Human Activity Recognition Using MEMS-based Motion Sensors

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

BY

SONIA SOUBAM

PHD1314

COMPUTER SCIENCE ENGINEERING

INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY DELHI

NEW DELHI- 110020

January 2025

THESIS CERTIFICATE

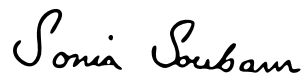
This is to certify that the thesis titled **Human Activity Recognition Using MEMS-based Motion Sensors**, submitted by **Sonia Soubam**, to the Indraprastha Institute of Technology, Delhi, for the award of the degree of **Doctor of Philosophy**, is a bona fide record of the research work done by her under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

January, 2025
Prof. Vinayak Naik
Professor
Indraprastha Institute of Information Technology,
Delhi New Delhi, 110020

CANDIDATE'S DECLARATION

The author hereby declares that the work presented in the thesis titled **"Human Activity Recognition Using MEMS-based Motion Sensors"**, submitted as partial fulfillment for the award of the degree of Doctor of Philosophy to the **IIIT-Delhi**, is an original research work carried out under the supervision of **Prof. Vinayak Naik** (IIIT-Delhi, India).

The results presented in this thesis have not been submitted in part or whole to any other university or institute for the award of any degree/diploma.



January, 2025
Sonia Soubam
PhD1314
Indraprastha Institute of Information Technology, Delhi
New Delhi 110020

ACKNOWLEDGEMENTS

I want to express my profound gratitude to several individuals who have been instrumental in completing this Ph.D. thesis. First and foremost, my heartfelt thanks go to my family. Their unwavering support, boundless love, and deep understanding have been the bedrock of my success, particularly during the most challenging phases of this journey. Their sacrifices have fueled my academic pursuits and instilled in me the strength to persevere. I am eternally grateful for their presence in my life.

My deepest appreciation is reserved for my advisor, Prof. Vinayak Naik. His exceptional guidance and mentorship have profoundly shaped my development as a researcher. Prof. Naik has been more than an advisor; he has been a mentor in the truest sense, providing patient guidance, insightful feedback, and constant encouragement. His ability to understand and support my research aspirations while pushing me toward excellence has been inspirational. His unwavering commitment to my success has left an indelible mark on my academic and personal growth, and for that, I am immensely grateful and honored. I thank my external co-advisor, Dr. Dipanjan Chakraborty, and collaborator, Dipyaman Banerjee, for their input.

I am equally grateful to my friends, from those I have known for years to those I have encountered during my academic endeavors. Their companionship and humor have been invaluable during the long hours dedicated to research and academic writing. Our discussions have been a source of personal joy and contributed significantly to my research, offering fresh perspectives and deepening my understanding. The moments of laughter and shared experiences have provided a welcome balance to the demanding nature of my research work. I deeply cherish these interactions and the unique blend of support and intellectual stimulation they have brought to my academic journey.

Without the understanding and support of my family, friends, and advisor, I would not have been able to overcome these challenges and reach this milestone. I am deeply grateful for their compassion and belief in me, and I will never forget their contributions to my success.

ABSTRACT

KEYWORDS: Human Activity Recognition; Wearables; Smartphone; Smartwatch; Motion Sensor; Pervasive Computing; Applied Machine Learning; Micro-electro-mechanical systems

Micro-electro-mechanical systems (MEMS) are miniature devices that integrate mechanical and electrical components on a single silicon chip, often including sensors and actuators. With sizes ranging from a few micrometers to millimeters, MEMS are diverse in design and functionality. They are pivotal in modern technology and are known for their precision, efficiency, and cost-effectiveness in mass production. They are crucial in advancing consumer electronics, medical devices, and automotive systems. MEMS motion sensors, such as accelerometers and gyroscopes, are essential for capturing detailed time series measurements that reflect dynamic environmental interactions. The thesis explores the utilization of MEMS motion sensors for human activity recognition.

We must address many questions to make sense of the motion sensor data. These concern sampling rate, size of data segments, feature extraction, fusing data from multiple sensors, and selection of classification techniques.

- We focus on finding the optimal sensor sampling rate by balancing the need for detailed data against resource constraints. Our analysis contrasts higher rates, which offer increased detail, against lower rates that are advantageous for their power-saving features.
- The research involves refining the windowing process to segment time-series sensor data into frames effectively, ensuring precise motion capture without losing context, and enhancing the accuracy of ground truth tagging for dependable data in model training and validation.
- A vital component is feature extraction, where significant information is extracted from sensor data, coupled with identifying complex features through signal processing, to represent diverse human movements accurately.
- The study examines using MEMS motion sensors as standalone wearables and in combination with environmental and object-embedded sensors. This approach allows for a thorough analysis of human motions, ranging from extensive activities to subtle micro-movements.

- We comprehensively evaluate various computational methods, from basic empirical algorithms to sophisticated machine learning and deep learning techniques. This evaluation centers on their effectiveness in interpreting and classifying motion data, considering factors like computational efficiency, adaptability, and accuracy.

We apply these approaches to explore the range and intricacy of human movements captured by these sensors. We select applications where the movements vary from mm, cm, to m. The sensors are worn, held in hands, or embedded in the surroundings.

- For movements in mm, we provide an innovative outlook on the analysis of handwriting micro-events in educational contexts, unlocking possibilities to delve into the subtleties of student interactions and learning dynamics. The sensor is worn on the wrist.
- For movements in cm, we pioneer a novel technique for personal health management centered on precisely monitoring liquid consumption, a key factor in promoting overall health and supporting informed lifestyle decisions. The sensing involves wrist-worn and object-embedded sensors.
- For movements in m, we propose a perceptive strategy to tackle the complexities of indoor parking in urban transport, offering an integrated solution that cleverly utilizes everyday devices and environmental data. The sensing is done through a combination of body-mounted and environmental sensors.

Together, these studies demonstrate how integrating diverse sensors and computational strategies can provide a far-reaching impact of this research in harnessing MEMS technology for practical everyday applications. They underscore the complexities inherent in accurately capturing human motion across diverse scenarios and the innovative approaches to navigating and overcoming these challenges.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ANN	Artificial Neural Network
BLSTM	Bidirectional Long Short-Term Memory
CNN	Convolutional Neural Network
CRF	Conditional Random Field
DT	Decision Tree
DL	Deep Learning
IMU	Inertial Measurement Unit
GARA	Google's Activity Recognition API
GPS	Global Positioning System
GMM	Gaussian Mixture Model
HAR	Human Activity Recognition
HEFC	Human-Engineered Features-based Classifiers
HMM	Hidden Markov Model
kNN	k-Nearest Neighbors
LDA	Latent Dirichlet Algorithm
LightGBM	Light Gradient Boosting Machine
LSTM	Long Short-Term Memory
ML	Machine Learning
NB	Naive Bayes
RF	Random Forest
RNN	Recurrent Neural Network
RPNN	Recurrent Probabilistic Neural Network
SVM	Support Vector Machine
TSDC	Temporal Sensor Data-based Classifiers
XGBoost	Extreme Gradient Boosting

NOTATION

$A(t)$	Accelerometer state at time t
$W(t)$	WiFi state at time t
$C(t)$	Combined state at time t
β	The rate of change in visible Wi-Fi APs as the user changes location.
acc	Accelerometer sensor
gyr	Gyroscope sensor
rot	Rotation Vector sensor
freeArm	Free Arm Movement Activities
resArm	Restricted Arm Activities
fullArm	Full Arm Movement Activities

CHAPTER 1

Introduction

This thesis delves into the domain of Human Activity Recognition (HAR) utilizing micro-electro-mechanical systems (MEMS)-based motion sensors, a field that stands at the intersection of advanced sensor technology and the growing demand for sophisticated activity recognition in various domains. MEMS, integrating mechanical and electrical components on a micro-scale, have revolutionized how we capture and interpret the nuances of human motion. This research predominantly focuses on leveraging MEMS motion sensors, such as accelerometers and gyroscopes, embedded in universally accessible devices like smartphones and smartwatches to recognize and analyze various human activities.

The significance of understanding human movements and behaviors through HAR is underscored by its unique intersection with technology, offering profound insights into daily activities. This capability enables a deeper analysis and interpretation of human motion and behavior, revolutionizing numerous aspects of life by enhancing health and well-being and improving efficiency and safety across various environments. Accurate monitoring and analysis of activities, facilitated by MEMS sensors, open up new possibilities in personalized healthcare, ergonomic design, and interactive technologies. Furthermore, these advancements contribute to the evolution of artificial intelligence, enhancing its ability to understand and predict human behavior with greater accuracy.

In the HAR, various forms of sensing are used—namely wearable, environmental, and object-embedded [69]—each with unique advantages and challenges. Wearable sensing, which involves sensors mounted directly on the body, excels in providing detailed, personal data and is particularly suited for activities that involve significant body movement. The ubiquity of smartphones and smartwatches has greatly facilitated data collection through wearable sensing, although issues related to wearability and the scope of data coverage remain considerations. Environmental sensing, characterized by sensors placed within an environment, offers a less intrusive means of data collection. It can simultaneously cover larger areas and multiple individuals, making it ideal

for public or communal spaces. However, this method may not capture the nuanced, individual-specific data that wearable sensors can, and it often involves more complex installation and maintenance. Object-embedded sensing integrates sensors into everyday objects, provides context-specific data and blends seamlessly into daily routines. While offering valuable insights into interactions with these objects, its application is limited to the context of the specific objects embedded with sensors.

Reflecting on the evolution of HAR, there has been a notable shift from the initial reliance on computer vision methods [26], a form of environmental sensing, to the adoption of MEMS-based sensors driven by the need for privacy-sensitive and ubiquitous computing. While vision-based HAR systems are effective in public spaces, privacy concerns [75] limit their use in personal environments. As a less intrusive alternative, MEMS sensors capture detailed human movements and align with the demand for discreet and comprehensive activity data. This thesis builds upon this technological evolution, focusing on using MEMS-based sensors in wearable devices like smartphones and smartwatches. Smartwatches, inherently designed as wearables to be worn on the wrist, straightforwardly capture human motion. In contrast, smartphones, typically seen as interactive handheld devices, are consistently carried by users and, in the context of this thesis, are treated as wearables, effectively contributing to the capture of human motion. By integrating advanced sensor technology into everyday wearable devices, this research marks a significant advancement in HAR, enabling nuanced capture of a wide range of human activities and opening new avenues for application across various domains.

Advancements in smartphone and smartwatch MEMS-based technologies have empowered HAR to provide valuable insights in diverse fields such as healthcare, sports, and entertainment. In healthcare, HAR is utilized to monitor patient activity levels and track movements in specific populations [27]. In sports, it aids in performance analysis and injury risk reduction [14]. In the realm of entertainment, HAR contributes to creating immersive and interactive experiences [24]. Integrating these varied sensing modalities in this research paves the way for new avenues in HAR application, significantly broadening its scope and enhancing its impact.

The evolution of HAR has been profoundly influenced by the adoption of MEMS-based sensors in commonly used devices like smartphones and smartwatches. Before

delving into the specific goals of this thesis, it is essential first to present an overview of how sensor data is transformed into activity recognition. The following section will provide this overview, setting the stage for a deeper understanding of the advancements in HAR and the methodologies that underpin this research.

1.1 Data Processing , Training, and Testing Pipeline

The process of motion sensor-based HAR includes several essential steps. The following provides a brief overview of these steps. A more detailed discussion of each step and their specific roles in HAR is covered in Chapter 2.

- **Noise Filtering**: The first step is refining sensor data by removing noise. Noise in HAR can arise from various sources, such as sensor inaccuracies, environmental factors, and human-induced variations. This is done using techniques ranging from simple moving averages to advanced methods like wavelet transforms and Kalman filters. Recent developments like empirical mode decomposition and adaptive filters are also used for more dynamic noise reduction.
- **Windowing and Ground Truth Tagging**: This step segments time-series sensor data into smaller windows for extracting features. The size of these windows is crucial, as it needs to balance capturing enough detail without including irrelevant data. Each data segment is then labeled with the correct activity, a process known as ground truth tagging, which is vital for the accuracy of the model.
- **Feature Extraction and Selection**: Here, raw data is transformed into meaningful features. These features can be based on time-domain, frequency-domain, or more complex models. The goal is to select features that reduce data complexity and improve the model's performance and interpretability.
- **Model Training and Activity Classification**: In this stage, algorithms are developed to recognize different activities. These can range from traditional machine learning models to more advanced deep learning networks. The decision on which algorithm to use hinges on the specific issue being tackled, the data at hand, and the expected results.
- **Performance Evaluation**: The final step is evaluating the model's performance. This is done using metrics like accuracy, precision, recall, and F1-score, which help understand the model's strengths and areas for improvement.

1.2 Research Goal and Objective

Focusing on the research goal and objectives, this thesis aims to optimize key facets of HAR utilizing MEMS-based motion sensors. Central to this research is determining

the optimal sensor sampling rate to balance the need for detailed data against resource constraints, a crucial consideration in the initial noise filtering stage. Additionally, the thesis emphasizes refining the windowing process, a vital step for effective feature extraction and ground truth tagging, to ensure accurate motion capture and maintain contextual integrity.

The research further explores advanced feature extraction techniques, aiming to enhance the transformation of raw sensor data into meaningful features, thereby improving model performance and interpretability. Another significant objective is to examine the application of MEMS motion sensors in various configurations, including their use in standalone wearables and in combination with other sensing methods. This approach allows for a comprehensive analysis of human motions, from extensive physical activities to subtle micro-movements.

Lastly, the thesis thoroughly evaluates computational methods, ranging from basic empirical algorithms to sophisticated machine learning and deep learning techniques. This evaluation is focused on assessing their effectiveness in interpreting and classifying motion data, considering factors such as resource limitations, time sensitivity of applications, and accuracy of outcomes. These goals are explored through three distinct HAR problems, each demonstrating the practical application of the optimized data processing pipeline in real-world scenarios.

Building upon the established research goals, this thesis delves into three distinct HAR problems, carefully selected to encompass a diverse range of human motions—from full-body movements to arm movements and even minute motions. These problems span varied application domains, including transportation, health, and education, and leverage different sensor deployments: standalone wearables, a combination of wearables with environmental sensors, and object-embedded sensors. Each problem presents its own set of unique challenges and opportunities, contributing significantly to the field of HAR. The following subsections offer an overview of these problems, illustrating how each one aligns with and advances the overarching objectives of this research.

1.2.1 Handwriting Micro-Event Recognition

Objective: The primary objective of this research is to develop a system capable of detecting handwriting micro-events—such as line changes, hand shifts, and active writing activity—using motion sensors embedded in commercially available smartwatches. This system provides a cost-effective and non-intrusive solution for analyzing handwriting patterns, enabling applications such as assessing students' cognitive load, academic habits, and performance in educational settings. For this thesis, the study focuses on a wearable sensor-based solution specifically designed for minute hand movements during writing tasks, which has applications in the education domain.

Background: Handwriting plays a crucial role in education, offering insights into cognitive and motor processes. Traditional handwriting analysis focuses on the written content, often relying on image-based methods or specialized tools like smart pens and tablets. In contrast, this research emphasizes handwriting pattern analysis, which examines subtle hand movements during writing tasks to gain insights into factors such as attentiveness and cognitive load. This approach preserves privacy by analyzing the writing process without accessing the text itself. Analogous to tracking physical activity patterns (e.g., steps taken or climbing stairs), handwriting pattern analysis shifts attention to the movements underlying the task rather than the text content.

Solution and Key Challenge: We leverage motion sensors from commercially available smartwatches to detect handwriting micro-events, offering a widely accessible and cost-effective solution for handwriting pattern analysis.

- **Input:** Time-series motion data from smartwatch sensors capturing hand movements during writing activities.
- **Output:** A classification of handwriting micro-events, including hand shifts, line changes, and active writing periods, to enable applications such as cognitive load estimation and academic performance assessment.

The key challenge lies in the variability of handwriting micro-event durations, both among individuals and within the same individual. This variability complicates the establishment of a fixed window size for event detection and data tagging, which are critical for supervised learning. To address this, we propose an approach that prioritizes shorter and less frequent micro-events during the ground truth tagging process. Our

analysis evaluates various feature types across machine learning algorithms and employs deep learning models, specifically Long Short-Term Memory (LSTM) networks, trained on raw sensor data.

1.2.2 Indoor Parking and Un-parking Event Detection and Localization

Objective: The objective of this work is to develop BluePark, an automated system designed to detect and localize parking and un-parking events in indoor environments without requiring additional dedicated sensors. The system leverages smartphone sensors and existing WiFi infrastructure to provide an accurate, efficient, and cost-effective solution to the challenges of indoor parking. For this thesis, the focus is on utilizing a combination of wearable and environmental sensors, specifically tailored to scenarios involving full-body movements to address a problem in the transportation domain.

Background: Existing solutions for indoor parking management often rely on dedicated infrastructure or manual logging, which can be costly and inconvenient. Additionally, drivers frequently face difficulties locating their parked cars in large basement parking facilities. While smartphone-based solutions exist for outdoor parking, they depend on GPS signals, which are unavailable in indoor environments. These limitations highlight the need for an accessible and efficient system for indoor parking management.

Solution and Key Challenge:

BluePark addresses these challenges by utilizing ubiquitous smartphone sensors and existing WiFi access points, offering a scalable and user-friendly solution. This approach facilitates the detection of driving state transitions, enabling accurate identification and localization of parking events.

- **Input:** Accelerometer data from the user's smartphone and nearby WiFi access points scans.
- **Output:** Detection of parking and un-parking events, along with vehicle localization within an accuracy of 10 meters. The system also provides real-time updates about parking and un-parking events to user's smartphone.

Differentiating between driving and parking states, especially at low speeds in GPS-

denied indoor environments, is a significant challenge. Additionally, the system must account for varying smartphone positions—such as in pockets, bags, or car mounts—necessitating robust and adaptable algorithms.

1.2.3 Liquid Intake Tracking

Objective: This research aims to develop a user-friendly and adaptable solution for accurately monitoring liquid intake. The proposed system integrates smartwatch motion sensors with a load sensor embedded in a detachable container base to automate liquid intake tracking. For this thesis, the study focuses on a combination of wearable and object-embedded sensor-based solutions, leveraging arm movements to address a problem in the fitness domain.

Background: Maintaining proper hydration is critical for health and managing chronic conditions. Existing methods for tracking liquid intake often rely on manual logging or specialized containers, which are cumbersome and prone to errors. Specialized container-based solutions are often expensive, restrict users to specific containers, and lack robustness against accidental logging. This work introduces a more flexible solution, allowing users to utilize their own containers and track various types of beverages while achieving precise intake measurements.

Solution and Key Challenges: The system employs a dual-sensing approach, combining a load sensor mounted on a detachable base with a smartwatch accelerometer. The load sensor measures changes in liquid weight to calculate the amount consumed, while the smartwatch accelerometer detects drinking motions. The detachable base design ensures adaptability to various container types, such as thermoses, disposable bottles, and plastic water bottles, as well as diverse liquid types, including water, milk, soda, and juices. The drinking motion detection by the smartwatch helps distinguish actual consumption from events such as spillages, refills, or misuse, which may also register as weight changes on the load sensor. A key challenge lies in differentiating drinking motions from other similar hand movements, such as touching the face, eating, or smoking. To address this, the load sensor acts as a confirmation of drinking-related hand movements, ensuring that liquid intake is logged only when both the load sensor and the smartwatch agree. If the two sensors do not corroborate, the data is discarded.

- **Input:** Motion data from a smartwatch during drinking actions and weight data from the detachable base when the container placed on a stable surface.
- **Output:** Precise liquid consumption records, accounting for individual sips and filtering out irrelevant movements, with a high degree of accuracy.

1.3 Scope and Limitations

1.3.1 Scope

The scope of this thesis is centered on developing and advancing HAR techniques using smartphones and smartwatches. The focus is primarily on activities involving physical movements, utilizing motion sensors independently or in combination with external sensors. This research deliberately excludes activities that are physiological or biometric in nature, as these typically require different types of sensors and analytical approaches. The research maintains a consistent methodological approach by employing only supervised learning methods throughout the studies. The applications of the developed HAR techniques are targeted specifically toward three distinct domains: transportation, health monitoring, and education. Concentrating on these areas contributes to the real-world application of HAR, where integrating HAR technology can yield considerable benefits in terms of transportation, well-being, and learning outcomes.

1.3.2 Limitations

This thesis strives to make a substantial contribution to the field of HAR, yet it is essential to recognize its limitations. It should be noted that the following are overarching limitations of the thesis as a whole. For specific limitations about individual problems, detailed discussions are provided in the respective chapters.

1. **Participant Diversity**: The data used in this research is derived from a limited number of participants. Despite efforts to ensure diversity and representation, potential biases due to the sample size and composition may influence the findings.
2. **Generalizability and Scalability**: The proposed systems need further evaluation for their applicability in varied settings. The findings presented in this thesis may not directly translate to larger and more diverse populations, which is a crucial consideration for the scalability of HAR systems.

3. **Privacy Concerns**: Utilizing smartphones and smartwatches for HAR inevitably raises privacy concerns. While this thesis does not delve into methods for preserving user privacy, it acknowledges the importance of this issue in the broader context of HAR applications.

4. **Battery Consumption**: Another critical consideration for smartphone and smartwatch applications is battery consumption. This research does not include an analysis of power consumption, which is an essential factor for the practical implementation and long-term viability of HAR techniques.

