would be A∈C^(N_sa×N_sc×N_tx×N_rx) which represents a four-dimensional complex valued tensor A which represents: [1]

- N_sa - Quantity of spatial angles or dimensions. The number of spatial angles used to measure or characterize the channel is indicated by this dimension. Depending on whether the channel is being measured at the transmitter or the receiver, it could reflect various angles of signal arrival or departure.
- N_sc - The count of subcarriers. The number of frequency subcarriers that the channel is characterized over is indicated by this dimension. Each subcarrier in an orthogonal frequency-division multiplexing (OFDM) wireless communication system represents a distinct frequency channel.
- N_tx - Quantity of antennas used for transmission. The number of antennas on the communication link's transmitter side is indicated by this dimension. It is a representation of the number of independent transmission streams that can be broadcast simultaneously or of the spatial variety.
- N_rx - The quantity of antennas used for receiving. The number of antennas on the communication link's receiving end is indicated by this dimension. It symbolizes the number of separate receive branches that can receive signals concurrently or the spatial variety.

```
PS C:\Users\tekpun\Desktop\gesture-recognition-csi-master\data\clap> csikit 1712789020.pcap
Hardware: Broadcom BCM43455c0
Backend: Nexmon CSI
Bandwidth: 80MHz
Antenna Configuration: 1 Rx, 1 Tx
Frame Count: 150
Subcarrier Count: 256
Length: 15.67s
Average Sample Rate: 9.60Hz
Average RSSI: -47.0dBm
CSI Shape: (150, 256, 1, 1)
```

Figure 1

As seen in figure 1, the CSI shape is (150,256,1,1) which is the frame count, subcarrier count and the number of receiver and transmitter antennas.

3.1.2. Beacon Frames in IEEE 802.11

One of the management frames listed in IEEE 802.11 that contains network information is the beacon frame. Periodically, beacon frames are sent out to alert other wireless LAN users to its existence and to synchronize all the service set members. Beacon's periodic feature is made use to send requests at predetermined intervals on a regular basis. The frame control is one important aspect to take note of. Beacon frames, which begin with 0x80 in 802.11, are always 20 MHz frames. They make the router's existence known. We therefore set our beacon to 0x80. Referring to figure 2, one can examine the patterns in beacon frames.[22]

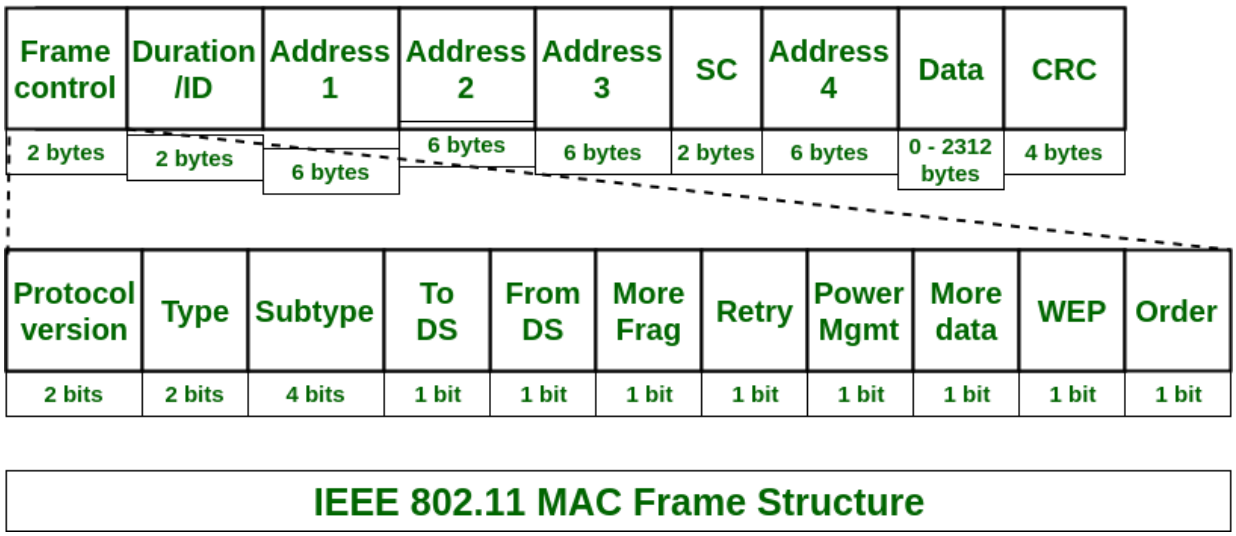


Figure 2

3.1.3. Frequency band and channel bandwidth

One of the IEEE 802.11 standard's throughput-limiting elements is interference. In the 2.4 GHz frequency, adjacent channel interference is caused by overlapping channels. In the 5 GHz frequency, co-channel interference phenomena still exist but there is no interference from adjacent channels. The IEEE 802.11 standard uses 5 GHz and 2.4 GHz bands. Coverage, attained throughput, degree of interference from other devices and systems, number of available channels, interference from neighbouring and co-channels, national legislation, bandwidth availability, security, and cost are the primary attributes.

The 2.4 GHz band has better coverage in open areas; but maximum range is not always required, and the 5 GHz band's coverage may be readily increased with the use of repeaters, which are currently inexpensive and widely accessible.[y] Since for the experiment in this project was based in a small room with little or

no interference, and 80Mhz bandwidth provides a high data rate, low interferences, and precise channel characterization.

Higher data rate	Maximum data rates are higher in the 5 GHz spectrum than in the 2.4 GHz band. This makes it possible for CSI data to be transmitted more quickly, which is advantageous for applications like beamforming if used with more than one antenna that call for real-time or fast data processing.
Less Interference	Compared to the 2.4 GHz band, the 5 GHz spectrum is usually less crowded and encounters less interference from non-Wi-Fi devices. This can lead to more accurate channel estimate and analysis by producing CSI data that is cleaner and has fewer artifacts from outside interference sources.
More Available channels	Compared to the 2.4 GHz band, the 5 GHz spectrum offers more channels that are available and more options for channel capacity. This gives CSI data collecting additional flexibility, enabling enhanced frequency-domain analysis spatial resolution and better spectrum utilization.
Better frequency resolution	The 5 GHz band provides superior frequency resolution for CSI data collecting and analysis because of its higher frequency. This allows for the detection of narrowband changes, including frequency-selective fading or interference from nearby channels, and allows for better granularity in channel estimation.

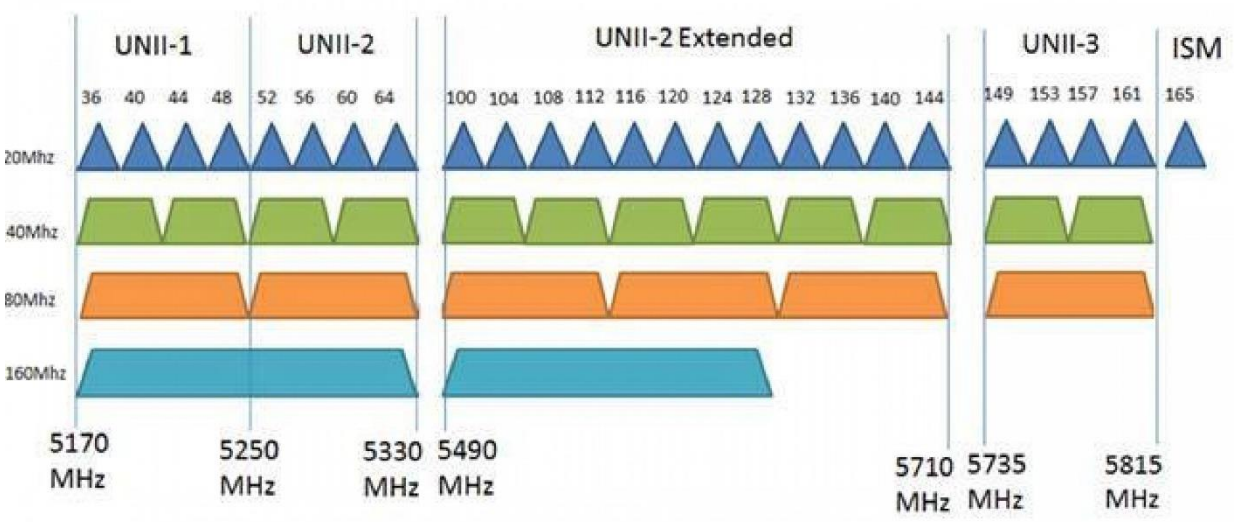


Figure 3

As for the project, channel 36 was used as the channel specification and thus, an 80Mhz bandwidth is formed by combining 4 20Mhz channel, 36,40,44,48 and uses channel 36 as the beacon.

3.2. Software Requirements

3.2.1. Nexmon-CSI Tool

Nexmon-CSI Tool is a project which allows extraction of CSI spatial frames in both the 2.4Ghz and 5 Ghz band with up to 80Mhz bandwidth and supports a range of Raspberry Pi devices. [e]

3.2.2. CSI-Kit

CSIKIT is a tool to extract Channel State Information (CSI) from Nexmon format to process and visualize by using python and numpy.[r]

By using CSIkit, you can view CSI information and visualize it in a plot.

```
PS C:\Users\tekpun\Desktop\gesture-recognition-csi-master\data\clap> csikit 1712789020.pcap
Hardware: Broadcom BCM43455c0
Backend: Nexmon CSI
Bandwidth: 80MHz
Antenna Configuration: 1 Rx, 1 Tx
Frame Count: 150
Subcarrier Count: 256
Length: 15.67s
Average Sample Rate: 9.60Hz
Average RSSI: -47.0dBm
CSI Shape: (150, 256, 1, 1)
```

Figure 4

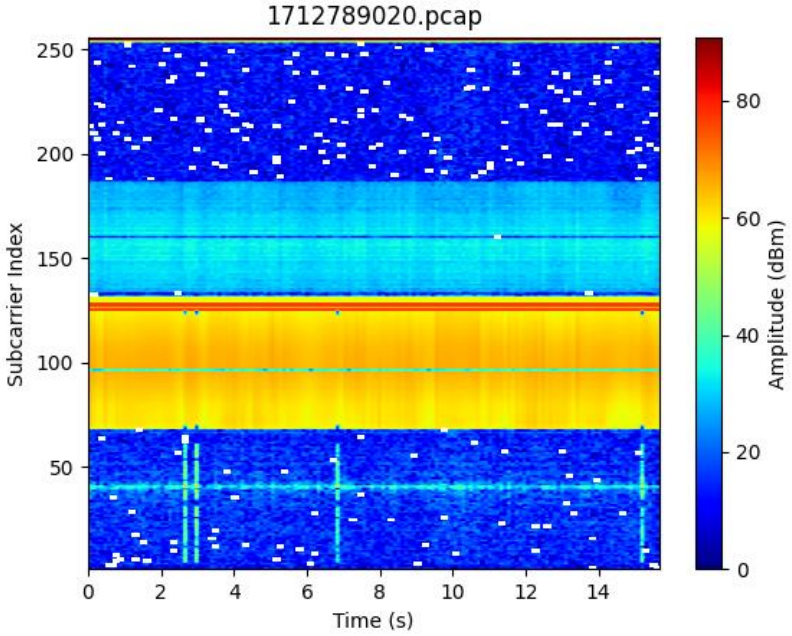


Figure 5

3.2.3. Python version 3.8

Python on a virtual environment with the following libraries installed: click==8.0.3 cycler==0.11.0 fonttools==4.28.3 importlib-metadata==4.10.0 itsdangerous==2.0.1 Jinja2==3.0.3 joblib==1.1.0 kiwisolver==1.3.2 llvmlite==0.37.0 MarkupSafe==2.0.1 matplotlib==3.5.0 numba==0.54.1 numpy==1.20.3 packaging==21.3 pandas==1.3.4 Pillow==8.4.0 pyparsing==3.0.6 python-dateutil==2.8.2 pytz==2021.3 PyWavelets==1.2.0 scikit-learn==1.0.1 scipy==1.7.3 setuptools-scm==6.3.2 six==1.16.0 threadpoolctl==3.0.0 tomli==1.2.2 torch==1.10.1 torchvision==0.11.2 typing-extensions==4.0.1 Werkzeug==2.0.2 zipp==3.7.0

3.2.4. Iperf3

Iperf3 is used to determine the highest bandwidth that may be achieved between two network endpoints. It produces UDP or TCP traffic between the client and server and provides information on jitter, packet loss, throughput, and other performance parameters. One instance of iperf3 functions as a server and another as a client in its client-server architecture. To personalize the testing conditions, users can provide several factors, including the protocol (TCP or UDP), port number, test length, and packet size. This can be used to set a communication link between transmitter and receiver.

3.3. Hardware Requirements

3.3.1. Raspberry Pi

2 Raspberry Pi, one to act as Receiver and the other one to act as a Trasnmitter.

3.3.2. Edge Server

Laptop to be used to store dataset and for CNN Machine learning model training

3.4. System Design

This system consists of 3 important components, the Transimitter (Tx), Reciever (Rx) and the Edge Server.

- Transmitter - Raspberry Pi 5 (Wireless Hotspot Access point to mimic router on channel 36 on 80Mhz bandwidth at 5Ghz)
- Receiver - Raspberry Pi 4 (Nexmon firmware patched and using it to extract CSI Data by going into monitor mode to sniff on access points)
- Server edge - Pulling data from the receiver via PuTTy and storing it for later use in machine learning model.

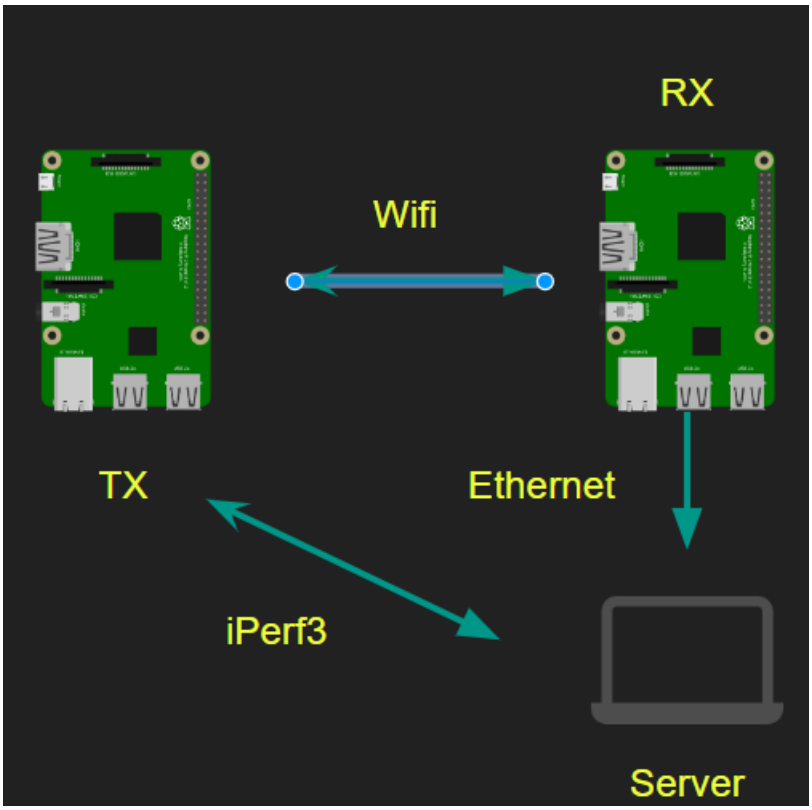
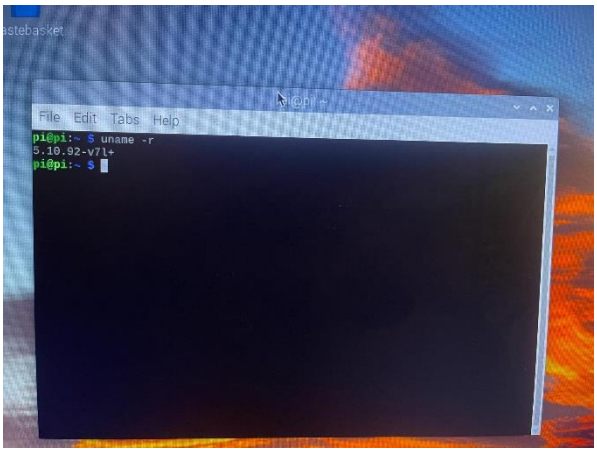


Figure 6: System Architecture

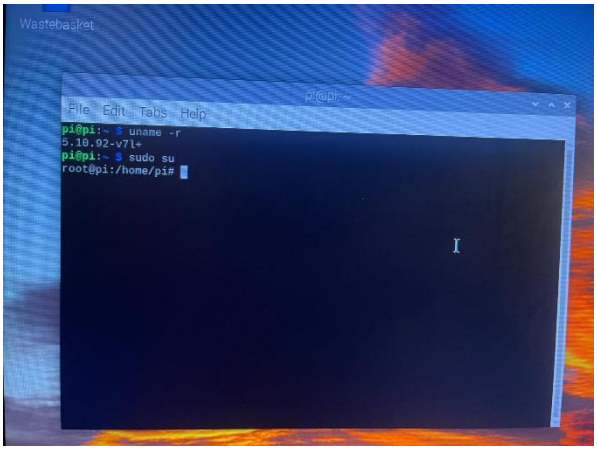
3.5. System Setup

3.5.1. Receiver setup

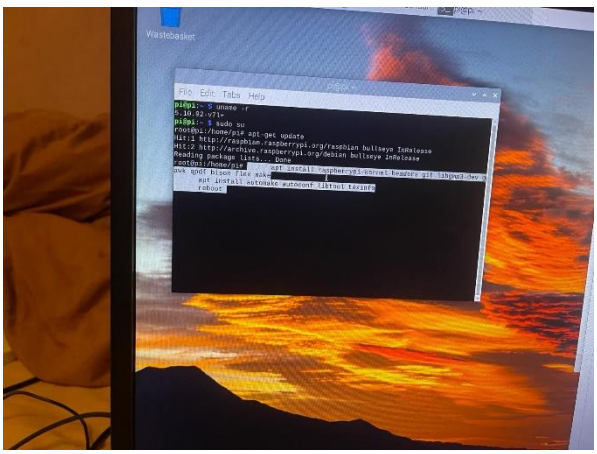
Since this Raspberry pi was to have the Nexmon firmware patched in it, the latest kernel version is 6.6+ Raspberry Bookworm that came with the standard OS download and was not suitable as it required kernal version 5.10, 5.4 or 4.9. The latest version that was available for Raspberry Pi 4+ which nexmon support was the kernal version 5.10.92. The older version for the OS can be found on Raspberry pi images website[t]. Once the kernal version has been downloaded, install dependencies, and do not run sudo apt upgrade, that will change automatically change the kernel version to the latest available. Only kernels upto version 5.10 are compatible with Nexmon at the time of writing. [e] Below is the steps to download and patch Nexon into the firmware and check if it is installed correctly.



Uname -r provides the kernel version and as u can see in figure above, kernel version is 5.10.92 which is compatible with nexmon.



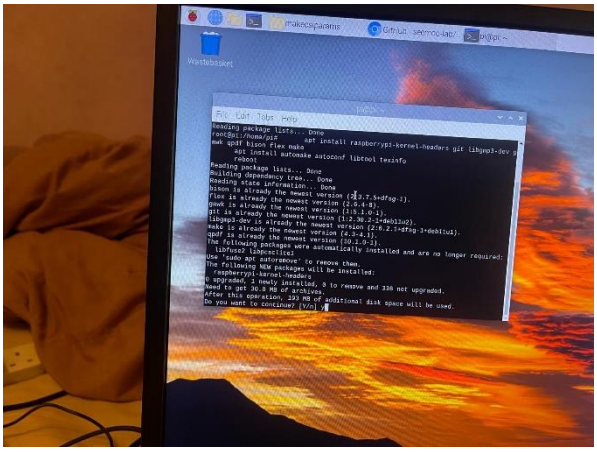
Sudo Su to enter into root user access, this will allow you to patch the kernal with the required files from nexmon.



