Universidad Autónoma de Baja California
Instituto de Ingeniería
Maestría y Doctorado en Ciencias e Ingeniería

A System for Human Activity Recognition using Wi-Fi CSI on Embedded Devices

Tesis que para obtener el grado de:

DOCTOR EN CIENCIAS

Presenta

Jesús Albany Armenta García

Director de Tesis:

Dr. Félix Fernando González Navarro

Codirector de Tesis:

Dr. Jesús Caro Gutiérrez

Mexicali, B. C. Octubre 2025

Dedications

This thesis is dedicated to my parents, José and Tomy, and to my fiancée Monserrat, for their endless support and encouraging for always giving my best during these years of research.

Acknowledgements

I would like to acknowledge to all the professors that provide me the required knowledge for realizing this investigation, specially to Dr. Félix F. González-Navarro, who guided me throughout these years.

To *Secretaría de Ciencia, Humanidades, Tecnología e Innovación* (SECIHTI), and the *Instituto de Ingeniería - Universidad Autónoma de Baja California* (II-UABC) for their financial support and for providing me a space to do this research.

Finally, I would like to thank my postgraduate colleagues, whose support made my doctoral journey both pleasant and rewarding.

Abstract

This thesis addresses the limitations of current safety alert systems that rely on wearable sensors by proposing a novel Human Activity Recognition system based on Wi-Fi Channel State Information (CSI) and designed following the Edge Computing paradigm, where data processing occurs directly on the device responsible for data collection. The research is structured in three phases.

First, an Structured Literature Review (SLR) was conducted to assess state-of-the-art methods for Wi-Fi CSI collection and processing, revealing two critical gaps: the lack of a user-friendly, high-performance CSI collection tool and the absence of edge-based implementations for near real-time Wi-Fi sensing with low-cost devices.

To address these gaps, this work first introduces the ESP32 CSI Web Collecting Tool, a tool for collecting Wi-Fi CSI from ESP32 devices. Experimental results demonstrate that the proposed tool outperforms existing ESP32-based solutions, achieving a packet rate of up to 85 packets per second (compared to 30 packets per second in prior tools) when transmitting CSI data to a computer via a USB port. Additionally, the tool supports local data storage via an SD card and real-time transmission to external devices through GPIO pins, enhancing its versatility for edge applications.

Leveraging this tool, a lightweight Deep Learning model was optimized for deployment on the resource-constrained ESP32 microcontroller. By considering the hardware limitations of the implementation device, the model achieves an overall classification accuracy of 90.65% with an inference time of 232 ms, enabling near real-time functioning at the edge.

The contributions of this work include a high-performance, open-source CSI collection tool for the ESP32 platform and the first known implementation of an embedded system leveraging Deep Learning model for Wi-Fi HAR on an ESP32 device, which demonstrates the feasibility of near real-time, device-free safety monitoring.

Content

Figures

Tables

Chapter 1

Introduction

1.1 Problem Statement

According to the World Health Organization (2021), falls represent the second leading cause of global mortality from unintentional injuries, with adults over 60 years of age constituting the highest-risk group for fatal falls. Furthermore, an estimated 37.3 million falls that occur annually require medical attention. Therefore, prompt intervention following such incidents is critical to ensure the well-being of the affected people.

Current safety alert systems for fall detection typically employ mechanisms to notify others when a monitored person suffers a fall. These systems rely either on wearable sensors or vision-based technologies, such as surveillance cameras, which may intrude upon daily activities. Wearable sensors require consistent use by the person, presenting challenges when the device is forgotten or inconvenient for performing certain activities. Vision-based solutions, on the other hand, must have line-of-sight, adequate lighting, and infrastructure installation while also raising privacy concerns for the monitored person (Damodaran *et al.*, 2020).

To address these limitations, an alternative approach involves the use of Wi-Fi Channel State Information (CSI) as a sensing technology. Prior research has demonstrated that human activities induce measurable variations in CSI, which can be characterized and associated with specific activities through mathematical modeling, signal processing techniques, and artificial intelligence methods, giving rise to what is known as Wi-Fi sensing. However, since CSI is informa-

tion generated in the physical layer, direct collection is not supported. Most existing tools require firmware modifications to outdated network interface cards. Moreover, Deep Learning-based solutions rely on the use of cloud services or high-performance computers for processing data. These two situations hinder the scalability of Wi-Fi sensing applications.

To overcome these challenges, this work proposes a solution based on low-cost microcontrollers with integrated Wi-Fi that facilitate CSI collection. Furthermore, these devices are capable of executing Deep Learning models locally, enabling near real-time data processing without reliance on external computational resources or cloud services – an edge computing approach with embedded devices.

1.2 Objectives and Goals

1.2.1 General Objective

Design and develop an embedded system for Human Activity Recognition with Wi-Fi Channel State Information (CSI) and Artificial Intelligence methods, which processes data upon arrival, i.e., a near real-time system. This system will process CSI to detect and classify human activities from a predefined set with an accuracy comparable to existing state-of-the-art approaches, achieving recognition accuracies over 90%.

1.2.2 Specific Objectives

The following strategic objectives will be pursued to fulfill the general objective:

- To identify and select state-of-the-art methods, tools, and techniques for CSI-based Wi-Fi sensing applications through a structured literature review, suitable for its implementation in embedded devices.
- To define the system architecture by defining a data processing pipeline for Wi-Fi CSI-based Human Activity Recognition for embedded devices.
- To design, train and evaluate a Deep Learning model for Human Activity Recognition to be implemented in an embedded device.

- To design and develop an accurate embedded system based on microcontrollers capable of collecting and processing Wi-Fi CSI with near real-time functioning.

1.3 Thesis Outline

This thesis is organized as follows: Chapter 2 presents the state-of-the-art for CSI-based Wi-Fi sensing, which was written by following a Structured Literature Review to identify the applications being developed, the tools and techniques being used for collecting, processing, and performing recognition for Wi-Fi CSI, and to identify the deployment platforms. Chapter 3 provides an explanation of what Wi-Fi CSI is and an in-depth description of each of the stages involved in the functioning of Wi-Fi sensing systems. Then, Chapter 4 describes a proof-of-concept for achieving Human Activity Recognition with Deep Learning models with microcontrollers. This proof-of-concept was developed by following an adaptation proposed from the traditional CRISP-DM methodology. Chapter 5 introduces the ESP32 CSI Web Collecting Tool, a CSI collecting tool that facilitates the development of Wi-Fi sensing systems by achieving higher packet rates and providing tools for configuring it through a web form, eliminating the need to modify the source code or reprogram. Next, Chapter 6 presents the process of enhancing a Human Activity Recognition model through data augmentation using two different algorithms, and then implementing it into an embedded system to evaluate its performance. Finally, conclusions and future work are stated in Chapter 7.

Chapter 2

State-of-the-Art

In this chapter, a structured literature review is presented to provide an overview of the state-of-the-art Wi-Fi sensing applications. The review aims to identify the types of applications being developed, the steps involved in collecting and processing Wi-Fi CSI, and the devices being used in each of the selected works.

Performing a literature review consists of evaluating and interpreting all available research relevant to a particular research question, topic area, or phenomenon of interest (Kitchenham, 2004). Hence, in this section, a detailed description of how this review was conducted and the results obtained from it are provided, while the steps followed for elaborating this review is illustrated in Fig. 2.1.

2.1 Methodology for the Structured Literature Review

2.1.1 Research Questions

A literature review starts by formulating research questions that need to be addressed in the selected literature to be included in the review, as they will provide relevant information related to Wi-Fi CSI-based sensing and help detail the state-of-the-art of Wi-Fi sensing. These research questions are:

- *Which Wi-Fi sensing applications are being developed related to monitoring people?*
  Only works related to monitoring a person were considered for inclusion in this work.

- *Which techniques are used for preprocessing Wi-Fi CSI?*
  Preproocessing Wi-Fi CSI is mandatory for obtaining reliable sensing results. Thus, the answer to this question will help identify the most common and effective processing techniques for Wi-Fi sensing applications.

- *What devices are used for collecting and processing CSI?*
  Although Wi-Fi is ubiquitous, not all Wi-Fi devices report CSI to the final user. As the general objective of this work is the development of a Wi-Fi sensing application using embedded devices, it is important to discover which are the Wi-Fi devices that allow CSI collection and where its processing occurs, e.g., in the same collecting device, in the cloud, or on a computer.

- *What detection algorithms are used for Wi-Fi sensing?*
  Identify which Machine Learning, Deep Learning, and signal processing techniques are used for recognition or estimation tasks.

These questions were defined according to the stages involved in the identified Wi-Fi sensing methodology. Having a complete understanding of each of the stages involved in Wi-Fi sensing systems is essential for starting into its development. Each stage has its own tools, methods, and techniques. Knowing them will allow the definition of a pipeline for the functioning of Wi-Fi sensing systems.

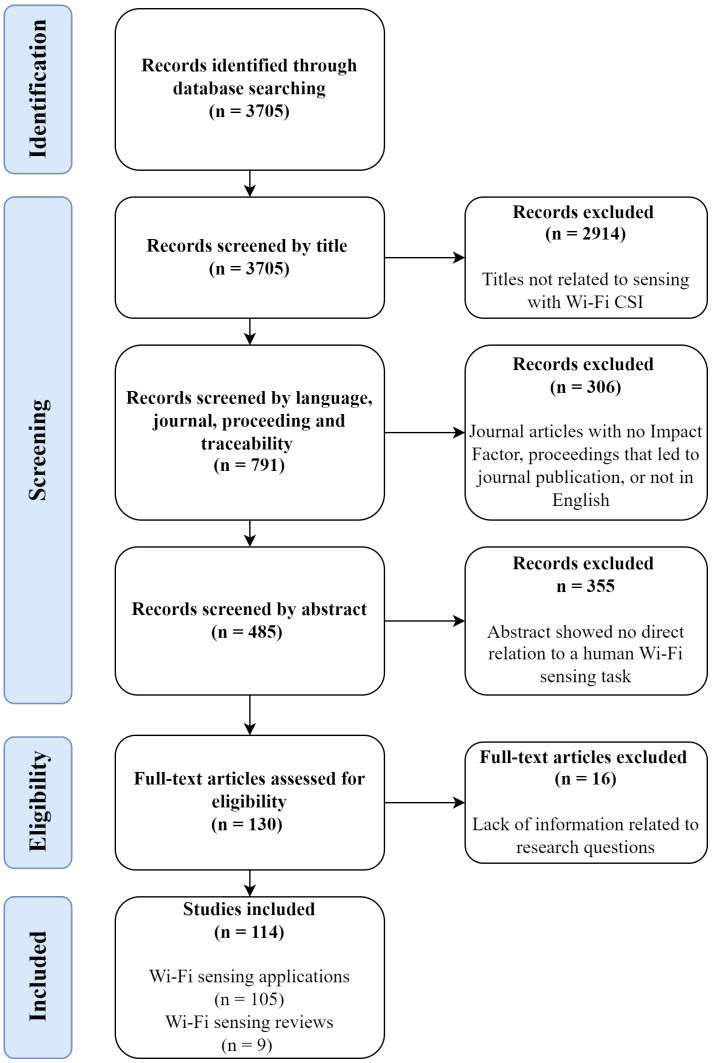

Figure 2.1: Followed steps for ellaborating the review.

2.1.2 Search for Works

The search for works was performed by a single reviewer and was limited to the SCOPUS database; however, it is a database recognized for listing and indexing publications from well-renowned journals and major conferences. This database also allows creating a *search string* to find articles, which were used to constrain the search to only works that might answer the established questions. Works containing words such as *Channel State Information* and *Wi-Fi*, and the words *sensing*, *application*, or *monitoring*, in the title, abstract, or as keywords, as well

as works that were published between 2016 and the first half of 2024, and that were written in English, were considered to be included. The search string used can be seen in Appendix A.1.

A total of 3705 works related to Wi-Fi sensing applications were found, where 48% of the works were published in the last two years, highlighting the interest of researchers in Wi-Fi as a sensing technology and the efforts to develop Wi-Fi sensing applications. A series of filters were implemented to reduce the number of papers to be reviewed. At first, works were discarded based on their titles, excluding those whose titles indicated no relation to human sensing or monitoring.

Once filtered by title, the remaining works were classified as either journal or magazine articles or conference proceedings, the latter of which included book chapters. Works with traceability were also excluded, i.e., works that were first presented as a conference proceeding and then extended as a journal article, considering only the extended versions for inclusion. Subsequently, journal articles were filtered, considering whether the journal has an Impact Factor. Both journal articles and conference proceedings were filtered according to their abstracts. This allowed the identification and keeping of only those works in which a Wi-Fi sensing application for monitoring people was developed. Finally, skimming was performed through the works, deciding whether or not to include a work in the review based on its content. This resulted in the inclusion of 105 works presenting Wi-Fi sensing systems, which will be analyzed and described in this chapter to provide an overview of the state-of-the-art for Wi-Fi sensing applications, and nine literature reviews with scopes that differ from the SLR presented in this work.

2.2 Other Wi-Fi Sensing Reviews

According to other Wi-Fi Sensing reviews, Wi-Fi sensing systems employ a methodology composed of four stages: CSI Data Collection, CSI Data Preprocessing, Use of Detection Algorithms, and Wi-Fi Sensing Application; each has its own background, and depending on the authors' interest, they focus their work on one or more of these stages. One example of a work that gives particular attention to the use of Wi-Fi CSI for multiple applications is the one presented by Ma

et al. (2019), which provided an overview of the functioning of Wi-Fi sensing systems, the preprocessing techniques applied to CSI data, the detection algorithms for acheving the sensing task, and the types of applications that could be given to such systems, exploring works involving Presence Detection, Fall Detection, Motion Detection, Activity Recognition, Gesture Recognition, and People Identification. Moreover, Al-qaness *et al.* (2019) stated that the stages of Wi-Fi sensing systems for Activity Recognition, Gesture Recognition, and Indoor Localization are CSI preprocessing, feature extraction, and classification. He *et al.* (2020) focused on discussing challenges and opportunities for Wi-Fi sensing. Tan *et al.* (2022) discussed the evolution of Wi-Fi sensing over a decade, specifically in Activity and Gesture Recognition and Indoor Localization, establishing that the development of these three types of applications has established milestones in Wi-Fi sensing and that there are still open challenges that need to be addressed. Furthermore, Soto *et al.* (2022a) focused on collecting and analyzing works related to Wi-Fi sensing for vital signs monitoring, presenting a guideline for the use of CSI for medical purposes. Similar to the latter, Ge *et al.* (2023) discussed works about healthcare monitoring systems using Wi-Fi CSI, expanding to systems for detecting Parkinson's disease or auxiliary systems such as fall detection systems, and even mentioning systems for human pose estimation through CSI.

Focusing on providing an overview of different detection algorithms that can be used for Wi-Fi sensing, Shao *et al.* (2022) give special attention to the detection stage of Wi-Fi sensing systems. The authors highlighted that there are different categories of detection methods, including model-based methods, data-based methods, and model-data hybrid-driven methods. Model-based methods build physical models and transform the CSI into a form that reflects environmental information for sensing. Data-based methods aim to uncover hidden attributes in the CSI data to facilitate prediction, identification, or classification using Machine Learning models. Model-data hybrid-driven methods preprocess the input that will be used by a Machine Learning model, resulting in a combination of the categories mentioned above. Similarly, Wang *et al.* (2019) focused on the recognition of human behavior, involving activity and gesture recognition, presence detection, and breathing rate monitoring. In this survey, the authors divided detection algorithms into three different categories: pattern-based applications, which use

pattern recognition methods such as Machine Learning algorithms; model-based applications, which recognize human behavior through mathematical or physical models that relate CSI variations and human behaviors; and deep-learning-based applications, which use Deep Learning models that do not require manual feature extraction. Finally, Hernandez & Bulut (2023) described CSI Data Preprocessing techniques for Wi-Fi sensing and how these can be implemented in embedded devices such as microcontrollers.

Table 2.1 presents a comparison between this review and others, based on the stated research questions. It can be seen that this review focused on including the most Wi-Fi sensing applications that can be found in the state-of-the-art and provides a comprehensive overview of every stage involved in Wi-Fi sensing systems: *CSI Data Collection, CSI Data Preprocessing, Detection Algorithms, and Wi-Fi Sensing Applications.*

Table 2.1: Comparative Table of Related Work

	Ma *et al.* (2019)	Wang *et al.* (2019)	Alqaness *et al.* (2019)	He *et al.* (2020)	Soto *et al.* (2022a)	Shao *et al.* (2022)	Ge *et al.* (2023)	Tan *et al.* (2022)	Hernandez & Bulut (2023)	Our work
Year	2019	2019	2019	2020	2022	2022	2022	2022	2023	2024
Describes available CSI collecting Tools	Yes	No	No	No	Yes	Partially	Yes	Yes	No	Yes
Describes the operational methodology of Wi-Fi sensing systems	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes	Yes
Sensing applications	AR, GR, PD, PI, VS, IL, and CC	AR, GR, VS, CC, PD, PI	IL, AR, and GR	VS, PI, AR, GR, GA, PD, and IL	VS	PD, AR, CC, IL, PI, and GR	VS, AR, GR, HP, and IL (Healthcare)	AR, GR, and IL	IL, AR, GR, VS, PD, and CC	AR, GR, VS, IL, CC, HP, GA, PD, and PI
Provides an insight on preprocessing techniques for Wi-Fi sensing	Yes	Yes	No	Yes	No	Yes	Yes	No	Yes	Yes
Proves an insight on detection algorithms for Wi-Fi sensing	Yes	Yes	Yes	Yes	No	Yes	Yes	No	No	Yes
Mention the CSI collecting tool used and antenna types in communication devices	No	No	No	No	No	No	Yes	No	No	Yes
Summarize the results obtained in each presented work	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes

Notes: AR-Activity Recognition, GR-Gesture Recognition, VS-Vital Signs Monitoring, IL-Indoor Localization, CC-Crowd Counting, HP-Human Pose Estimation, GA-Gait Analysis, PD-Presence Detection, PI-People Identification

2.3 Wi-Fi Sensing Applications using Channel State Information

Selected works were analyzed by first classifying them into a category depending on their application, i.e., what is being recognized or monitored. These categories were *Activity Recognition, Gesture Recognition, Vital Signs Monitoring, Indoor Localization, Crowd Counting, Human Pose Estimation, Gait Analysis, Presence Detection, and People Identification.* Once classified, relevant features, such as hardware used for collecting and processing, processing techniques, detection algorithms, and results, were analyzed and summarized from each work, allowing to highlight common practices for each category.

2.3.1 Activity Recognition

Activity Recognition applications mainly focus on recognizing human actions that involve changing positions or large-scale movements, such as moving multiple body parts from a predefined set of activities (Kim *et al.*, 2010).

Applications that fall into this category may achieve recognition with different detection algorithms such as statistical models, Machine Learning, and Deep Learning models.

Statistical-based Activity Recognition relies on the use of Hidden Markov Models, such as the system presented by Wang *et al.* (2017). This system was able to recognize between 11 daily activities, such as brushing teeth and opening the refrigerator, by extracting principal components from CSI for further processing with *Discrete Wavelet Transform* (DWT), obtaining an accuracy of 72% with this approach. Similarly, Zhao *et al.* (2021) used a Hidden Markov Model to recognize walk, sit down, push, swing, and wave activities, as well as to recognize if no activity is being performed. By processing CSI with a low-pass filter, extracting principal components, and finally processing with DWT, the authors obtained an accuracy of 97%. Similarly, Zhang *et al.* (2018) performed a statistical analysis on CSI collected for fall recognition with an accuracy of 95%.

Moving to Machine Learning, Li *et al.* (2016) used Random Forest to recognize the activities of walking, sit down, lie down, stand up, squat down, and crawl, obtaining an accuracy of 95.43% from features extracted from principal components

of CSI. Other Machine Learning algorithms, such as k-Nearest Neighbors (k-NN) and Support Vector Machine (SVM), were used by Arshad *et al.* (2017) with features extracted from CSI processed with a low-pass filter, achieving accuracies of over 89.2% in recognizing running, walking, and hand movement activities. It is important to note that this work also implemented an algorithm to identify the start and the end of an activity from CSI. Similar identification algorithms were implemented by Xiao *et al.* (2020); Zhu *et al.* (2022), which both extract principal components from CSI and apply a low-pass filter for noise reduction to then identify the start and ending points of an activity from the processed CSI. Natarajan *et al.* (2024) compared three different classifiers, Random Forest, Gradient Boosting, and Extreme Gradient Boosting, in a dataset containing four different activities, walk, sit down, stand up, and no activity, obtaining the highest cross-validation accuracy of 83.39% with Gradient Boosting algorithm. Moreover, TW-See is a system presented by Wu *et al.* (2019), which also processes CSI by applying a low-pass filter for then extracting principal components. Features are then extracted to use them as an input to a neural network that is able to recognize from among seven activities: walk, hand swing, fall, sit down, box, stand up, and no activity with an accuracy of 90.54% even when the person monitored and a concrete wall separates the Wi-Fi device.

Activity Recognition using Deep Learning models often involves the use of Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) architectures. For example, Nakamura *et al.* (2022) propose to extract spectrogram images from CSI. These images were later used as input to a ResNet model, which performs binary classification for fall and non-fall activities with an accuracy of over 92%. Zou *et al.* (2018b) used a CNN for generating discriminative features from CSI and an LSTM layer for interpreting latent temporal dependencies across time, obtaining a cross-validation accuracy of 97.6% for no activity, sit, walk, lie down, run, and stand activity recognition. An LSTM was also used by Damodaran *et al.* (2020) to compare if its performance with and without sophisticated preprocessing could be as good as with SVM with noise reduction, dimensionality reduction, and feature extraction, considering that with Deep Learning models, manual feature extraction is not performed. The authors found that in a LoS scenario, the LSTM outperforms the SVM. Moreover, Cui *et al.* (2021)

implemented a diversified deep ensemble learning approach. A CNN was used for extracting features from CSI, which were later used as an input for an ensemble classifier with a Random Forest, an SVM, and a Multiple-layer Perceptron. With this approach, the authors obtained an accuracy of 98.9% for identifying the activities of sit down, jump, wave, pick up, walk, and run. Showmik *et al.* (2023) explored the use of a Principal Component-based Wavelet CNN, i.e., a CNN that takes as input the second and third principal component extracted from CSI processed with a Savitzky-Golay filter, for later combining the approximation coefficients obtained by the DWT from processed CSI with the feature maps obtained from the convolutional layers, obtaining an accuracy of 95% for recognizing 16 different activities. With a more straightforward approach, Hernandez *et al.* (2022) used two different Dense Neural Network architectures, the first consisting of four dense layers for equipment-based exercise recognition, i.e., hand weights, total gym, curl bar, and weight bench, and the second consisting of only two dense layers for recognizing activities involving wrist, finger, and whole-body movements, obtaining an accuracy above 80.65% and 93%, respectively.

Aiming to improve performance by addressing problems not treated in the aforementioned works, Zhou *et al.* (2023a) considered that daily activities are performed sequentially, i.e., one after the other. Thus, they implemented a Temporal Convolutional Network (TCN), which exhibits a more extended memory compared to LSTM architectures, making it more suitable for daily activity monitoring and leading to accuracies above 89% for recognizing walk, hand movement, jump, box, sit down, and sit up activities. Another problem is the amount of data required to train accurate Machine Learning and Deep Learning models, which necessitates the implementation of data augmentation algorithms. Hence, Zhang *et al.* (2021) explored the creation of synthetic CSI data to improve activity recognition performance, obtaining an improvement of up to 34.6% for a dataset consisting of ten different activities, achieved by using an LSTM architecture. A problem stated and addressed by Zhang *et al.* (2023) is that model training is usually done only once, preventing the model from learning new activities. That is why they developed a Deep Learning model based on Incremental Learning, which consists of creating four scenes with different activities. The model is updated by implementing training iterations in which the scenes containing different

activities are introduced to the model, resulting in a model that, in its last iteration, can recognize up to 17 activities with an overall accuracy of 92.8% by also implementing data augmentation. Finally, Meneghello *et al.* (2023), Chen *et al.* (2024), and Lee *et al.* (2020) developed systems for activity recognition with accuracies over 96% in changing scenarios as Wi-Fi sensing applications might be deployed in scenarios different from where its model training took place.

The performance and activities detected in each presented work are summarized in Table 2.2.

Findings for Activity Recognition with Wi-Fi CSI

The presented works show that Wi-Fi CSI-based Activity Recognition is an emerging trend, with processing algorithms starting from statistical models, such as the Hidden Markov models, to Machine Learning-based, such as k-NN and SVMs. Recent approaches tend to use Deep Learning approaches, particularly CNNs and LSTMs, reporting accuracies exceeding 90% for recognizing between different activities. Another finding is that there is an interest for using network devices with multiple antennas, as this provides spatial diversity, allowing to capture more detailed information from CSI for distinguishing between activities.

2.3 Wi-Fi Sensing Applications using Channel State Information

Table 2.2: Activity Recognition Applications

Reference	Activities	Equipment	Detection	Performance
Damodaran *et al.* (2020)	Walk, run, sit, stand, and empty/no activity.	Tx-1 antenna, Rx-3 antennas	LSTM network	Accuracies $\geq$ 95%.
Nakamura *et al.* (2022)	Falling	Antenna Array	CNN	90% accuracy
Zhou *et al.* (2023a)	Walk, hand movements, jump, box, sit down, and sit up	Tx-1 OMNI antenna, Rx-3 OMNI antennas	CNN	Accuracies $\geq$ 89%.
Arshad *et al.* (2017)	Running, walking, and hand moving	Tx-3 OMNI antennas, Rx-1 OMNI antenna	k-NN and SVM	Accuracies $\geq$ 89.2%.
Wu *et al.* (2019)	No activity, walk, hand swing, fall, sit down, boxing, and stand up	Tx-1 OMNI antenna, Rx-3 OMNI antennas	Dense network	90.54% accuracy.
Cui *et al.* (2021)	Sit down, jump, wave, pick up, walk, and run	Tx-1 OMNI antenna, Rx-3 OMNI antennas	CNN and ensemble classifier	98.9% accuracy.
Xiao *et al.* (2020)	Dumbbell lifting, deep squat, kicking, and boxing	Tx-3 OMNI antennas, Rx-1 OMNI antenna	Dynamic Time Warping	97.8% accuracy.
Zhu *et al.* (2022)	3 literature activity datasets	Tx-2 OMNI antennas, Rx-3 OMNI antennas	CNN	Accuracies $\geq$ 97.5%.
Lee *et al.* (2020)	No activity, walk, cook, eat, bathe, use toilet, sleep, and fall	Not specified	CNN-LSTM network	Accuracies $\geq$ 96%.
Zhao *et al.* (2021)	No activity, walk, sit down, push, swing, and wave	Tx-2 OMNI antennas, 2 Rx-3 OMNI antennas	Hidden Markov Model	92.9% accuracy.
Chen *et al.* (2024)	Fall, walk, run, sit down, stand up, jump, wave hand, and stoop	not specified	CNN	97.57% accuracy.
Wang *et al.* (2017)	Run, walk, sit down, open refrigerator, fall, box, push one hand, brush teeth, and no activity	Not specified	Hidden Markov Model	72% accuracy.
Zhang *et al.* (2023)	17 activities, such as stretch, jump, and sit down	Not specified	Dense network	92.8% accuracy.
Natarajan *et al.* (2024)	Empty, walk, sit down, and stand up	Single onboard antenna	Gradient boosting classifier	83.39% cross-validation accuracy.
Hernandez *et al.* (2022)	Physical therapy activities and use of gym equipment	Single onboard antenna, up to 3 ESP32 as Rx	Dense network	Accuracy > 80%.
Showmik *et al.* (2023)	WiAr dataset	Tx-1 OMNI antenna, Rx-3 OMNI antennas	CNN	95% accuracy.
Meneghello *et al.* (2023)	Empty, walk, run, sit down, and jump	Tx-1 antenna, Rx-4 antennas, type not specified	CNN	96% accuracy in changing environments.
Zhang *et al.* (2021)	Phone call, check wristwathc, walk, walk fast, run, jump, lie down, play guitar, play piano, and play basketball	Dense-LSTM network	Tx and Rx-3 OMNI antennas	89% accuracy.
Li *et al.* (2016)	Walk, sit down, lie, stand up, squat down, fall, and crawl	Tx-2 OMNI antennas, Rx-3 OMNI antennas	Random Forest	95.43% accuracy for LoS scenario, and 91.2% for NLoS scenario.
Zou *et al.* (2018b)	No activity, sit, walk, lie, run, and stand	Transceivers with 3 OMNI antennas	Dense network	97.6% cross-validation accuracy.
Zhang *et al.* (2018)	Falling	Tx-2 OMNI antennas, Rx-3 OMNI antennas	Mathematical model	95% accuracy.

2.3.2 Gesture Recognition

Compared to Activity Recognition, Gesture Recognition can be defined as the task of recognizing small-scale movements such as finger, hand, and mouth movements, which are often referred to as gestures.

Hand gesture recognition applications are common in Gesture Recognition applications. For example, the system developed by Li *et al.* (2016). This system processes CSI by applying a Hampel filter, a low-pass filter, and a moving average filter for noise reduction and then extracts features to be used by the k-NN algorithm, which utilizes Dynamic Time Warping as a distance metric, achieving an accuracy of 90.4% in recognizing nine different hand gestures from ten volunteers. Similarly, Tian *et al.* (2018) used an SVM to recognize six one-hand gestures and three two-hand gestures, achieving accuracies above 96% by exploiting phase information in CSI to construct a virtual antenna array and reconstruct hand movement trajectories, thereby enhancing the system's performance. Following Deep Learning approaches, Tong *et al.* (2023) processed the CSI phase by calculating the phase difference between multiple antennas and applied noise reduction techniques such as low-pass filter and outlier removal for recognizing eight hand gestures from the Widar dataset (Yang *et al.*, 2020) by using a CNN with Attention Modules, commonly used in Natural Language Processing, with a mean accuracy of 92%. An Attention-based Bi-directional Gate Recurrent Unit was used by Meng *et al.* (2022), as well as the phase difference method, for recognizing four hand gestures using the CSI phase under three different scenarios. The first scenario consisted of the person performing the gestures in a room with a few static objects. The second scenario introduced more static objects to the room, and the third scenario introduced a dynamic object, which was a person walking, achieving accuracies of 96%, 95.5%, and 60%, respectively. Similarly, Gu *et al.* (2023) and Yang *et al.* (2023a) used the Widar dataset to experiment with Natural Language Processing Deep Learning model architectures, such as Attention Modules and Transformers, obtaining accuracies of 93.71% and 86.16%, respectively.

Other Deep Learning architectures, such as CNNs, have been used for gesture recognition. For example, Jia *et al.* (2020) used heat map images constructed

from CSI processed with a low-pass filter as input to a CNN to identify what a user was doing on a computer, based on their keyboard and mouse movements, with an average accuracy of 77%. Furthermore, Bu *et al.* (2022) developed a system that also uses CNNs for recognizing 12 different gestures with a cross-validation accuracy over 95% by using a CNN, which uses a CSI image processed with a low-pass filter and singular value decomposition. The image is generated by the superposition of CSI amplitude from antenna pairs. Zou *et al.* (2018)

Apart from hand gesture recognition, Niu *et al.* (2018) explored the use of a virtual multipath to enhance the sensing capabilities of the system to be able to extract fine-grained finger and chin movements from CSI to recognize finger gesture patterns by using a neural network and to count the number of syllables from words based on a peak finding algorithm respectively. With this approach, the authors reported an accuracy of 81% for recognizing between 8 finger gestures and an average counting accuracy of 92.8% for chin movements. Moreover, Chen *et al.* (2022) explored removing multipath reflections based on their delay through the Power Delay Profile (PDP) for recognizing six facial expressions: happiness, fear, surprise, happily surprised, and angrily surprised, with a tree-based Machine Learning algorithm, achieving a cross-validation accuracy of 94.8%.

Practical Gesture Recognition applications have also been developed. For example, Ali *et al.* (2017) developed Wikey, an application that recognizes which computer keyboard keystroke was pressed, as well as complete words from sentences, with an accuracy above 85%. This was achieved by extracting principal components for CSI, applying a low-pass filter and DWT, and then using k-NN with Dynamic Time Warping measure as distance metric. Furthermore, Niu *et al.* (2019) developed WiMorse, a system for recognizing letters according to their Morse code representation by forming dots and dashes with specific finger movements by measuring similarity between extracted signals from reference signals with Dynamic Time Warping, achieving an accuracy of 96.6% for Morse code letter recognition. Wu *et al.* (2020) developed a system for finger tracking and recognizing digits being drawn in the air by removing random CSI phase offset by proposing a CSI-quotient model by using two antennas at the receiver, which allows the reconstruction of the trajectory of finger drawings in the air. With this approach, the authors obtained an accuracy of over 90% for digit recognition with

Microsoft Azure OCR service. Similarly, Fu *et al.* (2019) extracted features from CSI principal components used for recognizing in real-time letters contained in a phrase by using Hidden Markov models with an accuracy of 74.84% and being able to recognize up to 12.5 letters per minute. Moreover, Hao *et al.* (2020) used K-Means and Bagging optimized SVM to recognize 12 gestures associated with sign language actions by using features extracted from principal components of CSI amplitude and phase difference processed with a low-pass filter and linear transformation to CSI phase, achieving an accuracy of 95.8%. Finally, Hernandez *et al.* (2022) used a Dense Neural Network for gesture recognition during therapy, specifically for finger and wrist movements. It is important to highlight that the authors used ESP32 microcontrollers for collecting CSI and obtained an accuracy of 85.21% and 97.13% for finger and wrist gesture recognition, respectively. The performance metrics and gesture sets of the reviewed works are compared in Table 2.3.

Findings for Gesture Recognition with Wi-Fi CSI

Collectively, these works confirm the viability of Wi-Fi CSI as a sensing technology for detecting small-scale movements. However, signal preprocessing is necessary to extract relevant features from the fine-grained distortions caused by gestures. Similar to Wi-Fi CSI for Activity Recognition, Deep Learning models are preferred over other classification methods, including architectures based on CNN with attention mechanisms, Transformers, and Recurrent units. The practical feasibility of this technology is demonstrated by its application in complex tasks such as sign language translation, keystroke recognition, and assistive technologies.

2.3 Wi-Fi Sensing Applications using Channel State Information

Table 2.3: Gesture Recognition Applications

Reference	Gestures	Equipment	Detection	Performance
Hao *et al.* (2020)	Sign language	Tx-2 OMNI antennas, Rx-3 OMNI antennas	SVM	95.8% accuracy.
Tian *et al.* (2018)	1 hand and 2 hands gestures	Tx-1 OMNI antenna, Rx-2 OMNI antennas	SVM	Accuracies $> 96\%$.
Tong *et al.* (2023)	WiDar 3.0 dataset	Tx-1 OMNI antenna, Rx-3 OMNI antennas	CNN with GRU network	Accuracies $\geq 81\%$.
Meng *et al.* (2022)	4 different hand gestures	Tx and Rx-3 OMNI antennas	Recurrent network	96% accuracy.
Zou *et al.* (2018)	6 hand gestures	Tx- 1 OMNI antenna, Rx-3 OMNI antennas	CNN	98% cross-validation accuracy.
Yang *et al.* (2023a)	WiDar 3.0 dataset	Tx-1 OMNI antenna, Rx-3 OMNI antennas	Transformer-based network	86.16% accuracy.
Chen *et al.* (2022)	6 different facial expressions	Tx-1 directional and 2 OMNI antennas, Rx-3 OMNI antennas	Tree-based model	94.8% cross-validation accuracy.
Bu *et al.* (2022)	12 different hand and finger gestures	Tx-1 OMNI antenna, Rx-3 OMNI antennas	CNN	95% cross-validation accuracy.
Jia *et al.* (2020)	Computer-related gestures	Tx-1 OMNI antenna, Rx-3 OMNI antennas	CNN and SVM	77% accuracy.
Li *et al.* (2016)	9 finger gestures	Tx-1 OMNI antenna, Rx-3 OMNI antennas	k-NN	90.4% accuracy.
Ali *et al.* (2017)	Keystrokes	Tx-2 antennas, Rx-3 antennas, type not specified	k-NN	97.5% accuracy for keystroke detection; 96.4% for keystroke recognition.
Hernandez *et al.* (2022)	Physical therapy gestures	Single onboard antenna, up to 3 ESP32 as Rx	Dense network	Accuracies $\geq 85.21\%$.
Niu *et al.* (2018)	8 different finger gestures and chin movement detection	Tx and Rx- 1 OMNI antenna	Dense network	81% accuracy for finger gestures; 92.8% accuracy for chin movement counting.
Niu *et al.* (2019)	Finger movements for morse letter recognition	Tx-3 OMNI antennas, Rx-2 OMNI antennas	Dynamic Time Warping	96.6% accuracy.
Wu *et al.* (2020)	Finger Tracking and digit recognition	Tx-1 OMNI antenna, Rx-3 OMNI antennas	Cloud OCR service	Median error of 1.27cm for finger tracking; 90% accuracy for digit recognition.
Fu *et al.* (2019)	Finger tracking for letter recognition	Tx-2 OMNI antennas, Rx-3 OMNI antennas	Hidden Markov Model	74.84% accuracy.
Gu *et al.* (2023)	WiDar 3.0 dataset	Tx-1 OMNI antenna, Rx-3 OMNI antennas	CNN with attention modules	93.71% cross-validation accuracy.

