Bachelor of Science in Computer Science & Engineering

A CSI-based Position Independent Gesture Recognition System to Operate Smart Home Appliances

by

Zerin Shaima Meem

ID: 1804057

Department of Computer Science & Engineering

Chittagong University of Engineering & Technology (CUET)

Chattogram-4349, Bangladesh.

May, 2024

A CSI-based Position Independent Gesture Recognition System to Operate Smart Home Appliances

Submitted in partial fulfilment of the requirements for
Degree of Bachelor of Science
in Computer Science & Engineering

by

Zerin Shaima Meem

ID: 1804057

Supervised by

Dr. Mahfuzulhoq Chowdhury

Professor

Department of Computer Science & Engineering

Chittagong University of Engineering & Technology (CUET)

Chattogram-4349, Bangladesh.

The thesis titled '**A CSI-based Position Independent Gesture Recognition System to Operate Smart Home Appliances**' submitted by ID: 1804057, Session 2021-2022 has been accepted as satisfactory in fulfilment of the requirement for the degree of Bachelor of Science in Computer Science & Engineering to be awarded by the Chittagong University of Engineering & Technology (CUET).

Board of Examiners

Chairman

Dr. Mahfuzulhoq Chowdhury
Professor
Department of Computer Science & Engineering
Chittagong University of Engineering & Technology (CUET)

Member (Ex-Officio)

Dr. Abu Hasnat Mohammad Ashfak Habib
Professor & Head
Department of Computer Science & Engineering
Chittagong University of Engineering & Technology (CUET)

Member (External)

Mohammad Obaidur Rahman
Associate Professor
Department of Computer Science & Engineering
Chittagong University of Engineering & Technology (CUET)

Declaration of Originality

This is to certify that I am the only author of the thesis and that neither it nor any of its components have ever been submitted to another university for credit toward a degree.

I attest that, to the best of my knowledge, my thesis does not violate anyone's copyright or proprietary rights, and that all concepts, methods, quotes, or other content derived from other people's work that I have included in my thesis—whether published or unpublished—have been properly cited in line with accepted referencing guidelines. I understand that I could face legal and disciplinary action from the Department of CSE, CUET, if any intentional or unintentional copyright infringement is discovered.

I hereby assign all rights to the Department of CSE, CUET, regarding the copyright of this thesis work. Department of CSE, CUET will be the owner of this copyright, and any use or reproduction of this work in any way is forbidden without the Department of CSE, CUET's consent.

Signature of the candidate

Date:

Acknowledgements

First and foremost, I extend my deepest gratitude to the Almighty for the strength and perseverance granted to me throughout this journey.

I am profoundly indebted to my supervisor, Dr. Mahfuzulhoq Chowdhury, Professor in the Department of Computer Science and Engineering at Chittagong University of Engineering and Technology (CUET). His unwavering support and insightful guidance have been pivotal during the most challenging phases of my research. Dr. Chowdhury's invaluable advice and encouragement provided me with the fortitude to navigate through moments of doubt and uncertainty. I am immensely thankful for his dedication and the significant role he has played in my academic growth.

I am also glad to extend my gratitude to my former supervisor, Dr. Asaduzzaman, Professor in the Department of Computer Science and Engineering at CUET. He introduced me to the intriguing field of Wi-Fi sensing and encouraged me to delve deeper into this area of research. His ability to maintain a cool demeanor has been a steadying influence, inspiring calmness, and clear thinking even in the most critical phases of my research.

I consider myself exceedingly fortunate to have had such exemplary guidance from both of my supervisors.

Abstract

Gesture recognition is becoming increasingly important for human-computer interaction (HCI) and smart home automation. Traditional gesture recognition systems rely on computer vision, sensors, or WiFi signals. However, computer vision systems require good lighting and direct visibility, which can raise privacy concerns. Sensor-based systems often require additional infrastructure or wearable devices, making them costly and intrusive for large-scale use. In contrast, WiFi sensing-based systems are gaining interest among researchers because they don't need extra infrastructure or wearable devices.

Previous WiFi sensor-based gesture recognition systems typically work from fixed positions, limiting their practicality in real-world scenarios where users interact from various locations within a room. For instance, deploying a gesture recognition system in a smart environment to control appliances requires the ability to recognize gestures from anywhere in the room, not just specific positions.

One existing method, proposed by Kazuya Ohara et al in [1], proposes a position-independent gesture recognition system using WiFi channel state information (CSI) from a smartphone carried by the user. They focused on hand movement velocity extracted from CSI data to classify five gestures with 68.7% accuracy.

In this thesis, we introduce a CSI-based position-independent gesture recognition system. We collected CSI data using a pair of ESP-32 microcontrollers. We implied Machine Learning approaches like Random Forest, Extreme Gradient Boost, Light-GBM, and Shapelet Learning method. We have also implemented CNN and LSTM Deep Learning approaches in the training data, that we collected from nine positions in a laboratory room and tested it on random positions. Our system accurately classified four gestures using Amplitude sub-carriers as features, achieving an impressive accuracy of 81.11% with the LSTM model. This performance surpasses many existing methods and demonstrates the effectiveness of our approach for real-world applications.

Keywords: CSI, WiFi, HCI, Position Independent Gesture Recognition

Table of Contents

List of Figures

List of Tables

List of Abbreviations

CFR Channel Frequency Response. 11

CSI Channel State Information. 2, 10, 59

DWT Discrete Wavelet Transform. 14, 15

HCI Human Computer Interaction. 1, 58

IoT Internet of Things. 1

RSS Received Signal Strength. 1

RSSI Received Signal Strength Indicator. 2

STFT Short Time Fourier Transform. 15

Chapter 1

Introduction

1.1 Introduction

Automated control and administration of networked home appliances have drawn interest due to the expansion of IoT in smart home technologies. Recently, many researchers have been researching gesture recognition techniques using various types of device-based and device-free sensors to achieve simpler and more intuitive control over networked appliances.

Gesture recognition is a technology that greatly simplifies HCI (Human Computer Interaction) by automating the work of recognizing human activities in device-based or device-free sensing systems. These gesture recognition techniques mainly fall into two groups: Computer vision-based and Wearable sensor-based. Some sensor-based techniques, like those proposed in [2] by D. De, P. Bharti et al. need users to wear gadgets constantly, which can be inconvenient in real-time use. Computer vision-based methods need a line of sight and good lighting, which may violate user privacy, as in the method proposed by H. Mliki et al. in [3], or J. Shotton et al. in [4].

In contrast, Wifi signal-based gesture detection systems have drawn a lot of interest as they protect user privacy and offer diverse applications. The extensive deployment of WiFi infrastructure in indoor spaces, such as workplaces, laboratories, and various indoor areas, has created highly favorable conditions for indoor wireless sensing. Other advantages of WiFi-based solutions include the availability of hands-free input, which is especially useful when users' hands are filthy, damp, busy, or covered in gloves, which makes touch input difficult. Among many ways of Wi-Fi sensing, Received Signal Strength (RSS) and Channel State

Information (CSI) are the two WiFi signals that are most frequently used for gesture recognition.

H. Abdelnasser at el. used RSSI signal in his proposed method[5]. Human movement causes reflection or changes in the received signal strength within the operating range of the transmitter and receiver, which is recorded as the RSSI value. As the signal only carries coarse-grained information, it is simple to extract this value from any device, but it displays less precision in gesture detection.

On the other hand, Gesture recognition using CSI data utilizes Wi-Fi signal propagation properties to recognize and comprehend hand gestures. CSI records the signal's amplitude, phase, and signal reflections from human movement at the sub-carrier level. This facilitates fine-grained tracking and hence demonstrates a phenomenal performance. CSI data is resistant to all environmental and anthropogenic changes. Additionally, it generates excellent accuracy with fewer user profiles, as in the method proposed by H. F. T. Ahmed et al. in [6].

1.2 System/ Design Overview

Our proposed work is illustrated in fig. 1.1 and involves a systematic approach structured around several key phases that follow: (1) Establishing experimental setup, (2) Collecting raw CSI train and test data, (3) CSI data parsing, (4) Preprocessing and Labeling the extracted data, (6) Training the MLand DL models (7) Gesture classification, (8) Comparing among the results of different models.

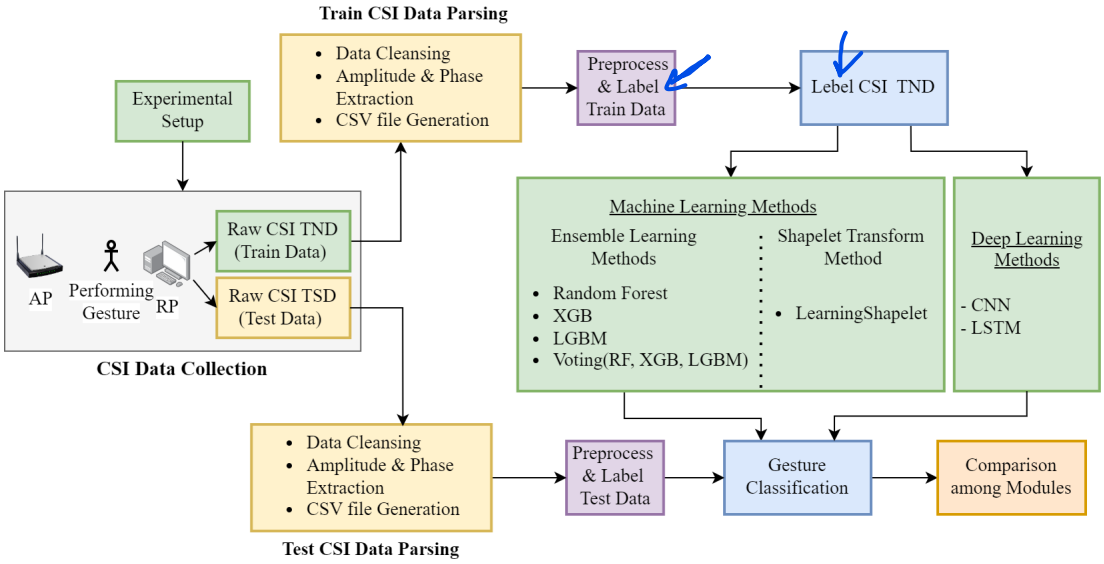


Figure 1.1: Overview of the proposed system

We established a controlled experimental setup for collecting Channel State Information (CSI) data using ESP-32 devices. From nine fixed positions, we gathered four types of gestures for the training set, and from different random positions, we collected data for testing. We parsed the CSI data into amplitude and phase values for classification using both Machine Learning and Deep Learning models. Additionally, we explored shapelet transformation and employed the LearningShapelet model for gesture classification. Chapter 3 will provide a detailed discussion of our methodology and its successful application.

1.3 Difficulties

In the development of our proposed system, we encountered several significant challenges, starting from the initial configuration of ESP32 microcontrollers to the training of our models. Below, we outline some of the key obstacles we faced:

1. **Configuring ESP32 Microcontrollers**: Our first challenge was to master the configuration and operation of ESP32 microcontrollers alongside their respective devices, which was crucial for understanding the hardware environment.

2. **Establishing a Controlled Environment**: Creating a controlled environment was necessary to minimize noise interference in our dataset.

3. **Preprocessing Raw Data**: Preprocessing the raw data was quite challenging. Particularly removing null subcarriers and filtering high-throughput signals within a CSV file, required intricate implementation.

4. **Parsing CSI Data**: Parsing CSI data and transforming it into amplitude and phase values, then saving it in CSV format, was another significant challenge due to the complexity of the data structure.

5. **Training Dataset Size and Reshaping**: Training the extracted datasets proved challenging, primarily due to the substantial reduction in dataset size post-data cleaning. Additionally, adapting the dataset to fit the requirements of different models demanded various reshaping efforts.

Passing through these challenges required innovative problem-solving strategies and a deep understanding of both the technical aspects and the broader objectives of our project.

1.4 Applications

In this thesis, we concentrate on developing a CSI-based position-independent gesture recognition system. Our approach involves collecting CSI data from actors performing four distinct gestures within a WiFi network using ESP32 microcontrollers. Through data processing and model evaluation, our primary aim is to create a system for operating smart home appliances. However, the potential applications extend beyond smart homes and include:

1. **Smart Home Automation**: Enables users to control appliances with hand gestures, enhancing convenience.

2. **Healthcare Monitoring**: Facilitates remote patient monitoring for movement analysis and rehabilitation tracking.

3. **Human-Computer Interaction (HCI)**: Enhances user experience by enabling intuitive gesture-based interactions with smart devices.

4. **Security Systems**: Adds biometric authentication to access control and surveillance systems.

5. **Virtual and Augmented Reality (VR/AR)**: Enables natural interaction with virtual environments and objects.

6. **Automotive Interfaces**: Integrates gesture controls for in-car systems, improving safety and convenience.

These applications offer opportunities for innovation and improved user experiences across various industries.

1.5 Objectives

The objective of this work is to present a real-life application of Gesture Recognition using CSI data. The proposed system will classify among position-independent gestures and use this technique to operate smart home appliances. The key objective and possible outcomes of this work are mentioned below:

1. To design a position-independent gesture recognition system for practical uses like operating smart home appliances.

2. To classify CSI data of four different gestures, using different machine learning and deep learning models.

3. To compare the performance of machine learning models like Random Forest, XGBoost, LGBM, and Shapelet Learning; and Deep Learning models like CNN and LSTM; for these types of sequential data.

1.6 Motivation

With the advancement of technology, computing has started to be embedded in our daily lives and environments in various forms, and smart appliances are one of them. Human-computer interaction (HCI) plays a crucial role in facilitating seamless communication between individuals and devices. Human gestures offer a natural means of interaction, reflecting how humans naturally communicate and engage with their surroundings.

Traditional gesture recognition systems often rely on cameras or wearable sensors, which can be cumbersome or invasive to users. WiFi-based gesture recognition

systems have gained popularity due to their simplicity and low deployment costs. Leveraging channel state information (CSI), these systems detect changes in wireless signals as users perform gestures, allowing for gesture recognition without the need for additional sensors or cameras.

However, existing WiFi-based systems have limitations, particularly regarding user position variability. Prior systems struggle to maintain accuracy when users change positions, limiting their practical usability in real-life scenarios such as operating smart home appliances.

To address this challenge, we propose a CSI-based position-independent gesture recognition system. This system aims to recognize gestures accurately regardless of the user's position, enhancing usability in various real-life applications, including smart home automation.

1.7 Contribution of the thesis

In this thesis, our goal is to classify various gestures from different positions within a room. For classification, we used CSI data and applied machine learning and deep learning approaches. The primary goal of this thesis was to develop a better position-independent gesture classification system employing ESP32 microcontroller WiFi connections and CSI data. The following is this thesis' main contribution:

1. Collecting CSI data Using a single pair of ESP-32 micro-controller.
2. Collecting a total of 380 sets of data, which contain a total of 2,63,660 data packets.
3. Analyzing the impact of Amplitude and Phase sub-carriers in classifying CSI data.
4. Comparing different ensemble learning methods like Extreme Gradient Boosting(XGB), Light Gradient Boosting Model(LGBM), and Random Forest. And then ensemble them to predict the best result.

5. Implementing a shapelet-learning method to utilize the time series property of our CSI data.
6. Analyzing the performance of a one-dimensional convolutional network for sequential data.
7. Developing an LSTM model, to utilize the short-term memory property of that model in our sequential data.
8. Evaluating all these machine learning and deep learning techniques mentioned above to show the association between the changing CSI data and the motions and classify the accurate gesture.
9. Proposing a position-independent gesture recognition system.

1.8 Thesis Organization

The subsequent sections of this thesis report are structured as follows:

- **Literature Review**: Chapter 2 offers a succinct overview of previous research endeavors focused on gesture classification using CSI data from WiFi networks. This chapter discusses the outcomes and constraints of these studies.
- **Methodology**: Chapter 3 outlines the proposed methodology for classifying position-independent gestures. It includes detailed diagrams and explanations of the hardware, experimental setup, CSI data collection, data preprocessing techniques, and model implementation.
- **Results & Discussion**: Chapter 4 delves into the experimental results and visualization of collected CSI data. It examines the phase and amplitude of CSI data, as well as subcarrier information. Additionally, this chapter provides a brief comparison of deep learning, machine learning, and shapelet learning models.
- **Conclusion**: Chapter 5 presents the conclusion of this work, offering an overall summary of our system. It also acknowledges limitations and suggests avenues for future research.

1.9 Conclusion

In this chapter, we give a simple overview of our Position-Independent Gesture Classification System. We explained Channel State Information (CSI) and WiFi sensing technology, which is the base of our project. We also talk about why we're doing this project and what we hope to achieve, as well as the challenges we've faced.

Looking ahead, the next chapter will dig deeper into the background information. We'll explain the important stuff behind our project, like why gesture recognition matters and what others have done in this area before us. This will help you understand why our work is important and how it fits into the bigger picture.

Chapter 2

Literature Review

2.1 Introduction

In fields like smart home, health tracking, and safety monitoring, gesture detection is becoming more and more crucial. Because of the rapid growth in human-computer interaction recently, there has been a lot of interest in this technology. Computer vision and wearable sensor systems are the two main approaches for recognizing gestures in recent years.

Among these approaches, sensor-based gesture recognition methods can be inconvenient as they need users to wear a gadget continuously. On the other hand, computer-vision-based methods often need a clear line of sight and specific lighting conditions, posing potential privacy concerns for users.

In contrast, gesture recognition systems based on commercial Wi-Fi have gained attention due to their unique advantages. They don't require users to wear specific devices all the time, and they don't rely on direct line-of-sight or specific lighting conditions, which helps to preserve user privacy. This makes Wi-Fi-based gesture recognition systems more flexible and user-friendly compared to other methods.

CSI-based human activity recognition technologies have emerged as a prominent area of research. These technologies leverage CSI data for various applications, including daily behavior recognition, fall detection, breath detection, sleep monitoring, and user identification. By analyzing subtle changes in CSI patterns, researchers can extract valuable insights about human activities and behaviors, paving the way for innovative applications in smart environments and health monitoring.

In this chapter, we will go through all these WiFi sensing-related works as well

as different methods that have been developed for gesture recognition. Then the advantages and disadvantages of these methods and their application in real-world scenarios will also be briefly discussed.

2.2 Background

The purpose of our system is to introduce a gesture recognition system, using Wifi sensing techniques, that can be utilized in real-world scenarios, like operating smart home appliances. Before diving into the methodology of our proposed method, we will do background studies about Wifi-Sensing, Channel State Information(CSI), and related works of our proposed system.

2.2.1 Wifi Sensing

Wi-Fi sensing has experienced rapid growth because of the increasing demand for effective Human-Computer Interaction (HCI) solutions. As Sheng Tan et al. have shown in their survey [7], Wi-Fi technologies offer high throughput and easy deployment, making them ideal for various applications. Human behavior perception technologies, such as daily behavior recognition[8], fall detection[9], breath analysis[10], sleep monitoring[11], and user identification[12] etc. have been made utilizing Wi-Fi sensing technology.
Both Receiving Signal Strength Information (RSSI) and Channel State Information (CSI) can be used to implement a WiFi sensing method. But Yongsen Ma et al. in their survey [13] described that CSI particularly stands out due to its ability to provide detailed and nuanced information compared to RSSI. By including amplitude, phase, and frequency response data of received signals, CSI enables a deeper understanding of signal propagation in complex environments affected by multipath phenomena. This detailed insight into multipath propagation and environmental interactions enhances the accuracy and reliability of wireless sensing applications.

The fine-grained nature of CSI has positioned it as a preferred method for wireless sensing and analysis, especially in implementing advanced applications within smart environments[14] and health monitoring systems[15]. Its capability to

distinguish multipath components and its comprehensive signal characterization make CSI a key enabler for innovative HCI solutions and intelligent sensing technologies. Overall, Wi-Fi-based sensing technologies leveraging CSI offer promising opportunities to enhance user experiences and enable new functionalities in various domains.

2.2.2 Channel State Information

In Wi-Fi devices adhering to the IEEE 802.11n/ac standard, CSI provides detailed channel frequency response (CFR) information. The signal propagated between a Transmitter and Receiver is captured in the form of CFR, represented as CSI as shown in eq. (2.1):

$$Y(f,t) = H(f,t) \times X(f,t) + N \tag{2.1}$$

Here, $Y(f,t)$ and $X(f,t)$ are the signals transmitted and received,N represents noise in the system, and $H(f,t)$ is the complex CSI matrix of f subcarrier at time t. The complex value here can be represented as:

$$H(f,t) = ||H(f,t)|| \times exp^{i\angle H(f,t)} \tag{2.2}$$

Here in eq. (2.2) $||H(f,t)||$ and $\angle H(f,t)$ represents the magnitude and phase of received signal of subcarrier f respectively.

CSI plays a vital role in diverse applications such as behavior recognition [16], human activity monitoring[17], and localization[18]. It effectively captures variations in signal propagation caused by the surrounding physical environment. This capability makes CSI-based Device-Free Wi-Fi Sensing (DFWS) a valuable tool for pervasive sensing applications, enabling non-intrusive monitoring and analysis without requiring additional sensors or hardware. Overall, CSI enhances the understanding and utilization of wireless channels in various contexts, driving innovations in wireless sensing and communication technologies.

2.2.3 ESP-32 for CSI Data Extraction

Espressif Systems created this powerful microcontroller known as the Espressif ESP32[19]. The Internet of Things (IoT) is used extensively because of its feature-rich capabilities, affordability, and versatility. ESP32's built-in WiFi stack allows it to send and receive data. Therefore, we can set up a reliable WiFi connection between the transmitter and receiver if we take two ESP32 boards and flash them as Access Point (AP) and Receiver Point (STA), respectively.

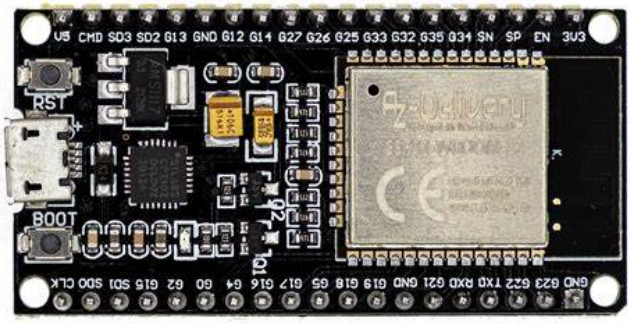


Figure 2.1: ESP32 board

In our study, we have used ESP32 microcontroller, shown in fig. 2.1 and ESP-32 CSI toolkit developed by Steven M. Hernandez [20] to obtain CSI data using these ESP-32 microcontrollers. With the aid of the ESP32 CSI Toolkit, Channel State Information (CSI) can be obtained straight from the ESP32 microcontroller. When two ESP32 boards are used, one as the access point (AP) and the other as the station (STA), CSI frames properly work between them. One initiates the CSI callback on the other by sending a UDP packet. Below we describe the transmitting and receiving edge of our connection to collect CSI data:

- **Active AP:** First an ESP-32 needs to be configured and flashed as an access point to establish a Wi-Fi network with a distinct SSID and password. After that, the ESP32 will wait for CSI data to be sent from a linked device.
- **Active STA:** An ESP32 can become a station by using this application. Whenever the application is flashed as an active station, it connects to a WiFi access point. This connection is established with some code and given the password and SSID of the access point.

2.2.4 Related Works

As we have mentioned earlier in this chapter, gesture recognition has been a very popular field of research for the past few years. Some of the common gesture recognition techniques are computer vision-based techniques, wearable sensor-based techniques, and device-free WiFi sensing-based techniques.

Munir Oudah et al. in [21], reviewed hand gesture recognition techniques in computer vision, focusing on classification methods, hand segmentation, classification algorithms, datasets, and camera types. They provided a comprehensive analysis of various hand gesture methods, discussing their advantages, limitations, and potential applications in different fields. In this paper, they mentioned that, besides time-consuming matching processes and high computational costs, computer-vision-based gesture recognition methods have limited ability to handle unclear views, especially in less illuminated places, and may pose potential privacy concerns.