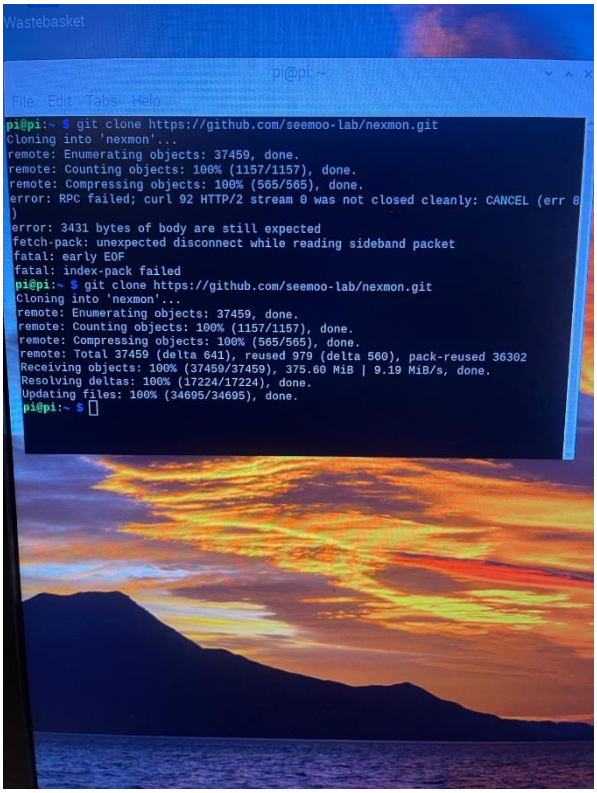
apt install raspberrypi-kernel-headers git libgmp3-dev gawk qpdf bison flex make

apt install automake autoconf libtool texinfo

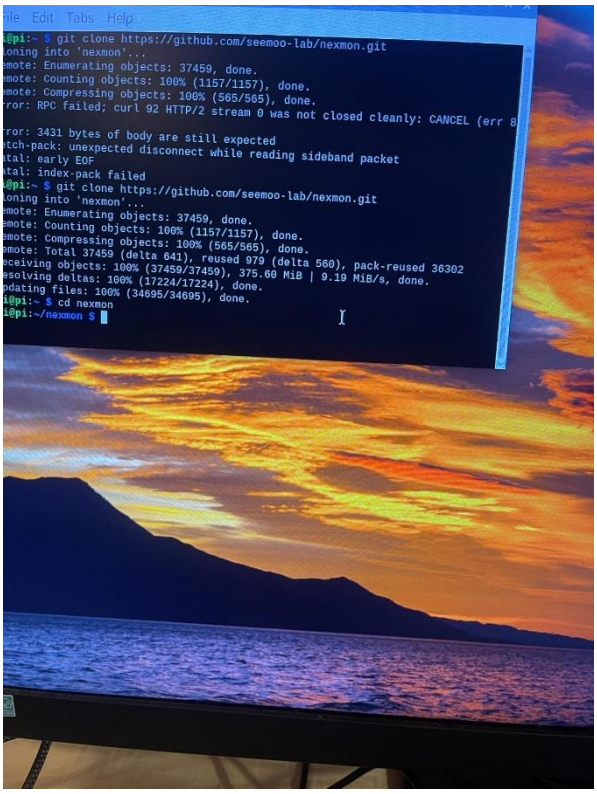
reboot



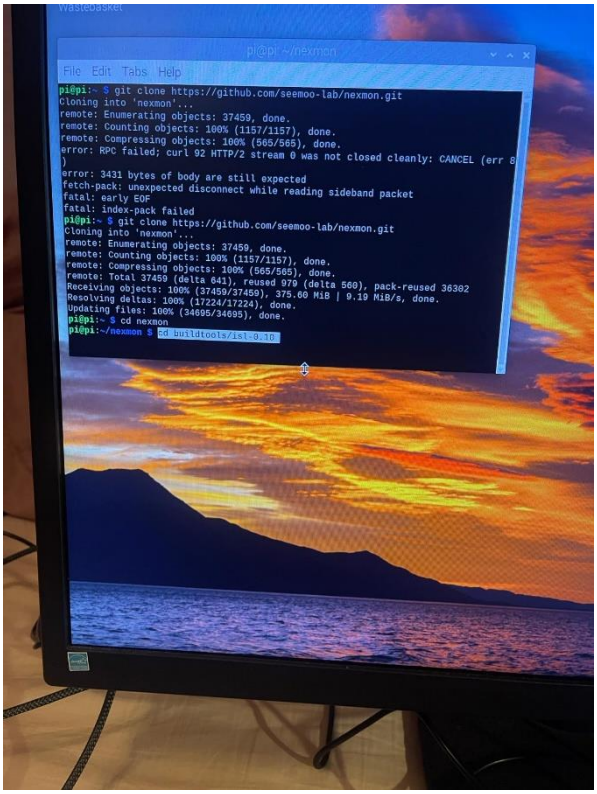
When prompted, key in “y” from keyboard and press enter. Once the installation is done, the RPi will reboot itself.



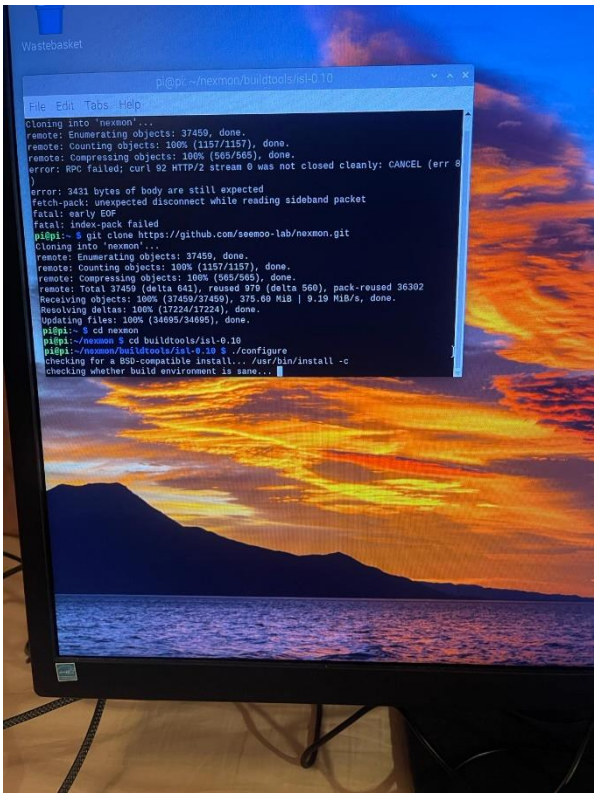
git clone https://github.com/seemoo-lab/nexmon.git, this will download the required file for nexmon from gitbhub.



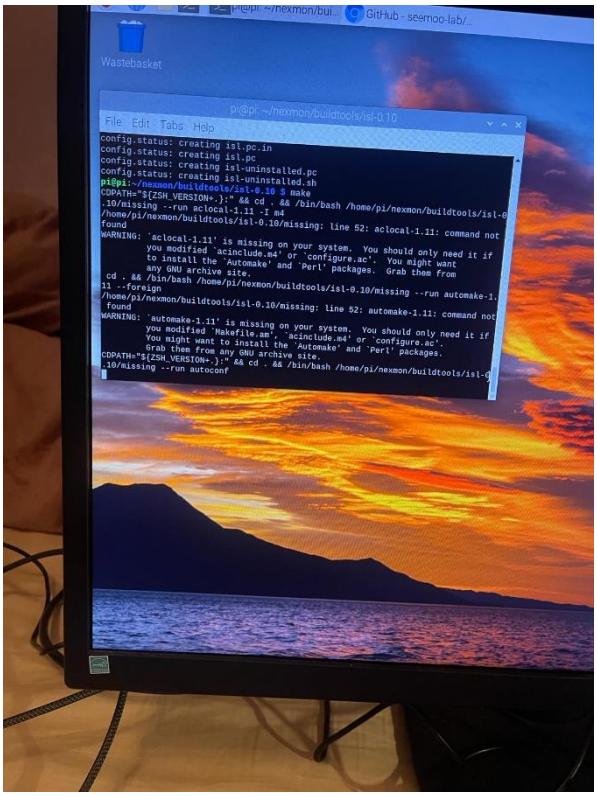
Go into the downloaded folder by doing cd nexmon.



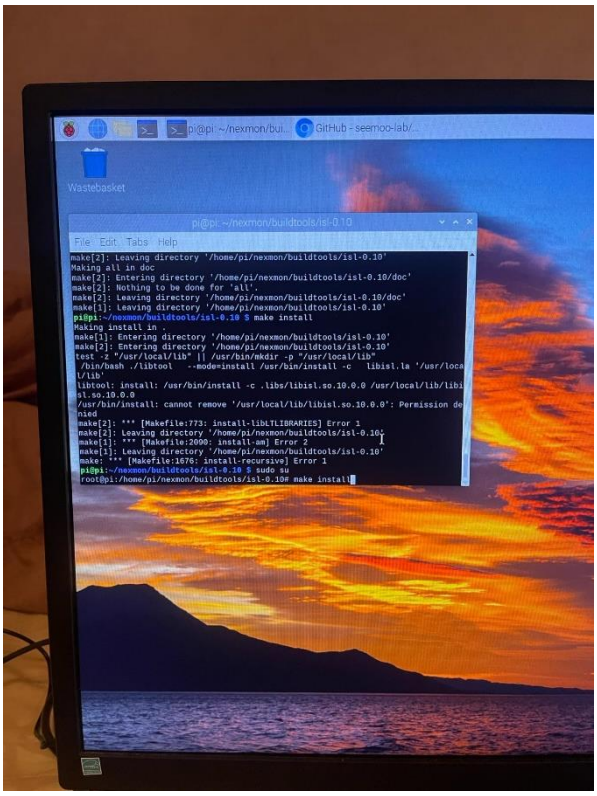
Once inside the folder, go into buildtools/isl-0.10 by doing cd buildtools/isl-0.10



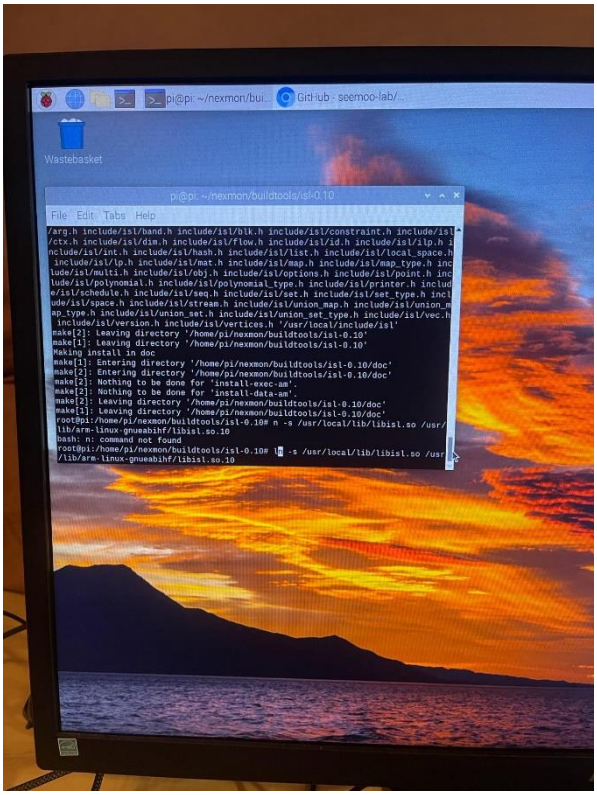
./configure



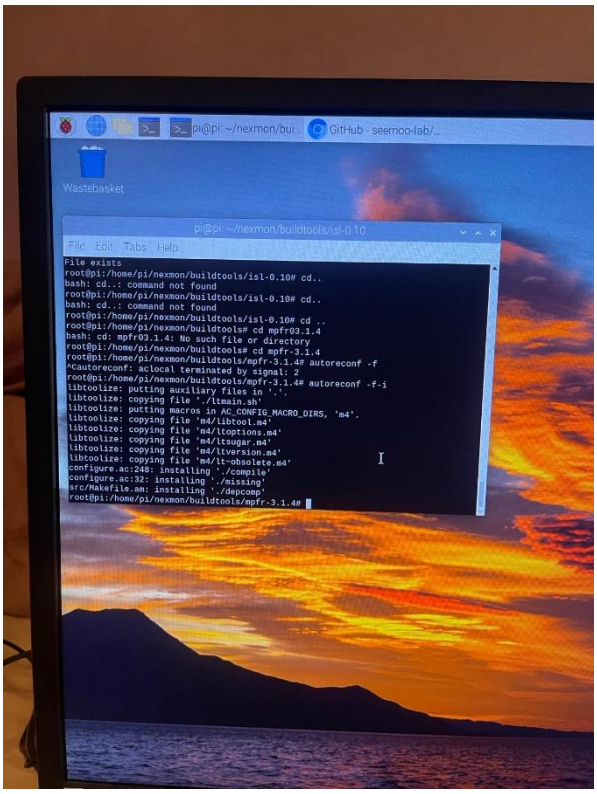
make



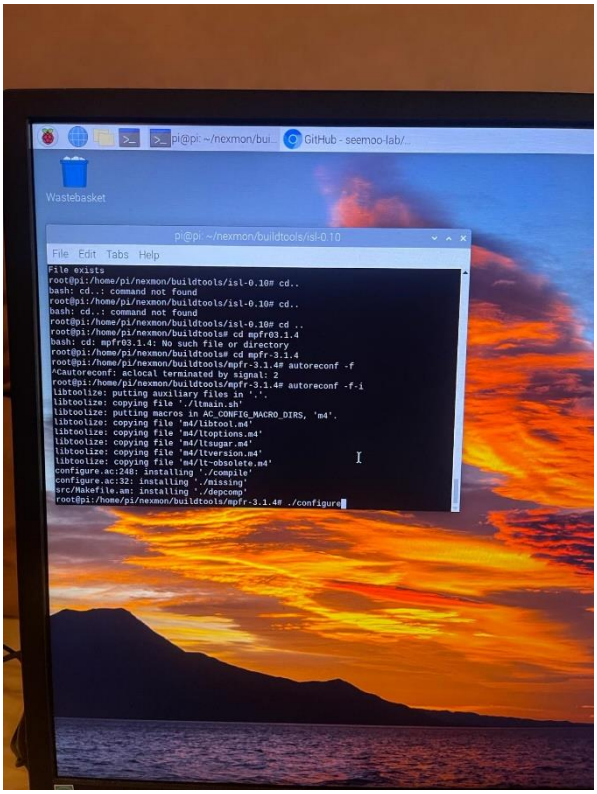
Before doing make install, you must be in root user. To do this do sudo su and them make install



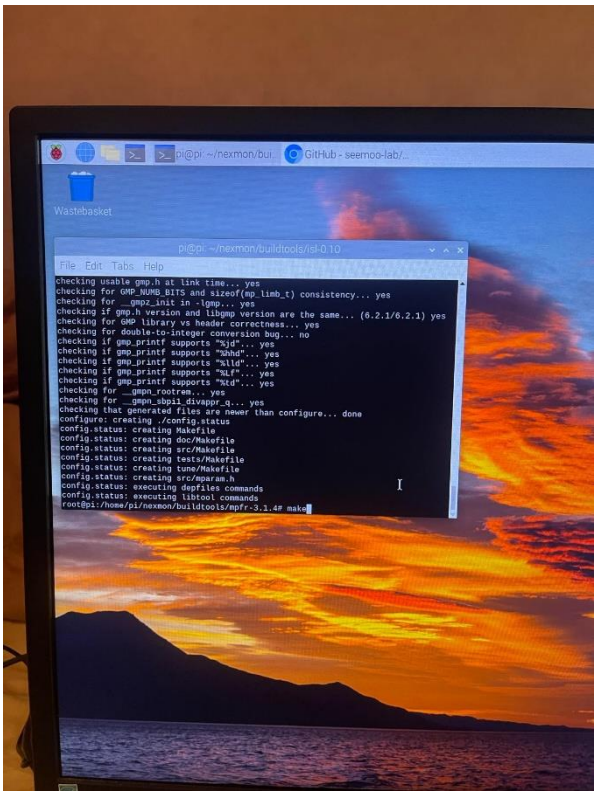
Once done, cd ln -s /usr/local/lib/libisl.so /usr/lib/arm-linux-gnueabihf/libisl.so.10



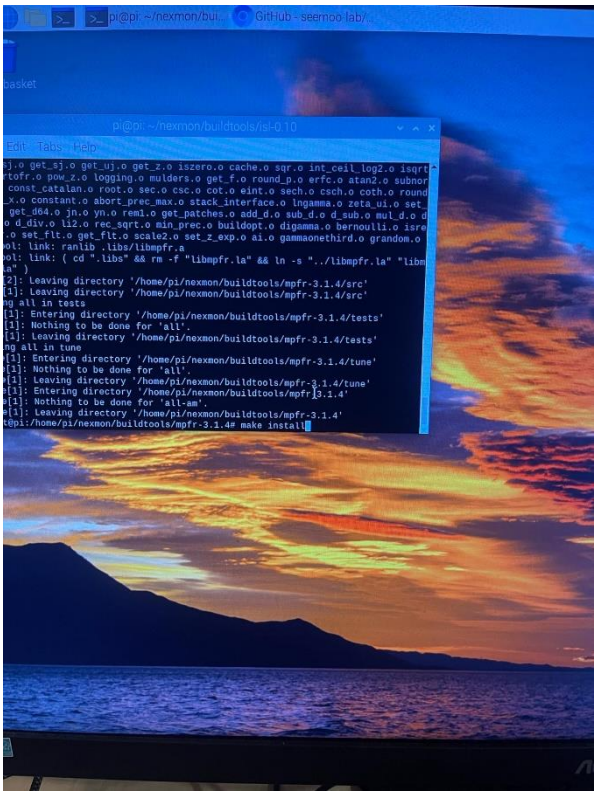
cd /home/pi/nexmon/buildtools/mpfr-3.1.4