2.3.3 Vital Signs Monitoring

Efforts to monitor physiological data using Wi-Fi technology have been made by developing Wi-Fi sensing applications for breathing monitoring, precisely estimating its value, or identifying breathing patterns using machine/Deep Learning or signal processing techniques.

Comparing the chest movement caused by breathing with activities or gestures, such as walking or finger tracking, reveals that chest movements are of a finer scale, involving millimetric movements. This situation led to Wang *et al.* (2016) to explore how the breathing depth of a person, its location relative to Rx and Tx, and its orientation can affect breathing detectability through Wi-Fi sensing. The authors found that all these parameters influence breathing detectability and that breathing cannot be detected when the person is beyond 2 meters from the Rx-Tx line-of-sight (LoS). More recent work by Mosleh *et al.* (2022) highlights how the distance between Rx and Tx equipped with multiple antennas influences the performance of breathing pattern classification using an LSTM network in an anechoic chamber. Path loss was artificially added to the data to measure the effect of distance between transceivers, finding that cross-validation accuracy drops 25% when the distance between devices increases from 2.3 to 72.76 meters.

As mentioned previously, breathing rate monitoring is achieved through the use of artificial intelligence or signal processing techniques. Two common approaches for breathing monitoring using signal processing techniques involve performing frequency analysis using the Fast Fourier Transform (FFT) or employing peak detection and counting algorithms. Liu *et al.* (2016) applied the FFT to CSI amplitude processed with Hampel filter and Wavelet transform to perform a spectral analysis for estimating the breathing rate of a person lying in bed, as well as for identifying his posture with accuracies over 85%. Wiphone, a system developed by Liu *et al.* (2021), also uses the FFT for estimating breathing rate, with an estimation error of 0.31 breaths per minute by using smartphones as Wi-Fi devices. Likewise, Yin *et al.* (2022) used smartphones as Wi-Fi devices for CSI collection and the FFT for spectral analysis of the five most sensitive subcarriers, selected based on a proposed breathing-to-subcomponent ratio. The breathing

rate value was defined by the most repeated frequency in the five selected subcarriers, resulting in an average detection error of 0.34 breaths per minute. Due to the 2020 pandemic, Atif *et al.* (2022) developed a breathing rate monitor using CSI amplitude collected with ESP32 microcontrollers and processed with a band-pass filter, which also performs spectral analysis to obtain the breathing rate value with a reported accuracy of 99.6%. A similar approach was followed by Alzaabi *et al.* (2024), which differs from the previous work due to the processing applied to the CSI amplitude. In this work, the authors applied interpolation and downsampling to the CSI amplitude to address an issue that CSI collection with ESP32 microcontrollers has: unstable packet rates. This was followed by the application of the DWT, a Hampel filter for outlier removal, and mean and median filters for smoothing the data. Authors found that with this approach, 95% of estimation errors range between -0.27 and 0.21 breaths per minute, with an RMSE of 1.04.

Peak detection and counting algorithms have the advantage of not depending on the FFT window size for frequency resolution, making them more appropriate for real-time applications. Zhang *et al.* (2017) used a peak detection and counting algorithm for breathing rate estimation. As CSI represents the channel frequency response (CFR) in the frequency domain, the authors converted it to the time domain, obtaining the channel impulse response (CIR) with the inverse FFT. This allowed them to select the three subcarriers where chest movement caused by breathing has more influence. With this approach, the authors found that breathing rate estimation can achieve accuracies over 90% even when the person being monitored is performing a wide range of *micro motions*. Wang *et al.* (2020c) experimented with breathing rate estimation with a peak detection algorithm in a through-the-wall scenario, selecting relevant subcarriers based on a signal energy and movement detection algorithm, which led them to achieve a maximum median error of 1.75 breaths per minute, while in LoS scenarios, median errors range around 0.25 breaths per minute. Similarly, Gui *et al.* (2023) employed a subcarrier method based on the relative difference between the two highest frequency components; the greater the difference, the more breathing rate information the subcarrier contains. With this approach, 80% of estimation errors obtained were below 0.38 breaths per minute. Furthermore, Dou & Huan (2021)

estimated breathing based on the peak-to-peak interval from the accumulated Doppler spectral energy, obtaining maximum estimation errors below 0.7 breaths per minute. Doppler shift extraction from CSI requires phase sanitization, which is achieved by fitting the phase error and subtracting it from the collected phase. Liu *et al.* (2018) used a peak-to-peak interval for breathing rate estimation on subcarriers selected based on a variance-based subcarrier selection algorithm after processing CSI amplitude with Hampel and band-pass filters. Additionally, the sleep posture was identified using Random Forest with an accuracy of 98%.

As for artificial intelligence approaches, Zeng *et al.* (2020) explored estimating the breathing rate of multiple persons by using the K-Means clustering algorithm and segment matching on CSI amplitude of subcarriers selected by a proposed selection algorithm based on Respiration Energy Ratio, obtaining mean absolute breathing rate errors of 0.21, 0.42, and 0.73 breaths per minute in scenarios with two, three, and four persons, respectively. Aside from breathing rate estimation, Xiaolong *et al.* (2021) extracted features from CSI amplitude and phase to construct an observation used as input to a neural network, which classified it as normal breathing pattern, apnea state, or no breathing with accuracies of 96.12%, 98.85%, and 99.33% respectively. Chen *et al.* (2018) first used an SVM to detect the presence of a person in a room and, once detected, estimate his breathing rate with a Root-MUSIC algorithm and cluster analysis for scenarios with more than one person, achieving an accuracy of 98.65% for breathing rate estimation for up to 9 people in a LoS scenario.

Heart rate is another vital sign that has received attention in Wi-Fi sensing applications. Applications focused on monitoring this vital sign aim to directly estimate it or perform pattern identification, highlighting that it is more challenging than monitoring breathing rate, as chest displacements caused by a heartbeat are considerably smaller. An example of this is the work presented by Liu *et al.* (2018). Aside from performing breathing rate estimation and sleep posture identification, they estimated heart rate through Power Spectral Density analysis, with errors of less than four beats per minute. Lee *et al.* (2018) identified both breathing and heart rate patterns using Dynamic Time Warping as a measurement of similarity between collected CSI processed with a band-pass filter and multipath

removal based on Power Delay Profile, with reference signals, resulting in accuracies of 94% and 90% for breathing and heart rate pattern identification, respectively. Similarly, Tsubota *et al.* (2023) applied wavelet transform and variational mode decomposition for signal decomposition to extract heart rate information from the CSI phase. This was performed to calculate the difference between two consecutive heartbeats for measuring heart rate variability, with a mean RMSE of 143.9 ms. This analysis also helped them conclude that performance can be improved by increasing the transmission power and using directional antennas. Wang *et al.* (2020b) support this statement as they explored the use of directional and omnidirectional antennas for heart rate estimation based on spectral analysis with FFT. During experimentation, the authors found that using omnidirectional antennas resulted in an error of 4.82 beats per minute in heart rate estimation. In contrast, directional antennas reduced the error to 1.19 beats per minute.

The performance of each presented work for Vital Signs Monitoring is summarized in Table 2.4.

Findings for Vital Signs Monitoring with Wi-Fi CSI

The findings indicate that accurate breathing rate estimation, with errors frequently below 0.5 breaths per minute, is achievable through different methods, ranging from foundational signal processing techniques to Machine and Deep Learning models. In contrast to activity and gesture recognition, these types of applications exhibit a reliance on frequency-domain analysis, e.g., FFT and PSD, and peak detection algorithms for precise estimation. A conclusion from the review of these works is that system performance, for both breathing and the more challenging heart rate monitoring, is highly susceptible to degradation from environmental factors, including distance, path loss, and NLoS conditions. Consequently, these constraints represent fundamental design considerations for developing robust Vital Signs Monitoring systems.

2.3 Wi-Fi Sensing Applications using Channel State Information

Table 2.4: Vital Signs Monitoring Applications

Reference	Vital Signs	Equipment	Detection	Performance
Liu *et al.* (2018)	Breathing and heart rate and sleep postures	Not specified	k-NN, SVM and Random Forest	Breathing rate errors < 0.4 bpm; 98% accuracy for posture identification; heart rate errors < 4 beats.
Wang *et al.* (2020b)	Breathing and heart rate	Tx with directional antenna	FFT analysis	Median estimation errors of 0.25 breaths per minute and of 1.19 beats per minute.
Zhang *et al.* (2017)	Breathing rate	Tx and Rx-3 OMNI antennas	Peak detection	Accuracies $\geq$ 90%.
Chen *et al.* (2018)	Breathing rate	Tx and Rx-3 OMNI antennas	SVM and MUSIC	98.65% multi-person breathing accuracy.
Wang *et al.* (2020c)	Breathing rate	Tx and Rx-3 OMNI antennas	Peak detection	Median errors of 0.25 bpm, 0.25 bpm and 0.29bpm for three different scenarios.
Xiaolong *et al.* (2021)	Breathing pattern	Tx-1 OMNI antenna, Rx-3 OMNI antennas	Dense network	Accuracies $\geq$ 96.12%.
Lee *et al.* (2018)	Breathing and heart pattern	Not specified	Dynamic Time Warping	Breathing pattern identificacion accuracy of 94% and 90% accuracy for heart pattern identification.
Zeng *et al.* (2020)	Breathing rate	Tx and Rx- 3 OMNI antennas	k-means	Breathing rate error of 0.73 bpm in a 4 person scenario.
Mosleh *et al.* (2022)	Breathing pattern	Tx-2 OMNI antennas, Rx-3 OMNI antennas	LSTM network	99.97% cross-validation accuracy.
Gui *et al.* (2023)	Breathing rate and sleep postures	Tx-1 OMNI antenna, Rx-3 OMNI antennas	Peak detection	Estimation errors < 0.38; 94.59% accuracy for posture identification.
Yin *et al.* (2022)	Breathing rate	Tx-4 OMNI antennas, Rx-smartphone	FFT analysis	Mean detection error of 0.34 bpm.
Liu *et al.* (2016)	Breathing rate and sleep postures	2 Tx and 2 Rx	FFT analysis	Breathing rate and posture identification accuracy > 85%.
Alzaabi *et al.* (2024)	Breathing rate	Single onboard antenna	PSD analysis	RMSE of 1.04 bpm.
Dou & Huan (2021)	Breathing rate	Tx and Rx-1 OMNI antenna	Peak detection	Breathing rate errors $\leq$ 0.7 bpm.
Tsubota *et al.* (2023)	Heart rate variability	Tx-1 OMNI antenna, Rx-3 OMNI antennas	Peak detection	RMSE of 143.9 ms.
Niu *et al.* (2018)	Breathing rate	Tx and Rx-1 OMNI antenna	FFT analysis	98.8% accuracy.
Liu *et al.* (2021)	Breathing rate	Not specified	FFT analysis	Breathing rate error of 0.31 breaths per minute (bpm).
Atif *et al.* (2022)	Breathing rate	1 ESP as Tx, 1 ESP as Rx; experiments done with onboard antenna, OMNI antennas, and directional antennas.	PSD analysis	99.6% accuracy.

2.3.4 Indoor Localization

Indoor Localization applications based on Wi-Fi CSI can detect a person's location inside a room without requiring the person to carry a device.

In the state-of-the-art, this type of application can be classified into two categories according to the approach followed: fingerprint-based or subspace-based. Fingerprint-based applications involve constructing a database with CSI samples that contain information about a person at each location of interest (reference point) within a room. These samples are then used to classify new samples based on fingerprints. In contrast, subspace-based applications determine the person's location by extracting and analyzing signal propagation features that can be estimated from CSI.

An example of a fingerprint-based Indoor Localization application is presented by Chen *et al.* (2017), where they delimited 64 reference points in an indoor scenario, collecting 120,000 CSI samples of a person located at each reference point. These samples were then converted into images to train a CNN for locating a person at one of the 64 points, with a mean estimation error of 1.36 meters, considering the scenario had dimensions of 16.3 meters by 17.3 meters. Similarly, Wang *et al.* (2021) developed ResLoc. This system also constructs images from CSI data collected in three different scenarios, using 15 to 20 reference points with a maximum distance error of 2.46 meters. Notably, 30% of the test samples used to evaluate a Deep Learning model resulted in an error of less than 0.3 meters. Moreover, Zhao *et al.* (2019) used a Bayesian adaptive Kalman filter to improve the classification performance of a CNN by up to 22% in a real indoor scenario, obtaining a mean distance error of 0.46 meters in a LoS scenario and of 1.11 meters in an NLoS scenario. In both scenarios, 42 reference points were defined, with a distance of 1.2 meters between adjacent points.

Addressing Indoor Localization as a regression problem, Zhou *et al.* (2018) used a regression-based deep neural network for predicting the indoor location of a person under two different scenarios, the first containing 40 reference points used to train the model and 12 testing points for evaluating it, while the second scenario had 62 reference points and 32 testing points. With this approach, the authors obtained a mean distance error of 1.08 meters under the first scenario

and 1.50 meters under the second scenario. Likewise, Kim *et al.* (2021) collected samples from 113 reference points used to train a variational Deep Learning model for predicting the position of a person through regression. To evaluate the model, the authors used testing samples with information from 22 test points that were not used as a reference for training, resulting in a localization accuracy of 1.28 meters, where both the reference and test points were spaced 0.6 meters apart.

By using Machine Learning models, Wang *et al.* (2018b) used Random Forest for Indoor Localization of 25 reference points, achieving an error of 0.17 meters. Similarly, Shi *et al.* (2018) used Bayesian methods to estimate the location of a person by dividing four different scenarios into multiple cells of the same size and tracking the person through them, achieving an accuracy over 90% for cell identification and a mean absolute error of 0.63 meters. Wu *et al.* (2018) achieved Indoor Localization under two different scenarios. The first scenario contained 19 reference points, while the second consisted of 30. Additionally, two other more challenging scenarios were considered. The first was a corridor with people passing through, and the second was a room connected to an outdoor area containing reference points. The Naive Bayes classifier was the Machine Learning algorithm selected to determine the person's location, achieving accuracies above 90% in the first two scenarios. In contrast, accuracies of 66.7% and 94.7% were obtained for the last two, respectively. Furthermore, Zhou *et al.* (2021) estimated the location of a person in an indoor scenario where 90.47% of the cases present errors below 4 meters by using a neural network in conjunction with an adaptive genetic algorithm to avoid the convergence of the neural network to a local optimum and using smartphones as CSI collection devices.

Moving on to subspace-based Indoor Localization, Li *et al.* (2016) proposed the use of a Dynamic-MUSIC method to detect reflection paths that capture the movement of a person and his position is determined by comparing the estimated angle of arrival from multiple receivers with angles previously registered for different positions in the room. Using four receivers, the authors obtained a median error of 41 cm with the proposed method. A 2-dimensional multiple packets-based matrix pencil (2D M-MP) was proposed by Yang *et al.* (2022). This matrix was used to estimate the angle of arrival and time of flight by accumulating multiple

CSI frames to allow real-time localization by reducing execution time if compared to other estimation algorithms such as MUSIC, considering that 2D M-MP presents an execution time of 0.074 seconds and MUSIC takes up 5 seconds, while localization error is of 0.42 meters. In addition, these same authors proposed an extension of their previous work, presenting a 3-D beamspace matrix pencil algorithm to estimate the angle of arrival, time of flight, and Doppler frequency shift, as well as a phase detection and compensation algorithm to ensure the stability of phase differences. This proposal reduces execution time and the median error for indoor localization from 0.48 meters with M-MP to 0.43 meters (Yang *et al.*, 2023b). Wang *et al.* (2018a) took a different approach compared to the previously described. They proposed using genetic algorithms to solve equations from a power fading model to achieve Indoor Localization, obtaining that in a LoS scenario, the model presents a median error of 0.5m. In contrast, in an NLoS scenario, the median error increases to 1.1 m.

A summary of the works discussed for Indoor Localization is presented in Table 2.5.

Findings for Indoor Localization with Wi-Fi CSI

Indoor localization presents remarkable differences when compared to the three previous types of applications presented. While Activity and Gesture Recognition and Vital Signs Monitoring focus on classifying human actions or estimating physiological signals, localization is inherently a problem of spatial estimation, requiring either the construction of detailed radio maps (fingerprint-based) or the precise calculation of signal propagation parameters (subspace-based). This fingerprint-based approaches, particularly those employing Deep Learning for pattern recognition in CSI fingerprints, achieve high accuracy but necessitate extensive, site-specific calibration data. In contrast, subspace-based methods offer a more generalized, model-driven solution by leveraging Angle of Arrival (AoA) and Time of Flight (ToF) estimations. However, they require sophisticated multi-antenna systems and complex signal processing.

2.3 Wi-Fi Sensing Applications using Channel State Information

Table 2.5: Indoor Localization Applications

Reference	Approach	Equipment	Detection	Performance
Li *et al.* (2016)	Subspace-based	2 Tx, antenna type and number not specified, 4 Rx-3 antennas, type not specified	MUSIC	Median error of 41 cm.
Kim *et al.* (2021)	Fingerprint-based regression	Tx-1 OMNI antenna, 16 Rx-3 OMNI antennas	Dense network	Localization accuracy of 1.28m in a complex environment.
Wang *et al.* (2021)	Fingerprint-based multiclass classification	Tx-1 OMNI antenna, Rx-3 OMNI antennas	Deep Residual Sharing Learning model	Maximum distance error of 2.46m.
Yang *et al.* (2023b)	Subspace-based	Tx-1 OMNI antenna, Rx-3 OMNI antennas	Based on AoA, ToF, and DFS	Median errors of 0.42 and 0.55 m in two different scenarios.
Yang *et al.* (2022)	Subspace-based	Tx-1 OMNI antenna, Rx-4 OMNI antennas	Based on AoA, ToF, and DFS	Localization error of 0.42 m.
Shi *et al.* (2018)	Fingerprint-based multiclass classification	Not specified	Bayesian models	Median distance error below 0.8m.
Wang *et al.* (2018b)	Fingerprint-based multiclass classification	4 Tx-1 OMNI antenna, Rx-1 OMNI antenna	Random Forest	Accuracy of 93.12% in an office environment.
Wu *et al.* (2018)	Fingerprint-based multiclass classification	Tx-2 OMNI antennas, Rx-3 OMNI antennas	Naive Bayes	Mean accuracy of 86% for four different scenarios.
Zhou *et al.* (2021)	Fingerprint-based regression	Cellphone as Rx	Dense network with Adaptive Genetic algorithm	Confidence probability within 4m is 90.47%.
Zhou *et al.* (2018)	Fingerprint-based regression	Tx-2 OMNI antennas, Rx-3 OMNI antennas	Dense network	Mean distance error of 1.08m and 1.50m in two scenarios.
Zhao *et al.* (2019)	Fingerprint-based multiclass classification	Tx-3 OMNI antennas, Rx-3 OMNI antennas	CNN	Mean distance error of 0.46 and 1.11m in LoS and NLoS scenarios.
Chen *et al.* (2017)	Fingerprint-based multiclass classification	Not specified	CNN	Mean estimation error of 1.36m and standard deviation of 0.9m.
Wang *et al.* (2018a)	Model-based estimation	11 Tx, 7 Rx, antennas not specified	Genetic algorithm	Median localization accuracy of 0.5m.

2.3.5 Crowd Counting

Wi-Fi CSI-based Crowd Counting applications determine the number of people in a room. These fingerprint-based applications can be addressed as a group classification or a regression problem.

Addressing Crowd Counting as a classification problem, Liu *et al.* (2017) presented WiCount. This system used CSI amplitude processed with low-pass and moving average filters and sanitized phase to count up to five moving people in a room by using a Dense Neural Network with a mean accuracy of 82.3%. Similarly, Cheng & Chang (2017) used a Dense Neural Network to count up to 9 people using only CSI amplitude with no processing, stating that noise reduction requires additional time and computational complexity. With this approach, the authors achieved an accuracy of 88%, and with a maximum of seven people, 90% of the errors were less than or equal to two people. Zou *et al.* (2018a) stated that a Crowd Counting classifier trained once is inconsistent in long-term deployments, as CSI measurements are affected by environmental dynamics and randomness of human movements. Therefore, they proposed the use of transfer learning and information fusion to generate a model robust to temporal and environmental disparities, obtaining an occupancy detection accuracy of 99.1% and a crowd counting accuracy of 92.8% with the proposed approach. In addition to Activity Recognition, Damodaran *et al.* (2020) employed an LSTM network architecture to determine the number of people in a room under two different conditions, resulting in two separate datasets. The first dataset contained CSI fingerprints from one to two people performing activities such as sitting, standing up, and walking. In contrast, up to three people remained static for the second dataset. The authors achieved accuracies of 99.72% and 97.40%, respectively. Sharma *et al.* (2022) used the Probability Mass Function as an image to serve as an input to a CNN, obtaining accuracies above 82% for scenarios with zero to four people.

Extending the functionality of Crowd counting applications to more challenging scenarios, Tian *et al.* (2021) proposed a variance-based method to determine whether a person is entering or leaving a room with an accuracy of nearly 100% in two different rooms with two to five people entering or exiting simultaneously.

For Crowd Counting, the authors used Dynamic Time Warping to compare the collected CSI with a pre-constructed database containing reference signals, obtaining an accuracy of 75% when five people were in the room. Ma *et al.* (2022) used the CSI phase information to estimate the number of people and used the MUSIC algorithm as a basis for crowd distribution visualization, obtaining accuracies above 97% for counting up to nine people walking and concluded that the performance of the application is dependent on the number of antennas, which needs to be more than the number of moving people based on results obtained from crowd distribution visualization. Guo *et al.* (2022) evaluated the performance of a Crowd Counting application in a scenario where the Wi-Fi receiver and transmitter are in different rooms, obtaining accuracies of 91.27% and 88.76% for counting up to five moving people in two different rooms.

Following the regression approach, Guo *et al.* (2017) extracted features from CSI amplitude and phase difference to estimate the number of people in the room with a logistic bounded exponential function obtaining accuracies above 89% when the samples used to evaluate the model are taken in the same indoor environment while involving different environments, the accuracy fell to 83%. Moreover, Choi *et al.* (2022) evaluated different Machine Learning regression models using CSI collected from ESP32 microcontrollers for Crowd Counting up to ten people and determined their location by dividing the room into four sections, achieving a mean absolute error of 0.41 for Crowd Counting and an accuracy of 98.1% for localization. By also using ESP32 microcontrollers as collecting devices, Kitagishi *et al.* (2022) designed a dense neural network for counting from zero to 20 people in steps of five in a 11.4×7.2 meter room, obtaining a counting error of $\pm$ 2.5 people for 76.5% of the cases.

Wang *et al.* (2020a) took a different approach from the one discussed above by determining the number of people based on the number of breathing signals detected through a spectrum analysis using FFT, with an average accuracy of 86% for counting up to four people. A summary of the works discussed for Crowd Counting is presented in Table 2.6.

Findings for Crowd Counting with Wi-Fi CSI

Crowd counting applications are distinguished for estimating a scalar quantity leveraging Deep Learning models for fingerprint-based classification and regression. A key finding is that the presented works demonstrate that crowd counting performance is highly dependent on the number of people, with accuracy decreasing as the crowd size increases. This behavior can be attributed to the fact that compound noise and multi-paths generated by the presence of multiple people create a more complex scenario when compared to other types of applications.

Table 2.6: Crowd Counting Applications

Reference	Approach	Equipment	Detection	Performance
Damodaran *et al.* (2020)	Group Classification	Tx-1 antenna, Rx-3 antennas, type not specified	LSTM network	99.72% accuracy for counting up to 3 people doing activities in a room.
Guo *et al.* (2017)	Regression	Not specified	Earth Mover Distance-based estimation	Accuracy over 81% for crowd counting in different rooms.
Guo *et al.* (2022)	Group Classification	Tx-1 antenna, Rx-3 antennas	Dense network	Accuracies of 88.76% and 91.27% in two scenarios.
Ma *et al.* (2022)	Group Classification	Not specified	CNN	Precision of 98.2% and 97% in two scenarios.
Kitagishi *et al.* (2022)	Regression	Single onboard antenna	Dense network	$\pm$ 2.5 counting error for up to 20 people.
Sharma *et al.* (2022)	Group Classification	Not specified	CNN	Accuracies $>$ 82% for up to 4 people.
Zou *et al.* (2018a)	Group Classification	Tx-3 OMNI antennas, Rx-3 OMNI antennas	Deep Learning model with Transfer Learning Kernel	Accuracy of 92.8% for crowd counting.
Choi *et al.* (2022)	Regression	Single onboard antenna	Machine Learning regression models	Median absolute error of 0.41 with up to 10 people.
Tian *et al.* (2021)	Group Classification	Tx-1 OMNI antenna, Rx-3 OMNI antennas	Dynamic Time Warping	Accuracy of 100% to 75% for one to five people.
Wang *et al.* (2020a)	Breathing-based	Tx and Rx-3 OMNI antennas	FFT analysis	Accuracy of 86%, estimation errors around 1 person.
Liu *et al.* (2017)	Group Classification	Tx-2 antennas, Rx-3 antennas	Dense network	Accuracy of 82.3% for up to five people.
Cheng & Chang (2017)	Group Classification	Tx-1 antenna, Rx-2 antennas	Dense network	Accuracy of 96.90% to 88.66% from one to nine people.

2.3.6 Human Pose Estimation

Wi-Fi CSI-based Human Pose Estimation applications are relatively more recent than other Human Wi-Fi sensing applications, with fewer works in the state-of-the-art. Human Pose Estimation involves defining a 3D skeleton that represents a person's current position and determining the locations of their joints and limbs.

WiPose, developed by Jiang *et al.* (2020), is one of the first systems for estimating 3D Human Pose using commercial Wi-Fi devices. WiPose aims to reconstruct 3D human body skeletons using commercial Wi-Fi devices and Deep Learning models for classification, resulting in a joint localization error of 2.83 cm. Guo *et al.* (2020) built images from CSI and used them as input for a Deep Learning network designed to map CSI images to human skeleton images. In addition, Ren *et al.* (2022) presented GoPose, an application that estimates human pose and tracks free-form movements of multiple limbs based on the reflection of Wi-Fi signals, with an average joint localization error of 4.7 cm. This application utilized the bi-dimensional angle of arrival of the reflected signals to identify moving limbs, estimating the position and trajectory of each limb using multiple transmitter-receiver pairs.

Recent approaches utilize self-attention and multi-head attention modules to enhance pose estimation, as seen in works by Zhou *et al.* (2022, 2023b). These studies evaluated the trained models using the Percentage of Correct Key Points (PCK) with a threshold of 50%, achieving PCK values of 96.80% and 88.74%, respectively. Additionally, Zhou *et al.* (2022) proposed an Attention-Based subcarrier selection model to pay more attention to feature-rich subcarriers, being the first work that applies Deep Learning for subcarrier selection. Table 2.7 summarizes the works selected and related to Human Pose Estimation.

Findings for Human Pose Estimation with Wi-Fi CSI

Compared to other applications, fewer works have explored these applications due to their complexity residing in reconstructing skeletal kinematics rather than classifying coarse movements or quantities. It can be seen that pose estimation requires sophisticated hardware, leveraging equipment configurations with multiple antennas and even multiple Rx and Tx network devices. These applications

are exclusively reliant on advanced Deep Learning architectures, such as CNNs and LSTMs, enhanced with self-attention modules to achieve joint localization by modelling spatial and temporal dependencies of human movement, evaluated through metrics like joint localization error and the PCK, obtaining results that remark the feasibility and potential of using Wi-Fi CSI for device-free Human Pose Estimation.

Table 2.7: Human Pose Estimation Applications

Reference	Equipment	Detection	Performance
Zhou *et al.* (2022)	Tx-3 OMNI antennas, Rx-3 OMNI antennas	Dense network	88.74% of correct keypoints for PCK 50.
Guo *et al.* (2020)	Tx-1 OMNI antenna, 2 Rx-3 OMNI antennas	CNN	Over 25% of captured human pose images matched the ground-truth skeleton strictly.
Jiang *et al.* (2020)	Tx- 1 OMNI antenna, 3 Rx-3 OMNI antennas	CNN-LSTM network	It can localize each joint on the human skeleton with an average error of 2.83cm.
Ren *et al.* (2022)	Tx-3 OMNI antennas, 4 Rx-3 OMNI antennas	CNN-LSTM network	Median joint estimation error of 4.3cm. The average joint localization error ranges from 3 to 8.4cm.
Zhou *et al.* (2023b)	Tx-1 OMNI antenna, Rx-3 OMNI antennas	CNN with self-attention Transformer	96.8% of correct keypoints for PCK 50.

2.3.7 Gait Analysis

Systems for Gait Analysis using Wi-Fi CSI focus on determining values associated with gait, such as the number of steps or pace of a person. An example of this type of application is WiSpeed, a system developed by Zhang *et al.* (2018) which, aside from detecting falls, also estimates pace based on a local peak algorithm for further estimating the stride length and the total walked distance, resulting in a mean absolute percentage error of 4.85% for pace estimation and errors around 5% for estimating the walked distance. Furthermore, Liu *et al.* (2019) developed Wi-Run, an application that estimates the steps of both multi-person and single-person running in place by processing CSI amplitude with a Savitzky-Golay filter and identifying the start of the running event based on a thresholding method, while steps are estimated using a peak detection algorithm. This approach presents a step estimation accuracy of 93.18% for single-person estimation and 81.4% for estimating steps from up to five people. Additionally,

Lin *et al.* (2022) estimated pace and walked distance by spectrum analysis with Fourier Transform. Once both parameters were estimated, the authors used their values to calculate the step length and walking direction. They achieved an estimation error for step length between 0.02 and 0.16 meters and an error for pace between 0.6 and 9.6 steps per minute. For estimating the walking direction, the developed system presented errors below 8.84 degrees as long as the person walked in a straight line.

Shah *et al.* (2020) performed gait analysis using Wi-Fi CSI to monitor people with Parkinson's disease by using spectrograms obtained from CSI to identify the freeze of gait and a fast or slow pace with a CNN, obtaining that the system can achieve a classification accuracy of 98.1% for recognizing a fast or slow pace, a voluntary stop or freeze of gait.

A summary of the works discussed for Gait Analysis is presented in Table 2.8.

Findings for Gait Analysis with Wi-Fi CSI

Wi-Fi CSI-based Gait Analysis demonstrates accurate results in quantifying gait-related metrics such as cadence, step count, and walked distance. A key finding obtained by reviewing works for this application is that foundational signal processing techniques for peak detection and frequency analysis are preferred over Machine and Deep Learning approaches. Furthermore, this technology has demonstrated its viability for healthcare support, as evidenced by its use in the detection of gait-freezing episodes due to neurodegenerative conditions like Parkinson's disease.

2.3.8 Presence Detection

Often referred to as Occupancy Detection, this type of applications aims to detect the presence of people inside a room, usually based on CSI fingerprints with an empty and occupied room, such as the work presented by Tan *et al.* (2018), where they measure the similarity between reference and testing CSI fingerprints with Euclidean distance, resulting in occupancy detection accuracy above 90% whether the person is moving or static in the room.

Similar to other types of Wi-Fi sensing applications, Presence Detection can be achieved using Machine Learning models. Zhou *et al.* (2017) collected a total of 2000 CSI samples from an occupied and an empty room. Principal components were then extracted from the CSI samples to reduce the computational complexity and used as input to an SVM, resulting in an accuracy above 97%. Moreover, Yang *et al.* (2018) integrated Wi-Fi devices, cloud services, and a user program for occupancy detection, delegating Machine Learning computations to the cloud service, resulting in an accuracy of 96.8%. In addition, Mesa-Cantillo *et al.* (2023) performed a comparative study by evaluating the performance of binary classifiers, including k-NN, SVM, Decision Trees, Naive Bayes, and Linear and Quadratic Discriminants, with time-domain features extracted from CSI amplitude. Results showed that the highest accuracy was achieved with SVMs, yielding a cross-validation accuracy of over 92% in two different scenarios. By also evaluating different Machine Learning models, Soto *et al.* (2022b) proposed using Dynamic Time Warping to obtain a feature associated with the similarity between a reference signal corresponding to an empty room and the signal corresponding to a new instance to be classified. Although obtaining an accuracy of 99.21% with an SVM with a total of 234 features, including the proposed, the authors applied a feature selection algorithm to reduce the number of features to five, resulting in an accuracy of 99.98%. Still, the authors do not specify if one

Table 2.8: Gait Analysis Applications

Reference	Gait Parameter	Equipment	Detection	Performance
Lin *et al.* (2022)	Cadence, walking distance, step length, and walking direction	2 Tx and 2 Rx, antenna types or number not specified	STFT analysis	Estimation error of 0.02-0.16m for step length, 0.01-0.16 Hz for cadence, and less than 8.84 degrees for walking direction.
Liu *et al.* (2019)	Number of steps	Tx-1 antenna, Rx- 3 antennas (type not specified)	Peak detection	Step estimation accuracy of 93.18%, 91.73%, 90.25%, 88.44%, and 81.4% for one to five runners.
Shah *et al.* (2020)	Freeze of gait detection, as well as walking slow, voluntary stop, wlking fast, sit and stand up classification.	Tx and Rx-1 OMNI antenna	CNN with encoder-decoder layers	By doing data fusion with radar signals, mean accuracy of 98.1%.
Zhang *et al.* (2018)	Walking distance	Tx-2 OMNI antennas, Rx-3 OMNI antennas	Peak detection	Mean absolute percentage error of 4.85%.

of the selected features is the Dynamic Time Warping value.

Deep Learning models have also been used in Presence Detection applications, obtaining satisfactory results with automatic feature extraction. For example, Shi *et al.* (2020) focused on detecting the presence of an infant in the back of a vehicle, differentiating it from a pet or an object based on CSI Radio Images and CNNs. A CSI Radio Image is a graphical representation of the CSI matrix formed by each subcarrier's amplitude and phase values. With this approach, the system differentiates the presence of a person from pets and objects with accuracies over 95%. Similarly, Liu *et al.* (2022) used two different images constructed from the Discrete Fourier Transform of the CSI amplitude and phase data as inputs, thereby achieving an accuracy of 95.55% with the use of a CNN.

Unlike fingerprint-based applications, WiFree, the system developed by Zou *et al.* (2018a), used a threshold-based method to identify whether CSI contains information of an occupied or an empty space by using an index to measure the similarity between the amplitude curves of different subcarriers, establishing that, if the index exceeded the threshold, the room was occupied. Moreover, Hang *et al.* (2019) calculated a motion indicator based on the correlation of CSI time and frequency domain data, as they found that when there is no one in the room, the cross-correlation between CSI samples is considerably higher compared to when a person is present. Hence, the room is occupied if the motion indicator is below a threshold. This approach enabled them to achieve an accuracy above 98%.

The performance of the previously discussed Presence Detection works is shown in Table 2.9.

Findings for Presence Detection with Wi-Fi CSI

The accuracies presented in these works demonstrate the robustness of Wi-Fi CSI for achieving Presence Detection, by mostly surpassing 95% accuracy across the reviewed works. In contrast to other types of applications, Presence Detection is versatile in technique, as it can be achieved not only by Machine and Deep Learning models, but also through threshold-based and similarity-based algorithms that leverage CSI properties shown in the presence of a person. Furthermore, these works state effective performance regardless of occupant movement

state, i.e., static or moving, demonstrating its practicality for real-world deployment in smart buildings, security systems, and specialized use-cases like vehicle occupancy monitoring.

Table 2.9: Presence Detection Applications

Reference	Equipment	Detection	Performance
Liu *et al.* (2022)	Tx-3 antennas, Rx-3 antennas	CNN	Presence detection performance of 95.55%.
Mesa-Cantillo *et al.* (2023)	Tx-2 OMNI antennas, Rx-3 OMNI antennas	SVM	92% cross-validation accuracy.
Zou *et al.* (2018a)	Tx-3 OMNI antennas, Rx-3 OMNI antennas	Deep Learning model with Transfer Learning Kernel	Presence detection accuracy of 99.1%.
Yang *et al.* (2018)	Tx-1 antenna, Rx-3 antennas	Signal Tendency Index algorithm	Presence detection accuracy of 96.8%.
Soto *et al.* (2022b)	Not specified	SVM	Accuracies > 99%.
Zhou *et al.* (2017)	Tx-2 antennas, Rx-3 antennas	SVM	Presence detection precision over 97%.
Shi *et al.* (2020)	Tx-3 OMNI antennas, Rx-3 OMNI antennas	CNN network	Presence detection accuracy above 95%.
Tan *et al.* (2018)	Tx-1 antenna, Rx-3 antennas	Similarity-based algorithm	Presence detection accuracy above 96%.
Hang *et al.* (2019)	Commercial router as Tx, Rx-3 antennas	Thresholding algorithm	Presence detection accuracy above 98%.