Chun Zhu et al. in their [22], proposed a Smart Assisted Living (SAIL) system for the elderly and disabled, addressing hand gestures and daily activity recognition using neural networks and multi-sensor fusion. It demonstrates the effectiveness of implemented algorithms through experiments with a prototype wearable sensor system. This system works fine but has some scalability and real-world deployment challenges, as well as from user usability aspects, it lacks practical applicability, as users need to wear the device continuously.

Fadel Adib et al. in [23] demonstrates how Wi-Fi signals can be used to see moving objects even through walls and identify the number of people in a closed room, achieved by MIMO interference nulling and treating the human motion as an antenna array.

Li Sun et al. introduces WiDraw[24], a system using commodity WiFi cards for hands-free drawing in the air, achieving a median hand tracking error below 5 cm. WiDraw utilizes the Received Signal Strength Indicator (RSSI) to track hand movements in the air using commodity WiFi cards, enabling hands-free drawing without the need for wearables or custom hardware

WiGest[5] proposed by Abdelnasser et al. utilizes standard WiFi equipment to detect hand gestures around a mobile device without modifications or training. This system employs a Discrete Wavelet Transformation (DWT) method for noise reduction in RSSI values, enhancing WiGest's quality and robustness. The system achieves an 87.5% accuracy in detecting basic primitives with a single access point around the mobile device.

Xi et al. in[25] introduce ARM, a system using CSI for device-free human activity recognition, achieving over 75% accuracy. They address the limitations of existing methods by proposing efficient algorithms and noise elimination techniques for accurate profiling and recognition. And emphasize that compared to RSSI-based systems, CSI-based activity recognition systems provide more detailed information for accurate activity recognition

Hasmath et al. in their survey on "Device-free human gesture recognition using Wi-Fi CSI" [6] enlisted previous research on Gesture recognition, using WiFi Sensing and showed how wifi sensing has gained tremendous research attention lately.

Zhou et al. in[26] paper introduces a device-free number gesture recognition approach, DeNum, based on deep learning and fine-grained Channel State Information (CSI) for accurate gesture recognition. DeNum achieves an average accuracy of 94% in an office scenario, showcasing its effectiveness in recognizing number gestures using wireless signals. However, this performance deteriorates from 94% to 67% in non-trained situations.

SignFi[27] proposed by Yongsen Ma et al. utilizes WiFi Channel State Information (CSI) and a 9-layer Convolutional Neural Network (CNN) to recognize 276 sign gestures accurately, surpassing existing technologies in gesture recognition. The system achieves an average recognition accuracy of 98.01% in the lab, 98.91% at home, and 94.81% in combined environments. However, for classification, these gestures need to be given at a specific distance between the transmitter and receiver.

WiCatch[28], proposed by Zengshan et al. is a Wi-Fi gesture recognition system using CSI to recognize hand motions with high accuracy. WiCatch employs a

data fusion-based interference elimination algorithm, virtual antenna arrays, and SVM classification to achieve recognition accuracies over 0.96. However as stated in this paper, the antenna spacing needs to be in a certain range for successful classification.

Hong Li et al. introduce WiFinger [29], utilizing WiFi signals for finger-grained gestures to input text on WiFi devices with high accuracy. WiFinger achieves up to 90.4% accuracy in recognizing 9-digit finger-grained gestures from American Sign Language. They also mentioned that, for stable and clear finger gesture patterns in the CSI stream, transceivers should be placed relatively close together, as patterns weaken with increasing distance in this system.

MSF-Net[30] proposed by Junxin Chen et al. introduces an AIoT-based system designed to recognize various activities indoors using Wi-Fi signals to enhance human safety. In their proposed system, first, they used Intel 5300 NICs to gather Channel State Information (CSI) from fixed locations; then transmitted the collected CSI data to a cloud server via IoT. AI Analytics is the central part of their system, where they pre-process the data, and transform WiFi-CSI signals using DWT and STFT. Using these techniques MSF-Net achieves Cohen's Kappa or CK measure of more than 75% on different existing datasets.

Hasmath et al. in[31] explore Wi-Fi CSI for human sign language recognition, comparing LSTM's use with amplitude values alone or combined with phase information. In their work, LSTM outperforms CNN, achieving 75-78% recognition rates on SignFi[27] datasets with minimal pre-processing, showcasing its potential for gesture recognition tasks.

Another paper[32] proposed by Marwa R. M. et al. proposed a passive desk body gesture recognition system using Wi-Fi CSI from an ESP32 toolkit to detect workers' moods and emotions efficiently within the Internet of Things ecosystem. Achieving high recognition accuracy of over 98% in-session and 72% out-session evaluations, the system offers a cost-effective and energy-efficient solution for on-desk gesture recognition. Also, here the user needs to be on his desk for gesture recognition to take place.

We have also encountered a few works that tried to implement a position-independent

gesture recognition system. The study by Ohara et al. in [1] investigates the use of Wi-Fi Channel State Information (CSI) for position-independent gesture recognition. They designed a system where users must carry a smartphone equipped with an Intel 5300 Network Interface Card (NIC). This setup enables the transmission of CSI data from the smartphone to a computer. The research primarily explores the potential of using the velocity attributes derived from CSI data to identify different hand gestures. Their technique demonstrates the ability to accurately recognize five distinct gesture types with an accuracy exceeding 60%. This approach highlights the feasibility of using velocity features in CSI data for effective gesture recognition tasks.

Another paper by Yong Zhang et al.[33] introduces an adaptable CSI activity recognition system based on meta-learning, requiring minimal retraining effort for new environments or activities. By utilizing a pre-trained model and time encoding on CSI data, the system achieves 79.5% for 5 gesture classes without extensive retraining, enhancing practicality in human activity recognition.

2.2.5 Research Gaps

All these studies stated above highlight the increasing interest in research on Wi-Fi sensing-based gesture recognition systems. However, there are some research gaps on CSI-based position-independent gesture recognition systems, that we have identified throughout these studies. These research gaps are stated below:

- There is very limited work on CSI-based position-independent gesture recognition systems.
- Some WiFi sensing-based systems require carrying smartphones or are applicable in a very limited range.
- The accuracy rates for this kind of system are very sub-optimal.

Later on in this thesis, we tried to solve these problems throughout our work.

2.3 Conclusion

In this chapter, a detailed overview of WiFi sensing-based gesture recognition systems is discussed. Also, the limitations, results, and implementation methods of existing systems are discussed in this section. We can see that CSI-based gesture recognition systems have gained a great research interest, especially when it comes to developing a smart home or smart environment. We have also realized that there is still room for improvement in developing a feasible CSI-based position-independent gesture recognition system. WiFi sensing-based systems are highly sensitive and can differentiate between people and their activities accurately. Moreover, it is non-intrusive and relatively cost-effective. A comprehensive explanation of our proposed method for a CSI-based position-independent gesture recognition system will be found in the forthcoming chapter.

Chapter 3

Methodology

3.1 Introduction

This chapter includes our proposed approach for our position-independent gesture recognition system in a smart home setting using a CSI-based WiFi signal is included in this chapter. We will go into great depth about the complete process of setting up the system, gathering the CSI data, analyzing this data, and using machine learning and deep learning techniques.

3.2 Proposed Gesture Recognition System

In this thesis, our goal is to establish a CSI-based position-independent gesture recognition system. We have proposed a fully operational methodology that can be used to reach our goal. The entire functionality can be divided into four parts:

1. Hardware Configuration
2. CSI Data Collection
3. Data Preprocessing
4. Learning Models

3.2.1 Hardware Configuration

The initial and pivotal phase of our system is configuring the hardware. We used a pair of ESP-32 microcontrollers [1], utilizing one as a receiver and the other as a transformer. These compact devices facilitate the transmission and reception

[1] **Source:**https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/windows-setup.html

of Channel State Information (CSI) data via a WiFi network. The hardware configuration, shown in fig. 3.1 entails several steps:

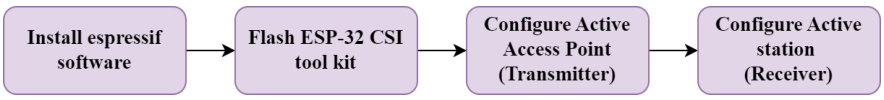


Figure 3.1: Hardware Configuration

Step1: Install Espressif IoT Development Framework (ESP-IDF). This step involves following a comprehensive installation guide to set up the ESP-IDF, specifically version 4.3, which our project necessitates.

Step2: Configuring the ESP-IDF to enable flashing of the ESP32s. This entails configuring the ESP32s as distinct modules to facilitate data collection and transmission.

Step3: One of the ESP32s is programmed with Active-AP to function as an access point or transmitter. This configuration enables the ESP32 to transmit CSI data while serving as an access point. It will be connected to a power source.

Step4: The other ESP32 is flashed with Active-STA to capture the transmitted CSI data as a receiver. This step establishes a WiFi connection between the two ESP32s, allowing for the retrieval of CSI data from the receiver. This ESP-32 will be connected to our laptop or computer, as shown in fig. 3.2.

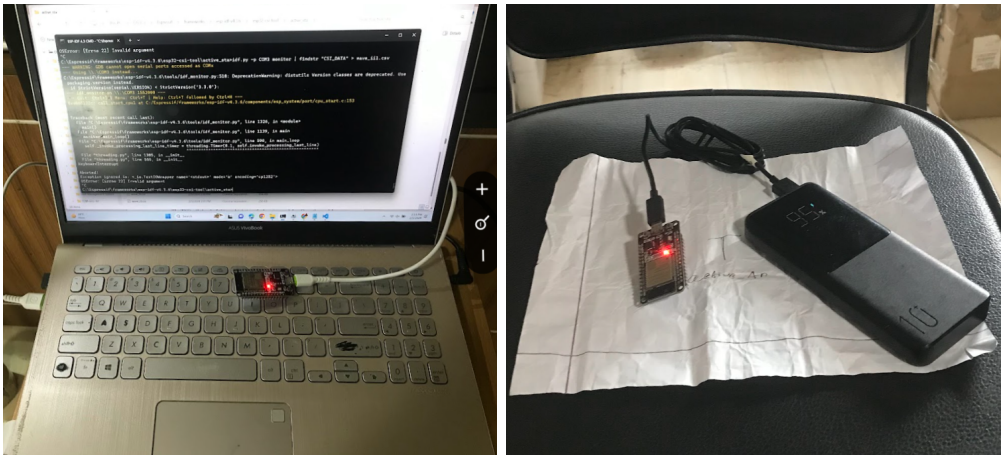


Figure 3.2: Hardware Setup

3.2.2 CSI Data Collection

After hardware configuration, our next task is to set up our environment and collect the CSI data, using our configured ESP-32 devices. The experimental setup is shown in fig. 3.3.

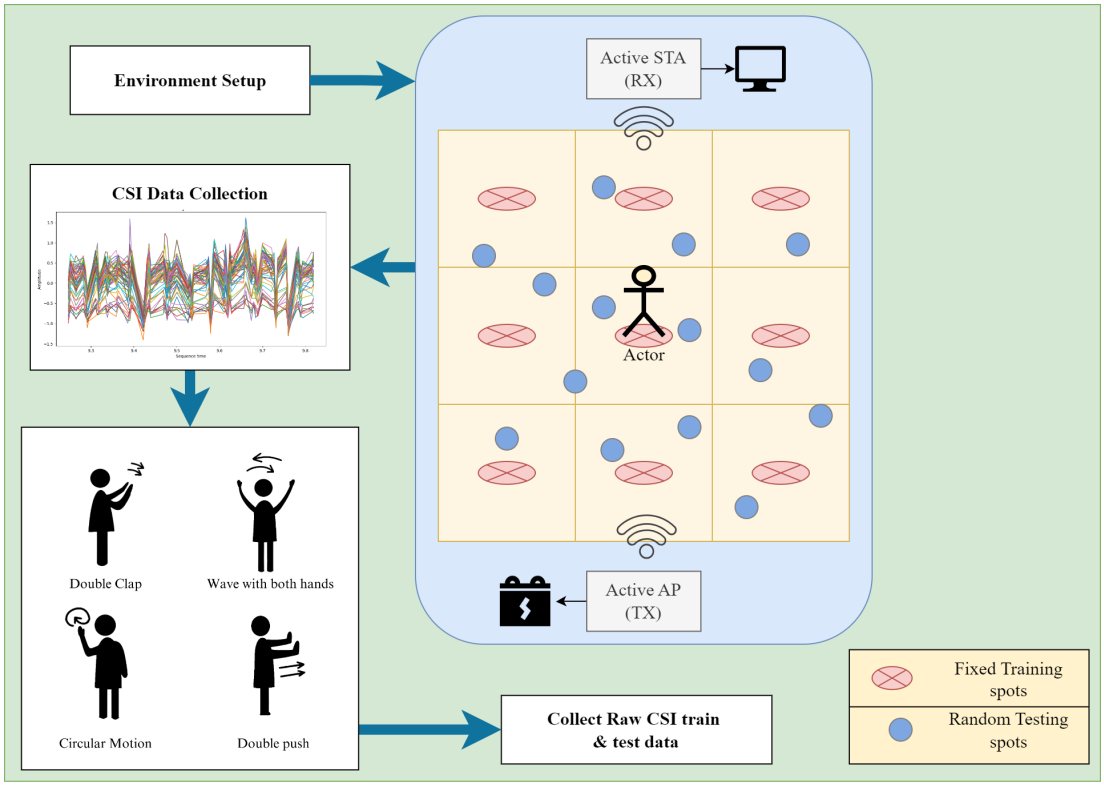
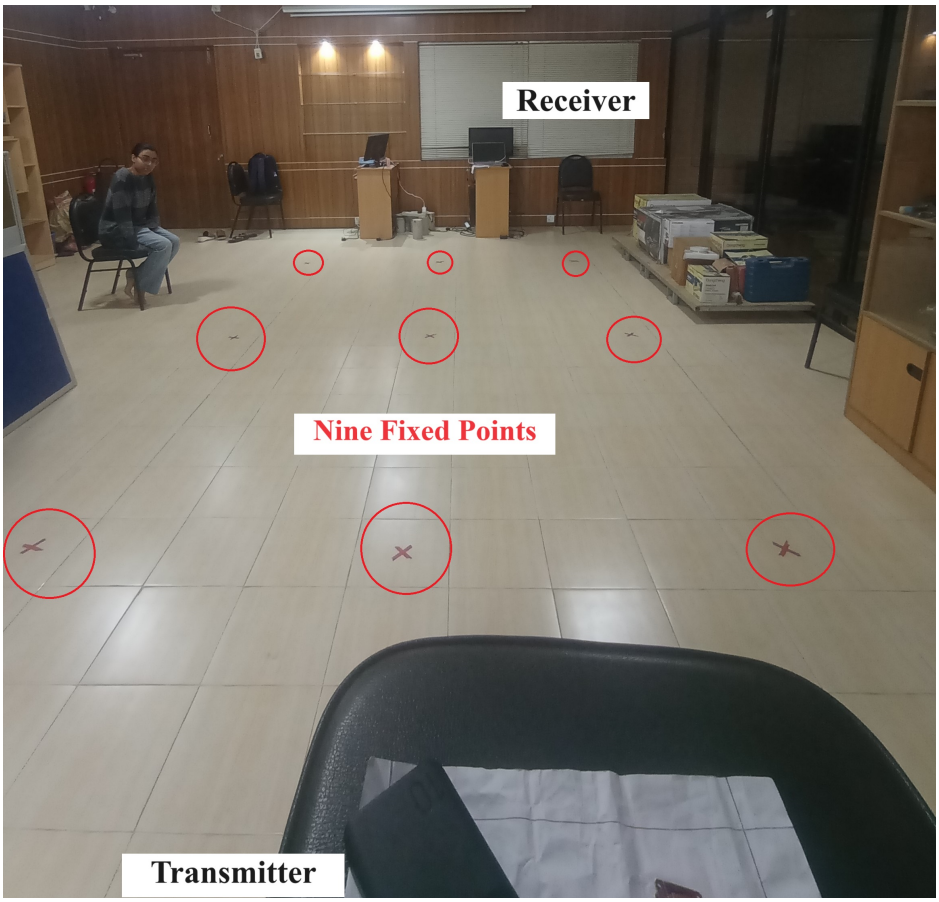


Figure 3.3: CSI Data Collection Process

This data collection process is mentioned below:

Step1: We initially segmented the room into a 3x3 grid and pointed the center of each grid cell. Subsequently, at each of these nine locations, we performed each of the four gestures ten times, resulting in a total of 360 sets of training data. Each gesture was executed independently for 20 seconds.

Step2: Following the training data collection, we conducted all four gestures at five randomly chosen positions within the room, again for 20 seconds each. This process yielded 20 sets of testing data. Below, in fig. 3.4 we can see the actual room arrangement for data collection.



(a) Marking Nine Fixed Points in the Room

(b) Actor giving Gestures from Marked positions

Figure 3.4: Arrangements for Data Collection

Step3: Once all the data were collected, they were ready for pre-processing.

3.2.3 Data Preprocessing

After collecting CSI data the first thing to do is data preprocessing. For preprocessing, we have followed some important steps which is shown in fig. 3.5 and discussed below:

1. After generating the CSV file, we individually parsed each set of data.
2. To start, we used the throughput value to filter the input. In this case, a high throughput signal is indicated by $sig\_mode = 1$.

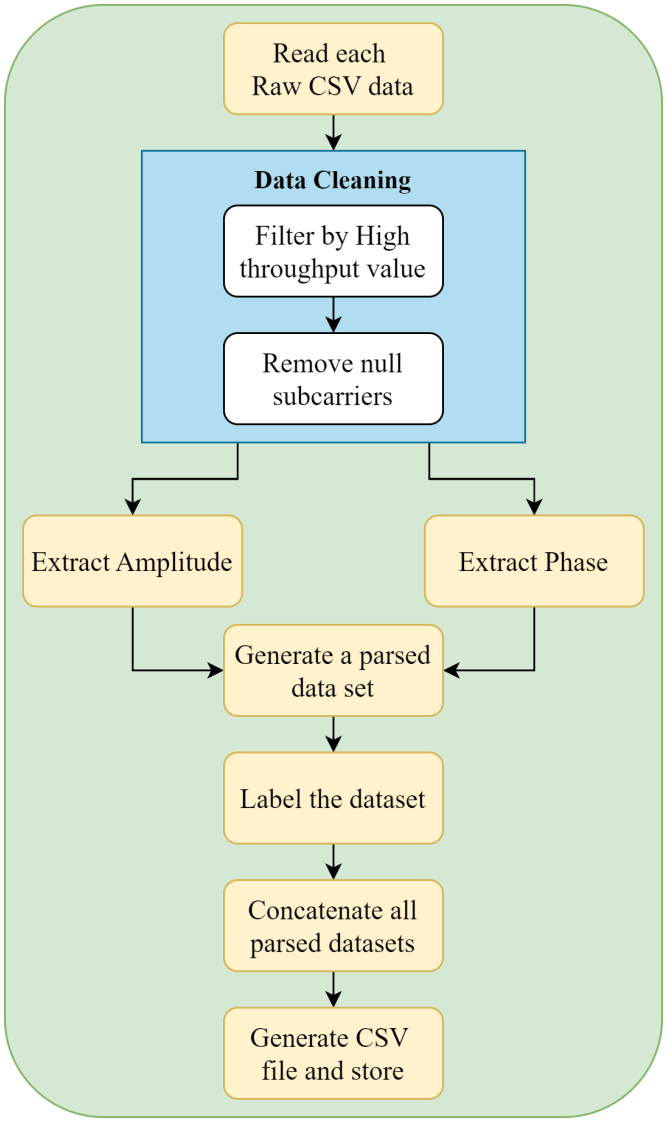


Figure 3.5: CSI Data Pre-processing

3. Next, after examining the sub-carriers, we discovered that certain CSI sub-carriers were giving null values for each occurrence of every signal. So, we removed those null sub-carriers.

4. Following the filtering process, 104 CSI sub-carriers remained. In CSI data, every subsequent sub-carrier consists of a pair of real and complex values. We were able to extract 52 amplitude sub-carriers and 52 phase sub-carriers for every row by using the appropriate formulae.

5. We created a parsed data set using the retrieved amplitude and phase values. There are 52 amplitude and 52 phase sub-carriers in a processed data stream.

6. We labeled each parsed data set, by its activity name.

7. Next, a complete training dataset and a complete testing dataset were created by concatenating all of these parsed and labeled CSI data sets.

8. Finally, after the testing and training datasets are prepared, we store them as CSV files to do farther research.

3.2.4 Approched Models

After labeling and creating the final train and test datasets, we applied ensemble learning, shapelet learning these two machine learning methods, and CNN and LSTM these two Deep learning methods to our data set for gesture classification, as shown in fig. 3.6.

1. Initially, we imported both the training and testing data in CSV format for each machine learning or deep learning method.

2. In our ensemble learning approach, we opted for three distinct models: LGBM, XGBoost (XGB), and Random Forest. To mitigate individual biases and enhance overall model performance, we employed the Voting ensemble method. This entailed combining the predictions of the Random Forest, XGBoost, and LightGBM classification models.

3. Another machine learning strategy we explored involved the application of

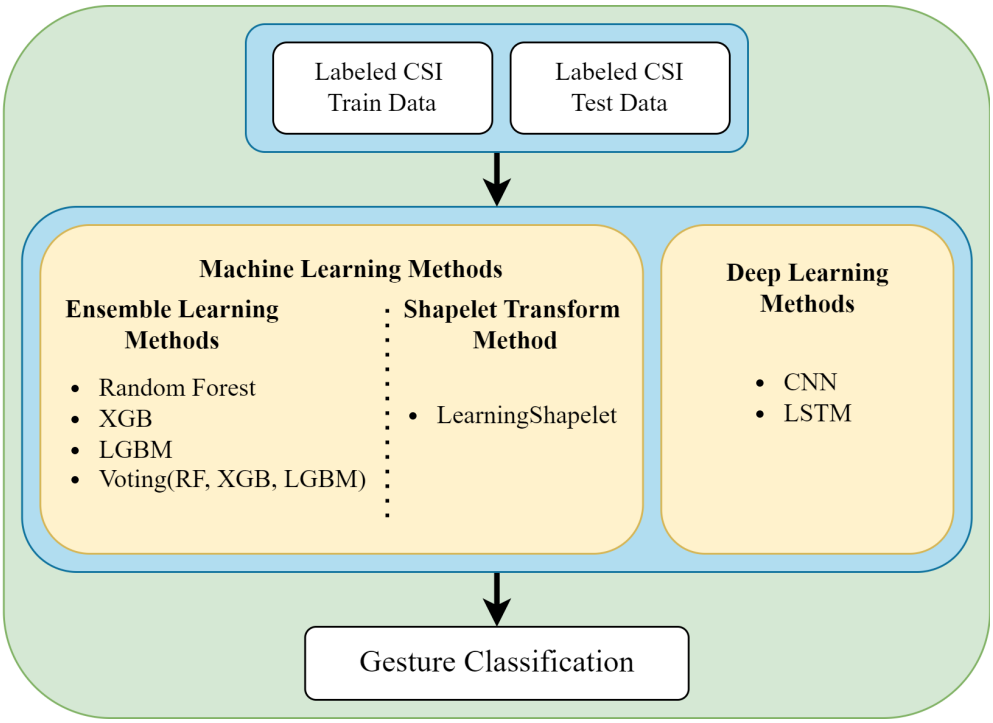


Figure 3.6: Approached Models

the shapelet transform method. This technique, tailored for time series classification tasks, entails identifying discriminative subsequences, termed shapelets, within the temporal data. Given that our CSI data follows a time series format, we applied the LearningShapelet method from the tslearn library to our dataset.