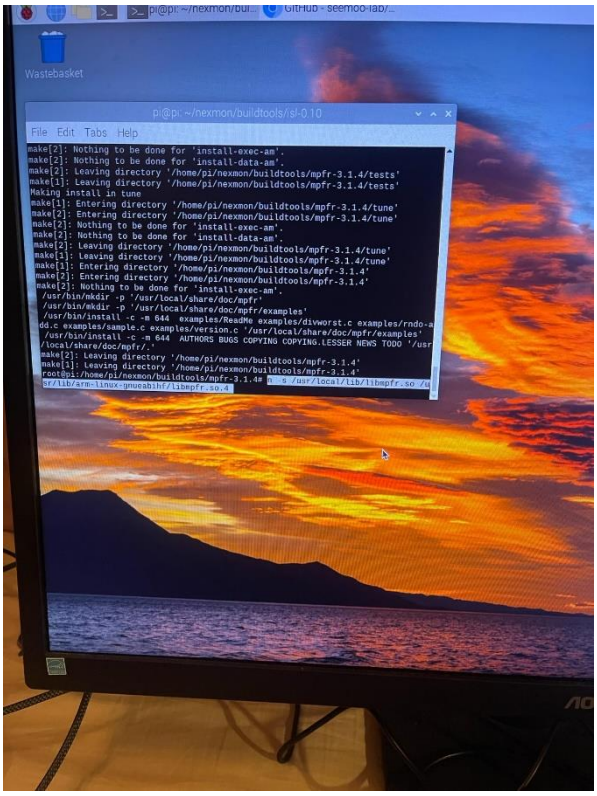
./configure



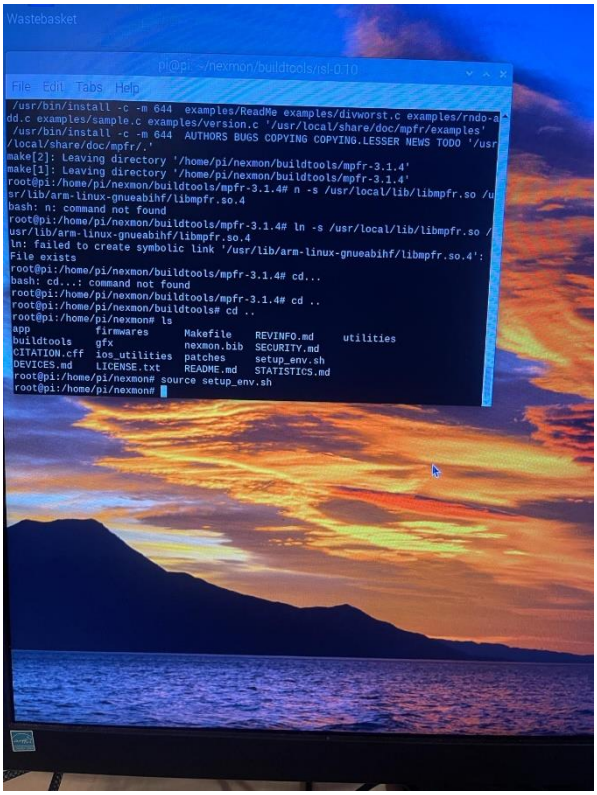
Make



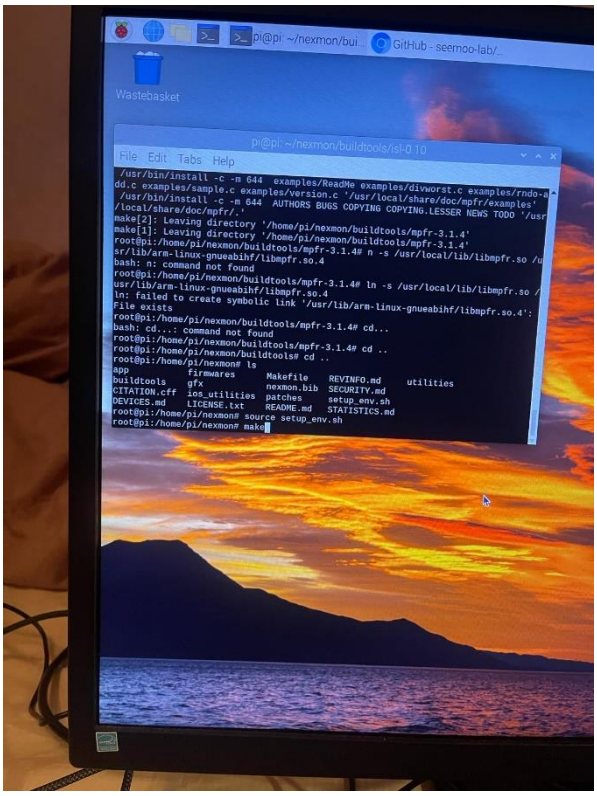
make install but make sure you are in root user.



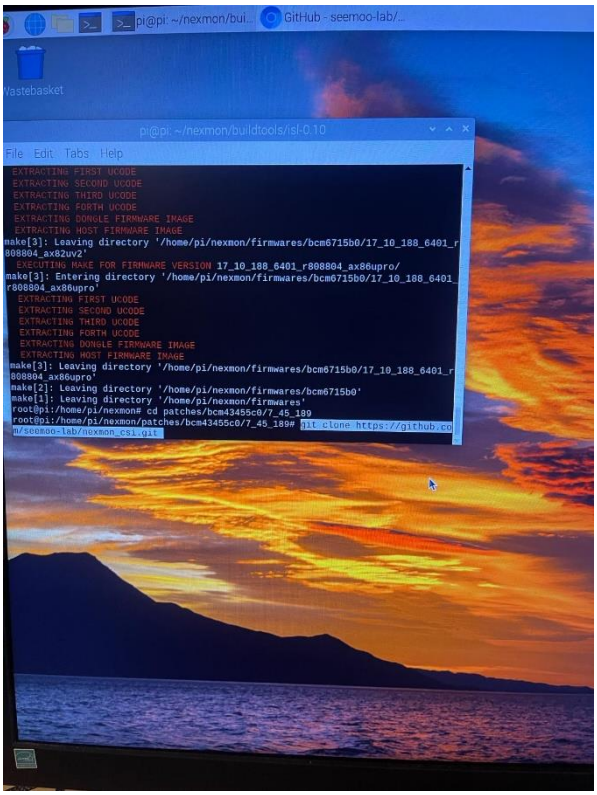
ln -s /usr/local/lib/libmpfr.so /usr/lib/arm-linux-gnueabihf/libmpfr.so.4



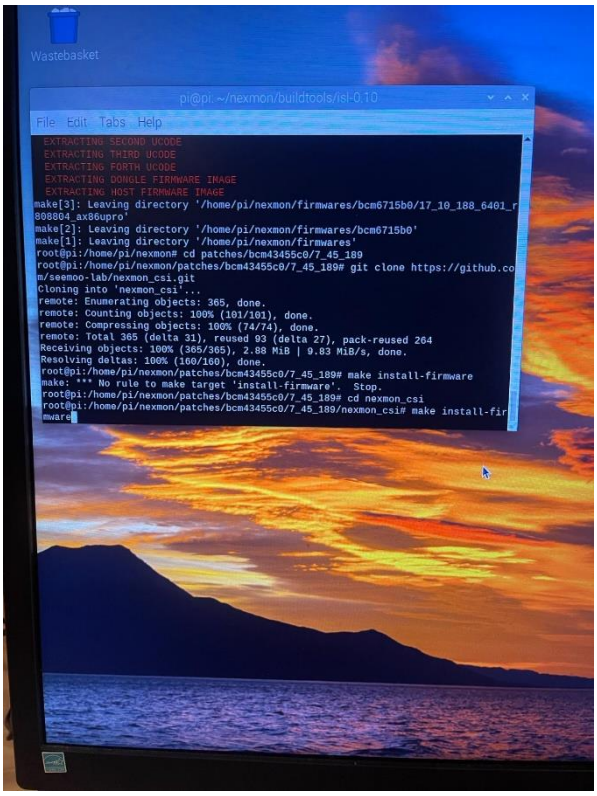
cd /home/pi/nexmon source setup_env.sh to setup build environment



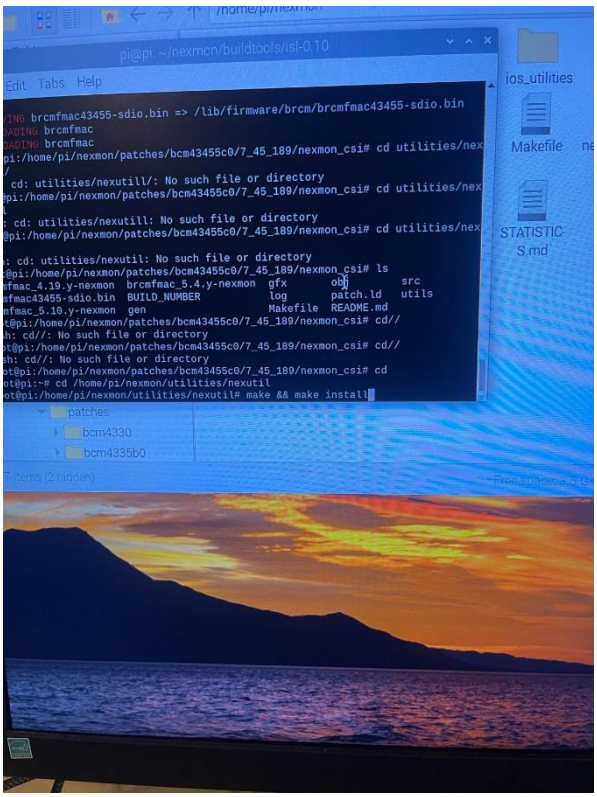
Run make to extract



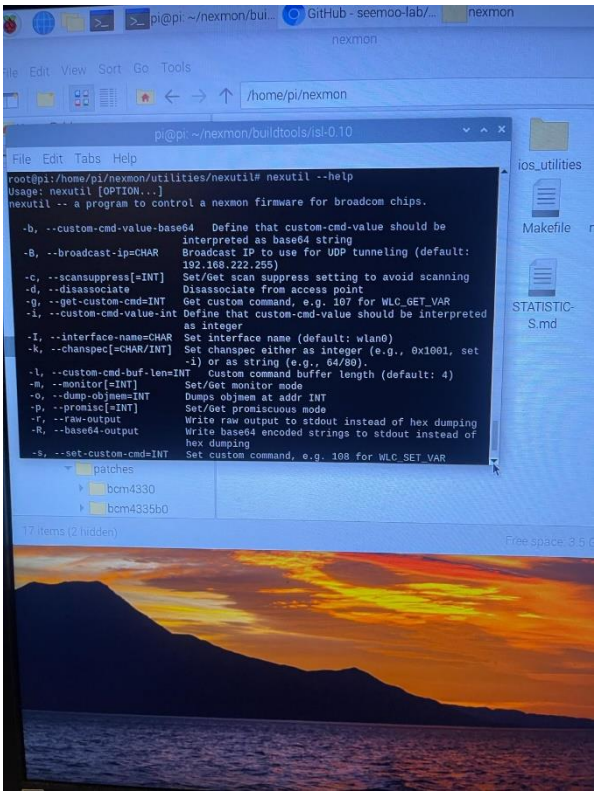
Cd into patches/bcm43455c0/7_45_189 and clone this repo git clone https://github.com/seemoo-lab/nexmon_csi.git



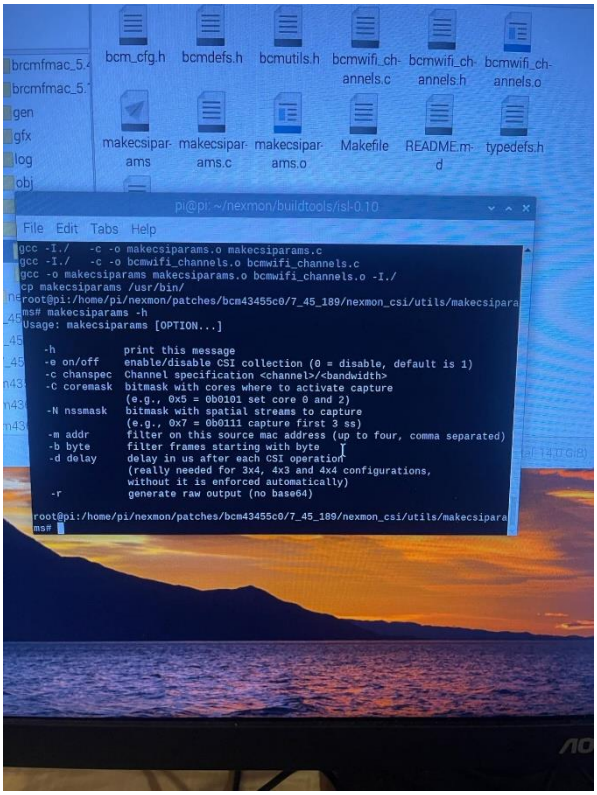
Cd nexmon_csi and run make install-firmware to compile nexmon firmware patch and install into onto the RPi



from the nexmon root directory switch to the nexutil folder: cd utilities/nexutil/. Compile and install nexutil: make && make install



Once done, you can do nexutil -help to check if it has been installed correctly.



You can also check if it has been installed by doing makecsiparams -h

- Running Nexmon
    - New Terminal
    - Makecsiparamms -c 36/80 -C 1 -N -1 -m < Mac address of Tx > -b 0x80 – this will generate a parameter to be used in the Nexmon utils program.
    - Pkill wpa_supplicant – this has to be done in root user mode.
    - Nexutil -iwlan0 -s500 -b -l34 -v< parameter >
    - Enable monitor mode on wlan0
    - Check if monitor mode is on by doing iw dev
    - Nexutil-k - checks the channel and which bandwidth it is in

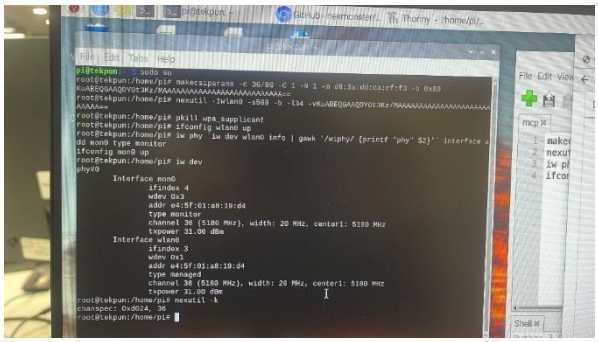


Figure 7

- On Figure 7, even going through the steps, the channel and bandwidth were still not changed, and the problem was that there has to be a pause after the nexutil -iwlan0 command is being processed before inputting the next step.

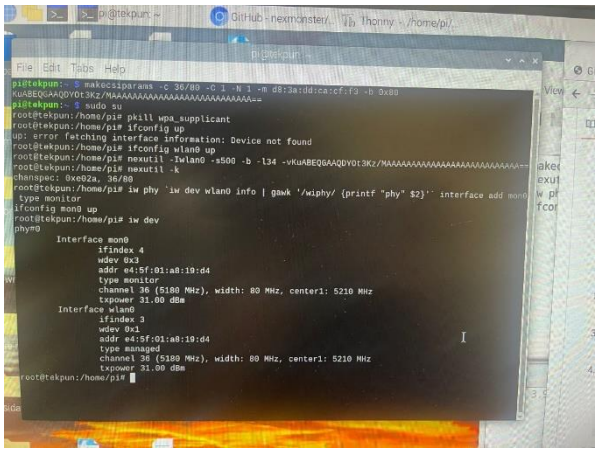
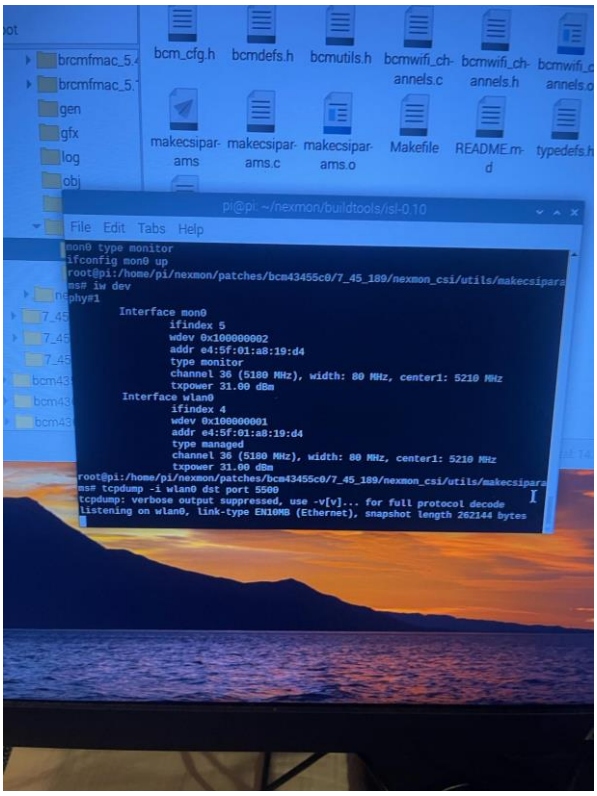
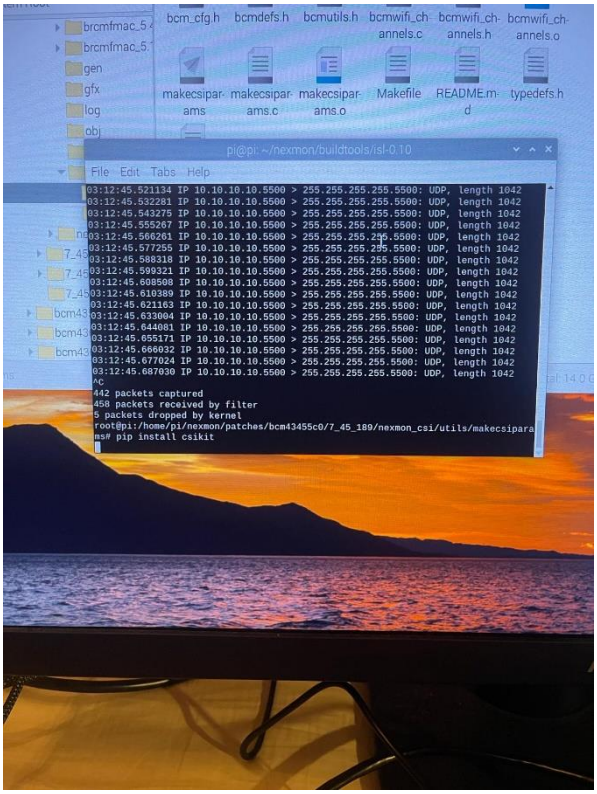


Figure 8

- Figure 8 shows the correct channel and the bandwidth which is 36 and 80Mhz. Once this is done, the receiver is ready to sniff for ping packets going from transmitter to edge server laptop and vice versa.



tcpdump -i wlan0 dst port 5500 to check for packets from Tx to edge server laptop



Packets captured from tcpdump

3.5.2. Transmitter setup

Setup A Wireless AP On Raspberry Pi 5

- Channel 36 or any accessible channel that uses 5Ghz band

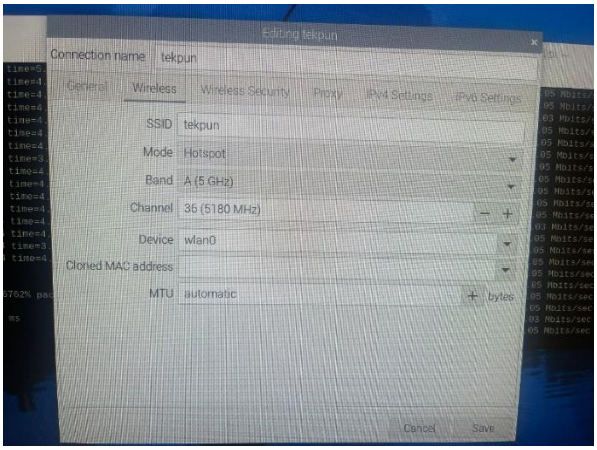
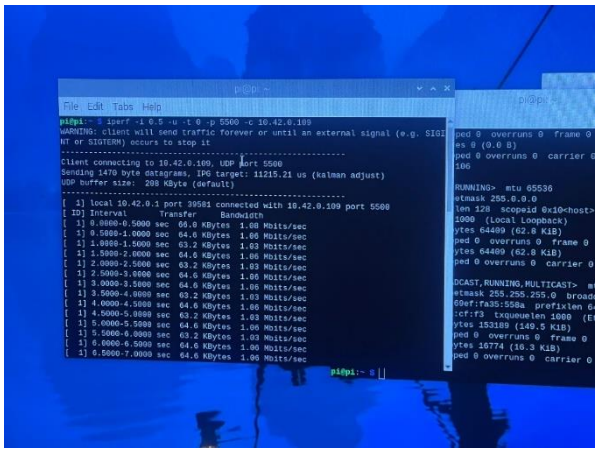


Figure 9

Figure 9 shows the access point created for this project which has SSID as tekpun, in the 5GHz band and on channel 36



- Use iPerf to set interval at 0.5 and send UDP packet at port 5500 @Rx Ip Address by using command: iperf -I 0.5 -u -t 0 -p 5500 -c <Edge server laptop ip address>
- To install iperf, Sudo apt install iperf3 on new terminal

3.5.3. Experiment environment and dataset collection

To generate a dataset representative of daily life, a real living space was utilized.