2.3.9 People Identification

Wi-Fi CSI-based People Identification applications attempt to identify a person within a group of n people based on CSI fingerprints specific to that person. CSI fingerprints that can be associated with a particular person are referred to in the literature as *Radio Biometrics*, which can be defined as variations captured in a wireless signal caused by the presence of a person, leading it to contain information about that person's identity. An example of a People Identification application is Freesense, a system developed by Xin *et al.* (2016) that applies a low-pass filter and extracts Principal Components from CSI for then implementing k-NN algorithm with the value of similarity obtained with Dynamic Time Warping as distance metric to measure the similarity between a reference signal and one captured when a person passes between the LoS of Tx and Rx. Freesense can identify people with an accuracy of 94.5% when identifying from a group of two people, but this accuracy falls to 75% when the group increases to nine people.

One of the first works to introduce the concept of Radio Biometrics for People Identification using Wi-Fi is the work presented by Xu *et al.* (2017). The authors employed a time-reversal technique to extract Radio Biometrics from CSI, constructing a real-valued vector for further classification with k-NN in a through-the-wall scenario. This approach achieved an accuracy of 98.78% with a group of 11 people. Aiming to identify the person who is in the driver's seat of a car from a group of up to seven people, Regani *et al.* (2020) observed that as the group size increases, the performance of the developed system decreases as it is more likely that people share similar Radio Biometrics, as they obtained a cross-validation accuracy of 99.13% with a Dense Neural Network when there were only two people in the group, decreasing to 53.85% when the group size increased to seven. Shah & Kanhere (2019) used breathing and heart rate patterns extracted from CSI as Radio Biometrics by analyzing chest displacements caused by cardiopulmonary activity. Authors obtained accuracies from 84% to 65% for identifying people from a group of two to five using an SVM and preprocessing techniques such as multipath removal, Hampel identifier, and wavelet transform.

Another approach for achieving People Identification is by extracting Radio Biometrics from a person's gait. When walking, a person may exhibit characteristic body movement patterns that can distinguish them from a group of people. As in Zeng *et al.* (2016), where the authors implemented high delay multipath removal as a filtering technique based on the PDP and, by using a tree-based Machine Learning classifier which had as inputs extracted features from gait patterns, they achieved accuracies from 92% to 80% when the group size varied from two to six people respectively. Features associated with gait patterns from the time and frequency domains were proposed as Radio Biometrics for People Identification by Lv *et al.* (2017), resulting in an accuracy of 90.9% in identifying people from a group of eight. Zhang *et al.* (2020b) proposed the gait-body coordinate velocity profile as a Radio Biometric extracted from CSI, which characterizes gait patterns from people by representing the velocities of body parts during walking. By using a CNN-based network architecture with LSTM, the authors obtained accuracies of 99%, 93.2%, and 76.2% for recognizing between two, five, and eleven people, respectively. Similarly, features from gait and breathing patterns were extracted by Wang *et al.* (2022) and used as inputs to a Deep Learning network

that led to an accuracy of 98.33% with a group size of 15 people. Zhang *et al.* (2020a) associated frequencies present in spectrograms generated from CSI with the movements of different body parts in a person's gait. The authors then extracted features from the spectrogram to be used in an SVM for classification, obtaining accuracies ranging from 92.83% to 78.28% when the group size varied from three to six people.

To enhance the performance of a People Identification system, Gu *et al.* (2021) placed a metal plate in front of Tx, following a model based on Rician fading, to generate multipaths that contain relevant Radio Biometrics information. With this addition and a CNN that uses CSI spectrogram as input images, the authors achieved an accuracy of 94.3% for recognizing between a spoofer and a legitimate user, i.e., binary classification. A similar approach was followed by Zhao *et al.* (2021), which also performs binary classification to distinguish legitimate users from spoofers using an SVM when volunteers perform a combination of gestures, thus obtaining an average accuracy of 92.9%.

A summary of the works associated with the People Identification task discussed in this section is presented in Table 2.10.

Findings for People Identification with Wi-Fi CSI

These works define what is known as radio biometrics, which can be derived from perturbations in the Wi-Fi signal caused by gait, breathing patterns, and body movements, which can be characterized to distinguish between individuals. Most works show that the performance degrades as the number of people to be identified increases, as the distinctiveness of radio biometrics diminishes, increasing inter-subject similarities. Methodologically, these applications employ different detection algorithms, such as Machine Learning algorithms to Deep Learning models based on architectures that are also commonly used in other Wi-Fi CSI applications, such as CNNs and LSTMs, which have been demonstrated to be able to capture and distinguish between up to 19 people under real environments.

Table 2.10: People Identification Applications

Reference	Max No. of People	Equipment	Detection	Performance
Zhang *et al.* (2020b)	11	1 Tx, 6 Rx	CNN-LSTM network	Accuracy of 76.2% for identificating 11 people.
Xin *et al.* (2016)	9	Tx-2 antennas, Rx-3 antennas	k-NN	Accuracy of 94.5% to 75.5% while varying the number of people.
Zhao *et al.* (2021)	10	Tx-2 OMNI antennas, 2 Rx-3 OMNI antennas	SVM	Mean accuracy of 92.9%.
Wang *et al.* (2022)	15	Tx-3 OMNI antennas, Rx-1 OMNI antenna	CNN	Accuracy in a non-interference environment of 98.33%.
Lv *et al.* (2017)	8	Tx-1 OMNI antennas, Rx-3 OMNI antennas	SVM	Accuracy from 98.7% to 90.9% while varying the number of people.
Zeng *et al.* (2016)	6	Tx-3 OMNI antennas, Rx-3 Omni antennas	Tree-based classifier	Accuracy of 92% to 80% while varying the number of people.
Shah & Kanhere (2019)	5	Tx-1 OMNI antenna, Rx-2 OMNI antennas	SVM	Accuracy of 84% to 65% while varying the number of people.
Gu *et al.* (2021)	19	Tx-1 OMNI antenna, Rx-1 OMNI antenna	CNN	Accuracy of 94.3% in a real environment.
Xu *et al.* (2017)	11	Tx-3 OMNI antennas, Rx-3 OMNI antennas	Time-Reversal Resonance Strength	Accuracy of 98.78% achieved for 11 people.
Regani *et al.* (2020)	7	Tx-2 antennas, Rx-3 antennas	Dense network	Accuracy of 99.13% to 53.85% while varying the number of people.
Zhang *et al.* (2020a)	6	Tx-1 OMNI antenna, 2 Rx-3 OMNI antennas	SVM	Accuracy of 92.82% to 78.28% while varying the number of people.

2.4 Summary

It was found that human Wi-Fi sensing applications can be classified into nine categories, according to the task being performed. These categories are Activity Recognition, Gesture Recognition, Vital Signs Monitoring, Indoor Localization, Crowd Counting, Human Pose Estimation, Gait Analysis, Presence Detection, and People Identification. Fig. 2.2 illustrates the distribution of works selected per application, reflecting the state-of-the-art for Wi-Fi sensing. As can be seen, most of the works focus on Activity Recognition, followed by Vital Signs Monitoring and Gesture Recognition. These three categories represent 50% of the total distribution. Meanwhile, Human Pose Estimation works represent 4.31%, which can be considered as one of the newest tasks achieved using Wi-Fi CSI, considering that the first works performing Human Pose Estimation are from 2020.

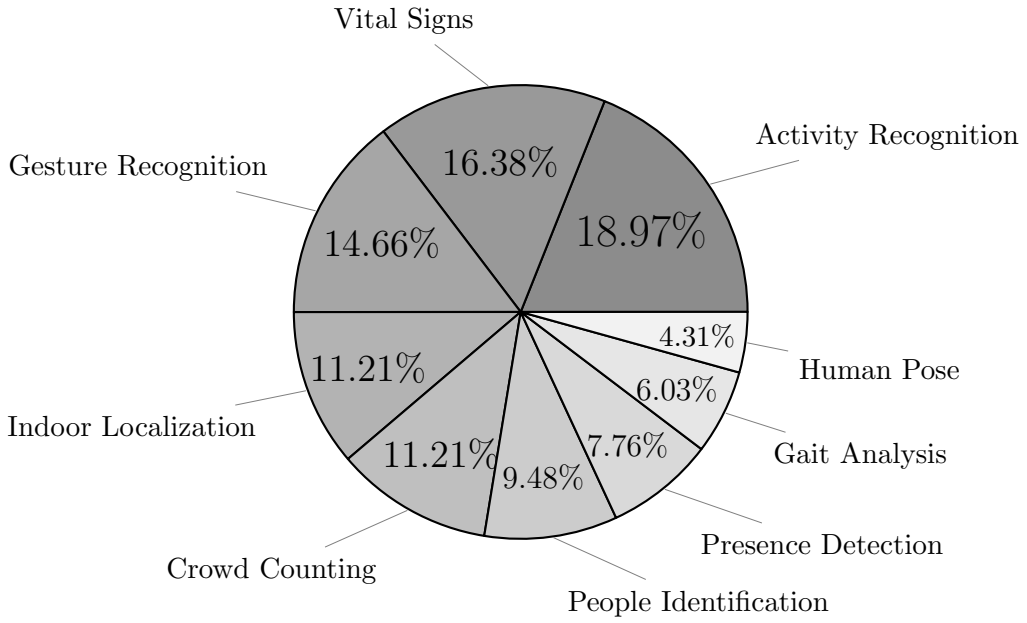

Figure 2.2: Distribution of applications per sensing category.

In the state-of-the-art, the authors used tools for collecting CSI from Wi-Fi devices. As will be explained in Chapter 3, CSI is not considered to be delivered to the final user, and its use is considered exclusively for physical layer manage-

ment. Thus, in order to speed up the experimentation process of Wi-Fi sensing applications, the authors choose to use CSI collecting tools that enable the collection of CSI. The most popular tools that can be found in the state-of-the-art are the *Linux 802.11n CSI Tool*, *Atheros CSI Tool*, *Nexmon*, and tools based on the use of ESP32 microcontrollers, being the Linux 802.11n CSI Tool the most used. This fact can be seen in Table 2.11.

Table 2.11: Used CSI Collecting Tools in the literature

Tool	Works
Linux 802.11n CSI Tool	Damodaran *et al.* (2020), Nakamura *et al.* (2022) Zhou *et al.* (2023a), Arshad *et al.* (2017), Wu *et al.* (2019), Xiao *et al.* (2020), Zhu *et al.* (2022), Lee *et al.* (2020), Zhao *et al.* (2021), Chen *et al.* (2024), Wang *et al.* (2017), Zhang *et al.* (2023), Showmik *et al.* (2023), Li *et al.* (2016), Zhang *et al.* (2018), Tian *et al.* (2018), Tong *et al.* (2023), Meng *et al.* (2022), Yang *et al.* (2023a), Bu *et al.* (2022), Chen *et al.* (2022), Jia *et al.* (2020), Li *et al.* (2016), Ali *et al.* (2017), Niu *et al.* (2019), Wu *et al.* (2020), Fu *et al.* (2019), Gu *et al.* (2023), Liu *et al.* (2018), Wang *et al.* (2020b), Zhang *et al.* (2017), Wang *et al.* (2020c), Xiaolong *et al.* (2021), Lee *et al.* (2018), Zeng *et al.* (2020), Gui *et al.* (2023), Dou & Huan (2021), Tsubota *et al.* (2023), Li *et al.* (2016), Kim *et al.* (2021), Wang *et al.* (2021), Shi *et al.* (2018), Wang *et al.* (2018b), Wu *et al.* (2018), Zhou *et al.* (2018), Zhao *et al.* (2019), Chen *et al.* (2017), Damodaran *et al.* (2020), Guo *et al.* (2017), Guo *et al.* (2022), Ma *et al.* (2022), Tian *et al.* (2021), Cheng & Chang (2017), Zhou *et al.* (2022), Guo *et al.* (2020), Jiang *et al.* (2020), Ren *et al.* (2022), Lin *et al.* (2022), Liu *et al.* (2019), Shah *et al.* (2020), Zhang *et al.* (2018), Mesa-Cantillo *et al.* (2023), Yang *et al.* (2018), Zhou *et al.* (2017), Shi *et al.* (2020), Tan *et al.* (2018), Hang *et al.* (2019), Zhang *et al.* (2020b), Xin *et al.* (2016), Zhao *et al.* (2021), Wang *et al.* (2022), Lv *et al.* (2017), Zeng *et al.* (2016), Shah & Kanhere (2019), Gu *et al.* (2021), Zhang *et al.* (2020a)
Atheros CSI Tool	Cui *et al.* (2021), Zhang *et al.* (2021), Hao *et al.* (2020), Zou *et al.* (2018), Mosleh *et al.* (2022), Zou *et al.* (2018a), Zhou *et al.* (2023b), Liu *et al.* (2022), Zou *et al.* (2018a)
Nexmon	Meneghello *et al.* (2023), Yin *et al.* (2022), Liu *et al.* (2021), Yang *et al.* (2023b), Yang *et al.* (2022), Zhou *et al.* (2021), Sharma *et al.* (2022), Soto *et al.* (2022b)
ESP32-based Tool	Natarajan *et al.* (2024), Hernandez *et al.* (2022), Hernandez *et al.* (2022), Alzaabi *et al.* (2024), Atif *et al.* (2022), Kitagishi *et al.* (2022), Choi *et al.* (2022)
Others	Zou *et al.* (2018b), Niu *et al.* (2018), Chen *et al.* (2018), Liu *et al.* (2016), Niu *et al.* (2018), Wang *et al.* (2018a), Wang *et al.* (2020a), Liu *et al.* (2017), Xu *et al.* (2017), Regani *et al.* (2020)

Every work in the state-of-the-art preprocess CSI so it can be used for sensing applications. Preprocessing techniques aim to extract useful information from CSI as freely as possible from noise and in a format that allows that extracted information can be used for detection algorithms. Wi-Fi sensing preprocessing techniques can be classified into the following categories depending on the operation that these perform to Wi-Fi CSI: *Noise Reduction*, *Signal Transformation*, *Complexity Reduction*, and *Feature Extraction*.

Table 2.12 categorizes the works discussed in this chapter according to the

corresponding technique implemented, making it easier to identify the most commonly used methods.

The categories and techniques outlined in this section will be further elaborated in Chapter 3.

Machine and Deep Learning algorithms are the predominant methodologies for Wi-Fi sensing tasks, typically implemented as binary or multi-class classifiers due to their demonstrated reliability and efficacy. In contrast, approaches for Vital Signs Monitoring, such as the extraction of breathing rates, often avoid these data-intensive methods in favor of signal processing techniques. Among these, the FFT is frequently employed, as a frequency-domain analysis is often sufficient to isolate breathing patterns from CSI without the need of complex computational models.

Table 2.13 classifies the works depending on the detection task that the authors performed. From this table, it is important to notice that most works perform multi-class classification. Activity Recognition and Gesture Recognition, two of the most common Wi-Fi sensing applications, involve recognizing an activity or gesture from a set of more than two different activities. In contrast, Presence Detection is achieved through binary classification, as it consists of detecting whether a person is present or not.

Table 2.12: Preprocessing techniques commonly found in literature

Category	Works
Noise Reduction	**Low-pass filter:** Hao *et al.* (2020), Zhou *et al.* (2023a), Tong *et al.* (2023), Xin *et al.* (2016), Arshad *et al.* (2017), Wu *et al.* (2019), Chen *et al.* (2022), Bu *et al.* (2022), Xiao *et al.* (2020), Zhu *et al.* (2022), Lee *et al.* (2020), Jia *et al.* (2020), Li *et al.* (2016), Ali *et al.* (2017), Zhao *et al.* (2021), Lin *et al.* (2022), Wang *et al.* (2022), Lv *et al.* (2017), Mosleh *et al.* (2022), Gui *et al.* (2023), Sharma *et al.* (2022), Mesa-Cantillo *et al.* (2023), Chen *et al.* (2024)
	Multipath removal: Zhang *et al.* (2017), Lee *et al.* (2018), Chen *et al.* (2022), Zeng *et al.* (2016), Shah & Kanhere (2019)
	DWT: Wang *et al.* (2020b), Xin *et al.* (2016), Zhao *et al.* (2021), Wang *et al.* (2017), Zou *et al.* (2018a), Yin *et al.* (2022), Liu *et al.* (2016), Guo *et al.* (2020), Gu *et al.* (2021), Yang *et al.* (2018), Zhang *et al.* (2023), Natarajan *et al.* (2024), Alzaabi *et al.* (2024)
	Hampel filter: Liu *et al.* (2018), Wang *et al.* (2020b), Guo *et al.* (2022), Chen *et al.* (2022), Li *et al.* (2016), Gui *et al.* (2023), Sharma *et al.* (2022), Shah & Kanhere (2019), Liu *et al.* (2016), Alzaabi *et al.* (2024), Wang *et al.* (2016), Dou & Huan (2021), Liu *et al.* (2019), Choi *et al.* (2022), Soto *et al.* (2022b), Tian *et al.* (2021), Tsubota *et al.* (2023)
	Moving Average filter: Liu *et al.* (2018), Wang *et al.* (2020c), Kitagishi *et al.* (2022), Yin *et al.* (2022), Alzaabi *et al.* (2024), Wang *et al.* (2016), Soto *et al.* (2022b), Hernandez *et al.* (2022)
	Savitzky-Golay filter: Zeng *et al.* (2020), Shah & Kanhere (2019), Alzaabi *et al.* (2024), Dou & Huan (2021), Liu *et al.* (2019), Choi *et al.* (2022), Niu *et al.* (2018), Niu *et al.* (2019), Wu *et al.* (2020), Showmik *et al.* (2023)
	Phase Sanitization: Wang *et al.* (2020b), Hao *et al.* (2020), Zhou *et al.* (2023a), Wang *et al.* (2020c), Guo *et al.* (2017), Tong *et al.* (2023), Guo *et al.* (2022), Meng *et al.* (2022), Zou *et al.* (2018), Kim *et al.* (2021), Ma *et al.* (2022), Wang *et al.* (2021), Liu *et al.* (2022), Xiaolong *et al.* (2021), Kitagishi *et al.* (2022), Zhou *et al.* (2022), Yang *et al.* (2023a), Wu *et al.* (2020), Xu *et al.* (2017), Yang *et al.* (2023b), Meneghello *et al.* (2023)
	MUSIC: Li *et al.* (2016), Tian *et al.* (2018), Chen *et al.* (2018), Yang *et al.* (2022)
Signal Transformation	**FFT:** Wang *et al.* (2020b), Xiao *et al.* (2020), Lin *et al.* (2022), Mosleh *et al.* (2022), Yin *et al.* (2022), Liu *et al.* (2016), Niu *et al.* (2018), Liu *et al.* (2021), Wang *et al.* (2020a)
	PSD: Liu *et al.* (2018), Alzaabi *et al.* (2024), Wang *et al.* (2016), Liu *et al.* (2019), Atif *et al.* (2022)
Complexity Reduction	**PCA:** Liu *et al.* (2018), Nakamura *et al.* (2022), Hao *et al.* (2020), Yang *et al.* (2023a), Xin *et al.* (2016), Zeng *et al.* (2020), Wu *et al.* (2019), Chen *et al.* (2022), Xiao *et al.* (2020), Zhu *et al.* (2022), Ali *et al.* (2017), Zhao *et al.* (2021), Lin *et al.* (2022), Lv *et al.* (2017), Chen *et al.* (2024), Shah & Kanhere (2019), Wang *et al.* (2017), Gu *et al.* (2021), Alzaabi *et al.* (2024), Hernandez *et al.* (2022), Shah *et al.* (2020), Zhou *et al.* (2017), Regani *et al.* (2020), Zhang *et al.* (2020a), Zhang *et al.* (2021), Li *et al.* (2016), Fu *et al.* (2019), Shi *et al.* (2018), Shi *et al.* (2020)
	Variance-based Subcarrier Selection: Liu *et al.* (2018), Xiaolong *et al.* (2021), Lee *et al.* (2018), Guo *et al.* (2020), Wang *et al.* (2016)
	Signal Segmentation: Guo *et al.* (2022), Tong *et al.* (2023), Lee *et al.* (2018), Xin *et al.* (2016), Zeng *et al.* (2020), Gui *et al.* (2023), Natarajan *et al.* (2024), Showmik *et al.* (2023)
Feature Extraction	**Manual Feature Extraction:** Liu *et al.* (2018), Hao *et al.* (2020), Tian *et al.* (2018), Guo *et al.* (2017), Guo *et al.* (2022), Xiaolong *et al.* (2021), Xin *et al.* (2016), Arshad *et al.* (2017), Wu *et al.* (2019), Chen *et al.* (2022), Li *et al.* (2016), Ali *et al.* (2017), Zhao *et al.* (2021), Lv *et al.* (2017), Zeng *et al.* (2016), Shah & Kanhere (2019), Wang *et al.* (2017), Zou *et al.* (2018a), Natarajan *et al.* (2024), Choi *et al.* (2022), Soto *et al.* (2022b), Niu *et al.* (2018), Zhou *et al.* (2017), Regani *et al.* (2020), Zhang *et al.* (2020a), Li *et al.* (2016), Fu *et al.* (2019), Shi *et al.* (2018), Wang *et al.* (2018b), Wu *et al.* (2018), Zhou *et al.* (2021)
	Automatic Feature Extraction: Nakamura *et al.* (2022), Zhang *et al.* (2020b), Zhou *et al.* (2023a), Tong *et al.* (2023), Meng *et al.* (2022), Zou *et al.* (2018), Kim *et al.* (2021), Ma *et al.* (2022), Wang *et al.* (2021), Liu *et al.* (2022), Kitagishi *et al.* (2022), Zhou *et al.* (2022), Yang *et al.* (2023a), Cui *et al.* (2021), Bu *et al.* (2022), Zhu *et al.* (2022), Lee *et al.* (2020), Jia *et al.* (2020), Wang *et al.* (2022), Mosleh *et al.* (2022), Sharma *et al.* (2022), Chen *et al.* (2024), Guo *et al.* (2020), Gu *et al.* (2021), Yang *et al.* (2018), Zhang *et al.* (2023), Hernandez *et al.* (2022), Showmik *et al.* (2023), Meneghello *et al.* (2023), Shah *et al.* (2020), Zhang *et al.* (2021), Shi *et al.* (2020), Jiang *et al.* (2020), Ren *et al.* (2022), Liu *et al.* (2017), Gu *et al.* (2023), Cheng & Chang (2017), Zou *et al.* (2018b), Zhou *et al.* (2018), Zhao *et al.* (2019), Chen *et al.* (2017), Zhou *et al.* (2023b)

Table 2.13: Detection tasks performed in the literature

Detection Task	Works
Binary Classification	**Activity Recognition:** Nakamura *et al.* (2022), Zhang *et al.* (2018)
	People Identification: Gu *et al.* (2021)
	Presence Detection: Liu *et al.* (2022), Mesa-Cantillo *et al.* (2023), Zou *et al.* (2018a), Yang *et al.* (2018), Soto *et al.* (2022b), Zhou *et al.* (2017), Shi *et al.* (2020), Tan *et al.* (2018), Hang *et al.* (2019)
Multi-class Classification	**Activity Recognition:** Zhou *et al.* (2023a), Arshad *et al.* (2017), Wu *et al.* (2019), Cui *et al.* (2021), Xiao *et al.* (2020), Zhu *et al.* (2022), Lee *et al.* (2020), Zhao *et al.* (2021), Chen *et al.* (2024), Wang *et al.* (2017), Zhang *et al.* (2023), Natarajan *et al.* (2024), Hernandez *et al.* (2022), Showmik *et al.* (2023),Meneghello *et al.* (2023), Zhang *et al.* (2021), Li *et al.* (2016), Zou *et al.* (2018b), Liu *et al.* (2018),Liu *et al.* (2016)
	People Identification: Zhang *et al.* (2020b), Xin *et al.* (2016), Zhao *et al.* (2021), Wang *et al.* (2022), Lv *et al.* (2017), Zeng *et al.* (2016), Shah & Kanhere (2019),Xu *et al.* (2017), Regani *et al.* (2020), Zhang *et al.* (2020a)
	Gait Analysis: Shah *et al.* (2020)
	Gesture Recognition: Hao *et al.* (2020), Tian *et al.* (2018), Tong *et al.* (2023), Meng *et al.* (2022), Zou *et al.* (2018), Yang *et al.* (2023a), Chen *et al.* (2022), Bu *et al.* (2022), Jia *et al.* (2020), Li *et al.* (2016), Ali *et al.* (2017), Hernandez *et al.* (2022), Niu *et al.* (2018), Niu *et al.* (2019), Fu *et al.* (2019), Gu *et al.* (2023)
	Crowd Counting: Chen *et al.* (2018), Guo *et al.* (2022), Ma *et al.* (2022), Sharma *et al.* (2022), Zou *et al.* (2018a), Tian *et al.* (2021), Liu *et al.* (2017), Cheng & Chang (2017)
	Vital Signs Monitoring: Xiaolong *et al.* (2021), Mosleh *et al.* (2022)
	Indoor Localization: Wang *et al.* (2021), Shi *et al.* (2018), Wang *et al.* (2018b), Wu *et al.* (2018), Zhao *et al.* (2019), Chen *et al.* (2017)
	Human Pose Estimation: Guo *et al.* (2020), Jiang *et al.* (2020)
Estimation	**Gesture Recognition:** Wu *et al.* (2020)
	Gait Analysis: Lin *et al.* (2022), Liu *et al.* (2019), Zhang *et al.* (2018)
	Vital Signs Monitoring: Liu *et al.* (2018), Zhang *et al.* (2017), Chen *et al.* (2018), Wang *et al.* (2020c), Zeng *et al.* (2020), Gui *et al.* (2023), Yin *et al.* (2022), Liu *et al.* (2016), Alzaabi *et al.* (2024), Wang *et al.* (2016), Dou & Huan (2021), Tsubota *et al.* (2023), Niu *et al.* (2018), Liu *et al.* (2021), Atif *et al.* (2022)
	People Counting: Guo *et al.* (2017), Kitagishi *et al.* (2022), Choi *et al.* (2022), Wang *et al.* (2020a)
	Indoor Localization: Li *et al.* (2016), Kim *et al.* (2021), Yang *et al.* (2023b), Yang *et al.* (2022), Zhou *et al.* (2021), Zhou *et al.* (2018), Wang *et al.* (2018a)
	Human Pose Estimation: Zhou *et al.* (2022), Ren *et al.* (2022), Zhou *et al.* (2023b)

Chapter 3

Wi-Fi Sensing Fundamentals

3.1 Channel State Information

Channel State Information (CSI) originates in the physical layer (PHY), and its purpose is to mitigate the multipath fading effects of a wireless communication channel as it allows MIMO-based systems to adapt to current channel conditions, allowing them to optimize aspects such as beamforming, power allocation, and modulation schemes.

To understand CSI, consider a MIMO system with n receiver antennas and m transmitter antennas. The received signal can be expressed as a signal vector Y of size n, defined at a time t as:

$$Y_t = HX_t + \eta_t \tag{3.1}$$

where H is a complex matrix of dimension $n \times m$ that represents the wireless channel, and it is what is known as CSI, X_t is the transmitted signal vector of size m, plus some noise vector η_t. Thus, the goal is to estimate H, which contains information about the multipath fading effects in the channel, based on knowledge of Y_t and X_t, as this will allow the system to adapt to current channel conditions, thereby enhancing communication (Biguesh & Gershman, 2006).

CSI is estimated based on predefined symbols, known as Long Training Field (LTF) symbols, sent by the transmitter in each PHY frame preamble. As Wi-Fi transmits using Orthogonal Frequency Division Multiplexing (OFDM), the

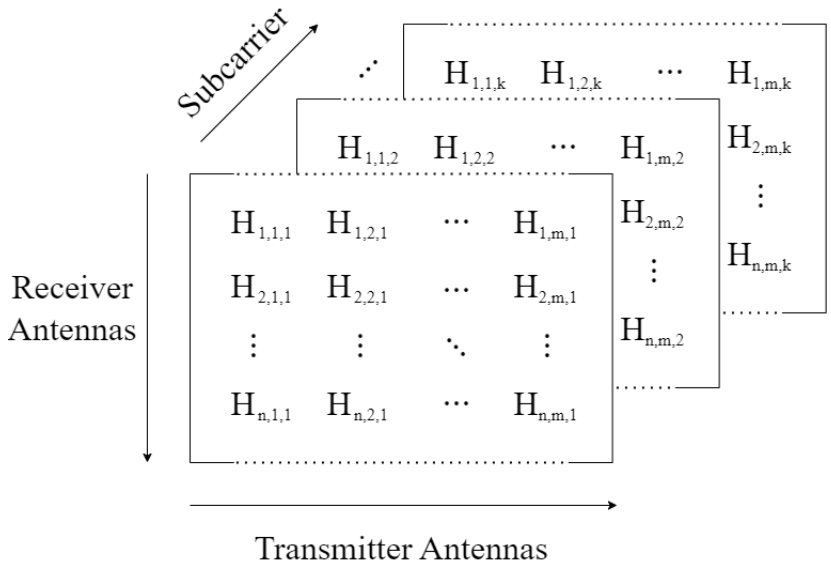

Figure 3.1: CSI matrix representation.

transmission of the signal is achieved with subcarriers orthogonal with each other and thus LTFs are transmitted across all subcarriers, resulting in H being a collection of matrices, where each matrix corresponds to one subcarrier (Ma *et al.*, 2019) as depicted in Fig. 3.1. Each matrix element of H for a subcarrier f for a stream between antenna a and antenna b for a time t is given by:

$$H_{a,b,f}(t) = \alpha_{a,b,f}(f)e^{-j\theta_{a,b,f}(t)} \tag{3.2}$$

where $\alpha_{a,b}$ is the amplitude that represents attenuation and $e^{-j\theta_{a,b}(f)}$ the phase shift due to multipath propagation.

In Wi-Fi 802.11n, a 20 MHz channel consists of 64 subcarriers, which are divided into 52 user data subcarriers, 4 pilot subcarriers used for synchronization and correction, and 8 null subcarriers that are used as guard bands for adjacent channels, each with a spacing of 312.5 kHz (Van Nee *et al.*, 2006). Fig. 3.2 shows the distribution of these subcarriers in the channel.

To conclude, CSI can be defined as a collection of matrices in which the number of matrices is given by the number of subcarriers, and the size of each

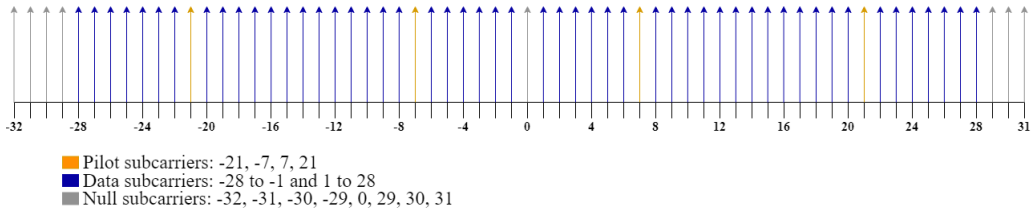

Figure 3.2: Indexes of subcarriers allocated in a 20 MHz channel for 802.11n.

depends on the number of receiving and transmitting antennas. As multipath effects are present in CSI, it also contains information on how the Wi-Fi signal is affected by its surroundings. Hence, it is possible to infer that the presence of people an their actions will result in variations perceived in CSI, leading to what is now known as *Wi-Fi sensing* and the development of *Wi-Fi sensing systems*.

3.2 Functionality of Wi-Fi sensing systems

According to the performed SLR, the functioning of every Wi-Fi sensing system can be divided at least into the following four stages:

- *CSI Collection Stage*: involves using CSI collection tools, which consist of hardware and software that enable the collection of CSI from Wi-Fi devices.
- *CSI Preprocessing Stage*: this stage may involve one or more techniques for removing noise from CSI, reducing computational complexity, signal transforms, and extracting features that can be used for further detection.
- *Detection Stage*: involves using Machine Learning, Deep Learning, or signal analysis techniques to identify patterns, perform classification, and estimate values from CSI that enable recognition or monitoring.
- *Application Stage*: this final stage relates the results obtained from the Detection stage to a specific Wi-Fi sensing application, e.g., associating a label obtained from a classifier with a corresponding human activity.

A summary of these four stages is depicted in Fig. 3.3, while a detailed description of each stage will be provided in the following sections.

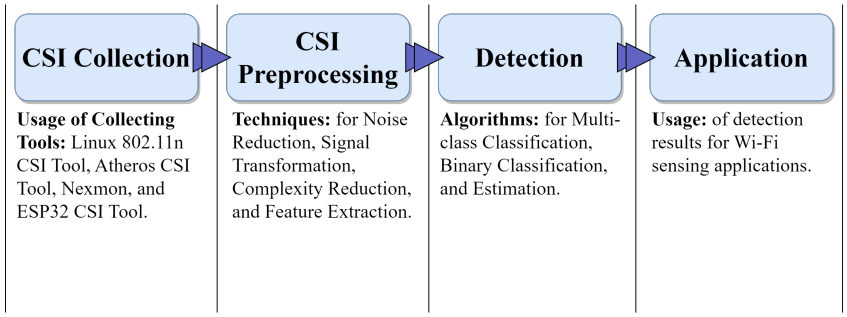

Figure 3.3: Functioning stages of Wi-Fi sensing systems.

3.3 CSI Collection Stage

As mentioned before, CSI is estimated in the PHY of Wi-Fi devices, allowing them to optimize wireless communication based on current channel conditions. However, most Wi-Fi devices do not have an interface for collecting CSI, as this was not considered in their design following current Wi-Fi standards, such as 802.11b/g/n/ac. Therefore, CSI cannot be collected directly for use in Wi-Fi sensing applications. This fact led to the development of Wi-Fi CSI collecting tools that have been widely used in the Wi-Fi sensing state-of-the-art: the *Linux 802.11n CSI Tool*, the *Atheros CSI Tool*, *Nexmon*, and the *ESP32 CSI Tool*.

3.3.1 The Linux 802.11n CSI Tool

The Linux 802.11n CSI Tool, developed by Halperin *et al.* (2011), uses explicitly the Intel 5300 Network Interface Card (NIC), which works with the 802.11n Wi-Fi standard, with a custom firmware applied to enable an Intel debug mode for collecting CSI for each received 802.11n packet from 30 groups of subcarriers of the 56 not null subcarriers in a 20 MHz channel. CSI is then sent to the kernel driver that passes CSI to a user-space program. Each CSI matrix entry reported has a signed 8-bit resolution for the complex number that can be converted to amplitude attenuation and phase shift between a single Tx-Rx antenna pair.

The Linux 802.11n CSI Tool is available in a well-known repository, along with all the necessary programs and scripts to collect and analyze CSI. It is important to highlight that this tool only works by using the Intel 5300 NIC and requires

Linux-based systems with a kernel version between 3.2 and 4.2, e.g., Ubuntu 12.04 and Ubuntu 14.04.4, released in 2012 and 2015, respectively.

3.3.2 The Atheros CSI Tool

Unlike the Linux 802.11n CSI Tool, which is restricted to the use of a single Wi-Fi NIC, the Atheros CSI Tool, developed by Xie *et al.* (2015), allows the use of Atheros 802.11n NICs. The authors have tested it on the AR9580, AR9590, AR9344, and QCA9558 models.

The Atheros CSI Tool can be installed on Ubuntu systems without requiring custom firmware for the network device, only by installing a modified version of the 4.1.10 Linux Kernel. Additionally, it is compatible with embedded devices running the Linux-based OpenWRT system.

Each CSI matrix entry has a 10-bit resolution and reports from up to 56 subcarriers for a 20 MHz channel and 114 subcarriers for a 40 MHz, depending on the desired configuration of the collecting system without subcarrier grouping.

3.3.3 Nexmon

Schulz *et al.* (2017) developed the Nexmon CSI extractor, a C-based open-source firmware patching framework for Broadcom and Cypress Wi-Fi chips supporting Wi-Fi 802.11a/g/n/ac. This tool enables the collection of CSI from smartphones, routers, and single-board computers, reporting information from up to 256 subcarriers within an 80 MHz channel, with resolutions ranging from 10 to 14 bits, depending on the used chip.