Road map: The subsequent chapters are structured as follows: In Chapter 2, we delve into the foundational aspects, providing essential background information about the motion sensor data, the data processing pipeline, and the classification algorithms central to our research. Moving forward, Chapter 3 is dedicated to an extensive literature review, where we explore existing works and research relevant to our study, providing a backdrop and insights into the current state of knowledge in the field. In Chapter 4, we present the handwriting micro-event detection system tailored for handwriting analysis, elucidating the intricacies of its implementation and significance in our study. Subsequently, Chapter 5 delves into a comprehensive examination of the indoor parking system we have employed, shedding light on its operational aspects and contributions. Chapter 6 focuses on the system developed for measuring liquid consumption, discussing its design and functionality in detail. Subsequently, Lastly, in Chapter 7, we draw our thesis to a close, summarizing our key findings and insights.

CHAPTER 2

Background

This chapter delves deeper into the foundational concepts and methodologies central to this thesis. We explore the mechanics of MEMS-based motion sensors in smartphones and smartwatches, focusing on how they capture movement information. Additionally, we examine the data schema associated with the Android platform, selected for this research due to its widespread use and accessibility. Following this, we detail the general data processing pipeline, a framework pivotal to all three HAR problems discussed in subsequent chapters. This section sheds light on how raw motion sensor data is transformed for analytical use in HAR, addressing the specific challenges and requirements of this process. The chapter then provides an overview of the machine learning paradigms employed in our study, elucidating their theoretical underpinnings and practical applications in the context of HAR. By presenting these foundational aspects, this chapter lays the groundwork for the methodologies and analyses that follow, equipping the reader with the essential background needed to fully grasp the complexities of the research presented in this thesis.

2.1 MEMS-based Motion Sensors in Wearable Devices

Motion sensors are devices that detect and measure physical movements, converting these motions into useful digital data. At the heart of their integration into mobile devices is the MEMS technology. MEMS technology is a fusion of microelectronics and precision mechanical engineering, enabling the embedding of miniature mechanical components within electronic circuits. These compact systems are meticulously engineered to sense and respond to their surroundings through various mechanisms, including capacitive, piezoresistive, and piezoelectric sensing. Among the most prominent MEMS motion sensors found in today's mobile devices are accelerometers, gyroscopes, and magnetometers. Each of these sensors plays a crucial role in capturing different aspects of movement and orientation, making them essential in a wide range of applications.

To grasp the functional principles of these sensors, we will delve into the specifics of each type:

- **MEMS Accelerometer:** An accelerometer measures acceleration by detecting changes in capacitance. Its fundamental structure includes a small mass connected to a spring, encased within a compact enclosure, and surrounded by fixed plates. When acceleration is applied, the mass shifts, altering the capacitance between it and the plates. This change in capacitance is then measured to determine the acceleration.
- **MEMS Gyroscope:** Gyroscopes measure angular rate using the Coriolis effect. They consist of a mass that oscillates continuously. When the sensor experiences angular motion, the Coriolis force causes a perpendicular deflection of this mass from its original path. This deflection, similar to the accelerometer's mechanism, leads to a change in capacitance, which is then measured to gauge the angular rate.
- **MEMS Magnetometer:** Magnetometers, designed to measure the Earth's magnetic field, primarily operate based on the Hall effect. A magnetic field near a conductive plate with a steady flow of electrons disrupts this flow, causing a separation of charges. This results in a voltage difference across the plate, proportional to the magnetic field's strength and direction. Some magnetometers also use the Magneto-Resistive effect, where materials like iron and nickel change their resistance in response to magnetic fields, enabling the measurement of the field's intensity.

Within the Android ecosystem, the Sensor Framework, a foundational component of the Android OS, oversees the acquisition and processing of data from sensors. This framework retrieves the raw sensor data and applies preprocessing measures, such as calibration, noise filtering, and sampling rate adjustments, to enhance data reliability. The refined data is then accessible to applications through the Android Sensor API, allowing developers to transform raw sensor readings into actionable insights.

The Android Sensor API utilizes a 3-dimensional coordinate system to represent motion sensor data, aligning with a device's x, y, and z axes. Figure 2.1 illustrates the orientation of these axes in relation to both a smartphone and a smartwatch, providing a visual understanding of how movements are mapped in 3D space. The data recorded for each axis is typically presented as a 'float' value:

- **X-axis (Horizontal axis):** Indicates left (-) to right (+) movement.
- **Y-axis (Vertical axis):** Represents bottom (-) to top (+) movement.
- **Z-axis (Perpendicular axis):** Captures forward and backward movement, from the screen (-) towards the back (+).

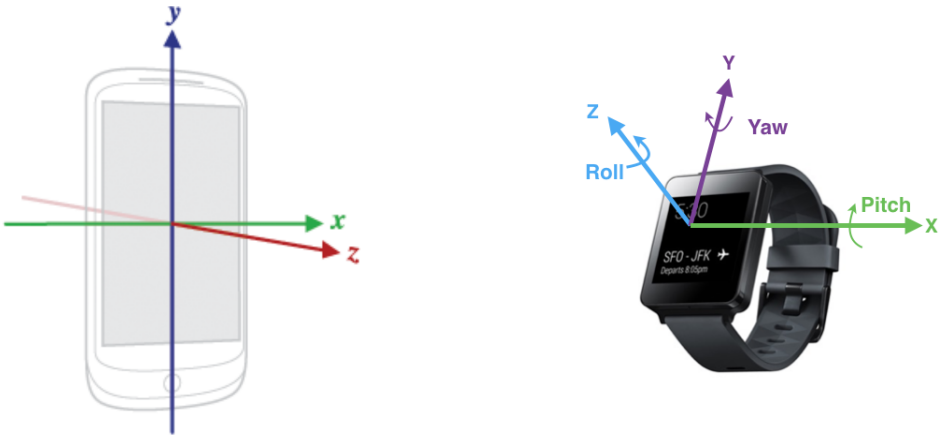


Figure 2.1: The coordinate system used by the Android Sensor API, relative to the (a) smartphone and (b) smartwatch. Image source for (a): https://developer.android.com/guide/topics/sensors/sensors_overview

Understanding these axes is crucial for interpreting the motion data accurately. The units of measurement vary depending on the sensor type: meters per second squared (m/s^2) for accelerometers, radians per second (rad/s) for gyroscopes, and microteslas (μT) for magnetometers. These units offer a standardized way to quantify and compare sensor readings. Each sensor reading includes a timestamp, marking the precise moment the data is captured. This timestamp is essential for synchronizing data from multiple sensors and conducting accurate time-series analyses.

The following is an example time-series representation of accelerometer data in human-readable form.

Timestamp	**Acc_x** (m/s^2)	**Acc_y** (m/s^2)	**Acc_z** (m/s^2)
1621294800000	9.81	0.00	0.00
1621294800010	9.82	-0.01	0.02
1621294800020	9.80	0.01	-0.01
1621294800030	9.81	0.00	0.00
1621294800040	9.82	0.01	0.02
⋮	⋮	⋮	⋮

In the table:

- The **Timestamp** column represents the time at which the data was collected in milliseconds since the Unix epoch.
- The difference between two consecutive timestamps is 10 milliseconds, which is equivalent to 0.01 seconds. This implies that data is collected 100 times every second, resulting in a sampling rate of 100Hz.

- **Acc_x**, **Acc_y**, and **Acc_z** represent the acceleration values recorded on the respective axes.

Sensor Sampling Rate: The sampling rate, which determines how frequently the sensor updates its data, is a pivotal factor significantly influencing the sensor's effectiveness. The significance of the sampling rate is evident from multiple important aspects:

- **Data Accuracy and Resolution:** The sampling frequency directly impacts the accuracy and resolution of the data captured by MEMS sensors. A higher sampling rate allows for more data points to be collected in a given period, leading to a finer resolution of movement and orientation. This is crucial in applications requiring precise motion tracking, such as medical devices or high-performance athletic training.
- **Real-time Processing:** In applications that require real-time feedback, such as gesture recognition in virtual reality systems, a higher sampling frequency ensures that the system can respond swiftly and accurately to changes in motion. This responsiveness is key to the smooth functioning and user experience of such systems.
- **Energy Consumption:** While a higher sampling rate provides more detailed data, it also demands more power, which can be a significant consideration in battery-operated devices like wearables. Balancing the need for detailed data with energy efficiency is crucial to designing systems that use MEMS motion sensors.
- **Noise Reduction and Signal Processing:** The choice of sampling frequency also affects the signal-to-noise ratio and the effectiveness of subsequent signal processing techniques. A rate that is too low may miss important fluctuations in the signal (under-sampling), while a too high rate can introduce noise (over-sampling), making it harder to extract meaningful insights from the data.
- **Application-Specific Requirements:** Different applications have varying requirements for sampling frequency. For instance, a fitness tracker might not need as high a sampling rate as a device used for earthquake detection. Understanding the application's specific needs is crucial in determining the optimal sampling rate.

2.2 Data Processing Pipeline

Our data processing pipeline converts the sequential sensor data into structured data tailored for efficient processing and learning by machine learning models. The general format for machine learning input is a tabular structure akin to a spreadsheet. This format consists of rows and columns, where each row represents an individual record instance, and each column describes a specific feature. In supervised learning, this

structure also incorporates labels or target values in a separate column, indicating the desired outcome for the model to predict. These labels vary from categorical values in classification tasks, like 'sitting' or 'not sitting,' to continuous values in regression tasks, such as calories burned. This section outlines the transformation of time-series motion sensor data into a structured format for machine learning, the learning process of these models from the structured data, and an overview of the methods used to evaluate these models.

2.2.1 Noise Filtering

In HAR, the integrity and quality of motion sensor data are often compromised by noise from various sources, including environmental factors like temperature changes or electromagnetic interference and sensor hardware issues like manufacturing imperfections or wear over time. This noise can lead to inaccuracies, such as misinterpreting random vibrations as intentional human activity. Effective noise reduction techniques are thus crucial in preprocessing motion sensor data for HAR. These techniques aim to eliminate undesired disturbances while preserving essential activity characteristics.

For example, the Moving Average Filter, which smoothens data by averaging over a set period, is useful in identifying sustained human activities by reducing short-term fluctuations to highlight longer-term trends. Applying a Gaussian function, the Gaussian Filter effectively diminishes high-frequency variations, which are often noise rather than meaningful movement.

Low-pass filters are beneficial in focusing on low-frequency signals that typically represent the primary data patterns in human movements, excluding high-frequency noise. Conversely, high-pass filters capture transient, quick movements by eliminating prolonged, consistent variations. Band-pass filters are ideal for isolating frequencies within a specific range pertinent to certain activities.

The Wavelet Transform offers a multi-resolution analysis by decomposing signals, concentrating on frequency bands with significant information, and suppressing irrelevant noise. The Kalman Filter provides a state estimate based on historical states and current noisy measurements tailored for systems necessitating real-time state predictions. Particle Filters use a probabilistic approach with random samples to cater to

systems exhibiting non-linear and non-Gaussian behaviors.

Emerging filtering techniques include the Empirical Mode Decomposition (EMD), which breaks down signals into their fundamental components, facilitating the study of non-linear data. Adaptive Filters modify their parameters based on the input, providing dynamic noise filtering. In the deep learning domain, autoencoders compress and reconstruct data, inherently refining the input by suppressing noise. Across these methods, the overarching goal remains: achieving a noise-reduced representation of the original data.

Choosing a specific noise filtering technique often involves a trade-off between the complexity of the noise and the computational resources available. It also requires understanding the frequency characteristics of both the noise and the signal of interest. Selecting the most effective noise filtering method is essential for ensuring the data's reliability and the subsequent analysis's success.

2.2.2 Windowing and Ground Truth Tagging

The windowing process segments time-series sensor data into concise, fixed-size windows, which are then used to extract features that capture human activity dynamics. The size of these windows significantly influences the model's efficacy [158]; smaller windows offer higher temporal resolution but may overlook complex activity patterns, while larger windows risk including irrelevant data, potentially reducing recognition accuracy.

Following windowing, ground truth tagging is essential for ensuring that each data segment is accurately labeled, forming the foundation for supervised learning. This labeling can be done manually, with semi-supervised techniques or automated algorithms. Their exactness is critical as the machine learning models train on these labels.

The relationship between window size and ground truth tagging is significant. Too small a window can lead to fragmented or ambiguous labels, while overly large windows may encompass multiple activities, complicating the labeling process. Therefore, finding an optimal window size is key to achieving precise annotations and maximizing model performance.

While fixed window sizes are standard in activity recognition, there is increasing interest in dynamic or adaptive window sizes [110]. These windows adjust based on the characteristics of the sensor data or the type of activity, offering enhanced flexibility and potentially better results. However, they require more complex algorithms and can increase the system's overall complexity. Determining the ideal parameters for adaptive windowing also presents unique challenges.

This research uses a fixed window size, valuing simplicity and computational efficiency. Although adaptive windows offer certain advantages, a fixed size streamlines the implementation process and simplifies parameter selection, making it suitable for various activity recognition tasks.

2.2.3 Feature Extraction and Selection

The feature extraction phase follows the windowing process in the data pipeline, where segmented time-series sensor data is transformed into a structured set of numerical features for machine learning models. This step is crucial for distilling significant and informative aspects from the data, thereby enhancing its suitability for algorithmic analysis.

In this phase, raw data is converted to highlight key patterns or characteristics, known as features, which are pivotal for the algorithms' learning and predictive functions. These features must be closely aligned with the specific requirements of the problem being addressed. For example, in image recognition, feature extraction might involve identifying edges, textures, or specific shapes within an image. Text analysis could involve extracting keywords, phrases, or semantic structures from a body of text. In the context of HAR, feature extraction typically involves computing statistical measures like mean and variance or frequency domain features, such as the power spectrum, from sensor readings.

Feature extraction in HAR can be broadly categorized into time-domain, frequency-domain, and time-frequency features. Time-domain features are statistical measures computed from each data window. Frequency-domain features analyze the signal's frequency components, while time-frequency features, like spectrograms, provide a combined perspective of the signal's time and frequency characteristics.

The subsequent step is feature selection, which significantly enhances model performance. It involves identifying and retaining the most relevant features while discarding redundant or irrelevant ones. This process not only reduces the computational complexity of the model, making it faster and more efficient but also helps mitigate the risk of overfitting. By simplifying the model through feature selection, the interpretability of the model is also improved, making it easier to understand and explain its decisions. Furthermore, focusing on the most pertinent features often increases the model's accuracy, as it is trained on high-quality, relevant data that directly contributes to the predictive task at hand. Feature selection methods include filter approaches like the Pearson correlation coefficient, which assesses features based on their correlation with the target class, and wrapper methods like recursive feature elimination, which evaluates features through classifier performance. Embedded methods, such as LASSO, incorporate feature selection within the learning algorithm itself.

By integrating effective feature extraction and selection, classifiers can accurately identify and categorize human activities, ensuring that the models are both precise and capable of handling the intricacies of human motion patterns.

2.2.4 Model Training and Classification

After completing the feature extraction phase, where key characteristics are derived from the data, the focus in the pipeline shifts to model training and classification. During model training, the primary goal of the classifier is to recognize and assimilate patterns from the data that has been labeled and processed through feature extraction. This crucial step involves meticulously adjusting the classifier's internal parameters to establish a precise correlation between the input features and their respective labels. This adjustment process is iterative, requiring the model to continually refine its parameters to reduce the discrepancies between its predictions and the actual labels. The specifics of these adjustments vary depending on the chosen algorithm. For a comprehensive exploration of the different algorithms used in this research, including their unique functions and selection criteria, detailed explanations are available in Section 2.3.

In training a model, it is imperative to carefully consider the balance between bias

and variance, critical factors affecting the model's performance. Bias arises from over-simplifying the learning algorithm, leading to underfitting. Bias occurs when the model is too simplistic or not adequately trained, causing it to miss the complex patterns in the data and perform poorly on both training and unseen data. On the other hand, variance is the error that results from the learning algorithm being too complex, leading to overfitting. In this scenario, the model becomes overly focused on the training data, including its noise and specific details, impairs its ability to generalize to new data. The key to effective model training is finding the right balance between bias and variance, known as the bias-variance tradeoff. Achieving this balance involves developing a complex model to learn the underlying patterns in the data but not so complex that it learns from the noise and errors. This balance is crucial for ensuring the model remains accurate and reliable, not just in the training data but also in its predictions or classifications of new, unseen data.

Once adequately trained, the model transitions to the classification stage, applying what it learned to categorize new, unseen data. This stage tests the model's ability to effectively use its training to make accurate predictions or classifications, reflecting the culmination of the preceding steps in the machine learning process.

2.2.5 Performance Metrics and Model Evaluation

Performance metrics play a crucial role in quantitatively assessing the effectiveness of a machine learning model, particularly post-training. During the training phase, a model iteratively adjusts its parameters based on a loss function to minimize errors. However, the comprehensive evaluation of its performance post-training necessitates diverse metrics. These metrics, which may include accuracy, precision, recall, F1-score, and others, offer insights into various aspects of the model's performance, often extending beyond the scope of the training's loss function.

The choice of these metrics is typically guided by the task's specific objectives, the dataset's nature, and the desired outcomes. For example, accuracy is a widely used metric for its simplicity and general applicability, particularly in balanced datasets. However, its effectiveness diminishes in imbalanced datasets where one class may dominate over others. In such cases, more nuanced metrics like precision and recall become cru-

cial. Precision is the ratio of true positive predictions among all positive predictions, while recall assesses the accuracy of positive predictions against the total number of actual positive cases. The F1-score, a harmonic mean that integrates precision and recall, becomes essential when balancing the impact of false positives and false negatives is important.

Various analytical tools and techniques are employed to delve deeper into the model's performance. The confusion matrix, for example, provides a detailed breakdown of the model's predictions, allowing for a more granular analysis of its performance in terms of true positives, true negatives, false positives, and false negatives. Another instrumental tool is the Receiver Operating Characteristic (ROC) curve, which presents an extensive overview of how the model performs under varying threshold settings. This curve and the Area Under the Curve (AUC) metric are instrumental in evaluating the trade-offs between true positive and false positive rates, thereby offering a nuanced understanding of the model's reliability and robustness under different operational scenarios. These metrics and tools collectively enable a thorough and multifaceted evaluation of a machine learning model, ensuring that its performance is understood in a broad sense and scrutinized in detail.

2.3 Algorithms

In this section, we explore the algorithms employed in this thesis, focusing on how they determine the association between features and their corresponding labels. We closely examine these algorithms' core principles and practical applications, providing insights into their strengths and limitations. This description offers a comprehensive understanding of each algorithm's functionality and role in effectively addressing the research challenges presented in this thesis.

2.3.1 Logistic Regression

Logistic Regression is a well-known statistical model that predicts the likelihood of an outcome based on input features using a generalized linear model. It employs a generalized linear model, where the logistic or sigmoid function maps these features to

a probability between 0 and 1, indicating the likelihood of an instance being categorized as 'positive.'

The logistic function, defined as follows, transforms the linear combination of input features into a probability.

$$\sigma(z) = \frac{1}{(1 + e^{-z})}$$

Here, z is the logit, the natural logorithm of the odds $\frac{p}{1-p}$, where p is the probability. The model expresses the log odds of the outcome as a linear combination of the input features,

$$z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

where $X_1, X_2, \ldots, X_p$ are the input features, and $\beta_0, \beta_1, \beta_2, \ldots, \beta_p$ are coefficients that need to be estimated. The coefficients quantify the importance or weight of each corresponding feature in determining the outcome's log odds. The goal during training is to adjust the coefficients to minimize the disparity between the predicted probabilities and the true class labels.

Training Process

Training a logistic regression model involves optimizing the coefficients to predict best the given outcomes based on the input features. This optimization is accomplished by minimizing a loss function, typically the negative log-likelihood. Given a dataset, the objective is to adjust the coefficients to maximize the likelihood of observing the provided outcomes.

For a single input data point y_i, the likelihood is given by:

$$L(\beta|X_i) = y_i \sigma(X_i\beta) + (1 - y_i)(1 - \sigma(X_i\beta)) \tag{2.1}$$

The objective during training is to minimize the negative log-likelihood across all data points:

$$J(\beta) = -\sum_{i=1}^{N} [y_i \log(\sigma(X_i\beta)) + (1 - y_i) \log(1 - \sigma(X_i\beta))] \tag{2.2}$$

Various optimization algorithms, including gradient descent, Newton-Raphson, and Quasi-Newton methods, are available to minimize this function and estimate the coefficients.

Multiclass Classification

By default, logistic regression is a binary classifier; that is, it categorizes input data into one of two distinct classes. It can be extended for multi-class classification using the one-vs-rest (OvR) and softmax regression approaches.

OvR involves creating multiple binary models, each distinguishing one class from the rest. For instance, in a three-class scenario (A, B, C), we would develop three models: A vs (B + C), B vs (A + C), and C vs (A + B). Each model is trained to identify its target class against the combined other classes. The class with the highest probability from these models is chosen as the predicted class.

In Softmax Regression, each class j has its corresponding logit z_j, calculated similarly to binary logistic regression but for each class. Specifically, the logit for class j is:

$$z_j = \beta_{0j} + \beta_{1j}X_1 + \beta_{2j}X_2 + \cdots + \beta_{pj}X_p$$

The probability of class j using the Softmax function:

$$S(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{i=1}^{K} e^{z_i}}$$

which produces a probability distribution over the K classes. The class with the highest probability is taken as the prediction.