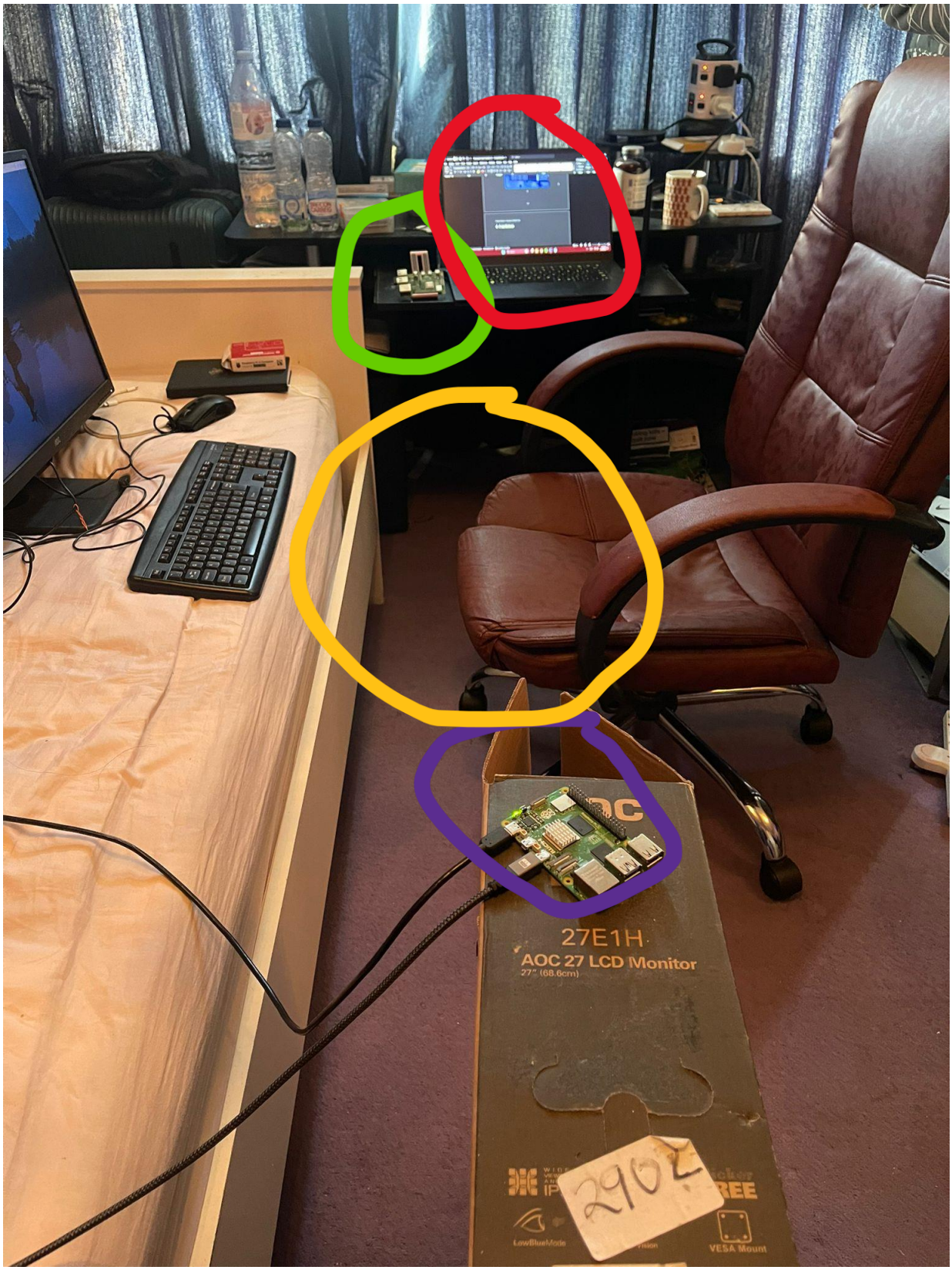


Figure 10

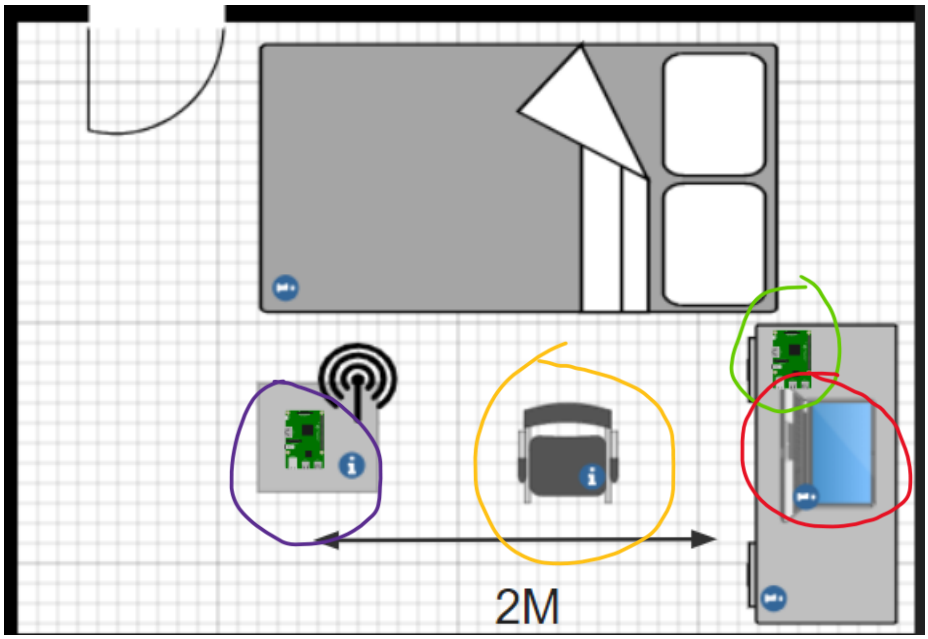
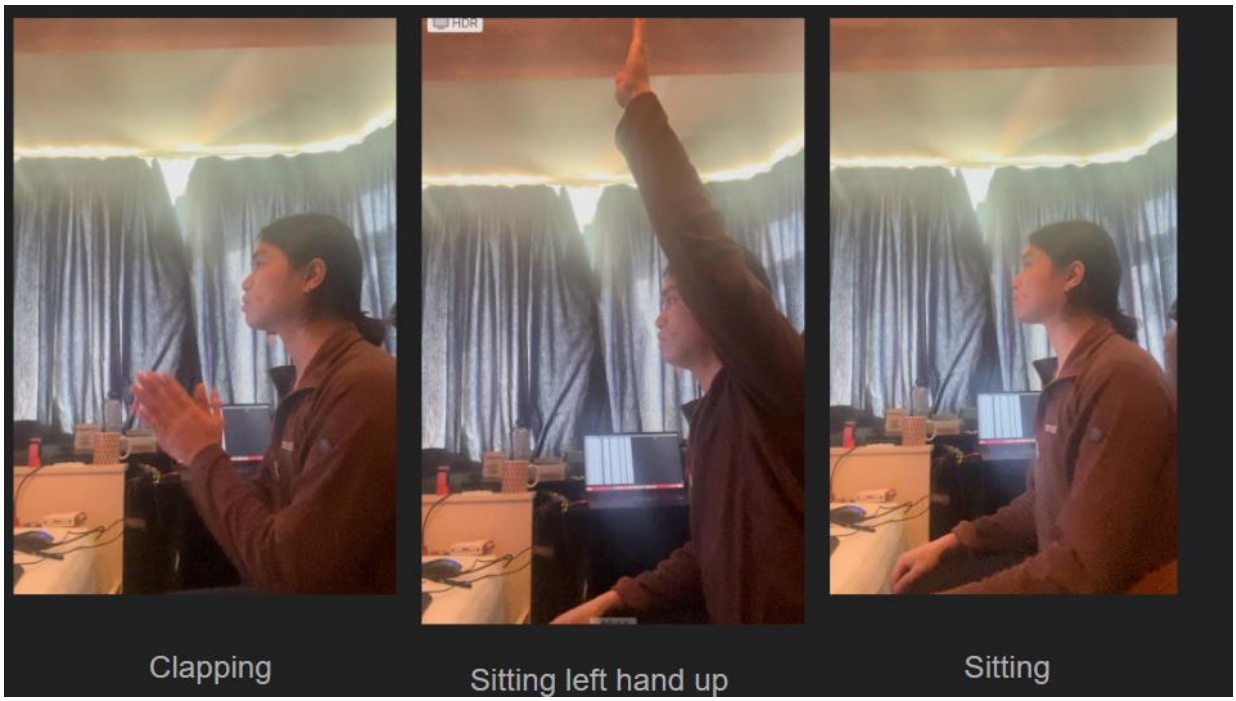


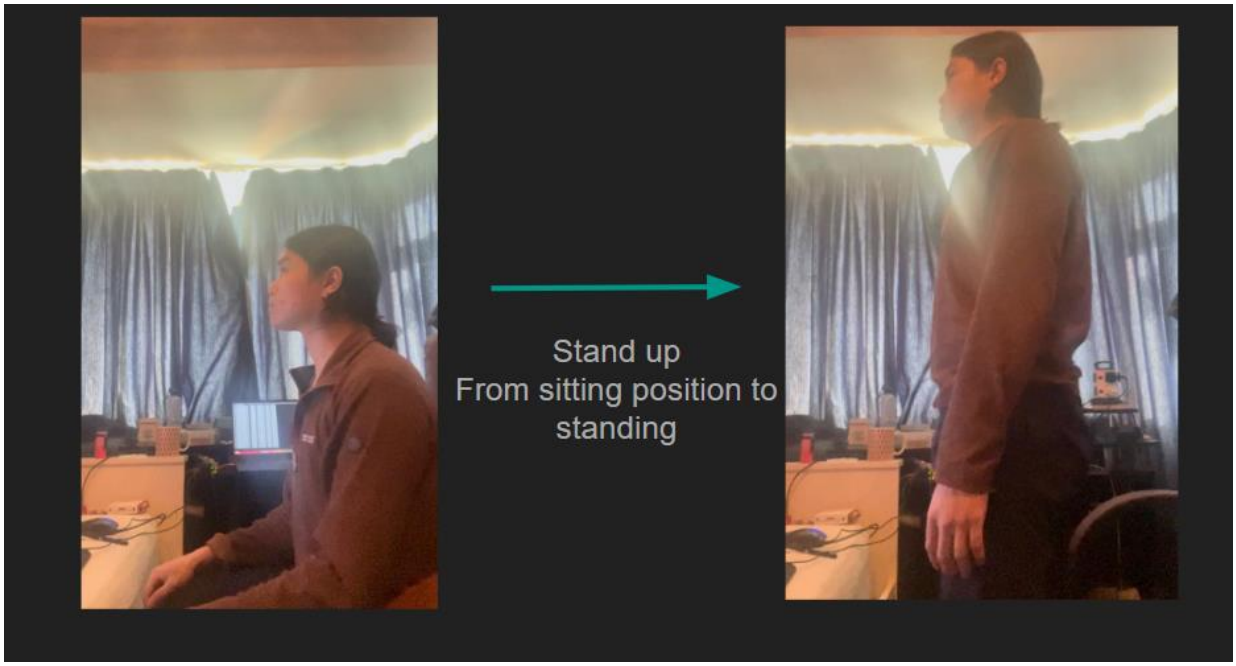
Figure 11

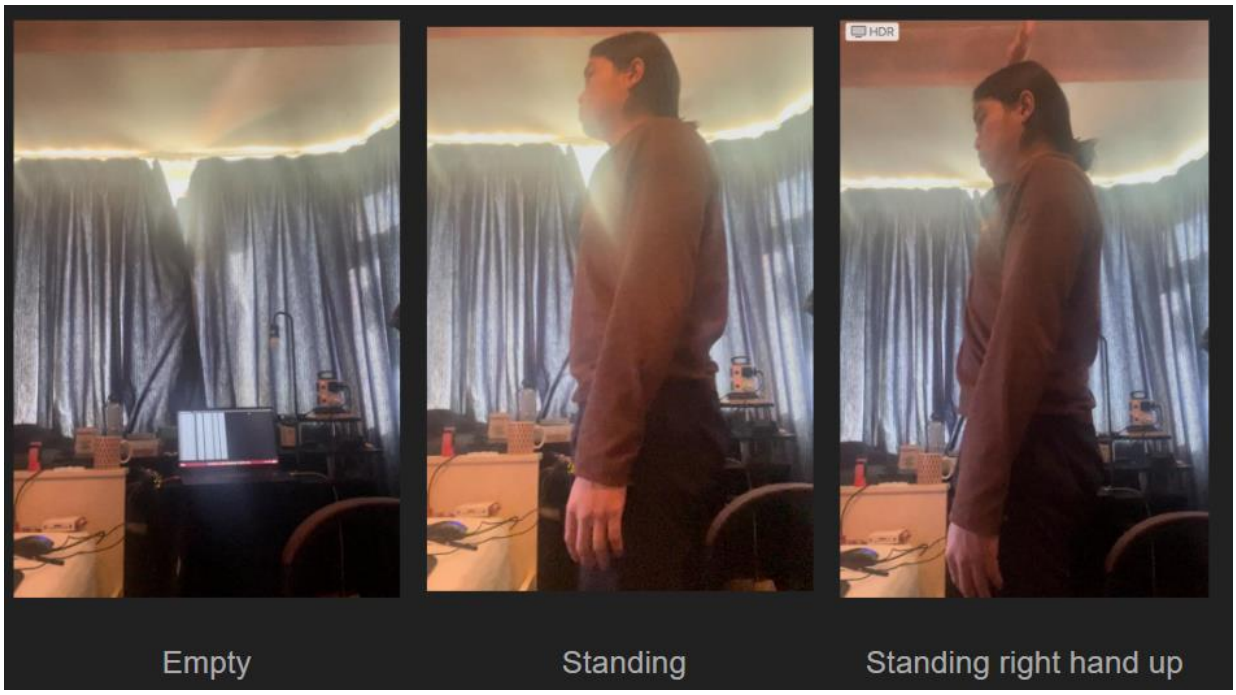
Color	Function
Purple	Transmitter – Provides access point
Green	Receiver – Sniffs for the access point and extract CSI Data
Red	Edge server – Analyses CSI data in machine learning algorithm
Brown	Area where actions were performed for dataset collection

There were online datasets available for use such as Wi-fi Activity Recognition using LSTM as proposed by S. Yousefi, whereby Linux 802.11n CSI tool was used to extract the CSI data and 3 antennas were used in in the receiver and was different from this project where only 1 antenna was used, thus making the dataset unreliable in this case.[u] and also Wi-Fi Human-to-Human Interactions Dataset by Rami Alazrai, where the data was collected in an indoor setting using twelve distinct human-to-human interactions conducted with 40 different pairs of individuals.[i] This data was also redundant for this project as it the data was captured in an outdoor based environment as compared to the environment for this project whereby its indoors.

7 Different activities were performed for dataset collection.


Clapping
Sitting left hand up
Sitting


Stand up
From sitting position to
standing



All gestures were performed in a 3-second period with 10Mhz sampling rate as the RPi was only capable of broadcasting beacon frames with minimum intervals of 10 ms. This process was repeated 200 times over a span of 4 days.

Each action was collected 200 times making the total dataset a total of 1400 pcap files. The collection process was automated by a python script where by “tcpdump -I wlan0 dst port 5500 -c 150” which collects a total of 150 packet each iteration, saves it as a {currenttimestamp}.pcap and a 2 sec break between each iteration.

Collected files on the Reciever was then extracted to edge server laptop via putty using putty followed by command on terminal pscp.exe -r -p 22 pi@tekpun:/< path of files to transfer over > "< path of where to store the files on edge server>

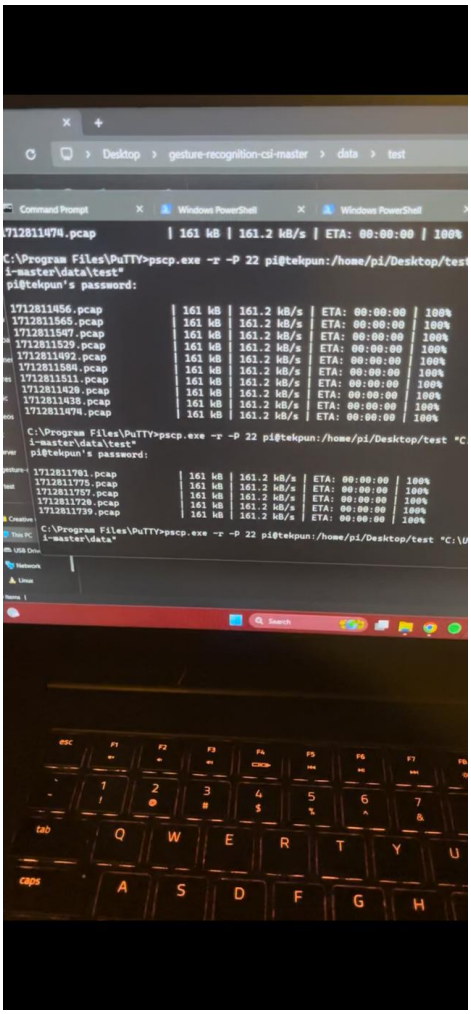


Figure 12

Figure 12 shows an example of receiving CSI data from receiver via ethernet cable over putty to edge server.

3.5.4. Machine learning Algorithms

Machine learning is the most efficient way in the field of data analytics to predict something by devising some models and algorithms. These analytical models are what allows dataset to produce reliable and valid results and decision based on patterns, features, historical data and trends in data[o] Z. He proposed a reliable deep learning (DL) attention mechanism-based Wi-Fi passive sensing technique based on channel state information (CSI). Convolutional neural network (CNN)-ABLSTM, a mix of CNNs and attention-based bi-directional long short-term memory (LSTM), is the name of the suggested technique. He also noted that because of the fine-grained qualities of CSI, CSI-based Wi-Fi passive sensing was designed to accomplish the high precision of human activity recognition (HAR). The algorithm evidently showed CNN-ABLSTM training results have an accuracy of roughly 96% to 97% and a rather excellent performance. [p] H. E also proposed a deep Convolutional neural network model trained on a Wi-Fi on a

dataset of Wi-Fi CSI data collected for seven daily human activities, including falls performed in an indoor environment. With 85% of the data used for training and 15% for testing, the overall test yielded a 95% accuracy and an average loss of 0.3%. [a] What can be noted is that both mentioned projects use CNN which is a particular kind of deep neural network that is made to analyse structured, grid-like data, like photographs. For applications like object identification, picture segmentation, and image classification, it's one of the most widely used and efficient structures. There are also other machine learning methods such as Recurring Neural Network(RNN) but Z.lieu stated while existing RNN works have demonstrated that it can provide efficient CSI feedback and reconstruction for time-varying MIMO channels, the complexity that comes along with it such as having a large number of parameters in RNN layers as compared to CNN layers and still having not much of a difference in the output accuracy does not justify it. [s]

Aspect	Recurring Neural Network (RNN)	Convolutional Neural Network(CNN)
Architecture	Intended to record sequential data's connections with time.	Mostly utilized in grid-like data for the extraction of spatial features.
Application	Ideal for jobs like activity recognition, gesture recognition, and time-series prediction that need for sequential data processing.	Ideal for applications like object detection, localization, and image classification that call for the extraction of spatial features.
Network Depth	Able to handle deep architectures; nevertheless, vanishing, or overflowing gradients may cause problems for longer sequences.	Generally shallow designs, although for classification tasks, they can be coupled with deep fully linked layers.
Handling Noisy Data	Potentially fragile to long-term dependence and noisy inputs.	Efficient in removing spatial patterns from noisy data
Memory Requirements	Memory intensive, requires more resources for training and inference	Lesser intensive than RNN but deeper and complex architecture may require more resources

By this, it is concluded that both may be of use in the case of machine learning for deep learning, the use of CNN is much more justifiable due to the limitations of computational power the project was based on and the amount of dataset that has been collected via CSIkit and Nexmon tool which produces CSI in matrices and heatmap images.