Furthermore, Nexmon was ported to be compatible with the Broadcom 43684 chipset, supporting the high-efficiency (HE) PHY introduced in 802.11ax, which features a bandwidth of up to 160 MHz and 4x4 MIMO, where each channel is divided into 2048 subcarriers. This milestone allows obtaining information from 32768 subcarriers in a 4x4 MIMO system (Gringoli *et al.*, 2021).

3.3.4 The ESP32 CSI Tool

The ESP32 CSI Toolkit, developed by Hernandez & Bulut (2020), uses an ESP32 microcontroller that implements the ESP Wi-Fi API for collecting CSI. These

devices can act as both an AP or a station under TCP protocol and 802.11b/g/n standards in 20 and 40 MHz channels with support to receiving frames with legacy long training field (LLTF), high throughput LTF (HT-LTF), and space-time block code HT-LTF (STBC-HT-LTF) in the preamble. The ESP Wi-Fi API enables the ESP32 CSI Toolkit to collect the real and imaginary parts of CSI with an 8-bit resolution for each of the 64 subcarriers in a 20 MHz channel or 128 subcarriers in a 40 MHz channel.

This tool operates in two modes: *Active CSI Collection* and *Passive CSI Collection*. In active mode, a communication link is established between an ESP32 working as an access point (AP) and an ESP32 working as a station (STA). Moreover, in passive mode, the ESP32 listens passively to packets on a given channel, reporting CSI for each packet it perceives.

This tool utilizes ESP32 microcontrollers, offering a flexible, low-cost, and easy-to-deploy solution for Wi-Fi sensing applications.

3.4 CSI Preprocessing Stage

The preprocessing stage is a recommended step for extracting relevant information from CSI for Wi-Fi sensing applications. This section will focus on techniques commonly used in this stage, according to the classification proposed in Section 2.4 of Chapter 2: *Noise Reduction, Signal Transform, Dimensionality Reduction* and *Feature Extraction*.

3.4.1 Noise Reduction

Wi-Fi CSI captures multipath effects, including those from noise sources that negatively impact the performance of Wi-Fi sensing systems, e.g., other Wi-Fi signals or devices and objects or people moving within the signal range. Thus, Noise Reduction techniques are used to mitigate the effects of these noise sources, improving the quality of data and, therefore, the performance of Wi-Fi sensing systems.

Low-pass Filter

An ideal low-pass filter is a filter that passes signals with frequencies below a cut-off frequency f_c while eliminating frequencies above this value. The desired band of frequencies, i.e., those below f_c, is called the pass-band, while those above f_c receive the name of the stop-band. However, in a real application, the low-pass filter attenuates signals with frequencies above f_c, presenting a transition band whose slope depends on the filter design, converting f_c into a point in the slope in which the attenuation will be of -3 dB concerning the magnitude presented in the pass-band region (Floyd, 2008).

Some authors prefer instead the use of a *Moving Average Filter*, which can be seen as a low-pass filter that takes a window of M samples from a signal x to produce an output signal y at time n (Chen & Chen, 2003), defined as:

$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n-k] \tag{3.3}$$

Low-pass filters are commonly applied to the CSI amplitude and phase to attenuate frequencies above the range of interest, depending on the Wi-Fi sensing application being implemented.

Multipath Removal

As mentioned before, CSI captures multipath propagation effects, including propagation delay. Since CSI represents the channel's frequency response, applying an Inverse Fourier Transform yields the channel's Power Delay Profile (PDP). The PDP profiles the multipath arrivals of a signal as a function of propagation delay (Xie *et al.*, 2019). Thus, multipath removal eliminates reflected paths captured in CSI based on the PDP.

For example, Zhang *et al.* (2017) performed multipath removal by analyzing the propagation delays in an indoor environment. They determined that reflected paths with propagation delays exceeding 500 ns should be removed, as experiments showed this to be the maximum delay caused by reflected paths in such settings.

Discrete Wavelet Transform

The wavelet transform consists of decomposing an input signal from functions derived from a selected waveform, known as *wavelet*, which has finite duration and zero mean. The selected wavelet is referred to as the mother wavelet ψ.

The derivatives from ψ are obtained by scaling and shifting ψ, defined as:

$$\psi_{(a,b)}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \tag{3.4}$$

where a refers to a scaling factor and b to a shift factor. In DWT, a is discretized along the sequence 2^j, where j is a positive integer value (Daubechies, 1992). When computed, the DWT is achieved through multilevel decomposition, where the input signal is passed in parallel through low-pass and high-pass filters defined according to ψ. The output from the high-pass filter is referred to as *detail coefficients* and maintains the high-frequency components of the input signal. Meanwhile, *approximation coefficients* obtained from the low-pass filters keep the low-frequency components from the input signal.

The decomposition is performed iteratively, being repeated according to a defined level. At each level, the approximation coefficients obtained in the previous level are filtered by the same high-pass and low-pass filters (Tzanetakis *et al.*, 2001).

In Wi-Fi sensing, DWT is used as a filtering technique, where the approximation coefficients are taken as preprocessed CSI, as human-related movements can be found in the low-frequency components.

Hampel Identifier

The Hampel identifier is one of the most common preprocessing techniques used for Wi-Fi sensing, as it identifies and replaces outliers from the CSI amplitude or phase.

The Hampel identifier considers as an outlier an entry x_i from a fixed-size sliding window x of a time series and replaces it with the median m of x if the following condition is met:

$$|x_i - m| > tS \tag{3.5}$$

where $S = 1.4826 \times median(|x - m|)$, being equal to the standard deviation σ for normally distributed data, and t is a configurable threshold value for outlier detection (Pearson, 2002).

Savitzky Golay Filter

Savitzky-Golay (SG) filters are lowpass filters that smooth data based on local least-squares polynomial approximation, reducing noise while maintaining the shape and height of waveform peaks. Suppose having a signal f represented as follows:

$$f(-n)..., f(-2), f(-1), f(0), f(1), f(2), ...f(n) \tag{3.6}$$

for each signal time step i, $f(i)$ is replaced by the value $\hat{f}(i)$, obtained by fitting a polynomial to f on the time steps between $i - k$ and $i + k$. This example works well for understanding the main idea of SG filters; however, in practice, least-squares polynomial fitting is typically achieved through convolution.

The main task of this smoothing method is to find coefficients $a_0, ..., a_d$ such that the polynomial of degree d:

$$p(x) = \sum_{i=0}^{d} a_i x^i \tag{3.7}$$

minimizes the mean-squared approximation error for the group of input time steps centered on x:

$$\sum_{x=-k}^{k} (p(x) - f(x))^2 \tag{3.8}$$

being k the half-width of the approximation interval (also called window length). From the previous error equation, a matrix A of dimensions $(2k + 1)X(d + 1)$, called the design matrix, can be obtained. The matrix product Aa, where $a = [a_0, a_1, ..., a_d]^T$, gives a column vector of values for the polynomial p. Being $y = [f(-k), ..., f(k)]^T$ the vector of input samples, the problem is reduced to find a vector a that minimizes $(Aa - y)^2$, which solution can be written as:

$$a = (A^T A)^{-1} A^T y = Hy \tag{3.9}$$

The H matrix depends only on k and d and is independent of the input samples; therefore, the same coefficients will be obtained from each approximation interval of input samples, and consequently, least-squares smoothing can be seen as a shift-invariant discrete convolution process (Schafer, 2011).

3.4.2 Signal Transformation

Fourier Transform

The Fourier Transform is a tool that decomposes a signal in its frequency components for further analysis in the frequency domain.

Fourier series establish that a periodic signal $x(t)$ can be approximated by the sum of sinusoidal functions at different frequencies:

$$x(t) = a_o + \sum_{k=1}^{\infty} [a_k cos(kw_o t) + b_k sin(kw_o t)] \tag{3.10}$$

where a_o, a_k, and b_k are real numbers and w_o the fundamental frequency. The Fourier series represented in Eq. (3.10) can be expressed in its complex exponential form given by:

$$x(t) = \sum_{k=-\infty}^{\infty} c_k e^{jw_o kt} dt \tag{3.11}$$

The Fourier Transform is a generalization of the Fourier series, as it extends its applicability to non-periodic signals. For a given signal $x(t)$, the Fourier Transform $X(w)$ of $x(t)$ is defined for an angular frequency w by the following integral:

$$X(w) = \int_{-\infty}^{\infty} x(t) e^{-jwt} \tag{3.12}$$

where $X(w)$ is a complex value that includes both the magnitude and phase for the corresponding frequency component w related to $x(t)$.

For a discrete signal $x[n]$ with a value N such that $x[n] = 0$ for every $n \leq -N$ and $n \geq N$. Under these circumstances, a discrete version of the Fourier

Transform, named Discrete Fourier Transform (DFT), can be used. The DFT can be defined as follows:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi kn}{N}} \quad , \quad k = 0, 1, ..., N-1 \tag{3.13}$$

Since $x[n]$ might take complex values in the DFT, calculating a single value k involves N complex multiplications and $N-1$ complex sums. This means that many arithmetic operations are performed for high values of N, leading to high computational resource demands when computing the DFT.

The Fast Fourier Transform (FFT) refers to a series of algorithms designed to optimize the computation of the DFT. These algorithms decompose the DFT of a signal $x[n]$ of length N into smaller DFTs that are further combined, taking advantage of the symmetry and periodic properties of the complex exponentials. (Kamen & Heck, 2007; Oppenheim & Schafer, 2021).

Power Spectral Density

To understand the Power Spectral Density (PSD), consider the Fourier transform equation from Eq. (3.12). From this equation, the total energy for a finite signal $x(t)$ can be determined as the area under the curve of $|X(w)|^2$, such that:

$$\int_{-\infty}^{\infty} |x(t)^2| dt = \int_{-\infty}^{\infty} |X(w)|^2 dw \tag{3.14}$$

Thus, the energy spectral density of a signal x(t) is defined as follows:

$$E(w) = |X(w)|^2 \tag{3.15}$$

which is measured as energy per unit frequency.

Otherwise, if $x(t)$ is not a finite signal with $E(w)$ tending to infinity, the PSD $S(w)$ measure is used, with measurement unit of power per unit frequency. The PSD can be defined by the following equation:

$$S(w) = \lim_{T\to\infty} \left(\frac{E(w)}{T} \right) = \lim_{T\to\infty} \left(\frac{|X(w)|^2}{T} \right) \tag{3.16}$$

where T is the length of the signal (Youngworth *et al.*, 2005).

3.4.3 Dimensionality Reduction

Wi-Fi sensing applications often seek to reduce the amount of information to be processed or to retain only relevant information, as CSI subcarriers may capture noise, thereby improving the system's performance. In the state-of-the-art, there are two main algorithms for reducing or selecting information for Wi-Fi sensing: Principal Component Analysis (PCA) for obtaining meaningful information from subcarriers and the Variance-based Subcarrier Selection algorithm, which aims to select k subcarriers with relevant information according to the sensing application.

Principal Component Analysis

Principal Component Analysis (PCA) is a statistical technique used to reduce the dimensionality of a dataset by finding new variables, which are linear functions of the original variables of the dataset, that maximize variance and that are uncorrelated with each other (Jolliffe & Cadima, 2016).

Given a dataset of dimensions $n \times m$, where n is the number of entries or observations and m, the number of variables also known as features, can also be seen as a data matrix X. The main goal is to find a linear combination of the columns of X given by:

$$Xa = \sum_{j=1}^{m} a_j x_j \tag{3.17}$$

where a is a vector of length m. The variance of any linear combination is given by:

$$a^T S a = \frac{\sum_{i=1}^{n} (x_i \bar{x})^2}{n} \tag{3.18}$$

where S is the covariance matrix of the data matrix and a^T is the transpose of a. Therefore, in order to find the linear combination that maximizes the variance, it is necessary to find the a vector that maximizes $a^T S a$. Vector a is an eigenvector with its corresponding eigenvalue λ_k of the covariance matrix S where the largest eigenvalue and its respective eigenvector need to be found. S matrix, like other symmetric matrices, contains n eigenvalues with their respective eigenvectors, which are the solution to find k linear combinations:

$$X_{ak} = \sum_{j=1}^{m} a_{kj} x_j \tag{3.19}$$

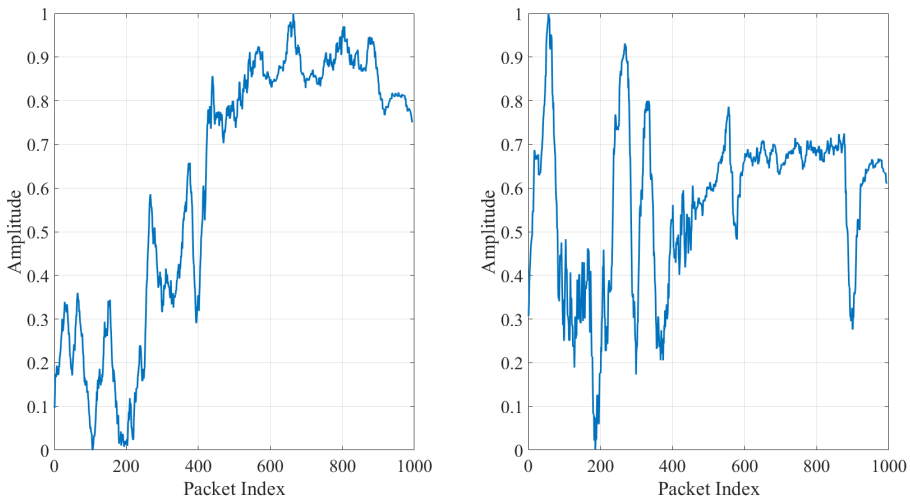

Figure 3.4: Comparison between a subcarrier with the highest variance value (left) and with the lowest variance value (right). Both subcarriers correspond to the same sample where a fall was captured during the data collection of this work.

where the linear combinations X_{ak} are called *principal components*.

Variance-based Subcarrier Selection

It is a straightforward yet effective subcarrier selection method. Liu *et al.* (2015) observed that subcarriers with higher variance are more sensitive to minute movements, e.g., chest movements caused by breathing. Therefore, they decided to use the variance of CSI amplitudes in a moving time window to measure each subcarrier's sensitivity to minute movements. Once the variance of each subcarrier is obtained, the N subcarriers with the highest variance values can be selected to achieve a sensing task.

Fig. 3.4 compares two subcarriers from a sample recording of a fall activity. The fall was expected to occur around the 5-second mark (approximately at a packet index of 250). It can be observed that the subcarrier with the highest variance exhibits a pronounced change in amplitude values before and after the 5-second mark, making the detection of the fall distinguishable. In contrast, the subcarrier with the lowest variance shows only marginal amplitude variations around the same time, rendering the fall far less observable. This demonstrates

how subcarrier selection based on variance can significantly impact the performance of activity recognition in Wi-Fi-based sensing applications.

It is important to note that subcarriers have different central frequencies and wavelengths, leading to frequency-selective fading in CSI measurements due to multi-path propagation. As a result, the amplitude ranges of CSI values vary across subcarriers. Directly comparing variances (or other statistics) between subcarriers with different amplitude ranges can introduce scale-related biases. To address this, normalization, such as min-max scaling, should be applied to each subcarrier individually, rescaling its values based on its range. This ensures that all subcarriers are on a common scale before comparing variance or proceeding with further analysis.

3.5 Detection Stage

The Detection stage involves tasks such as binary classification, multi-class classification, or estimation. These tasks can be achieved using Machine Learning algorithms, Deep Learning models, or signal processing techniques. The selection of the approach depends on the type of application for which the task will be implemented. For instance, HAR applications may utilize Machine Learning algorithms and Deep Learning models to identify the activity being performed by a person from a predefined set of recorded activities, i.e., a multi-class classification task. On the other hand, Presence detection applications rely on algorithms and models designed for binary classification. Meanwhile, applications focused on vital sign monitoring often make use of signal processing techniques to estimate breathing rates, such as peak detection algorithms.

This section will focus on describing the fundamentals of two network architectures that have demonstrated an outstanding performance for HAR, as well as for other Wi-Fi sensing applications: the *Convolutional Neural Networks* and the *Long Short-Term Memory* network architectures.

3.5.1 Convolutional Neural Networks

The discretized convolution operation can be seen as a linear operation denoted as:

$$s(t) = \sum_{\tau=-\infty}^{\infty} x(\tau)w(t-\tau) \tag{3.20}$$

where $x(t)$ is referred to as input and $w(t)$ the kernel, while the output $s(t)$ is referred to as a feature map. In Machine Learning, $x(t)$, $w(t)$, and thus $s(t)$ are multidimensional arrays referred to as tensors. The convolution operation is commonly applied to images that have at least two dimensions, leading to the need to perform convolution over more than one axis at a time. Thus, for a two-dimensional image I and a two-dimensional kernel K, the convolution operation is expressed as:

$$C(i,j) = \sum_{m}\sum_{n} I(i+m, j+n)K(m,n) \tag{3.21}$$

It is worth highlighting that for practical implementation in Machine Learning, the operation being performed is named cross-correlation, which is the same as convolution but without kernel flipping, as the goal of the algorithms is to learn the appropriate kernel values in the appropriate place (Goodfellow *et al.*, 2016).

The convolution operation is the first of three stages, which is known as the convolutional layer. These three stages are involved in the functioning of a Convolutional Neural Network (CNN). The convolutional layer is used to produce a set of linear activations. The second stage consists of running each linear activation through a nonlinear activation function, while in the third stage, a pooling function is applied at the output to replace the output with a summary statistic of nearby outputs of the first two stages, e.g., *max pooling* which replaces the output with the maximum output within a rectangular neighborhood for obtaining a representation invariant to small translations of the input (Goodfellow *et al.*, 2016).

CNNs are commonly used for working with images, i.e., two-dimensional data. These can be referred to as 2D-CNNs. However, when working with sequential data, such as signals, the use of 2D-CNNs would not be adequate. In these

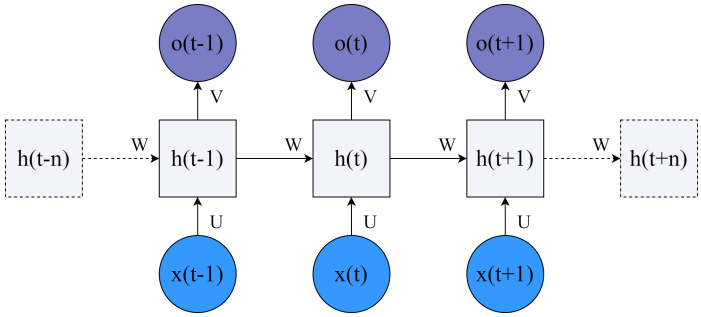

Figure 3.5: RNN computational graph representation.

cases, 1D-CNNs are a preferable alternative for sequential or linear data, i.e., one-dimensional data, as their computational complexity is significantly lower compared to 2D CNNs, making them suitable for real-time and hardware-limited devices (Kiranyaz *et al.*, 2021). The main difference between 1D and 2D CNNs is that 1D-CNNs replace two-dimensional arrays with one-dimensional arrays for both kernels and feature maps.

3.5.2 Long Short-Term Memory

To understand the functioning of Long Short-Term Memory (LSTM) networks, first, it is important to know the basis of Recurrent Neural Networks (RNNs).

RNNs are a type of neural network for processing sequential data, where the output at time t is a function of the previous output at time $t-1$. This can be expressed by the equation

$$\mathbf{h}(t) = f\left(\mathbf{h}(t-1); \mathbf{x}(t); \theta\right) \tag{3.22}$$

where $\mathbf{h}^{(\mathbf{t})}$ represents the state of the network and $\mathbf{x}^{(t)}$ the input at time t with learnable network parameters θ. These models have the particularity that, as they are specified in terms of a transition function f, their input size will always be the same, regardless of the sequence length. The same transition function with the same parameters can be used at every time step.

An RNN with input $\mathbf{x}$, state $\mathbf{h}$, and output $\mathbf{o}$ can be represented as a computational graph as shown in Fig. 3.5. For this model, the RNN has input-to-hidden connections parametrized by $\mathbf{U}$, hidden-to-hidden recurrent connections

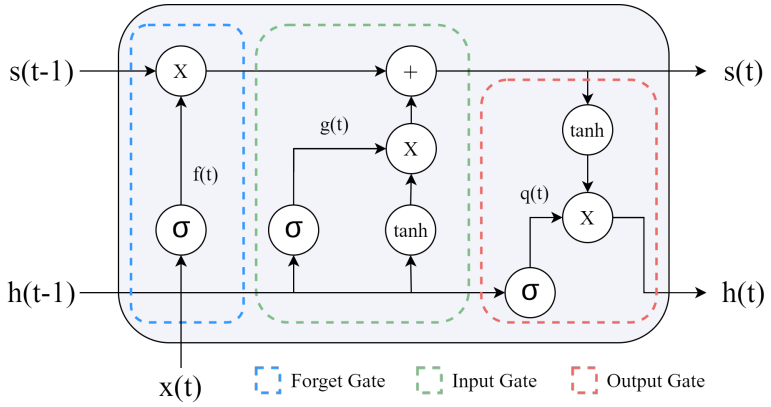

Figure 3.6: Graphical representation of an LSTM cell.

parametrized by $\mathbf{W}$, and hidden-to-output connections parametrized by $\mathbf{V}$. Thus, to map each timestep t of input $\mathbf{x}$ to output $\mathbf{o}$ first, the state of the network is found by the equation:

$$\mathbf{h}^(t) = tanh(\mathbf{b} + \mathbf{W}\mathbf{h}(t-1) + \mathbf{U}\mathbf{x}(t)) \tag{3.23}$$

where $\mathbf{b}$ is a bias vector. Then, output $o(t)$ can be obtained as:

$$\mathbf{o}(t) = \mathbf{c} + \mathbf{V}\mathbf{h}(t) \tag{3.24}$$

where $\mathbf{c}$ is a second bias vector (Goodfellow *et al.*, 2016).

Although RNNs can learn long-term dependencies, they are affected by the vanishing and exploding gradient problems that arise from the exponentially smaller weights given to long-term interactions.

A solution to these gradient problems is proposed by Hochreiter & Schmidhuber (1997), where self-loops are introduced to create paths that allow the gradient to flow for long durations. The weight of this self-loop is controlled by another hidden unit, known as the forget gate (Gers *et al.*, 1999). This type of gated RNN is known as LSTM.

LSTMs have what are called cells represented in Fig. 3.6, which have an internal recurrence in addition to the outer recurrence of RNNs. The cell state unit $s_i(t)$ weight is controlled with a forget gate unit $f_i(t)$ defined by setting its

value via sigmoid function σ

$$f_i(t) = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j(t) + \sum_j W_{i,j}^f h_j(t-1)\right) \tag{3.25}$$

for an input vector $\mathbf{x}(t)$, with a current hidden layer vector $\mathbf{h}(t)$, and $\mathbf{b}^f$, $\mathbf{U}^f$, and $\mathbf{W}^f$ being the biases, input weights, and recurrent weights for the forget gate, respectively. Thus, the cell state is updated by the equation

$$s_i(t) = f_i(t)s_i(t-1) + g_i(t)tanh\left(b_i + \sum_j U_{i,j}x_j(t) + \sum_j W_{i,j}h_j(t-1)\right) \tag{3.26}$$

where $tanh$ function is being applied to what is known as the input modulation gate, often included as part of the input gate, parameters $\mathbf{b}$, $\mathbf{U}$, and $\mathbf{W}$ are the biases, input weights, and recurrent weights into the cell. Meanwhile, the input gate unit $g_i(t)$ is defined as

$$g_i(t) = \sigma\left(b_i^g + \sum_j U_{i,j}^g x_j(t) + \sum_j W_{i,j}^g h_j(t-1)\right) \tag{3.27}$$

The output $h_i(t)$ of the cell is computed based on the cell state $s_i(t)$ and the output gate, referred to as $q_i(t)$. The latter, with parameters $\mathbf{b}^o$, $\mathbf{U}^o$, and $\mathbf{W}^o$, is defined by the equation

$$q_i(t) = \sigma\left(b_i^o + \sum_j U_{i,j}^o x_j(t) + \sum_j W_{i,j}^o h_j(t-1)\right) \tag{3.28}$$

resulting in $h_i(t)$ being defined as

$$h_i(t) = tanh(s_i(t))q_i(t) \tag{3.29}$$

The input, forget, and output gates can be trained to learn what information to store in memory, how long to store it, and when to read it, respectively (Gers *et al.*, 2003; Goodfellow *et al.*, 2016).

Chapter 4

Human Activity Recognition on an ESP32

4.1 Introduction

Following the review presented in Chapter 2, this chapter addresses a critical gap identified in the field of Wi-Fi CSI-based sensing: the lack of use of embedded devices for processing or collecting CSI, specifically, the ESP32 devices. These are low-cost microcontrollers that provide Application Programming Interfaces (APIs) for collecting CSI, which have been relegated only to collecting devices without exploiting their computational capabilities for IoT solutions. For instance, works that leverage this device as a CSI collection tool transmit CSI estimations to another device with more computational power, such as a desktop computer, for further analysis and processing. One example of usage of the ESP32 only as a collecting device is presented by Natarajan *et al.* (2024), where the CSI collected was sent to a computer for filtering the CSI through a Wavelet-based denoising technique. Then, features were extracted to perform HAR of four activities: empty room (no activity), walking, sitting, and standing, for smart LED lighting control. By using an ensemble classifier comprising a Random Forest classifier, a Gradient Boosting classifier, and an Extreme Gradient Boosting classifier, the authors obtained a mean cross-validation accuracy of 83.39%. Similarly, Alzaabi *et al.* (2024) developed a system that sent CSI collected with ESP32 devices to a computer that processes the data through a Python script. This script

applies a Fourier method for interpolation and downsampling, addressing the issue of unstable packet rates inherent in the device. Additionally, discrete wavelet transform and principal component analysis are employed to extract relevant information related to a person's breathing. The breathing rate was estimated based on power spectral density analysis with a root mean square error of 1.04 breaths per minute.

Moreover, other works have used single-board computers (SBCs) to preprocess CSI and classify the preprocessed samples using Deep Learning models. For instance, Tong *et al.* (2024) used the ESP32 CSI Tool for CSI collection, sending the estimations to a Jetson Nano for preprocessing data using a Hampel identifier and a Savitzky-Golay filter, and for classifying using a CNN-based model. With this approach, the authors obtained that following this methodology, an accuracy of 95.57% can be obtained for recognizing between the activities of left leg raise, right arm raise, and stretching out. Likewise, Touhiduzzaman *et al.* (2024) developed a physical rehabilitation tracking system for hand-based exercises. The CSI collected with an ESP32 was sent to a Raspberry Pi for preprocessing, classification, and exercise counting, leveraging a Deep Learning model consisting of only Dense and Dropout layers, achieving an overall accuracy of 91.22% for recognizing finger and wrist movements. Solely using the ESP32 for collecting and processing, Burimas *et al.* (2024) demonstrated the capabilities of the ESP32 for not only collecting CSI using the ESP32 CSI Tool, but also for processing to perform the detection task. By employing a Hampel identifier and a low-pass filter, the authors mitigated the effects of noise in CSI. The authors performed a linear interpolation to treat the unstable packet rate that ESP32 devices have. Meanwhile, breathing rate was estimated through a peak detection algorithm also functioning in the device, achieving a mean absolute deviation of 2.7 breaths per minute.

Differing from these works, this chapter assesses the ESP32's capabilities and limitations for both collecting CSI and performing predictions using a Deep Learning model, unlike related works that only utilize the ESP32 for CSI collection, rather than exploiting its capabilities to also achieve the detection task. This evaluation follows an adaptation of the CRISP-DM methodology in the form of

a proof-of-concept that defines a data pipeline for further design and deployment of Wi-Fi sensing applications for embedded devices.

This chapter is organized as follows: Section 4.2 introduces the data pipeline proposal for developing Edge AI-based applications, referred to as the *Embedded CRISP-DM*, which reuses the steps of the CRISP-DM methodology. Section 4.3 starts with describing the proposed methodology by describing the CSI data collection and preprocessing processes. In Section 4.4, the tasks of model definition, training, and optimization are described in detail, including the tools, APIs, configurations, and hardware used. Next, Section 4.5 presents the evaluation and deployment process of the defined model in an ESP32. Finally, conclusions are drawn in Section 4.7.

4.2 Development of a Human Activity Recognition Model for Embedded Devices

Embedded devices, such as those based on the ESP32, have hardware limitations compared to computers where DL models are typically trained, evaluated, and deployed. Table 4.1 presents the specifications for the ESP32-S3 Development Board Series, where it can be seen that memory is much more limited compared to the available memory in personal computers, which is in the order of gigabytes. As DL models are loaded into memory for execution, it is necessary to consider these memory constraints and processor throughput when deploying models on these devices.

Traditional DL and ML-based projects can be guided by adaptations of the Cross Industry Standard Process for Data Mining, better known as the CRISP-DM (Wirth & Hipp, 2000) methodology for data mining projects. The CRISP-DM framework consists of the following phases:

1. **Business Understanding:** This phase focuses on comprehending the project, its objectives, and requirements, for further problem definition.

2. **Data Understanding:** This phase starts by collecting data, which is further explored to identify potential data issues and discover insights or features from data that help formulate hypotheses.

3. **Data Preparation:** In this phase, tasks such as data cleansing, feature extraction, and data transformation are performed to construct datasets from raw data.

4. **Modeling:** During this phase, appropriate modeling techniques are selected and applied. From an ML or DL perspective, model tuning and training take place in this phase.

5. **Evaluation:** This phase involves evaluating the model to determine whether it meets the project objectives.

6. **Deployment:** The final phase involves organizing and presenting the gained knowledge. Can be from a simple report to a repeatable process that uses the generated model.

Although CRISP-DM is effective for data mining projects, working with embedded devices raises issues that need to be addressed due to the hardware constraints. Thus, additional tasks must be performed in each phase to achieve correct functioning in these devices. Fig. 4.1 provides an overview of the additional tasks to be performed in each phase.

Table 4.1: ESP32-S3 Series Features.

Specification	Description
CPU	Xtensa Dual-Core 32-bit LX7, with single-precision FPU
Frequency	Up to 240 MHz
Internal Memory	512 kB SRAM
Flash	Up to 16 MB Quad SPI
External Memory	Up to 16 MB Octal SPI

4.2.1 The Embedded CRISP-DM

Business Understanding

The Business Understanding phase primarily focuses on comprehending the project objectives and requirements before developing a project plan. In the context of embedded projects, it is essential to select an appropriate embedded device based

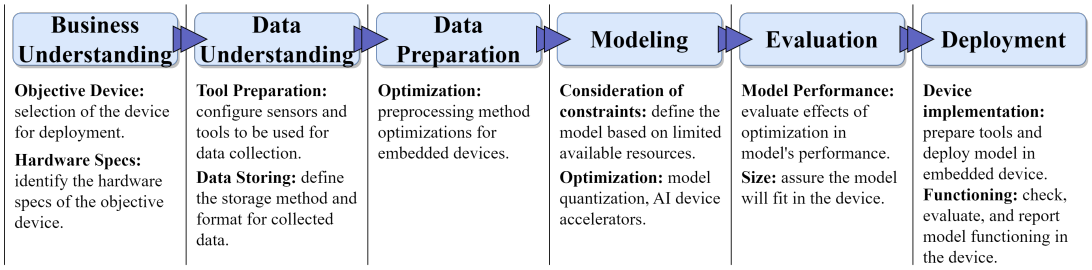

Figure 4.1: Overview of the Embedded CRISP-DM tasks.

on these established project requirements and objectives. When choosing a device, factors such as processing capabilities, clock speed, communication interfaces, and available memory must be carefully considered. Furthermore, certain solutions, like IoT applications, may necessitate an embedded device with integrated Wi-Fi, while deployments in extreme temperatures will require a device that can operate effectively across a wider temperature range.

Therefore, device selection and a thorough understanding of hardware specifications are tasks that must be performed in this phase. Development boards were used for prototyping and evaluation for this research. However, if custom PCB prototypes are required, this phase must also include selecting the board components, such as sensors, communication interfaces, and actuators, based on the project's requirements and objectives, as well as managing the manufacturing of the custom prototype.

Data Understanding

The CRISP-DM documentation states that the Data Understanding phase starts with data collection. However, to collect data with embedded devices or custom prototypes, the device's sensors or tools must be configured for collecting data. This may involve calibrating the sensors or setting up the data collection tools.

Once tools and sensors are set, another important step is to define and prepare the storage device, method, and format for the collected data. Since embedded devices typically have smaller internal memory capacities than computers, collected data must be saved in an external storage device, e.g., a memory card. Collected data can also be transmitted to a computer, requiring defining, configuring, and implementing a communication interface at both ends.

Data Preparation

In the Data Preparation phase, it must be considered that the tasks of data cleansing, feature extraction, and data transformation can be separated into two different stages: the *offline stage* and the *online stage*.

The offline stage refers to the project stage where model selection, tuning, training, and evaluation can be performed on a computer, and so can data preprocessing. The processing time and computational resources used during the offline stage will differ from those in the online stage, when the model is deployed and executed on the embedded device.

If preprocessing methods are not selected or optimized for these embedded devices, the time required for preprocessing may make the embedded solution unfeasible for working in the online stage, especially if real-time performance is necessary, depending on the project's requirements and objectives. Complex preprocessing methods can block other tasks being performed on the device, such as data collecting and model execution. Therefore, choosing and optimizing preprocessing methods with these considerations in mind is essential.

Modeling

In the Modeling phase for embedded devices, device constraints must be considered when selecting or defining the DL model to be trained. The model's parameters, weights, inputs, and outputs must all be allocated in memory to be executed. Hence, the denser the model, the more memory it would require to be allocated. For example, the well-known VGG16 network architecture (Simonyan & Zisserman, 2015) requires allocating 528 MB of memory to be used. Allocating this amount of memory in an embedded device such as an ESP32-S3 would be impossible, as this device can only use up to 32 MB of virtual addresses for external PSRAM. Consequently, the model to be implemented must be selected considering the device's available memory.

Additionally, it is recommended to consider quantizing the model. This transforms its inputs, outputs, and weights from 32-bit floating-point values into 8-bit integer values. This transformation reduces the model's size and latency, as inte-

ger operations are faster to perform than floating-point operations, at the expense of a slight decrease in the model's accuracy.

Evaluation

In the Evaluation phase, it is important to evaluate the impact of quantization on the model's performance and size to check if it meets the project requirements and objectives and fits the device constraints. If not, it is always possible to return to the previous phase and select another model architecture or make adjustments to hyperparameters to boost its performance or reduce its size even more.

Deployment

The Deployment phase would require preparing the tools for deploying the model in the embedded device. APIs, such as TensorFlow Lite, may provide a workflow, tools, and libraries for converting the model to a format that can be implemented and for its use in the embedded device, as well as provide additional optimizations to a device-specific set of instructions to reduce the model's latency. Once implemented, an additional evaluation step must be performed to check if it is functioning in the device still meets the project requirements and objectives, as the prediction task might be executed concurrently with other tasks running on the device's processor.

The experiments presented in this work follow the proposed Embedded CRISP-DM, considering that previous sections and chapters encompass the Business Understanding phase by defining the research objectives, goals, and theoretical framework for Wi-Fi sensing applications. In addition, it is important to highlight that Data Understanding and Data Preparation phases will be referred to as *Data Collection and Preprocessing* to align with the lexicon used in Wi-Fi sensing applications, and *Evaluation* will include evaluations performed once the model was deployed in the device, as well as the deployment process in the ESP32.

4.3 Data Collection and Preprocessing

A modified version of the ESP32 CSI Tool (Hernandez & Bulut, 2020) was used to collect CSI for this proof-of-concept. Two ESP32-DevkitC development boards

were used, one as an active station (STA) sending packets at a rate of 50 packets per second, while the other functioned as an active access point (AP) positioned at two meters from the STA. The AP received the packets and sent the CSI measurements to a computer running a Python script, which collects CSI in intervals of 15 seconds. Fig. 4.2 illustrates the scenario for the CSI collection. Modifications applied consisted of changing the CSI measurements to binary representation, which reduce the amount of bytes that need to be transmitted through serial port. This will be further explained in Chapter 5.

During each 15-second interval, a volunteer was asked to perform one of the following activities:

- **Fall**: the volunteer stood next to a mattress placed on the floor between the AP and STA. After five seconds, the volunteer lies on the mattress at a fast pace, simulating a fall.
- **Run**: the volunteer jogged back and forth between P_1 and P_2 repeateadly during the recording time.
- **Walk**: the volunteer walked back and forth between P_1 and P_2 repeateadly during the recording time.

A total of 40 samples per activity were collected, resulting in 120 samples. Each sample was theoretically expected to have dimensions of 64×750. However, some samples did not record the theoretical 750 packets, referred to as timesteps, expected from configuring the tool at a packet rate of 50 packets per second for 15 seconds. Thus, every sample was truncated to 650 timesteps as it was observed that this was the minimum number of timesteps recorded in a sample.