4. Among our deep learning methodologies, the CNN model was constructed with multiple layers, including Convolutional (Conv1D) layers, normalization layers, max-pooling layers, and dense (fully connected) layers, along with a dropout layer to prevent overfitting. While the LSTM model was composed of an LSTM layer followed by a dense layer utilizing softmax activation.

5. Subsequently, each of these models underwent testing on the testing dataset to evaluate their performance.

3.3 Implementation Procedure

Before starting the data collection process for our position-independent gesture recognition system in a constrained indoor environment, we have to fulfill some software and hardware requirements.

3.3.1 System Requirements

In this section, the hardware and software requirements needed for our proposed system are described.

1. **Hardware Requirements**
   (a) **Data Collection :**
      - A pair of ESP32 micro-controller
      - A Power Bank
      - A USB type-B cable
   (b) **Data Processing :**
      - A fully functional laptop
      - Minimum RAM 16GB
      - SSD 256 GB
2. **Software Requirements**
   - Operating System : Windows
   - Espressif IDF
   - ESP32 CSI Toolkit
   - VSCode
   - Jupyter Notebook
   - Python 3.10
   - Tensorflow

- Keras
- Numpy
- Pandas
- Tslearn
- Google Colab

3.3.2 ESP32 Setup

As mentioned earlier, we had to flash each ESP-32 micro-controller, one as an Access Point or transmitter and another as an Active station or receiver. To do this, first, we installed Espressif IoT Development Framework (ESP-IDF) [2] fig. 3.7 to build firmware for supported chips. Then, to allow for the collection of Channel State Information (CSI) from the ESP32 Wi-Fi-enabled micro-controller, we have used ESP-32-CSI-tool [3].

Installation Process:

1. First, we downloaded Espressif IoT Development Framework (ESP-IDF) for Windows. Our project requires version (v4.3) of ESP-IDF.
2. Installed VSCode with ESP-IDF extension.
3. Installed ESP-IDF in Windows, and cloned esp32-csi-tool, using the "git clone https://github.com/StevenMHernandez/esp32-csi-tool" command.
4. Once the cloning was done, we were set for the hardware setup.

Flash ESP32:

1. First we configured *active_ap* (Typically used as the CSI-RX) as shown in fig. 3.8 and the following configuration steps.
2. Configuration steps:
   - Serial flasher config > 'idf.py monitor' baud rate > Custom Baud Rate
   - Serial flasher config > Custom baud rate value > 1552000

[2] **Source:**https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/windows-setup.html
[3] **Source:**https://github.com/StevenMHernandez/ESP32-CSI-Tool

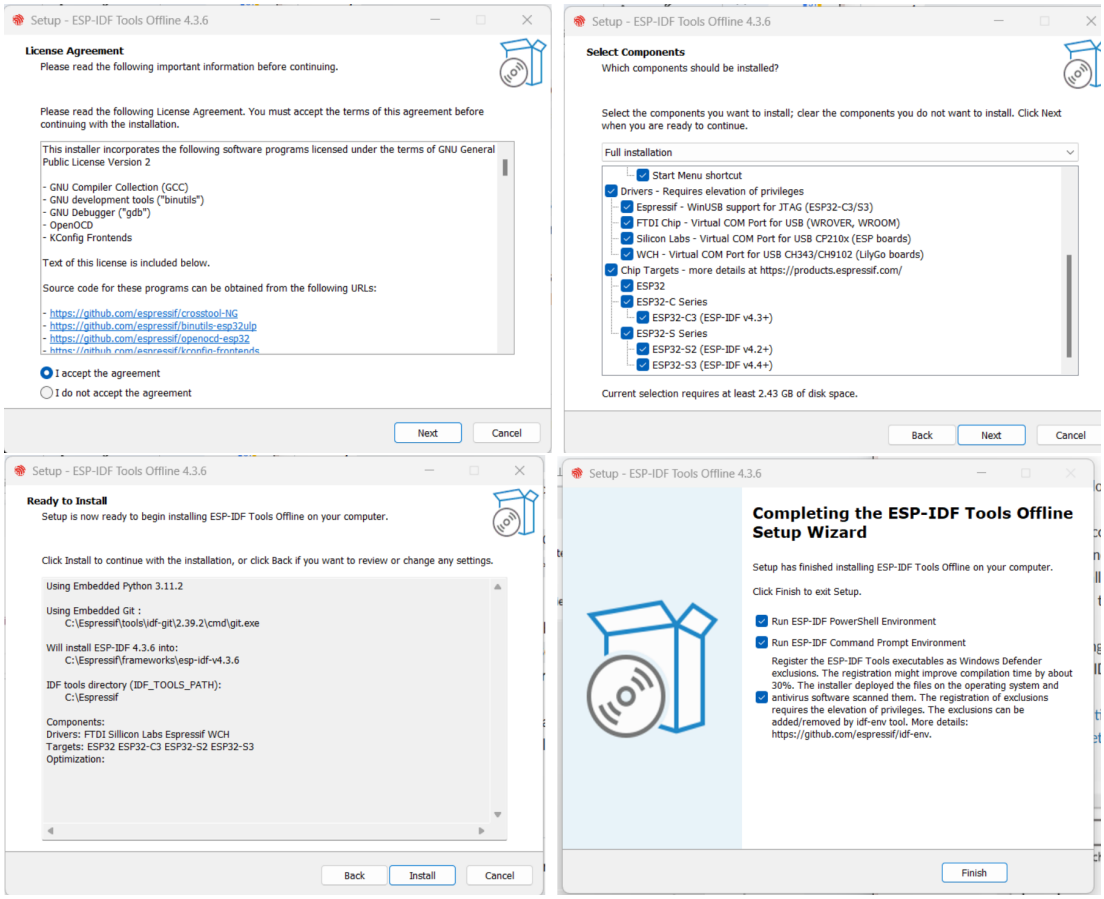


Figure 3.7: ESP-IDF Installation Process

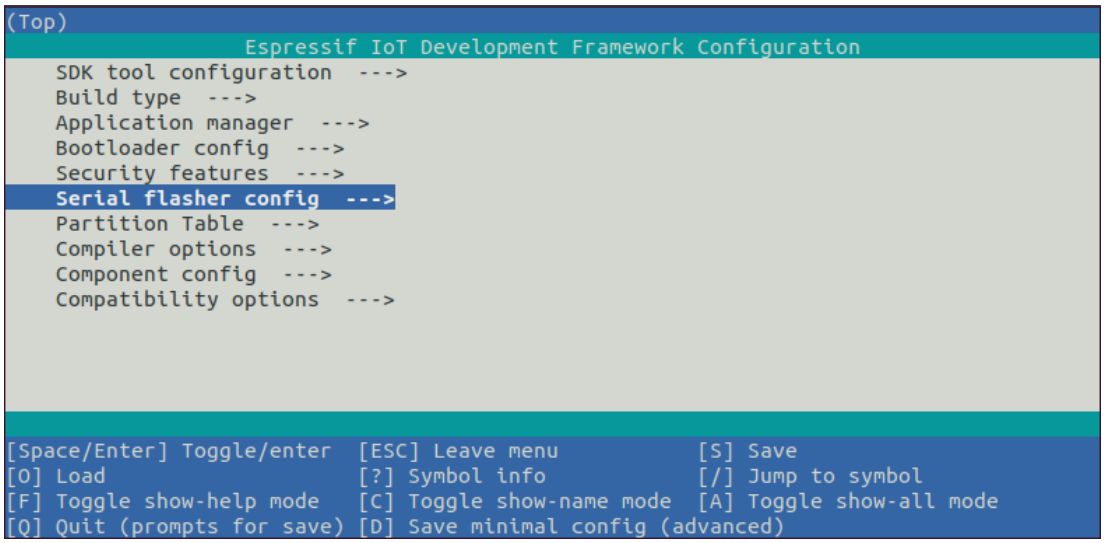


Figure 3.8: Project Configuration Window

- Component config > Common ESP32-related > Channel for console output > Custom UART

- Component config > Common ESP32-related > UART console baud rate > 1552000

- Component config > Wi-Fi > WiFi CSI(Channel State Information) (Press space to select)

- Component config > FreeRTOS > Tick rate (Hz) > 1000

- ESP32 CSI Tool Config > **** all

3. Connected one ESP32 micro-controller with a port COM6 of our laptop, using a USB cable and flashed it.

4. After flashing active_ap, we configured, connected and flashed active_sta with the similar process of active_ap. Our active station was connected to port COM8. Connected active access point and active station can be seen in the device manager like fig. 3.9.

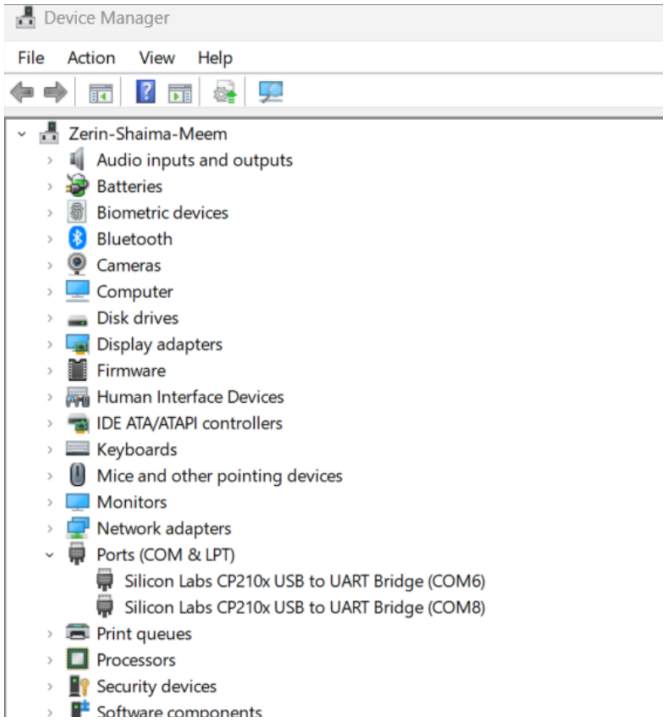


Figure 3.9: Viewing COM6 as active_ap and COM8 as active_sta in Device manager

5. Once we finished flashing both active station(RX) and active access point (TX), we detached the TX from the laptop and connected it to a power bank.

3.3.3 CSI Data Collection Process

After configuration, we chose a laboratory room for our experiment. We divided the room into a 3x3 grid and collected train and test data for our system. We collected a total 360 sets of train data and 20 sets of test data. Each set of data collection process is described bellow:

- Monitor active AP with command “idf.py -p COM6 monitor”.
- As soon as the command started executing, the monitor started showing data received from the transmitter as shown in fig. 3.10

Figure 3.10: CSI data collection

- Then by command “idf.py -p COM6 monitor | findstr “CSI_DATA” > test1.csv” we stored the CSI data in CSV format. And visualized the CSV file in VSCode as in fig. 3.11.

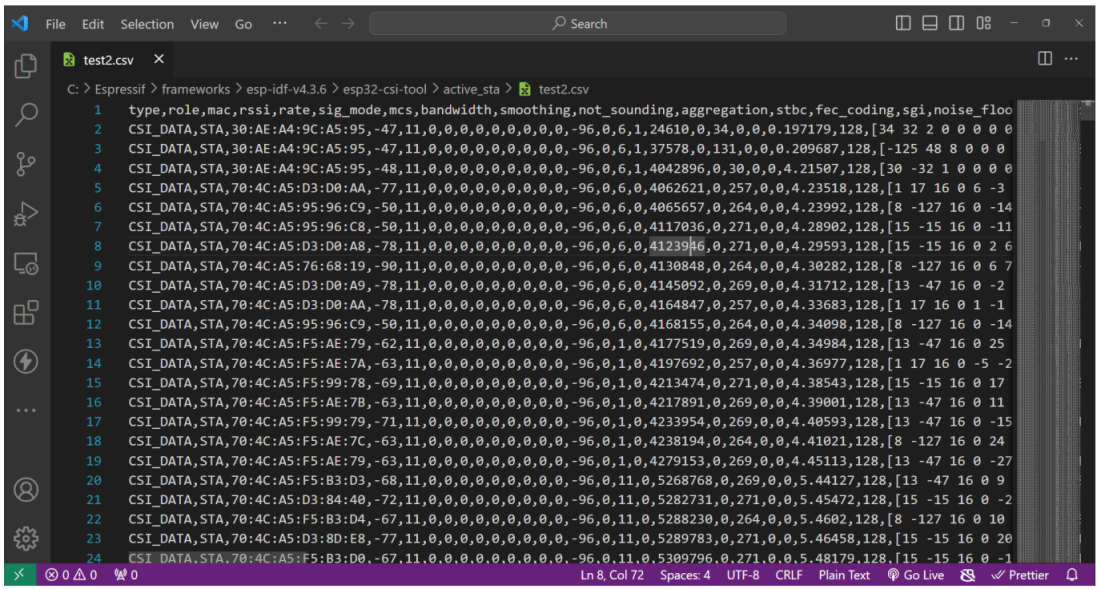


Figure 3.11: Stored CSI data in CSV format

3.3.4 Data Preprocessing

1. **Storing CSI data in CSV format:** At this stage, we already collected 10 sets of data per gesture in each of the 9 positions of our 3x3 grid. Also, we collected 5 sets of test data for each of the 4 gestures from random positions inside the grid. Each of these $360 + 20 = 380$ data sets was collected individually and saved as a CSV file with a name in the format **gesture__position__i.csv**.

   Where,

   gesture $= \{clap, wave, push, circle\}$, **position** $= \{i, ii, iii, iv, v, vi, vii, viii, ix\}$ and i ranges from 1 to 10.

   For test data, file naming format was **test__gesture__i.csv**. Where i ranges from 1 to 5.

   The format of each set of raw data is shown in fig. 3.22.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	type	role	mac	rssi	rate	sig_mode	mcs	bandwidth	smoothing	not_sounding	aggregation	stbc	fec_coding
2	CSI_DATA	STA	30:AE:A4:9C:A5	-67	11	0	0	0	0	0	0	0	0
3	CSI_DATA	STA	30:AE:A4:9C:A5	-67	11	0	0	0	0	0	0	0	0
4	CSI_DATA	STA	30:AE:A4:9C:A5	-74	11	1	3	1	1	1	0	0	0

(a) First few columns CSI data

S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE
local_timestamp	ant	sig_len	rx_state	real_time_set	real_timestamp	len	CSI_DATA					
23753	0	34	0	0	0.211462	128	[34 32 2 0 0 0 0 0 0 0 0 0 22 -21 23 -19 26 -18 24 -18 24 -17 25 -15 26 -14 23 -14 22 -12 22 -12 24 -12 21 -13					
37148	0	131	0	0	0.224475	128	[-125 48 8 0 0 0 0 0 0 0 0 0 6 28 5 27 2 30 2 30 0 30 -2 29 -2 29 -4 26 -3 26 -4 26 -4 26 -3 24 -2 24 -2 23 -2 23					
41305	0	135	0	0	0.228841	384	[-121 112 8 0 0 0 0 0 0 0 0 0 7 -25 7 -24 10 -22 12 -25 14 -21 12 -23 11 -23 11 -20 16 -17 15 -21 17 -17 15 -18					

(b) Last few columns of CSI data

Figure 3.12: Raw CSI data

2. **Data Cleaning:** For cleaning the data and extracting phase and amplitude values from the complex sub-carriers, we implemented CSI extracting code fig. 3.13.

```
s=f"/content/drive/MyDrive/real data v1/{geture}/{gesture}_{pos}{k}.csv"
clean_csi = (Data_Preprocess(s)
             .filter_by_sig_mode(1)
             .get_csi()
             .remove_null_subcarriers()
             .get_amplitude_from_csi()
             .get_phase_from_csi()
             )
```

Figure 3.13: Parsing CSI data

Step 1 : Filter by High Throughput Value

We implemented the filter_by_sig_mode() function to filter data by high throughput signals, as shown in fig. 3.14. Here, sig_mode=1 indicates a high throughput signal.

```
def filter_by_sig_mode(self, sig_mode):
    self.csi_df = self.csi_df.loc[self.csi_df['sig_mode'] == sig_mode]
    return self
```

Figure 3.14: Filtering CSI data by $sig\_mode = 1$

Step 2 : Get CSI Sub-carriers:

After filtering the dataset, at this point, our data set had a total of 26 columns, among which the last one was named CSI Data. Each row of this column represents a data packet, holding 126 sub-carriers and, each pair of this 126 data represents a complex number. We only need that last column, with these CSI data. To get this, the code is shown in fig. 3.15:

```
def get_csi(self):
    raw_csi_data = self.csi_df['CSI_DATA'].copy()
    preprocessed_data = raw_csi_data.str.strip('[]').str.split()
    self.csi_data = np.array([np.fromstring(' '.join(data), dtype=int, sep=' ')
    for data in preprocessed_data if isinstance(data, list) and len(data)==128])
    return self
```

Figure 3.15: Removing Null Sub-carriers

Step 3 : Remove Null Sub-carriers:

After analyzing the data, we found that among 128 sub-carriers, 24 sub-carriers contained a null value always. We removed those null sub-carriers from our data sets and left with 104 complex sub-carriers. Hence, we reduced the sub-carrier to 52 from 64, as shown in fig. 3.16.

```
def remove_null_subcarriers(self):
    remove_null_subcarriers = self.NULL_SUBCARRIERS
    csi_data_T = self.csi_data.T
    csi_data_T_clean = np.delete(csi_data_T, remove_null_subcarriers, 0)
    csi_data_clean = csi_data_T_clean.T
    self.csi_data = csi_data_clean

    return self
```

Figure 3.16: Removing Null Sub-carriers

3. **Amplitude and Phase Extraction:** As we mentioned earlier, each consecutive pair of CSI data represents the imaginary and real component of a complex number. We know the formula to extract the amplitude and phase of any complex number $z = x + iy$ is:
   Amplitude: $|z| = \sqrt{x^2 + y^2}$
   Phase: $\theta = \tan^{-1}\left(\frac{y}{x}\right)$

   We implemented these formulae as in fig. 3.17, to extract 52 amplitude sub-carrier and 52 phase sub-carrier values from 104 complex sub-carriers.

```
def get_amplitude_from_csi(self):
    amplitude = np.array([np.sqrt(data[::2]**2 + data[1::2]**2) for data in self.csi_data])
    self.amplitude = amplitude
    return self

def get_phase_from_csi(self):
    phase = np.array([np.arctan2(data[::2], data[1::2]) for data in self.csi_data])
    self.phase = phase
    return self
```

Figure 3.17: Amplitude & Phase Extraction

4. **Data Merging and Labeling:** After extracting the amplitude and phase, we concatenated them as a data set with 52 amplitude and 52 phase sub-carriers. Then added another column for the label of that particular data, merged all train data sets, and created the final train data set. Similarly, we parsed, labeled, merged, and generated the final test data set, with all raw test data sets.

3.3.5 Implementing Machine Learning Approaches

In our research, we utilized a variety of machine learning and deep learning techniques. For the machine learning part, we employed ensemble learning methods, including XGBoost (XGB), LightGBM (LGBM), and Random Forest. Additionally, given the time series nature of our Channel State Information (CSI) data, we used a shapelet transform model specifically for classifying gestures.

3.3.5.1 Ensemble Learning Approaches

1. At first we implemented three ensemble learning classifiers as the base Ensemble Learning Method: XGBClassifier, LGBMClassifier, and RandomForestClassifier; according to the following specifications fig. 3.18:

```
rf = RandomForestClassifier(n_estimators=100, random_state=42,verbose=1)
xgb = XGBClassifier(
    max_depth=5,
    n_estimators=100,
    learning_rate=0.3,
    random_state=42,
    objective='multi:softmax',  # For multi-class classification
    num_class=4                 # Number of classes
)
lgbm = LGBMClassifier(
    num_leaves=31,
    max_depth=-1,  # No limit
    learning_rate=0.1,
    n_estimators=100,
    objective='multiclass',
    num_class=4,
    random_state=42,
    class_weight='balanced'  # Useful for imbalanced datasets
)
```

Figure 3.18: Applying different Ensemble Learning Methods

Here's a brief overview of these three popular machine learning algorithms:

- **LightGBM(LGBM):** LightGBM is a gradient boosting framework that uses tree-based learning algorithms. It is designed for distributed and efficient training, particularly for large datasets. Some of its key features are: (1)Gradient-based One-Side Sampling (GOSS), (2) Exclusive Feature Bundling (EFB), and (3) Leaf-wise (Best-first) Tree Growth like fig. 3.19.

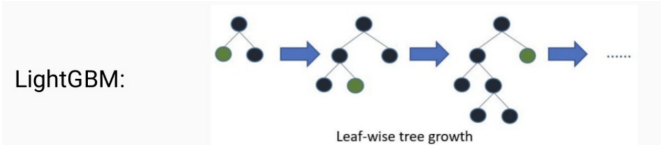


Figure 3.19: LGBM architecture[4]

GOSS is a technique to filter out data instances to speed up training without significant accuracy loss. It keeps all the instances with large gradients and performs random sampling on the instances with small gradients. EFB bundles mutually exclusive features to reduce the number of features and enhance efficiency.

- **XGBoost(XGB):** XGBoost stands for eXtreme Gradient Boosting. It is also a gradient-boosting framework but is optimized for performance and speed. XGBoost is popular for its efficiency in handling various types of data and robustness to overfitting. Key features of XGB model are: (1) Regularization, (2) Block Structure to Support the Parallelization of Tree Construction, and (3) Level-wise expansion like fig. 3.20.

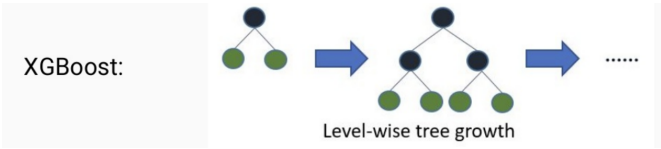


Figure 3.20: XGB architecture

XGBoost includes L1 and L2 regularization, which prevents over-fitting and improves model generalization. It also divides data into blocks that are then used to construct trees in parallel to speed up the learning process.

- **Random Forest (RF):** Random Forest fig. 3.21 is an ensemble learning method. It is built using multiple decision trees and voting on the most popular output class or averaging predictions.

[4]Source: `https://rohitgr7.github.io/lightgbm-another-gradient-boosting/`

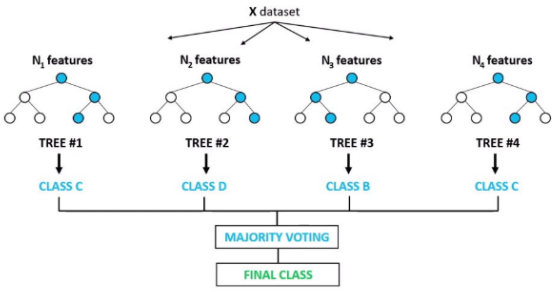


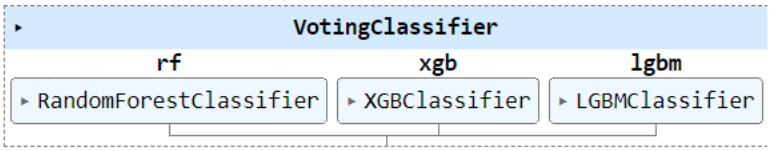
Figure 3.21: Random Forest architecture[5]

Random forests can provide insights into which features are most important in making predictions. Its robustness to noise allows it to perform well even in noisy data, reduces the risk of over-fitting, and improves predictive accuracy.