Regularization Techniques

Logistic regression models are prone to overfitting, particularly in high-dimensional spaces with many features. To counteract this, regularization techniques are employed to enhance the model's generalizability. Regularization in logistic regression works by

imposing a penalty on the size of the coefficients, effectively encouraging the model to maintain simpler, smaller coefficients and thus reducing the risk of overfitting. This is achieved through two primary regularization methods:

- **L1 Regularization (Lasso Regression):** This method applies a penalty equal to the absolute value of the coefficients. The loss function with L1 regularization becomes:

$$J(\beta) = \left[-\sum_{i=1}^{N} [y_i \log(\sigma(X_i\beta)) + (1 - y_i) \log(1 - \sigma(X_i\beta))] \right] + \lambda \sum_{j=1}^{p} |\beta_j| \quad (2.3)$$

  It not only helps in preventing overfitting but can also lead to feature selection, as some coefficients may be reduced to zero, effectively removing less significant features from the model.

- **L2 Regularization (Ridge Regression):** In contrast, L2 regularization penalizes the square of the coefficient values. The loss function with L2 regularization is:

$$J(\beta) = \left[-\sum_{i=1}^{N} [y_i \log(\sigma(X_i\beta)) + (1 - y_i) \log(1 - \sigma(X_i\beta))] \right] + \lambda \sum_{j=1}^{p} \beta_j^2 \quad (2.4)$$

  While it keeps the coefficients small, it does not set them to zero, thus retaining all features but reducing their impact on the model.

In both methods, λ is a hyperparameter determining regularization strength. Choosing an optimal λ is critical and often achieved via cross-validation. Both methods add a layer of constraint to the logistic regression model, aiding in creating more robust and reliable predictions, especially in scenarios with complex or high-dimensional data.

Advantages:

- *Simplicity and Interpretability:* Logistic regression models are straightforward and easy to interpret, making them a good starting point for binary classification problems.
- *Probabilistic Interpretation:* It provides probabilities for outcomes, which can be a valuable way to understand the model's confidence in its predictions.
- *Performance in Linearly Separable Data:* It performs well when the decision boundary between classes is linear, and the features are roughly linearly related to the log odds.
- *Efficient and Scalable:* Logistic regression can be relatively efficient regarding computational resources, making it scalable for large datasets.

Disadvantages:

- *Linearity Assumption:* The assumption of a linear association between the predictors and the log odds of the outcome may not hold true in all scenarios. It does not perform well with non-linear data unless transformations or polynomial features are used.
- *Susceptibility to Overfitting:* In high-dimensional spaces where the number of features is large, it can overfit unless regularization techniques are applied.
- *Outlier Sensitivity:* Logistic regression can be unduly influenced by outliers. Proper data preprocessing to detect and manage outliers is crucial.
- *Issues with Multicollinearity:* The presence of highly correlated predictors can lead to unstable estimates of the model coefficients. This underscores the importance of understanding the dataset and possibly applying dimensionality reduction techniques.

2.3.2 Naive Bayes Algorithm

The Naive Bayes algorithm, rooted in Bayesian probability theory, provides a method for updating probabilities with new evidence. This algorithm is characterized by its assumption of conditional independence of features given a class label. It assumes that, within a specific class, the occurrence or non-occurrence of one feature is independent of the occurrence or non-occurrence of any other feature. While this assumption may not always hold true in complex real-world scenarios, such as in text classification, where words often co-occur in clusters or phrases, Naive Bayes remains effective and efficient in many practical applications due to its simplicity.

Mathematically, for input features $X_1, X_2, ..., X_n$ and a class label Y, the assumption of conditional independence is formulated as:

$$P(X_1, X_2, ..., X_n|Y) = P(X_1|Y) \cdot P(X_2|Y) \cdot ... \cdot P(X_n|Y)$$

In application, Naive Bayes calculates the posterior probability $P(Y|X_1, X_2, ..., X_n)$ for each class Y when classifying an instance with features $X_1, X_2, ..., X_n$. This is done using Bayes' theorem:

$$P(Y|X_1, X_2, ..., X_n) = \frac{P(X_1, X_2, ..., X_n|Y) \cdot P(Y)}{P(X_1, X_2, ..., X_n)}$$

The denominator, which remains constant across all classes, can be ignored for classification purposes. The classification decision is thus based on:

$$Y = \arg\max_{Y} P(X_1, X_2, ..., X_n|Y) \cdot P(Y)$$

By incorporating the conditional independence assumption, this simplifies to:

$$Y = \arg\max_{Y} P(X_1|Y) \cdot P(X_2|Y) \cdot ... \cdot P(X_n|Y) \cdot P(Y)$$

This approach allows Naive Bayes to compute class probabilities efficiently and make predictions, even with the simplifying feature independence assumption. Despite its straightforward nature, Naive Bayes often performs remarkably well, particularly in scenarios where the independence assumption is reasonably accurate or where the dataset is large enough to mitigate its potential limitations.

Training Process

The training of a Naive Bayes classifier involves estimating parameters within the Bayesian framework, focusing on two key aspects:

1. **Estimating Class Prior Probabilities $P(Y)$:** These probabilities are calculated based on the frequency of each class in the training dataset. They represent the initial likelihood of encountering each class before considering any feature information.
2. **Computing Likelihood Estimations $P(X_i|Y)$:** This step assesses the relationship between features and class labels. The approach to likelihood estimation varies depending on the nature of the features:
   (a) For Discrete Features: The process involves calculating the frequencies of feature occurrences in the dataset relative to each class.
   (b) For Continuous Features: It typically assumes a probability distribution (often Gaussian) for these features and calculates its parameters (like mean and standard deviation) based on the training data.

Various methodologies are employed to estimate the probabilities essential for the classifier. These methodologies cater to different data types and scenarios:

- **Frequency-based Estimation:** Used for discrete data, it calculates likelihoods from observed data frequencies.

- **Maximum Likelihood Estimation (MLE):** Primarily for continuous data, this technique fits a pre-assumed distribution (like Gaussian) to the data by adjusting its parameters.
- **Kernel Density Estimation:** A non-parametric method for continuous data, estimating the probability density function without assuming its form.
- **Laplace Smoothing:** Applied to handle zero probabilities in discrete datasets, adjusting likelihood estimates slightly away from zero.
- **Bayesian Parameter Estimation:** Integrates prior beliefs about parameters, informed by domain knowledge or previous data, updating these based on observed training data.

Each method has its specific application, with Frequency-based Estimation and Laplace Smoothing being more suitable for discrete data, while MLE and Kernel Density Estimation are preferred for continuous data. Bayesian Parameter Estimation offers a comprehensive approach applicable to both data types.

Advantages

1. *Efficiency*: Naive Bayes requires minimal training data to estimate necessary parameters, rendering it fast in operation.
2. *Implementation*: Its simplicity ensures ease of implementation.
3. *Multiclass Suitability*: It demonstrates commendable performance in multiclass classification scenarios.
4. *Adaptability*: It supports diverse likelihood distributions, such as Gaussian and Multinomial, adapting to various data types.
5. *High-dimensional Data Handling*: It has proven efficacy in tasks like text categorization with large feature spaces.

Disadvantages

1. *Independence Assumption*: The primary drawback of Naive Bayes is the assumption of feature independence, which is often not met in real-world applications.
2. *Zero Frequency Issue*: It encounters difficulties with probabilities for unseen feature-class combinations. Techniques like Laplace estimation can address this.
3. *Continuous Data Assumption*: It assumes continuous features are normally distributed, which, if unmet, impacts performance.
4. *Relative Performance*: While simple, its accuracy can be surpassed by more sophisticated classifiers, particularly where the independence assumption is violated.
5. *Probability Estimations*: Although classifications may be accurate, the derived probability estimates can be unreliable.

2.3.3 Decision Tree

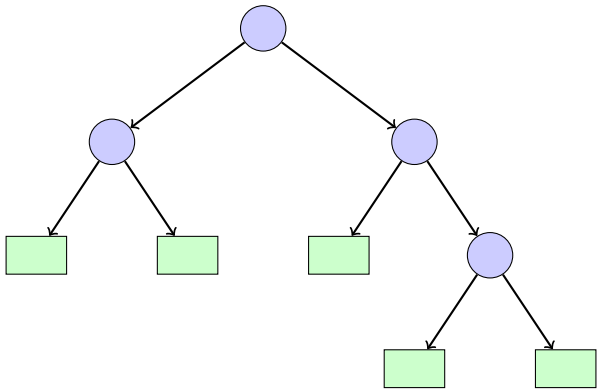


Figure 2.2: A schematic representation of a decision tree, emphasizing decision nodes represented as circles and final outcomes depicted as rectangles.

The Decision Tree is a widely used machine learning tool that employs a strategy known as recursive partitioning. This approach involves a top-down, greedy search through the possible branches, building a hierarchical tree structure based on the partitioning of datasets. Figure 2.2 illustrates a typical decision tree structure. In this tree:

- Each circular node represents a decision point, where data is split based on specific criteria.
- Arrows indicate the flow from one node to another, guiding the decision-making process.
- Rectangular leaf nodes signify the final outcomes or decisions.

This particular tree starts from a single root node and branches out into subsequent nodes and leaves. The branching nature of the tree illustrates how initial data inputs are segmented into smaller subsets, eventually leading to a decision or classification. This hierarchical, intuitive structure makes decision trees a popular choice for tasks like classification, regression, and decision-making in various domains.

Training Process and Theoretical Underpinnings

Training a decision tree involves selecting the most informative attributes for data splitting. This informativeness is quantified using metrics like entropy, information gain, and Gini impurity.

- **Entropy** measures dataset randomness or unpredictability. For binary classifications, it's defined as $E(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-)$, where p_+ and p_- are the proportions of positive and negative instances in S.
- **Information Gain** assesses entropy reduction. When dataset S is split into subsets S_1 and S_2 using feature F, information gain $IG(S, F)$ is $E(S) - \sum_{i=1}^{2} \frac{|S_i|}{|S|} E(S_i)$.
- **Gini Impurity** evaluates misclassification likelihood. For binary classifications, it's $G(S) = 1 - (p_+^2 + p_-^2)$.

The decision tree is constructed recursively:

1. **Feature Selection**: The algorithm selects the attribute that maximizes information gain or minimizes entropy or Gini impurity at each internal node.
2. **Partitioning**: The dataset is split based on the selected attribute's value. Numerical attributes use a binary split, while categorical attributes may have a multi-way split.
3. **Recursion**: The algorithm reapplies the first two steps to each subset until a stopping criterion is met.
4. **Pruning**: To prevent overfitting, pruning removes less predictive tree parts, enhancing generalizability.

Stopping Criteria : The model uses criteria like pre-specified tree depth, minimum data points in a node, or insignificant improvements in information gain or Gini impurity to balance complexity and fitting.

Advantages

1. *Interpretability*: Decision Trees allow for clear visualization and interpretation of the decision-making process.
2. *Minimal Data Preparation*: They often require less preprocessing, reducing tasks such as normalization.
3. *Versatility*: They are suitable for both classification and regression tasks and handle numeric and categorical data.
4. *Non-Parametric*: They model non-linear relationships without assumptions about the function form.
5. *Feature Importance*: Their structure provides insights into feature significance. The importance of a feature is typically derived from the number of splits that rely on the feature and the improvement in prediction accuracy it provides at each split.

Disadvantages

1. *Overfitting*: Deep Decision Trees might overfit the training data, especially in deep trees, impacting generalization. This can be addressed through pruning.
2. *Instability*: Minor changes in the training data can lead to significant tree structure changes. Ensemble methods like Random Forests can mitigate this.
3. *Optimization Issues*: Finding the optimal tree is an NP-complete problem. Common methods do not guarantee global optimality.
4. *Biases*: Imbalanced datasets can lead to biases in classification; balancing the dataset is recommended.
5. *Complex Decision Boundaries*: They are unsuitable for XOR-like problems or very smooth decision boundaries.

2.3.4 Random Forest

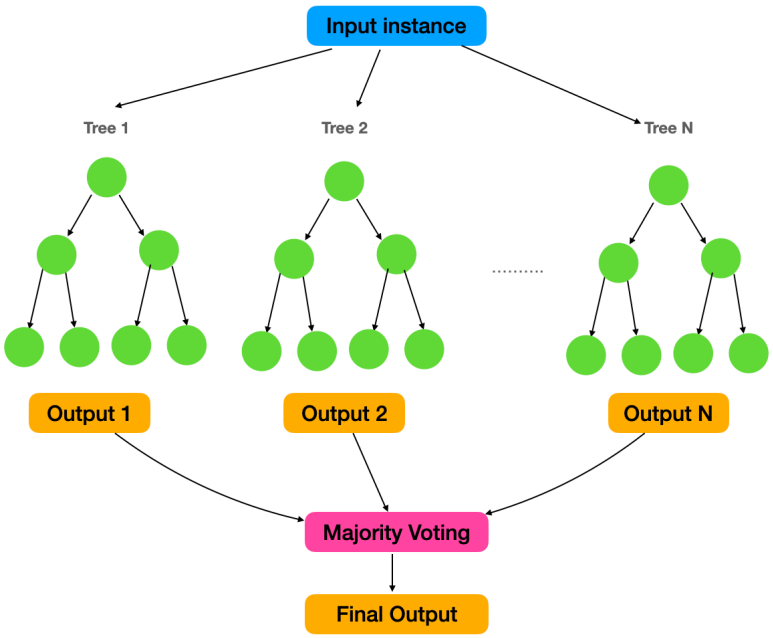


Figure 2.3: Random Forest

Random Forest employs an ensemble learning approach, utilizing a collection of decision trees to enhance the precision and robustness of classification or regression analyses. Rooted in the decision tree algorithm, it captures non-linear relationships and addresses complex data patterns by aggregating the outputs of individual decision trees. Random Forests not only refine generalization but also curtail the risk of overfitting.

The main feature of Random Forest is the use of randomness during training. Two primary sources of randomness are employed: taking random subsets of data points

and choosing random features. The random data subsets, often chosen using a method called bootstrapping, make sure each tree sees different data. Concurrently, only a few randomly selected features are considered for every decision or split in a tree. This ensures the trees in the forest are different from each other. A user-defined parameter or a heuristic measure often determines the number of features considered at each split.

Building each tree in a Random Forest is similar to building a regular decision tree. However, due to the dual randomizations, every tree is trained on unique data subsets and features. This makes sure that the trees are diverse. Measures like Gini impurity and information gain help decide the best splits in the tree. The trees keep growing until certain conditions, like a set depth or a minimum number of data points in the end parts of the tree, are met.

Prediction Process and Aggregation

During the prediction phase, the individual trees in the Random Forest independently generate their predictions. For classification tasks, each tree assigns a class label based on a majority voting scheme, where the class with the most votes is chosen as the final prediction. In regression tasks, the predictions from all the trees are averaged to derive the final predicted value. Aggregating predictions from multiple trees helps to smooth out the noise and reduce the impact of individual outliers or misclassifications. It provides a more robust and accurate prediction than a single decision tree.

Understanding Feature Importance

Random Forest gives a measure of feature importance, indicating each feature's contribution to the ensemble. Feature importance is typically estimated based on the decrease in the impurity or loss function caused by a particular feature. The more a feature contributes to reducing the impurity or improving the prediction performance, the higher its importance. This information can guide feature selection, dimensionality reduction, and understanding of underlying data patterns.

Advantages

1. *Accuracy:* Leveraging multiple decision trees, Random Forests tend to produce accurate predictions across a diverse set of problems.

2. *Feature Importance:* The model can rank the significance of different features, aiding in interpretability and feature selection.

3. *Robustness:* With its ensemble approach, Random Forest is less susceptible to overfitting than individual decision trees.

Disadvantages

1. *Complexity:* Being computationally intensive, Random Forests can be slower in both training and prediction, especially as the number of trees increases.

2. *Interpretability:* Though it offers feature importance metrics, the model as a whole is more challenging to understand than a singular decision tree.

3. *Memory Consumption:* Maintaining multiple trees requires significant memory, especially with larger ensembles.

2.3.5 XGBoost

XGBoost, short for Extreme Gradient Boosting, is a powerful machine learning algorithm that combines the principles of gradient boosting with various optimization techniques to create highly accurate predictive models. The core of XGBoost lies in the gradient-boosting decision tree (GBDT), a decision tree-based ensemble learning like Random Forest. Random Forest uses a bagging approach, which was discussed above. GBDT, on the other hand, uses a boosting approach. Boosting is an iterative process where each subsequent model is trained to improve the weaknesses of the previous models. The term "gradient-boosting" specifically refers to a boosting algorithm that minimizes a loss function by optimizing the gradients of the loss function concerning the predictions of the weak learners. The final prediction is computed by taking a weighted sum of the predictions from all the trees. In random forest, the technique of bagging is employed to reduce variance and prevent overfitting, whereas in GBDT, the technique of boosting is used to minimize bias and prevent underfitting.

XGBoost offers several adjustable parameters that can be fine-tuned to enhance the algorithm's performance. Among these parameters, the key ones include:

- **max_depth**: This parameter determines the maximum depth allowed for the decision trees in the XGBoost model. Adjusting this parameter can control the complexity and depth of the individual trees.

- **eta**: The learning rate, also known as the shrinkage parameter, influences the step size at each boosting iteration. It affects the contribution of each tree to the overall model, and a lower learning rate typically leads to a more refined model.

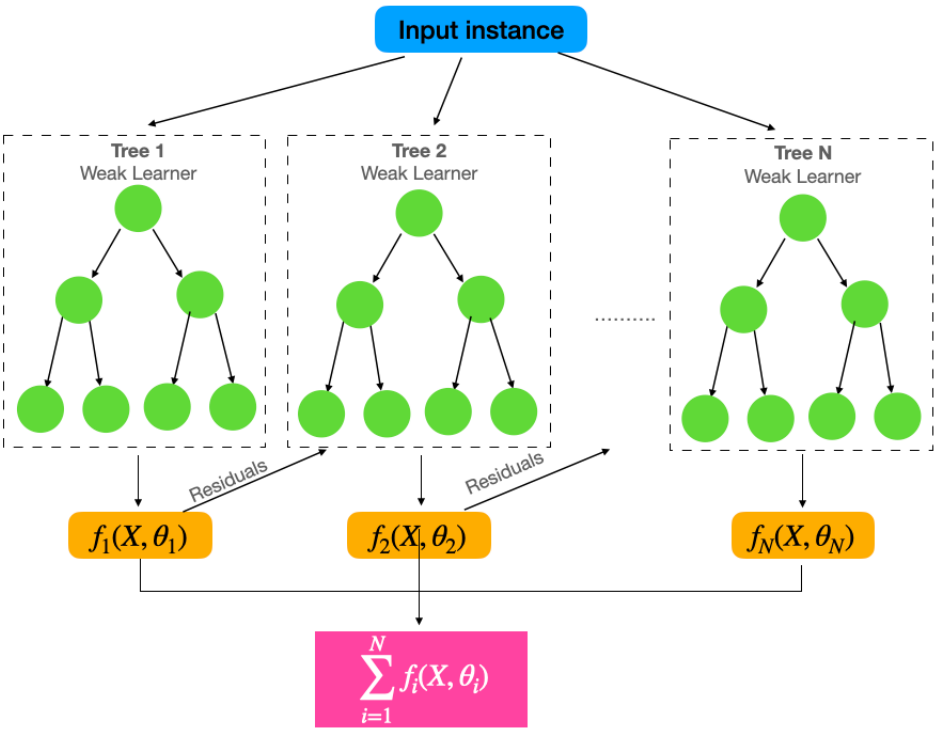


Figure 2.4: Simplified representation of XGBoost

- **gamma**: This parameter specifies the minimum loss reduction required to allow a tree split during the construction process. It helps control the algorithm's sensitivity to adding new branches and prevents overfitting.
- **subsample**: This parameter determines the fraction of the training data used to train each individual tree in the ensemble. XGBoost can introduce randomness and improve generalization by selecting a subset of the data.

By adjusting these parameters in XGBoost, practitioners can optimize the model's performance and tailor it to the specific requirements of their machine learning tasks.

XGBoost is a highly scalable and precise implementation of gradient boosting tailored to leverage the computational power available for boosted tree algorithms. Its primary goal is to optimize machine learning model performance and computational speed. Unlike GBDT, XGBoost constructs trees in parallel rather than sequentially. It follows a level-wise strategy, examining gradient values and using partial sums to assess the quality of potential splits at each possible split point in the training dataset.

Feature Importance Estimation

XGBoost provides a measure of feature importance, allowing us to understand the contribution of different features in the model's predictions. Feature importance is determined based on the number of times a feature is used in constructing decision trees and the improvement in the objective function achieved by those splits.

Understanding the importance of features helps in feature selection, dimensionality reduction, and gaining insights into the underlying data patterns.

Advantages:

- *Predictive Performance*: XGBoost achieves remarkable predictive accuracy by leveraging an ensemble of optimized weak models. This approach enables XGBoost to capture intricate patterns and produce reliable predictions.
- *Scalability and Efficiency*: XGBoost exhibits scalability through parallel and distributed computing, allowing it to manage extensive datasets and speed up model training.
- *Versatility*: XGBoost is versatile and capable of handling various data types and tasks, making it suitable across multiple domains.
- *Regularization*: The model incorporates regularization techniques, reducing the potential for overfitting and enhancing model generalization.
- *Robustness to Outliers*: XGBoost remains relatively unaffected by outliers due to its tree-based splitting methods.
- *Built-in Cross-Validation*: The algorithm can execute cross-validation at each boosting iteration, determining the optimal number of boosting rounds needed.