CSI amplitude was calculated from each sample, per subcarrier $s \in \mathbb{C}$, which is defined as:

$$\alpha(t) = \sqrt{Im(s)^2 + Re(s)^2} \tag{4.1}$$

From the 40 amplitude values per activity, 28 were used to construct the training dataset, while the 12 remaining were used to construct the test dataset to evaluate the DL model.

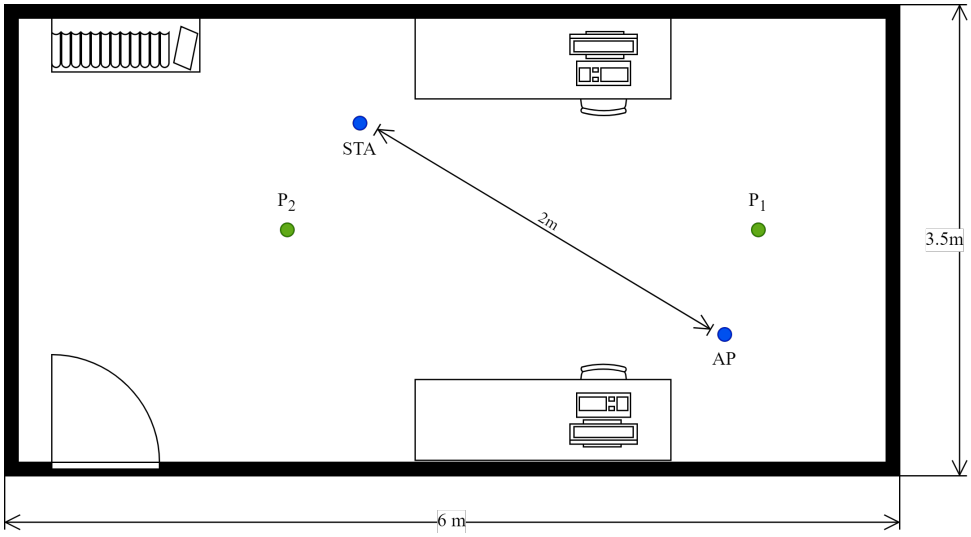

Figure 4.2: Collection scenario.

4.4 Modeling: The Human Activity Recognition Model

4.4.1 Model Definition

The model selected is based on the 1D-CNN model architecture proposed by Fard Moshiri *et al.* (2021), as illustrated in Fig. 4.3. This model exhibited high performance for HAR using a Single Board Computer (SBC), and its simple architecture is compatible with the TensorFlow Lite API for microcontrollers.

The model architecture presents a first Conv1D layer that has 64 filters and a kernel size of three, followed by a ReLU activation layer, a maxpool layer, and a dropout layer with a drop rate of 0.2 to prevent overfitting. This sequence is repeated, starting with another Conv1D layer that has 32 filters. It then connects to a flatten layer, followed by a dense layer with 16 units and ReLU activation function. Finally, a dense layer with three units and a softmax activation is used to obtain the class prediction. A summary of the model's hyperparameters can be found in Table 4.2.

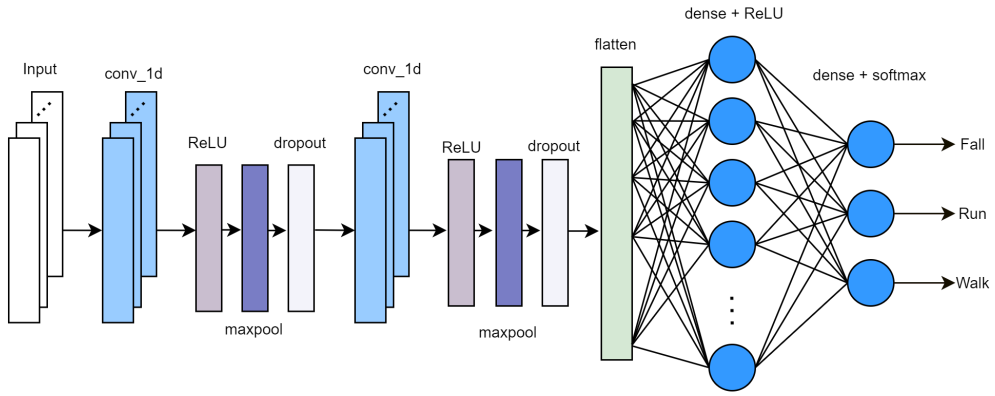

Figure 4.3: Model architecture.

Table 4.2: Model architecture description.

Layer	**Description**
Input	650×64
Conv1D ReLU Maxpool Dropout	filters: 64, kernel size: 3, strides: 1 rate: 0.2
Conv1D ReLU Maxpool Dropout	filters: 32, kernel size: 3, strides: 1 rate: 0.2
Flatten Dense ReLU	 units: 16
Dense	units: 3, activation: softmax

4.4.2 Model Training and Optimization

A Python script was developed using the TensorFlow/Keras API to implement and train the described model with training hyperparameters defined based on a grid-search algorithm. The model was trained with a six-batch size, resulting in 14 iterations over 200 epochs. An Adam optimizer with a learning rate of $1e^{-4}$ was used to update the model's weights. Training was conducted on an HP ProDesk computer with an Intel i7 10700T processor and 16 GB of DDR4 RAM. Model performance was evaluated using a hold-out validation method, with the

dataset partitioned into a 70% training set and a 30% test set to ensure the model's generalization capability.

For this proof-of-concept, only the model's weights were quantized using the TensorFlow Lite converter, converting them from 32-bit floating-point values to 8-bit integers, while model activations, inputs, and outputs were kept as float32. Prior to quantization, the model's parameters: activations, inputs, outputs, and weights occupied 394.64 KB. After quantization, the model's memory size was reduced to 108 KB.

4.5 Evaluation: Prediction on the Edge with a Deep Learning Model

The trained model was evaluated before optimization on the computer where it was trained and after optimization on the device for deployment, which was an ESP32-S3 development board with an ESP32-S3-WROOM-1-N16R8 SoC that includes a 16 MB flash memory and 8 MB of PSRAM that can be used for model allocation. To deploy the model in the ESP32, the TensorFlow Lite Micro for Espressif Chipsets API was used. This API requires that the resultant model from the converter is transformed from a flat binary buffer to a hexdump with C include file style to be included in an ESP-IDF project and be callable from the API functions.

To evaluate the model's performance, the metrics of accuracy, recall, precision, and f1-score were selected, as well as the average time taken to classify a single sample in both a computer and the ESP32 after optimization. Results showed that the model maintained the same performance after weight quantization, with an overall accuracy of 88.88%. Moreover, the average time taken by the computer for sample classification was 69.22 ms, while for the ESP32 it was 165.9 ms. Although ESP32 doubles the time, it is still quite impressive considering that, according to the manufacturer, its maximum power consumption is 1500 mA at 3.3 V, and its reduced processing capabilities when compared to the computer's.

Model's precision, recall and F1-score per class can be seen in Table 4.3. Meanwhile, it can be seen from the confusion matrix presented in Fig. 4.4 that

most classification errors were between samples whose labels were *Run* but were classified as *Walk*.

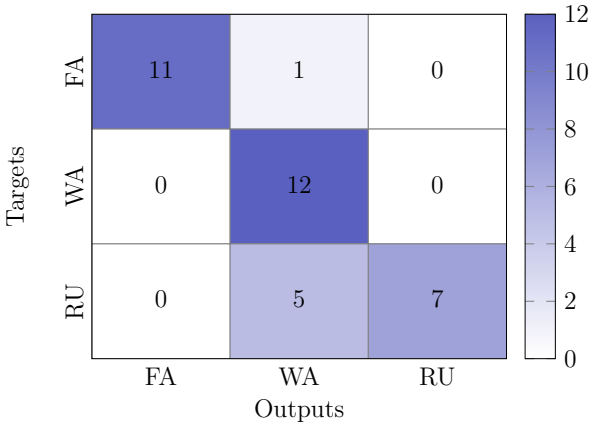

Figure 4.4: Confusion matrix obtained for the quantized embedded model.

Table 4.3: Performance metrics per activity for the quantized embedded model.

Activity	Precision	Recall	F1-score
FA	100%	91.67%	95.65%
WA	66.00%	100.0%	80.00%
RU	100%	58.33%	86.11%

4.6 Requirements Established for further System Design and Development

Drawing upon the results obtained in this chapter and the findings detailed in Chapter 2, the subsequent requirements for the development of a robust Wi-Fi-based HAR embedded system have been established:

- The Deep Learning model designed for HAR must achieve an accuracy greater than 90%. This threshold represents the baseline performance for most state-of-the-art Wi-Fi sensing systems dedicated to HAR, thereby serving as a benchmark for model efficacy in accurate activity recognition.
- The model must be optimized through quantization to facilitate its deployment on a microcontroller. This step is imperative to accommodate the

stringent hardware constraints, particularly regarding memory usage and CPU processing. Given that system resources must be shared with other processes, including variables, dynamic memory allocations, and program code, the model's memory usage and computational complexity must be minimized. The design shall be constrained by the 512 kB of internal memory available on commercial ESP32 microcontrollers.

- To ensure near real-time functionality, CSI data must be processed immediately upon acquisition. While near real-time operation does not impose a hard timing constraint, the processing latency must remain negligible from a user perspective. Consequently, a maximum total latency of 500 ms has been established for the system proposed in Chapter 6.

4.7 Conclusions

The proof-of-concept presented in this chapter provided a guideline for the definition, implementation, and evaluation of current tools and APIs for the development of Edge AI-based applications with embedded devices, specifically with microcontrollers. It also introduces a technical implementation of the CRISP-DM methodology, the Embedded CRISP-DM data pipeline, for developing Edge AI projects that extend beyond just Wi-Fi sensing applications.

By following the Embedded CRISP-DM pipeline, it was possible to deploy a DL model in an ESP32, considering its hardware constraints, such as limited memory. This pipeline suggest additional activities to be performed in each phase, ensuring replicability in other Edge AI projects. The resulting model achieved an overall accuracy of 88% for recognizing between three different activities: Fall, Walk, and Run. This performance was maintained after applying an 8-bit integer quantization to the model's weights, which reduced the model's size by nearly four times to fit within the ESP32's memory.

Additionally, by performing the proof-of-concept, it was possible to note deficiencies and areas for improvement concerning the most widely used CSI collection tool in the literature for ESP32 devices. The tool has limitations that

hinder the development of Wi-Fi sensing applications, especially for those unfamiliar with these microcontrollers. This is because, to configure the collection tool, it is necessary to access configuration menus present in the development framework for ESP32 devices and the ESP-IDF framework, or even modify the tool's source code if needed. Additionally, it was observed that it is impossible to configure the device at packet rates higher than 30 through the USB port to a computer at standard baud rates, as this causes the communication channel to become saturated, needing to modify the tool source code in order to achieve a packet rate of 50 packets per second.

Chapter 5

The ESP32 CSI Web Collecting Tool

5.1 Introduction

Current Wi-Fi CSI collection tools are primarily attached to specific network devices, such as the Linux 802.11n CSI Tool, which is only compatible with the Intel 5300 NIC by modifying its firmware. Current approaches aim to utilize the ESP32, a low-cost microcontroller with integrated Wi-Fi and Bluetooth, which features APIs that enable the collection of CSI through C language programming.

As its name suggests, the ESP32 CSI Tool, developed by (Hernandez & Bulut, 2020), utilizes ESP32 devices for CSI collection, making it the most widely used tool for CSI collection with microcontrollers. However, configuring this tool may require experience using the ESP32's development framework, as well as modifications to its source code to achieve optimal performance. To mitigate these accessibility and performance constraints, this chapter presents the *ESP32 CSI Web Collecting Tool* as an alternative that simplifies tool configuration and achieves packet rates of up to 180 packets per second. It supports USB-to-UART serial transmission through the ESP32 USB port, allowing for packet rates of up to 80, which outperforms the original ESP32 CSI Tool.

This chapter is organized as follows: Section 5.2 provides an in-depth description of the tool, detailing the communication parameters that can be configured within the tool, as well as the interaction between tasks that comprise the tool's

operation. In Section 5.3, the tool's performance is evaluated under different scenarios. Finally, conclusions are drawn in Section 5.4.

5.2 Tool Description

The proposed ESP32 CSI Web Collecting Tool requires two ESP32 devices to establish communication. One ESP32 is configured as a Wi-Fi receiver, referred to as Rx, which receives UDP packets sent from the other ESP32. The latter is configured as a Wi-Fi transmitter, referred to as Tx, which sends UDP packets at a configurable rate.

When an ESP32 starts the tool for the first time, or if the reset button on the board is pressed, the device enters in AP mode. This mode allows configuring the tool running on the device through a web form, where the fields for configuring the tool parameters depend on the desired device operation mode, either Rx or Tx. The parameters to be configured in Rx mode are:

- **MAC address:** MAC address of the Tx device.
- **Packet rate:** rate at which UDP packets will be received. This configuration helps avoid UDP packet duplication when packet rates are below 30.
- **Wi-Fi channel:** Wi-Fi channel. The channel must match the one selected in Tx.
- **Sample format:** Format in which the data will be sent or saved. It can be in either ASCII or binary format.
- **Message structure:** Measurements to be added into the message sent or saved data in the selected format. Values such as RSSI, timestamp, and antenna index, among others, can accompany CSI –see Appendix B.2 for details.
- **Informer mode:** Specify if measurements will be sent through UART serial communication I/O pins (Serial Communication Mode), through the

USB port to be seen in the ESP-IDF Terminal or by other program monitoring the USB port (Console Mode), or saved in an SD file (SD Mode).

- **MISO pin:** Define the Master In Slave Out (MISO) GPIO pin if SD Mode is selected as the Informer Mode.
- **MOSI pin:** Define the Master Out Slave In (MOSI) GPIO pin if SD Mode is selected as the Informer Mode.
- **CLK pin:** Define the clock (CLK) GPIO pin for synchronization if SD Mode is selected as the Informer Mode.
- **CS pin:** Define the Chip Select (CS) GPIO pin if SD Mode is selected as the Informer Mode.
- **Tx pin:** Define the Tx GPIO pin if the Serial Communication Mode is selected as the Informer Mode.
- **Rx pin:** Define the Rx GPIO pin if the Serial Communication Mode is selected as the Informer Mode.
- **Baudrate:** Define the baud rate to be used for communication if the Serial Communication Mode is selected as the Informer Mode.

On the other hand, the parameters to be configured in Tx mode are:

- **MAC address:** Mac address of the Rx device.
- **Packet rate:** rate at which UDP packets will be sent to Rx.
- **Wi-Fi channel:** Wi-Fi channel. The channel must match the one selected in Rx.

Fig. 5.1 shows the web form available when Rx operation mode is selected, while Fig. 5.2 shows it when Tx is selected.

Once the web form is submitted, the specified configuration is stored in the ESP32's non-volatile storage (NVS). Then, the ESP32 restarts and checks if configuration data is stored in the NVS. If configuration data is found, the ESP32

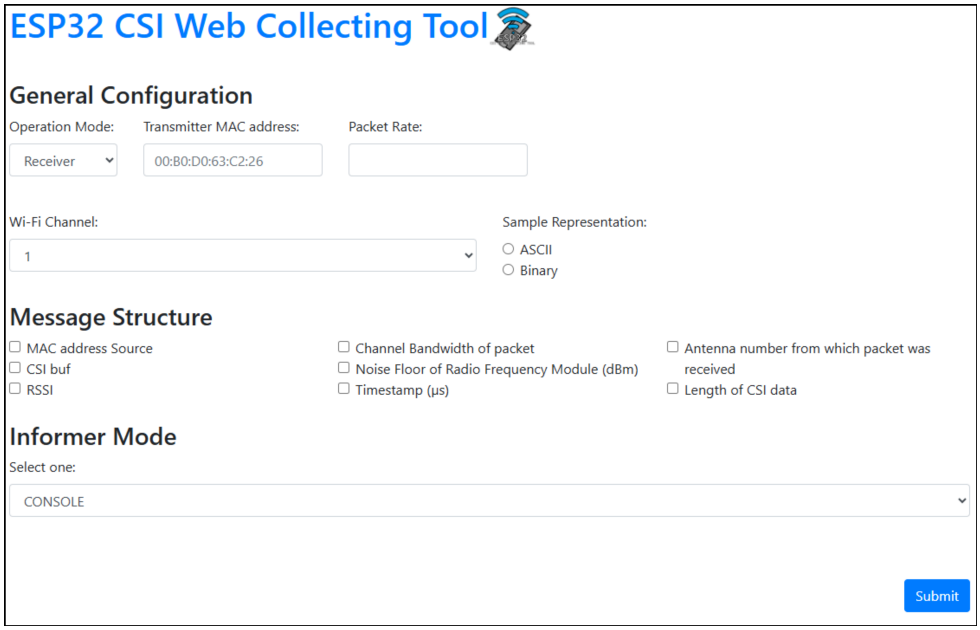

Figure 5.1: Web form for configuring the device as a receiver.

will start according to the defined operation mode and communication parameters. It is important to note that as configuration data is stored in the NVS, if the device is turned off, its configuration will be restored the next time it is turned on.

The ESP32 CSI Web Collecting tool is built on FreeRTOS (Barry, 2016), a real-time operating system for microcontrollers that enables both concurrent and parallel executions with the creation of tasks that can be scheduled into the available cores of the microcontroller. Hence, the tool comprises four tasks handled by FreeRTOS and its priority scheduling algorithm, exploiting the two cores that the ESP32 has. These tasks are the *Wi-Fi Task*, the *HTTP Server*, the *CSI Task*, and the *Informer Task* –see Appendix B.1 for details related to FreeRTOS task monitoring.

5.2.1 The Wi-Fi Task

The Wi-Fi Task is responsible for starting every other tool task when needed. At device startup, this task checks if tool configuration data has been created when

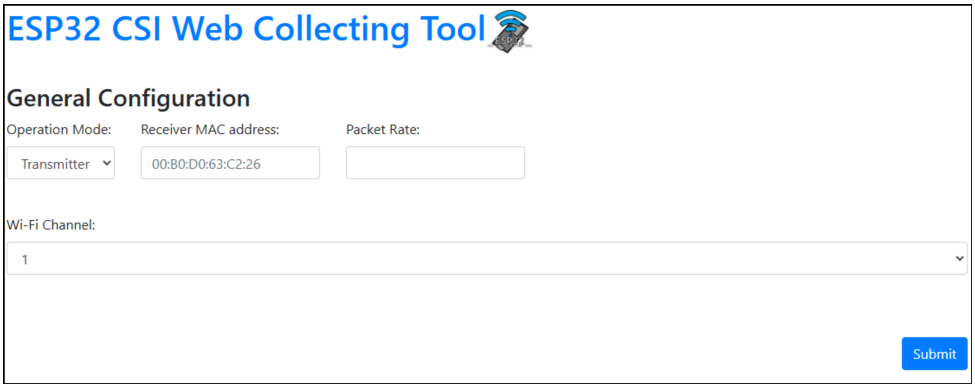

Figure 5.2: Web form for configuring the device as a transmitter.

the web form is submitted in the NVS. If configuration data can not be found, the device will start as an AP with network parameters shown in Table 5.1.

Table 5.1: Startup AP parameters.

Network Parameter	Value
SSID	CSI_Collecting_ESP32
Password	root1234
Wi-Fi Channel	1
Visibility	Visible
Bandwidth	20 MHz
IP address	192.168.1.1

Furthermore, when configuring the tool, the Wi-Fi Task starts the HTTP Server to handle HTTP methods related to the web form.

However, if configuration data is found, the device will load the configuration defined previously through the web form. It will change to the respective operation mode for CSI collection, initiating the CSI Task.

5.2.2 The HTTP Server

The HTTP Server registers Uniform Resource Identifiers (URIs) for fetching resources, such as the web page containing the web form or executing JavaScript functions to handle HTTP requests. One of the main URIs of the server is the one that handles a POST request, i.e., sends data to the server, which triggers

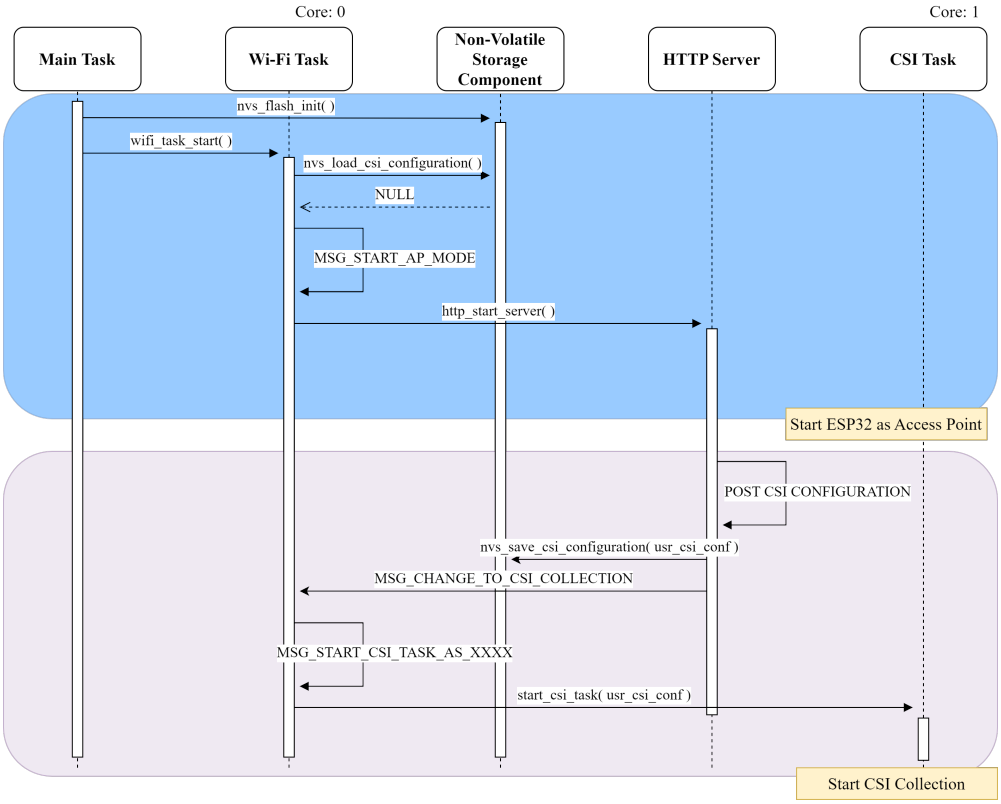

Figure 5.3: Sequence diagram for explaining the process of device starting as AP and changing its operation mode to CSI collection. The HTTP Server runs in the same core as the Wi-Fi task.

when the *Submit* button of the web form is pushed. This request is sent to the server, i.e., the ESP32 working as an AP, a JSON file, which is a text value file that stores data in pairs of key and value, containing the tool configuration for collecting CSI. The server then notifies the Wi-Fi task to start CSI collection, setting the device operation mode to Rx or Tx.

To provide an in-depth understanding of the interactions between the Wi-Fi task and the HTTP Server, Fig. 5.3 illustrates a sequence diagram that describes how these tasks interact with each other through function calls, return values, and message passing using task communication queues.

5.2.3 The CSI Task

The CSI task starts when the Wi-Fi task receives the configuration parameter values for collecting CSI and initiates the task accordingly. The functioning of this task varies depending on the device's operation mode. If the device is configured as Tx, it creates a socket using the User Datagram Protocol (UDP). This Internet communication protocol enables packets to be sent to the device configured as Rx. Then, it sets a software periodic timer, which invokes the callback function at a defined interval to send a packet through the socket to Rx. This leads to performing a CSI estimation for the received packet. The period at which the timer will invoke the callback function is defined as the inverse of the sampling rate submitted in the web form. The interactions between each task and component can be seen in Fig. 5.4.

On the other hand, when the device is configured as Rx, the CSI tasks help define a function that will be invoked as a callback by the Wi-Fi task when a UDP packet sent by the Tx is received, i.e., every CSI estimation is performed. The rest of the time, the task becomes idle, waiting to be stopped if a change in the operation mode is needed. Additionally, when the device is configured as Rx, the CSI task is also responsible for starting the Informer task, which is the task that sends or saves the estimated CSI. These interactions are described in Fig. 5.5.

5.2.4 The Informer Task

As mentioned before, the Informer task is started by the CSI task when the device is configured as Rx. Depending on the Informer mode selected in the web form, this task will either set up an SD card for saving CSI estimations into a file or install and set the UART configuration for sending CSI to a second device with asynchronous serial communication via GPIO pins. Using the USB interface of the development boards does not require additional configuration from this task.

Once set, this task will receive the CSI from the Wi-Fi task and then save or send these estimations in the defined format, depending on the Informer mode.

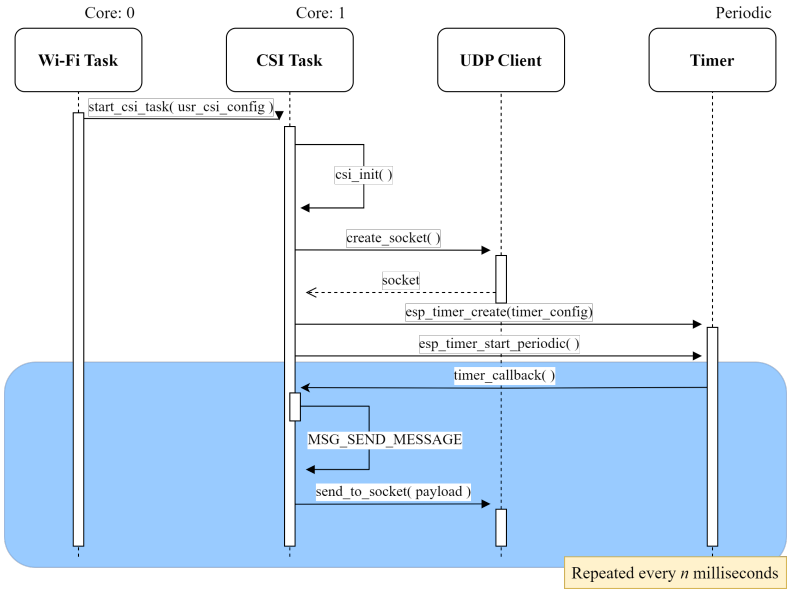

Figure 5.4: Sequence diagram for explaining the process of transmitting UDP packets for generating CSI.

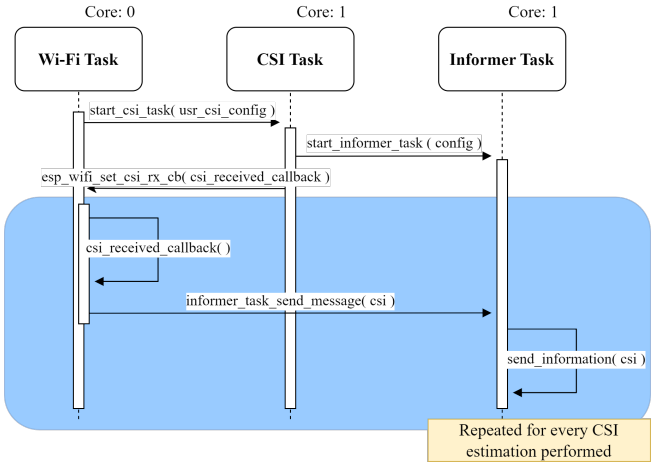

Figure 5.5: Sequence diagram for explaining the process of reporting estimated CSI from UDP packets received.

5.3 Performance Evaluations

A first evaluation was performed to test the stability of the ESP32 CSI Web Collecting tool, which consisted of sending UDP packets from Tx and registering the number of packets received at Rx, which translates to the number of CSI estimations available for further use. To determine the maximum packet rate at which the tool can be configured to handle only packet reception and CSI estimation, the Informer task in Rx was disabled to avoid possible bottlenecks that may originate from sending or saving CSI measurements to another device.

Table 5.2 presents the results for this first evaluation. As Rx performs CSI estimation per received packet, it can be seen that Rx did not receive every packet sent by Tx. This can be because UDP does not guarantee packet reception, in addition to interference generated by other Wi-Fi devices in the surrounding area. Packet rates above 200 were not supported.

Table 5.2: Stability and maximum packet rate evaluation results.

Configured Packet Rate	**Rx CSI Estimations**	**Packet Period (ms)**
20	18.7302	50
40	39.9134	25
60	59.3816	16.67
80	77.8126	12.5
100	97.8766	10
120	115.0033	8.33
140	136.4282	7.14
160	148.4420	6.25
180	163.8078	5.55
200	182.0546	5

Data transmission via serial communication occurs sequentially, i.e., one bit is sent at a time, so the rate at which data will be transmitted through the channel must be considered. This rate is known as the baud rate, and its value determines the number of CSI measurements that can be transmitted to a computer through a UART interface, subsequently dictating the maximum packet rate value that can be handled.

Theoretically, the packet rate, as a function of the baud rate, is given by the following equation:

$$packet\ rate = \frac{baud\ rate}{frame\ size} \tag{5.1}$$

If the chosen sample format for the tool is ASCII, each digit of a single CSI subcarrier's measurement, whether real or imaginary part, will be encoded using 8 bits. According to the Wi-Fi API for the ESP32, both real and imaginary parts of a CSI subcarrier's measurement have an 8-bit resolution ranging from -128 to 127. In the worst-case scenario, measurements from a single subcarrier would require 64 bits to be represented if every measurement is a three-digit negative value. This results in a total of 4096 bits for a complete CSI measurement. Additionally, since subcarrier measurements are separated by commas, the total frame size would be approximately 5112 bits. By substituting this frame size and the configured baud rate of 230400 bps into Equation 6.1, the maximum supported packet rate would be approximately 45 packets per second. Besides, if the format is set to binary, the frame size would be 1024 bits, resulting in a maximum theoretical supported packet rate of 225 packets per second.

To evaluate the tool, the maximum packet rate value that can be handled when using the USB to UART interface will be found for both sample format options in the tool, ASCII and binary. For this experiment, an ESP32 was configured as Tx to send UDP packets at varying rates, ranging from 10 to 100 packets per second, incrementing in steps of 10 to a second ESP32 configured as Rx. Rx was configured in Console mode, allowing the device to send CSI estimations through its USB port to a computer running a Python script at a baud rate of 230400 bps, and to register the average number of measurements received per second for 20 minutes. Both Rx and Tx were ESP32-DevkitCVIE development boards, which feature an ESP32-D0WD-V3 dual-core chip with adjustable clock frequency from 80 MHz to 240 MHz, along with 8 MB of Flash memory and 8 MB of PSRAM.

The theoretical value calculated for the sample format set to ASCII can be confirmed by the results obtained using the tool presented in Fig. 5.6. As can be seen, the average number of CSI measurements received at the computer matches the packet rate configured at the tool with only slight differences. However, this behavior changes once the configured packet rate exceeds 30 packets per second.

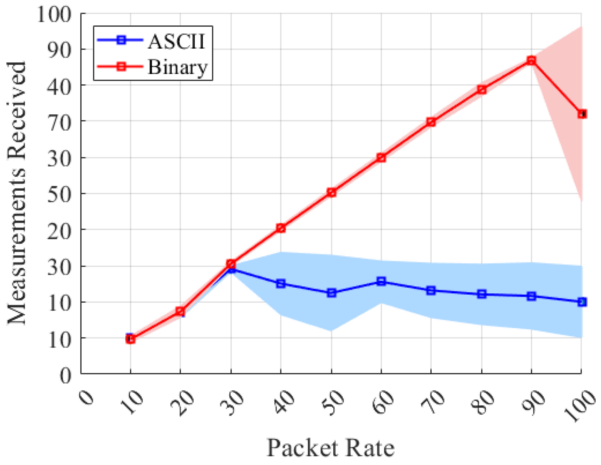

Figure 5.6: CSI measurements received in the computer at different Tx-Rx packet rates. Shaded area represents the variance obtained for each packet rate.

Beyond this point, regardless of whether the rate is configured with higher values, the measurements received at the computer remain around 30 measurements per second.

However, it can also be observed that when the sample format is set to binary, the maximum supported packet rate is less than half of the theoretical value of 225 packets per second. The average number of CSI measurements received at the computer matches the configured packet rate until it reaches values above 80 packets per second, as the measurements received at the computer typically remain around 85 packets per second. This discrepancy can be attributed to the time it takes to execute the write instruction for sending data through the USB interface of the board, which is slower compared to writing bytes directly to a UART port configured on the GPIO pins of the board. This same behavior was observed when saving CSI measurements in an SD card.

The baud rate can be incremented to achieve higher packet rates that match the number of measurements received, or the tool can be configured to transmit CSI measurements through a UART port that is not associated with the board's USB interface. Nevertheless, it is important to consider the baud rates supported by the device connected to the ESP32, taking into account the crystal oscillator frequency used for UART.

5.4 Conclusions

In this chapter, we introduced the ESP32 CSI Web Collecting Tool. This tool offers several advantages over other CSI collection tools, as it can be configured via a web form without requiring alterations to the source code or adjustments to compilation parameters in the development framework. This feature enables reconfiguration in the field, allowing researchers to change communication parameters or adapt the device to their needs without removing it from the experimentation area.

The collecting tool also presents three informer operation modes, which allow sending CSI measurements through the device USB port to another device; it also allows sending to another device by using the device's GPIO for serial communication under UART communication protocol or saving the measurements on an SD card.

Evaluation results indicate that this tool outperforms existing options when using the ESP32 as the collecting device, as it supports higher packet rates, which is often essential for various Wi-Fi sensing applications.

Chapter 6

Enhancing the Human Activity Recognition Model for Embedded Systems with Data Augmentation

6.1 Introduction

This chapter focuses on the development, implementation, and evaluation of a Human Activity Recognition model designed for embedded devices, utilizing samples of five different activities collected with the ESP32 CSI Web Collecting Tool. However, the number of collected samples was insufficient for achieving satisfactory results. To address this issue, two data augmentation techniques are proposed to enhance the performance of Deep Learning models. The first technique involves the combination of signals through EMD, while the second leverages Generative Adversarial Network (GAN) for generating synthetic samples. These samples were evaluated and introduced for training a model for implementation in an embedded device, which evaluation indicates outstanding performances by achieving accuracies above 90% and a near real-time functioning. The methodology followed during this chapter is summarized in Fig. 6.1.

This chapter is organized as follows: Section 6.2 details the data collection for creating the activity dataset. Then, Section 6.3 outlines the preprocessing steps applied to the collected data. To assess the quality of synthetic samples, Section 6.4 describes the evaluation process of these samples, while Section 6.5 presents the results of these evaluations. In Section 6.6, the results for evaluating

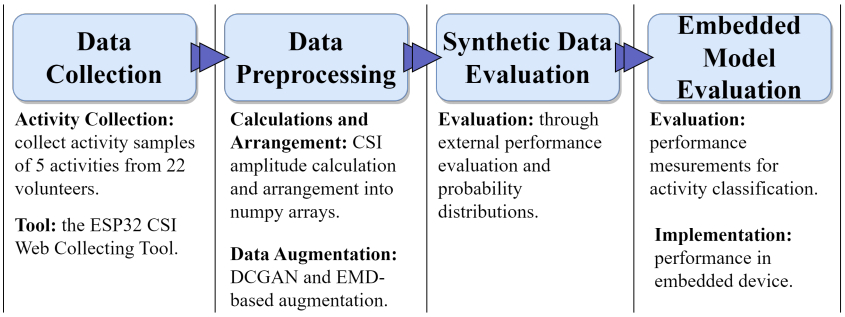

Figure 6.1: Overview of the methodology followed for this chapter.

and implementing a DenseNet-based model, incorporating data augmentation for training, into an embedded device are presented. Finally, conclusions are drawn in Section 6.7.

6.2 Data Collection

The ESP32 CSI Web Collecting Tool was flashed onto two ESP32-DevkitCVIE development boards, one for Rx and one for Tx. These boards feature an ESP32-D0WD-V3 dual-core chip with an adjustable clock frequency from 80 MHz to 240 MHz, 8 MB of flash memory, and 8 MB of PSRAM.

The tool was configured to operate on a 20 MHz Wi-Fi channel, transmitting and receiving UDP packets at a rate of 50 packets per second. The informer mode of the tool was set to *Console*, which enables the tool to send CSI estimations through the device's USB port in binary format. Additionally, a Python script was developed to collect and save these estimations into comma-separated values files.

A total of 22 volunteers were asked to perform 15 repetitions of five different activities, each lasting 20 seconds. Each volunteer provided their informed consent to be part of this research. The collected samples were used to construct the *Activity Recognition* dataset.

The volunteers were asked to perform the following activities:

- **Lie Down**: the volunteer starts standing at point A_p. After five seconds, the volunteer sits on the camping cot and lies down.