2. Then we used the Voting Classifier technique, as shown in fig. 3.22b to combine XGBoost, LightGBM, and Random Forest Classifiers. Here we specified the voting classifier as hard voting. As a result, the final output prediction is the class that receives the majority of votes from the models for each sample.

```
# Soft Voting
voting_clf_soft = VotingClassifier(
    estimators=[('rf', rf), ('xgb', xgb), ('lgbm', lgbm)],
    voting='soft'
)
```

(a) Soft Voting Classifier



(b) Applying Voting Classifier Ensemble Learning Method

Figure 3.22: Applying Voting Classifier to Ensemble RF, XGB, and LGBM

A Voting Classifier is an ensemble technique that combines conceptually different machine learning classifiers and uses a majority vote (hard voting) or the average predicted probabilities (soft voting) to predict the class labels.

[5]**Source:**https://www.freecodecamp.org/news/how-to-use-the-tree-based-algorithm-for-machine-learning/

3.3.5.2 Shapelet Learning Approaches

A shapelet is a short segment of a time series, that best predicts the target variable like fig. 3.23.

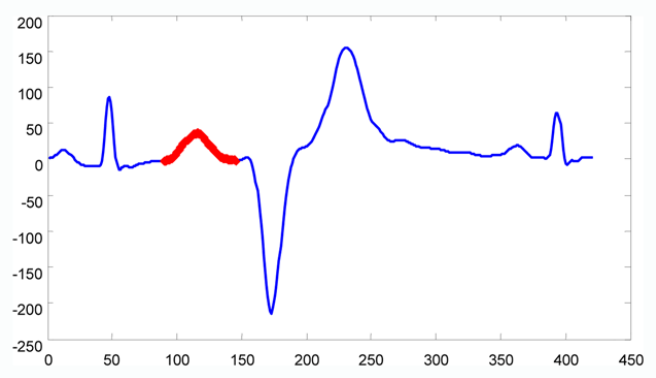


Figure 3.23: Time Series Shapelet[6]

To use the time-series property of our CSI data, we implemented the Learning-Shapelet model, proposed by J. Grabocka et al. in [34]. This method is tailored using a stochastic gradient learning algorithm. To implement this model, first, we transferred our dataset from 2D to 3D, where each instance of time series data was a window of length 200, each containing 52 amplitude sub-carriers. Then, instead of trying out all possible shapelets, we learned, near-to-optimal top-K shapelets by capturing their interaction fig. 3.24 using grabocka_params.

```
# Get statistics of the dataset
n_ts, ts_sz = X_train.shape[:2]
n_classes = len(set(y_train))

# Set the number of shapelets per size as done in the original paper
shapelet_sizes = grabocka_params_to_shapelet_size_dict(n_ts=n_ts,
                                                       ts_sz=ts_sz,
                                                       n_classes=n_classes,
                                                       l=0.1,
                                                       r=1)
print(shapelet_sizes)
```

```
{20: 5}
```

Figure 3.24: Calculating number of Shapelets and their Size

Here,

n_ts: Number of time series in the dataset.

ts_sz: Length of each time series (the number of time points).

[6]Source: https://datascience.codata.org/articles/10.5334/dsj-2018-006

n_classes: Number of distinct classes in the classification task.

l: Fraction of the length of time series to be used for base shapelet length

r: Number of different shapelet lengths to use

This function returns a dictionary of shapelets of a number of shapelets mapped to their shapelet size fig. 3.25.

```
def grabocka_params_to_shapelet_size_dict(n_ts, ts_sz, n_classes, l, r):

    base_size = int(l * ts_sz)
    base_size = max(base_size, 1)
    r = min(r, ts_sz)
    d = {}
    for sz_idx in range(r):
        shp_sz = base_size * (sz_idx + 1)
        n_shapelets = int(np.log10(n_ts *
                                   (ts_sz - shp_sz + 1) *
                                   (n_classes - 1)))
        n_shapelets = max(1, n_shapelets)
        d[shp_sz] = n_shapelets
    return d
```

Figure 3.25: Internal implementation of grabocka_params_to_shapelet_size_dict function

Finally, we defined and learned the LearningShapelet classification model using appropriate parameters as in fig. 3.26.

```
# Define the model using parameters provided by the authors
shp_clf = LearningShapelets(n_shapelets_per_size=shapelet_sizes,
                            optimizer=tf.optimizers.Adam(0.001),
                            batch_size=16,
                            weight_regularizer=0.01,
                            max_iter=1000,
                            random_state=42,
                            verbose=1,
                            scale=True)
```

Figure 3.26: Defining LearningShapelet Model

3.3.6 Implementing Deep Learning Approaches

On the deep learning side, we developed and assessed models based on Convolutional Neural Networks (CNNs) and Long Short-Term Memory networks (LSTMs) for our system.

No	Layer Type	Kernel Size	Strides	Number of Filters/Units	Activation Function
1	Conv1D	3	1	64	relu
2	Batch Normalization	-	-	-	-
3	MaxPooling1D	2	2	-	-
4	Conv1D	3	1	128	relu
5	Batch Normalization	-	-	-	-
6	MaxPooling1D	2	2	-	-
7	Flatten	-	-	-	-
8	Dense	-	-	100	relu
9	Dropout	-	-	-	0.5
10	Dense	-	-	4	softmax

Table 3.1: Summary of the CNN Model Architecture for Amplitude

3.3.6.1 Convolutional Neural Network (CNN)

This section describes the convolutional neural network (CNN) developed to classify gesture signals captured from Channel State Information (CSI). CNN is popular due to its proven efficacy in extracting hierarchical features from sequential data, like the temporal and spatial dependencies of CSI data.

The steps for utilizing CNN for our gesture classification are:

- **Model Architecture:** The CNN model, designed for amplitude classification, consists of several layers each contributing to the model's ability to discern distinct patterns in the data as shown in fig. 3.27:
- **Model Summary:** The implementation of our CNN model is described below and also a summary of layers shown in table 3.1,
- **Separate Model Implementations:** To comprehensively analyze the CSI data, we developed three distinct CNN models to optimize gesture recognition using Channel State Information (CSI). The first model processed 52 amplitude features, the second analyzed 52 phase features, and the third combined both datasets, using 104 features in total. Each model was designed to capture unique aspects of the CSI data, thereby enhancing the robustness and accuracy of the gesture classification.

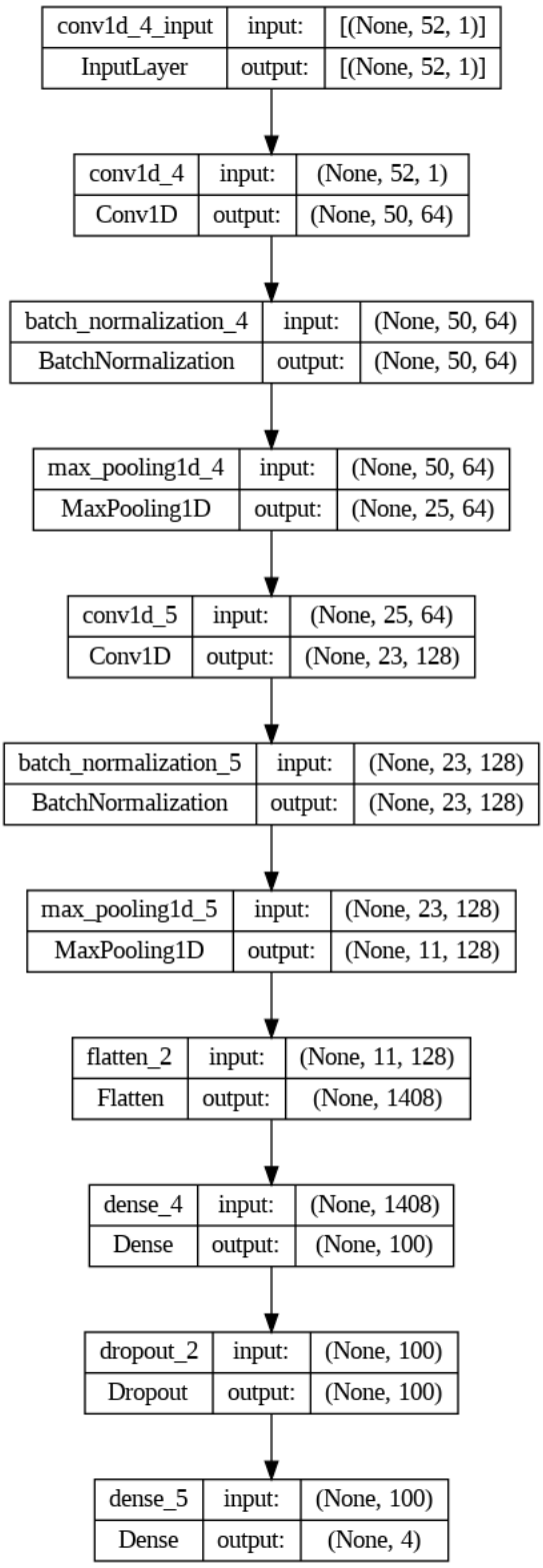


Figure 3.27: Architecture of CNN Model

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_4 (Conv1D)           (None, 50, 64)            256

 batch_normalization_4 (Bat  (None, 50, 64)            256
 chNormalization)

 max_pooling1d_4 (MaxPoolin  (None, 25, 64)            0
 g1D)

 conv1d_5 (Conv1D)           (None, 23, 128)           24704

 batch_normalization_5 (Bat  (None, 23, 128)           512
 chNormalization)

 max_pooling1d_5 (MaxPoolin  (None, 11, 128)           0
 g1D)