3.5.5. Edge server setup

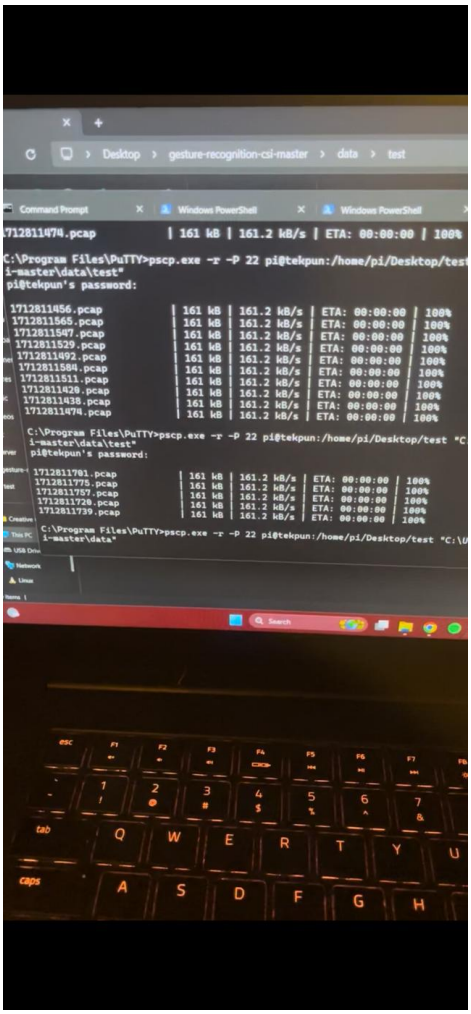
SSID:	tekpun
Protocol:	Wi-Fi 5 (802.11ac)
Security type:	WPA-Personal
Manufacturer:	Intel Corporation
Description:	Intel(R) Wireless-AC 9560 160MHz
Driver version:	23.30.0.6
Network band:	5 GHz
Network channel:	36
Link speed (Receive/Transmit):	86/86 (Mbps)
Link-local IPv6 address:	fe80::2ce1:390c:f2e3:dce5%25
IPv4 address:	10.42.0.109
IPv4 DNS servers:	8.8.8.8 (Unencrypted)
Physical address (MAC):	3C-F0-11-FF-06-40

Connect to new access point created from transmitter

```
[  5] 21568.00-21569.01 sec   128 KBytes  1.05 Mbits/sec  0.188 ms  0/90 (0%)
[  5] 21569.01-21570.01 sec   130 KBytes  1.06 Mbits/sec  0.139 ms  0/91 (0%)
[  5] 21570.01-21571.00 sec   127 KBytes  1.05 Mbits/sec  0.263 ms  0/89 (0%)
[  5] 21571.00-21572.00 sec   127 KBytes  1.04 Mbits/sec  0.170 ms  0/89 (0%)
[  5] 21572.00-21573.00 sec   128 KBytes  1.05 Mbits/sec  0.227 ms  0/90 (0%)
[  5] 21573.00-21574.00 sec   128 KBytes  1.05 Mbits/sec  0.268 ms  0/90 (0%)
[  5] 21574.00-21574.83 sec   106 KBytes  1.04 Mbits/sec  0.182 ms  0/74 (0%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-21574.83 sec  0.00 Bytes  0.00 bits/sec  0.182 ms  49/1936873 (0.0025%)
iperf3: interrupt - the server has terminated

C:\Users\tekpun\Downloads\iperf-3.1.3-win64\iperf-3.1.3-win64>iperf3 -s -p 5500
-----------------------------------------------------------
Server listening on 5500
-----------------------------------------------------------
Accepted connection from 10.42.0.1, port 39098
[  5] local 10.42.0.109 port 5500 connected to 10.42.0.1 port 52011
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-1.01   sec   124 KBytes  1.00 Mbits/sec  416354372.539 ms  0/87 (0%)
[  5]   1.01-2.00   sec   128 KBytes  1.06 Mbits/sec  1249908.134 ms  0/90 (0%)
[  5]   2.00-3.01   sec   128 KBytes  1.05 Mbits/sec  3752.483 ms  0/90 (0%)
[  5]   3.01-4.00   sec   127 KBytes  1.04 Mbits/sec  12.216 ms  0/89 (0%)
[  5]   4.00-5.01   sec   130 KBytes  1.05 Mbits/sec  0.255 ms  0/91 (0%)
[  5]   5.01-6.00   sec   127 KBytes  1.05 Mbits/sec  0.219 ms  0/89 (0%)
[  5]   6.00-7.00   sec   128 KBytes  1.05 Mbits/sec  0.352 ms  0/90 (0%)
[  5]   7.00-8.01   sec   128 KBytes  1.05 Mbits/sec  0.356 ms  0/90 (0%)
[  5]   8.01-9.01   sec   128 KBytes  1.05 Mbits/sec  0.297 ms  0/90 (0%)
[  5]   9.01-10.00  sec   127 KBytes  1.05 Mbits/sec  0.267 ms  0/89 (0%)
```

Install iperf3 from https://iperf.fr/iperf-download.php and cd into it in terminal followed by command -s -p 5500 which allows it to be a server in port 5500 and listens for any packets being sent over.



Download putty from https://www.putty.org/ and cd into it in terminal followed by command pscp.exe -r -p 22 pi@tekpun:/< path of files to transfer over > "< path of where to store the files on edge server> Note that the receiver has to be connected to edge server laptop via ethernet cable for this to work.

Python files to be used for dataset preprocessing using CSIkit, torch, numpy. All python files are used in conjunction with each other. Code are also provided in the appendix of this report.

Python file name	Python file function
Csitest.py	Visualises CSI data obtained from Wi-Fi device using the csikit library. Specifies a file path to a .pcap file to read the CSI data. Extracts CSI matrix, frame count and subcarrier count. CSI matrix is dependant on the amplitude and not phase. Selects the antenna for transmitter and receiver then clips the values to -20dBm and plots the CSI matrix over time and subcarrier index using matplotlib with colour representation of dBm levels matching the colour

	intensity. Plot will show signal characteristics, amplitude and shift variation over time and frequency.
Dataset.py	Initializes gathering of all CSI data from label directories and sub directories of their folders then reads the CSI data and extract its matrix while applying transformation by clamping and selection subcarrier count. It then returns the CSI data with its corresponding label.
Models.py	Two CNN model architecture are defined CNN1 and CNN2 by torch. Each of their characteristics and layers are explained in the later part of the report.
Train.py	Splits data gathered from dataset.py into training validation and testing set using torch then initializes the selected model, optimizer and learning rate schedule in torch. After each epoch cycle, it shows the training accuracy, loss and validation accuracy and loss. Once the listed epoch cycle is completed, best accuracy will be shown and generating a confusion matrix and loss plot evaluating the model's performance.
Utils.py	Used in conjunction with other codes to provide analysis and visualization of CSI data, training progress and model evaluation
Inference_server.py	Loads a pre trained model. Takes an input and generate a prediction using said model. After competition, prints the prediction and total time taken.
Tcpwrite.py	Does tcpdump in port 5500 and saves the file as {timestamp}.pcap for 150 packets and then pauses for 2 seconds before running again, this process is repeated as many times as written in the code.

4. Pre-processing & Testing

4.1. Data pre-processing

To make the dataset that we collected we must remove unnecessary noise that was collected.

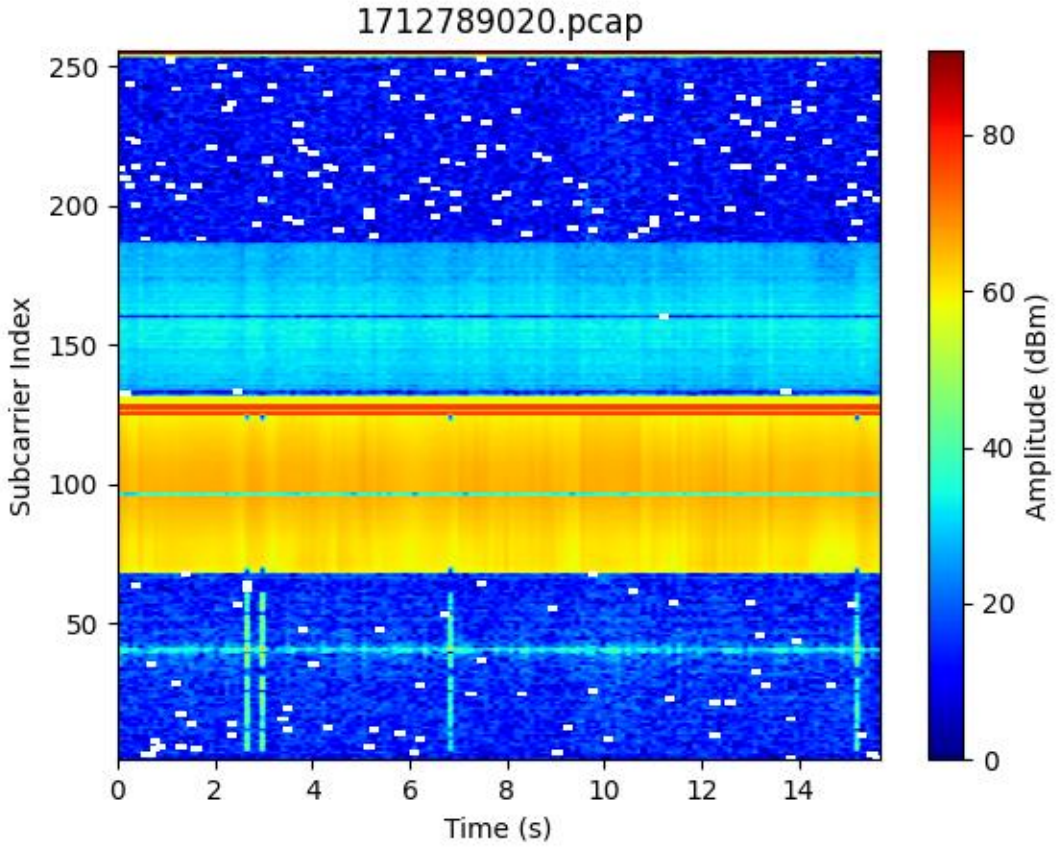


Figure 13

Figure 13 above shows a CSI data that was collected.

- Subcarrier Index:
  - Within the available frequency spectrum, data transmission takes place over several subcarriers in Wi-Fi communication systems. Within the channel, each subcarrier corresponds to a certain frequency bin.
  - Which subcarrier inside the channel is being measured or observed is indicated by the subcarrier index in CSI data.
  - A channel's subcarrier count and frequency are defined by Wi-Fi standards like IEEE 802.11n. For instance, there are usually 52 subcarriers distributed at regular intervals in the 20 MHz bandwidth of 802.11n and using 80MHz bandwidth will result in having 256 subcarriers.
- Amplitude:

- The magnitude or strength of the received signal on each subcarrier is represented by the amplitude in CSI data.
- It displays the received signal's strength or intensity at a given frequency or subcarrier.
- The distance between the transmitter and receiver, environmental obstructions, interference from other devices, and wireless channel conditions are a few examples of the variables that might affect the amplitude value.

- To remove the unnecessary noise and extreme values, we can see that anything below -20dBm is the minimum value thus we remove anything below -20dBm using CSIkit

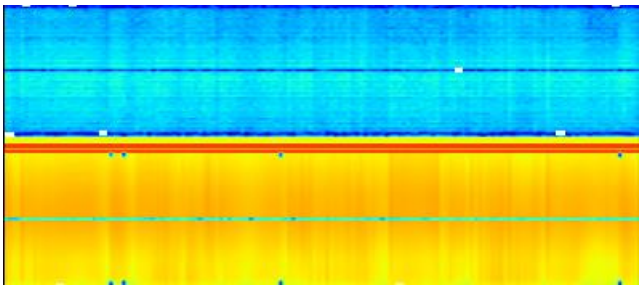


Figure 14

- Figure 14 above shows the CSI data after it has been clipped off without noise and extreme values.

4.2. Results after data preprocessing

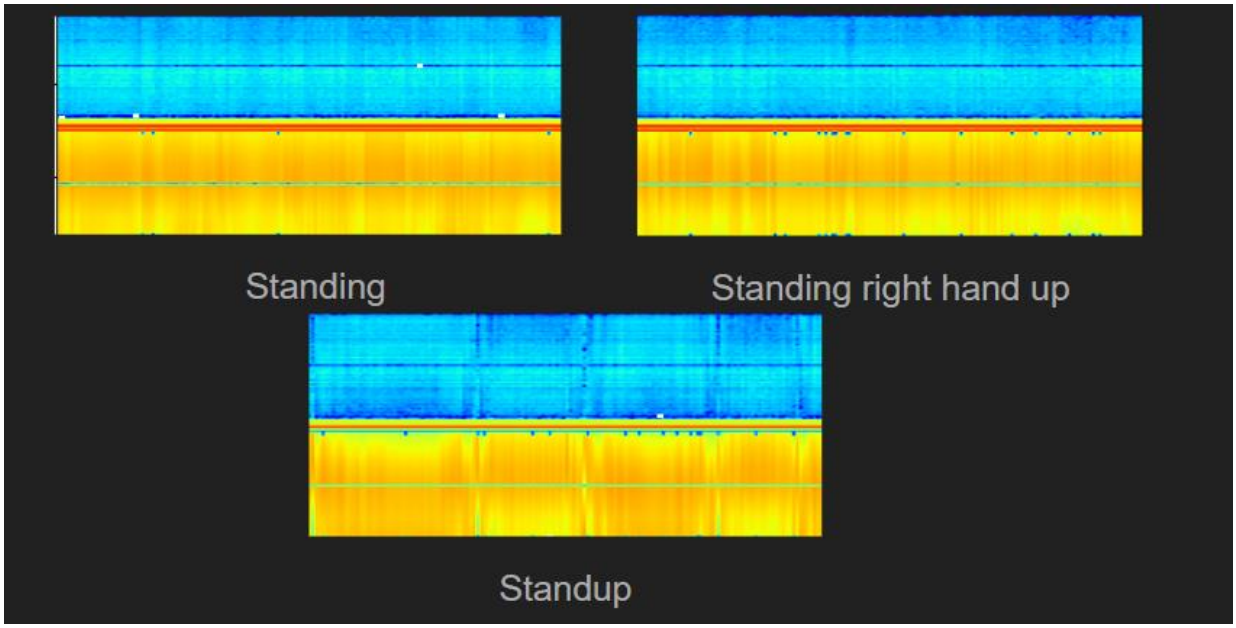


Figure 15

Figure 15 shows the results for standing, standing right hand up and standup after preprocessing the data

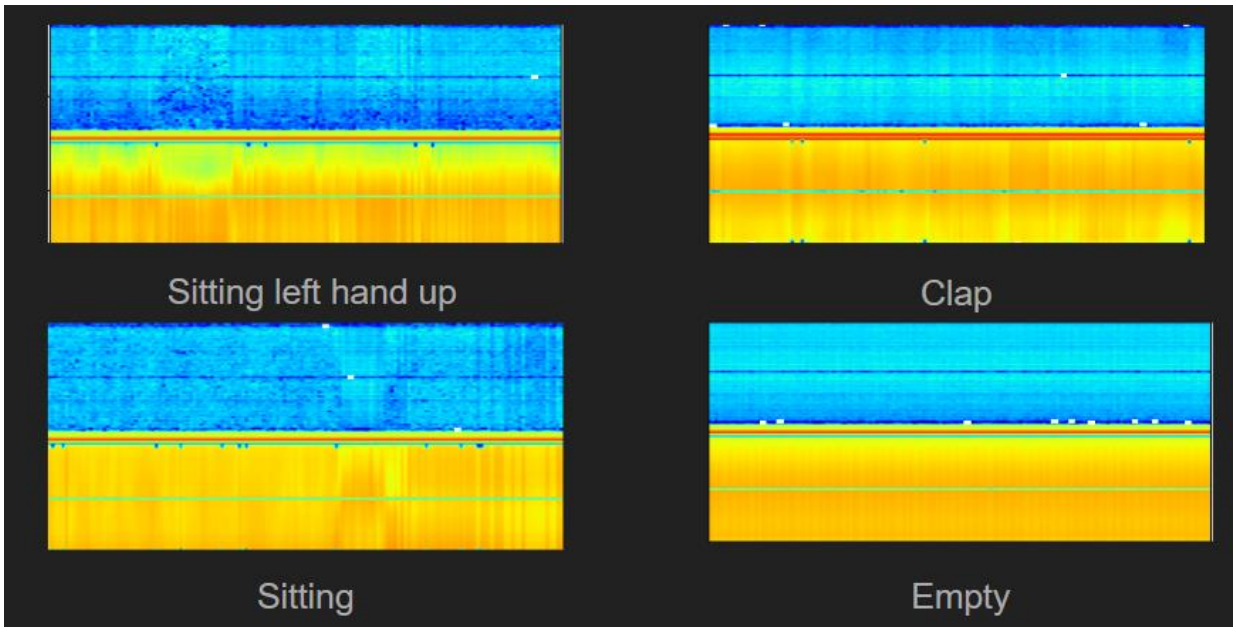


Figure 16

Figure 16 shows the results for sitting left hand up, clap, sitting and empty after preprocessing.

4.3. Machine Learning Model Training

Two different CNN models were used for the machine learning neural network using torch library from python.

No.	CNN1	CNN2
1	5 Layers of convolutional layers	5 Layers of convolutional layers
2	Batch normalization after each convolutional layer	Batch normalization after each convolutional layer
3	ReLu after each convolutional layer	ReLu after each convolutional layer
4	Max pooling after every two convolutional layers	Max pooling after every two convolutional layers
5	Dropout layers after max pooling layer	Dropout layers after max pooling layer
6	Adaptive average pooling layer	No Adaptive average pooling layer
7	2 Fully connected layers with dropout	3 Fully connected layers with dropout

8	Input dimension is variable	Input dimension is fixed at (128x150)
9	7 Output dimension as specified by total actions	7 Output dimension as specified by total actions

- 1. Number of convolution layers
  - CNN1 and CNN2 each have five convolutional layers.
  - In convolutional neural networks, the essential building blocks known as convolutional layers oversee using learnable filters to perform convolution operations on input data to extract features.
- 2.Batch normalization
  - In both models, batch normalization is used following each convolutional layer.
  - By normalizing each layer's activations, batch normalization decreases internal covariate shift and speeds up the training process.
- 3. Activation function
  - In both models, the Rectified Linear Unit (ReLU) activation functions are applied after every convolutional layer.
  - ReLU gives the network non-linearity, which enables it to discover intricate connections between the characteristics in the input data.
- 4. Pooling layers
  - In both models, max pooling is implemented following every two convolutional layers.
  - By extracting the most significant features, max pooling lowers the spatial dimensions of the feature maps, assisting in the reduction of computational complexity and control overfitting.
- 5. Dropout layers
  - In both models, dropout layers come after max pooling layers.
  - During training, dropout forces the network to learn more robust features by arbitrarily setting a portion of the input units to zero. This helps prevent overfitting.
- 6.Adaptive average pooling layers:
  - After the convolutional layers, CNN1 has an adaptive average pooling layer.
  - Adaptive average pooling makes sure the network can handle input images of different sizes by dynamically adjusting the spatial dimensions of the feature maps to a predetermined size.
- 7. Fully connected layers
  - In its classifier component, CNN1 has two fully connected layers (with dropout).
  - In its classifier component, CNN2 has three fully connected layers (with dropout).