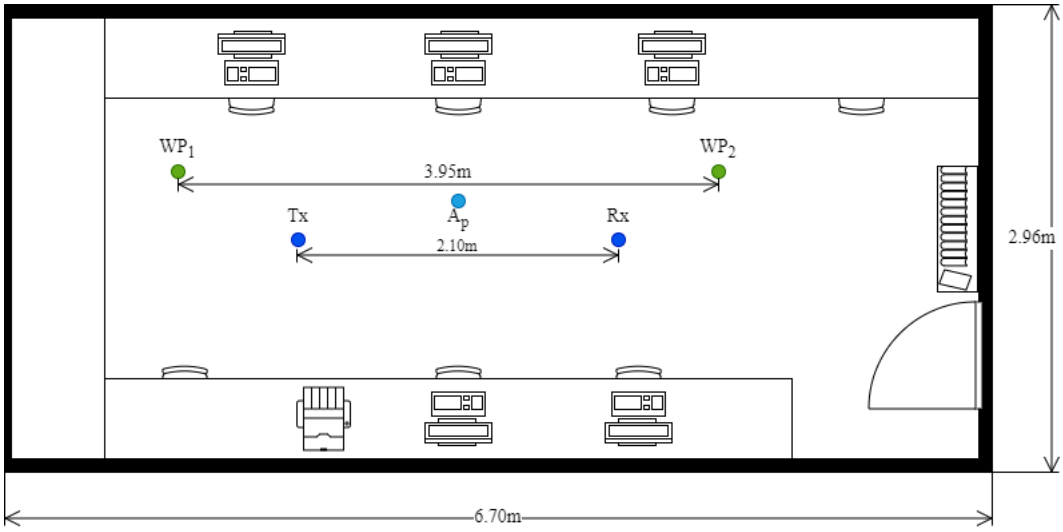

Figure 6.2: CSI collection scenario. WP_1 and WP_2 are the reference points for starting and ending walking. A_p is the point where fall, sit down, lie down, get up activities took place.

- **Get Up**: the volunteer remains on the camping cot. After five seconds, the volunteer starts getting up and stands still at point A_p.
- **Sit Down**: the volunteer stands behind a chair placed at point A_p. After five seconds, the volunteer walks around the chair and sits down.
- **Fall**: the volunteer stands next to a mattress at floor level at point A_p. After five seconds, the volunteer lies on the mattress at a fast pace, simulating a fall.
- **Walk:** the volunteer walks back and forth between WP_1 and WP_2 points repeatedly during the recording time.

Each activity CSI sample collection was carried out in the scenario illustrated in Fig. 6.2.

A total of 1647 activity samples were collected and partitioned into training and test sets for hold-out validation, the method used to evaluate the Deep Learning model, as detailed in Table 6.1.

Table 6.1: Breakdown of Activity Recognition samples.

Activity	Samples for training	Samples for testing	Total
Lie Down	239	90	329
Get Up	240	89	329
Sit Down	242	90	332
Fall	238	89	327
Walk	239	91	330

6.3 Data Preprocessing

6.3.1 Preprocessing for Data Augmentation

The collected samples consist of complex pairs for each CSI estimation across all subcarriers. These pairs were utilized to calculate the CSI amplitude, defined as:

$$|H| = \sqrt{Re^2 + Im^2} \tag{6.1}$$

As a result, 64 amplitude values were calculated for each CSI estimation. However, this number was later reduced to 52 by excluding those identified as null and pilot subcarriers, as well as subcarriers identified as problematic based on empirical experimentation. Furthermore, only 47 of the 52 were retained for further experiments since visualization experiments indicated that certain subcarriers exhibited behavior similar to the null subcarriers.

CSI amplitudes were arranged into numpy arrays to facilitate their handling through Python scripts and TensorFlow. Samples were truncated into 850 measurements to assess that numpy arrays must be the same length, because although the tool was configured at a rate of 50 packets/s, there is variability in the number of packets received at Rx per second. With these considerations, two numpy arrays were created, the first with a shape of $1198 \times 47 \times 850$ with the samples of the training set, while the second with a shape of $449 \times 47 \times 850$ with the samples of the test set. Only the training set was used for augmenting its size through the following data augmentation algorithms.

6.3.2 The EMD-based algorithm

The algorithm developed for synthetic subcarrier generation based on EMD is an adaptation from Zhang *et al.* (2019). Suppose having a dataset of CSI amplitudes $\mathbb{H}$ with n samples. Each sample $\mathbf{H}_i$ is a tensor with 47 subcarriers and 850 timesteps, i.e., a tensor of dimension 47×850. EMD is applied to each subcarrier sc_k of $\mathbf{H}_i$ for finding its IMF. For each sc_k, 7 IMFs, plus a residue, are extracted by EMD. This gives as a result n tensors $\mathbb{S}$ of dimensions $47 \times 8 \times 850$, which contain 850 steps for each IMF and residue for each sc_k of sample H_n.

Following the example, the n tensors $\mathbb{S}$ are further concatenated for constructing the tensor $\mathbb{M}$ of dimension $n \times 47 \times 8 \times 850$, i.e., the total number of samples $\times$ the number of subcarriers (47) $\times$ the number of modes extracted for each subcarrier (8) $\times$ the timesteps. This structure allows the creation of synthetic samples by adding the modes taken from a pair of tensors $\mathbf{A}$ and $\mathbf{B}$, which are part of $\mathbb{M}$, alternately from the subcarrier at index k. Therefore, each synthetic sample $\hat{\mathbf{H}}_i$ can be seen as a group of 47 synthetic subcarriers, each being the result of the combination between modes of a pair:

$$synth\_sc_k = A_{(k,1,:)} + B_{(k,2,:)} + \cdots + A_{(k,8,:)} + B_{(k,8,:)} \tag{6.2}$$

$$\hat{\mathbf{H}}_i = [synth\_sc_1, synth\_sc_2, \ldots, synth\_sc_{47}] \tag{6.3}$$

This process is illustrated in Fig. 6.3, which shows how a synthetic subcarrier is created from the same subcarrier index of two different samples.

With this algorithm, it was possible to create a total of 10315 synthetic samples by combining samples from individual volunteers only from the training set, resulting in a set of size $11513 \times 47 \times 850$ considering the real samples. However, due to the available RAM in the computer used to train the models, only 9584 samples were used, resulting in an augmentation of eight times the original size.

6.3.3 Synthetic CSI amplitude generation with DCGANs

The discriminator and generator models utilized for generating synthetic samples are based on the frameworks proposed and validated in Xiao *et al.* (2019), specifically a DCGAN tailored for generating synthetic CSI amplitude samples. These models were adapted to this work to directly work with CSI amplitudes of

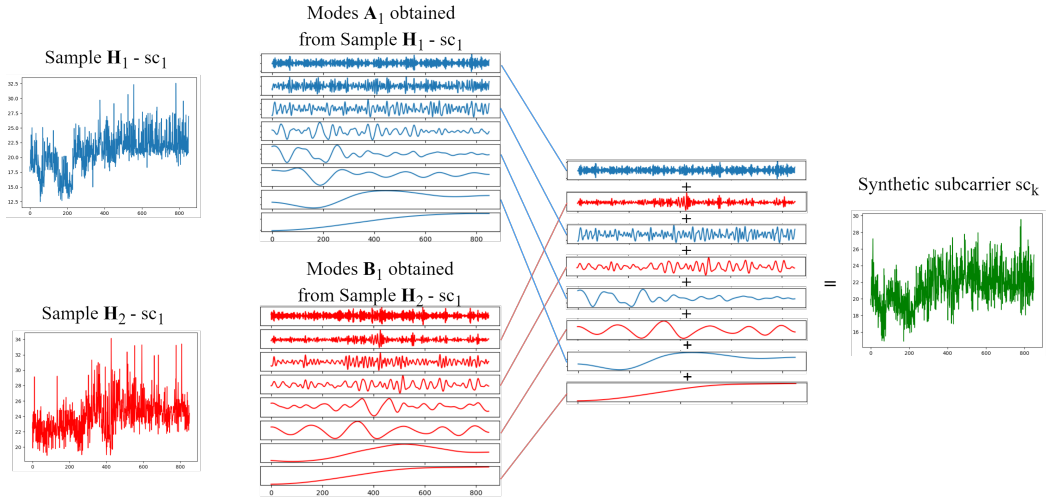

Figure 6.3: Process for generating a synthetic subcarrier from the same subcarrier index of two different fall samples.

every subcarrier as real inputs for the discriminator model, resulting in an input size of *timesteps* $\times$ 1 by using 1D-Convolutional layers and their Transpose. For each activity, one DCGAN was trained, resulting in five models, each of these specialized in generating synthetic CSI amplitude samples for a specific activity according to the model architecture presented in Table 6.2 for the discriminator and in Table 6.3 for the generator. The training of the DCGAN was conducted over 400 epochs, with a batch size of 64 and a learning rate of 0.0001 with Adam optimizer.

Table 6.2: DCGAN discriminator architecture.

Layer	Description	Output Shape
Dropout	rate: 0.2	(850 , 52)
Conv1D Batch Normalization LeakyReLU	filters: 96, kernel size: 3, strides: 1	(848, 96)
Conv1D Batch Normalization LeakyReLU	filters: 96, kernel size: 3, strides: 1	(846, 96)
Conv1D Batch Normalization LeakyReLU	filters: 96, kernel size: 3, strides: 5	(169, 96)
Dropout	rate: 0.5	(169, 96)
Conv1D Batch Normalization LeakyReLU	filters: 192, kernel size: 3, strides: 1	(167, 192)
Conv1D Batch Normalization LeakyReLU	filters: 192, kernel size: 3, strides: 1	(165, 192)
Conv1D Batch Normalization LeakyReLU	filters: 192, kernel size: 3, strides: 5	(33, 192)
Dropout	rate: 0.5	
Conv1D Batch Normalization LeakyReLU	filters: 192, kernel size: 3, strides: 1	(31, 192)
Conv1D Batch Normalization LeakyReLU	filters: 192, kernel size: 1, strides: 1	(31, 192)
Conv1D Batch Normalization LeakyReLU	filters: 192, kernel size: 1, strides: 5	(31, 192)
Global Avg. Pool		(192)
Dense	units: 1, activation: sigmoid	(1)

Table 6.3: DCGAN generator architecture.

Layer	Description	Output Shape
Dense Batch Normalization ReLU	units: 85×512	(43520)
Reshape	new shape: = (85, 512)	(85, 512)
Conv1D Transpose Batch Normalization ReLU	filters: 256, kernel size: 5, strides: 5	(425, 256)
Conv1D Transpose Batch Normalization ReLU	filters: 128, kernel size: 5, strides: 2	(850, 128)
Conv1D Transpose Batch Normalization	filters: 1, kernel size: 5, activation: tanh, strides: 1	(850, 1)

6.4 Evaluating the Effect of Synthetic data

It is important to notice that there is no single metric to evaluate the performance of data augmentation algorithms for time series. According to Iglesias *et al.* (2023), there are three approaches to evaluate these algorithms.

The first approach involves external performance evaluation, which compares two Deep Learning models, one trained without synthetic data and the other trained with synthetic data. This comparison assesses whether data augmentation has improved model performance. However, this approach does not measure the quality of synthetic time series.

The second approach consists of the use of metrics related to GANs, such as Inception Score, Mode Score, Fréchet Inception Distance, and Discriminative Score. Nevertheless, this approach requires the use of an external pre-trained model.

At last, the third approach aims to measure the similarity between the probability distributions of synthetic and real samples, thereby evaluating the quality of the data. This approach can rely on visual comparisons with dimensionality reduction techniques and statistical measures.

For this research, the first and third approaches were used to evaluate the performance of the data augmentation algorithms.

For external performance evaluation, a modified version of the Bi-LSTM DenseNet121 based on the architecture proposed by Zhang *et al.* (2021) was implemented for Activity Recognition. Modifications included adapting to use CSI amplitudes of a single subcarrier as input rather than CSI spectrograms of all subcarriers. The model architecture is illustrated in Fig. 6.4 and described in Table 6.4.

A total of five models were trained under the following circumstances:

1. Model training with no data augmentation.

2. Model training with EMD-based data augmentation, doubling the number of samples per activity in the training partition. The obtained model will be referred to as the x2 EMD Model.

3. Model training with EMD-based data augmentation, triplicating the number of samples per activity in the training partition. The obtained model will be referred to as the x3 EMD Model.

4. Model training with DCGAN data augmentation, doubling the number of samples per activity in the training partition. The obtained model will be referred to as the x2 DCGAN Model.

5. Model training with DCGAN data augmentation, triplicating the number of samples per activity in the training partition. The obtained model will be referred to as the x3 DCGAN Model.

All these models perform classification per sample subcarrier, meaning that to assign a class label to a sample, every subcarrier must be processed by the model, accumulating the softmax result across subcarriers for a single sample. Each model was trained over 300 epochs with a batch size of 32 and a learning rate of 0.0001, utilizing the Adam optimizer for gradient-based optimization. Model performance was assessed using the separate test set described previously, composed of samples excluded from the training phase. Evaluation metrics included

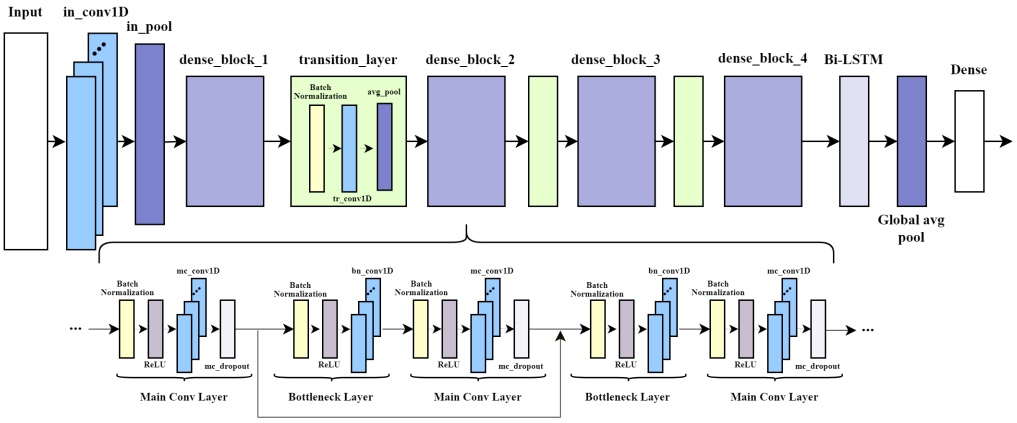

Figure 6.4: DenseNet121 with Bi-LSTM architecture

average accuracy, precision, and recall, and F1-score, computed across all five target classes to ensure a comprehensive performance comparison.

Meanwhile, to measure the similarity between probability distributions, histograms, and probability density functions were computed for visual assessment. Moreover, statistical measures derived from the data, such as the mean, variance, median, and skewness, were calculated to conduct a quantitative analysis.

Table 6.4: DenseNet121 with Bi-LSTM architecture description. Dense blocks are ordered according to Fig. 6.4

Layer	Description	Output Shape
in_conv1D	filters: 11, kernel size: 7, strides: 2	(425, 11)
in_pool	pool size: 3, strides: 2	(212, 11)
dense_block_1		
bn_conv1D	(filters: 44, kernel size: 1, strides: 1) ×6	(212, 77)
mc_conv1D	(filters: 11, kernel size: 3, strides: 1) ×6	
mc_dropout	(drop_rate = 0.3) ×6	
transition_layer		
tr_conv1D	filters: 5, kernel size: 1, strides: 1	(106, 5)
avg_pool	pool size: 2, strides: 2	
dense_block_2		
bn_conv1D	(filters: 44, kernel size: 1, strides: 1) ×12	(106, 137)
mc_conv1D	(filters: 11, kernel size: 3, strides: 1) ×12	
mc_dropout	(drop_rate 0.3) ×12	
transition_layer		
tr_conv1D	filters: 5, kernel size: 1, strides: 1	(53, 5)
avg_pool	pool size: 2, strides: 2	
dense_block_3		
bn_conv1D	(filters: 44, kernel size: 1, strides: 1) ×24	(53, 269)
mc_conv1D	(filters: 11, kernel size: 3, strides: 1) ×24	
mc_dropout	(rate 0.3) ×24	
transition_layer		
tr_conv1D	filters: 5, kernel size: 1, strides: 1	(26, 5)
avg_pool	pool size: 2, strides: 2	
dense_block_4		
bn_conv1D	(filters: 44, kernel size: 1, strides: 1) ×16	(26, 181)
mc_conv1D	(filters: 11, kernel size: 3, strides: 1) ×16	
mc_dropout	(rate: 0.3) ×16	
transition_layer		
tr_conv1D	filters: 5, kernel size: 1, strides: 1	(13, 5)
avg_pool	pool size: 2, strides: 2	
Bi_LSTM	units: 128 each	(13, 256)
glob_avg_pool	-	(256)
Dense	units: 5, activation: softmax	(5)

6.5 Results for Data Augmentation

Figure 6.5 presents a comparative visualization of synthetic samples, generated with both DCGAN and EMD-based methods, for each respective activity. Visual inspection reveals that the synthetic data produced by the EMD-based method exhibits a more distinct representation of the underlying activities, which are characterized by pronounced, physiologically consistent alterations in signal amplitude. For instance, the EMD-based synthetic fall sample demonstrates a pronounced spike in amplitude values over the initial 250 packets (corresponding to the first five seconds of the activity), accurately capturing the volunteer's initial position adjacent to the mattress. This is followed by an amplitude transition as the fall occurs and the volunteer comes to rest upon it. In contrast, the synthetic fall sample produced by the DCGAN lacks this clear, interpretable correspondence to the kinematic events of the fall, suggesting a lower fidelity in representing the temporal dynamics of the activity.

6.5.1 Training Results

As mentioned, five models were trained for Activity Recognition: the baseline model with no data augmentation, the x2 EMD model, the x3 EMD model, the x2 DCGAN model, and the x3 DCGAN model. From the training set, 30% of the samples were taken for validation tests. After each epoch, the model is validated using these validation samples that were not used during the training steps, obtaining the validation accuracy.

The training and validation accuracies obtained after each epoch are plotted in Figs. 6.6, 6.7, 6.8, 6.9, 6.10, and 6.11.

6.5.2 Statistical Evaluation

The statistical measures for the training data with no data augmentation are presented in Table 6.5. It is important to note that every activity exhibits positive skewness, indicating that the right tail of the distribution is longer than the left tail. This suggests that the data are clustered towards the left side of the

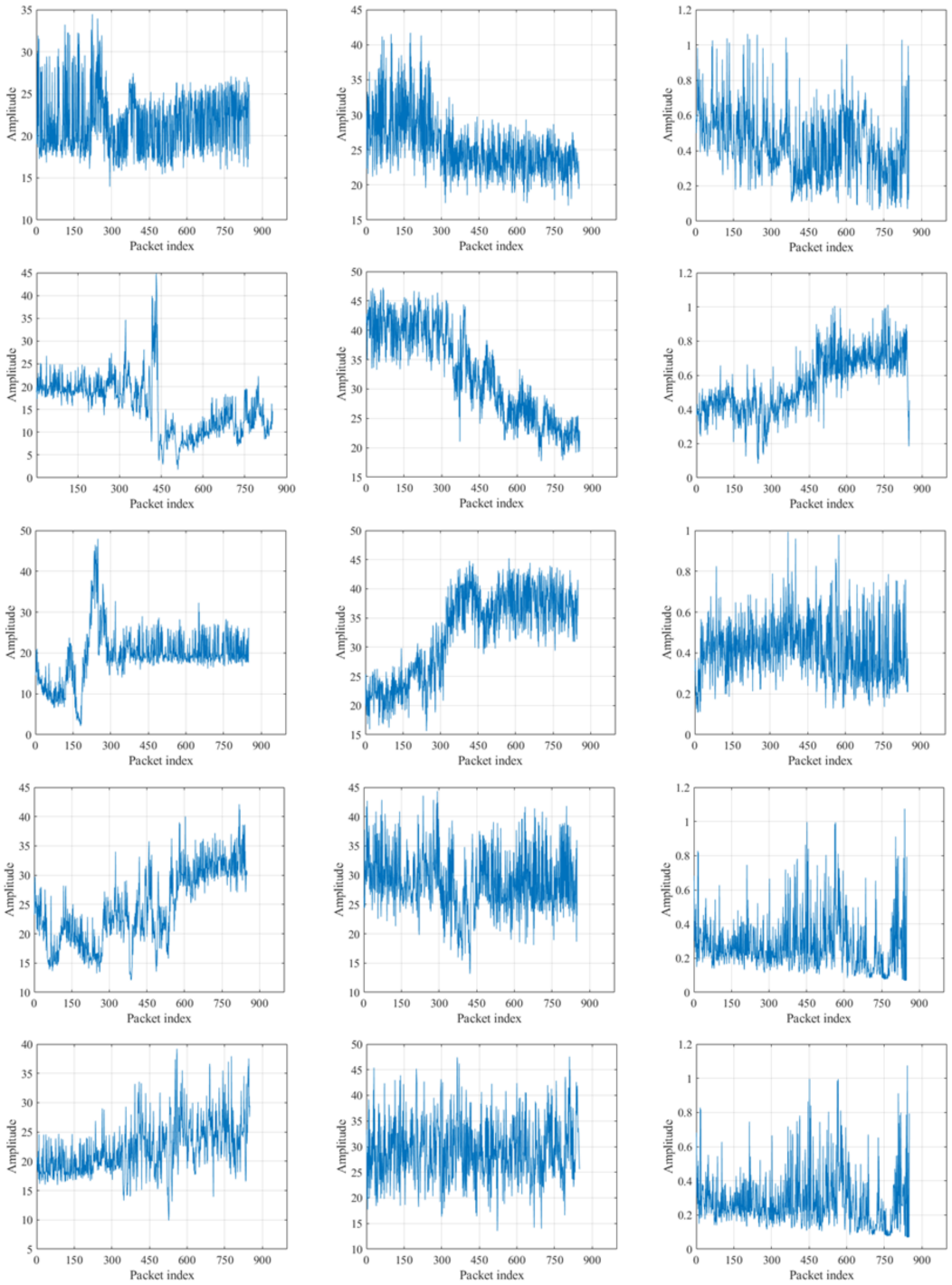

Figure 6.5: From left to right, original subcarrier amplitude, synthetic amplitude generated with the EMD-based algorithm, and synthetic amplitude generated with DCGAN. From top to bottom, fall, get up, lie down, sit down, and walk.

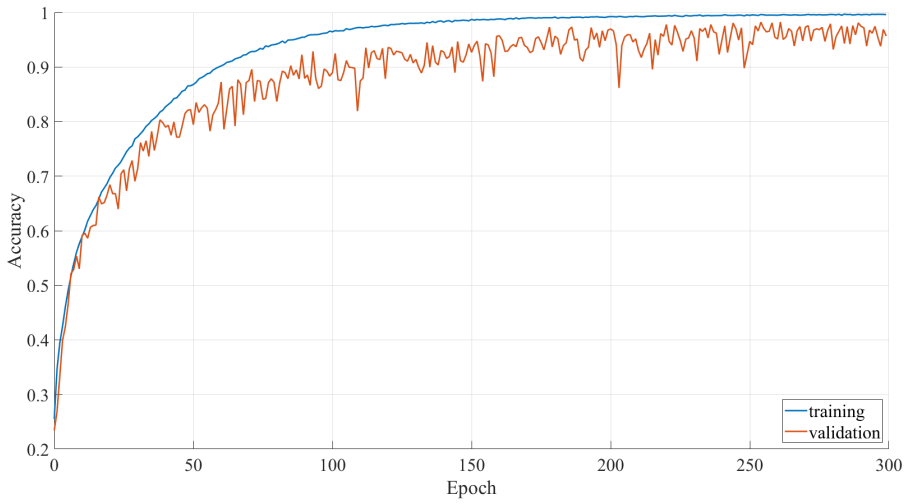

Figure 6.6: Training and validation accuracy with no data augmentation.

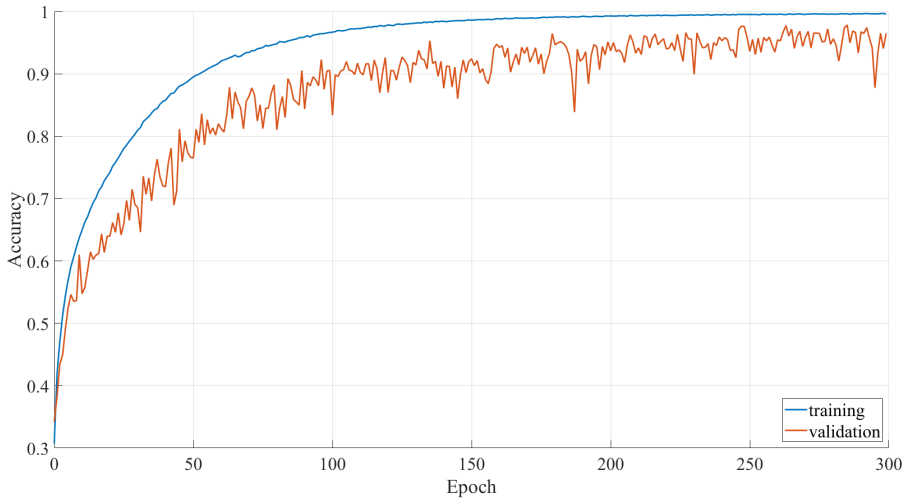

Figure 6.7: Training and validation accuracy with x2 EMD Model.

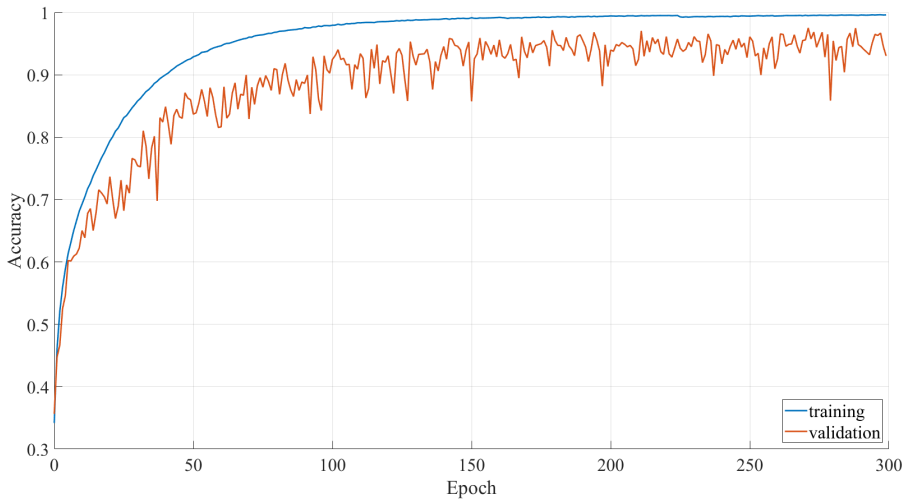

Figure 6.8: Training and validation accuracy with x3 EMD Model.

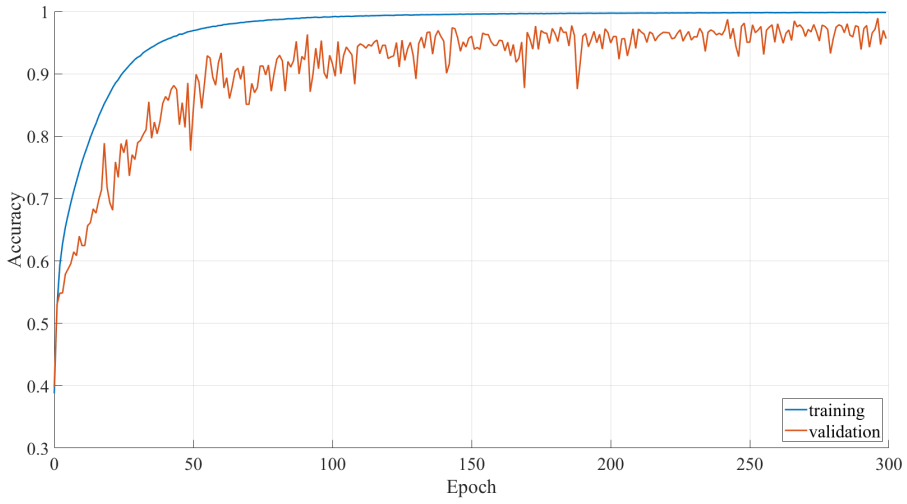

Figure 6.9: Training and validation accuracy with x6 EMD Model.

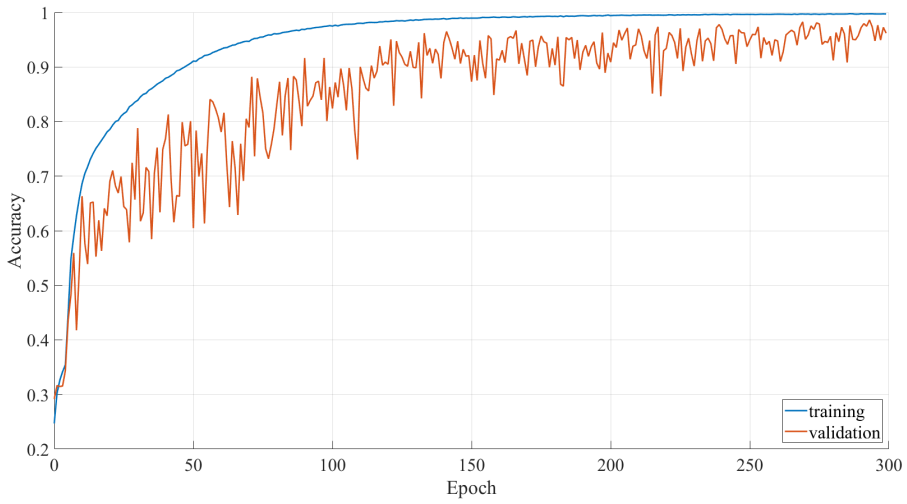

Figure 6.10: Training and validation accuracy with x2 DCGAN Model.

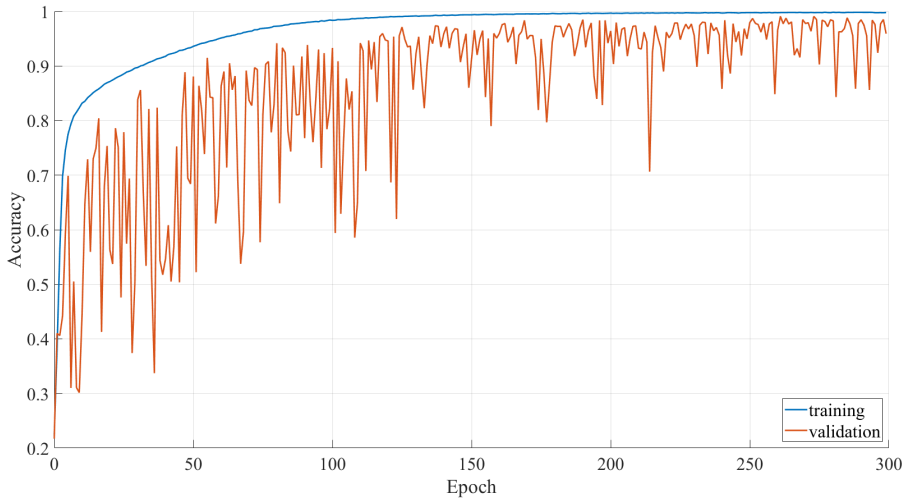

Figure 6.11: Training and validation accuracy with x3 DCGAN Model.

distribution. A similar trend can also be observed in the synthetic data generated using the EMD algorithm and DCGAN, as shown in Tables 6.6 and 6.7.

When LoS exists between Rx and Tx, alongside non-LoS paths generated due to multipath propagation, i.e., scattering, diffraction, and reflection, signal fading captured in CSI follows a Rician distribution (Awon *et al.*, 2012). Given that LoS was present between the devices during the data collection, real CSI amplitude values are expected to follow this distribution. This expectation is supported by Fig. 6.12, where it can be seen that the amplitude values from samples for each activity follow this distribution. Moreover, the Rician distribution is also present in the probability distribution shown in Fig. 6.13, corresponding to the synthetic samples generated using the EMD-based algorithm, thereby confirming the quality of the data produced by this algorithm. In contrast, probability distributions for synthetic samples generated with DCGAN show a higher density on the left side of the distribution for fall, lie down, and sit down activities compared to the same activities in EMD-based and real data distributions.

Table 6.5: Data statistics training data with no data augmentation.

Measure	**FA**	**LD**	**GU**	**SD**	**WA**
Mean	0.4187	0.4174	0.4226	0.4603	0.4331
Median	0.3955	0.4005	0.4069	0.4604	0.4218
Skewness	0.3599	0.3427	0.3138	0.0825	0.2663
Variance	0.0432	0.0421	0.0410	0.0349	0.0323

Table 6.6: Data statistics for EMD-based synthetic data

Measure	**FA**	**LD**	**GU**	**SD**	**WA**
Mean	0.4428	0.4350	0.4365	0.4553	0.4526
Median	0.4306	0.4213	0.4238	0.4490	0.4437
Skewness	0.2586	0.3038	0.2941	0.1819	0.2496
Variance	0.0334	0.0336	0.0332	0.0312	0.0284

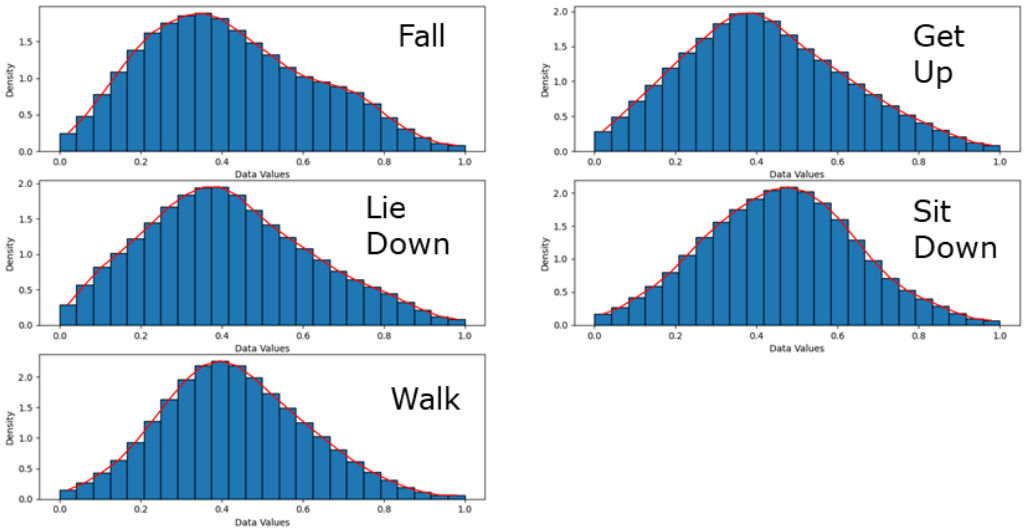

Figure 6.12: Probability distribution for real CSI amplitude data.

Table 6.7: Data statistics for DCGAN synthetic data

Measure	FA	LD	GU	SD	WA
Mean	0.3955	0.4181	0.4120	0.4079	0.4645
Median	0.3685	0.4077	0.3956	0.4093	0.4443
Skewness	0.3701	0.2322	0.3310	0.1314	0.3643
Variance	0.0536	0.0482	0.0447	0.0402	0.0390

6.5.3 Evaluation through Classification

To analyze the performance of data augmentation algorithms through external performance evaluation using a Bi-LSTM DenseNet121 model, it is necessary first to analyze the model's performance when trained with the baseline model, i.e., without data augmentation. Thus, Fig. 6.15 illustrates the confusion matrix for the trained model with no data augmentation over the samples in the test set, presenting an overall accuracy of 59.91%; while Table 6.8 presents the precision, recall, and F1-score metrics per activity class. Analysis of the confusion matrix reveals a recurring misclassification between the Lie Down and Fall activities that can likely be attributed to the kinematic similarity between the two movements. The volunteer actions involved a transition from a standing to a lying

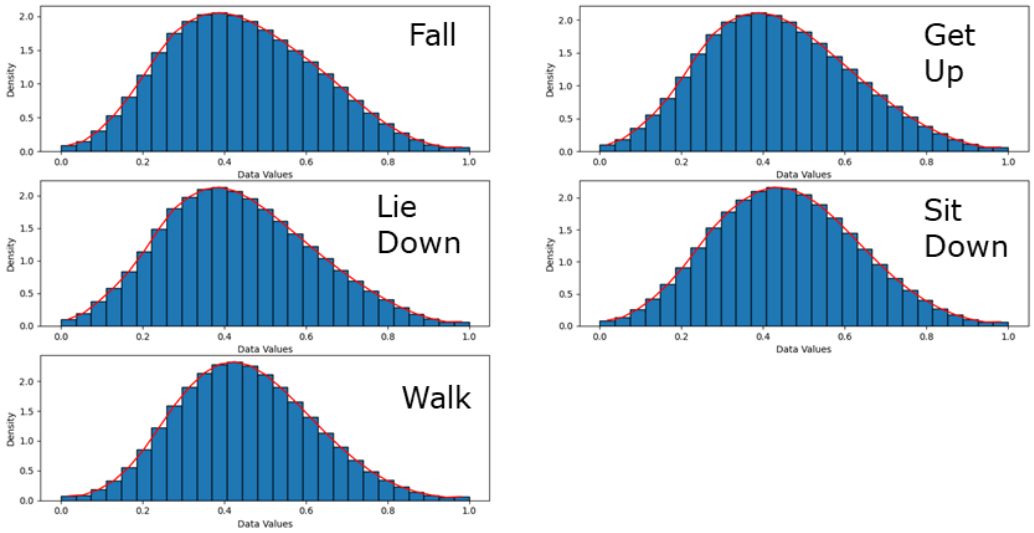

Figure 6.13: Probability distribution for synthetic CSI amplitude data generated with EMD-based algorithm.