Limitations:

- *Computational and Memory Requirements*: XGBoost can be computationally intensive and may require significant memory, especially with larger datasets. This can pose challenges in resource-constrained settings.
- *Hyperparameter Sensitivity*: Achieving the best performance with XGBoost often necessitates careful hyperparameter tuning.
- *Interpretability Challenges*: While XGBoost models can deliver high performance, their ensemble nature can make interpretation less straightforward, especially when analyzing complex feature interactions.
- *Black-Box Nature*: XGBoost is often perceived as a black-box model, where practitioners may not grasp the intricate details of its workings, further complicating interpretation efforts.
- *Overfitting with Noisy Data*: Despite its regularization mechanisms, XGBoost can still overfit when dealing with particularly noisy datasets.

2.3.6 Support Vector Machine

The Support Vector Machine (SVM) algorithm is a robust tool in machine learning, particularly for classification tasks. Its fundamental principle revolves around identifying an optimal hyperplane that effectively separates different classes while maximizing the margin, the distance separating the hyperplane, and the nearest data points of each class. This maximization of the margin is crucial for enhancing the model's generalization ability and reducing misclassification risks. The data points that lie closest to the decision boundary or hyperplane, known as support vectors, are pivotal in defining the hyperplane's position and orientation.

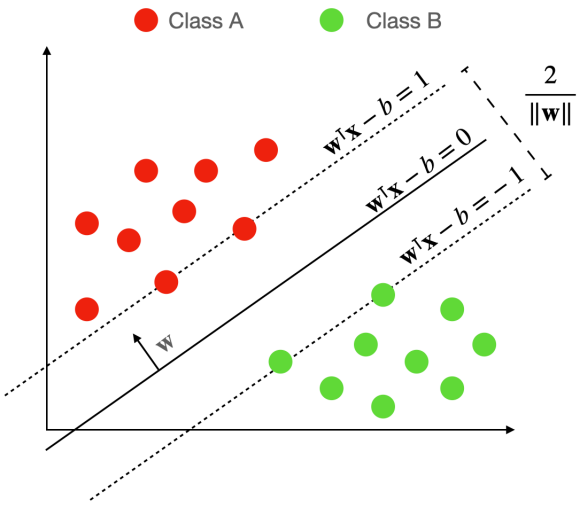


Figure 2.5: SVM with hard margin

The concept of SVM can be extended into two distinct types based on the nature of the data and the desired outcome: Hard-margin SVM and soft-margin SVM.

Hard-margin SVM: SVM represents data points as vectors in a multidimensional space, aiming to find the hyperplane that best separates the classes. In scenarios where data is linearly separable, as depicted in Figure 2.5, the hyperplane is defined by the equation $\mathbf{w}^\intercal\mathbf{x} - b = 0$. The support vectors, lying on the margins defined by $\mathbf{w}^\intercal\mathbf{x} - b = 1$ and $\mathbf{w}^\intercal\mathbf{x} - b = -1$, are instrumental in determining the hyperplane's characteristics. The training process of SVM focuses on maximizing the margin, calculated as $\frac{2}{|\mathbf{w}|}$. A larger margin indicates a better separation between classes, leading to improved model performance. This optimization is typically achieved through algorithms like quadratic programming.

Soft-margin SVM: Real-world data often presents scenarios where perfect separation is not feasible. Soft-margin SVM addresses this by introducing slack variables, allowing for some degree of misclassification while still achieving effective classification. This approach, illustrated in Figure 2.6, involves balancing the maximization of the margin with the accommodation of classification errors. The slack variables, denoted as ξ_i, measure the extent of margin violation or incorrect classification. The optimization objective becomes a balance between maximizing the margin and minimizing the sum of slack variables. The equation for the optimization objective in soft-margin SVM is:

$$\min_{\mathbf{w},b,\xi_i \geq 0} \frac{1}{2}|\mathbf{w}|^2 + C\sum_{i=1}^{n} \xi_i \tag{2.5}$$

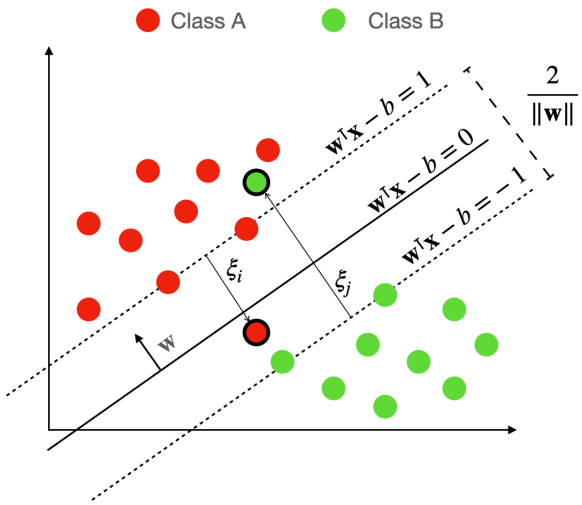


Figure 2.6: SVM with soft margin

The parameter C controls this trade-off, with larger values emphasizing minimizing misclassifications. The constraint ensures that each data point is either correctly classified or has a margin violation accounted for by the slack variable ξ_i.

In hard and soft-margin SVM, the Lagrange multipliers associated with the support vectors are crucial in defining the decision boundary and influencing the model's predictions. These multipliers highlight the most influential support vectors, offering insights into the significance of individual data points in the classification process.

2.3.7 Dual Form of SVM

Having established the primal optimization problems in hard and soft-margin SVMs, we now focus on their dual forms. The dual form of SVM is derived from the primal form using Lagrangian duality, which provides a different but equivalent way to solve the optimization problem. This transformation is mathematically elegant and computationally advantageous, especially for large datasets.

In the primal problem, we directly optimize the weights $\mathbf{w}$ and the bias b. However, in the dual problem, the focus shifts to optimizing a set of Lagrange multipliers, which correspond to the constraints of the primal problem. The utility of the dual form lies in its dependence on the dot products of input vectors, which can be effectively replaced by a kernel function in scenarios where the data is not linearly separable. This is the foundation of the kernel trick, allowing SVMs to operate in higher-dimensional spaces without explicitly computing the coordinates in these spaces.

The dual form of the SVM optimization can be represented as:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle \tag{2.6}$$

subject to the constraints:

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \quad \text{and} \quad 0 \leq \alpha_i \leq C \quad \text{for all } i \tag{2.7}$$

Here, α_i are the Lagrange multipliers, and C is the regularization parameter from the soft-margin SVM. The dual form involves maximizing the Lagrangian concerning α_i, subject to certain constraints. The beauty of the dual form lies in its ability to handle non-linear separations by incorporating kernel functions, which effectively transform the input space into a higher-dimensional space where a linear separation is possible.

The solution to the dual problem gives us the optimal values of α_i, which are then used to compute the optimal weights $\mathbf{w}$ and bias b for the decision function. This decision function is used to classify new data points, making the dual form an integral part of SVM's predictive power.

Some of the commonly used kernels in SVM are:

- **Linear Kernel**: The linear kernel calculates the dot product between the input feature vectors, which can be represented as:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$$

This kernel is suitable for linearly separable data or when the decision boundary is expected to be a straight line.

- **Polynomial Kernel**: The polynomial kernel introduces non-linearity by raising the dot product of the input features to a certain power and adding a constant term. It can be expressed as:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i \cdot \mathbf{x}_j + r)^d$$

Here, γ, r, and d are hyperparameters that control the kernel's behavior. This kernel is effective in capturing non-linear relationships between the data points.

- **Radial Basis Function (RBF) Kernel**: The RBF kernel gauges the similarity between data points based on their Euclidean distance. It can be defined as:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

Here, σ is a hyperparameter that controls the kernel's width and affects the smoothness of the decision boundary. The RBF kernel can capture complex non-linear relationships and is widely used in SVM.

- **Sigmoid Kernel**: The sigmoid kernel maps the input features to a range between 0 and 1 using the sigmoid function. It is defined as:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i \cdot \mathbf{x}_j + r)$$

The hyperparameters γ and r control the kernel's shape and steepness. The sigmoid kernel is useful for problems with sigmoid-like decision boundaries.

These kernels allow SVM to handle non-linear data by mapping it into higher-dimensional feature spaces. The choice of kernel depends on the problem at hand and the data characteristics.

Advantages

- *High-Dimensional Space Efficiency:* SVMs are particularly effective in high-dimensional spaces, which is common in many data-rich environments. They can handle the complexity and dimensional characteristics of such data without losing performance, making them ideal for applications like text and image classification.

- *Overfitting Avoidance:* SVMs are less prone to overfitting, especially when choosing the right regularization parameters. The balance between margin maximization and classification error minimization inherently prevents the model from fitting too closely to the training data.

- *Flexible Kernel Selection:* SVMs allow the use of different kernel functions to transform the input data into a higher-dimensional space. This flexibility enables the model to adapt to various types of data distributions, enhancing its applicability to a wide range of datasets.
- *Suitability for Classification Tasks:* SVMs are well-suited for both binary and multi-class classification tasks. They can create complex decision boundaries, even in cases where the data points are not linearly separable.
- *Small Sample Size Performance:* SVMs can perform well even with small sample sizes, making them useful when data collection is expensive or limited. This is particularly beneficial in fields like medical diagnosis or rare event prediction.

Limitations

- *Computational Overhead:* The training time for SVMs can be quite high, especially for large datasets. This is due to the complexity of solving the optimization problem of finding the support vectors, which can be computationally intensive.
- *Kernel Function and Parameter Sensitivity:* The performance of SVMs is highly sensitive to the choice of kernel function and its parameters. Selecting the wrong kernel or parameters can lead to poor model performance, making kernel selection and parameter tuning critical yet challenging.
- *Limited Interpretability:* SVM models, especially those with non-linear kernels, often lack interpretability. Understanding how the model makes decisions can be difficult, a significant drawback in fields where explanation and transparency are essential.

2.3.8 Neural Networks

Inspired by the human brain's neural structure, neural networks represent a powerful and extensively studied class of machine learning models. These computational models have evolved into a diverse and versatile family of algorithms, attracting substantial interest due to their exceptional accomplishments in diverse and complex tasks across various domains. Their remarkable capacity to learn from data and navigate intricate and non-linear relationships within complex datasets has led to their widespread adoption and successful application in tasks such as computer vision, natural language processing, and robotics. This section will describe a neuron, the fundamental building block of neural networks. Subsequently, we will delve into a detailed exploration of some of the most commonly used neural networks in Human Activity Recognition, namely the Multilayer Perceptron, Convolutional Neural Network, Recurrent Neural Network, and Long Short-Term Memory. The significance of each type of neural network in the

context of HAR will be discussed to provide a comprehensive understanding of their application in this domain.

Neuron

The neuron serves as a foundational element within artificial neural networks. A neuron takes multiple inputs, applies weights, sums them up, and then applies an activation function to produce an output.

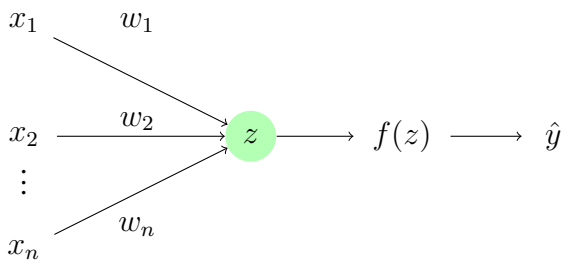


Figure 2.7: Representation of a perceptron with inputs $(x_1, x_2, \ldots, x_n)$, weights $(w_1, w_2, \ldots, w_n)$, a summation node (z), and output $\hat{y}$ after activation $f(z)$.

Figure 2.7 shows a neuron's graphical representation, showcasing the information flow. The input nodes, denoted as $x_1, x_2, \ldots, x_n$, represent the input features of the neural network. Each input node is associated with a weight $w_1, w_2, \ldots, w_n$ that signifies its impact on the final output. The inputs and weights are linearly combined at the summation node, which aggregates the weighted inputs as follows:

$$z = \sum_{i=1}^{n} w_i \cdot x_i + b$$

The term b represents the bias or intercept. It allows linear combinations that do not necessarily pass through the origin. The combined output then passes through an activation function (f), introducing non-linearity to the neural network and enabling it to model complex relationships between inputs and outputs. The final output of the perceptron is denoted as $\hat{y}$, which can be used for further computations or serve as the output of a standalone perception.

Different activation functions are used to achieve this non-linearity and introduce a wide range of transformations. Frequently employed activation functions encompass:

1. **Sigmoid function** (σ): The sigmoid activation function, denoted as $\sigma(z) = \frac{1}{1+e^{-z}}$, transforms the input z into a range between 0 and 1. It finds extensive

use in binary classification tasks due to its ability to produce probabilities for positive class membership. The sigmoid function exhibits an S-shaped curve in its plot.

2. **Hyperbolic tangent (tanh)**: The hyperbolic tangent activation function, represented as $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$, compresses input values into the range [-1, 1]. It is useful for handling data with zero mean and exhibits an S-shaped curve similar to the sigmoid function but symmetric around the origin.

3. **Softmax**: The softmax function is widely used in multi-class classification problems. Given a vector of raw scores $z = (z_1, z_2, \ldots, z_K)$, the softmax function is defined as $\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$ for $i = 1, 2, \ldots, K$, where K is the number of classes. It transforms raw scores into probability values, ensuring that the probabilities of all classes sum up to 1. The softmax function enables the model to make confident predictions and is essential for multi-class classification tasks.

4. **Rectified Linear Unit (ReLU)**: The ReLU activation function is defined as $\text{ReLU}(z) = \max(0, z)$. It introduces non-linearity to neural networks, addressing the vanishing gradient problem and accelerating convergence during training. ReLU's plot is a linear function for $z \geq 0$ and becomes flat for $z < 0$.

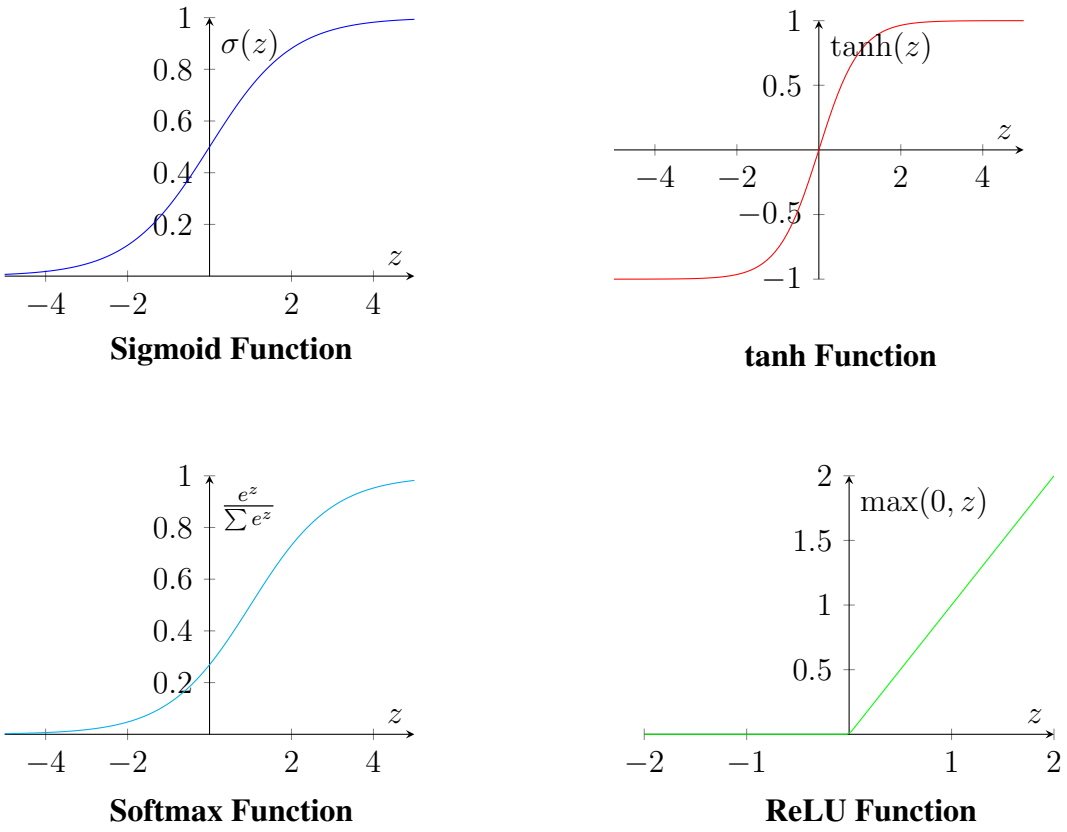


Figure 2.8: Graph depicts the behavior of four widely-used activation functions – Sigmoid, Tanh, ReLU, and Softmax.

Figure 2.8 shows the graphical representations of the four activation functions discussed above. The choice of the activation function depends on the specific problem

requirements and the neural network's architecture. Experimentation with different activation functions is common to optimize model performance.

Neural networks typically consist of multiple interconnected neurons organized in layers, allowing them to perform more intricate computations and learn higher-level abstractions from the data. These networks vary in structure and information flow, enabling them to address different problem requirements effectively. The arrangement and connectivity of perceptrons within a network are carefully designed to capture and process information in a way suited to the problem at hand. In the following subsection, we will delve into some of the variations of neural networks, exploring their architectures and applications in greater detail.

Multi-layer Perceptron

The multi-layer perceptron (MLP) constitutes a foundational form of artificial neural network extensively employed across diverse machine learning applications. It represents an expansion of the single neuron framework, incorporating multiple layers of interconnected neurons collaborating to process and obtain insights from input data. The standard architecture of an MLP encompasses three distinctive layer types: the input layer, one or more hidden layers, and the output layer. Each neuron in the input layer corresponds to a neural network's input feature or variable. Within each layer, a collection of neurons is organized, and connections between neurons in adjacent layers are comprehensive, spanning all neurons in the preceding and succeeding layers.

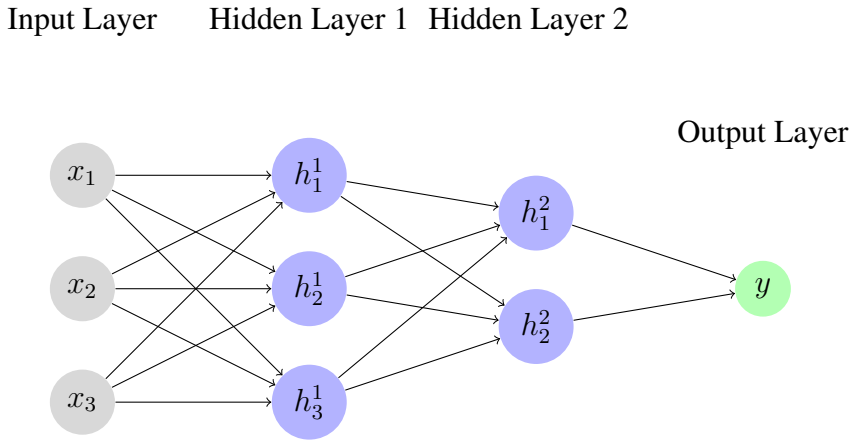


Figure 2.9: Multi-Layer Perceptron (MLP)

The diagram in Figure 2.9 represents a Multi-Layer Perceptron showcasing a spe-

cific configuration. It's important to note that the number of input features and hidden layers can vary depending on the problem at hand. As the MLP's entry point, the input layer is where the input features are first introduced. In the diagram, the input layer contains three nodes labeled as x_1, x_2, and x_3. The hidden layers act as intermediary processing stages, where the input data undergoes a series of computational transformations. The diagram has two hidden layers: "Hidden Layer 1" and "Hidden Layer 2." The number of nodes in each hidden layer is shown as three and two, respectively, represented by h_1^1, h_2^1, h_3^1 in Hidden Layer 1, and h_1^2, h_2^2 in Hidden Layer 2. It's important to note that the number of hidden layers can vary, and each hidden layer can have a different number of nodes depending on the complexity of the problem or the desired network architecture. The output layer gives the final output of the MLP. In this diagram, the output layer consists of a single node labeled as y, representing the predicted output or the result of the MLP's computation. The output layer takes the processed information from the hidden layers and generates the final output based on the problem's objectives, such as classification or regression. Each node performs a combined linear combination and activation function operation in both the hidden and output layers, similar to individual neurons. The arrows represent connections between nodes, indicating how information and computational tasks are transmitted between layers. Each arrow corresponds to a weighted connection. The weights are learned during training, adjusting the network's behavior to optimize the desired objective.

MLPs are trained using a process called backpropagation, which involves two main phases: the forward pass and the backward pass. In the forward pass, the input data is fed into the input layer, and computations progress through each layer of neurons in the network. Like the individual neurons discussed before, each neuron within the hidden and output layers performs a dual operation of linear combination followed by an activation function. The output layer produces the predicted output, and the discrepancy between this prediction and the actual target is quantified using a loss function. The objective is to minimize this loss function, thereby enhancing the model's accuracy. In the backward pass or backpropagation, the network computes the gradients of the loss function in relation to its weights and biases. The error backtracks from the output layer to the hidden layers, and the chain rule of calculus is used to calculate the gradients at each of these layers. The role of the loss function becomes crucial here, as it guides the learning process by quantifying the model's prediction accuracy. With the gradients

available, the weights and biases of the neurons in each layer are revised using an optimization algorithm, including stochastic gradient descent (SGD) or its derivatives, for instance, Adam or RMSprop. These algorithms modify the weights and biases in the direction that minimizes the loss function, facilitating faster and more efficient learning. The training process iteratively repeats the forward pass, loss calculation, backward pass, and weight update for multiple iterations or epochs. This allows the model to learn from the data and fine-tune its parameters. By iteratively updating the weights and biases through backpropagation, the MLP learns to map the input data to the correct output and eventually gains the ability to make precise predictions on new, unseen data.