    - Convolutional layers extract features, which are then combined by fully connected layers to generate predictions for the output classes.
- 8. Input dimensions
    - CNN2 requires fixed input dimensions of 128x150, but CNN1 can accommodate changeable input dimensions because of the adaptive average pooling layer.
- 9. Output dimensions
    - There are 7 actions that were performed thus having 7 parameters, which is why it's always set to 7 and it also determines the output dimension of both models.

Overall, the topologies of the two models are very similar, however there are a few minor differences, including the number of layers and the inclusion of an adaptive average pooling layer. Performance concerns, processing resources, and task-specific requirements would all play a role in which model is selected. To have more valid readings, experiments were conducted using both CNN1 and CNN2 models to gauge both their best accuracy.

4.3.1. Training model experiments

All model has the original data split into training set, validation set, and testing set at the ratio of 6: 2: 2 with different epoch cycle.

4.3.1.1. CNN2 Model -10 Epoch cycle

```
Epoch 0: 82.707259 (sec)
     Train acc: 0.174, loss: 0.288
     Val   acc: 0.232, loss: 0.134
Epoch 1: 81.137097 (sec)
     Train acc: 0.275, loss: 0.077
     Val   acc: 0.296, loss: 0.094
Epoch 2: 79.378478 (sec)
     Train acc: 0.387, loss: 0.043
     Val   acc: 0.407, loss: 0.042
Epoch 3: 81.998081 (sec)
     Train acc: 0.458, loss: 0.033
     Val   acc: 0.514, loss: 0.028
Epoch 4: 84.531608 (sec)
     Train acc: 0.488, loss: 0.032
     Val   acc: 0.529, loss: 0.026
Epoch 5: 81.539844 (sec)
     Train acc: 0.514, loss: 0.028
     Val   acc: 0.304, loss: 0.117
Epoch 6: 77.006886 (sec)
     Train acc: 0.487, loss: 0.029
     Val   acc: 0.525, loss: 0.027
Epoch 7: 80.456265 (sec)
     Train acc: 0.502, loss: 0.027
     Val   acc: 0.479, loss: 0.032
Epoch 8: 83.416436 (sec)
     Train acc: 0.526, loss: 0.024
     Val   acc: 0.529, loss: 0.026
Epoch 9: 83.463757 (sec)
     Train acc: 0.573, loss: 0.023
     Val   acc: 0.650, loss: 0.019
Best acc: 0.650
Loss:
```

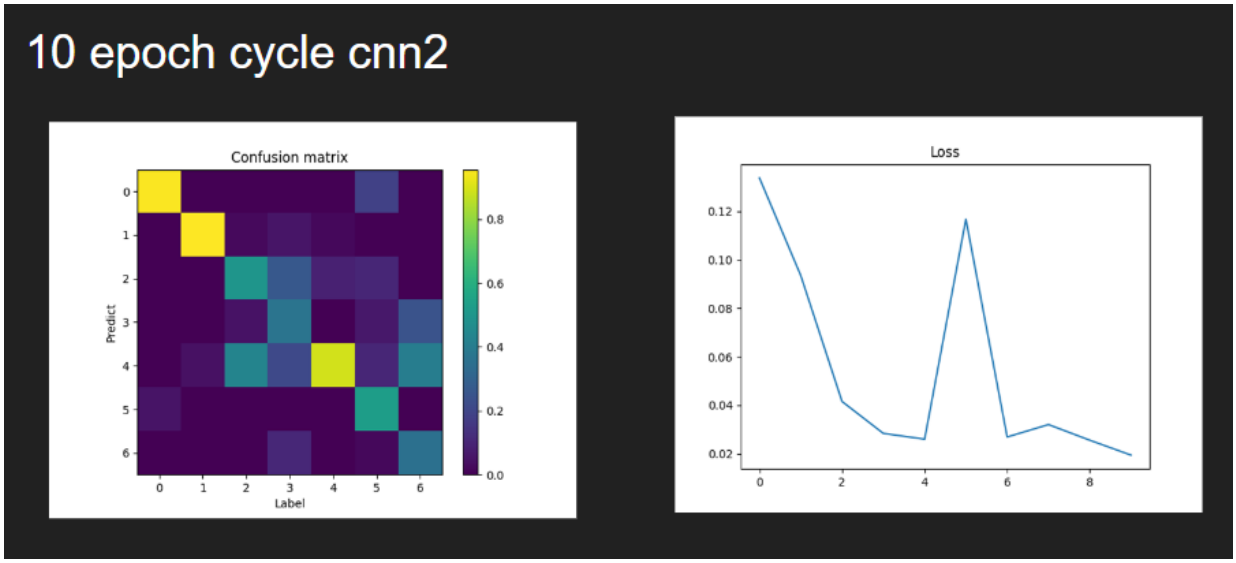


Figure 17

Figure 17 shows that CNN2 model yielded 0.650 best accuracy over an average total time of 13 minutes.

Confusion matrix for figure 17 also shows that there were constant correct predictions for most of the labels with label 0,1 and 4 being the most accurate predictions.

The loss graph shows a constant downward trend until the fourth epoch cycle which is good, but it spikes up to 0.10 on the fifth epoch cycle, while this is good, it still shows that the model is not learning effectively and needs more epoch cycles.

4.3.1.2. CNN1 Model- 10 Epoch cycle

```
Epoch 0: 161.087225 (sec)
     Train acc: 0.306, loss: 0.031
     Val   acc: 0.146, loss: 0.035
Epoch 1: 162.217443 (sec)
     Train acc: 0.421, loss: 0.027
     Val   acc: 0.179, loss: 0.031
Epoch 2: 157.619548 (sec)
     Train acc: 0.423, loss: 0.023
     Val   acc: 0.136, loss: 0.047
Epoch 3: 160.124103 (sec)
     Train acc: 0.473, loss: 0.021
     Val   acc: 0.443, loss: 0.024
Epoch 4: 155.730630 (sec)
     Train acc: 0.519, loss: 0.020
     Val   acc: 0.407, loss: 0.026
Epoch 5: 158.914167 (sec)
     Train acc: 0.533, loss: 0.019
     Val   acc: 0.432, loss: 0.023
Epoch 6: 159.052964 (sec)
     Train acc: 0.575, loss: 0.019
     Val   acc: 0.239, loss: 0.052
Epoch 7: 160.160373 (sec)
     Train acc: 0.564, loss: 0.018
     Val   acc: 0.532, loss: 0.022
Epoch 8: 160.010958 (sec)
     Train acc: 0.620, loss: 0.016
     Val   acc: 0.464, loss: 0.021
Epoch 9: 159.851960 (sec)
     Train acc: 0.637, loss: 0.016
     Val   acc: 0.564, loss: 0.017
Best acc: 0.564
```

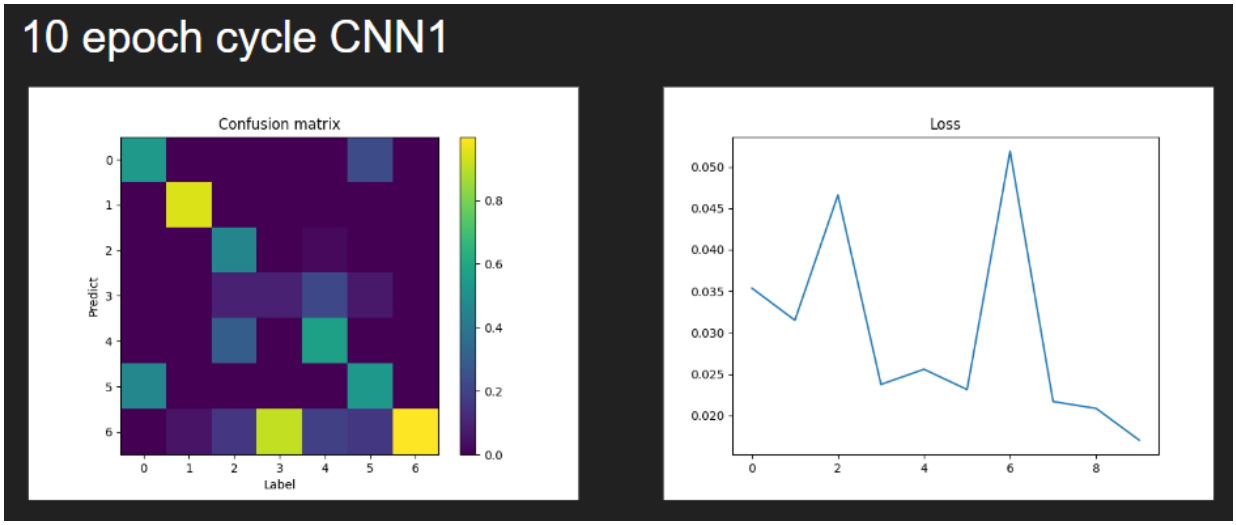


Figure 18

Figure 18 shows that CNN1 Model yielded 0.564 best accuracy over a total average of 23 minutes.

Confusion matrix on figure 18 shows that while label 1 and label 6 had the most accuracy, there were also instances where it predicted wrongly such as label 3 being predicted as 6, overall, not the best accuracy.

The loss is also very volatile and not reliable at all spiking up to 0.045 at second epoch cycle and going down half its value at third epoch cycle and spiking up to 0.50 at the sixth epoch cycle. This shows that the model is not learning effectively.

4.3.2. 30 Epoch Cycle

In machine learning, increasing training duration often leads to increased performance thus the epoch cycle was increased to 30 and both machine learning model CNN1 and CNN2 were trained again to determine if it yielded different results as compared to 10 epoch cycles.

4.3.2.1. CNN2 Model-30 Epoch cycle

```
Epoch 19: 77.078559 (sec)
     Train acc: 0.707, loss: 0.013
     Val   acc: 0.561, loss: 0.029
Epoch 20: 77.667824 (sec)
     Train acc: 0.704, loss: 0.013
     Val   acc: 0.814, loss: 0.009
Epoch 21: 76.873350 (sec)
     Train acc: 0.729, loss: 0.011
     Val   acc: 0.786, loss: 0.010
Epoch 22: 83.015656 (sec)
     Train acc: 0.755, loss: 0.011
     Val   acc: 0.811, loss: 0.009
Epoch 23: 77.824140 (sec)
     Train acc: 0.762, loss: 0.010
     Val   acc: 0.821, loss: 0.009
Epoch 24: 79.340562 (sec)
     Train acc: 0.767, loss: 0.011
     Val   acc: 0.807, loss: 0.009
Epoch 25: 80.576248 (sec)
     Train acc: 0.769, loss: 0.011
     Val   acc: 0.818, loss: 0.009
Epoch 26: 77.021431 (sec)
     Train acc: 0.777, loss: 0.010
     Val   acc: 0.800, loss: 0.009
Epoch 27: 82.796827 (sec)
     Train acc: 0.774, loss: 0.011
     Val   acc: 0.821, loss: 0.009
Epoch 28: 81.447676 (sec)
     Train acc: 0.779, loss: 0.010
     Val   acc: 0.807, loss: 0.009
Epoch 29: 79.544751 (sec)
     Train acc: 0.752, loss: 0.011
     Val   acc: 0.832, loss: 0.008
Best acc: 0.832
```

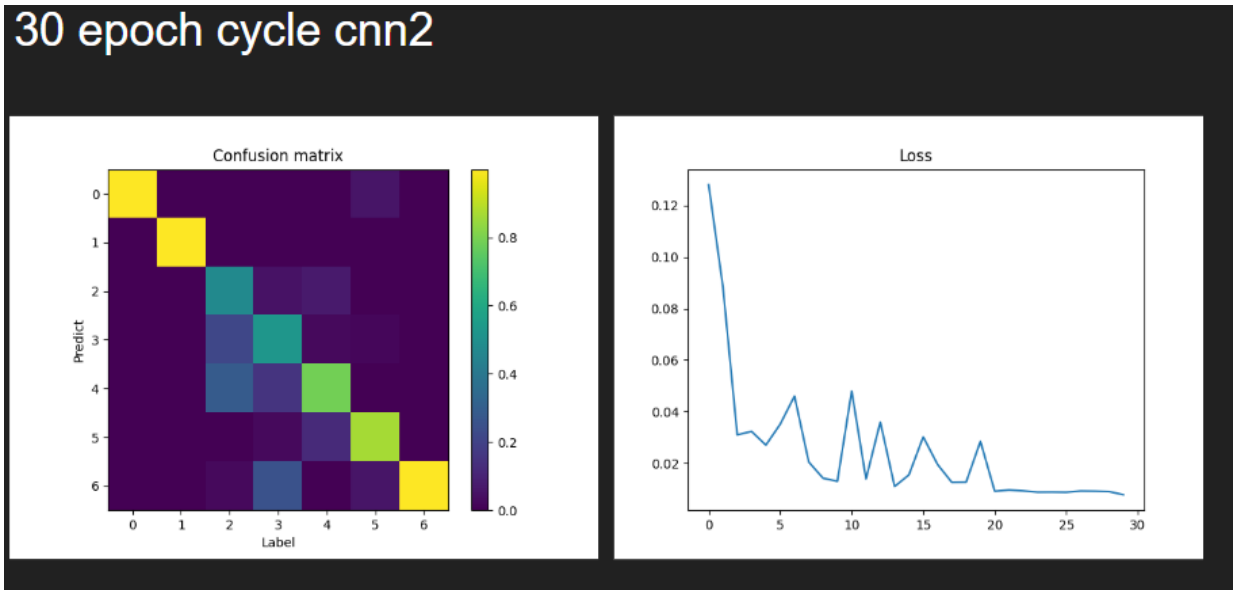


Figure 19

Figure 19 shows that CNN2 accuracy is the best at 0.832 with an average timing of 31 minutes and a huge improvement from the previous 10 epoch cycle's accuracy of 0.650.

The loss remained almost more constant after 20 cycles at 0.01 meaning that the discrepancy between prediction and the actual label is more accurate and the model is learning effectively.

The confusion matrix also shows that most of the labels are being predicted as per their actions with all being above 0.5 and a slight confusion on label 2 and 3.

4.3.2.2. CNN1 Model 30-Epoch cycle

```
Epoch 20: 154.241493 (sec)
     Train acc: 0.751, loss: 0.011
     Val   acc: 0.561, loss: 0.021
Epoch 21: 153.043396 (sec)
     Train acc: 0.752, loss: 0.010
     Val   acc: 0.696, loss: 0.014
Epoch 22: 154.163507 (sec)
     Train acc: 0.755, loss: 0.011
     Val   acc: 0.682, loss: 0.014
Epoch 23: 158.790156 (sec)
     Train acc: 0.764, loss: 0.010
     Val   acc: 0.750, loss: 0.011
Epoch 24: 153.242477 (sec)
     Train acc: 0.758, loss: 0.010
     Val   acc: 0.739, loss: 0.012
Epoch 25: 165.905567 (sec)
     Train acc: 0.781, loss: 0.010
     Val   acc: 0.696, loss: 0.013
Epoch 26: 149.051559 (sec)
     Train acc: 0.764, loss: 0.010
     Val   acc: 0.736, loss: 0.012
Epoch 27: 155.126926 (sec)
     Train acc: 0.781, loss: 0.010
     Val   acc: 0.721, loss: 0.012
Epoch 28: 158.625252 (sec)
     Train acc: 0.765, loss: 0.010
     Val   acc: 0.696, loss: 0.012
Epoch 29: 154.696031 (sec)
     Train acc: 0.754, loss: 0.010
     Val   acc: 0.786, loss: 0.010
Best acc: 0.786
```

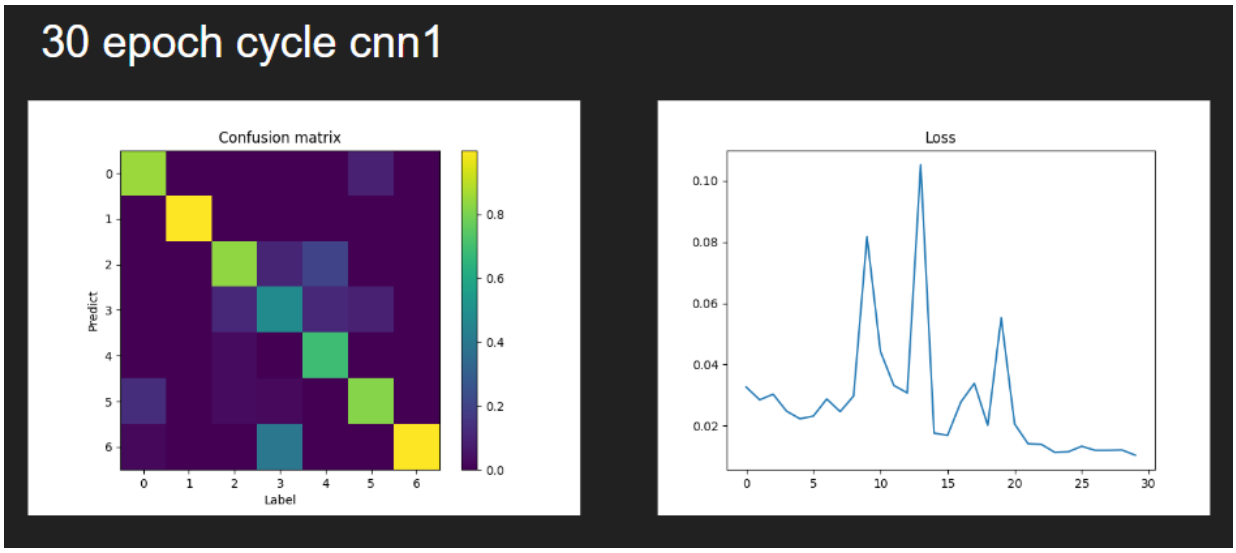


Figure 20

Figure 20 shows that CNN1 has a reading of 0.786 after 30 epoch cycle, while this is still good, it does not justify the amount of time, 93 minutes, as compared to CNN1 which took lesser time and had better overall accuracy.

The loss also does not have a down trend but has spikes going up to as high as 0.10 loss at 14 epoch cycle, this shows that the prediction while being correct, it is still volatile and unreliable and shows that the machine is not learning effectively.

The confusion matrix shows that accuracy for most of the activity labels do come out as predicted at minimum of 0.4 but there are instances like on label 3 where it still predicts it at a rate of 0.5 as activity 6.

As per the results obtained from the confusion matrix and the loss graph of CNN2, it was concluded from the experiments done and selected as the machine learning model to be used for further experimental use.