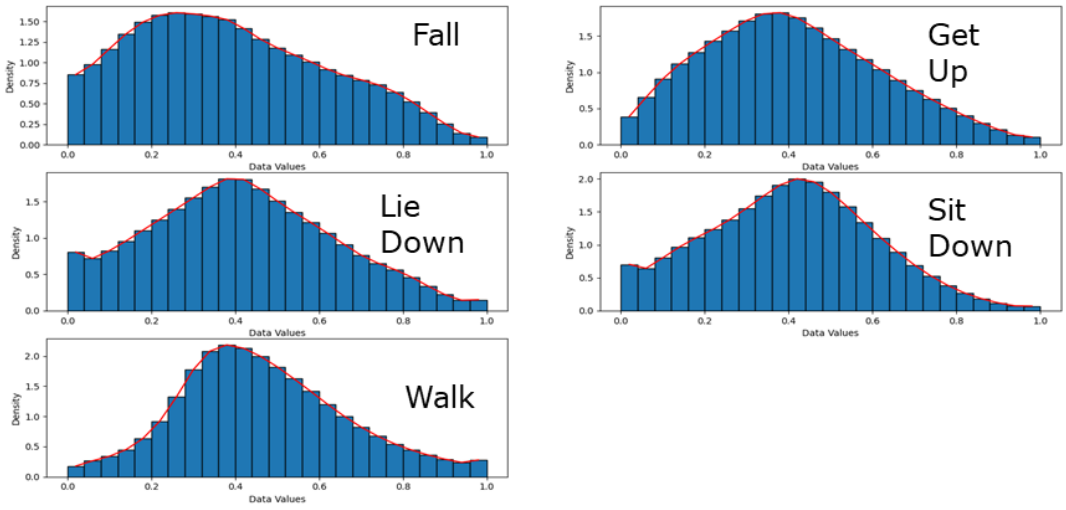

Figure 6.14: Probability distribution for synthetic CSI amplitude data generated with DCGAN.

position, differing only in the pace at which these activities occurred, which was not captured or learned by the classification model.

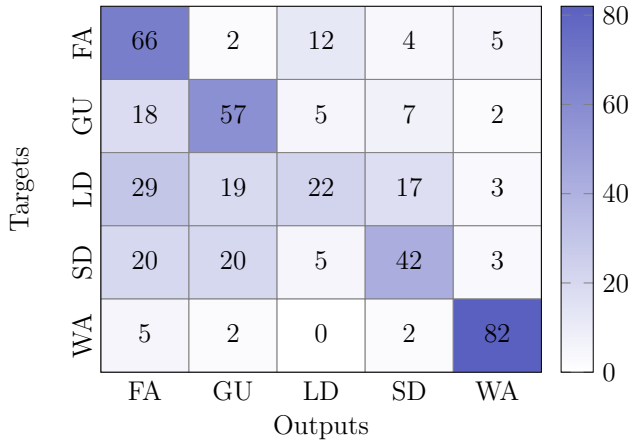

Figure 6.15: Confusion matrix for activity classification with no data augmentation

Table 6.8: Performance metrics per activity with no data augmentation.

Activity	**Precision**	**Recall**	**F1-score**
FA	47.83%	74.16%	58.15%
GU	57.00%	64.04%	60.32%
LD	50.00%	24.44%	32.84%
SD	58.33%	46.67%	51.85%
WA	86.32%	90.11%	88.17%

The x2 DCGAN model presented even a lower classification performance if compared to the model with no data augmentation, obtaining an overall accuracy of 55.68%, while the x3 DCGAN model presented an overall accuracy of 61.69%, a slightly better performance if compared to the same model.

Confusion matrices obtained for the x2 and x3 DCGAN models are shown in Fig. 6.16, while the breakdown of performance metrics obtained for each activity are presented in Table 6.9 and Table 6.10 respectively.

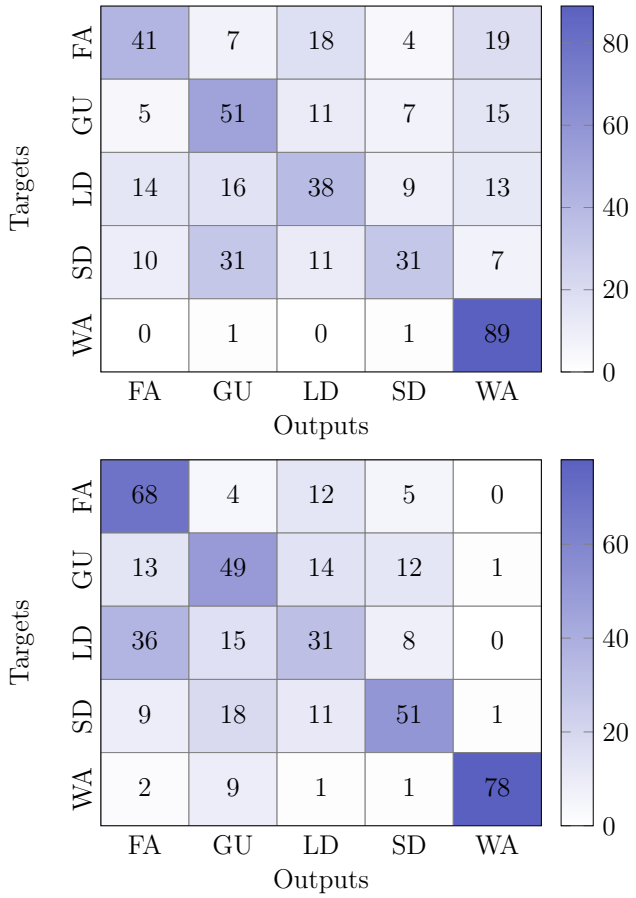

Figure 6.16: Confusion matrices for activity classification with DCGAN-based data augmentation. Top: x2 augmentation; Bottom: x3 augmentation.

Table 6.9: Performance metrics per activity with the x2 DCGAN model.

Activity	Precision	Recall	F1-score
FA	58.57%	46.07%	51.57%
GU	48.11%	57.30%	52.31%
LD	48.72%	42.22%	45.24%
SD	59.62%	34.44%	43.66%
WA	62.24%	97.80%	76.07%

Table 6.10: Performance metrics per activity with the x3 DCGAN model.

Activity	Precision	Recall	F1-score
FA	53.13%	76.40%	62.67%
GU	51.58%	55.06%	53.26%
LD	44.93%	34.44%	38.99%
SD	66.23%	56.67%	61.08%
WA	97.50%	85.71%	91.23%

In contrast to DCGAN models, the EMD models showed that the Bi-LSTM DenseNet121 model performance can be significantly enhanced with the use of the EMD-based data augmentation algorithm, thus proving the effectiveness of the algorithm for generating synthetic CSI amplitudes with quality. The x2 EMD model showed an overall accuracy of 83.74%, also presenting most classification errors between fall and lie down activities, as can be seen in Fig. 6.17. These classification errors are reduced in the x3 and x6 EMD models, which enhances the performance of the model by presenting an overall accuracy of 92.20% and 95.54%, respectively. The breakdown of performance metrics for these models is presented in Table 6.11, Table 6.12, and Table 6.13.

Table 6.11: Performance metrics per activity with EMD-based x2 data augmentation.

Activity	Precision	Recall	F1-score
FA	77.89%	83.15%	80.43%
GU	85.54%	79.78%	82.56%
LD	77.27%	75.56%	76.40%
SD	80.61%	87.78%	84.04%
WA	98.82%	92.31%	95.45%

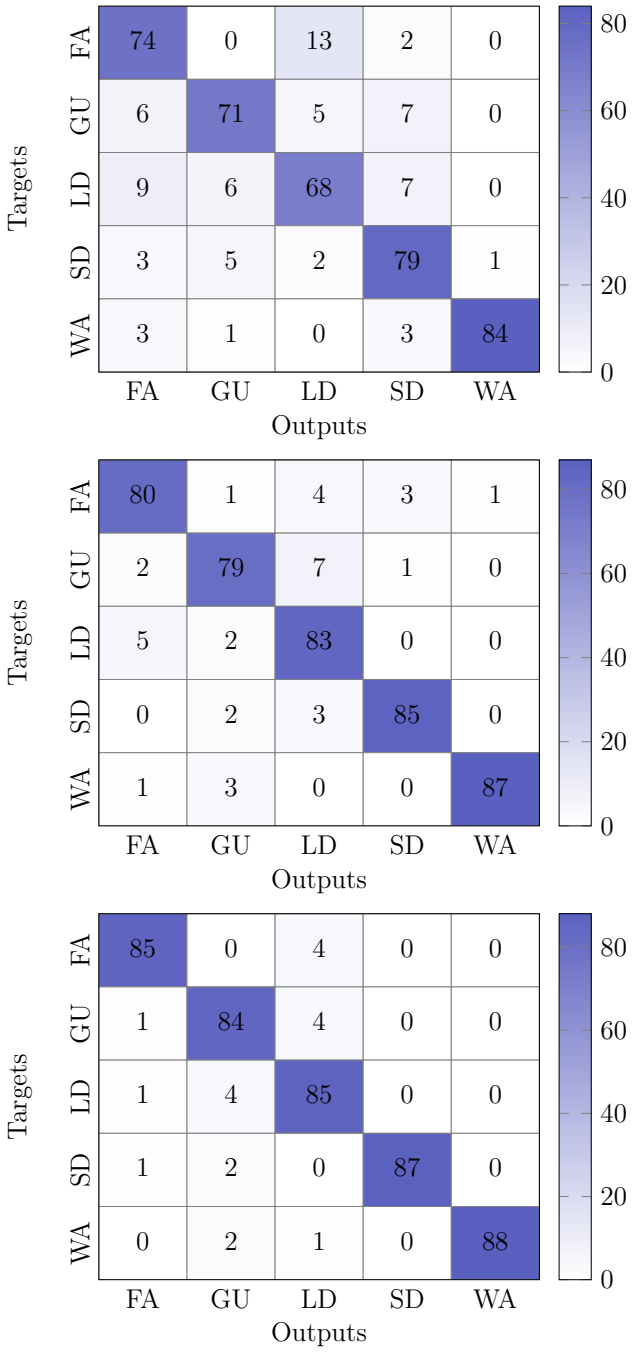

Figure 6.17: Confusion matrices for activity classification with EMD-based data augmentation. Top to bottom: x2, x3, and x6 augmentation.

Table 6.12: Performance metrics per activity with EMD-based x3 data augmentation.

Activity	Precision	Recall	F1-score
FA	90.91%	89.89%	90.40%
GU	90.80%	88.76%	89.77%
LD	85.57%	92.22%	88.77%
SD	95.51%	94.44%	94.97%
WA	98.86%	95.60%	97.21%

Table 6.13: Performance metrics per activity with EMD-based x6 data augmentation.

Activity	Precision	Recall	F1-score
FA	96.59%	95.51%	96.05%
GU	91.30%	94.38%	92.82%
LD	90.43%	94.44%	92.39%
SD	100.00%	96.67%	98.31%
WA	100.00%	96.70%	98.32%

The performance obtained with the DCGAN models reinforces what was mentioned during the analysis of the synthetic samples generated at the start of this section: the DCGAN architecture used to generate the synthetic samples seems to fail to capture the activity into it, as introducing them into model training degrade the model's performance in the case of x2 augmentation, and only slightly improving the model's performance for the x3 augmentation case. Meanwhile, the EMD-based algorithm demonstrates that it is more effective in generating synthetic samples through signal processing techniques, as it outperforms the DCGAN-based generation in both evaluation approaches used.

6.6 Results for Embedded Solution

An ESP32-S3 N16R8 development board with 16 MB of flash memory and 8 MB of external PSRAM, was used for model implementation and for classifying every sample of the test partition. In addition for measuring the performance of the model, the amount of memory used for allocating the model during classification

was measured, as well as the time taken for the model to perform the classification of a single subcarrier of a sample.

As the EMD-based data augmentation algorithm proves to enhance the model performance, two additional training sets were constructed with x6 and x8 data augmentation over for training the DenseNet model described in Table 6.4, with the difference that the number of layers per dense block were halved to reduce computational complexity for the ESP32 and the Bi-LSTM was omitted as it was found that TensorFlow Lite does not support microcontroller optimization for this layer. In addition, this section presents a comparison between an optimized model with integer quantization only to model's weights, maintaining model activations, inputs, and outputs as 32-bit float, and an optimized model with full-integer quantization.

The first model, trained with the x6 augmented training set over 300 epochs, presented an overall accuracy of 91.09% when evaluated using the test partition with 449 samples. Its confusion matrix, presented in Fig. 6.18, shows that Fall activity is the one with the most false positives, having a total of 26. However, only the number of false negative were only two, which were misclassified as lie down due to the similarity between the two activities. A complete breakdown of performance metrics per class is presented in Table 6.14.

Although this model surpasses the 90% of overall accuracy, the time taken by the ESP32 for classifying a single subcarrier was of 918 ms. This means that for classifying a sample consisting of 47 subcarriers, the model would need more than 43 seconds, which is far to be considered near real-time for Human Activity Recognition.

Table 6.14: Performance metrics per activity for the embedded model with weight-only quantization and x6 data augmentation.

Activity	**Precision**	**Recall**	**F1-score**
FA	76.99%	97.75%	86.14%
GU	91.86%	88.76%	90.29%
LD	93.98%	86.67%	90.17%
SD	98.72%	85.56%	91.67%
WA	98.88%	96.70%	97.78%

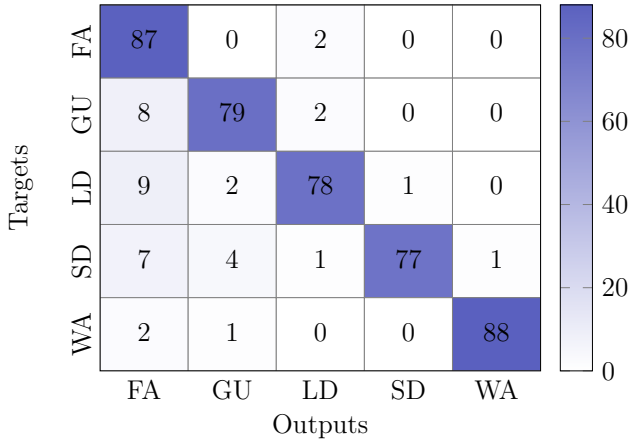

Figure 6.18: Confusion matrix obtained for the embedded model with weight-only quantization and x6 data augmentation.

Following full-integer quantization to uint8, the model exhibited a marginal improvement in overall accuracy, attaining a value of 91.54%. This enhancement is primarily attributable to a higher number of correct classifications for the Sit Down activity, as evidenced by the corresponding confusion matrix (Fig. 6.19) and per-class metrics (Table 6.15). Furthermore, the execution time for classifying a single subcarrier was reduced considerably compared to the previous model, decreasing from 918 ms to 220 ms, and taking 10 seconds to classify a complete sample.

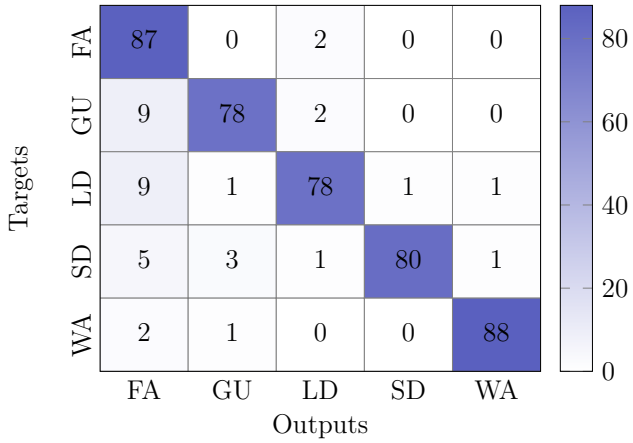

Figure 6.19: Confusion matrix obtained for the embedded model with uint8 quantization and x6 data augmentation.

Table 6.15: Performance metrics per activity for the embedded model with uint8 quantization and x6 data augmentation.

Activity	Precision	Recall	F1-score
FA	77.68%	97.75%	86.57%
GU	93.98%	87.64%	90.70%
LD	93.98%	86.67%	90.17%
SD	98.77%	88.89%	93.57%
WA	97.78%	96.70%	97.24%

In both previous cases, the obtained accuracy was acceptable in relation to the established goals, but the execution time for a sample was not. Therefore, an additional experiment was performed. This experiment involved selecting a representative subcarrier for each sample using the variance-based subcarrier selection method and classifying it with the DenseNet model, which was trained on a partition with x8 data augmentation. This means that the execution time will remain at 220 ms as only a single subcarrier needs to be classified.

The obtained overall accuracy showed worse performance compared to the previous models presented, as evident in its confusion matrix in Fig. 6.20. Its overall accuracy decreased to 86.64%, presenting precisions below 80% for Fall and Walk activities, as can be seen in Table 6.16.

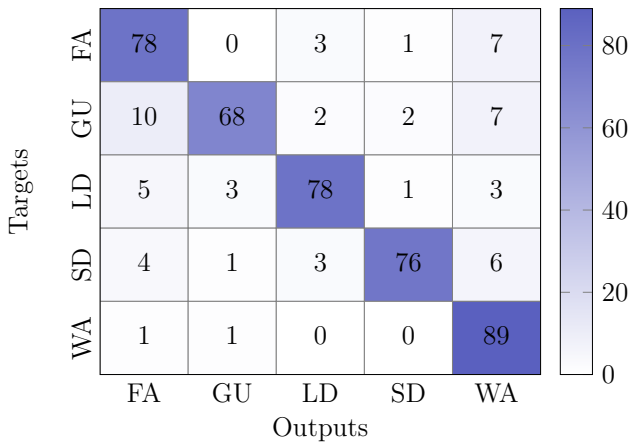

Figure 6.20: Confusion matrix obtained for the embedded model with uint8 quantization, x8 data augmentation, and variance-based subcarrier selection.

Table 6.16: Performance metrics per activity for the embedded model with uint8 quantization, x8 data augmentation, and variance-based subcarrier selection.

Activity	Precision	Recall	F1-score
FA	79.59%	87.64%	83.42%
GU	93.15%	76.40%	83.95%
LD	90.70%	86.67%	88.64%
SD	95.00%	84.44%	89.41%
WA	79.46%	97.80%	87.68%

To improve the performance of the human activity recognition model embedded in the device, it was decided to introduce an LSTM layer with 32 hidden units in place of the Bi-LSTM layer, applying full-integer quantization. With this addition, the overall accuracy of the model with variance-based subcarrier selection raised to 90.65%. The confusion matrix and a breakdown of every metric per class of this DenseNet LSTM-based model are presented in Fig. 6.21 and Table 6.17 respectively.

Moreover, it is important to consider that this performance improvement comes at the cost of a higher computational complexity if compared to the DenseNet-only model. This can be seen in terms of the execution time and the memory used to allocate the model. The execution time is increased by 12 ms, meaning that classifying a sample consisting of a single subcarrier would take 232 ms, which is still acceptable for near real-time functioning for activity recognition. The DenseNet-only model requires 83 kB of memory allocation, which is below 20% of the total allocable memory for development boards with an internal allocable memory of 512 kB. Meanwhile, by introducing the LSTM layer, the memory allocated increased to 126 kB, representing a 24% usage of the total internal memory, which must be considered if the device is also collecting and processing data. However, this increment was not meaningful as the development board being used has an external allocable memory of 8 MB. This last statement can be confirmed by Table 6.18 and Table 6.19, which show the status of the device for the DenseNet-only and embedded LSTM models, respectively. It can be seen that external RAM, which can be identified as *Free SPIRAM*, is not being used as the available internal RAM is sufficient to allocate the models.

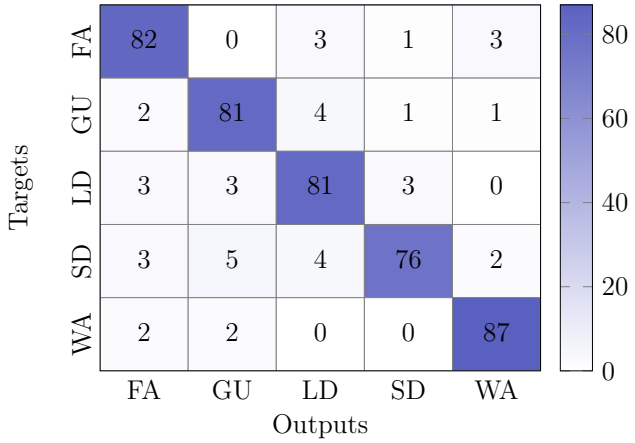

Figure 6.21: Confusion matrix obtained for the embedded LSTM model with uint8 quantization, x8 data augmentation, and selection of a single subcarrier with the variance-based selection method.

Table 6.17: Performance metrics per activity for the embedded LSTM model with uint8 quantization, x8 data augmentation, and selection of a single subcarrier with the variance-based selection method.

Activity	**Precision**	**Recall**	**F1-score**
FA	89.13%	92.13%	90.61%
GU	89.01%	91.01%	90.00%
LD	88.04%	90.00%	89.01%
SD	93.83%	84.44%	88.89%
WA	93.55%	95.60%	94.57%

Table 6.18: Device status for embedded DenseNet-only model

Task Name	**Core Set**	**CPU Time (%)**	**Stack Size**	**Free Stack Size**
IDLE1	CORE1	98	812	-
IDLE0	CORE0	1	804	-
Predictor Task	CORE0	97	8192	5140

Free Internal RAM: 334263 bytes
Free SPIRAM: 8085716 bytes

Table 6.19: Device status for embedded LSTM model

Task Name	Core Set	CPU Time (%)	Stack Size	Free Stack Size
IDLE1	CORE1	99	812	-
IDLE0	CORE0	3	804	-
Predictor Task	CORE0	95	8192	5140

Free Internal RAM: 333935 bytes
Free SPIRAM: 8085716 bytes

6.7 Conclusions

In this chapter, a Deep Learning model was trained for deployment on an ESP32-S3 microcontroller, with its performance enhanced through two data augmentation algorithms. The first technique involved generating synthetic samples using a DCGAN, a GAN model, while the second employed a signal-processing-based approach using EMD. Upon evaluating the performance of both algorithms, the EMD-based approach proved superior, as the generated data exhibited the expected probability distribution. Results obtained by evaluating the performance of the classification models revealed that the overall for the Bi-LSTM Densenet increase from 59.91% to 95.54% when triplicating the amount of samples of the training set by introducing synthetic samples generated with the EMD-based algorithm.

Following the selection of the optimal data augmentation algorithm, multiple Deep Learning models based on a simplified DenseNet architecture were trained using the augmented dataset and subsequently deployed. The best-performing model achieved an overall accuracy of 90.65% in recognizing five distinct activities: Fall, Get Up, Lie Down, Sit Down, and Walk. On the microcontroller, the model demonstrated an execution time of 232 ms while occupying only 126 kB of memory, enabling near real-time operation. For optimal performance, it is recommended that the deployment device include allocable external memory.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

Based on the results obtained during the development of this investigation, the following particular conclusions were drawn:

1. The review of state-of-the-art Wi-Fi CSI-based sensing applications revealed key insights into standard processing techniques. These applications leverage signal processing to mitigate noise and extract relevant features. Furthermore, signal analysis techniques, such as FFT, PSD, and peak detection algorithms, are employed alongside machine and deep learning models to perform detection tasks. The findings demonstrate that CNN and LSTM-based architectures are among the most prevalent, proving their effectiveness in extracting spatio-temporal dependencies from CSI data, particularly for HAR and Gesture Recognition. However, a significant practical challenge was identified: CSI data collection often relies on specific legacy Wi-Fi devices that require custom firmware. This frequently necessitates acquiring devices from third-party or second-hand markets, which adversely affects the scalability of these sensing systems.

2. The proposed Embedded CRISP-DM data pipeline was leveraged in a proof-of-concept, demonstrating that end-to-end HAR is feasible on resource-constrained embedded devices. By adapting the CRISP-DM methodology with hardware-aware tasks, this pipeline enabled the deployment of every

Deep Learning model presented in Chapter 6. Its successful implementation for Wi-Fi CSI-based HAR on ESP32 microcontrollers confirms the pipeline's practicality and replicability for embedded AI applications using this sensing technology and devices.

3. To facilitate the construction of a HAR training dataset, this work developed the ESP32 CSI Web Collecting Tool. This tool enables in-field configuration without source code modification using low-cost commercial devices, the ESP32 microcontroller. A key contribution is its superior performance, achieved by leveraging the ESP32's dual-core architecture and transmitting data in a raw binary format, which enables higher packet rates than existing ESP32-based collectors. The tool's versatility is further demonstrated through three configurable transmission modes: USB, UART serial via GPIO, and local storage to an SD card.

4. A Bi-LSTM DenseNet121 model was trained evaluated for HAR with Wi-Fi CSI. The performance of this model was significantly enhanced from 59.91% to 95.54% accuracy by leveraging an EMD-based data augmentation algorithm, which was first proposed for processing EEG signals. By tripling the training dataset with synthetically generated samples, the model's generalization was greatly improved for recognizing five activities: Fall, Get Up, Lie Down, Sit Down, and Walk. The hold-out validation method, which used data from unseen volunteers for testing, confirms the robustness of this approach.

5. The results obtained demonstrate that it is possible to implement a reliable Deep Learning model in microcontrollers, specifically the ESP32, and thus develop an embedded Wi-Fi CSI-based HAR system for these devices, despite their limited hardware constraints. This was achieved by an optimized DenseNet-based model for embedded devices, which presented an overall accuracy of 90.65% for HAR of the same five activities. Its implementation in the embedded device revealed that it needs 232 ms for processing a single CSI sample consisting of 850 amplitude values, which can be considered

near real-time functioning for HAR with Wi-Fi CSI. Moreover, this optimized model requires only 126 kB for allocation in ESP32 devices, making it suitable for devices with RAM constraints, as it is less than 25% of the available memory of commercial ESP32 devices with no external PSRAM.

7.2 Future Work and Lines of Investigation

7.2.1 Future Work

As future work, the next tasks are considered to be addressed to extend the results presented:

1. Train a Deep Learning model with all synthetic samples generated with the EMD-based data augmentation algorithm, which was not possible due to memory limitation in the computer being used for training the model.

2. Implement to the ESP32 CSI Web Collecting tool Octa-SPI communication with DMA for device-to-device communication for transmitting CSI. This will enable the model to run on another ESP32, which will function as the system's Neural Processing Unit (NPU).

3. Explore the use of the ESP32-P4 and STMN6x7 series microcontrollers to define which will be implemented as the NPU-based latency of the Deep Learning model implemented in the microcontroller.

4. Design and assembly of a development board for Wi-Fi sensing applications that enables CSI collecting and processing in the edge with microcontrollers.

5. Evaluate the performance of the system with people over 60 years old, as they have been identified as the highest-risk group for fatal falls.

7.2.2 Lines of Investigation

The following lines of investigation arose based on the lessons learned and results obtained:

1. An open line of investigation arises from the privacy and security vulnerabilities exposed by the passive sniffing of Wi-Fi CSI. Off-the-shelf Wi-Fi devices can act as passive sniffers to capture CSI data from any transmitting device within range, including devices such as smartphones, laptops, and IoT sensors, without requiring any association or connection to the network. This capability transforms ordinary Wi-Fi signals into an unregulated surveillance medium, enabling third-party adversaries to infer detailed human activities with alarming accuracy. Thus, the open research challenge is the development of methods and tools for privacy preserving by obfuscating CSI.

2. The necessity of transitioning from performing Wi-Fi sensing in controlled environments to real-world applications involves a comprehensive study of its performance under complex, dynamic, and multi-variable environmental conditions. The work presented in this document, like much of the foundational research in this field, has primarily focused on simplified scenarios. To achieve practical deployment, future works can be centered in the collection, analysis, and characterization of how different environmental conditions degrade the performance for HAR with Wi-Fi CSI, ultimately leading to the development of possible ease of use metrics for Wi-Fi sensing based on the surroundings, e.g., occupant density and spectrum congestion.

3. A promising area of research involves expanding data augmentation methods for Wi-Fi Channel State Information (CSI) and time-series data. This includes going beyond the two methods examined in this study to conduct a thorough comparison of various techniques. Additionally, new methods can be developed using advanced generative models, such as Transformers, to synthesize time-series data.

4. A future direction involves developing an open-source toolbox to streamline the conversion of deep learning models into hardware description languages (HDL), leveraging FPGAs as reconfigurable, low-latency, and energy-efficient AI accelerators to enable robust edge computing without reliance on cloud connectivity.

Appendix A

Structured Literature Review Supplementary

A.1 Search String

(TITLE-ABS-KEY(“Channel State Information”) AND TITLE-ABS-KEY(“Wi-Fi”))
AND
(TITLE-ABS-KEY(sensing OR application OR monitoring))
AND
(LIMIT-TO(DOCTYPE, “ar” OR “cp” OR “ch”))
AND
(PUBYEAR >= 2015 AND PUBYEAR <= 2024)

Appendix B

ESP32 CSI Web Collecting Tool Supplementary

B.1 Device Status

Table B.1: Device status when waiting for tool configuration.

Task Name	Core Set	CPU Time (%)	Stack Size	Free Stack Size
IDLE1	CORE1	98	924	-
IDLE0	CORE0	96	912	-
Default Wi-Fi[1]	CORE0	2	5376	4168
Wi-Fi Task	CORE0	<1	9152	6820

Free Internal RAM: 186867 bytes
Free SPIRAM: 4177588 bytes
[1] Default Wi-Fi task is created by default by the device to handle Wi-Fi events.

Table B.2: Device status when Tx is sending UDP packets at 50 packets/s.

Task Name	Core Set	CPU Time (%)	Stack Size	Free Stack Size
IDLE1	CORE1	98	924	-
IDLE0	CORE0	94	912	-
Default Wi-Fi[1]	CORE0	6	5376	3660
Wi-Fi Task	CORE0	<1	9152	6820
CSI Task	CORE1	1	8192	6272

Free Internal RAM: 186867 bytes
Free SPIRAM: 4177588 bytes
[1] Default Wi-Fi task is created by default by the device to handle Wi-Fi events.

Table B.3: Device status when Rx sends received packets through USB port at a rate of 50 packets/s.

Task Name	Core Set	CPU Time (%)	Stack Size	Free Stack Size
IDLE1	CORE1	68	924	-
IDLE0	CORE0	92	912	-
Default Wi-Fi[1]	CORE0	6	5376	4024
Wi-Fi Task	CORE0	<1	9152	6816
CSI Task	CORE1	<1	8192	6284
Informer Task	CORE1	29	8192	5996

Free Internal RAM: 172323 bytes
Free SPIRAM: 4177724 bytes
[1] Default Wi-Fi task is created by default by the device to handle Wi-Fi events.

B.2 Message structures

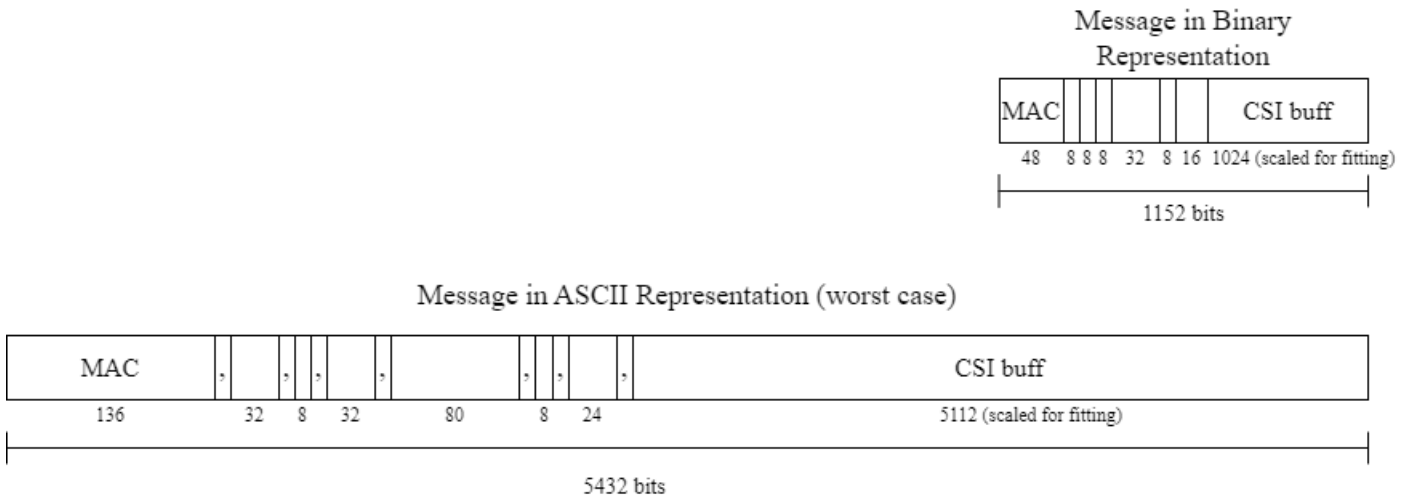

Figure B.1: Message structures when the tool is configured to send CSI in ASCII and binary formats.

Appendix C

Scholarly Output

C.1 Conferences

- Garcia-Reyes, C. I., Armenta-Garcia, J. A., Gonzalez-Navarro, F. F., Caro-Gutierrez, J., & Ibarra-Esquer, J. I. (2024). Human Activity Recognition on the Edge with Wi-Fi Channel State Information. 2024 Mexican International Conference on Computer Science (ENC), 1-12.

- Torres-Cerda, M. A., Gonzalez-Navarro, F. F., Caro-Gutierrez, J., & Armenta-Garcia, J. A. (2023). CSI crowd-counting: An experimental study using Machine Learning and Deep Learning Algorithms. 2023 Mexican International Conference on Computer Science (ENC), 1-6. https://doi.org/10.1109/ENC60556.2023.10508604

- Armenta-Garcia, J. A., Gonzalez-Navarro, F. F., & Caro-Gutierrez, J. (2022). WiFi CSI Subcarrier Selection Methods for Sensing Applications. 2022 Computer Science, Computer Engineering, & Applied Computing (CSCE), 1-6.

C.2 Journal Papers

- Armenta-Garcia, J. A., Gonzalez-Navarro, F. F., & Caro-Gutierrez, J. (2024). Wireless sensing applications with Wi-Fi Channel State Information, preprocessing techniques, and detection algorithms: A survey. Com-

puter Communications, 224, 254-274. https://doi.org/https://doi.org/10.1016/j.comcom.2024.06.011

- Armenta-Garcia, J. A., Gonzalez-Navarro, F. F., Caro-Gutierrez, J., Galaviz-Yanez, G., Ibarra-Esquer, J. E., & Flores-Fuentes, W. (2023). Mining Wi-Fi Channel State Information for breathing and heart rate classification. Pervasive and Mobile Computing, 91, 101768. https://doi.org/https://doi.org/10.1016/j.pmcj.2023.101768

- Armenta-Garcia, J. A., Gonzalez-Navarro, F. F., & Caro-Gutierrez, J., & Garcia-Reyes, C. I. (2025). Tools and Methods for Achieving Wi-Fi Sensing in Embedded Devices. Sensors. (Second Round of Revision)

C.3 Science Outreach Publications

- Armenta-Garcia, J. A., Gonzalez-Navarro, F. F., Ibarra-Esquer, J. E., & Caro-Gutierrez, J. (2023). Detección inalámbrica de frecuencia respiratoria utilizando señales Wi-Fi y aprendizaje automático. Revista Ciencia UANL, 26(121), 45-50.

- Armenta-Garcia, J. A., Gonzalez-Navarro, F. F., Ibarra-Esquer, J. I., & Caro-Gutierrez, J. (2023). Identificación de Frecuencias Respiratorias Anormales Utilizando Señales Wi-Fi y Aprendizaje Automático. Komputer Sapiens, 1, 19-24.