 flatten_2 (Flatten)         (None, 1408)              0

 dense_4 (Dense)             (None, 100)               140900

 dropout_2 (Dropout)         (None, 100)               0

 dense_5 (Dense)             (None, 4)                 404

=================================================================
Total params: 167032 (652.47 KB)
Trainable params: 166648 (650.97 KB)
Non-trainable params: 384 (1.50 KB)
_________________________________________________________________
```

Figure 3.28: Model Summary

- **Final Ensemble Result:** To obtain the final classification result, we averaged the outputs from these three models as shown in fig. 3.29.

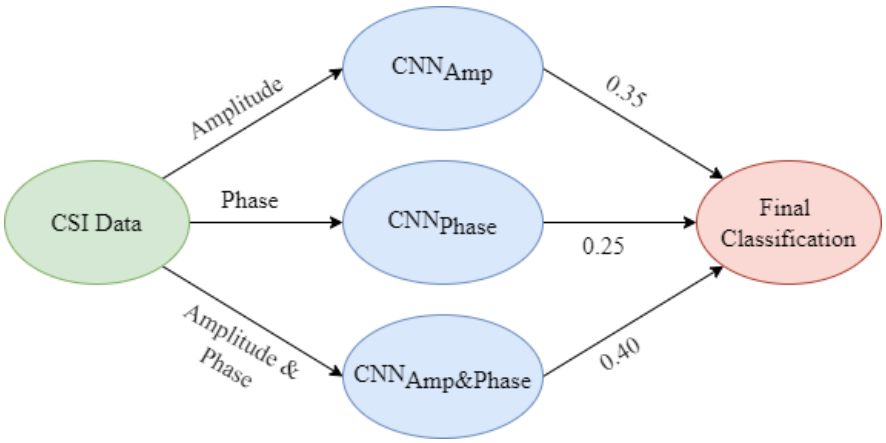


Figure 3.29: Weighted Average of CNN Models

This ensemble approach leveraged the strengths of each model, ensuring more reliable and accurate gesture recognition.

Lets discuss the layers of each of our CNN models:

Convolutional Layer (Conv1D): The foundational component of a convolutional neural network is the CONV layer. As our data is sequential, we used Conv1D, or the one-dimensional convolution technique to extract feature from our data. It involves applying windows of filters or kernels to the input sequence. In our system, we used 64 filters in the first layer and 128 filters in the second layer of kernel size 3 like fig. 3.30.

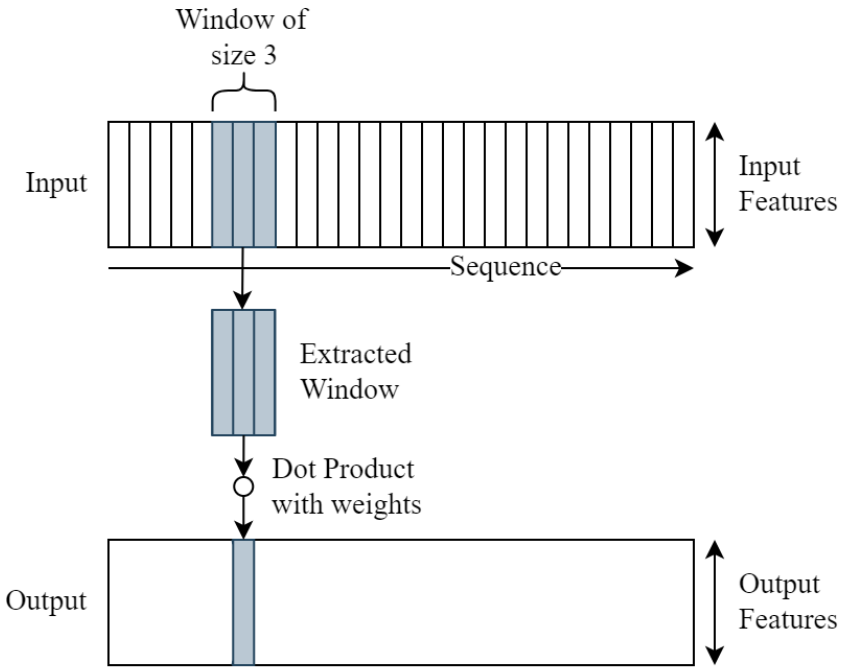


Figure 3.30: 1D convolutionary neural network

It is useful to detect patterns like spikes or drops in time-series data.

Batch Normalization: Batch normalization stabilizes learning by normalizing the output of the previous layer. It also adjusts activations to have a mean of zero and a standard deviation of one, scaling and shifting based on learned parameters.It increases training speed, allows for higher learning rates, and reduces the need for other regularization methods.

MaxPooling Layer: In Convolutional Neural Networks, pooling layers are used to reduce the dimensionality of the input volume, summarizing the features in the previous layer. In our case, the pooling size of 2 means the layer reduces the feature dimension by half, which helps to decrease computational load and minimize overfitting by providing an abstracted form of the representation.

Flatten: This layer converts the multi-dimensional output of previous layers into a single long feature vector. It is necessary for transitioning

from convolutional layers to dense layers, which require 1D input.

Dense Layer: A dense layer or fully connected layer, as shown in fig. 3.31 interprets the features extracted by convolutional and pooling layers to perform classification or regression. Every neuron in a dense layer is connected to all neurons in the previous layer, fully integrating the learned features.

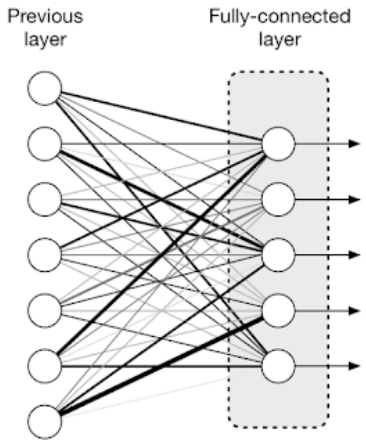
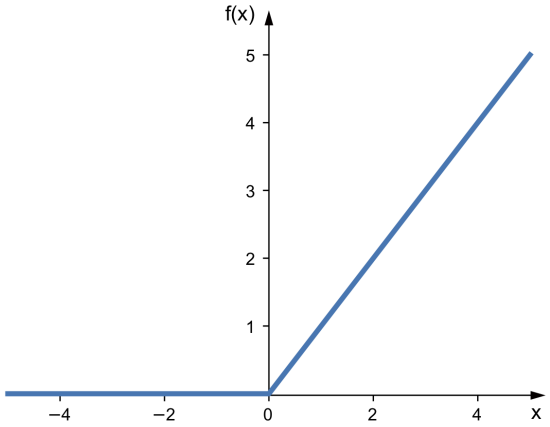


Figure 3.31: Fully Connected Layer[7]

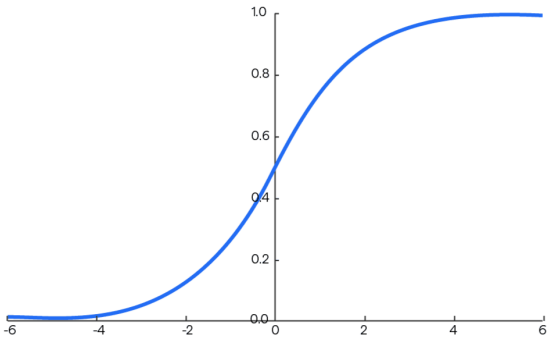
Dropout: This layer prevents overfitting by randomly dropping units (and their connections) during training. This layer prevents neurons from co-adapting too much, ensuring that the network generalizes well to unseen data.

Activation Function: Another important issue was choosing the right activation function. We used ReLU for the internal layers and softmax for the output layer. These activation layers add nonlinearity to the network. Graphically these function can be presented as in fig. 3.32

[7]Source: `https://sciagaprogramisty.blogspot.com`



(a) ReLU[8]



(b) Softmax[9]

Figure 3.32: Activation Functions

1. ReLU (Rectified Linear Unit): Mathematically, ReLU is defined as:$f(x) = max(0, x)$. Also it can be expressed as a piece-wise defined function:

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } otherwise \end{cases}$$

ReLU helps introduce non-linearity to the model, allowing it to learn complex patterns in the data. It is computationally efficient and helps alleviate the vanishing gradient problem.

2. Softmax: Softmax is typically used in the output layer of a neural network, like we included it in the last dense layer of our Conv1D model

[5]Source: `https://sebastianraschka.com/faq/docs/relu-derivative.html`
[6]Source: `https://botpenguin.com/glossary/softmax-function`

when the task involves classification. The softmax function is defined as:

$$\begin{aligned}\text{softmax}(x_i) &= \frac{e^{x_i}}{\sum_j e^{x_j}} \\ &= \frac{e^{x_i}}{e^{x_1} + e^{x_2} + \cdots + e^{x_n}}\end{aligned}$$

Where, x_i is the raw output score for class i, and the sum is over all classes. Softmax ensures that the output probabilities sum up to 1, making it suitable for multi-class classification tasks.

3.3.6.2 LSTM (Long Short-Term Memory)

Finally, we built an LSTM (Long Short-Term Memory) model, which is a type of recurrent neural network (RNN) designed to handle sequence prediction problems. LSTM is particularly suited for time series data or any data where the sequence is important because it can remember information for long periods, which is a key advantage over simpler RNNs that tend to lose information over time.

Here's a breakdown of the model configuration and its components:

- **Model Architecture:** Our LSTM model was composed with an input layer, an LSTM layer, and a dense layer as shown in fig. 3.33.

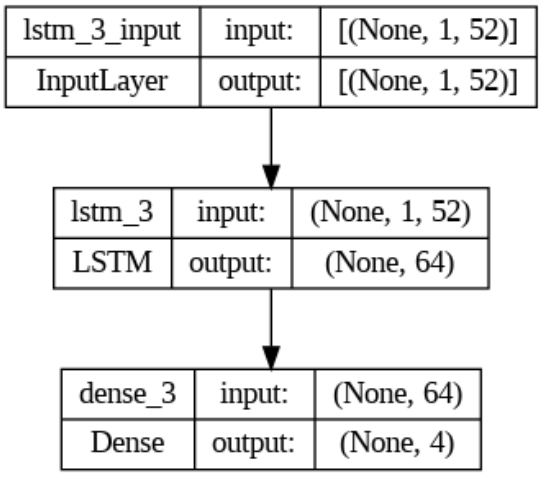


Figure 3.33: LSTM Model Architecture

1. LSTM Layer: We initiate the model with an LSTM layer consisting of 64 units. We set the input shape parameter to (number of time steps,

number of features per time step) format. It defines how the input data to the LSTM layer should be structured.

2. Dense Output Layer: Following the LSTM, there is a dense (fully connected) layer with four units for each gesture class. Here we used the softmax activation function because is standard for multi-class classification tasks as it outputs a probability distribution over the classes.

- **Model Summery:** The summary of the implementation of our LSTM model is described below in fig. 3.34:

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_3 (LSTM)               (None, 64)                29952

 dense_3 (Dense)             (None, 4)                 260

=================================================================
Total params: 30212 (118.02 KB)
Trainable params: 30212 (118.02 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 3.34: Summary of the LSTM Model Architecture

This LSTM model is typically set up for our experiment involving sequential time series data where we need to capture temporal dependencies and nuances across the data for accurate predictions.

3.4 Conclusion

Our suggested technique and every detail of our execution were covered in detail in this chapter. We went over every stage of our implementation process for our position-independent gesture recognition system. Here, we talked about implementing the hardware setup and data collection and how we fed the preprocessed data to the machine learning and deep learning models after that. We also discussed the suggested method's system requirements. The next chapter will provide an overview of our method's outcomes and efficacy.

Chapter 4

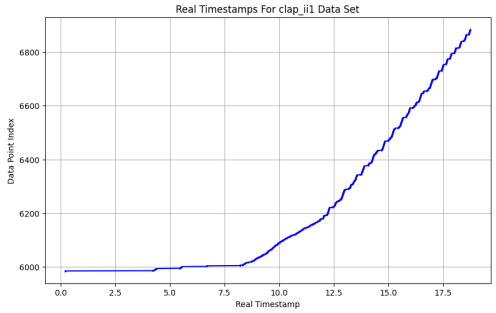
Results and Discussions

4.1 Introduction

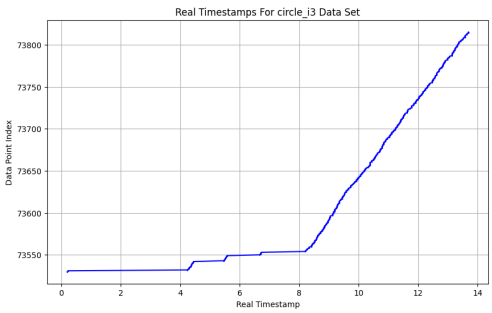
The previous chapter provided a detailed explanation of the process for creating a position-independent gesture classification system based on CSI data. Now that the process has been put into practice, it is time to evaluate the results. This chapter will examine the findings of the models that were employed as well as the method's performance evaluation. As our system works with multi-class classification, performance measures that will be utilized to assess our models are Accuracy, Precision, Recall, and Confusion Matrices. This chapter has also covered the results' graphs and charts. We have also examined CSI data and its changes for various gestures to gain further understanding. Finally, this chapter concludes with a comparison of the implemented models.

4.2 Data Analysis

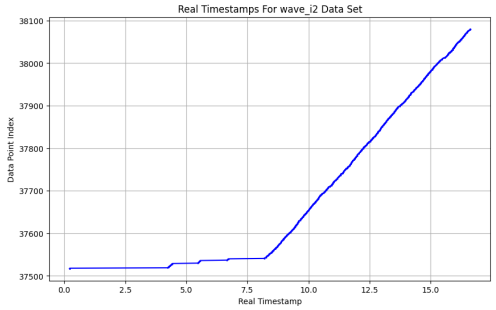
For this thesis work, we have collected 360 sets of training data from nine fixed positions and 20 sets of testing data from random positions. After collecting the data, we analyzed each set of data by plotting packet numbers for time, as in fig. 4.1. We found that our device started collecting data uniformly after 8 seconds of turning on the device.



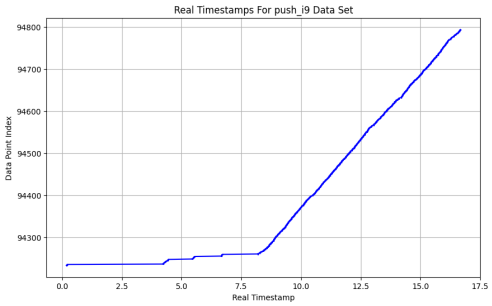
(a) Data Packets Vs Time



(b) Data Packets Vs Time



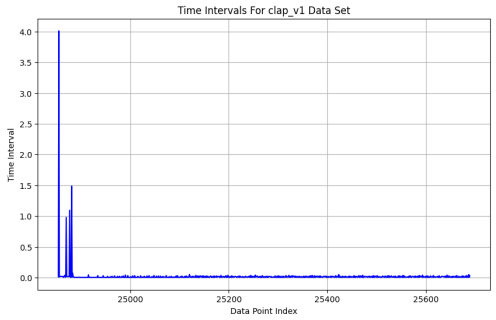
(c) Data Packets Vs Time



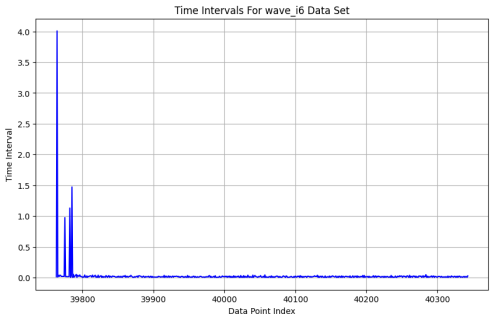
(d) Data Packets Vs Time

Figure 4.1: First Few-Second Delay for Uniform Data Collection

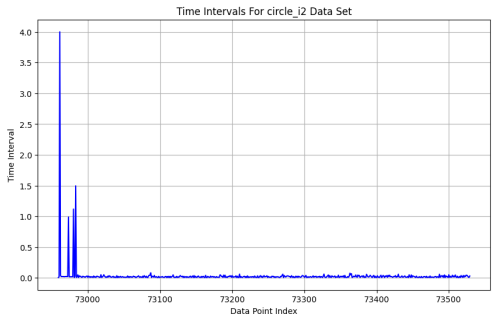
Also, by plotting time intervals between each consecutive data packet, as in fig. 4.2, we were able to visualize this delay.



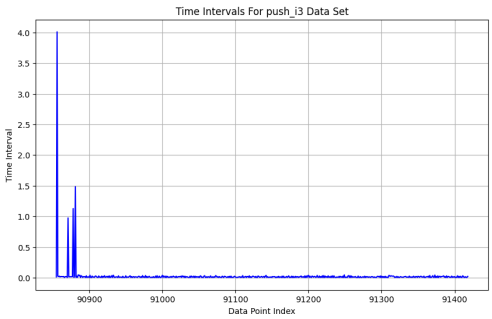
(a) Data Packets Vs Time



(b) Data Packets Vs Time



(c) Interval between Data Packets Vs Time

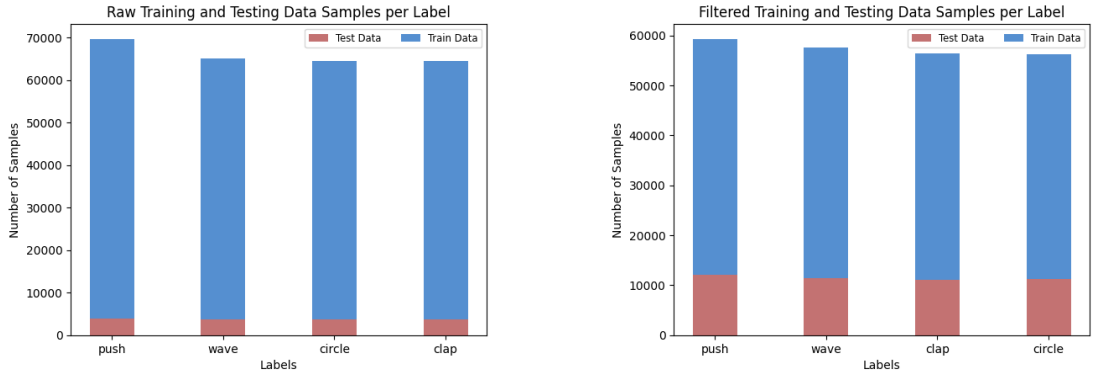


(d) Interval between Data Packets Vs Time

Figure 4.2: Time Intervals between consecutive Data Packets

From our 360 train data sets and 20 test data sets, initially, we got 2,63,660 raw

data packets. After filtering, and removing these 380 sets of CSI data, we got a total of 2,29,490 rows of CSI data as shown in fig. 4.3.



(a) Raw data Train-Test Distribution

(b) Filtered Data Train-Test Distribution

Figure 4.3: Train-Test Distribution, Before and After Filtering

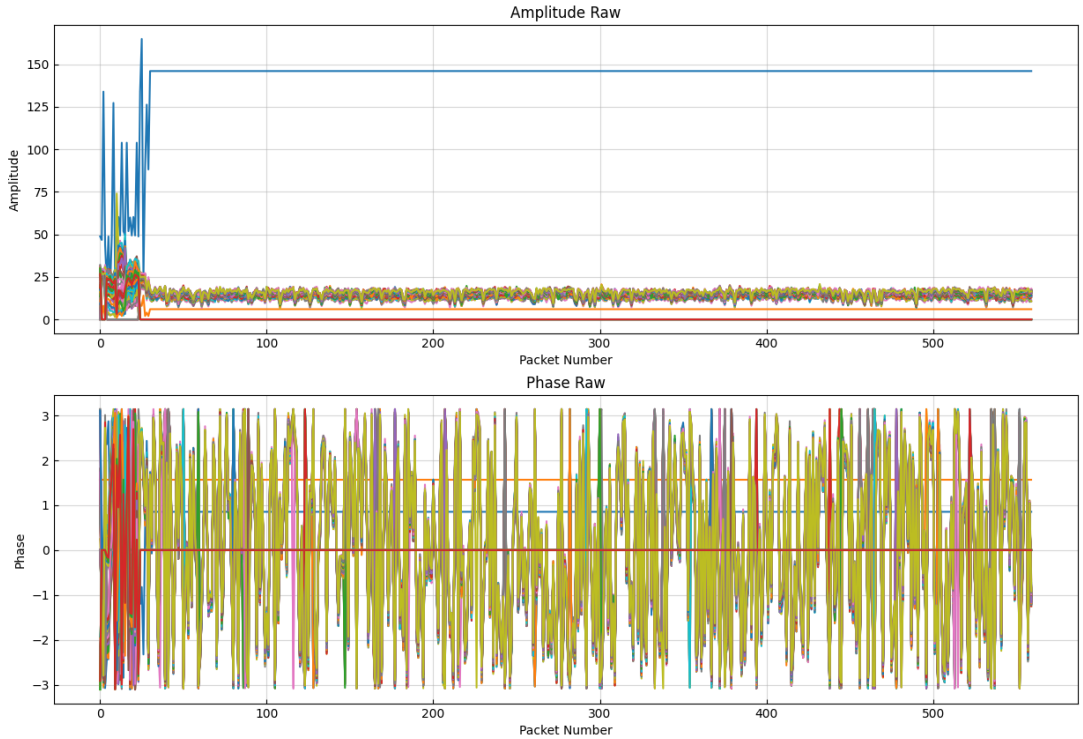
4.3 CSI Data Analysis

We have gathered CSI data on four gestures from nine fixed spots and five random positions in our lab space for our thesis project. Here, to understand the variations in each gesture, we have examined the data by showing its amplitude and phase sub-carriers with respect to time.

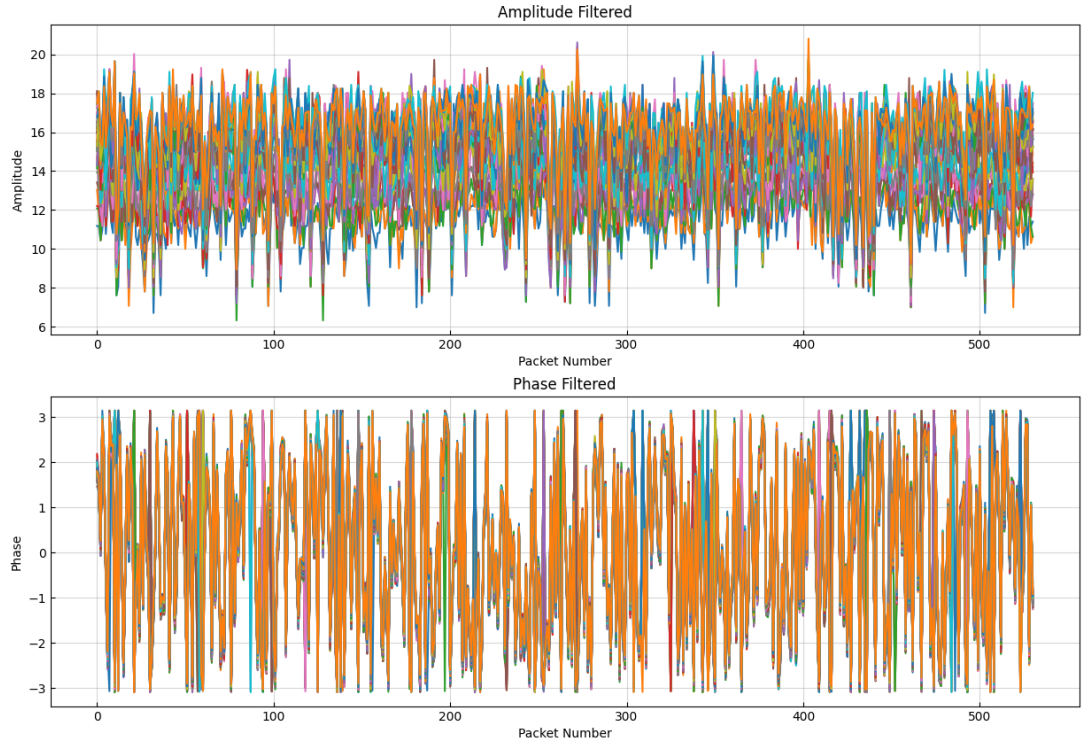
4.3.1 Amplitude and Phase Variation of CSI data

We have analyzed and visualized the amplitude and phase variations of the parsed CSI data for each gesture class before and after filtering. Filtering involved selecting data by high-throughput signals and eliminating all null sub-carriers. Below in fig. 4.4, fig. 4.5, fig. 4.6, & fig. 4.7 we present, a single instance of amplitude and phase signal profiles corresponding to each of the four gestures, before and after filtering.

- **CSI Data for Double Clap:**



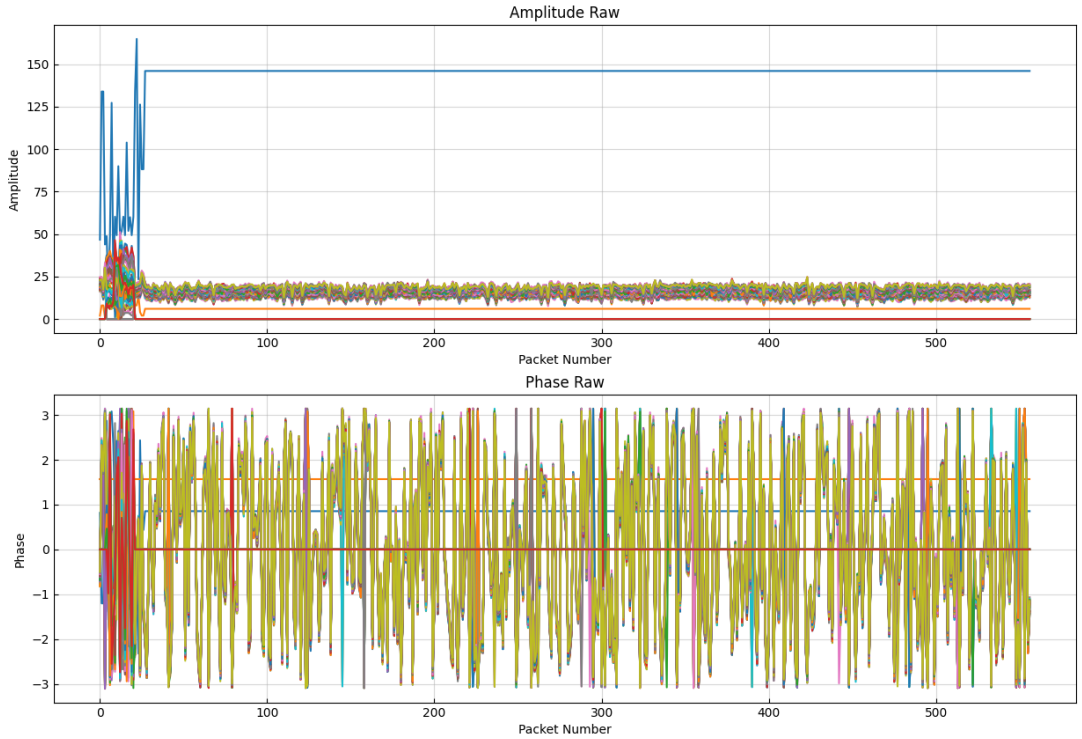
(a) Raw data



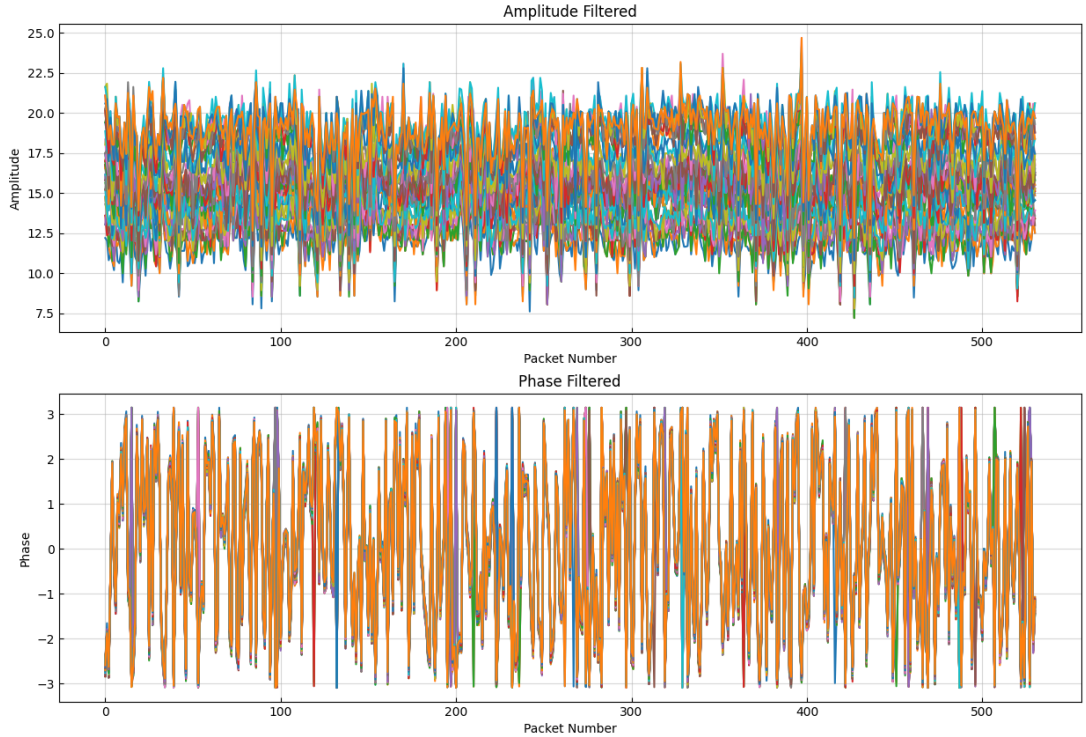
(b) Filtered data

Figure 4.4: Amplitude and Phase Variation Double Clap

- **CSI Data for Wave:**



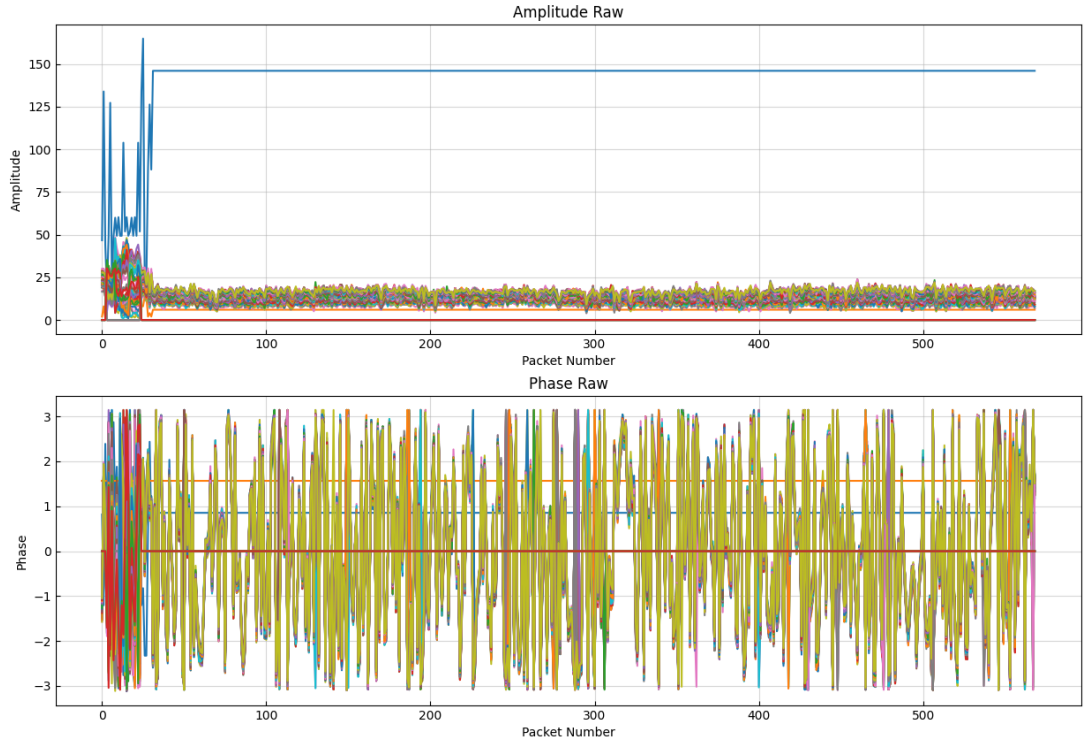
(a) Raw data



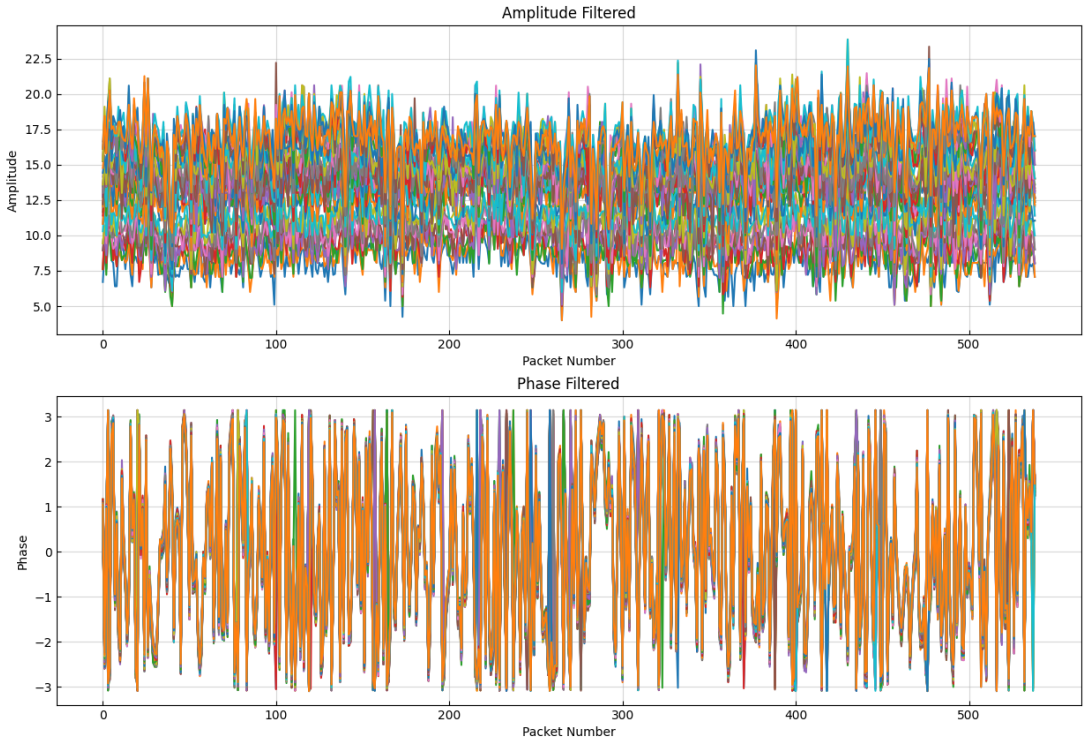
(b) Filtered data

Figure 4.5: Amplitude and Phase Variation Wave

- **CSI Data for Circular Motion:**



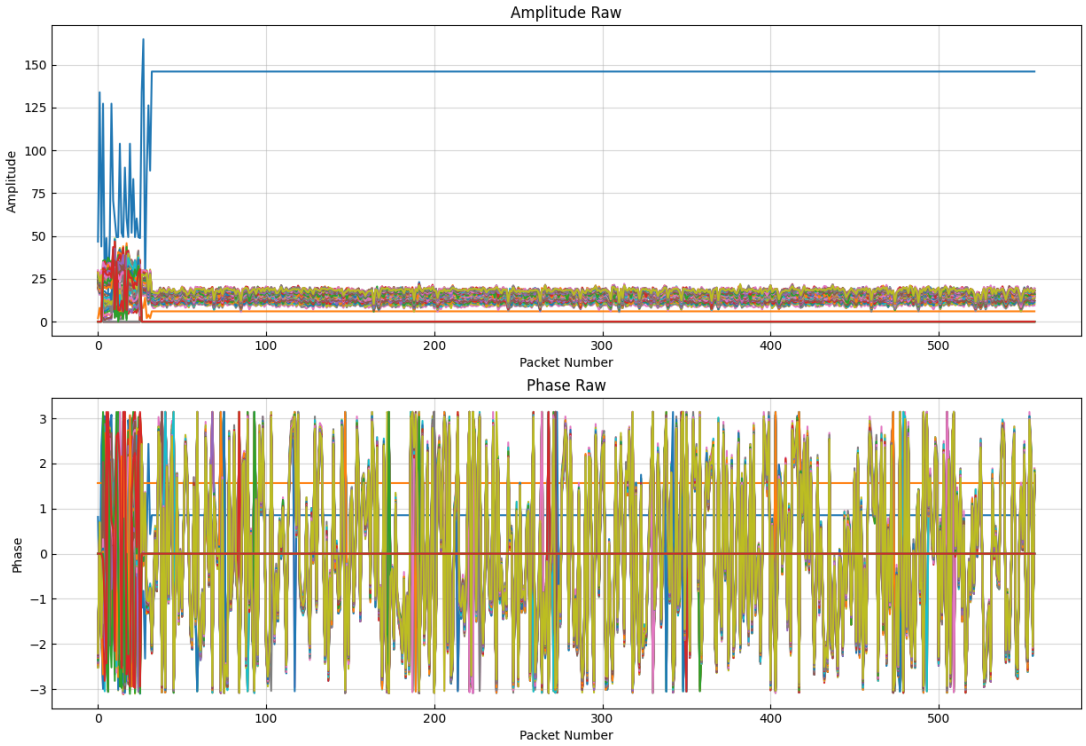
(a) Raw data



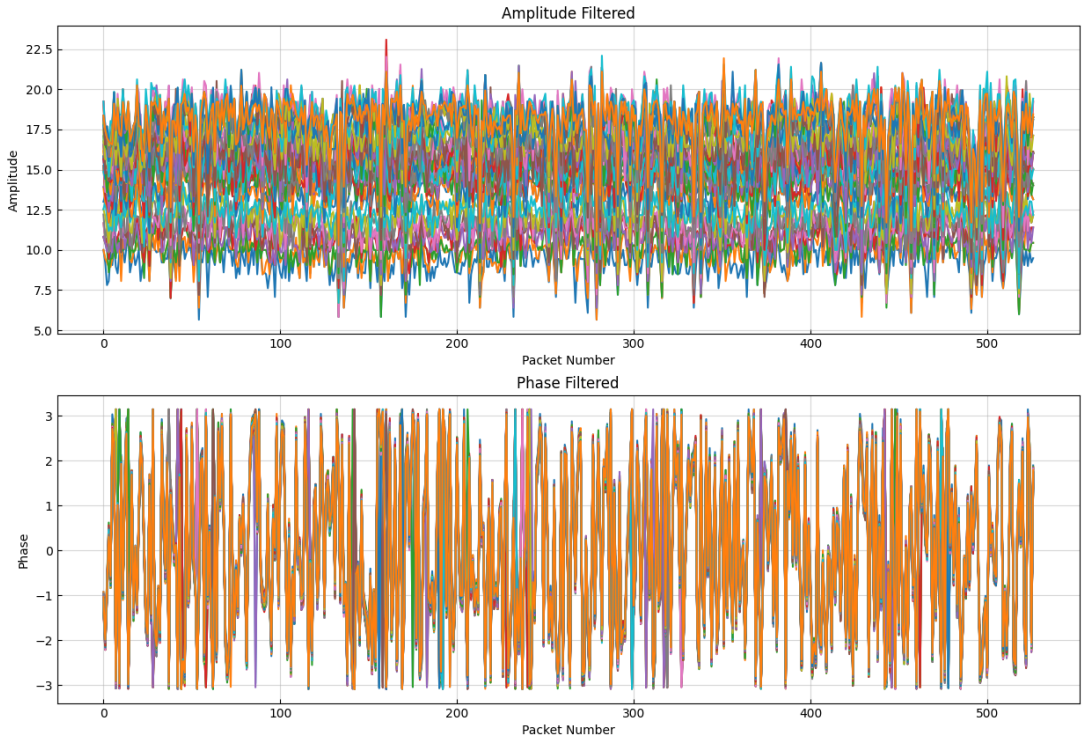
(b) Filtered data

Figure 4.6: Amplitude and Phase Variation Circular Motion

- **CSI Data for Double Push:**



(a) Raw data



(b) Filtered data

Figure 4.7: Amplitude and Phase Variation Double Push

4.3.2 Feature Importance