Convolutional Neural Network

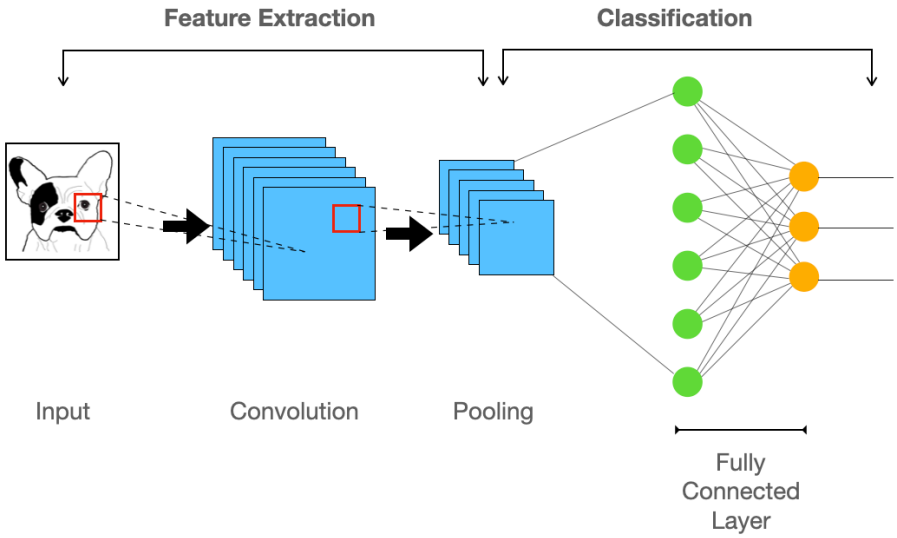


Figure 2.10: Convolutional Neural Network

Convolutional Neural Networks (CNNs) are specially designed neural networks that process grid-like data, such as images. One of the key aspects of CNNs is their ability to automatically capture spatial features from the input data. These spatial features represent patterns and structures that exist in the spatial arrangement of the data. In the case of images, these patterns can be edges, corners, textures, or shapes. CNNs capture the spatial features by employing filters or kernels in the Convolutional layer. These filters slide over the input data, detecting specific visual patterns at different locations and producing feature maps. By applying multiple filters, the network learns to recognize various spatial features and creates hierarchical input representations. For in-

stance, in image recognition, early layers of a CNN can detect basic features like edges and corners. These simple features are then combined and processed in deeper layers to recognize more complex structures, such as shapes and textures. This hierarchical approach allows the network to automatically identify intricate objects in the images, demonstrating the power of CNNs in visual tasks.

The architecture of a typical CNN, shown in Figure 2.10, comprises several essential layers, each serving specific purposes. The input layer acts as the entry point for the data, accepting the input image as a grid of pixel values. Following the Convolutional layer, the Pooling layer plays a crucial role in downsampling the feature maps, reducing the spatial dimensions to decrease the computational complexity of subsequent layers. Additionally, pooling helps the network achieve translation invariance, enabling it to recognize patterns regardless of their exact position in the input image. Finally, the fully connected layers integrate the extracted features and make the final predictions. These fully connected layers establish connections between every neuron in one layer and every neuron in the following layer, resembling the structure of a traditional neural network. The output from these layers typically represents the probabilities for different classes in classification tasks or specific values in regression models.

Training a CNN involves a process similar to training Multi-Layer Perceptrons (MLPs) with some key differences. Like MLPs, the training of CNNs also involves two main phases: the forward pass and the backward pass (backpropagation). In the forward pass, the input data is fed into the CNN's input layer, and computations progress through each layer of neurons in the network. One key aspect differentiating CNNs from MLPs is the utilization of filters or kernels in the Convolutional layer. The same set of adjustable weights is applied uniformly across all spatial positions of the input, with each filter processing the entire input data. This shared weight structure allows the CNN to detect the same visual patterns at different locations, extracting spatially invariant features. In other words, the CNN learns to recognize certain patterns regardless of their exact position in the input, significantly contributing to its ability to understand complex visual data like images. After the forward pass, CNN computes a loss function to quantify the disparity between the predicted output and the actual target, similar to MLPs. The backpropagation then to calculates the gradients of the loss function in relation to the network's weights and biases. The calculated gradients are utilized to adjust the weights and biases in each neuron layer through an optimization algorithm

like stochastic gradient descent (SGD) or its variations, including Adam and RMSprop. The training process iteratively repeats the forward pass, loss calculation, backward pass, and weight update for multiple iterations or epochs. By iteratively updating the weights and biases through backpropagation, the CNN trains to accurately associate input data with the correct output, eventually gaining the ability to make precise predictions on novel, unseen data. Using shared weights and spatially local connectivity in CNNs leverage spatial correlations within data, rendering them especially effective for applications such as image recognition, object detection, and computer vision tasks.

In summary, CNNs are powerful tools for analyzing grid-like matrix datasets, particularly images. The layered architecture of CNNs, which includes Convolutional, Pooling, and fully connected layers, enables them to automatically identify important features and make precise predictions. The learning process, driven by backpropagation and gradient descent, enables the network to optimize its filters and improve its performance.

Recurrent Neural Network

Recurrent Neural Networks (RNNs) are another specialized neural network designed explicitly for processing sequential data, such as time series, natural language, or audio. In contrast to Convolutional Neural Networks (CNNs), which excel in recognizing spatial features from grid-like data like images, RNNs focus on capturing temporal dependencies and understanding the order of events in sequential data. This temporal aspect is vital for tasks involving sequential patterns and dynamic temporal relationships.

The fundamental mechanism that grants RNNs their temporal capabilities lies in the recurrent connections within the network. These connections allow information to persist and flow through the network, enabling the RNN to maintain a memory of past inputs. As each sequence element is processed, the RNN updates its internal hidden state, which acts as a summary of the sequence up to that point. This hidden state serves as context for the network to interpret subsequent elements in the sequence, effectively providing the RNN with a context-aware perspective.

In natural language processing, for instance, RNNs can analyze the preceding words

in a sentence to provide context for predicting the next word. Similarly, in time series forecasting, RNNs can learn patterns over time to make predictions based on historical data. This recurrent nature empowers RNNs to capture long-range dependencies and contextual information, making them highly effective in tasks involving sequential data. Through this ability to process and understand sequences, RNNs have become a cornerstone in various domains, ranging from natural language understanding and translation to speech recognition and time series analysis.

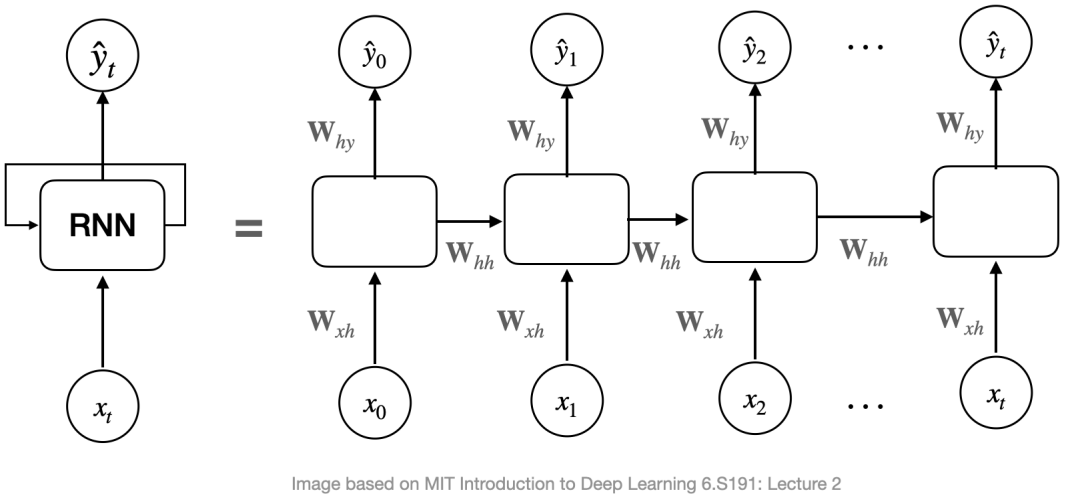


Image based on MIT Introduction to Deep Learning 6.S191: Lecture 2

Figure 2.11: Recursive Neural Network

Figure 2.11 illustrates the folded (left) and unfolded (right) representation of a Recurrent Neural Network (RNN). In the folded representation, the RNN is depicted as a single recurrent layer with connections between each time step, emphasizing the network's ability to maintain memory and capture sequential dependencies. The unfolded representation showcases the RNN over multiple time steps, making it easier to understand the recurrent connections and how information is passed from one time step to the next. In an RNN, sequential data, such as time series or natural language sentences, is processed one time step at a time. At every time step t, the RNN processes an input vector $\mathbf{x}_t$, which corresponds to the data at that specific moment. The input vectors at different time steps are fed into the RNN one after another, allowing the network to process the sequential data and generate hidden states for each time step. To achieve the recurrent behavior, the RNN employs three sets of weights: $\mathbf{W}_{xh}$, $\mathbf{W}_{hh}$, and $\mathbf{W}_{hy}$. The weight matrix $\mathbf{W}_{xh}$ connects the input to the hidden state, determining how the current input is processed and contributing to the computation of the current hidden state. On the other hand, $\mathbf{W}_{hh}$ represents the recurrent weights, connecting the hidden state at

the current time step to the hidden state at the next time step. This enables the RNN to retain information from previous time steps and capture temporal dependencies.

During the forward pass of an RNN, at each time step, the input vector $\mathbf{x}_t$ at time step t is multiplied by the input-to-hidden weight matrix $\mathbf{W}_{xh}$ to compute the activation of the hidden layer. Additionally, the hidden state from the previous time step, $\mathbf{h}_{t-1}$, is multiplied with the recurrent weight matrix $\mathbf{W}_{hh}$ to incorporate the historical information. The two weighted inputs are combined, and a non-linear activation function (e.g., sigmoid or tanh) is applied to the sum, producing the current hidden state $\mathbf{h}_t$. This process is repeated at each time step, allowing the RNN to process sequential data and generate hidden states for each time step.

Finally, the hidden state $\mathbf{h}_t$ at each time step is used to generate the output $\mathbf{y}_t$ using the output-to-hidden weight matrix $\mathbf{W}_{hy}$. The output can be either the final prediction at the last time step or an intermediate output for sequence-to-sequence tasks.

The RNN utilizes backpropagation through time (BPTT) to update the network's weights and biases during training. It extends the conventional backpropagation algorithm found in feedforward neural networks, tailored to accommodate the recurrent characteristics of RNNs and their processing of sequential data. During BPTT, the forward pass is first performed, and the hidden states and outputs for each time step are computed. Then, the loss function is calculated based on the predictions and the true targets. The backward pass is initiated by computing the gradient of the loss in relation to the output at each time step. These gradients are then backpropagated through time to update the weights and biases. To perform backpropagation through time, the gradients at each time step are accumulated over the entire sequence, as the gradients at each time step depend on the gradients from the subsequent time steps. This process is analogous to unrolling the RNN over time, creating a computational graph that spans all time steps. Once the gradients have been computed, an optimization algorithm, such as stochastic gradient descent (SGD) or its variants (e.g., Adam, RMSprop), is used to update the weights and biases in the direction that minimizes the loss function. This process is repeated for multiple iterations or epochs to iteratively refine the model's parameters.

During the training of RNNs, a common challenge that arises is the issue of vanishing and exploding gradients. As the network is trained using backpropagation through

time (BPTT), gradients are propagated backward through the layers to update the model's weights. In some cases, the gradients may become extremely small as they are propagated back, resulting in vanishing gradients. This phenomenon is especially pronounced in RNNs due to their recurrent nature, where information from earlier time steps has to pass through several recurrent connections before reaching the later time steps. If the gradients associated with these connections vanish, the RNN will have difficulty learning long-term dependencies and understanding distant relationships between input elements. On the other hand, exploding gradients can occur when the gradients grow rapidly during backpropagation. This leads to weight updates becoming extremely large, causing the model's weights to grow exponentially and resulting in unstable training. Exploding gradients can also diverge the optimization process, making it challenging to train the RNN effectively.

In practice, RNNs can be effectively used in scenarios involving sequential data where the sequence length is not too long. RNNs can capture temporal dependencies for relatively short sequences and perform well in tasks like time series prediction, natural language processing, speech recognition, and handwriting recognition. They can effectively process and learn from sequences with manageable lengths, allowing them to retain and utilize important information from past time steps. However, as the sequence length increases, traditional RNNs may encounter challenges with vanishing gradients, especially in tasks with long-range dependencies. The vanishing gradient problem can hinder the network's ability to learn and capture information from distant time steps, potentially leading to degraded performance on tasks involving long sequences. In such cases, exploring more advanced architectures like Long Short-Term Memory (LSTM) networks, discussed in the next subsection, or Gated Recurrent Units (GRUs) designed to tackle the vanishing gradient problem may be necessary. These specialized RNN variants can handle longer sequences more effectively by allowing information to flow more freely through the network and retain essential information over extended time intervals.

2.3.9 LSTM

Long Short-Term Memory (LSTM) is a specialized variant of Recurrent Neural Networks (RNNs) explicitly designed to tackle the challenges of vanishing and exploding

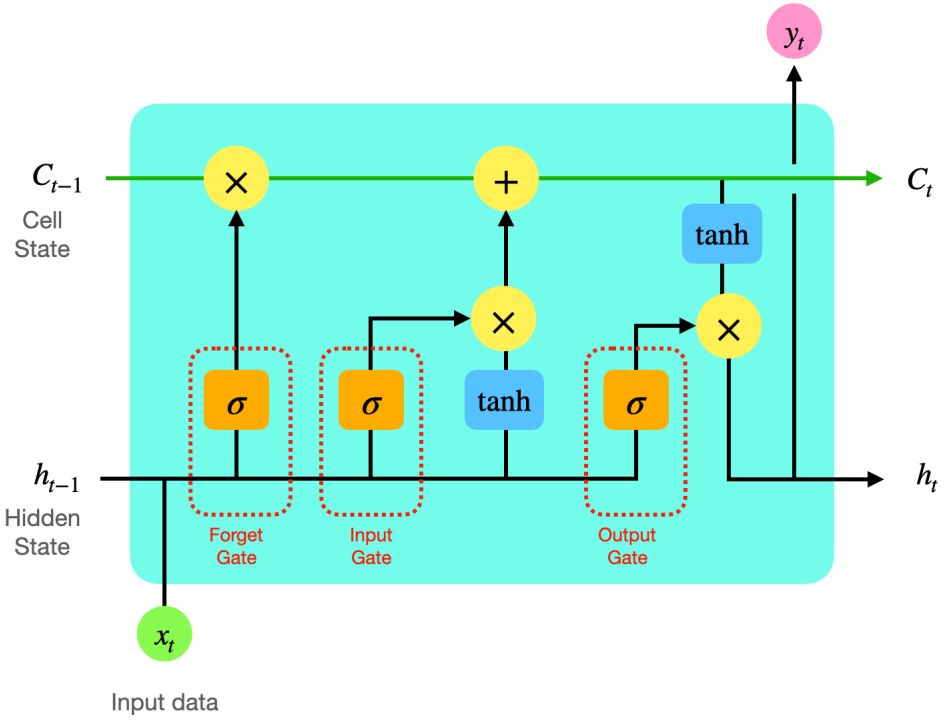


Figure 2.12: Architecture of a Long Short-Term Memory (LSTM) network

gradients during training. The distinguishing feature of LSTMs is their incorporation of a memory cell with gated mechanisms to regulate information flow, rendering them adept at processing long sequences.

Figure 2.12 shows a simplified architecture of an LSTM cell. The elements of an LSTM cell can be categorized into two primary groups: gates and states. A gate in an LSTM cell is a vital element responsible for controlling the flow of information within the cell. It uses a sigmoid activation function to produce a gate vector that selectively determines what information to allow or restrict. The states in an LSTM cell represent the memory and information storage elements. There are three gates and three states in an LSTM cell, which are described as follows:

- **Forget Gate** controls the information to be forgotten from the previous cell state. It takes the concatenation of the previous hidden state ($\mathbf{h}_{t-1}$) and the current input ($\mathbf{x}_t$) as input and produces a forget gate vector ($\mathbf{f}_t$) using the sigmoid activation function. The forget gate vector determines which elements of the previous cell state ($\mathbf{C}_{t-1}$) should be discarded.

- **Input Gate** regulates the information to be added to the cell state from the candidate cell state. Like the forget gate, it takes the concatenation of the previous hidden ($\mathbf{h}_{t-1}$) state and the current input ($\mathbf{x}_t$) as input and produces an input gate vector ($\mathbf{i}_t$) using the sigmoid activation function. The input gate vector determines which elements of the candidate cell state ($\mathbf{G}_t$) should be added to the updated cell state ($\mathbf{C}_t$). The role of the candidate gate is clarified in subsequent

explanations.

- **Output Gate** controls the exposure of the cell state as the output of the LSTM cell. It takes the concatenation of the previous hidden state and the current input, along with the candidate cell state, as input and produces an output gate vector ($\mathbf{o}_t$) using the sigmoid activation function. The output gate vector determines which elements of the updated cell state should be passed as the output hidden state ($\mathbf{h}_t$).
- **Candidate Cell State** ($\mathbf{G}_t$) provides potential new information to be added to the cell state. To compute the candidate cell state, the previous hidden state and the current input are concatenated and subsequently processed through the hyperbolic tangent activation function.
- **Memory Cell** ($\mathbf{C}_t$) results from the combination of the forget and input gates; it retains and carries forward relevant information across time steps. The updated cell state is calculated as the element-wise multiplication (Hadamard product) of the forget gate values ($\mathbf{f}t$) with the previous cell state ($\mathbf{C}t-1$) and the input gate values ($\mathbf{i}_t$) with the candidate cell state ($\mathbf{G}_t$). This process selectively remembers or forgets information over time.
- **Hidden State** ($\mathbf{h}_t$) represents the relevant information to be passed to the next time step or used for downstream tasks. The hidden state is calculated by taking the element-wise multiplication (Hadamard product) of the output gate values ($\mathbf{o}_t$) with the hyperbolic tangent of the updated cell state ($\mathbf{C}_t$).

The equations governing the LSTM cell operations are as follows:

$$f_t = \sigma(\mathbf{w}_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f)$$

$$i_t = \sigma(\mathbf{w}_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i)$$

$$o_t = \sigma(\mathbf{w}_o \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o)$$

$$\mathbf{G}_t = \tanh(\mathbf{w}_g \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_g)$$

$$\mathbf{C}_t = \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \mathbf{G}_t$$

$$h_t = o_t \odot \tanh(\mathbf{C}_t)$$

Here, $\mathbf{w}_f$, $\mathbf{w}_i$, $\mathbf{w}_o$, and $\mathbf{w}_g$ represent the weight matrices, while the corresponding bias vectors are denoted as $\mathbf{b}_f$, $\mathbf{b}_i$, $\mathbf{b}_o$, and $\mathbf{b}_g$. These learnable parameters enable the LSTM cell to adaptively adjust its values during training. The output y_t is derived by subjecting the current hidden state h_t to an additional linear transformation, and then

applying an activation function if necessary. The specific form of the transformation depends on the task being performed and the architecture of the overall LSTM model.

The training process of an LSTM follows similar principles to other neural networks but adapts to its recurrent architecture and memory cells. The training process of an LSTM involves initializing all the weights and the biases with small random values. The training data, consisting of input sequences and corresponding target sequences, are then fed into the LSTM. During the forward pass, the LSTM processes each input sequence one-time step at a time. At each time step, the LSTM computes the hidden state and the output using specific equations that involve these weights and biases. To train the LSTM, the forward pass generates predictions based on the computed output. The loss function is then computed using the predictions and the corresponding target sequences, quantifying the discrepancy between the predicted and actual values. The objective of training is to minimize this loss function by updating the LSTM's weights and biases through an optimization algorithm, such as stochastic gradient descent (SGD) or its variants (e.g., Adam, RMSprop). The optimization process involves backpropagation through time (BPTT), where gradients are computed and accumulated over all time steps. These gradients depend on the gates' activations, which, in turn, depend on the weights and biases. The accumulated gradients are then used to update the LSTM's weights and biases, iteratively improving the model's performance. By iteratively repeating the forward and backward pass for multiple epochs, the LSTM learns to model sequential dependencies, capture long-term patterns, and generate accurate predictions on new, unseen sequential data. The use of the parameters $\mathbf{W}_f$, $\mathbf{W}_i$, $\mathbf{W}_o$, $\mathbf{W}_g$, $\mathbf{b}_f$, $\mathbf{b}_i$, $\mathbf{b}_o$, and $\mathbf{b}_g$ enables the LSTM to effectively process sequential data and retain essential information for making accurate predictions.

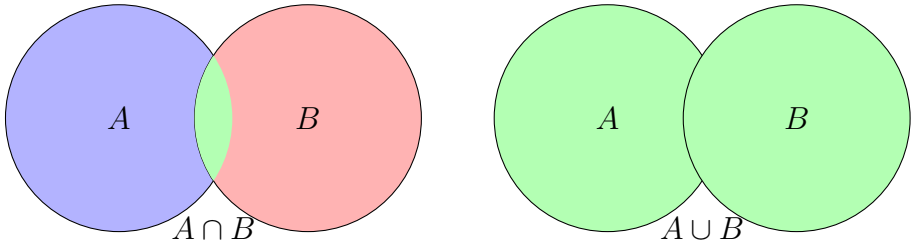
In summary, the forget gate helps selectively prevent irrelevant information from being propagated during backpropagation, thus avoiding the vanishing gradient problem. On the other hand, the input gate selectively updates the memory cell with only the important information, allowing LSTMs to retain long-term dependencies and effectively capture relevant patterns in sequential data. With the output gate, which controls the flow of information from the memory cell to the output, LSTMs can handle long-range dependencies, address the vanishing and exploding gradient issues, and perform well on tasks involving sequential data. The ability of LSTM cells to handle long-term dependencies and mitigate vanishing and exploding gradients has made them a popular

choice in modern deep learning applications.