4.3.2.3. CNN2 Model 60 epoch cycle

```
Epoch 55: 79.817483 (sec)
     Train acc: 0.808, loss: 0.009
     Val   acc: 0.839, loss: 0.009
Epoch 56: 80.686520 (sec)
     Train acc: 0.786, loss: 0.009
     Val   acc: 0.818, loss: 0.009
Epoch 57: 82.830771 (sec)
     Train acc: 0.795, loss: 0.009
     Val   acc: 0.821, loss: 0.009
Epoch 58: 79.988487 (sec)
     Train acc: 0.792, loss: 0.009
     Val   acc: 0.829, loss: 0.009
Epoch 59: 79.266586 (sec)
     Train acc: 0.789, loss: 0.009
     Val   acc: 0.821, loss: 0.009
Best acc: 0.850
```

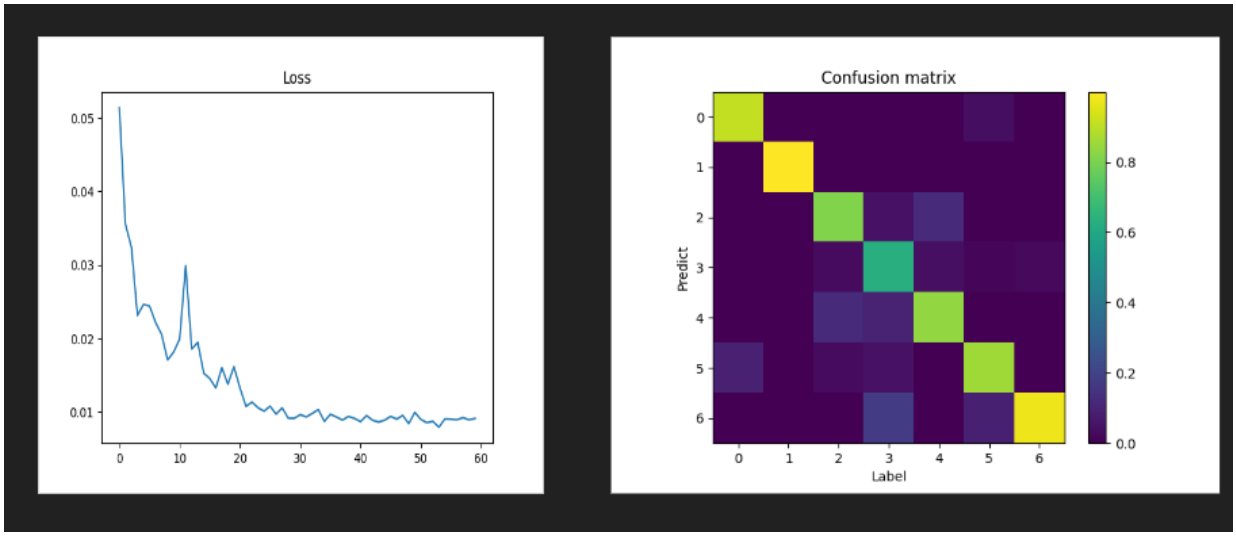


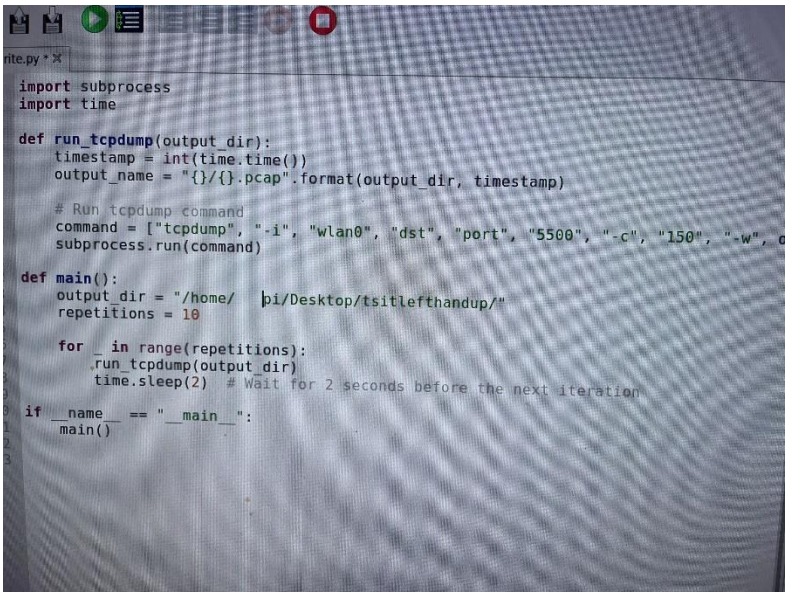
Figure 21

To further evaluate the accuracy of CNN2 model 60 epoch cycles were carried out which resulted in CNN2 model having a best accuracy of 0.850 and a loss of 0.009.

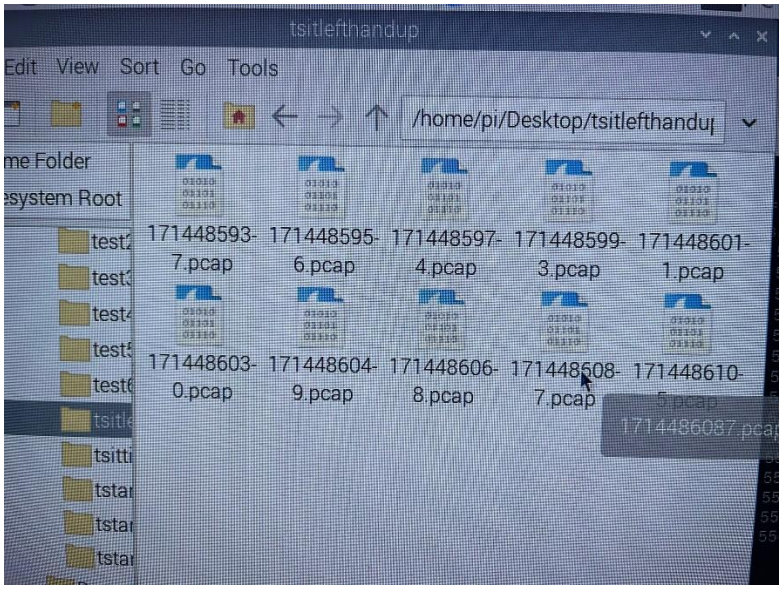
The confusion matrix in figure 21 also shows that most of the activity labels are at least above 0.5 to their prediction meaning that the overall accuracy of the predication is better and there is no label and prediction mismatch above 0.2

4.4. Testing

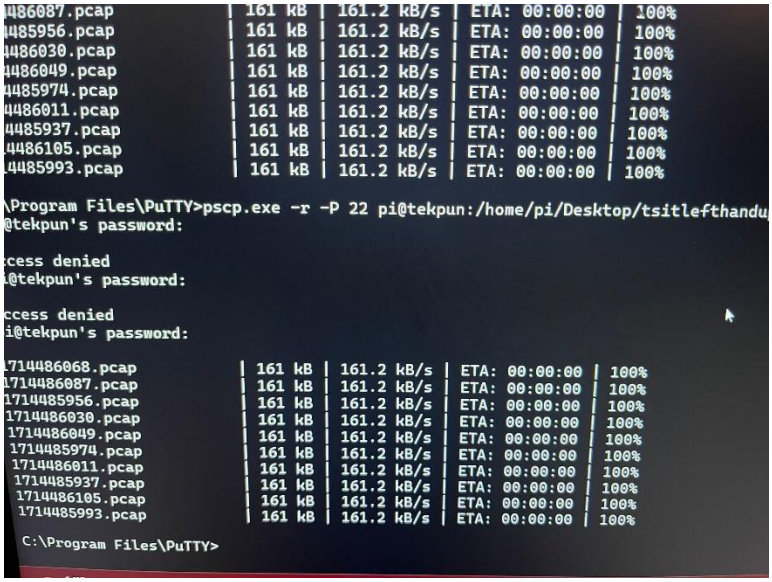
For the test, since the actions were movements and pictures would not show the movement being done, the movement will be uploaded on my blog under testing & prediction. A total of 7 actions were done and each action was collecting 10 sample of pcap files and is then sent to edge server to be processed in CNN2 model for prediction. The step for the process is as follows.



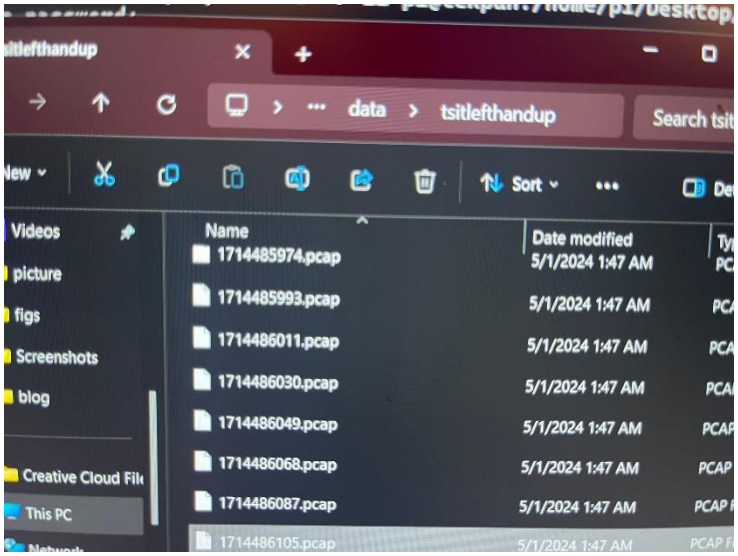
The CSI file will be saved as a {timestamp}.pcap in the desired path and once 150 sample frames of CSI has been collected, it will pause for 2 seconds before carrying out the next collection



10 CSI Sample collected



Edge server data collection is done using putty, cd to the path of where the putty is located and pi@tekpun:<path of file you want to extract> "<path of folder you want the extract folder to be in>"



Edge server file collected.

```
class InferenceServer():
    def inference(self, files, verbose=True):
        with torch.no_grad():
            for idx, (data, label) in enumerate(loader):
                output = self.model(data.to(self.device))
                pred = output.argmax(dim=1).cpu().detach()
                pred_all.extend(pred.tolist())

        end_time = time.time()
        pred_all = [dataset.get_label(p) for p in pred_all]

        if verbose:
            print(f'Predict: {pred_all}')
            print(f'Inference time: {end_time-start_time:.2f} (sec)')

        return pred_all


if __name__ == '__main__':
    server = InferenceServer('data/')
    server.inference('tsitlefthandup/1714486105.pcap')
```

Inputting collected files path into python code, the file path of the csi.pcap that is going to be test has to be written on the server.interfence(<'path of csi'>)

```
Predict: ['empty']
Inference time: 0.12 (sec)
PS C:\Users\tekpun\Desktop\gesture-recognition-csi-master> & c:/Users/tekpun/Desktop/gesture-
rs/tekpun/Desktop/gesture-recognition-csi-master/src/inference_server.py
Predict: ['sitting']
Inference time: 0.12 (sec)
PS C:\Users\tekpun\Desktop\gesture-recognition-csi-master> & c:/Users/tekpun/Desktop/gesture-
rs/tekpun/Desktop/gesture-recognition-csi-master/src/inference_server.py
Predict: ['sitting']
Inference time: 0.11 (sec)
PS C:\Users\tekpun\Desktop\gesture-recognition-csi-master> & c:/Users/tekpun/Desktop/gesture-r
rs/tekpun/Desktop/gesture-recognition-csi-master/src/inference_server.py
Predict: ['standup']
Inference time: 0.10 (sec)
PS C:\Users\tekpun\Desktop\gesture-recognition-csi-master> & c:/Users/tekpun/Desktop/gesture-re
rs/tekpun/Desktop/gesture-recognition-csi-master/src/inference_server.py
Predict: ['standup']
Inference time: 0.10 (sec)
PS C:\Users\tekpun\Desktop\gesture-recognition-csi-master> & c:/Users/tekpun/Desktop/gesture-rec
rs/tekpun/Desktop/gesture-recognition-csi-master/src/inference_server.py
Predict: ['sitting']
Inference time: 0.10 (sec)
PS C:\Users\tekpun\Desktop\gesture-recognition-csi-master> & c:/Users/tekpun/Desktop/gesture-rec
rs/tekpun/Desktop/gesture-recognition-csi-master/src/inference_server.py
Predict: ['sitting']
Inference time: 0.11 (sec)
PS C:\Users\tekpun\Desktop\gesture-recognition-csi-master> & c:/Users/tekpun/Desktop/gesture-recog
rs/tekpun/Desktop/gesture-recognition-csi-master/src/inference_server.py
Predict: ['sittinglefthandup']
Inference time: 0.11 (sec)
PS C:\Users\tekpun\Desktop\gesture-recognition-csi-master> & c:/Users/tekpun/Desktop/gesture-recogn
```

When the python file server_inference is ran, the prediction and inference time is shown. Below is the table to show the results for all the actions and their predictions.

Action	Prediction Out of 10 sample
Clapping	10/10
Empty	9/10

Sitting	8/10
Sitting left hand up	0/10 -6 came out as sitting,2 as empty and 2 as standup
Standing	9/10
Standing right hand up	10/10
Stand up	7/10

Actions such as clapping and standing right hand up received a 100% prediction rate for 10 samples and actions such as stand up, standing, sitting, empty got also above 70% prediction rate and sitting left hand up received a 0% prediction rate. This might be due to having actions that are too similar. As for sitting left hand up action, 6 came out as sitting which shows that the action must be different enough for the signal propagation to differentiate between each action. But in contrast however, standing and standing right hand up was both predicted correctly so it might be other factors such as placement of transmitter and receiver's height level that might be affecting it.

5. Conclusion

While Activity recognition based on Wi-fi CSI data can be successfully used to predict certain activities, the activity performed must be different and diverse and not similar in body structure or movement. As this report was based on only using one antenna on transmitter and receiver, the CSI data that were collected lacked spatial diversity which made it hard to differentiate between signals arriving from different path as compared to having multiple antennas which will enhance beamforming where signals can be focused in specific direction and thus improving signal strength and reducing interface. Multiple-input and multiple-output (MIMO) systems will also have more accurate and precise data as compared to Single-input single-output (SISO) system. Thus, it would have been better to at least have one either the receiver or transmitter to have multiple antennas as compared to both the raspberry pi that was used in this experiment which only has one antenna for receiving and transmitting. Experiments could also have been carried out in different environment to have more accurate dataset, but this could not be done so due to space limitations and living situation and also, instead of having one subject perform the multiple actions 200 times each, having multiple subjects perform the actions multiple times would have made the dataset more diverse and also larger which would result in more training data set, validation set and test set. Less spatial interference would have also been ideal so that signals would be able to properly propagate through the space and there would no disturbance to it thus having lesser noise and more reliable dataset. As for future works, as this project still relies on user to manually collect data to be able to use in

the machine learning for prediction, what could be done instead is to collect CSI data in real-time and then feed this data to the edge server in real time which gives the prediction instantaneously or with a bit of delay fomr the machine learning algorithm and this would also allow user to see the changes in the amplitude and phases when different actions are being made and will be easier to judge if the CSI data collected is unreliable or not. Instead of focusing on human activity recognition only, further enhancements could have been made so that different action would produce different outputs such as having right hand up could light up a LED or standing up could rotate an actuator, these are just small implementations, but it could be developed into a system for a smart home.

Bibliography

[1] S. -C. Kim and Y. -H. Kim, "Efficient classification of human activity using PCA and deep learning LSTM with WiFi CSI," 2022 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Jeju Island, Korea, Republic of, 2022, pp. 329-332, doi: 10.1109/ICAIIC54071.2022.9722627. keywords: {Deep learning;Transforms;Information filters;Feature extraction;Classification algorithms;Wireless fidelity;Channel state information;LSTM;CSI;PCA;Wearable device;RNN;Smart Home;Smart device},

[2] H. Boudlal, M. Serrhini and A. Tahiri, "Design and Deployment of a Practical Wireless Sensing System for HAR with WiFi CSI in the 5GHz Band," 2023 IEEE International Conference on Networking, Sensing and Control (ICNSC), Marseille, France, 2023, pp. 1-6, doi: 10.1109/ICNSC58704.2023.10319031. keywords: {Wireless communication;Wireless sensor networks;Smart homes;Robustness;Sensors;Indoor environment;Communication system security;WiFi signal;Channel State Information;Wireless sensing;Noise elimination;Human Activity Recognition},

[3] Francesco Gringoli, Matthias Schulz, Jakob Link, and Matthias Hollick. 2019. Free Your CSI: A Channel State Information Extraction Platform For Modern Wi-Fi Chipsets. In Proceedings of the 13th International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization (WiNTECH '19). Association for Computing Machinery, New York, NY, USA, 21–28. https://doi.org/10.1145/3349623.3355477

[4] Forbes, Glenn & Massie, Stewart & Craw, Susan. (2020). WiFi-based Human Activity Recognition using Raspberry Pi. 722-730. 10.1109/ICTAI50040.2020.00115.

[5] https://downloads.raspberrypi.org/raspios_armhf/images/raspios_armhf-2022-01-28/

[6] I. Dolińska, M. Jakubowski and A. Masiukiewicz, "Interference comparison in Wi-Fi 2.4 GHz and 5 GHz bands," 2017 International Conference on Information and Digital Technologies (IDT), Zilina, Slovakia, 2017, pp. 106-112, doi: 10.1109/DT.2017.8024280. keywords: {Channel allocation;Interchannel interference;Buildings;Correlation coefficient;Bit rate;Wireless fidelity;interference;MISTI;Wi-Fi;channel assignment},

[7] S. Yousefi, H. Narui, S. Dayal, S. Ermon and S. Valaee, "A Survey on Behavior Recognition Using WiFi Channel State Information," in IEEE Communications Magazine, vol. 55, no. 10, pp. 98-104, Oct. 2017, doi: 10.1109/MCOM.2017.1700082. keywords: {Antennas;Wireless fidelity;OFDM;Behavioral sciences;Wireless communication;Doppler shift;Receivers},

[8] Baha' A. Alsaify, Mahmoud M. Almazari, Rami Alazrai, Mohammad I. Daoud,A dataset for Wi-Fi-based human activity recognition in line-of-sight and non-line-of-sight indoor environments,Data in Brief,Volume 33,2020,106534,ISSN 2352-3409, https://doi.org/10.1016/j.dib.2020.106534.(https://www.sciencedirect.com/science/article/pii/S2352340920314165)Abstract: The aim of this paper is to present a dataset for Wi-Fi-based human activity recognition. The dataset is comprised of five experiments performed by 30 different subjects in three different indoor environments. Keywords: Wi-Fi; Channel State Information (CSI); Human activity recognition (HAR); Non-line-of-sight (NLOS) environment; Line-of-sight (LOS) environment

[9] S. Angra and S. Ahuja, "Machine learning and its applications: A review," 2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC), Chirala, Andhra Pradesh, India, 2017, pp. 57-60, doi: 10.1109/ICBDACI.2017.8070809. keywords: {Handheld computers;Big Data;Computational intelligence;Conferences;Hafnium;Machine Learning;SVM;clustering;feature selection;decision tress;classification;logistic regression},

[10] Z. He, X. Zhang, Y. Wang, Y. Lin, G. Gui and H. Gacanin, "A Robust CSI-Based Wi-Fi Passive Sensing Method Using Attention Mechanism Deep Learning," in IEEE Internet of Things Journal, vol. 10, no. 19, pp. 17490-17499, 1 Oct.1, 2023, doi: 10.1109/JIOT.2023.3275545.keywords: {Sensors;Wireless fidelity;Feature extraction;Convolutional neural networks;Robustness;Internet of Things;Behavioral sciences;Attention mechanism;channel state information (CSI);deep learning (DL);fine-grained sensing;Wi-Fi-based passive sensing},

[11] H. E. Zein, F. Moeurad-Chehade and H. Amoud, "Leveraging Wi-Fi CSI Data for Fall Detection: A Deep Learning Approach," 2023 5th International Conference on Bio-engineering for Smart Technologies (BioSMART), Paris, France, 2023, pp. 1-4, doi: 10.1109/BioSMART58455.2023.10162090. keywords: {Time series analysis;Neural networks;Medical services;Data augmentation;Data models;Convolutional neural networks;Fall detection;Fall detection;Wi-Fi signals;Channel State Information (CSI);Convolutional Neural Networks (CNN);time series data augmentation},