We also researched the importance of Amplitude and Phase values as features, in this system. While applying the Random Forest method, we utilized their built-in feature importance attribute. It measures the average gain of each feature. The feature importance of all 52 Amplitude sub-carriers and 52 Phase sub-carriers are shown in fig. 4.8. Here, attributes with the prefix 'A' represent one of the 52 sub-carriers, and attributes with the prefix 'P' represent one of the 52 phase sub-carriers. We found that Amplitude sub-carriers hold more significance than Phase sub-carriers in our CSI data. That's why in further works of this project, most of the time we ignored Phase properties in our system.

4.3.3 Heatmaps and Data Connections of CSI data

A heatmap of Channel State Information data is a graphical representation that visualizes the properties of wireless channels over time and frequency. In the context of CSI, the heatmap typically illustrates how certain characteristics—such as the amplitude or phase of the signal received under different subcarrier frequencies—change over time or in different conditions.

Here, in fig. 4.9 light colors denote low values, and dark ones denote high values. The colors span from light to dark. To visualize datasets and identify different patterns in the same class data, collected from different fixed points of the room, we utilized heatmaps.

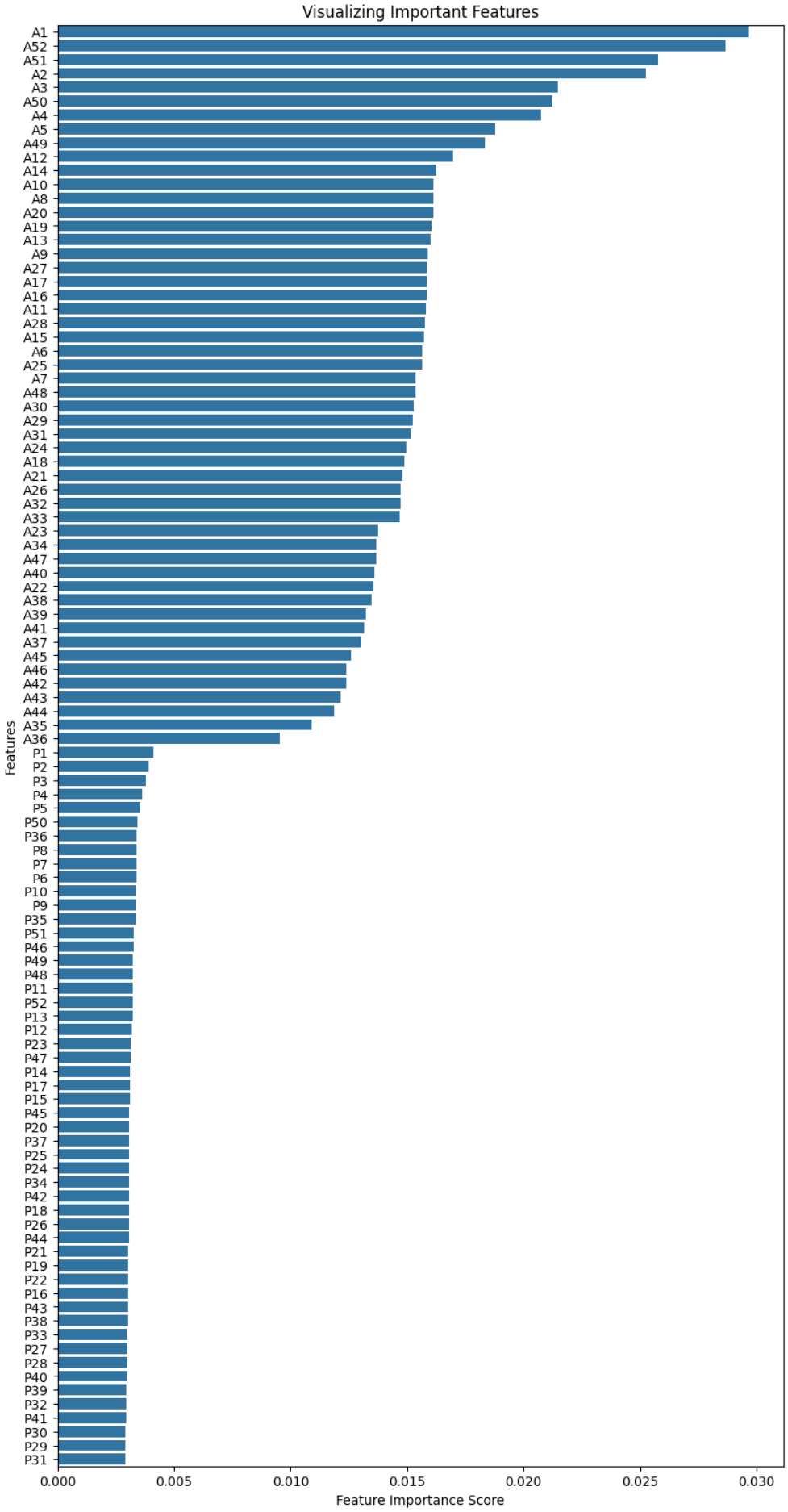
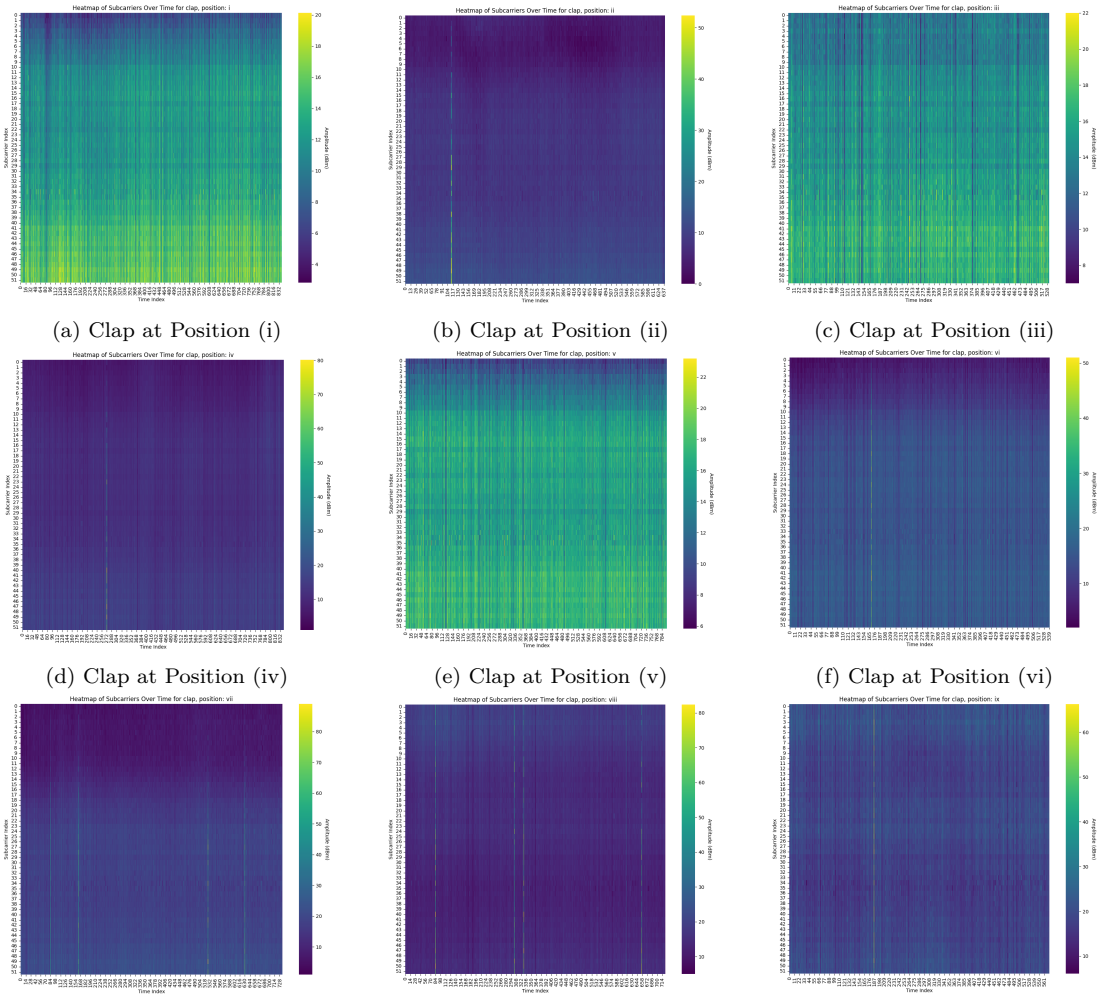


Figure 4.8: Feature Importance



(a) Clap at Position (i) (b) Clap at Position (ii) (c) Clap at Position (iii)

(d) Clap at Position (iv) (e) Clap at Position (v) (f) Clap at Position (vi)

(g) Clap at Position (vii) (h) Clap at Position (viii) (i) Clap at Position (ix)

Figure 4.9: Heatmaps for CSI data in different positions of 3*3 grid

This fig. 4.9 represents the same gesture class "Clap" from different positions of the room. We can see significant changes in shades of the heatmaps according to the distance of the actor from the transmitter and receiver.

4.4 Evaluation Measures

For the purpose of evaluating our proposed position-independent gesture classification model, we have considered a number of metrics such as Accuracy, Precision, Recall, F1 score, and the use of Confusion Matrices. Here we present the details about these metrics in brief.

- **Confusion Matrix**: A table that describes how well a classification model

performs on a set of data for which the true values are known is called a confusion matrix.

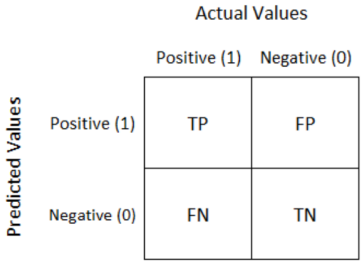


Figure 4.10: Confusion Matrix

As shown in fig. 4.10, a confusion matrix provides a summary of the number of accurate and inaccurate predictions, broken down by class using count values. This is essential for visualizing any classifier's performance since it makes it simple to identify instances of class misunderstanding.

- **Accuracy**: One of the most logical performance metrics for classification models is accuracy. It can be expressed simply as the ratio of accurately anticipated observations to all observations. It indicates the proportion of total classifications that were accurate. The equation used to determine accuracy is given by:

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{4.1}$$

  Better performance is indicated by a higher accuracy rate; perfect categorization is denoted by 100% accuracy.

- **Precision**: The ratio of accurately predicted positive observations to the total number of predicted positives is known as precision. It is an indicator of how well the optimistic predictions turned out. When False Positives are more concerning than False Negatives, precision is very helpful. The precision for each class is defined as:

$$Precision = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{4.2}$$

- **Recall** (or Sensitivity): The ratio of accurately predicted positive observations to all actual class observations is known as recall. It gauges how well

the model can identify positive samples. The results are more comprehensive the higher the recall. The recall for each class is calculated as:

$$Recall = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{4.3}$$

- **F1 Score**: Recall is defined as the proportion of correctly predicted positive observations to all actual class observations. It evaluates the model's ability to recognize positive samples. The higher the recall, the more precise the results are. The F1 Score is defined as:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.4}$$

Each of these measures gives different insights into the performance of the classification model and, when used together, provides a comprehensive view of model effectiveness, especially in scenarios involving multiple classes.

4.5 Impact Analysis

The demand for intuitive and seamless interaction with technology in smart environments is escalating. The advent of a CSI-based position-independent gesture classification system, as presented in our thesis, offers a novel and effective way to interface with smart home appliances without the need for physical contact or additional hardware. This system leverages the ubiquitous presence of WiFi, making it an ideal solution in modern smart homes and other IoT environments. Below we discuss the various impacts of this technology:

4.5.1 Impact on Human-Computer Interaction

Our proposed system significantly enhances the ease of interaction within smart environments and reduces the need for traditional physical controls. This method of gesture recognition can be seamlessly integrated into any IoT-enabled environment such as smart homes, smart offices, healthcare facilities, and more, enhancing user interaction without the physical limitations of traditional interfaces. Thus, it will leave a great impact on HCI.

This system introduces a more accessible and inclusive way to control home environments, which is particularly beneficial for individuals with mobility, and speech disability. It increases comfort and convenience by allowing gesture recognition irrespective of the actor's position within the room, enhancing user experience and interaction with the environment.

4.5.2 Economic Impact

The use of the ESP-32 device for data collection is considerably less expensive than other devices that use computer vision or sensors like Kinect xbox. Hence it reduces the cost of deployment.

Additionally, implementing our CSI-based gesture recognition system can lead to significant cost savings in smart environments by reducing the need for additional sensors and controls. Facility managers and homeowners can retrofit existing WiFi systems with our technology, avoiding the higher costs associated with other smart control systems that require new infrastructure or specialized equipment.

4.5.3 Ethical and Privacy Impact

Our proposed system uses Channel State Information (CSI) to recognize human gestures, and reduce privacy concerns, compared to computer vision-based systems that require video data. This approach avoids the need to capture sensitive visual recordings and preserves the privacy of individuals.

However, it may also raise important ethical and privacy considerations. Since the system operates using WiFi signals, there is a potential for misuse if the CSI data is intercepted or misappropriated. It is crucial to implement strong encryption and access controls to ensure that gestures and their associated data remain secure and private. Moreover, transparency regarding data collection and use is essential to maintain trust and ensure ethical deployment.

In summary, our thesis work not only pushes forward the boundaries of IoT and smart home technology but also presents new challenges and responsibilities in the realms of data security and ethical use of technology. As the implementation of

such systems expands, continuous attention to these concerns will be imperative for sustaining user trust and safety.

4.6 Evaluation of Performance

In this thesis work, after collecting and labeling CSI data we have fed it into both machine learning and deep learning models and examined the accuracy, precision, recall, and f1_score resulting from the models in this section. As machine learning approaches, we have used Ensemble Learning methods like Random Forest, XGBoost, LGBM, and Voting classifier to ensemble all these methods. We also implemented a ShapeletLearning method to utilize the time-series property of our data. As deep learning methods, we used CNN and LSTM approaches to classify our data. In this section, we will present some performance comparisons and analyses of our work.

4.6.1 Machine Learning Model

Here we presented the overall performance of our implemented machine-learning models which are the Ensemble Learning models and the Shapelet Learning model as mentioned earlier. As ensemble learning models we have used three base models LGBM Classifier, XGB Classifier, and Random Forest Classifier. To ensemble these models we also implemented a Voting classifier in our model. Later, to utilize the time-series property of our CSI data, we generated Shapelet Learning model to classify our data.

4.6.1.1 Ensemble Learning Methods

In our work, we have used the ensemble learning methods stated above. In table 4.1, we have shown the Accuracy, Precision, Recall, and F1 Score of these models to analyze their overall performance.

Table 4.1: Overall Model Performance of Ensemble Learning

Model	Accuracy	Precision	Recall	F1_Score
Light-GBM	67.9%	68.0%	67.9%	67.8%
XGBoost	68.7%	68.9%	68.7%	68.6%
Random Forest	80.5%	80.6%	80.5%	80.4%
Voting Classifier	75.5%	75.7%	75.5%	75.4%

We can visualize this comparison in fig. 4.11.

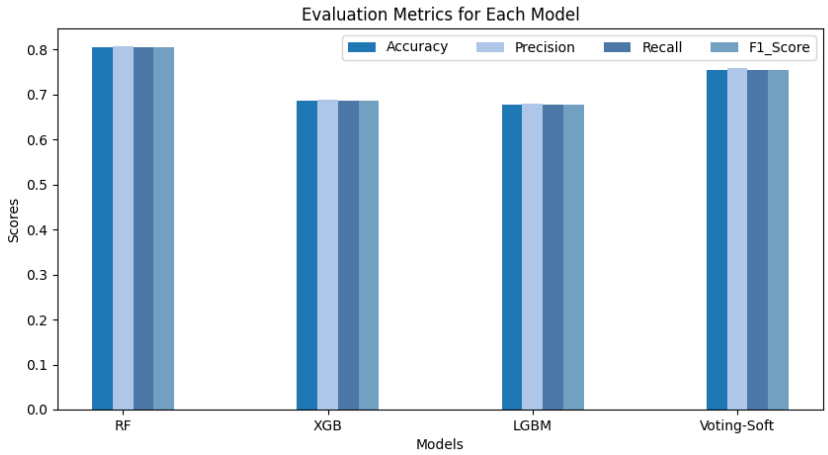


Figure 4.11: Evaluation of Ensemble Models

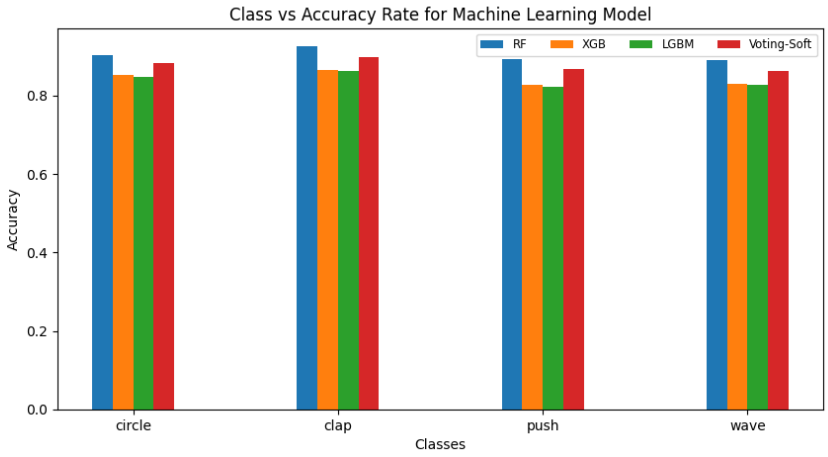


Figure 4.12: Accuracy of Classifying each Gesture by each Ensemble model

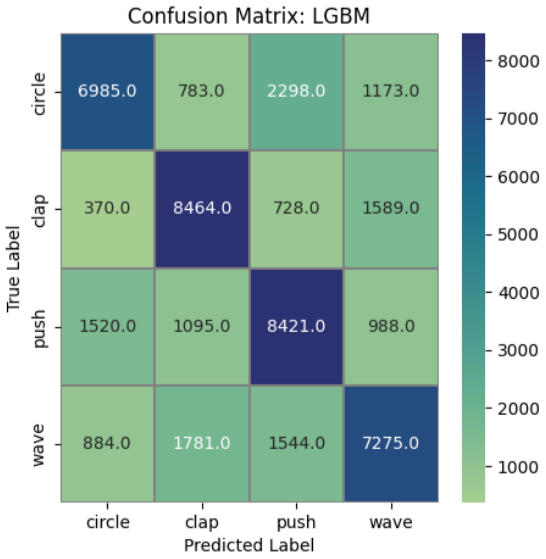


Figure 4.13: Confusion Matrix for Light-GBM

Table 4.2: Result Illustration for LGBM Classifier

Class	Accuracy	Precision	Recall	F1_Score
Circle	84.68%	71.57%	62.14%	66.53%
Clap	86.17%	69.81%	75.90%	72.73%
Push	82.19%	64.82%	70.03%	67.32%
Wave	82.65%	65.98%	63.34%	64.64%

To see how accurately each class is classified in each of these models, in fig. 4.12 we plotted the accuracy of classifying each class by each of these models.

Confusion Matrices and evaluation measures of each class on each model are presented below:

- **Light-GBM:** Confusion matrix for LGBM model is shown in fig. 4.13, and performance measure per class using this model is shown in table 4.2.
- **XGBoost:** Confusion matrix for XGB model is shown in fig. 4.14, and performance measure per class using this model is shown in table 4.3.
- **Random Forest:** Confusion matrix for Random Forest model is shown in fig. 4.15, and performance measure per class using this model is shown in table 4.4.
- **Voting Classifier on previous models:** Confusion matrix for ensemble

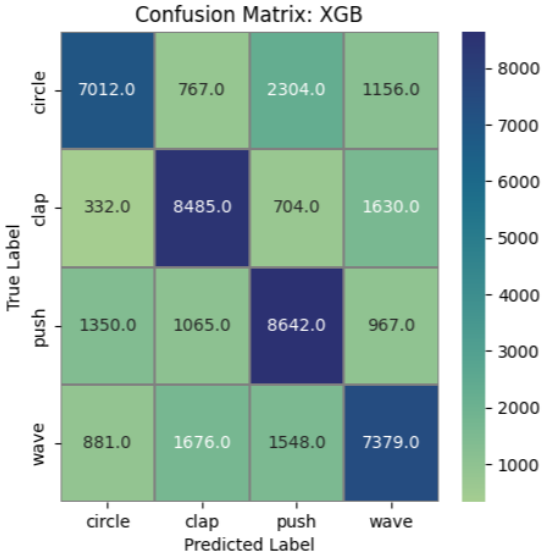


Figure 4.14: Confusion Matrix for XGBoost

Table 4.3: Result Illustration for XGBoost Classifier

Class	Accuracy	Precision	Recall	F1_Score
Circle	85.20%	73.23%	62.38%	67.37%
Clap	86.54%	70.74%	76.09%	73.32%
Push	82.70%	65.47%	71.87%	68.52%
Wave	82.87%	66.28%	64.25%	65.25%

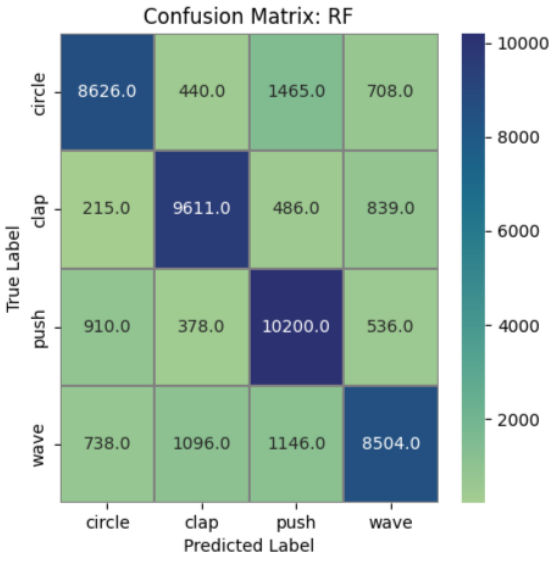


Figure 4.15: Confusion Matrix for Random Forest