2.3.10 Jaccard Index

The Jaccard index, named after the French mathematician Paul Jaccard, is a similarity coefficient first introduced in set theory in the late 19th century. Paul Jaccard, a botanist and geographer, developed this measure to compare the similarities between plant species in ecological studies. Till today, the Jaccard Index continues to serve as a valuable tool across numerous fields, enabling researchers to tackle a wide range of research questions and address real-world problems.

The Jaccard Index quantifies the extent of overlap between two sets and provides a normalized measure of their similarity. It is measured as the fraction of the intersection size over the union size of the sets.



Let A and B be two sets, with n elements in set A and m elements in set B. The Jaccard Index, denoted as $J(A, B)$, can be mathematically expressed as:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

where, $|A|$ represents the cardinality i.e., the number of elements in set A, and $|\cdot|$ denotes the set cardinality operator.

The value of Jaccard Index varies between 0 to 1, with 0 representing no common elements between the sets, and 1 indicating complete similarity, where the sets are exactly the same. The closer the value of Jacard Index is to 1, the more similar the two sets are, and vice versa.

The Jaccard Index is commonly employed in various domains, including information retrieval, data mining, and pattern recognition. It is particularly useful for measuring the similarity between sets when the order or frequency of elements is not relevant.

For example, it is often used in evaluating the similarity of documents based on their word sets or assessing the performance of clustering algorithms by comparing the resulting clusters to a reference set.

As a similarity measure, the Jaccard Index offers advantages such as simplicity, scale-invariance, intuitive interpretation, and applicability to various data types. However, it has limitations including sensitivity to set size differences, lack of consideration for element frequency, disregard for set order, and its restriction to set-based comparisons. Awareness of these limitations helps ensure the appropriate use and interpretation of the Jaccard Index in different contexts.

In summary, the Jaccard Index provides a normalized measure of similarity between sets, quantifying their overlap. Its simplicity and interpretability make it a valuable tool in various applications, especially when dealing with unordered data.

2.3.11 DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a density-based clustering algorithm widely used for discovering clusters in datasets with arbitrary shapes. Unlike traditional clustering algorithms, such as k-means, DBSCAN does not require the specification of the number of clusters beforehand. Instead, it identifies dense regions of data points and groups them into clusters while detecting outliers as noise. The working principle of DBSCAN revolves around two essential parameters: ϵ, representing the radius around a data point, and $MinPts$ indicating the minimum number of data points within ϵ that is needed to form a dense region. The pseudocode representation is provided in Algorithm 1.

The algorithm initiates by randomly choosing data points and searching for neighboring points within ϵ distance. If the label of the current point x is already defined, the algorithm continues to the next point. Otherwise, it performs a range query to find the neighbors of x within a distance of ϵ. If the number of neighbors is less than $MinPts$, the point x is labeled as noise, and the algorithm proceeds to the next point. If the number of neighbors is greater than or equal to $MinPts$, a new cluster is created with a unique label C. The current point x is assigned the label C, and the neighborhood of x, excluding x itself, is considered the seed set for expanding the cluster. The algorithm

Algorithm 1 DBSCAN

Inputs:
D: Database of points
$distFunc$: Distance Function
ϵ: Distance threshold
$MinPts$: Minimum number of points

procedure DBSCAN(D, $distFunc$, ϵ, $MinPts$)
  for each point x in database D **do** ▷ Go through each point
    if $label(x) \neq$ undefined **then continue** ▷ Ignore processed points
    Neighbors N := RANGEQUERY(D, $distFunc$, x, ϵ)
    if $|N| < minPts$ **then**
      $label(x)$:= Noise
      continue
    $C :=$ Next cluster label ▷ Start a new cluster
    $label(x) := C$
    Seed Set $S := N \setminus \{x\}$ ▷ Expand neighborhood
    for each point q in S **do**
      if $label(q)$= Noise **then** $label(q) := C$
      if $label(q) \neq$ undefined **then continue**
      $label(q) := C$
      Neighbors N := RANGEQUERY(D, $distFunc$, q, ϵ)
      if $|N| \geq minPts$ **then** ▷ Core point check
        $S := S \cup N$

procedure RANGEQUERY(D, $distFunc$, Q, ϵ)
  Neighbors N := empty list
  for each point x_i in database D **do**
    if $distFunc(Q, P) < \epsilon$ **then**
      $N := N \cup \{Q\}$
  return N

then enters a loop to process each point q in the seed set. For each point q, if it was previously labeled as noise, it is reassigned to the current cluster C. If the label of q is already defined, the algorithm continues to the next point. Otherwise, q is assigned the label C, and a range query is performed to find the neighbors of q within a distance of ϵ. If the number of neighbors exceeds or equal $MinPts$, the new neighbors are added to the seed set for further expansion. The algorithm continues this process until all points in the dataset D have been visited. The result is a set of clusters containing a group of densely connected points, and the noise points are labeled accordingly. The algorithm defines a separate procedure called RANGEQUERY to facilitate the range query. This procedure inputs the dataset D, the distance function $distFunc$, a query point Q, and the distance threshold ϵ. It iterates through each point x_i in the database D and checks if the distance between Q and x_i according to the distance function is less than ϵ. If the

condition is satisfied, x_i is added to the list of neighbors N. The procedure returns the list N containing the neighbors of the query point Q. At the end of the DBSCAN algorithm, all data points are either assigned to a cluster or marked as noise. The resulting clusters can have varying shapes and sizes, and the algorithm can handle datasets with irregular densities.

DBSCAN's ability to automatically determine the number of clusters and its tolerance towards noise make it a valuable algorithm in various domains, such as spatial data analysis, image processing, and anomaly detection.

Choosing the parameters When selecting the parameters for the DBSCAN algorithm, careful consideration is required to ensure optimal clustering results. The choice of parameters can significantly impact the clustering outcome, so following some guidelines and best practices is important. Here are some advice on choosing the parameters of DBSCAN:

- Understanding the Data: Gain a deep understanding of the data and its characteristics before choosing the parameters. Consider the nature of the dataset, such as its dimensionality, density, and distribution. Analyze the inherent cluster structures, including the clusters' sizes, shapes, and densities. Understanding the data will help in selecting appropriate parameter values.
- Epsilon (ϵ) Value: The epsilon value determines the radius within which points are considered neighbors. A small epsilon may result in many small clusters, while a large epsilon may merge multiple clusters into a single cluster. The choice of epsilon depends on the density of the data. A common approach is to use a density-based measure, such as the k-distance plot or the average nearest neighbor distance, to estimate a suitable epsilon value.
- Minimum Points ($MinPts$): The minimum points parameter defines the least number of points necessary to form a dense region or core point. Choosing an appropriate MinPts value is crucial to differentiate between noise points and meaningful clusters. A higher MinPts value will lead to more conservative clustering, requiring denser regions to be considered clusters. The choice of MinPts depends on the clusters' desired granularity and the data's density.
- Visual Inspection and Evaluation: Visualize the clustering results for different parameter combinations to assess the quality of the clusters. Plot the clusters and examine their coherence and separation. If ground truth labels are available, evaluate the clustering performance using external measures, such as the Rand Index or Fowlkes-Mallows Index. Iteratively adjust the parameters and evaluate the results until satisfactory clustering is achieved.
- Consider Domain Knowledge: Incorporate domain knowledge or prior information about the data when selecting DBSCAN parameters. Domain expertise can

provide insights into the expected cluster sizes, densities, and distances. Such knowledge can guide the parameter selection process and lead to more meaningful and interpretable clustering results.

- Experiment with Parameter Sensitivity: DBSCAN is known to be sensitive to parameter values, especially when the cluster densities vary significantly. Perform sensitivity analysis by running DBSCAN with different parameter combinations and observe the stability and consistency of the clustering results. This analysis helps in identifying parameter ranges that yield consistent and reliable clusters.

It is important to note that parameter selection in DBSCAN is often an iterative process that involves a balance between capturing meaningful clusters and avoiding overfitting or underfitting. Experimentation, visual inspection, and domain knowledge are crucial in making informed decisions about the parameter values, leading to effective and meaningful clustering results.

Advantages of DBSCAN

One of the primary advantages of DBSCAN is its ability to discover arbitrarily shaped clusters. Unlike traditional clustering algorithms, such as k-means, DBSCAN does not assume any specific cluster shape, allowing it to identify clusters with irregular boundaries, elongated shapes, and varying densities. Such adaptability is especially beneficial for handling datasets characterized by intricate and non-linear configurations.

Another significant advantage of DBSCAN is its ability to determine the number of clusters present in the data automatically. This feature alleviates the need for prior knowledge or user-defined parameters regarding the cluster count. By analyzing the density of data points, DBSCAN can detect densely populated regions as clusters and classify sparse areas as noise or outliers. This automatic determination of cluster count is particularly beneficial when dealing with datasets where the number of clusters is unknown or varies across different regions.

DBSCAN exhibits robustness to noise and outliers, which are common challenges in real-world datasets. The algorithm differentiates between noise points, core points, and border points. Noise points are identified as data points that do not satisfy the density requirement and do not fall within the neighborhood of any core point. By separating noise points from meaningful clusters, DBSCAN effectively addresses the presence of noisy data, enabling more accurate and reliable clustering results.

Additionally, DBSCAN is independent of the initial selection of seed points or centroids, a requirement in some other clustering algorithms. This independence makes DBSCAN less sensitive to initialization and provides robust results even with different starting configurations.

Furthermore, DBSCAN demonstrates efficient processing capabilities, particularly for large datasets. The algorithm's computational complexity is linear with respect to the number of data points, as it avoids the need for pairwise distance computations between all data points. This efficiency makes DBSCAN suitable for handling large-scale datasets, where computational scalability is critical.

DBSCAN also offers flexibility in parameter tuning, allowing users to adapt the algorithm to different datasets and achieve desired clustering results. The distance threshold (ϵ) and the minimum number of points ($MinPts$) required to form a core point are adjustable parameters influencing the cluster formation process. By fine-tuning these parameters in line with their domain knowledge and the unique features of the dataset, users can customize the clustering results to their needs.

Disadvantages of DBSCAN

Despite its numerous advantages, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) exhibits limitations and disadvantages that should be considered when applying the algorithm. Understanding these drawbacks is crucial for selecting appropriate clustering techniques in specific scenarios. Firstly, DBSCAN is sensitive to parameter selection, particularly the choice of the distance threshold (ϵ) and the minimum number of points ($MinPts$) required to form a core point. Inaccurate parameter settings can lead to inadequate cluster detection, resulting in either an excessive number of clusters or an insufficient number of clusters being identified. Determining suitable parameter values requires domain knowledge and experimentation, making the parameter selection process subjective and time-consuming.

Secondly, DBSCAN faces challenges when dealing with high-dimensional data. The algorithm's performance tends to degrade as the number of dimensions increases. This issue, commonly referred to as the "curse of dimensionality," arises due to the exponential growth in the volume of the feature space as the dimensionality expands.

Consequently, the notion of density becomes less informative, and data points tend to be widely scattered in high-dimensional space, diminishing the effectiveness of the density-based clustering approach.

Moreover, DBSCAN assumes uniform density throughout the dataset, making it less suitable for scenarios where density varies significantly across different regions. In such cases, the algorithm may struggle to capture clusters with varying densities accurately. It might fail to identify clusters with substantial density differences or erroneously merge clusters that should remain separate due to variations in local densities. Addressing varying density patterns requires careful parameter selection or the exploration of alternative clustering algorithms that can handle such variations effectively.

Furthermore, DBSCAN encounters difficulties when handling clusters with widely differing densities. The reliance on a single global density threshold (ϵ) makes it challenging for the algorithm to capture clusters with large density variations accurately. Regions with high densities may dominate the clustering process, causing the algorithm to overlook clusters with lower densities that are embedded within them. Adjusting the parameters to accommodate density variations can lead to over-segmentation of dense regions or merging clusters with different densities, impacting the quality of the clustering results.

DBSCAN's performance is also influenced by the scaling of data features and the presence of noise or outliers. Inconsistent scaling of features can affect the distance-based computations, requiring appropriate normalization or scaling techniques. Additionally, substantial noise or outliers can impact the clustering process, potentially leading to the formation of spurious clusters or affecting the determination of core and border points.

Moreover, while DBSCAN can identify clusters with arbitrary shapes, it may struggle to handle complex scenarios where clusters exhibit varying densities and intricate interconnections. Overlapping clusters or clusters with intricate structures can pose challenges for accurately capturing cluster boundaries, resulting in merging distinct clusters or creating overly fragmented clusters.

Lastly, although DBSCAN is generally efficient, it can become computationally expensive for large datasets. As data points increase, the algorithm's time and memory

requirements grow linearly, impacting its scalability. Conducting distance calculations and neighborhood searches for many data points can pose challenges and may necessitate optimizations or alternative clustering algorithms designed for scalability.

Considering these disadvantages is essential when applying DBSCAN. Understanding the algorithm's limitations, including its sensitivity to parameter selection, challenges with high-dimensional data, varying density patterns, density variations, data scaling, noise, and complex cluster structures, enables researchers and practitioners to make informed decisions and explore alternative clustering approaches when necessary.

CHAPTER 3

Literature Review

This literature review encompasses an extensive overview of the current advancements in HAR, specifically emphasizing the utilization of motion sensors in smartphones and smartwatches. The review begins by offering insights into the wide range of applications of HAR across various domains. It then proceeds to explore the different categories of sensor deployment and their implications in activity recognition. Furthermore, the review examines the different movement ranges in HAR research. In the subsequent sections, the focus shifts toward discussing various inference algorithms employed in HAR. This includes a comprehensive analysis of machine learning and deep learning techniques for activity recognition tasks. The strengths and limitations of these algorithms are highlighted to provide a well-rounded understanding of their effectiveness. Lastly, the review presents the relevant studies and research on the problems the thesis addresses.

3.1 Applications of HAR

In this thesis, we explore the use of smartphone and smartwatch technology, demonstrating their multifaceted applications in Education, Transport, Fitness, and Health. This section delves into foundational and modern research across these areas, spotlighting significant advances and situating our work in the context of the overall academic dialogue. Moreover, we aim to shed light on these systems' practical implications and real-world significance in various contexts, emphasizing their potential to transform everyday experiences and enhance the quality of life in these key sectors.

3.1.1 Fitness and Health

Smartphones and smartwatches have revolutionized the landscape of personal fitness and health monitoring, transitioning from elementary functions like step counting to

intricate analyses of sleep quality and exercise form. A pivotal study in this domain by Bao and Intille [16] advanced activity recognition using wire-free bi-axial accelerometers at five key body locations. Their innovative approach, involving the training of a decision tree with annotated sensor data, enabled the identification of twenty distinct activities. Their findings, particularly highlighting the thigh-mounted accelerometer as the most effective location for recognizing a wide range of daily activities, followed by sensors placed at the hip and dominant wrist, were instrumental in guiding the future development of wearable sensor technology for enhanced activity recognition.

Building on these earlier advancements, Kwapisz et al. [79] marked a significant transition by utilizing commercially available smartphones for activity recognition. This shift from specialized research equipment to everyday devices broadened the potential applications and accessibility of such technology. Their research, which involved participants carrying Android phones during various activities, laid the foundation for the widespread adoption of smartphone-based health monitoring. The implications of their work paved the way for subsequent innovations such as energy expenditure tracking [114], gait velocity estimation [108], workout analysis [56], and real-time feedback on sports skill development [93].

The scope of health monitoring using smart devices has significantly expanded beyond physical activities. Research focusing on sleep tracking [60, 55, 104, 25] has illustrated the capabilities of these devices in providing insights into sleep patterns and quality. This has important implications for identifying potential sleep disorders and the influence of lifestyle choices. Furthermore, smartwatches have been employed in behavioral modification strategies, notably in smoking cessation [116, 32], and in novel applications like automated dietary recording [135], illustrating the breadth of their potential in health management.

Furthermore, the utility of smartphones and smartwatches in mental health monitoring has become increasingly apparent. Studies like those by Garcia-Ceja et al. [48], which utilize sensor data for emotion classification, offer novel perspectives on continuous emotional well-being monitoring. Similarly, Saeb et al. [131] demonstrated the potential of these devices in predicting depressive symptoms by analyzing behavior patterns. The exploration of physical activity's influence on mental health disorders, as investigated by Faurholt-Jepsen et al. [45] and Quiroz et al. [122], further exemplifies

the multifaceted capabilities of wearable technology in health research. Tizzano et al.'s [152] work, employing advanced machine learning techniques, marks another stride in mood and emotion recognition through sensor data analysis.

In summary, the evolution of smartphones and smartwatches in fitness and health monitoring is marked by diversity and depth. From fundamental step counting to sophisticated behavioral and mental health interventions, these devices have significantly altered the landscape of personal health management. Our research on liquid intake tracking contributes to this evolving field, offering innovative solutions for hydration monitoring. This work aligns with the broader trajectory of wearable technology, underscoring its increasing importance and potential in enhancing health awareness and care.

3.1.2 Transportation

The advent of smartphones and smartwatches has revolutionized transportation systems, transcending their initial role as mere GPS navigation tools to become integral components in advanced transportation management. These devices have evolved into a comprehensive sensor ecosystem, capturing various transportation dynamics and significantly contributing to urban and traffic management.

A key area of focus has been transportation mode detection, aiming to discern various means of transport. This problem has garnered considerable interest due to its applications in urban planning, traffic management, and transport demand analysis. Notable contributions include Hemminki et al. [62], who developed an accelerometer-based system for transportation mode detection. This system not only overcomes GPS limitations, such as the need for unobstructed satellite views and high power consumption, but also provides more granular data on user mobility, essential for nuanced urban planning. Complementing these are studied by Dongyoun et al. [138], Fang et al. [44], and Alaoui et al. [2], each contributing sensor-based innovations to this evolving field.

Promoting public transport has also been a significant area of interest. Despite public transport being cheaper and more environmentally friendly than private transport, factors such as unpredictable waiting times often discourage its use. In response, Zhou et al. [183] advanced this domain by integrating mobile data with server-side process-

ing for accurate public transportation demand forecasting, utilizing a combination of location, temporal, and contextual data. Similarly, ComfRide [155] employs a dynamic input/output automata model in a smartphone application to suggest comfortable public transport routes based on individual preferences. BuStop [98] uses GPS data from public buses and smartphone sensors to predict bus arrival times, efficiently managing commuter expectations.

Road safety applications have seen significant advancements through the rich sensor environment of smart devices. Bi et al. [19] utilized these sensors to monitor driver behavior, enhancing road safety. Addressing pedestrian safety, Vinayaga et al. [156] developed a real-time system for detecting pedestrian distractions using sensors in mobile and wearable devices, prioritizing efficiency, accuracy, and energy conservation. Dunlop et al. [41] employed smartphone sensors for crowdsourcing road safety information, while Dong et al. [37] focused on evaluating road surface conditions using similar sensor data. The SmartRoad system [67] leverages participatory sensing to detect traffic regulators, offering a cost-effective alternative to traditional surveys.

Another innovative application is automatic spot detection in crowded areas. Our research on indoor parking event detection using a combination of WiFi signals and smartphone accelerometer data represents a significant effort in this area. This approach excels beyond traditional GPS-based systems in environments where GPS signals are poor or absent, and a detailed discussion is provided in section 3.5.2.

In conclusion, incorporating smartphones and smartwatches into transportation systems has led to innovative solutions that enhance efficiency, safety, and user convenience. From mode detection and traffic analysis to road safety and real-time updates, these technologies are reshaping transportation infrastructure management and offering significant benefits to users and urban planners alike.

3.1.3 Education

In the educational context, the applications of smartphones and smartwatches are predominantly categorized into two interconnected realms: accessible learning and behavioral analytics.

Accessible learning is dedicated to removing barriers in education, ensuring inclu-

siveness for all students, regardless of their abilities. This includes using smartphone and smartwatch technologies in innovative tools like Braille interface [94, 36] and sign language translation [66, 165], catering to specific learning needs. Further enhancing this inclusivity, Augmented Reality (AR) and Virtual Reality (VR) based smartphone apps play a crucial role. AR/VR apps make learning STEM subjects [141, 12, 11, 64] making them more engaging and immersive, eliminating the need for costly equipment. VR technology, applied in areas like history [128] and geography[117], offers virtual experiences of locations like historical sites and physical landscapes, enriching students' understanding. Additionally, some works have focused on the design aspect of smartphone-based learning [112, 167], making educational applications more intuitive and enjoyable, enhancing student engagement and learning efficiency.