[12] Z. Liu, M. del Rosario and Z. Ding, "A Markovian Model-Driven Deep Learning Framework for Massive MIMO CSI Feedback," in IEEE Transactions on Wireless Communications, vol. 21, no. 2, pp. 1214-1228, Feb. 2022, doi: 10.1109/TWC.2021.3103120.keywords: {Massive MIMO;Wireless communication;Bandwidth;Downlink;Coherence;Fadingchannels;Deeplearning;MassiveMIMO;channel estimation;MarkovNet;compression;model-driven learning},

[13] T. Schmidt, T. Ambegoda and K. Gunasekera, "Vehicle Classification Using Raspberry Pi: A Guide to Capturing WiFi CSI Data," 2023 Moratuwa Engineering Research Conference (MERCon), Moratuwa, Sri Lanka, 2023, pp. 702-707, doi: 10.1109/MERCon60487.2023.10355444. keywords: {Training;Data analysis;Pipelines;Machine learning;Data collection;Planning;Complexity theory;Channel state information (CSI);Raspberry Pi;Nexmon;Vehicle Classification},

[14] Wang, Fei & Han, Jinsong & Zhang, Shiyuan & He, Xu & Huang, Dong. (2018). CSI-Net: Unified Human Body Characterization and Action Recognition

[15]Z. Tang, A. Zhu, Z. Wang, K. Jiang, Y. Li and F. Hu, "Human Behavior Recognition Based on WiFi Channel State Information," 2020 Chinese Automation Congress (CAC), Shanghai, China, 2020, pp. 1157-1162, doi: 10.1109/CAC51589.2020.9326793. keywords: {Wireless fidelity;Wireless communication;OFDM;Receiving antennas;Data mining;Computer science;Channel state information;WiFi;CSI;Behavior Recognition;CNN;GRU},

[16]H. Boudlal, M. Serrhini and A. Tahiri, "Design and Deployment of a Practical Wireless Sensing System for HAR with WiFi CSI in the 5GHz Band," 2023 IEEE International Conference on Networking, Sensing and Control (ICNSC), Marseille, France, 2023, pp. 1-6, doi: 10.1109/ICNSC58704.2023.10319031. keywords: {Wireless communication;Wireless sensor networks;Smart homes;Robustness;Sensors;Indoor environment;Communication system security;WiFi signal;Channel State Information;Wireless sensing;Noise elimination;Human Activity Recognition},

[17] C. Tian et al., "Human Activity Recognition With Commercial WiFi Signals," in IEEE Access, vol. 10, pp. 121580-121589, 2022, doi: 10.1109/ACCESS.2022.3223437.

keywords: {Wireless communication;Sensors;Wireless sensor networks;Wireless fidelity;Fresnel reflection;Receivers;Human activity recognition;Wireless sensing;channel state information;WIFI;dynamic time warping},

[18] J. Xi, Z. Xu, L. Chen and J. Li, "Human Counting and Action Recognition with WiFi via Deep Learning," 2020 Cross Strait Radio Science & Wireless Technology Conference (CSRSWTC), Fuzhou, China, 2020, pp. 1-3, doi: 10.1109/CSRSWTC50769.2020.9372572. keywords: {Deep learning;Wireless communication;Feature extraction;Classification algorithms;Wireless fidelity;Channel state information;Residual neural networks;Human counting;Human action recognition;Deep learning;Channel State Information},

[l9] D. Khan and I. W. -H. Ho, "Deep Learning of CSI for Efficient Device-free Human Activity Recognition," 2021 IEEE 7th World Forum on Internet of Things (WF-IoT), New Orleans, LA, USA, 2021, pp. 19-24, doi: 10.1109/WF-IoT51360.2021.9595661. keywords: {Wireless communication;Deep learning;Location awareness;Wireless sensor networks;Neural networks;Manuals;Activity recognition;Human Activity Recognition (HAR);Channel State Information (CSI);Deep Neural Networks (DNN)},

[20] X. Xie and Z. Guo, "A CNN Approach of Activity Recognition via Channel State Information (CSI)," 2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), Guilin, China, 2019, pp. 329-336, doi: 10.1109/CyberC.2019.00063. keywords: {Conferences;Distributed computing;Knowledge discovery;CNN;CSI;IoT;Gesture-Recognition},

[21] X. B. Maxama and E. D. Markus, "A Survey on Propagation Challenges in Wireless Communication Networks over Irregular Terrains," 2018 Open Innovations Conference (OI), Johannesburg, South Africa, 2018, pp. 79-86, doi: 10.1109/OI.2018.8535598. keywords: {Fading channels;Propagation losses;Conductivity;Wireless communication;Predictive models;Receivers;Interference;Pathloss;fading;Outdoor propagation models;Ionosphere;MIMO;Orthogonal Frequency Division Multiplexing},

[22] J. Choi, "Sensor-Aided Learning for Wi-Fi Positioning With Beacon Channel State Information," in IEEE Transactions on Wireless Communications, vol. 21, no. 7, pp. 5251-5264, July 2022, doi: 10.1109/TWC.2021.3138850.

keywords: {Distance measurement;Wireless fidelity;Trajectory;Feature extraction;Wireless communication;Performance evaluation;Mobile handsets;Indoor positioning;channel state information;neural networks;unsupervised learning;sensor fusion},

Appendix

```python
import numpy as np
import matplotlib.pyplot as plt

from CSIKit.reader import get_reader
from CSIKit.util import csitools
from CSIKit.tools.get_info import display_info

file = "data/empty/1712757697.pcap"
# file = "output_m1.pcap"
# file = "brushteeth_1590158472.dat"


METRIC = "amplitude"
# METRIC = "phase"

my_reader = get_reader(file)
csi_data = my_reader.read_file(file)
csi_matrix, no_frames, no_subcarriers = csitools.get_CSI(csi_data, metric=METRIC)
timestamps = csi_data.timestamps

metadata = csi_data.get_metadata()

print(no_frames, no_subcarriers)
print(csi_matrix.shape)


if len(csi_matrix.shape) == 4:
    # if more than 1Tx, 1Rx, use the first Tx, Rx
    csi_matrix = np.transpose(csi_matrix[:, :, 0, 0])
else:
    csi_matrix = np.transpose(csi_matrix)
```

```python

csi_matrix = np.clip(csi_matrix, -20, csi_matrix.max())  # clip the small values

x = [t-timestamps[0] for t in timestamps]
print(csi_matrix.shape)
plt.figure(figsize=(8, 3))
plt.subplots_adjust(left=0.1, right=1, bottom=0.2)

plt.imshow(csi_matrix, extent=[0, max(x), 1, no_subcarriers], aspect='auto', cmap='jet')

plt.xlabel("Time (s)")
plt.ylabel("Subcarrier Index")

plt.title(csi_data.filename)


if METRIC == "amplitude":
    cbar = plt.colorbar()
    cbar.set_label('Amplitude (dBm)')
else:
    cbar = plt.colorbar(ticks=[-np.pi, 0, np.pi])
    cbar.set_label('Phase')
    cbar.set_ticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi], labels=[r'$-\pi$', r'$-\frac{\pi}{2}$', 0, r'$\frac{\pi}{2}$', r'$\pi$'])

plt.show()


```

Csitest.py

```python
import glob
import os

from CSIKit.reader import NEXBeamformReader
from CSIKit.util import csitools
import torch
from torch.utils.data import Dataset
from torchvision import transforms

class CsiDataSet(Dataset):
    def __init__(self, root='C:/Users/tekpun/Desktop/tekpfyp/data/', files=None):

        self.root = root
        self.file = []
        self.label = []
        self.reader = NEXBeamformReader()
        self.labels = ['clap', 'empty', 'sitting','sittinglefthandup', 'standing', 'standingrighthandup','standup']
        self.transform = None

        if files is None:
            """ Training """

            for idx, label in enumerate(self.labels):
                files = [os.path.relpath(i, root) for i in glob.glob(f'{root}/{label}/*.pcap')]
                # Debug code: print file names and their corresponding numeric parts
                for file in files:
                    basename = os.path.splitext(os.path.basename(file))[0]
                    numeric_part = ''.join(filter(str.isdigit, basename))
                    print(f"File: {file}, Numeric part: {numeric_part}")

                files.sort(key=lambda x: int(''.join(filter(str.isdigit, os.path.splitext(os.path.basename(x))[0]))))
```

```python
                self.file.extend(files)
                self.label.extend([idx for _ in files])
                self.transform = transforms.RandomAffine(0, translate=(0.3, 0))
        else:
            """ Inferencing """
            if type(files) is list:
                self.file.extend(files)
            else:
                self.file.append(files)

    def __len__(self):
        return len(self.file)

    def __getitem__(self, index):
        file = os.path.join(self.root, self.file[index])

        csi_data = self.reader.read_file(file)
        csi_matrix, _, _ = csitools.get_CSI(csi_data, metric='amplitude')

        csi_matrix = torch.from_numpy(csi_matrix).float()
        csi_matrix = csi_matrix[:, 64:192, 0, 0].T  # shape 128*150
        csi_matrix = csi_matrix.clamp(min=-20)  # clamp the small values
        csi_matrix = csi_matrix.reshape(1, 128, 150)
        #print("Shape before reshape:", csi_matrix.shape)
        #csi_matrix = csi_matrix.unsqueeze(0)
        #print("Shape after reshape:", csi_matrix.shape)
```

```python
        if self.transform is not None:
            csi_matrix = self.transform(csi_matrix)

        if self.label:
            return csi_matrix, self.label[index]
        else:
            return csi_matrix, -1

    def get_label(self, idx):
        return self.labels[idx]
```

Dataset.py

```python
import time

import torch
from torch.utils.data import DataLoader

from dataset import CsiDataSet
import models

class InferenceServer():
    def __init__(self, root):

        self.root = root
        model_path = 'models/model-4.pkl'

        self.device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
        self.model = models.CNN2()
        self.model.load_state_dict(torch.load(model_path, map_location=torch.device(self.device)))
        self.model.to(self.device)
        self.model.eval()

    def inference(self, files, verbose=True):
        dataset = CsiDataSet(root=self.root, files=files)
        loader = DataLoader(dataset, batch_size=1, shuffle=False)
        pred_all = []

        start_time = time.time()

        with torch.no_grad():
            for idx, (data, label) in enumerate(loader):
                output = self.model(data.to(self.device))
                pred = output.argmax(dim=1).cpu().detach()
                pred_all.extend(pred.tolist())

        end_time = time.time()
        pred_all = [dataset.get_label(p) for p in pred_all]

        if verbose:
            print(f'Predict: {pred_all}')
            print(f'Inference time: {end_time-start_time:.2f} (sec)')

        return pred_all


if __name__ == '__main__':
    server = InferenceServer('data/')
    server.inference('tsitlefthandup/1714486105.pcap')

```

Inference_server.py

```python
import torch
import torch.nn as nn

class CNN2(nn.Module):
    def __init__(self, num_classes=7):
        super(CNN2, self).__init__()

        self.cnn = nn.Sequential(
            nn.Conv2d(1, 32, 3, 1, 1),
            nn.BatchNorm2d(32),
            nn.ReLU(),

            nn.Conv2d(32, 64, 3, 1, 1),
            nn.BatchNorm2d(64),
            nn.ReLU(True),
            nn.MaxPool2d(2, 2, 0),

            nn.Conv2d(64, 128, 3, 1, 1),
            nn.BatchNorm2d(128),
            nn.ReLU(True),
            nn.MaxPool2d(2, 2, 0),
            nn.Dropout(0.2),

            nn.Conv2d(128, 128, 3, 1, 1),
            nn.BatchNorm2d(128),
            nn.ReLU(True),
            nn.MaxPool2d(2, 2, 0),
            nn.Dropout(0.2),
        )

        self.fc = nn.Sequential(
            nn.Linear(128*16*18, 256),
            nn.Dropout(0.3),
            nn.Linear(256, 64),
            nn.Dropout(0.25),
            nn.Linear(64, num_classes)
        )

    def forward(self, x):
        out = self.cnn(x)
        feature = torch.flatten(out, 1)
        out = self.fc(feature)
        return out

class CNN1(nn.Module):
    def __init__(self, num_classes=7):
        super(CNN1, self).__init__()

        self.cnn = nn.Sequential(
            nn.Conv2d(1, 32, 3, 1, 1),
            nn.BatchNorm2d(32),
            nn.ReLU(),

            nn.Conv2d(32, 64, 3, 1, 1),
            nn.BatchNorm2d(64),
            nn.ReLU(True),

            nn.Conv2d(64, 128, 3, 1, 1),
            nn.BatchNorm2d(128),
```

```
            nn.ReLU(True),
            nn.MaxPool2d(2, 2, 0),
            nn.Dropout(0.2),

            nn.Conv2d(128, 192, 3, 1, 1),
            nn.BatchNorm2d(192),
            nn.ReLU(True),
            nn.Dropout(0.2),

            nn.Conv2d(192, 64, 3, 1, 1),
            nn.BatchNorm2d(64),
            nn.ReLU(True),
            nn.Dropout(0.2),

            nn.AdaptiveAvgPool2d((1, 1))
        )

        self.fc = nn.Sequential(
            nn.Linear(64, 32),
            nn.Dropout(0.25),
            nn.Linear(32, num_classes)
        )

    def forward(self, x):
        out = self.cnn(x)
        feature = torch.flatten(out, 1)
        out = self.fc(feature)
        return out

```

Models.py

```python
import time

import torch
import torch.nn as nn
from torch.utils.data import DataLoader, random_split

import models
import utils
from dataset import CsiDataSet


def train():

    EPOCH = 60
    ID = '6'

    dataset = CsiDataSet(root='data/')

    val_n = int(len(dataset)*0.2)
    test_n = int(len(dataset)*0.2)
    train_n = len(dataset) - val_n - test_n
    print(f'Training set: {train_n}, Validation set: {val_n}, Testing set: {test_n}')
    train_set, val_set, test_set = random_split(dataset, [train_n, val_n, test_n])

    train_loader = DataLoader(train_set, batch_size=64, shuffle=True, num_workers=2)
    val_loader = DataLoader(val_set, batch_size=64, shuffle=True, num_workers=2)
    test_loader = DataLoader(test_set, batch_size=64, shuffle=False, num_workers=2)

    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

    model = models.CNN1()
```

```python
    print(model)

    optimizer = torch.optim.AdamW(model.parameters(), lr=0.001)
    scheduler = torch.optim.lr_scheduler.StepLR(optimizer, 20, gamma=0.1)

    model = model.to(device)
    criterion = nn.CrossEntropyLoss()

    loss_list = []
    acc_list = []

    for epoch in range(EPOCH):
        start_time = time.time()

        train_acc = 0.
        train_loss = 0.
        val_acc = 0.
        val_loss = 0.

        model.train()

        for idx, (data, label) in enumerate(train_loader):
            optimizer.zero_grad()
            output = model(data.to(device))
            batch_loss = criterion(output, label.to(device))
            batch_loss.backward()
            optimizer.step()

            pred = output.argmax(dim=1).cpu().detach()
            train_acc += pred.eq(label).sum().item()
```

```
            train_loss += batch_loss.item()

        model.eval()
        with torch.no_grad():
            for idx, (data, label) in enumerate(val_loader):
                output = model(data.to(device))
                batch_loss = criterion(output, label.to(device))

                pred = output.argmax(dim=1).cpu().detach()
                val_acc += pred.eq(label).sum().item()
                val_loss += batch_loss.item()

        scheduler.step()

        train_acc = train_acc/len(train_set)
        train_loss = train_loss/len(train_set)
        val_acc = val_acc/len(val_set)
        val_loss = val_loss/len(val_set)

        loss_list.append(val_loss)
        acc_list.append(val_acc)

        end_time = time.time()
        print(f'Epoch {epoch}: {end_time-start_time:2f} (sec)')
        print(f'      Train acc: {train_acc:.3f}, loss: {train_loss:.3f}')
        print(f'      Val   acc: {val_acc:.3f}, loss: {val_loss:.3f}')

    print(f'Best acc: {max(acc_list):.3f}')
    print('Loss:')
    for loss in loss_list:
        print(loss, end=' ')
    print(' ')

    torch.save(model.state_dict(), f'models/model-{ID}.pkl')
    utils.plotlist(loss_list, f'figs/loss-{ID}.png', 'Loss')

    # inference

    pred_all = []
    label_all = []
    test_acc = 0.
    start_time = time.time()

    model.eval()
    with torch.no_grad():
        for idx, (data, label) in enumerate(test_loader):
            output = model(data.to(device))
            pred = output.argmax(dim=1).cpu().detach()
            test_acc += pred.eq(label).sum().item()

            pred_all.extend(pred.tolist())
            label_all.extend(label.tolist())
    end_time = time.time()
    test_acc = test_acc/len(test_set)

    print(f'Inference time: {end_time-start_time:.2f} (sec)')
    print(f'Inference acc: {test_acc:.3f}')

    utils.confusion(pred_all, label_all, f'figs/conf-{ID}.png', num_classes=7)


if __name__ == '__main__':
    train()

```

Train.py

```python
from CSIKit.reader import NEXBeamformReader
from CSIKit.util import csitools
import numpy as np
import matplotlib.pyplot as plt
import torch

def read_file(file):
    print(file)
    csi_data = NEXBeamformReader().read_file(file)
    csi_matrix, no_frames, no_subcarriers = csitools.get_CSI(csi_data, metric='amplitude')

    print(csi_data.get_metadata())

    csi_matrix = torch.from_numpy(csi_matrix)
    csi_matrix = csi_matrix[:, 64:192, 0, 0].T  # shape 128*150
    csi_matrix = csi_matrix.clamp(min=-20)  # clamp the small values
    print(csi_matrix.size())

    plot_data(csi_matrix, title=file)

def plot_data(csi_matrix, title=''):
    plt.figure(figsize=(8, 3))
    plt.subplots_adjust(left=0.1, right=1, bottom=0.2)

    plt.imshow(csi_matrix, aspect='auto', cmap='inferno')

    plt.xlabel("Time (s)")
    plt.ylabel("Subcarrier Index")
    plt.title(title)

    cbar = plt.colorbar()
    cbar.set_label('Amplitude (dBm)')

    plt.show()

def plotlist(plotlist, path, title):
    plt.plot(plotlist)
    plt.title(title)
    plt.savefig(path)
    plt.close()

def confusion(pred, label, path, num_classes):
    confusion = np.zeros((num_classes, num_classes))
    for o, t in zip(pred, label):
        confusion[o, t] += 1
    confusion = confusion/(np.sum(confusion, axis=0)+1e-6)

    plt.imshow(confusion)
    plt.colorbar()
    plt.title('Confusion matrix')
    plt.xlabel('Label')
    plt.ylabel('Predict')
    plt.savefig(path)
    plt.close()
```

Utils.py

```python
import subprocess
import time

def run_tcpdump(output_dir):
    timestamp = int(time.time())
    output_name = "{}/{}.pcap".format(output_dir, timestamp)

    # Run tcpdump command
    command = ["tcpdump", "-i", "wlan0", "dst", "port", "5500", "-c", "150", "-w", output_name]
    subprocess.run(command)

def main():
    output_dir = "/home/pi/Desktop/test/"
    repetitions = 10

    for _ in range(repetitions):
        run_tcpdump(output_dir)
        time.sleep(2)  # Wait for 2 seconds before the next iteration

if __name__ == "__main__":
    main()
```

Tcpwrite.py