Table 4.4: Result Illustration for Random Forest Classifier

Class	Accuracy	Precision	Recall	F1_Score
Circle	90.24%	82.23%	76.75%	79.39%
Clap	92.47%	83.39%	86.18%	84.76%
Push	89.27%	76.70%	84.83%	80.56%
Wave	88.96%	80.32%	74.50%	77.06%

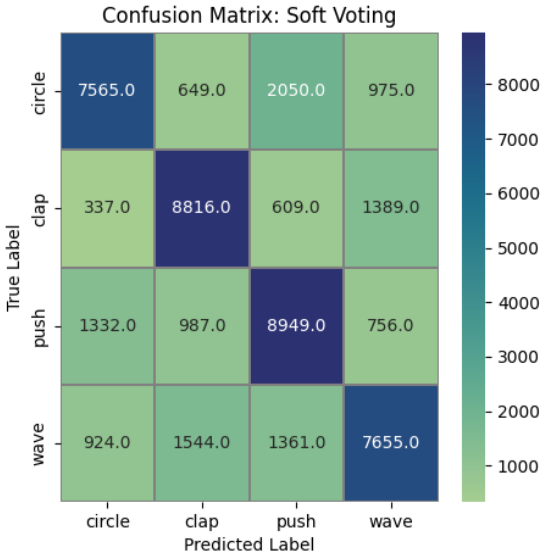


Figure 4.16: Confusion Matrix for Soft-Voting on LGBM, SGB and RF

method, Soft Voting model, is shown in fig. 4.16, and performance measure per class using this model is shown in table 4.5.

4.6.1.2 Shapelet Learning Method

To utilize the time-series property of our data, we trained a Shapelet Learning model. Shapelet Learning uses a stochastic gradient learning algorithm for choosing the optimal shapelet from time series data. Hence it is categorized as a

Table 4.5: Result Illustration for Soft Voting Classifier

Class	Accuracy	Precision	Recall	F1_Score
Circle	88.23%	80.06%	69.16%	74.21%
Clap	89.74%	77.57%	81.28%	79.38%
Push	86.66%	71.88%	80.63%	76.00%
Wave	86.36%	73.71%	70.71%	72.18%

machine learning approach.

Shapelets are discriminative sub-sequences of time series that best predict the target variable. For our work, we obtained 5 shapelets of size 20 presented in fig. 4.17, which best predicts time-series data.

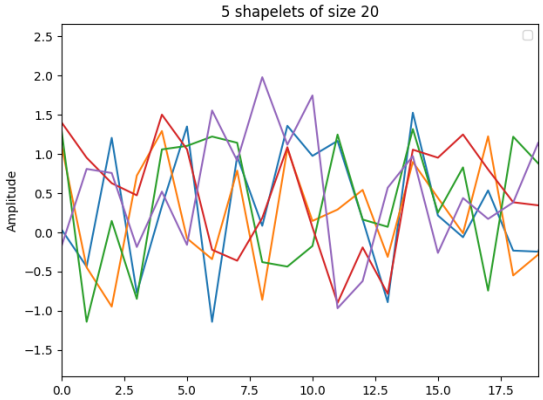


Figure 4.17: Top 5 Candidate Shapelets

The accuracy, precision, recall, and F1 scores obtained from this method are described in table 4.6.

Table 4.6: Performance of Shapelet Learning Model

Model	Accuracy	Precision	Recall	F1_Score
Learning Shapelet	74.5%	77.7%	74.5%	74.0%

The test set in this model had a total of 247 time-series windows. The performance of our model on these data is shown below in fig. 4.18, in confusion matrix format.

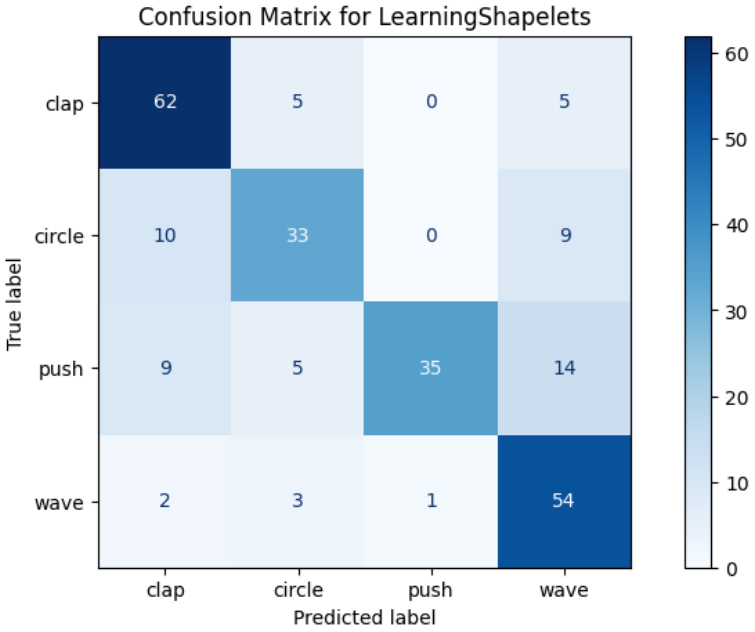
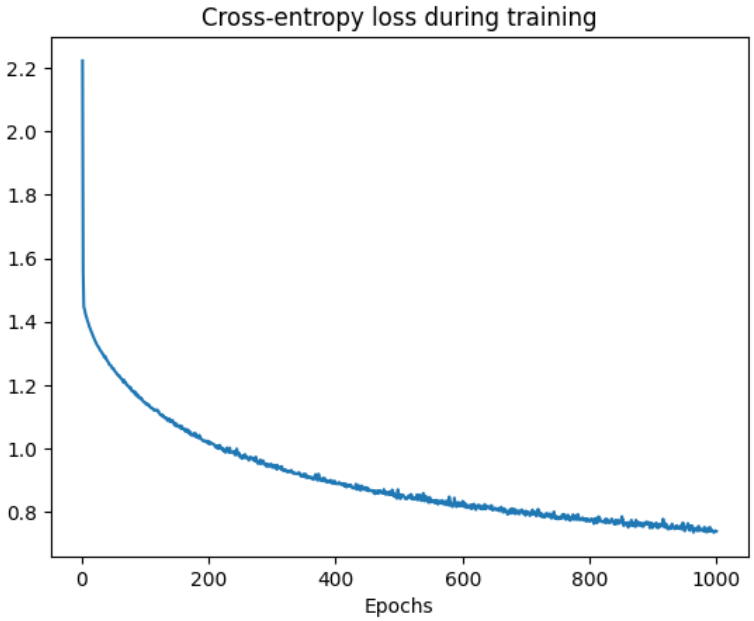
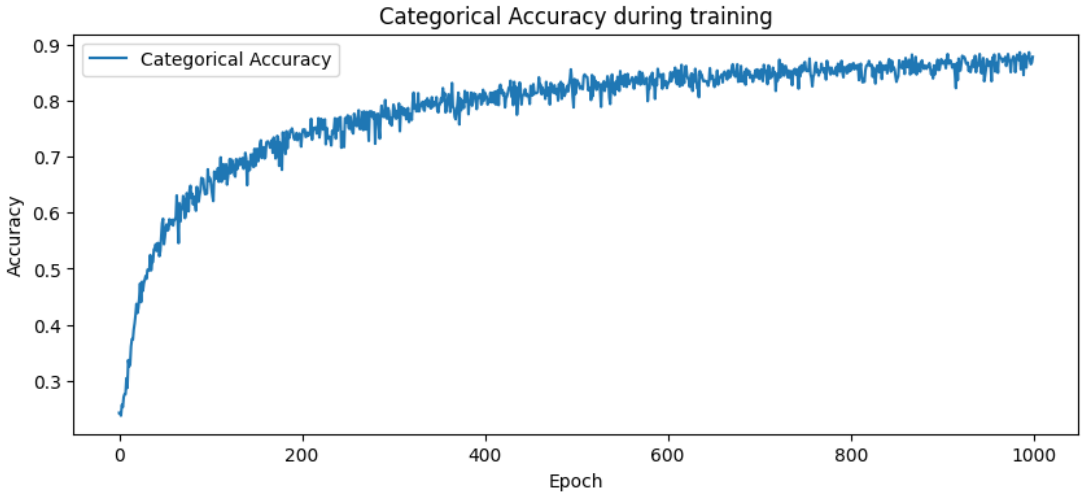


Figure 4.18: Shapelet Learning Confusion Matrix

During training the model reached 87.83% categorical accuracy. The evaluation of loss and accuracy during training the model is plotted in fig. 4.19:



(a) Evaluation of Loss



(b) Evaluation of Accuracy

Figure 4.19: Evaluation of Cross-Entropy loss and Categorical Accuracy over each Epoch

4.6.2 Deep Learning Model

For our proposed system, we have also implemented two deep-learning methods for gesture classification. The outcomes from these deep learning models are described here.

4.6.2.1 Convolutional Neural Network(CNN)

For this system, we tailored a CNN model with 1D convolution layers.

Our dataset has 104 attributes per time step. Among these 104 attributes, 52

are amplitude sub-carriers, and 52 are phase sub-carriers. We studied our CNN model performance taking only amplitude sub-carriers, only phase sub-carriers, and both amplitude and phase sub-carriers as features.

We found that amplitude sub-carriers are more informative and give more optimal results than using only phase sub-carriers. But if we use both, we get the optimal result. Here, in table 4.8 we describe the difference in the efficiency of the CNN model using different sets of features.

Table 4.7: CNN Model Performance over different feature set

Feature	Accuracy	Precision	Recall	F1_Score
Amplitude	73.31%	73.71%	73.31%	73.51%
Phase	59.77%	62.84%	59.77%	61.27%
Amplitude & Phase	76.55%	77.48%	76.55%	77.01%
Weighted-Average	78.13%	78.69%	78.13%	78.41%

Evaluation measures for different feature sets are shown in fig. 4.21b.

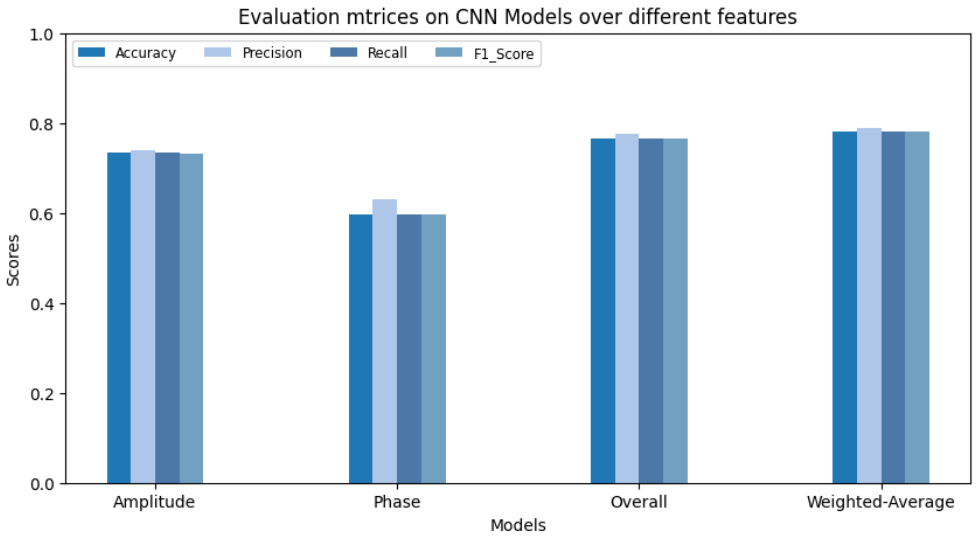
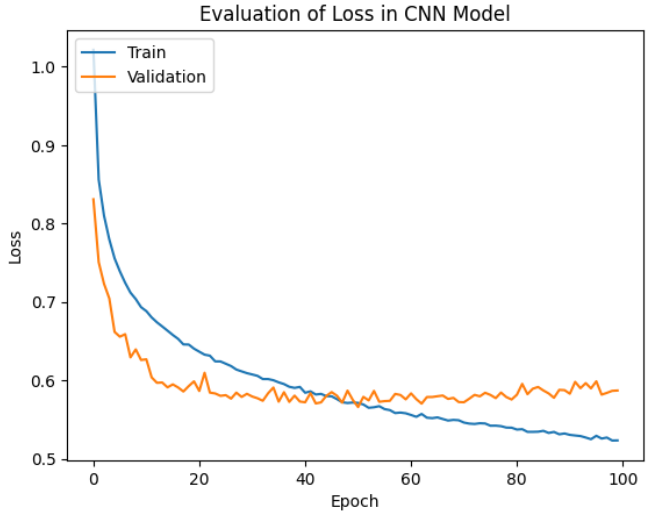


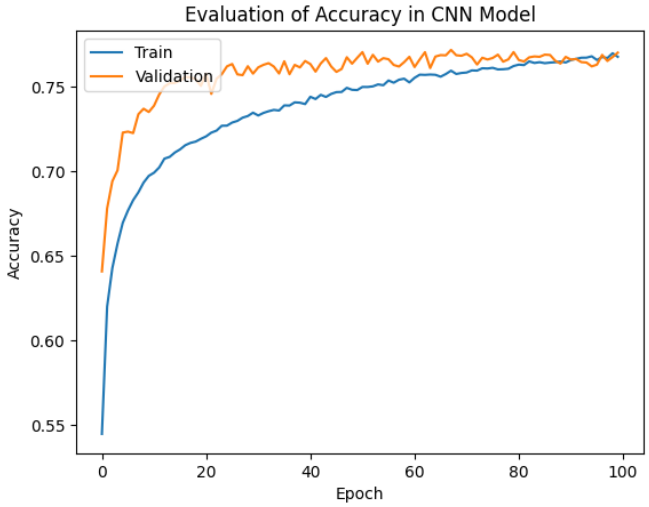
Figure 4.20: Evaluation of CNN model, using Different Feature set

Per class Accuracy comparison koi??

Evaluation of Loss and Accuracy, during training CNN model, with amplitude sub-carriers as features are shown in fig. 4.21:



(a) Evaluation of loss during training CNN model



(b) Evaluation of accuracy during training CNN model

Figure 4.21: Evaluation of loss and Accuracy over each Epoch in CNN

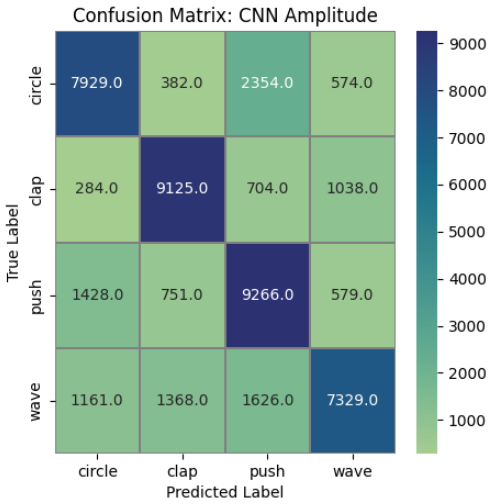


Figure 4.22: CNN with Amplitude Subcarriers as Features

Table 4.8: Only Amplitude Sub-carriers

Class	Accuracy	Precision	Recall	F1_Score
Circle	73.40%	73.40%	70.55%	71.95%
Clap	78.48%	78.49%	81.83%	80.12%
Push	66.42%	66.42%	77.06%	71.35%
Wave	76.98%	76.99%	63.82%	69.79%

- **Only Amplitude Sub-carriers:** Confusion matrix of CNN model, using only 52 amplitude subcarriers as features is shown in fig. 4.22, and performance measure per class using this model is shown in table 4.8.
- **Only Phase Sub-carriers:** Confusion matrix of CNN model, using only 52 phase subcarriers as features is shown in fig. 4.23, and performance measure per class using this model is shown in table 4.9.

Table 4.9: Only Phase Sub-carriers

Class	Accuracy	Precision	Recall	F1_Score
Circle	71.67%	71.67%	46.71%	56.66%
Clap	73.72%	73.73%	51.90%	60.92%
Push	55.04%	55.05%	67.91%	60.81%
Wave	51.79%	51.79%	71.66%	60.12%

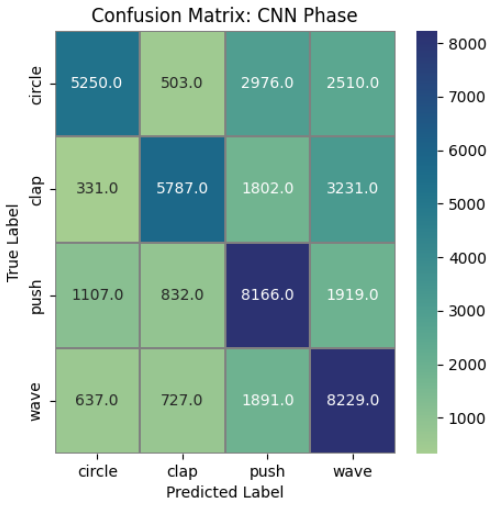


Figure 4.23: CNN with Phase Subcarriers as Features

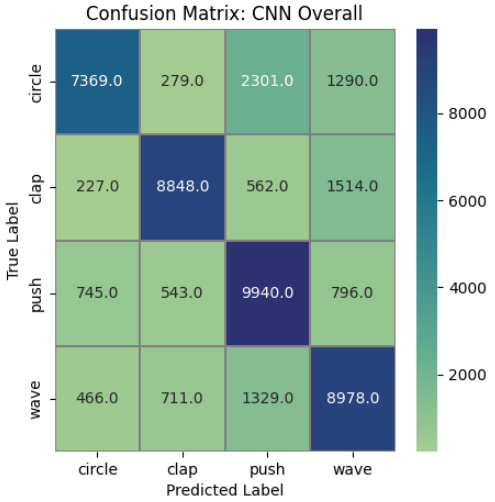


Figure 4.24: CNN with both Amplitude & Phase Subcarriers as Features

- **Both Amplitude & Phase Sub-carriers:** Confusion matrix of CNN model, using both 52 amplitude subcarriers and 52 phase subcarriers as features is shown in fig. 4.24, and performance measure per class using this model is shown in table 4.10.
- **Weighted Average of previous three models:** Confusion matrix of the weighted average of previous three CNN models, is shown in fig. 4.25, and performance measure per class using this model is shown in table 4.11.

Table 4.10: Both Amplitude & Phase Sub-carriers

Class	Accuracy	Precision	Recall	F1_Score
Circle	83.67%	83.67%	65.57%	73.52%
Clap	85.23%	85.23%	79.35%	82.18%
Push	70.33%	70.34%	82.67%	76.01%
Wave	71.37%	71.34%	78.18%	74.62%

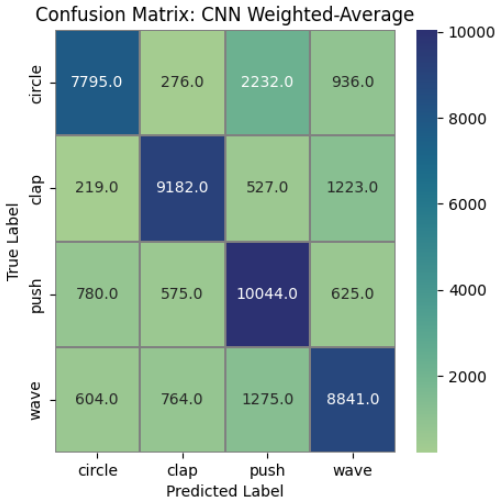


Figure 4.25: Weighted-Average Model

Table 4.11: Weighted Average CNN model

Class	Accuracy	Precision	Recall	F1_Score
Circle	82.94%	82.94%	69.36%	75.54%
Clap	85.04%	85.04%	82.34%	83.67%
Push	71.34%	71.35%	85.53%	76.96%
Wave	76.51%	76.05%	76.99%	76.52%

4.6.2.2 Long Short-Term-Memory(LSTM)

LSTM networks are capable of learning and remembering patterns over long sequences of data. This is crucial for time series data classification tasks, like ours where past observations influence future predictions. That's why finally, we implemented an LSTM model. Analyzing the low effectiveness of phase sub-carriers in previous models, we used amplitude sub-carriers only, in this model. The accuracy, precision, recall, and F1 scores obtained from this method are described in table table 4.12.

Table 4.12: LSTM model Performance

Class	Accuracy	Precision	Recall	F1_Score
LSTM	81.11%	81.10%	81.11%	81.11%

Confusion Matrices and evaluation measures of each class are presented below in fig. 4.26:

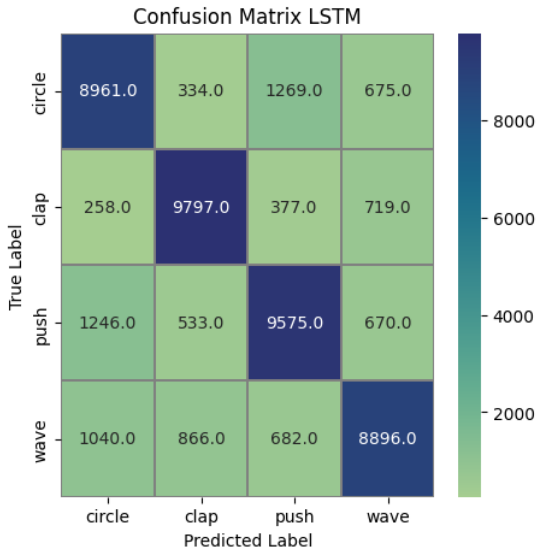


Figure 4.26: LSTM Confusion Matrix

Performance measure of LSTM model, per class is shown in table 4.13.

Table 4.13: Evaluation Measures of each class

Class	Accuracy	Precision	Recall	F1_Score
Circle	77.88%	77.89%	79.73%	78.80%
Clap	84.96%	84.97%	87.86%	86.39%
Push	80.44%	80.44%	79.63%	80.04%
Wave	81.16%	81.17%	77.46%	79.27%

Evaluation of loss and accuracy, during model training as in fig. 4.27:

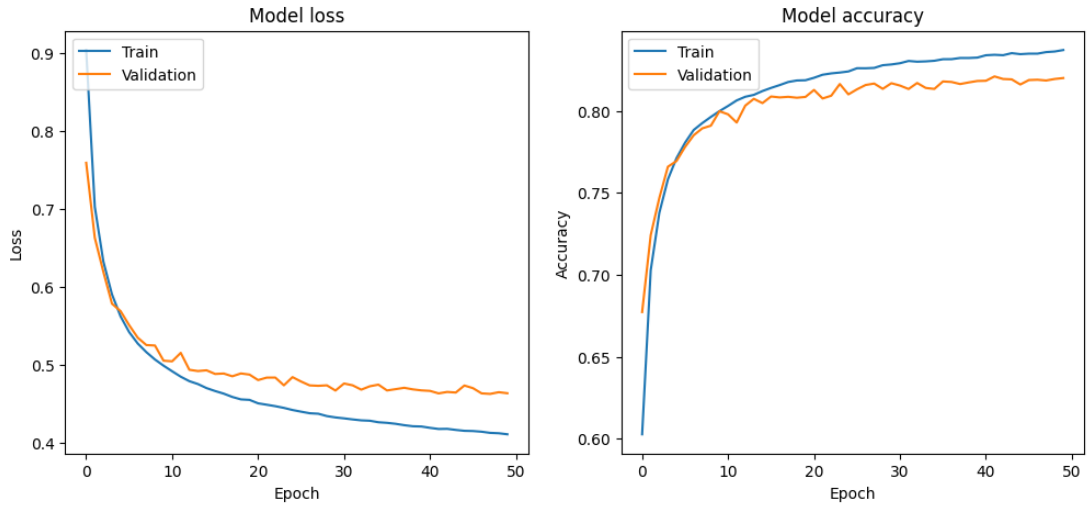


Figure 4.27: Evaluation of Loss and Accuracy during training

4.6.3 Model Performance Comparison

Here in this section, we have tried to compare our implemented models. LSTM method achieved the highest accuracy than any other machine learning, or deep learning methods, demonstrating the effectiveness of long-term memory for sequential data.

In the machine learning ensemble approaches, we have used three base models in XGBoost, Light-GBM, and Random Forest. Then used Soft Voting classifier to ensemble these models. We have also implemented a shapelet learning model. Performance differences among each model are shown in fig. 4.28.

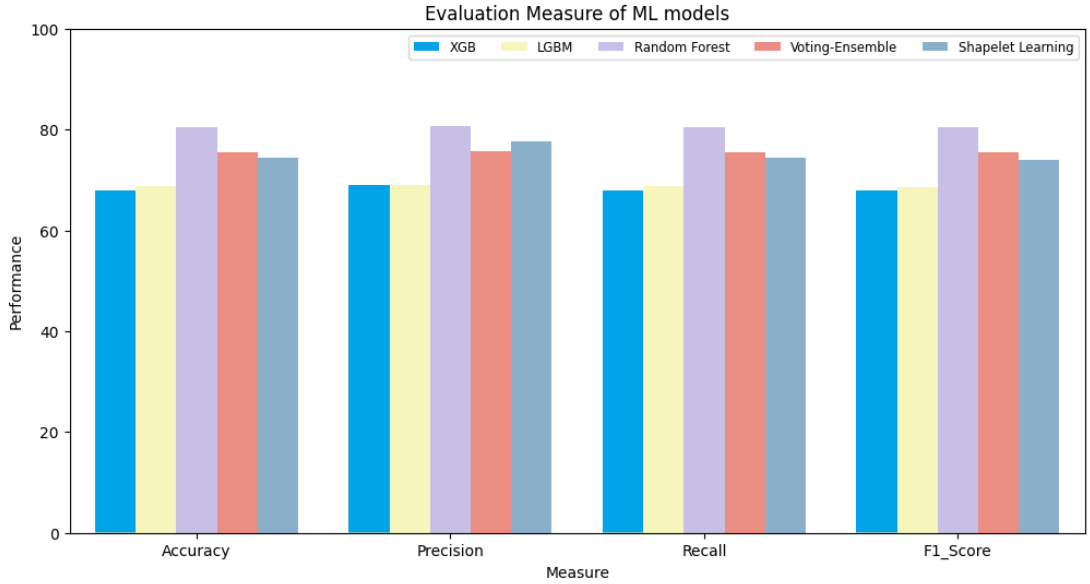


Figure 4.28: Comparison among Machine Learning Models

In deep learning, we implemented a CNN and an LSTM model. We can see the performance difference between them in fig. 4.29

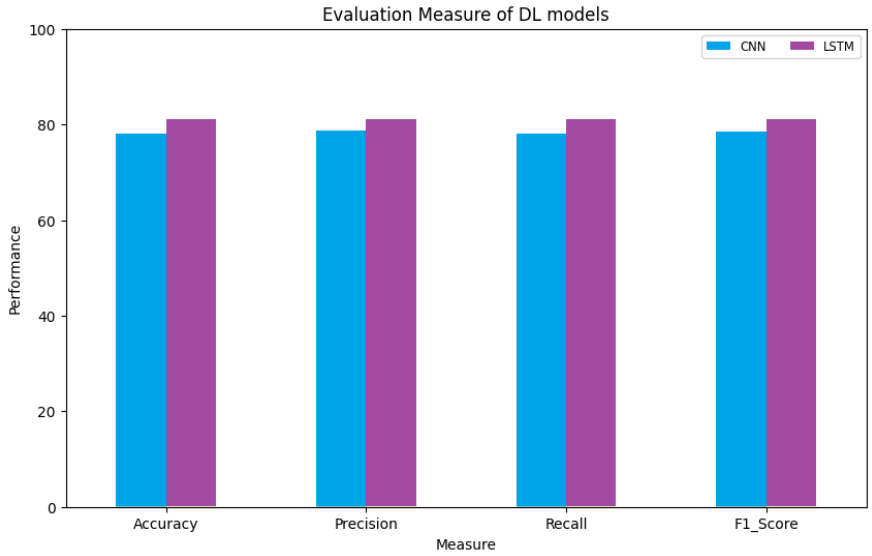


Figure 4.29: Comparison among Deep Learning Models

Evaluating both Machine learning, and deep learning approaches, we can see that Random Forest(80.5% accuracy) is the best machine learning approach and LSTM(81.11% accuracy) is the best deep learning approach for our system. Comparing them, we can see in fig. 4.30 that LSTM surpasses Random Forest with slightly better performance.

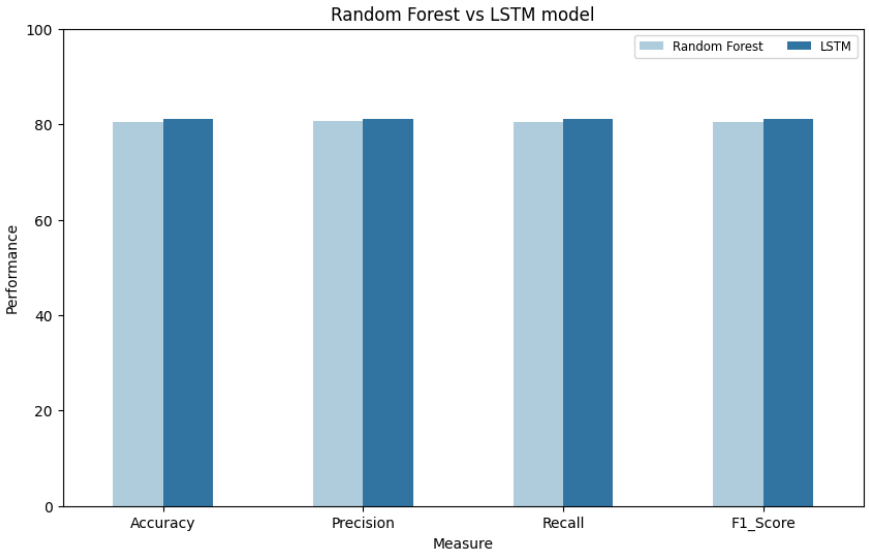


Figure 4.30: Random Forest vs LSTM model

4.6.4 Comparison Between Datasets

In our system, we have used 2 types of data. Our train data was from nine fixed positions, and train data was from random positions. To check, if our models perform better in fixed positions or not, we have also analyzed our models with fixed position data only. And we found that all of our models perform relatively better in this case than using both fixed and random points.

Differences between machine learning models, with random points and fixed points are shown in fig. 4.31.

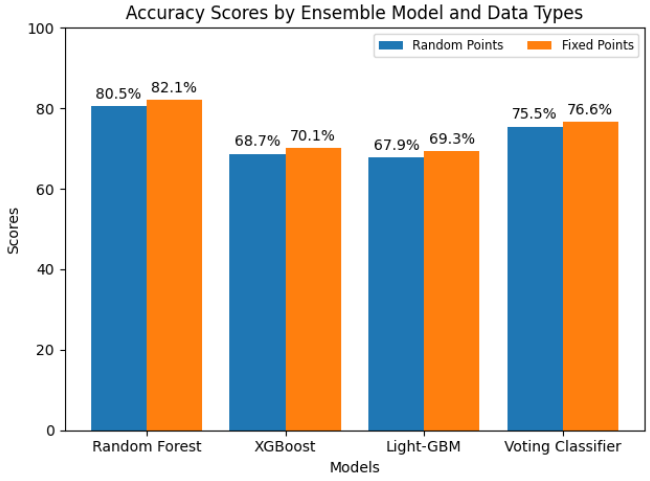


Figure 4.31: Machine Learning Models, with and without Random points

Differences between deep learning models, with random points and fixed points are shown in fig. 4.31.

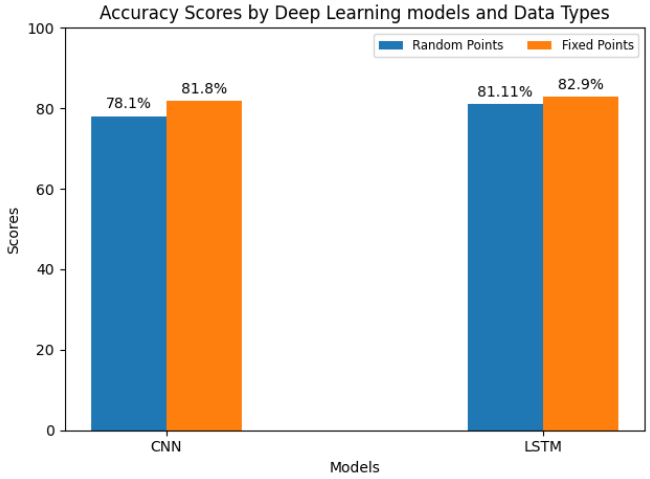


Figure 4.32: Deep Learning Models, with and without Random points

The comparison above reveals that changes in position significantly affect our system's performance. Our dataset included data from 9 specific points within a room arranged in a 3x3 grid, which allowed our models to better understand and classify gestures across different positions. As a result, we observed only a slight difference in performance between datasets with fixed points and mixed points. This finding underscores the importance of training models on diverse positional data to improve their ability to generalize and classify gestures accurately in varying environments.

4.6.5 Comparison With Existing Framework

For comparison, we have considered four papers. These papers are proposed by Kazuya Ohara et al.[1], Hasmath Farhana et al.[31], Yong Zhang et al.[33], and Marwa R. M. Bastwesy et al.[32]. Among these only Ohara et al. proposed a position-independent gesture recognition system. They obtained CSI data from a smartphone using a computer with Intel 5300 NIC. Also in their proposed system, users always need to carry a smartphone that transmits signals to the receiver machine. Others did not explicitly mention the classification accuracy concerning changes in the position of the actor.

The comparison between these four existing methods and our proposed method is shown in table 4.14.

Table 4.14: Overview of Existing Methods vs Proposed Method

Author	**Year**	**Tool**	**Method**	**Accuracy**
Ohara et al.[1]	2018	Intel 5300 NIC	HMM	67.8%
Hasmath et al.[31]	2021	Intel 5300 NIC	LSTM	78.0%
Yong et al.[33]	2022	802.11 based CSI tool	Meta-learning	79.5%
Marwa et al.[32]	2023	ESP32	Random Forest	72%
Proposed Work	-	ESP32	LSTM	81.11%

The overall comparison between the existing and proposed method is graphically represented in fig. 4.33.

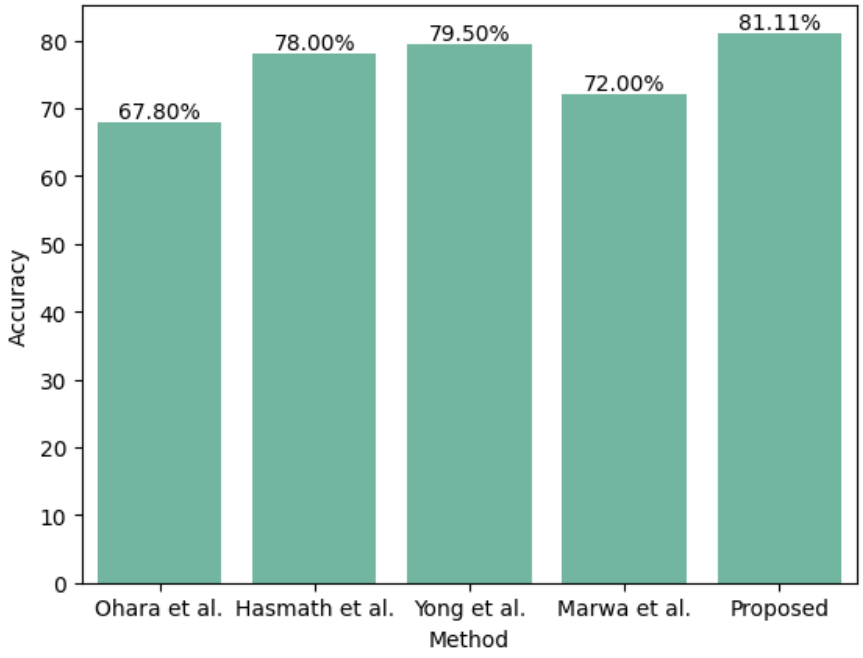