Behavioral analytics enhances personalized learning by examining students' behavioral and physical activities, identifying patterns and learning behaviors. Notable in this field are projects like StudentLife [162] and its extension, SmartGPA [163]. The StudentLife study, using a monitoring app, evaluated how academic workload affects students' stress, sleep, mood, social interactions, mental well-being, and academic performance. It found that students start with high positivity, low stress, and healthy sleep and activity habits, but these decline as academic pressure increases. Building on this, the SmartGPA study further investigates how smartphone data can uncover the impact of cognitive, behavioral, and environmental factors on college students' learning. Analyzing behaviors like study and social habits over a 10-week term, SmartGPA identified strong correlations between these behaviors and students' GPAs. It developed a predictive model that accurately estimates cumulative GPA from smartphone behavior data, aligning closely with academic results. These insights are vital for customizing educational content and approaches to meet individual needs, particularly for students needing extra support. Behavioral analytics thus becomes instrumental in personalizing and enhancing the accessibility of education.

This thesis proposes a predictive analytics tool by tracking writing micro-events, a novel approach to understanding and supporting diverse learning needs. This investigation into writing patterns can help identify areas where students need extra assistance and aims to contribute to the development of more personalized and inclusive educational technologies.

3.2 Human Activity Detection based on Sensor Deployment

Sensor deployment in human activity recognition, encompassing MEMS and other technologies, is typically categorized into three types: wearables, object-embedded, and environmental [69]. Wearable sensors, often used for health monitoring and fitness tracking, are attached directly to the body. Object-embedded sensors, integrated into everyday items, are pivotal in smart home and interactive device applications. Environmental sensors, placed in specific locations, play a crucial role in systems such as security and assisted living. This section aims to delve into how these varied sensor deployments are utilized in human activity recognition, highlighting their respective strengths and weaknesses. While wearable sensors form the primary focus of this thesis, our research also incorporates environmental and object-embedded sensors to broaden our study's scope and address specific challenges. This integrated, hybrid approach overcomes some inherent limitations of solely using body-mounted sensors, as detailed in the following sub-sections.

3.2.1 Wearable

Wearable sensor deployment integrates sensors into items worn on the body, ranging from commonly used devices like smartwatches and fitness bands to more specialized forms such as smart glasses and sensor-embedded clothing. These wearables have become crucial in human activity recognition and personal health monitoring, providing a discreet but effective way to continuously track various physiological and physical parameters. Notably, their design focuses on being lightweight and comfortable, ensuring they fit seamlessly into daily life without being obtrusive, all while gathering valuable health and activity data.

The affordability and multifunctional nature of smartphones and smartwatches have made these devices a common possession for most people. Smartwatches are traditionally considered wearables. However, smartphones, which are objects of daily interaction, are also explored as wearable devices. This is due to their constant presence with the user, whether held in hand, carried in a pocket, or placed in a bag. This the-

sis specifically focuses on detecting human motion utilizing smartphones and smartwatches. Equipped with various sensors, these devices are instrumental in a range of applications in HAR, as explored in Section 3.1. They are particularly effective in fitness and health monitoring, enabling the tracking of energy expenditure [114], the measurement of gait velocity [108], and the analysis of sleep patterns [60, 55, 104, 25]. Additionally, their application in mental health and behavioral studies is noteworthy, assisting in areas such as emotion classification [48], the detection of depressive symptoms [131], and the study of student behavior and well-being [162, 163]. Beyond health and wellness, these devices also play a critical role in enhancing road safety [19, 156] and supporting specialized educational needs [94, 36, 66, 165]. The diversity of these applications highlights the significant role that smartphones and smartwatches play in various facets of HAR.

Earables, also known as smart earphones, represent another widely-used device that has significantly evolved in the wearable technology sector. Originally designed for audio purposes, they have now transitioned to include IMUs and physiological sensors, which has expanded their role in human activity analysis. For instance, Hossain et al. [65] implemented an earable device with a microphone, a 6-axis IMU, and dual-mode Bluetooth to track head and mouth-related activities, including headshaking, nodding, speaking, and eating. Another research [123] initiative proposed using an in-ear earable device coupled with an IMU sensor on weight equipment, functioning as a personal digital trainer that provides real-time feedback during free weight exercises. These innovative applications exemplify the practical implementations discussed in Choudhury's paper [31], which delves into the evolving landscape of earables, highlighting the challenges, opportunities, and wide-ranging applications of this technology. These developments indicate a promising future for earables in enhancing personal health management and fitness training.

Smart glasses represent another innovative wearable that integrates advanced computing capabilities into a glasses-like frame. These devices often resemble traditional eyeglasses but are equipped with features such as displays for augmented reality, cameras for capturing and AR applications, and wireless connectivity for syncing with other devices. They are ideal for hands-free computing and navigation. In HAR, smart glasses are applied across various categories: Health and Fitness Monitoring [179, 33, 1] to track activities and health; Gesture and Eye Movement Recognition for interface con-

trol [102, 170]; Environmental Interaction for informative navigation [61, 153, 42]; Assistive Technology for disability support [72, 103, 169]; Workplace Productivity and Training [90, 132, 147] for professional development; and Social Interaction and Behavioral Analysis [121, 124] for studying human behavior. These categories leverage the glasses' unique capabilities like visual recording and sensor integration, exploring diverse aspects of human activity.

In terms of specialized wearables, sensor-embedded shoe soles are instrumental in gait analysis [88, 27], particularly beneficial for elderly Parkinson patients [27]. Finger rings, integrating sensors, have applications ranging from sleep tracking [4] to women's safety [142]. Smart clothing, integrating textile-based sensors, is revolutionizing data collection in sports performance and rehabilitation [137]. Lastly, smart patches, adhering directly to the skin for continuous health monitoring, have shown significant promise in areas like glucose level tracking, sweat diagnosis, and electrophysiology [91, 73].

Chest straps are another wearable pivotal in sports science and patient monitoring due to their accurate heart rate tracking capabilities [119]. In field sports such as football, rugby, and hockey, players commonly utilize body-mounted sports performance trackers [85]. These compact devices are equipped with GPS units, and various sensors are invaluable for collecting detailed movement and exertion data. This information is crucial for performance monitoring, offering coaches and players real-time feedback, and providing tactical insights for game strategies and training plans.

Wearable technologies stand out for their capacity to provide continuous, personalized monitoring, irrespective of the user's environment. They excel in gathering real-time data that closely mirrors the user's physical activities and physiological states, offering a unique perspective into individual health and behavior patterns, yet they face notable drawbacks. Among these are limited battery life, necessitating frequent charging, and discomfort or poor ergonomic design, which can impact user adoption and compliance. Accuracy and reliability of data are also concerns, as sensor readings can be affected by various external factors. Privacy and security issues arise with handling sensitive personal data, and the cost of advanced devices may limit their accessibility. Additionally, wearables often have limited processing capabilities and may depend on other devices to function fully, posing a challenge for users without such companion

technology.

3.2.2 Environmental

Environmental sensors are designed to detect and measure specific environmental elements. They capture minute environmental variations, providing essential data for recognizing and analyzing human activities. Atmospheric sensors, such as temperature, humidity, air quality, pressure, and gas sensors, provide data about the ambient environment. Temperature and humidity sensors are used to infer the presence of individuals in indoor spaces or to detect specific activities that affect environmental conditions, like cooking or exercising. Air quality sensors are crucial in identifying activities that alter air composition, such as cooking [50] and smoking [113]. Gas sensors contribute to safety monitoring by detecting hazardous gases, which can indicate specific human activities or environmental risks.

Spatial and Signal-Based Sensors, including audio, RF, WiFi, and GPS sensors, offer diverse capabilities. Audio sensors capture sounds used to analyze sound events [176] and human-to-human interactions [3]. Radiofrequency communication, such as WiFi, radio frequency identification devices and Bluetooth, are particularly useful in indoor positioning systems [177], helping to track human movement and occupancy in environments where GPS is ineffective. RF signals have also been used for activity recognition and sleep posture monitoring [175, 96]. Non-contact health monitoring systems hold particular promise for long-term health monitoring. For example, Uysal et al. [154] proposed a real-time respiratory rate monitoring system, while Khamis et al. [76] introduced a radio-based device-free system for automated hand hygiene monitoring. GPS sensors provide precise outdoor location data, essential for applications like fitness tracking [85] and outdoor navigation [115].

Optical and infrared sensors, such as light and infrared sensors, can detect environmental visual and thermal changes. Light sensors can determine changes in lighting conditions caused by human presence or movement, aiding in applications like automated lighting control in smart buildings. Infrared sensors are used for thermal imaging and presence detection, offering a non-intrusive way to monitor human activity, especially in low-light conditions [140, 150, 157]. These sensors are effective in ensuring

energy efficiency, enhancing security, and providing valuable data for ambient-assisted living applications.

In conclusion, environmental sensors offer a versatile and non-intrusive solution in the field of human activity recognition, with diverse applications spanning healthcare, smart home automation, and indoor navigation. While challenges such as environmental variability and sensor placement persist, these sensors' adaptability and broad applicability in HAR are clearly demonstrated through these varied studies.

3.2.3 Object-Embedded

Object-embedded sensor deployment is an approach where sensors are attached to everyday objects, enabling the detection of human activities through interactions with these objects. This method excels in providing context-specific information, greatly enhancing the accuracy of activity recognition. However, it involves the extensive instrumentation of many objects, which can be both labor-intensive and costly.

Several studies have contributed significantly to this field. Chen et al. [28] developed a smart toothbrush equipped with inertial sensors. Using a recurrent probabilistic neural network (RPNN), this toothbrush not only tracks its movements but also assesses the effectiveness of brushing techniques, turning a routine activity into a valuable data source.

Pentelligence [133] made strides in handwriting digit recognition. Their system, which detects the motion and sound of a pen on paper, showcases the adaptability of object-embedded sensors in capturing subtle human activities.

In healthcare, Landero et al. [51] introduced a system using magnetic door sensors connected to a Raspberry Pi to monitor cupboard door openings. This non-intrusive setup serves as a continuous memory assessment tool, aiding in the early detection of cognitive impairments.

The Smart Sensory Furniture Project [21]is introduced as an ambient assisted living system specifically designed for monitoring elderly individuals. This system incorporates sophisticated sensor technology into furniture to safeguard the safety and well-being of elderly individuals living alone by identifying potentially hazardous actions

and atypical behavioral patterns.

Li et al. [84] proposed the use of passive UHF RFID tags attached to everyday objects to enable the detection of human-object interactions, facilitating applications like interactive storytelling with toys, monitoring daily activities at home, and analyzing customer browsing behaviors in retail settings.

Despite their detailed context-specific insights, object-embedded sensors face challenges in scalability and cost-effectiveness, especially when compared to wearables or environmental sensors. The need for extensive object instrumentation is a considerable hurdle, particularly in larger or more dynamic settings. Therefore, their use is often most appropriate in specialized situations where detailed interaction data is crucial.

3.3 Based on the Range of Movement

Human activities can be broadly classified into three types based on the range of movement involved. These are: full-body motion activities, involving movement of the entire body; free-arm motion activities, characterized by unrestricted arm movements; and precise hand and arm motion activities, which entail specific movements within a limited range. Understanding these movement ranges in HAR is essential, as it guides the tailoring of MEMS sensors to specific movement types. This targeted approach enhances data collection, ensuring that relevant data is captured efficiently with minimal extraneous noise. It also facilitates the customization of algorithms suited to each movement category. Importantly, this tailored approach contributes to optimizing energy consumption, a vital consideration in portable and wearable technologies where battery longevity is crucial. In the following subsections, we will delve into research studies focused on each movement range, examining the specific data collection methods and algorithms applied to each.

3.3.1 Full-Body Motion

Full-body motion activities encompass actions that involve the displacement of the entire body. These activities range from simple daily tasks like walking, jogging, and climbing stairs to more complex actions such as fall detection, sports participation,

yoga, and physiotherapy.

Numerous studies have proposed systems aimed at recognizing and assisting with these activities. For example, Gupta et al. [57] developed a system to support amateur yoga practitioners in learning proper execution without a trainer, which is particularly useful during situations like the pandemic. This system utilizes motion sensors like accelerometers and gyroscopes to recognize the 12 steps of sun salutation and evaluate their correctness. The deep learning model incorporates convolutional layers for yoga step recognition and provides feedback on execution speed and angular deviation from standard postures. The system's effectiveness was validated using a prototype and training data collected from professional yoga trainers.

In another study, Mekruksavanich et al. [100] employed a pre-existing dataset to recognize activities such as office work, reading, writing, resting, playing games, eating, and cooking using a complex activity detection approach.

Furthermore, Yu et al. [174] presented a fall detection system based on a hidden Markov model (HMM) that utilizes a single-motion sensor for real-world home monitoring of elderly individuals. This system incorporates a new acceleration signal representation and a sensor orientation calibration algorithm to address sensor misplacement issues effectively. The system demonstrates high accuracy and outperforms benchmark systems, highlighting its potential for precise fall detection in real-life scenarios with a low false alarm rate.

Collectively, these studies emphasize the potential of sensor-based systems in recognizing and assisting with full-body motion activities. By leveraging motion sensors and advanced algorithms, these systems offer valuable support and feedback to individuals engaging in various physical activities.

3.3.2 Free Arm Motion

Free arm movement activities primarily involve unrestricted arm motion, allowing for a wide range of movements. These activities can encompass actions such as drinking, smoking, eating, waving, lifting objects, throwing a ball, performing bicep curls, painting or drawing, and playing certain musical instruments like the violin or guitar.

Several studies have proposed systems to recognize these activities. For instance, Bavan et al. [17] explored using a single inertial sensor and supervised machine learning techniques to classify shoulder rehabilitation activities. The results showed promising differentiation between patient exercises and non-specific movements, potentially guiding treatment strategies and improving patient engagement.

In a different context, Wang et al. [164] proposed a swimming performance monitoring system based on wearable inertial sensors. A single sensor node placed on the lumbar region collects raw data from four swimming styles. The system employs data fusion and statistical analysis methods to extract posture features and information, followed by an action recognition method based on a Hidden Markov Model. The system demonstrates high recognition accuracy, offering valuable insights for future swimming training applications.

Furthermore, Anand et al. [8] focused on swing sports such as tennis, badminton, and golf. The study proposes a sports analytics system that can effectively distinguish players' hand movements, detect shots, and compare two novel techniques for shot classification. By leveraging correlation-based feature selection with mRMR and CNN-BLSTM neural networks, the developed system provides valuable insights for player improvement and injury prevention. The commercialized applications, TennisTraq and ShuttleTraq, are available on the Samsung Galaxy Appstore. Collectively, these studies highlight the potential of sensor-based systems in recognizing and assisting with free-arm movement activities.

3.3.3 Precise Hand and Arm Motions

Precise Hand and Arm Motions activities entail precise or specific motions confined to a limited or restricted range, often for specific tasks or to interact with objects. Examples of these activities are typing, writing, brushing teeth, hand washing, making finger gestures, tapping as input, using a computer mouse, sewing, knitting, playing certain musical instruments like the piano or keyboard, and assembling small objects such as puzzles or model kits.

MoLe [160] exploits motion sensors from a smartwatch to infer keyboard typing. The authors demonstrate that motion sensors in smartwatches can be used to infer fine-

grained hand movements and, by extension, the keys pressed on a keyboard. This work highlights the potential privacy risks of wearable devices with motion sensors.

Serendipity [168] uses the built-in movement sensors in regular smartwatches to recognize small finger movements. It can tell the difference between the five types of finger movements, like pinching, tapping, and rubbing fingers, with a high degree of accuracy. The authors claim that this technology could be useful for interacting with different devices or researching how we move our fingers and hands.

Luo [95] presents a system that uses a wrist-worn IMU sensor to monitor tooth-brushing activities accurately. It is designed to provide real-time feedback on the user's brushing technique, helping to improve oral hygiene habits. The authors demonstrate the effectiveness of their system through a series of experiments, showing that it can accurately detect different brushing techniques and provide useful feedback to the user.

TapSkin [178] uses a smartwatch's built-in IMU sensors to recognize on-skin input to detect different types of taps on the skin around the watch, providing a new method of interaction for smartwatch users.

These studies demonstrate the potential of sensor-based systems in recognizing and enhancing precise hand and arm motions.

3.4 Inference Algorithm used in HAR

Over the past two decades, IMU-based HAR has witnessed remarkable progress by adopting and advancing different forms of algorithms. In this section, we provide an overview of the evolution of HAR algorithms, highlighting their strengths, limitations, and the specific requirements they cater to in different HAR tasks. We grouped the algorithms based on their methodological similarities, chronological appearance in the field, and the nature of the data they are prepared to handle.

3.4.1 Machine Learning Approaches for Human Activity

The early studies in this field primarily focused on the analysis of accelerometers [22, 47] to detect human movement. These works utilized correlation analysis to investigate

the relationship between sensor data and various types of movements, with an emphasis on establishing associations or correlations between accelerometer readings and specific activities or gestures.

As the domain evolved, researchers grew interested in applying machine learning approaches to the recognition and classification of activities using data from accelerometers. A range of fundamental machine learning approaches, including Decision Trees [16, 136], k-Nearest Neighbors (kNN) [47, 46, 79, 126], Naive Bayes [38, 16], and Logistic Regression, were initially investigated.

The k-Nearest Neighbors (kNN) algorithm is a classification method based on the proximity of data points in feature vector spaces and their corresponding classes. By measuring distances between data points, kNN determines the class membership of a given sample. Foerster et al. [47] were among the first to apply the k-Nearest Neighbors (k-NN) classification approach in their research. Using time-domain features extracted from three uni-axial accelerometers, they employed this method to differentiate nine human activities. In a subsequent study [46], they expanded upon this approach by combining k-NN with a hierarchical decision method, incorporating frequency-domain features for the same activity recognition task. kNN has also been utilized in other studies, such as those conducted by Kwapisz et al. [79] and Ranjan et al. [126], for activity recognition purposes.

Decision Tree is another method that has gained widespread usage. It has a distinctive inverted tree structure, wherein each branch node makes decisions based on specific feature values. The interpretability of decision trees is a notable advantage, as their structure allows for easy comprehension and analysis. In 2004, Bao and Intille [16] conducted a study employing decision-tree learning algorithms on accelerometer data, emphasizing the potential of accelerometers in HAR. Similarly, Sen et al. [136] employed a decision tree to detect eating movements in their research.

Naive Bayes assumes feature independence given the class, simplifying probability calculations by treating each feature as independent and contributing to the overall classification decision. Despite its naive assumption, Naive Bayes has demonstrated effectiveness in various classification tasks, utilizing the probabilistic framework of Bayes' theorem to estimate class likelihood based on observed features. [38, 16].

Using elementary machine learning techniques, such as kNN, Decision Trees, and Naive Bayes, has been prevalent in HAR studies due to their comprehensibility and simplicity. These techniques have demonstrated their effectiveness in accurately identifying and classifying basic activities, such as walking, running, sitting, and standing. However, their limitations become evident when faced with activity recognition tasks involving non-linear feature relationships. To address the challenges associated with activities that involve non-linear relationships, researchers have increasingly turned to more advanced algorithms such as Random Forests (RF) and Support Vector Machines (SVM). These sophisticated algorithms offer enhanced capabilities in capturing intricate patterns and modeling non-linear relationships within the feature space.

The SVM is a supervised machine learning that identifies an optimal hyperplane that effectively separates different classes within a dataset. The primary goal of an SVM model is to maximize the margin between these classes. Kernel methods are utilized in SVM to convert data that is not linearly separable into a higher-dimensional space where it can be separated linearly. Though SVMs are inherently binary classifiers, they can be extended to multiple classes using one-vs-rest or one-vs-one schemes. One significant advantage of using SVM over other algorithms, such as k-Nearest Neighbors or Decision Trees, is its ability to handle non-linear and high-dimensional datasets. Despite their robustness, SVMs can be computationally expensive and time-consuming for larger datasets with high-dimensional feature spaces.

Anguita et al. [9] employed a multiclass hardware-friendly SVM for activity recognition on smartphones. The activities detected included walking, standing, and sitting, among others, demonstrating the versatility of SVMs in recognizing a range of human activities. In the healthcare sector, SVMs have been used for critical applications such as fall detection. Doukas and Maglogiannis [39] combined movement data from wearable sensors with sound data to detect falls in elderly patients using SVMs. Similarly, Zhang et al. [181] used a one-class SVM algorithm, a variant of the standard SVM used for outlier detection, for fall detection using wearable sensors. These studies highlight the effectiveness of SVMs in detecting critical events like falls, which can be crucial in healthcare monitoring systems. Furthermore, Huynh and Schiele [71] utilized SVMs for activity recognition from wearable sensors to reduce the amount of supervision required in activity recognition. This work underscores the potential of SVMs in creating more autonomous HAR systems.

Tree-based ensemble learning approaches, such as Random Forest, XGBoost, LightGBM, and AdaBoost, are machine learning methods that leverage the power of combining multiple decision trees for making predictions. These approaches create a collection of decision trees and utilize their collective predictions to arrive at a final prediction. Random Forest employs the bagging technique, where each tree is trained independently on different subsets of the data to reduce variance and overfitting. A practical application of this method is found in the work of Likforman-Sulem [87] et al., where Random Forest was employed for emotional state recognition from handwriting and drawing. This application demonstrates the versatility of ensemble methods in recognizing complex human activities. Parate et al. [116] also utilized Random Forest to detect the hand movement of smoking.

On the other hand, XGBoost and LightGBM utilize the boosting technique, where the models are trained sequentially, with each subsequent model focusing on correcting the errors of the previous models. XGBoost and LightGBM further enhance the boosting approach by introducing optimized algorithms for more efficient and accurate model training. They differ from each other in their algorithmic implementation, handling of categorical features, and performance optimizations. An example of an effective application of XGBoost is seen in the work of Zhang [182] et al., where XGBoost was used for indoor activity recognition using smartphone data. This study showcases the efficacy of gradient-boosting methods in managing high-dimensional sensor data. On the other hand, Csizmadia et al. [34] utilized LightGBM for activity recognition in children using wearable devices. Their study underscores the potential of ensemble methods in accurately recognizing a broad spectrum of activities.