Copyright Registrations

- Software register in INDAUTOR titled “ESP32 CSI Web Collecting Tool: Herramienta para la Recolección de Wi-Fi CSI con Dispositivos ESP-32” (2024).

- Software register in INDAUTOR titled “ESP32 CSI Web Collecting Tool: Plataforma Web para la Configuración de una Herramienta de Recolección de Wi-Fi CSI” (2024).

- Software register in INDAUTOR titled "Wi-Fi CSI-PY: Herramienta de Recepción y Transformación de Información de Estado del Canal Wi-Fi Recolectada a partir de Dispositivos ESP-32" (2023).

Outreach Workshops

- STEAM workshop titled "Imaginación Artificial" at *Festival Pequeño Gran Cimarroón 2025*
- STEAM workshop titled "Imaginación Artificial" at *Fiestas del Sol 2024*
- STEAM workshop titled "Imaginación Artificial" at *Festival Pequeño Gran Cimarroón 2024*
- STEAM workshop titled "Cimarrón a la Cima" at *Fiestas del Sol 2023*
- STEAM workshop at *Festival Aeroespacial 2023*.
- STEAM workshop at *Festival Aeroespacial 2022*.

References

Al-qaness, M.A.A., Abd Elaziz, M., Kim, S., Ewees, A.A., Abbasi, A.A., Alhaj, Y.A. & Hawbani, A. (2019). Channel state information from pure communication to sense and track human motion: A survey. *Sensors*, **19**. 8, 10

Ali, K., Liu, A.X., Wang, W. & Shahzad, M. (2017). Recognizing keystrokes using wifi devices. *IEEE Journal on Selected Areas in Communications*, **35**, 1175–1190. 17, 19, 42, 44, 45

Alzaabi, A., Arslan, T. & Polydorides, N. (2024). Non-contact wi-fi sensing of respiration rate for older adults in care: A validity and repeatability study. *IEEE Access*, **12**, 6400–6412. 21, 24, 42, 44, 45, 64

Arshad, S., Feng, C., Liu, Y., Hu, Y., Yu, R., Zhou, S. & Li, H. (2017). Wi-chase: A wifi based human activity recognition system for sensorless environments. In *2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 1–6. 12, 15, 42, 44, 45

Atif, M., Muralidharan, S., Ko, H. & Yoo, B. (2022). COVID-Beat: a low-cost breath monitoring approach for people in quarantine during the pandemic. *Journal of Computational Design and Engineering*, **9**, 992–1006. 21, 24, 42, 44, 45

Awon, N.T., Islam, M.A., Rahman, M.M. & Islam, A.Z.M.T. (2012). Effect of AWGN & fading (raleigh & rician) channels on BER performance of a wimax communication system. *CoRR*, **abs/1211.4294**. 106

Barry, R. (2016). Mastering the freertos real time kernel. 81

Biguesh, M. & Gershman, A. (2006). Training-based mimo channel estimation: a study of estimator tradeoffs and optimal training signals. *IEEE Transactions on Signal Processing*, **54**, 884–893. 46

Bu, Q., Yang, G., Ming, X., Zhang, T., Feng, J. & Zhang, J. (2022). Deep transfer learning for gesture recognition with WiFi signals. *Personal and Ubiquitous Computing*, **26**, 543–554. 17, 19, 42, 44, 45

Burimas, R., Horanont, T., Thapa, A. & Lamichhane, B.R. (2024). Monitoring the sleep respiratory rate with low-cost microcontroller wi-fi in a controlled environment. *Applied Sciences*, **14**. 65

Chen, C., Han, Y., Chen, Y., Lai, H.Q., Zhang, F., Wang, B. & Liu, K.J.R. (2018). Tr-breath: Time-reversal breathing rate estimation and detection. *IEEE Transactions on Biomedical Engineering*, **65**, 489–501. 22, 24, 42, 44, 45

Chen, H., Zhang, Y., Li, W., Tao, X. & Zhang, P. (2017). Confi: Convolutional neural networks based indoor wi-fi localization using channel state information. *IEEE Access*, **5**, 18066–18074. 25, 28, 42, 44, 45

Chen, H.C. & Chen, S.W. (2003). A moving average based filtering system with its application to real-time qrs detection. In *Computers in Cardiology, 2003*, 585–588. 52

Chen, X., Zou, Y., Li, C. & Xiao, W. (2024). A deep learning based lightweight human activity recognition system using reconstructed wifi csi. *IEEE Transactions on Human-Machine Systems*, **54**, 68–78. 14, 15, 42, 44, 45

Chen, Y., Ou, R., Li, Z. & Wu, K. (2022). Wiface: Facial expression recognition using wi-fi signals. *IEEE Transactions on Mobile Computing*, **21**. 17, 19, 42, 44, 45

Cheng, Y.K. & Chang, R.Y. (2017). Device-free indoor people counting using wi-fi channel state information for internet of things. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 1–6. 29, 31, 42, 44, 45

Choi, H., Fujimoto, M., Matsui, T., Misaki, S. & Yasumoto, K. (2022). Wi-cal: Wifi sensing and machine learning based device-free crowd counting and localization. *IEEE Access*, **10**, 24395–24410. 30, 31, 42, 44, 45

Cui, W., Li, B., Zhang, L. & Chen, Z. (2021). Device-free single-user activity recognition using diversified deep ensemble learning. *Applied Soft Computing*, **102**, 107066. 12, 15, 42, 44, 45

Damodaran, N., Haruni, E., Kokhkharova, M. & Schäfer, J. (2020). Device free human activity and fall recognition using wifi channel state information (csi). *CCF Transactions on Pervasive Computing and Interaction*, **2**, 1–17. 1, 12, 15, 29, 31, 42

Daubechies, I. (1992). *Ten lectures on wavelets*. SIAM. 53

Dou, C. & Huan, H. (2021). Full respiration rate monitoring exploiting doppler information with commodity wi-fi devices. *Sensors*, **21**. 21, 24, 42, 44, 45

Fard Moshiri, P., Shahbazian, R., Nabati, M. & Ghorashi, S.A. (2021). A csi-based human activity recognition using deep learning. *Sensors*, **21**. 72

Floyd, T.L. (2008). *Dispositivos electrónicos*, 756–760. Pearson Education, México, 8th edn. 52

Fu, Z., Xu, J., Zhu, Z., Liu, A.X. & Sun, X. (2019). Writing in the air with wifi signals for virtual reality devices. *IEEE Transactions on Mobile Computing*, **18**, 473–484. 18, 19, 42, 44, 45

Ge, Y., Taha, A., Shah, S.A., Dashtipour, K., Zhu, S., Cooper, J., Abbasi, Q.H. & Imran, M.A. (2023). Contactless wifi sensing and monitoring for future healthcare - emerging trends, challenges, and opportunities. *IEEE Reviews in Biomedical Engineering*, **16**, 171–191. 8, 10

Gers, F., Schmidhuber, J. & Cummins, F. (1999). Learning to forget: continual prediction with lstm. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, vol. 2, 850–855 vol.2. 62

Gers, F.A., Schraudolph, N.N. & Schmidhuber, J. (2003). Learning precise timing with lstm recurrent networks. *J. Mach. Learn. Res.*, **3**, 115–143. 63

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning*. The MIT Press. 60, 62, 63

Gringoli, F., Cominelli, M., Blanco, A. & Widmer, J. (2021). Ax-csi: Enabling csi extraction on commercial 802.11ax wi-fi platforms. In *Proceedings of the 15th ACM Workshop on Wireless Network Testbeds, Experimental Evaluation & CHaracterization*, WiNTECH '21, 46–53, Association for Computing Machinery, New York, NY, USA. 50

Gu, Y., Yan, H., Dong, M., Wang, M., Zhang, X., Liu, Z. & Ren, F. (2021). Wione: One-shot learning for environment-robust device-free user authentication via commodity wi-fi in man-machine system. *IEEE Transactions on Computational Social Systems*, **8**, 630–642. 39, 40, 42, 44, 45

Gu, Y., Yan, H., Zhang, X., Wang, Y., Huang, J., Ji, Y. & Ren, F. (2023). Attention-based gesture recognition using commodity wifi devices. *IEEE Sensors Journal*, **23**. 16, 19, 42, 44, 45

Gui, L., Ma, C., Sheng, B., Guo, Z., Cai, J. & Xiao, F. (2023). In-home monitoring sleep turnover activities and breath rate via wifi signals. *IEEE Systems Journal*, **17**, 2355–2365. 21, 24, 42, 44, 45

Guo, L., Lu, Z., Wen, X., Zhou, S. & Han, Z. (2020). From signal to image: Capturing fine-grained human poses with commodity wi-fi. *IEEE Communications Letters*, **24**, 802–806. 32, 33, 42, 44, 45

Guo, X., Liu, B., Shi, C., Liu, H., Chen, Y. & Chuah, M.C. (2017). Wifi-enabled smart human dynamics monitoring. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, SenSys '17, Association for Computing Machinery, New York, NY, USA. 30, 31, 42, 44, 45

Guo, Z., Xiao, F., Sheng, B., Sun, L. & Yu, S. (2022). Twcc: A robust through-the-wall crowd counting system using ambient wifi signals. *IEEE Transactions on Vehicular Technology*, **71**, 4198–4211. 30, 31, 42, 44, 45

Halperin, D., Hu, W., Sheth, A. & Wetherall, D. (2011). Tool release: Gathering 802.11n traces with channel state information. *SIGCOMM Comput. Commun. Rev.*, **41**, 53. 49

Hang, T., Zheng, Y., Qian, K., Wu, C., Yang, Z., Zhou, X., Liu, Y. & Chen, G. (2019). Wish: Wifi-based real-time human detection. *Tsinghua Science and Technology*, **24**, 615–629. 36, 37, 42, 45

Hao, Z., Duan, Y., Dang, X., Liu, Y. & Zhang, D. (2020). Wi-sl: Contactless fine-grained gesture recognition uses channel state information. *Sensors*, **20**. 18, 19, 42, 44, 45

He, Y., Chen, Y., Hu, Y. & Zeng, B. (2020). Wifi vision: Sensing, recognition, and detection with commodity mimo-ofdm wifi. *IEEE Internet of Things Journal*, **7**, 8296–8317. 8, 10

Hernandez, S.M. & Bulut, E. (2020). 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. 50, 70, 78

Hernandez, S.M. & Bulut, E. (2023). Wifi sensing on the edge: Signal processing techniques and challenges for real-world systems. *IEEE Communications Surveys & Tutorials*, **25**, 46–76. 9, 10

Hernandez, S.M., Touhiduzzaman, M., Pidcoe, P.E. & Bulut, E. (2022). Wi-pt: Wireless sensing based low-cost physical rehabilitation tracking.

In *2022 IEEE International Conference on E-health Networking, Application & Services (HealthCom)*, 113–118. 13, 15, 18, 19, 42, 44, 45

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, **9**, 1735–1780. 62

Iglesias, G., Talavera, E., González-Prieto, Á., Mozo, A. & Gómez-Canaval, S. (2023). Data augmentation techniques in time series domain: a survey and taxonomy. *Neural Computing and Applications*, **35**, 10123–10145. 97

Jia, L., Gu, Y., Cheng, K., Yan, H. & Ren, F. (2020). Beaware: Convolutional neural network(cnn) based user behavior understanding through wifi channel state information. *Neurocomputing*, **397**, 457–463. 16, 19, 42, 44, 45

Jiang, W., Xue, H., Miao, C., Wang, S., Lin, S., Tian, C., Murali, S., Hu, H., Sun, Z. & Su, L. (2020). Towards 3d human pose construction using wifi. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, MobiCom '20, Association for Computing Machinery, New York, NY, USA. 32, 33, 42, 44, 45

Jolliffe, I.T. & Cadima, J. (2016). Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **374**, 20150202. 57

Kamen, E.W. & Heck, B.S. (2007). *Fundamentals of Signals and Systems using the Web and MATLAB*. Pearson Prentice Hall, Upper Saddle River, NJ, 3rd edn. 56

Kim, E., Helal, S. & Cook, D. (2010). Human activity recognition and pattern discovery. *IEEE Pervasive Computing*, **9**, 48–53. 11

Kim, M., Han, D. & Rhee, J.K.K. (2021). Multiview variational deep learning with application to practical indoor localization. *IEEE Internet of Things Journal*, **8**, 12375–12383. 26, 28, 42, 44, 45

Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M. & Inman, D.J. (2021). 1d convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing*, **151**, 107398. 61

Kitagishi, T., Hangli, G., Michikata, T. & Koshizuka, N. (2022). Wi-monitor: Wi-fi channel state information-based crowd counting with lightweight and low-cost iot devices. In A. González-Vidal, A. Mohamed Abdelgawad, E. Sabir, S. Ziegler & L. Ladid, eds., *Internet of Things*, 135–148, Springer International Publishing, Cham. 30, 31, 42, 44, 45

Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele Univ.*, **33**. 4

Lee, H., Ahn, C.R. & Choi, N. (2020). Fine-grained occupant activity monitoring with wi-fi channel state information: Practical implementation of multiple receiver settings. *Advanced Engineering Informatics*, **46**, 101147. 14, 15, 42, 44, 45

Lee, S., Park, Y.D., Suh, Y.J. & Jeon, S. (2018). Design and implementation of monitoring system for breathing and heart rate pattern using wifi signals. In *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 1–7. 22, 24, 42, 44

Li, F., Al-qaness, M.A.A., Zhang, Y., Zhao, B. & Luan, X. (2016). A robust and device-free system for the recognition and classification of elderly activities. *Sensors*, **16**. 11, 15, 42, 44, 45

Li, H., Yang, W., Wang, J., Xu, Y. & Huang, L. (2016). Wifinger: Talk to your smart devices with finger-grained gesture. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp 16, Association for Computing Machinery, New York, NY, USA. 16, 19, 42, 44, 45

Li, X., Li, S., Zhang, D., Xiong, J., Wang, Y. & Mei, H. (2016). Dynamic-music: accurate device-free indoor localization. In *Proceedings of the 2016 ACM*

International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp '16, 196–207, Association for Computing Machinery, New York, NY, USA. 26, 28, 42, 44, 45

Lin, C.L., Chang, W.J. & Tu, G.H. (2022). Wi-fi-based tracking of human walking for home health monitoring. *IEEE Internet of Things Journal*, **9**, 8935–8942. 34, 35, 42, 44, 45

Liu, J., Wang, Y., Chen, Y., Yang, J., Chen, X. & Cheng, J. (2015). 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*, MobiHoc '15, 267–276, Association for Computing Machinery, New York, NY, USA. 58

Liu, J., Chen, Y., Wang, Y., Chen, X., Cheng, J. & Yang, J. (2018). Monitoring vital signs and postures during sleep using wifi signals. *IEEE Internet of Things Journal*, **5**, 2071–2084. 22, 24, 42, 44, 45

Liu, J., Zeng, Y., Gu, T., Wang, L. & Zhang, D. (2021). Wiphone: Smartphone-based respiration monitoring using ambient reflected wifi signals. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, **5**. 20, 24, 42, 44, 45

Liu, M., Zhang, L., Yang, P., Lu, L. & Gong, L. (2019). Wi-run: Device-free step estimation system with commodity wi-fi. *Journal of Network and Computer Applications*, **143**, 77–88. 33, 35, 42, 44, 45

Liu, S., Zhao, Y. & Chen, B. (2017). Wicount: A deep learning approach for crowd counting using wifi signals. In *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*, 967–974. 29, 31, 42, 44, 45

Liu, X., Cao, J., Tang, S., Wen, J. & Guo, P. (2016). Contactless respiration monitoring via off-the-shelf wifi devices. *IEEE Transactions on Mobile Computing*, **15**, 2466–2479. 20, 24, 42, 44, 45

Liu, Y., Wang, T., Jiang, Y. & Chen, B. (2022). Harvesting ambient rf for presence detection through deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, **33**, 1571–1583. 36, 37, 42, 44, 45

Lv, J., Yang, W. & Man, D. (2017). Device-free passive identity identification via wifi signals. *Sensors*, **17**. 38, 40, 42, 44, 45

Ma, X., Xi, W., Zhao, X., Chen, Z., Zhang, H. & Zhao, J. (2022). Wisual: Indoor crowd density estimation and distribution visualization using wi-fi. *IEEE Internet of Things Journal*, **9**, 10077–10092. 30, 31, 42, 44, 45

Ma, Y., Zhou, G. & Wang, S. (2019). Wifi sensing with channel state information: A survey. *ACM Comput. Surv.*, **52**. 7, 10, 47

Meneghello, F., Garlisi, D., Fabbro, N.D., Tinnirello, I. & Rossi, M. (2023). Sharp: Environment and person independent activity recognition with commodity ieee 802.11 access points. *IEEE Transactions on Mobile Computing*, **22**, 6160–6175. 14, 15, 42, 44, 45

Meng, W., Chen, X., Cui, W. & Guo, J. (2022). Wihgr: A robust wifi-based human gesture recognition system via sparse recovery and modified attention-based bgru. *IEEE Internet of Things Journal*, **9**, 10272–10282. 16, 19, 42, 44, 45

Mesa-Cantillo, C.M., SÃ¡nchez-RodrÃguez, D., Alonso-GonzÃ¡lez, I., Quintana-SuÃ¡rez, M.A., Ley-Bosch, C. & Alonso-HernÃ¡ndez, J.B. (2023). A non intrusive human presence detection methodology based on channel state information of wi-fi networks. *Sensors*, **23**. 35, 37, 42, 44, 45

Mosleh, S., Coder, J.B., Scully, C.G., Forsyth, K. & Kalaa, M.O.A. (2022). Monitoring respiratory motion with wi-fi csi: Characterizing performance and the breathesmart algorithm. *IEEE Access*, **10**, 131932–131951. 20, 24, 42, 44, 45

Nakamura, T., Bouazizi, M., Yamamoto, K. & Ohtsuki, T. (2022). Wi-fi-based fall detection using spectrogram image of channel state information. *IEEE Internet of Things Journal*, **9**, 17220–17234. 12, 15, 42, 44, 45

Natarajan, A., Krishnasamy, V. & Singh, M. (2024). Design of a low-cost and device-free human activity recognition model for smart led lighting control. *IEEE Internet of Things Journal*, **11**, 5558–5567. 12, 15, 42, 44, 45, 64

Niu, K., Zhang, F., Xiong, J., Li, X., Yi, E. & Zhang, D. (2018). Boosting fine-grained activity sensing by embracing wireless multipath effects. In *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '18, Association for Computing Machinery, New York, NY, USA. 17, 19, 24, 42, 44, 45

Niu, K., Zhang, F., Jiang, Y., Xiong, J., Lv, Q., Zeng, Y. & Zhang, D. (2019). Wimorse: A contactless morse code text input system using ambient wifi signals. *IEEE Internet of Things Journal*, **6**, 9993–10008. 17, 19, 42, 44, 45

Oppenheim, A.V. & Schafer, R.W. (2021). *Discrete-Time Signal Processing*. Pearson Prentice Hall, Georgia Institute of Technology, 3rd edn. 56

Pearson, R. (2002). Outliers in process modeling and identification. *IEEE Transactions on Control Systems Technology*, **10**, 55–63. 54

Regani, S.D., Xu, Q., Wang, B., Wu, M. & Liu, K.J.R. (2020). Driver authentication for smart car using wireless sensing. *IEEE Internet of Things Journal*, **7**, 2235–2246. 38, 40, 42, 44, 45

Ren, Y., Wang, Z., Wang, Y., Tan, S., Chen, Y. & Yang, J. (2022). Gopose: 3d human pose estimation using wifi. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, **6**. 32, 33, 42, 44, 45

Schafer, R.W. (2011). What is a savitzky-golay filter? [lecture notes]. *IEEE Signal Processing Magazine*, **28**, 111–117. 55

Schulz, M., Wegemer, D. & Hollick, M. (2017). Nexmon: The c-based firmware patching framework. 50

Shah, S.A., Tahir, A., Ahmad, J., Zahid, A., Pervaiz, H., Shah, S.Y., Abdulhadi Ashleibta, A.M., Hasanali, A., Khattak, S. & Abbasi, Q.H. (2020). Sensor fusion for identification of freezing of gait episodes using wi-fi and radar imaging. *IEEE Sensors Journal*, **20**, 14410–14422. 34, 35, 42, 44, 45

Shah, S.W. & Kanhere, S.S. (2019). Smart user identification using cardiopulmonary activity. *Pervasive and Mobile Computing*, **58**, 101024. 38, 40, 42, 44, 45

Shao, S., Fan, M., Yu, C., Li, Y., Xu, X. & Wang, H. (2022). Machine learning-assisted sensing techniques for integrated communications and sensing in wlans : Current status and future directions. *Progress In Electromagnetics Research*, **175**, 45–79. 8, 10

Sharma, A., Jiang, W., Mishra, D., Jha, S. & Seneviratne, A. (2022). Optimised cnn for human counting using spectrograms of probabilistic wifi csi. In *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 01–06. 29, 31, 42, 44, 45

Shi, D., Lu, J., Wang, J., Li, L., Liu, K. & Pan, M. (2020). No one left behind: Avoid hot car deaths via wifi detection. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 1–6. 36, 37, 42, 44, 45

Shi, S., Sigg, S., Chen, L. & Ji, Y. (2018). Accurate location tracking from csi-based passive device-free probabilistic fingerprinting. *IEEE Transactions on Vehicular Technology*, **67**, 5217–5230. 26, 28, 42, 44, 45

Showmik, I.A., Sanam, T.F. & Imtiaz, H. (2023). Human activity recognition from wi-fi csi data using principal component-based wavelet cnn. *Digital Signal Processing*, **138**, 104056. 13, 15, 42, 44, 45

Simonyan, K. & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. 69

Soto, J.C., Galdino, I., Caballero, E., Ferreira, V., Muchaluat-Saade, D. & Albuquerque, C. (2022a). A survey on vital signs monitoring based on wi-fi csi data. *Computer Communications*, **195**, 99–110. 8, 10

Soto, J.C.H., Galdino, I., Gouveia, B.G., Caballero, E., Ferreira, V., Muchaluat-Saade, D. & Albuquerque, C. (2022b). Wi-fi csi-based human presence detection using dtw features and machine learning. In *2022 IEEE Latin-American Conference on Communications (LATINCOM)*, 1–6. 35, 37, 42, 44, 45

Tan, Q., Han, C., Sun, L., Guo, J. & Zhu, H. (2018). A csi frequency domain fingerprint-based method for passive indoor human detection. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 1832–1837. 34, 37, 42, 45

Tan, S., Ren, Y., Yang, J. & Chen, Y. (2022). Commodity wifi sensing in ten years: Status, challenges, and opportunities. *IEEE Internet of Things Journal*, **9**, 17832–17843. 8, 10

Tian, L., Chen, L., Xu, Z. & Chen, Z.D. (2021). A people-counting and speed-estimation system using wi-fi signals. *Sensors*, **21**. 29, 31, 42, 44, 45

Tian, Z., Wang, J., Yang, X. & Zhou, M. (2018). Wicatch: A wi-fi based hand gesture recognition system. *IEEE Access*, **6**, 16911–16923. 16, 19, 42, 44, 45

Tong, G., Li, Y., Zhang, H. & Xiong, N. (2023). A fine-grained channel state information-based deep learning system for dynamic gesture recognition. *Information Sciences*, **636**, 118912. 16, 19, 42, 44, 45

Tong, T.V.A., Bui-Thanh, B. & T. H., P.N. (2024). Human activity recognition using wireless signals and low-cost embedded devices. In *2024 Tenth International Conference on Communications and Electronics (ICCE)*, 7–12. 65

Touhiduzzaman, M., Hernandez, S.M., Pidcoe, P.E. & Bulut, E. (2024). Wi-pt-hand: Wireless sensing based low-cost physical rehabilitation tracking for hand movements. *ACM Trans. Comput. Healthcare*. 65

Tsubota, K., Ohhira, T. & Hashimoto, H. (2023). Heart rate variability and body-movement extractions using wi-fi channel state information. In *2023 IEEE/SICE International Symposium on System Integration (SII)*, 1–6. 23, 24, 42, 44, 45

Tzanetakis, G., Essl, G. & Cook, P. (2001). Audio analysis using the discrete wavelet transform. 318–323. 53

Van Nee, R., Jones, V.K., Awater, G., Van Zelst, A., Gardner, J. & Steele, G. (2006). The 802.11n mimo-ofdm standard for wireless lan and beyond. *Wireless Personal Communications*, **37**, 445–453. 47

Wang, F., Zhang, F., Wu, C., Wang, B. & Ray Liu, K.J. (2020a). Passive people counting using commodity wifi. In *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, 1–6. 30, 31, 42, 44, 45

Wang, H., Zhang, D., Ma, J., Wang, Y., Wang, Y., Wu, D., Gu, T. & Xie, B. (2016). Human respiration detection with commodity wifi devices: Do user location and body orientation matter? In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '16, 25–36, Association for Computing Machinery, New York, NY, USA. 20, 44, 45

Wang, J., Xiong, J., Jiang, H., Jamieson, K., Chen, X., Fang, D. & Wang, C. (2018a). Low human-effort, device-free localization with fine-grained subcarrier information. *IEEE Transactions on Mobile Computing*, **17**, 2550–2563. 27, 28, 42, 45

Wang, W., Liu, A.X., Shahzad, M., Ling, K. & Lu, S. (2017). Device-free human activity recognition using commercial wifi devices. *IEEE Journal on Selected Areas in Communications*, **35**, 1118–1131. 11, 15, 42, 44, 45

Wang, X., Yang, C. & Mao, S. (2020b). On csi-based vital sign monitoring using commodity wifi. *ACM Trans. Comput. Healthcare*, **1**. 23, 24, 42, 44

Wang, X., Yang, C. & Mao, S. (2020c). Resilient respiration rate monitoring with realtime bimodal csi data. *IEEE Sensors Journal*, **20**, 10187–10198. 21, 24, 42, 44, 45

Wang, X., Wang, X. & Mao, S. (2021). Indoor fingerprinting with bimodal csi tensors: A deep residual sharing learning approach. *IEEE Internet of Things Journal*, **8**, 4498–4513. 25, 28, 42, 44, 45

Wang, X., Li, F., Xie, Y., Yang, S. & Wang, Y. (2022). Gait and respiration-based user identification using wi-fi signal. *IEEE Internet of Things Journal*, **9**, 3509–3521. 38, 40, 42, 44, 45

Wang, Y., Xiu, C., Zhang, X. & Yang, D. (2018b). Wifi indoor localization with csi fingerprinting-based random forest. *Sensors*, **18**. 26, 28, 42, 44, 45

Wang, Z., Jiang, K., Hou, Y., Dou, W., Zhang, C., Huang, Z. & Guo, Y. (2019). A survey on human behavior recognition using channel state information. *IEEE Access*, **7**, 155986–156024. 8, 10

Wirth, R. & Hipp, J. (2000). Crisp-dm: Towards a standard process model for data mining. In *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, vol. 1, 29–39, Manchester. 66

World Health Organization (2021). Falls. 1

Wu, D., Gao, R., Zeng, Y., Liu, J., Wang, L., Gu, T. & Zhang, D. (2020). Fingerdraw: Sub-wavelength level finger motion tracking with wifi signals. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, **4**. 17, 19, 42, 44, 45

Wu, X., Chu, Z., Yang, P., Xiang, C., Zheng, X. & Huang, W. (2019). Tw-see: Human activity recognition through the wall with commodity wi-fi devices. *IEEE Transactions on Vehicular Technology*, **68**, 306–319. 12, 15, 42, 44, 45

Wu, Z., Xu, Q., Li, J., Fu, C., Xuan, Q. & Xiang, Y. (2018). Passive indoor localization based on csi and naive bayes classification. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, **48**, 1566–1577. 26, 28, 42, 44, 45

Xiao, C., Han, D., Ma, Y. & Qin, Z. (2019). Csigan: Robust channel state information-based activity recognition with gans. *IEEE Internet of Things Journal*, **6**, 10191–10204. 94

Xiao, F., Chen, J., Xie, X., Gui, L., Sun, L. & Wang, R. (2020). Seare: A system for exercise activity recognition and quality evaluation based on green sensing. *IEEE Transactions on Emerging Topics in Computing*, **8**, 752–761. 12, 15, 42, 44, 45

Xiaolong, Y., Xin, Y., Liangbo, X., Hao, X., Mu, Z. & Qing, J. (2021). Sleep apnea monitoring system based on commodity wifi devices. *Computers, Materials & Continua*, **69**, 2793–2806. 22, 24, 42, 44, 45

Xie, Y., Li, Z. & Li, M. (2015). Precise power delay profiling with commodity wifi. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, MobiCom '15, 53–64, ACM, New York, NY, USA. 50

Xie, Y., Li, Z. & Li, M. (2019). Precise power delay profiling with commodity wi-fi. *IEEE Transactions on Mobile Computing*, **18**, 1342–1355. 52

Xin, T., Guo, B., Wang, Z., Li, M., Yu, Z. & Zhou, X. (2016). Freesense: Indoor human identification with wi-fi signals. In *2016 IEEE Global Communications Conference (GLOBECOM)*, 1–7. 37, 40, 42, 44, 45

Xu, Q., Chen, Y., Wang, B. & Liu, K.J.R. (2017). Radio biometrics: Human recognition through a wall. *IEEE Transactions on Information Forensics and Security*, **12**, 1141–1155. 38, 40, 42, 44, 45

Yang, J., Zou, H., Jiang, H. & Xie, L. (2018). Device-free occupant activity sensing using wifi-enabled iot devices for smart homes. *IEEE Internet of Things Journal*, **5**, 3991–4002. 35, 37, 42, 44, 45

Yang, M., Zhu, H., Zhu, R., Wu, F., Yin, L. & Yang, Y. (2023a). Witransformer: A novel robust gesture recognition sensing model with wifi. *Sensors*, **23**. 16, 19, 42, 44, 45

Yang, R., Yang, X., Wang, J., Zhou, M., Tian, Z. & Li, L. (2022). Decimeter level indoor localization using wifi channel state information. *IEEE Sensors Journal*, **22**, 4940–4950. 26, 28, 42, 44, 45

Yang, X., Li, Q., Zhou, M. & Wang, J. (2023b). Phase-calibration-based 3-d beamspace matrix pencil algorithm for indoor passive positioning and tracking. *IEEE Sensors Journal*, **23**, 19670–19683. 27, 28, 42, 44, 45

Yang, Z., Zhang, Y., Zhang, G. & Zheng, Y. (2020). Widar 3.0: Wifi-based activity recognition dataset. 16

Yin, Y., Yang, X., Xiong, J., Lee, S.I., Chen, P. & Niu, Q. (2022). Ubiquitous smartphone-based respiration sensing with wi-fi signal. *IEEE Internet of Things Journal*, **9**, 1479–1490. 20, 24, 42, 44, 45

Youngworth, R.N., Gallagher, B.B. & Stamper, B.L. (2005). An overview of power spectral density (PSD) calculations. In H.P. Stahl, ed., *Optical Manufacturing and Testing VI*, vol. 5869, 58690U, International Society for Optics and Photonics, SPIE. 56

Zeng, Y., Pathak, P.H. & Mohapatra, P. (2016). Wiwho: Wifi-based person identification in smart spaces. In *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 1–12. 38, 40, 42, 44, 45

Zeng, Y., Wu, D., Xiong, J., Liu, J., Liu, Z. & Zhang, D. (2020). Multisense: Enabling multi-person respiration sensing with commodity wifi. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, **4**. 22, 24, 42, 44, 45

Zhang, F., Chen, C., Wang, B. & Liu, K.J.R. (2018). Wispeed: A statistical electromagnetic approach for device-free indoor speed estimation. *IEEE Internet of Things Journal*, **5**, 2163–2177. 11, 15, 33, 35, 42, 45

Zhang, J., Xu, W., Hu, W. & Kanhere, S.S. (2017). Wicare: Towards in-situ breath monitoring. In *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MobiQuitous 2017, 126–135, Association for Computing Machinery, New York, NY, USA. 21, 24, 42, 44, 45, 52

Zhang, J., Wu, F., Wei, B., Zhang, Q., Huang, H., Shah, S.W. & Cheng, J. (2021). Data augmentation and dense-lstm for human activity recognition using wifi signal. *IEEE Internet of Things Journal*, **8**, 4628–4641. 13, 15, 42, 44, 45, 98

Zhang, L., Wang, C., Ma, M. & Zhang, D. (2020a). Widigr: Direction-independent gait recognition system using commercial wi-fi devices. *IEEE Internet of Things Journal*, **7**, 1178–1191. 39, 40, 42, 44, 45

Zhang, Y., Zheng, Y., Zhang, G., Qian, K., Qian, C. & Yang, Z. (2020b). Gaitid: Robust wi-fi based gait recognition. In D. Yu, F. Dressler & J. Yu, eds., *Wireless Algorithms, Systems, and Applications*, 730–742, Springer International Publishing, Cham. 38, 40, 42, 44, 45

Zhang, Y., He, F., Wang, Y., Wu, D. & Yu, G. (2023). CSI-based cross-scene human activity recognition with incremental learning. *Neural Computing and Applications*, **35**, 12415–12432. 13, 15, 42, 44, 45

Zhang, Z., Duan, F., Solé-Casals, J., Dinarés-Ferran, J., Cichocki, A., Yang, Z. & Sun, Z. (2019). A novel deep learning approach with data augmentation to classify motor imagery signals. *IEEE Access*, **7**, 15945–15954. 94

Zhao, B., Zhu, D., Xi, T., Jia, C., Jiang, S. & Wang, S. (2019). Convolutional neural network and dual-factor enhanced variational bayes adaptive kalman filter based indoor localization with wi-fi. *Computer Networks*, **162**, 106864. 25, 28, 42, 44, 45

Zhao, Y., Gao, R., Liu, S., Xie, L., Wu, J., Tu, H. & Chen, B. (2021). Device-free secure interaction with hand gestures in wifi-enabled iot environment. *IEEE Internet of Things Journal*, **8**, 5619–5631. 11, 15, 39, 40, 42, 44, 45

Zhou, M., Long, Y., Zhang, W., Pu, Q., Wang, Y., Nie, W. & He, W. (2021). Adaptive genetic algorithm-aided neural network with channel state information tensor decomposition for indoor localization. *IEEE Transactions on Evolutionary Computation*, **25**, 913–927. 26, 28, 42, 44, 45

Zhou, R., Lu, X., Zhao, P. & Chen, J. (2017). Device-free presence detection and localization with svm and csi fingerprinting. *IEEE Sensors Journal*, **17**, 7990–7999. 35, 37, 42, 44, 45

Zhou, R., Hao, M., Lu, X., Tang, M. & Fu, Y. (2018). Device-free localization based on csi fingerprints and deep neural networks. In *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 1–9. 25, 28, 42, 44, 45

Zhou, S., Guo, L., Lu, Z., Wen, X. & Han, Z. (2023a). Wi-monitor: Daily activity monitoring using commodity wi-fi. *IEEE Internet of Things Journal*, **10**, 1588–1604. 13, 15, 42, 44, 45

Zhou, Y., Zhu, A., Xu, C., Hu, F. & Li, Y. (2022). Perunet: Deep signal channel attention in unet for wifi-based human pose estimation. *IEEE Sensors Journal*, **22**, 19750–19760. 32, 33, 42, 44, 45

Zhou, Y., Huang, H., Yuan, S., Zou, H., Xie, L. & Yang, J. (2023b). Metafi++: Wifi-enabled transformer-based human pose estimation for metaverse avatar simulation. *IEEE Internet of Things Journal*, 1–1. 32, 33, 42, 44, 45

Zhu, A., Tang, Z., Wang, Z., Zhou, Y., Chen, S., Hu, F. & Li, Y. (2022). Wi-atcn: Attentional temporal convolutional network for human action prediction using wifi channel state information. *IEEE Journal of Selected Topics in Signal Processing*, **16**, 804–816. 12, 15, 42, 44, 45

Zou, H., Zhou, Y., Yang, J. & Spanos, C.J. (2018a). Device-free occupancy detection and crowd counting in smart buildings with wifi-enabled iot. *Energy and Buildings*, **174**, 309–322. 29, 31, 36, 37, 42, 44, 45

Zou, H., Zhou, Y., Yang, J. & Spanos, C.J. (2018b). Towards occupant activity driven smart buildings via wifi-enabled iot devices and deep learning. *Energy and Buildings*, **177**, 12–22. 12, 15, 42, 44, 45

Zou, H., Yang, J., Zhou, Y., Xie, L. & Spanos, C.J. (2018). Robust wifi-enabled device-free gesture recognition via unsupervised adversarial domain adaptation. In *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, 1–8. 17, 19, 42, 44, 45