Figure 4.33: Performance Comparison between Existing vs Proposed Method

From the above comparison, we can say that our proposed method has shown significantly better performance in comparison with the existing methods.

4.6.6 Discussion

Based on our analysis, the LSTM (Long Short-Term Memory) model emerges as the most suitable choice for our gesture recognition system designed to operate smart home appliances. This conclusion is drawn from our work with sequential, time-series data, where each data point is closely linked to its preceding data points.

The LSTM model excels in this context because it effectively captures dependencies and patterns in sequential data. It is adept at understanding the sequence of events over time, which is crucial for interpreting gestures in our application. As a result, the LSTM model outperformed other approaches, such as tree-based ensemble models like Random Forest.

4.7 Conclusion

In this thesis, our main goal was to create a system that can recognize gestures within a smart environment using WiFi data. We began by collecting and preparing the WiFi data for analysis. Next, we tested different machine learning and deep learning techniques to find the most effective model for gesture classification. After comparing their performances, we found that the LSTM (Long Short-Term Memory) deep learning model worked best for accurately classifying gestures regardless of where they occurred in an indoor space. This outcome highlights the LSTM's ability to understand sequential patterns in WiFi data, making it ideal for our smart environment application. Overall, our research demonstrates how deep learning can enhance gesture recognition systems in indoor settings.

Chapter 5

Conclusion

5.1 Conclusion

In today's world, WiFi technology has made wireless connectivity widespread, making many tasks easier. Recently, using WiFi signals for sensing applications like gesture recognition has become more popular. Such systems are especially useful in smart environments like smart homes and offices. In this thesis, we proposed a WiFi IoT CSI-based position-independent gesture recognition system to operate smart home appliances. We developed this system utilizing a pair of ESP32 microcontrollers to collect CSI data. In the preprocessing phase, we eliminated null subcarriers and applied high-throughput filtering to refine the data. We then extracted crucial information about CSI amplitude and phase from this filtered data, which we used to train our models. This study comprehensively compares various machine learning and deep learning approaches, including different ensemble learning techniques, the shapelet learning method, and deep learning models like CNNs and LSTMs. It also includes a detailed analysis of the data utilized across these methodologies. Our findings show that the LSTM method achieved the highest accuracy at 81.11%, slightly surpassing the Random Forest's accuracy of 80.50%, demonstrating the effectiveness of long-term memory for this type of sequential data.

5.2 Limitations & Future Work

Our proposed gesture recognition system has some limitations:

- It was tested in a real-world environment, which might include some disturbances.

- The system isn't yet designed to work effectively in rooms with many people, which could affect its accuracy.

As interest in human-computer interaction grows, IoT-based gesture recognition systems are becoming increasingly popular. They are affordable and secure, making them an exciting area for further research. Future improvements could include:

- Using more ESP32 microcontrollers to collect data might provide a clearer picture.
- Gathering data with more people around could make the system more practical.
- Focusing more on the phase data and combined datasets could improve system performance.
- Applying advanced deep learning methods could enhance how well the system predicts gestures.

These steps will help overcome current challenges and expand the possibilities for gesture recognition technology.

References

[1] K. Ohara, T. Maekawa, S. Sigg and M. Youssef, 'Preliminary investigation of position independent gesture recognition using wi-fi csi,' in *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, IEEE, 2018, pp. 480–483 (cit. on pp. iv, 16, 77, 78).

[2] D. De, P. Bharti, S. K. Das and S. Chellappan, 'Multimodal wearable sensing for fine-grained activity recognition in healthcare,' *IEEE Internet Computing*, vol. 19, no. 5, pp. 26–35, 2015 (cit. on p. 1).

[3] H. Mliki, F. Bouhlel and M. Hammami, 'Human activity recognition from uav-captured video sequences,' *Pattern Recognition*, vol. 100, p. 107 140, 2020 (cit. on p. 1).

[4] J. Shotton *et al.*, 'Real-time human pose recognition in parts from single depth images,' in *CVPR 2011*, Ieee, 2011, pp. 1297–1304 (cit. on p. 1).

[5] H. Abdelnasser, M. Youssef and K. A. Harras, 'Wigest: A ubiquitous wifi-based gesture recognition system,' in *2015 IEEE conference on computer communications (INFOCOM)*, IEEE, 2015, pp. 1472–1480 (cit. on pp. 2, 14).

[6] H. F. T. Ahmed, H. Ahmad and C. Aravind, 'Device free human gesture recognition using wi-fi csi: A survey,' *Engineering Applications of Artificial Intelligence*, vol. 87, p. 103 281, 2020 (cit. on pp. 2, 14).

[7] S. Tan, Y. Ren, J. Yang and Y. Chen, 'Commodity wifi sensing in ten years: Status, challenges, and opportunities,' *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 17 832–17 843, 2022 (cit. on p. 10).

[8] Z. Wang, B. Guo, Z. Yu and X. Zhou, 'Wi-fi csi-based behavior recognition: From signals and actions to activities,' *IEEE Communications Magazine*, vol. 56, no. 5, pp. 109–115, 2018 (cit. on p. 10).

[9] N. Damodaran, E. Haruni, M. Kokhkharova and J. Schäfer, 'Device free human activity and fall recognition using wifi channel state information (csi),' *CCF Transactions on Pervasive Computing and Interaction*, vol. 2, pp. 1–17, 2020 (cit. on p. 10).

[10] Y. Zeng, D. Wu, R. Gao, T. Gu and D. Zhang, 'Fullbreathe: Full human respiration detection exploiting complementarity of csi phase and amplitude of wifi signals,' *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 3, pp. 1–19, 2018 (cit. on p. 10).

[11] B. Yu *et al.*, 'Wifi-sleep: Sleep stage monitoring using commodity wi-fi devices,' *IEEE internet of things journal*, vol. 8, no. 18, pp. 13 900–13 913, 2021 (cit. on p. 10).

[12] Z. Wang *et al.*, 'A survey of user authentication based on channel state information,' *Wireless Communications and Mobile Computing*, vol. 2021, pp. 1–16, 2021 (cit. on p. 10).

[13] Y. Ma, G. Zhou and S. Wang, 'Wifi sensing with channel state information: A survey,' *ACM Computing Surveys (CSUR)*, vol. 52, no. 3, pp. 1–36, 2019 (cit. on p. 10).

[14] Q. Pu, S. Gupta, S. Gollakota and S. Patel, 'Whole-home gesture recognition using wireless signals,' in *Proceedings of the 19th annual international conference on Mobile computing & networking*, 2013, pp. 27–38 (cit. on p. 10).

[15] J. Liu, Y. Wang, Y. Chen, J. Yang, X. Chen and J. Cheng, 'Tracking vital signs during sleep leveraging off-the-shelf wifi,' in *Proceedings of the 16th ACM international symposium on mobile ad hoc networking and computing*, 2015, pp. 267–276 (cit. on p. 10).

[16] Z. Wang *et al.*, 'A survey on human behavior recognition using channel state information,' *Ieee Access*, vol. 7, pp. 155 986–156 024, 2019 (cit. on p. 11).

[17] M. De Sanctis, E. Cianca, S. Di Domenico, D. Provenziani, G. Bianchi and M. Ruggieri, 'Wibecam: Device free human activity recognition through wifi beacon-enabled camera,' in *Proceedings of the 2nd workshop on Workshop on Physical Analytics*, 2015, pp. 7–12 (cit. on p. 11).

[18] Z. Yang, Z. Zhou and Y. Liu, 'From rssi to csi: Indoor localization via channel response,' *ACM Computing Surveys (CSUR)*, vol. 46, no. 2, pp. 1–32, 2013 (cit. on p. 11).

[19] L. Espressif Systems (Shanghai) Co., *Getting started with esp32*, `https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html`, 2016 (cit. on p. 12).

[20] S. M. Hernandez and E. Bulut, 'Lightweight and Standalone IoT Based WiFi Sensing for Active Repositioning and Mobility,' in *21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM) (WoWMoM 2020)*, Cork, Ireland, Jun. 2020 (cit. on p. 12).

[21] M. Oudah, A. Al-Naji and J. Chahl, 'Hand gesture recognition based on computer vision: A review of techniques,' *journal of Imaging*, vol. 6, no. 8, p. 73, 2020 (cit. on p. 13).

[22] C. Zhu and W. Sheng, 'Wearable sensor-based hand gesture and daily activity recognition for robot-assisted living,' *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 41, no. 3, pp. 569–573, 2011 (cit. on p. 13).

[23] F. Adib and D. Katabi, 'See through walls with wifi!' In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, 2013, pp. 75–86 (cit. on p. 13).

[24] L. Sun, S. Sen, D. Koutsonikolas and K.-H. Kim, 'Widraw: Enabling hands-free drawing in the air on commodity wifi devices,' in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, 2015, pp. 77–89 (cit. on p. 13).

[25] W. Xi *et al.*, 'Device-free human activity recognition using csi,' in *Proceedings of the 1st Workshop on Context Sensing and Activity Recognition*, 2015, pp. 31–36 (cit. on p. 14).

[26] Q. Zhou, J. Xing, J. Li and Q. Yang, 'A device-free number gesture recognition approach based on deep learning,' in *2016 12th International Conference on Computational Intelligence and Security (CIS)*, IEEE, 2016, pp. 57–63 (cit. on p. 14).

[27] Y. Ma, G. Zhou, S. Wang, H. Zhao and W. Jung, 'Signfi: Sign language recognition using wifi,' *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 1, pp. 1–21, 2018 (cit. on pp. 14, 15).

[28] Z. Tian, J. Wang, X. Yang and M. Zhou, 'Wicatch: A wi-fi based hand gesture recognition system,' *IEEE Access*, vol. 6, pp. 16 911–16 923, 2018 (cit. on p. 14).

[29] H. Li, W. Yang, J. Wang, Y. Xu and L. Huang, 'Wifinger: Talk to your smart devices with finger-grained gesture,' in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2016, pp. 250–261 (cit. on p. 15).

[30] J. Chen, X. Xu, T. Wang, G. Jeon and D. Camacho, 'An aiot framework with multi-modal frequency fusion for wifi-based coarse and fine activity recognition,' *IEEE Internet of Things Journal*, pp. 1–1, 2024. DOI: 10.1109/JIOT.2024.3400773 (cit. on p. 15).

[31] H. F. Thariq Ahmed, H. Ahmad, S. K. Phang, H. Harkat and K. Narasingamurthi, 'Wi-fi csi based human sign language recognition using lstm network,' in *2021 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, 2021, pp. 51–57. DOI: 10.1109/IAICT52856.2021.9532548 (cit. on pp. 15, 77, 78).

[32] M. R. M. Bastwesy, H. Choi and Y. Arakawa, 'Tracking on-desk gestures based on wi-fi csi on low-cost microcontroller,' in *2023 Fourteenth International Conference on Mobile Computing and Ubiquitous Network (ICMU)*, 2023, pp. 1–6. DOI: 10.23919/ICMU58504.2023.10412222 (cit. on pp. 15, 77, 78).

[33] Y. Zhang, X. Wang, Y. Wang and H. Chen, 'Human activity recognition across scenes and categories based on csi,' *IEEE Transactions on Mobile*

Computing, vol. 21, no. 7, pp. 2411–2420, 2022. DOI: 10.1109/TMC.2020.3041756 (cit. on pp. 16, 77, 78).

[34] J. Grabocka, N. Schilling, M. Wistuba and L. Schmidt-Thieme, 'Learning time-series shapelets,' in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 392–401 (cit. on p. 36).