Probabilistic Models

Probabilistic graphical models utilize graphs to represent and understand uncertainty in a system. In HAR, which involves recognizing and understanding human activities from sensor data, Hidden Markov Models and Conditional Random Fields are frequently used. These models are particularly helpful for capturing the sequential patterns and relationships between different activities based on the observation IMU sensor data. These models are particularly well-suited for activities that exhibit sequential patterns and dependencies as they help capture the temporal dependencies and relationships

between different activities.

Hidden Markov Models (HMMs) are probabilistic models that learn the joint distribution of observed and hidden variables during training. In HAR, the sensor readings are the observed variables, and the activities are the hidden variables. HMM uses the learned joint distribution to compute conditional probabilities using Bayes' rule for prediction. HMMs aim to maximize the probability of the observed data through Maximum Likelihood Estimation (MLE) during training. The structure of HMMs is such that each observed state is contingent only upon the hidden state, and every hidden state is determined solely by the previous hidden state. The output in HMMs is typically obtained through the Viterbi algorithm or the forward-backward algorithm. The Viterbi algorithm identifies the most probable sequence of hidden states responsible for producing the observed data, while the forward-backward algorithm computes the posterior probabilities of the hidden states at each time step. The predicted output in HMMs is based on the most likely sequence of hidden states or the probabilities of the hidden states given the observed data. SwimSense [164] used HMM on data collected from inertial sensors node mounted on the lumbar area to analyze swimming postures. Yu et al. [174] created a fall detection system using HMM and motion sensor technology. The system utilizes a single motion sensor and introduces a new representation for acceleration signals in HMMs, eliminating the need for manual feature engineering.

Conditional Random Fields (CRFs) directly learn conditional probabilities without explicitly modeling the joint distribution. In contrast to HMM, CRFs do not have the same strict dependencies. In CRFs, the observed state is conditioned on both the current hidden state and the neighboring hidden states. This flexibility enables CRFs to capture more complex dependencies and consider a wider context of hidden states when making predictions or inferences. The output in CRFs is computed using probabilistic inference techniques such as belief propagation or gradient-based optimization methods. The complexity inherent in the calculations for CRFs means they can be more computationally intensive than HMMs, potentially leading to longer processing times and greater resource utilization.

FluidMeter [59] is a system designed to track human daily fluid intake using inertial sensors in smartwatches. It employs a two-level classification approach based on CRF to recognize drinking moments and estimate the amount of fluid intake in each

episode. The high-level classifier separates drinking moments from other activities, while the low-level classifier analyzes sensor data to identify the sequence of micro-activities during drinking. The system considers factors such as arm posture and sip duration to estimate fluid intake. Nguyen et al. [109] used a Hidden Conditional Random Field (HCRF), a variant of the traditional CRF that incorporates hidden variables into the model, to monitor the movements of baseball players using multiple inertial measurement units.

Statistical Model Apart from probabilistic graphical models such as HMM and CRF, other statistical models like the Latent Dirichlet Algorithm and Gaussian Mixture Models have been used for HAR.

Latent Dirichlet Algorithm (LDA) is a generative probabilistic model typically used to classify text in a document to a particular topic. LDA assumes that each document is composed of various topics, and a distribution of words represents each topic. The primary objective of LDA is to learn these topic distributions based on the provided documents. The algorithm iteratively estimates the proportions of topic mixtures and the distributions of words that best explain the observed data.

Huynh et al. [70] proposed using topic modeling to detect complex activities. When LDA is applied to complex HAR using IMU sensors, a segment or window of sensor data represents a document. Each segment does not necessarily correspond to a specific activity. Instead, these segments capture patterns or behaviors found in the sensor data. The topics derived from LDA can be associated with various patterns, such as walking, running, or other activities. Similar to traditional LDA, these topics are characterized by distributions of sensor readings. In the context of activity recognition, the "documents" are the sequences of activities a person performs, and the "words" are the individual activities. The "topics" are the activity patterns that the authors aim to discover. By applying LDA to the activity data, the authors can identify which patterns of activity (topics) are present in each sequence of activities (document).

Another popular statistical model is the Gaussian Mixture Model (GMM), which represents complex probability distributions by combining multiple Gaussian distributions. It supposes that the observed data points are generated from a mixture of Gaussian components, each characterized by its mean and covariance. GMMs can capture

complex data structures with multiple modes and can be trained using the Expectation-Maximization (EM) algorithm. The study by Azadi et al. [13] demonstrates the application of GMM in HAR for alpine skiing activities. Using smartphone IMU and unsupervised learning techniques, the researchers successfully developed a system that can accurately detect skiing activities in various conditions. Shin et al. [139] focused on developing a locomotion mode recognition (LMR) algorithm using GMM and IMU sensors for classifying terrains during walking. The algorithm aimed to address walking difficulties elderly individuals face and prevent fall accidents. In the research by Wang et al. [159], an HMM and GMM were employed for automated recognition of human daily activities using sensor signals from a waist-worn accelerometer. The HMM was used to model training signals for each activity class, while the GMM captured the continuous observations for each hidden state. The classification of a new test signal was based on the HMM with the highest likelihood.

3.4.2 Deep Learning Approaches

Deep learning, a subfield of machine learning that focuses on training artificial neural networks with multiple layers, has gained significant traction in the field of HAR utilizing IMU sensors. These neural networks, comprising interconnected nodes or neurons organized into input, hidden, and output layers, excel at capturing complex patterns and structures from high-dimensional data. In HAR, various neural network architectures have been employed to detect and classify human activities based on IMU sensor data, effectively capturing human movements' temporal and spatial dependencies.

Convolutional Neural Networks (CNNs) have shown efficacy in processing grid-like data, such as images or spectrograms, and have been adapted for HAR by considering IMU sensor features or axes as spatial dimensions. By employing convolutional layers with filters or kernels, CNNs learn hierarchical representations of spatial patterns and variations, enabling the recognition of activity-related spatial characteristics. For example, Gupta et al. [57] utilized CNNs to assist individuals in learning correct yoga exercise execution.

Recurrent Neural Networks (RNNs) are specifically suited for sequential data analysis and are well-suited for processing sequential readings from IMU sensors. Tradi-

tional RNNs suffer from limitations in capturing long-term dependencies, but the introduction of Long Short-Term Memory (LSTM) networks [63] overcomes this issue by incorporating memory cells and gating mechanisms. LSTM networks effectively capture long-term dependencies in sequential IMU data. For instance, Chen et al. [29] employed a recurrent probabilistic neural network (RPNN) to monitor brushing technique correctness, leveraging probabilistic units integrated into the hidden states of the traditional RNN to model uncertainty and generate probabilistic predictions.

In the realm of IMU-based handwriting recognition, Li et al. [86] introduced an interactive approach utilizing smartwatch sensors to recognize handwritten words. The performance of RNN and LSTM was compared, with LSTM demonstrating superior performance. In sign language recognition, Hou et al. [66] leveraged LSTM to handle time series data and extract meaningful features. Bidirectional LSTM (BLSTM) was employed in the hidden layers to enable sentence-level recognition. BLSTM, a modified form of LSTM, is designed to capture temporal dependencies in sequences by processing inputs in both forward and reverse order. This bidirectional approach aids the model in learning contextual information and differentiating two-handed signs within sentences.

Deep learning is often combined with additional deep learning or machine learning techniques to achieve superior results across various applications. This fusion of techniques capitalizes on their respective capabilities to enhance performance and accuracy in the application of interest [8, 101, 148, 100].

For instance, Anand et al. proposed a generalized system for shot detection and classification in swing sports using Bidirectional Long Short-Term Memory (BLSTM) and Convolutional Neural Network (CNN) [8]. By combining the temporal modeling abilities of BLSTM and the spatial analysis capabilities of CNN, the system achieved accurate detection and classification of shots.

Similarly, Mekruksavanich et al. proposed a hybrid framework for HAR by combining LSTM and CNN [101]. By automatically extracting spatial-temporal features from the data, this approach negates the requirement for manual feature extraction. The combined model improved the accuracy of activity recognition in HAR tasks. In addition, Stankoski utilized CNN-based Deep Learning techniques to generate virtual streams from wrist-worn device data [148]. These virtual streams provided a source for

extracting relevant features, which were then integrated into a Hidden Markov Model (HMM) framework for detecting eating activities. Furthermore, Mekruksavanich et al. employed a hybrid CNN-LSTM model and smartwatch data for detecting complex activities such as cooking and office work [100]. The combined model effectively captured the temporal and spatial dependencies in the data, resulting in accurate recognition of these activities.

Integrating deep learning techniques with other algorithms holds great potential for enhancing the accuracy and performance of various applications, including activity recognition and motion analysis.

Ref.	Sensors	Actions	Device	Algorithm	Mobility	Domain
[5]	acc, gyr, mag	NA	wrist-worn sensor	DTW, FSS	Free Arm	Healthcare
[8]	acc, gyr	NA	smartwatch	DL, mRMR, CNN, BLSTM	Free Arm	NA
[16]	acc	NA	Body worn IMU	Decision Tree	Full Body	Fitness
[18]	gyr	Gait detection	NA	NA	Full Body	Healthcare
[20]	IMU	Driving Safety	smartwatch, smartphone	NA	Res. Arm	Transport
[23]	IMU	NA	NA	NA	Full Body	NA
[27]	imu	walking in different terrain	Insole	NA	Full Body	Fitness
[29]	grav. acc	NA	Toothbrush	RPNN	Res. Arm	Healthcare
[32]	acc	smoking	smartwatch	ML, ANN	Res. Arm	Fitness
[34]	acc, gyr, mag	HAR for children	smartphone, smartwatch	LightGBM	Full Body	NA
[35]	gyr	writing	smartphone	NA	Res. Arm	Interface
[38]	acc, gyr	NA	smartphone	NaiveBayes	Free Arm	Healthcare

[37]	acc, GPS	Road surface condition and defect	smartphone	k-means	Full Body, vehicular	Transport
[40]	acc, mag, gyr	Physiotherapy	smartwatch	DL	Full Body	NA
[58]	acc, gyr	Text input	finger ring	NA	Res. Arm	Interaction
[57]	acc, gyr	yoga	NA	DL	Full Body	Fitness
[59]	gyr, acc	Hydration tracking	smartwatch, Microsoft Band	CRF	Free Arm	Fitness
[66]	acc, gyr	ASL	smartwatch	DL, Hybrid	Free Arm	Interaction
[68]	acc, mag, grav.	driving safety	smartwatch, mag. wearable	NA	Limited Body	Transport
[78]	acc, gyr, mag	gesture aided e-learning	XSENS	Rule-based	Free Arm	Education
[93]	acc, gyr, geo-mag	On-site sport skill improvement	smartwatch	NA	Full Body	Sports
[86]	acc, gyr	Letter recognition, word recognition	smartwatch	RNN, LSTM	Res. Arm	Interface
[101]	acc, gyr	ADL	smartwatch	DL, Hybrid LSTM	Full Body	Healthcare

[100]	acc	Complex Human Activity Recognition	smartwatch	DL, CNNLSTM	Full Body	NA
[106]	acc, crowd	NA	smartphone	NA	vehicular	parking
[116]	acc, gyr, mag, quaternion	smoking	Invensense MPU9150	NA	Free Arm	Healthcare
[129]	Sound, Vision	transportation mode	NA	NA	Vehicular	Transport
[126]	acc, gyr, mag	Objects have unique hallmark	smartwatch	Nearest Neighbor	Full Body	NA
[136]	acc, gyr	Eating	smartwatch	J48 Decision Tree	Free Arm	Healthcare
[148]	acc, gyr	Detection of eating segments	smartwatch	DL, ML	Free Arm	Healthcare
[151]	acc	Eating	smartwatch	NA	Free Arm	Healthcare
[164]	acc, gyr, mag	Swiming motion and posture	NA	HMM	Full Body	Fitness
[168]	acc, gyr	finger gestures	smartwatch	NA	Res. Arm	Interface
[174]	imu	fall detection	NA	HMM	Full Body	Healthcare
[180]	acc	eating and drinking	Mobile Cardiac Monitor	NA	Free Arm	Healthcare

Table 3.1: Overview of Sensor-Based Activity Recognition Research and Applications.

3.5 Problem Specific Related Work

In this section, we will provide the literature reviews of the specific problems addressed in this thesis.

3.5.1 Handwriting Micro-Event Recognition

The use of smartwatches for detecting activity and gestures has gained momentum, primarily due to the availability of high-fidelity sensors. Different studies have used motion sensors to detect movements ranging from arm-related daily activities such as eating [136] and drinking [143] to finer finger gestures and taps. ViBand [81] used high-frequency bio-acoustic signals for detecting movements like flicks, claps, scratches, and taps, with the help of a custom kernel that increased the smartwatch's accelerometer sampling rate to 4 kHz. TapSkin [178] utilized inertial sensors and a microphone in an off-the-shelf smartwatch to create eleven distinct tap gestures for user interaction. In the domain of handwriting analysis using motion sensors, the existing works can be divided into two categories: identifying handwritten content and utilizing handwriting for user authentication. To detect handwritten content, Airwriting [7] employs a custom glove with motion sensors, while Gyropen [35] transforms a smartphone into a pen using its accelerometer and gyroscope. Wang et al. [161] utilized an accelerometer-based digital pen to recognize handwritten digits and gestures. Arduser et al. [10] detected large-size letters drawn on a whiteboard using sensor data from a smartwatch. SHOW [89] allows text input through writing on horizontal surfaces using the elbow as support, while Li et al. [86] use motion sensors on a smartwatch to write with the fingers. On the other hand, to use handwriting as an authentication method, Levy et al. [83] proposed a wrist-mounted sensor-based handwriting signature verification system that verifies a signature using data collected during actual signature writing. Another approach proposed by Griswold-Steiner et al. [54] did not restrict itself to a person's signature but relied on long-term user behavior and an onboard smartwatch accelerometer sensor.

In addition to recognizing handwritten content and user authentication, the potential of studying handwriting to gain insight into a person's mental state, such as mood [130] and cognitive load [15], has been explored in other research areas. However, these studies primarily rely on tablets or digitizer pens, which may not be available to everyone or

everywhere. In this work [144], we utilize a smartwatch, which is much more accessible, to detect micro-events in writing. We can track our daily writing by monitoring the micro-events, expanding the scope of smartwatch-based handwriting analysis beyond content detection and authentication. In the future, we can use the writing micro-event information to conduct studies similar to prior research [15, 130] on a person's mental well-being.

3.5.2 Indoor Parking Detection and Localization

Parking in urban environments presents a significant challenge, with the high density and rapid pace of city life intensifying the competition for available spaces. A 2017 study [1] indicates that the average American motorist spends around 17 hours annually searching for parking on streets, in lots, and garages. Traditional parking methods, which lack technological assistance, typically involve a time-consuming process of searching for vacant spaces, compounded by the difficulty in recalling the vehicle's location, especially in large, crowded lots.

Smart parking technologies emerge as a promising solution to these challenges. Systems integrating cameras or sensors within parking facilities can provide real-time occupancy data and even predict spot availability, benefiting both drivers and parking management. However, their high setup and maintenance costs are significant drawbacks. Crowdsourced solutions offer a more economical alternative, relying on user reports for spot identification, particularly in open-street parking. Yet, this approach is hampered by its reliance on manual reporting, which adds cognitive load to users. Furthermore, they are vulnerable to incorrect data input, arising either from unintentional mistakes or deliberate tampering, such as when individuals falsely report occupied spaces in sought-after areas to reduce competition for those spots.

To address these issues, our research utilizes inbuilt smartphone sensors and ambient WiFi APs, avoiding the hefty costs of installing new sensors and automating the reporting process, thereby mitigating the manual burden typical of crowdsourced methods. Innovative systems in this domain vary in focus and application. Lan et al. [80] developed a system for indoor parking, tracking a driver's path within a building. It

[1] https://www.usatoday.com/story/money/2017/07/12/parking-pain-causes-financial-and-personal-strain/467637001/

assumes that a driver nearing their parked vehicle will likely leave soon, a premise that isn't always accurate. PocketParker, introduced by Nandugudi et al. [106], is a smartphone-based crowdsourced outdoor parking system designed to predict parking lots' availability. Their method involved using existing activity detection techniques to determine user states and the transition between these states, aiding in the detection of parking and unparking events. The primary aim of their research was to create a model that predicts parking space availability, including assessing the impact of non-PocketParker users on these predictions. Nawaz et al. [107] proposed ParkSense, which utilizes a smartphone application linked to parking payment systems to detect parking events and capture the WiFi signature of the parked vehicle's location. This information is then used to determine when a driver returns to their vehicle, employing the Jaccard Index of visible WiFi Access Points for detecting unparking events.

Our research [146, 145] introduces significant advancements in smartphone-based parking detection systems, particularly tailored to indoor parking environments' complex demands. Unlike existing systems predominantly designed for outdoor settings, our approach effectively tackles common indoor issues such as poor GPS signal reception and the repetitive visibility of WiFi APs. Notably, our system is adept at identifying both parking and unparking events, a comprehensive capability not found in systems like Lan et al. [80], which focus only on detecting unparking events based on driver proximity assumptions. We achieve this through an innovative use of smartphone sensors, implementing a rule-based and thresholding approach that leverages accelerometer and WiFi AP data for real-time, unsupervised transportation mode detection. This method stands in stark contrast to the complex supervised classifiers used in studies like Wang et al. [159] and Reddy et al. [127], or the hierarchical systems developed by Hemminki et al. [62]. Our approach's independence from the smartphone's position and user interaction makes it exceptionally suited for the indoor parking scenario, where traditional methods falter. Furthermore, we have refined existing WiFi-based detection methods, such as those used in ParkSense by Nawaz et al. [107], by adapting the Jaccard Index for the unique setting of indoor parking. This tailored adaptation allows our system to more accurately identify parking events where WiFi APs are frequently encountered, enhancing reliability and effectiveness. Overall, our approach broadens the scope of smartphone-based parking systems to include indoor environments and introduces novel methodologies to address specific challenges inherent in these settings.

3.5.3 Liquid Intake Consumption Tracking

Within the scope of tracking liquid consumption monitoring and providing hydration reminders, conventional methods typically depend on manual data entry by users or the utilization of containers specifically designed for tracking intake. However, these methods have their limitations. To address these, we propose a novel, fully automated system for liquid intake tracking, adaptable to any container. This innovative system comprises two key components: a cost-effective, detachable base with an integrated load sensor, and a commercially available smartwatch equipped with an accelerometer. Our review of related work in this field reveals two main research streams: (a) studies using wrist-worn motion sensors to detect eating and drinking behaviors, and (b) research aimed at measuring the volume of liquid consumed. The following subsections will delve into each of these categories in detail.

Use of wrist-worn sensors for detection of eating and drinking activities

Wrist-worn acceleration sensors, either on one wrist [5, 38, 136] or both the wrists [180] have been used to capture hand movements to detect eating and drinking. Apart from [5], the focus of these works has been to detect eating activity. In [5], Amft et al. used a wrist-worn acceleration sensor to detect drinking motion. They combined the acceleration sensor with a magnetometer worn on the user's shoulder to distinguish between nine different container types with 75% accuracy and three different fluid levels with 72% accuracy. Fluid-level information is more valuable than just the drinking motion. In our work, we not only detect the dinking activity but also the amount of liquid consumed in each sip.

Drinking gestures has been studied only as a comparison class. In human activity recognition literature, most of the classification is done using multi-class classification, where multiple activity classes are defined. SVM, NN, Random Forest, J48 decision tree, Naive Bayes, Conditional Random Field [151, 136, 116, 38] are some of the popular multi-class classifiers in human activity recognition. They classify given data into one of the several pre-defined activities, such as walking, driving, and climbing stairs. However, there are times when a user is not performing any of these listed activities. Listing out and training for all the possible activities is challenging, close to

impossible. Given a data instance that does not belong to any of the listed activities, a multi-class classifier will classify it as one of these activities. A common approach is to set a threshold value on the class deciding numerical factor, and any data instance that does not satisfy this threshold is classified as $unknown$. In this paper, we use a one-class classifier. A one-class classifier is a classifier that aims to identify objects of a specific single class for which it is trained. A one-class classifier, therefore, allows us to concentrate on spotting drinking gestures and not be concerned about other activities.

Use of smart containers to detect the volume of consumed liquids

Chiu et al. [30] presented a Playful Bottle system targeted towards office workers, which uses accelerometers in the mobile phone to detect drinking events and photos taken from a mobile camera to estimate the volume consumed. The mobile is attached to a drinking mug using a stand made from LEGO bricks. Experiments show over 96% accuracy in volume detection. The system with an attached mobile is not practical. It also restricts users to use a certain type of drinking mug. Dong et al. [38] present a system in which an electronic band is attached to the bottle, which consists of an accelerometer, used to detect drinking motions and volume. The system achieves about 99% accuracy in detecting drinking events and about 75% for volume detection. For volume estimation, the authors use characteristics such as the angle of tilt and duration or speed of the tilt of the bottle.

HydraCoach[2] is a bottle that uses a flow sensor to detect the volume of liquid intake and costs 30$. As the rotor rotates with the liquid flow through the straw, the magnetic sensor detects the speed of the rotation, using which the volume can be detected. The flow sensor rotates when the liquid flows through the straw, and the magnetic sensor attached to the straw detects the rotation from which the consumed volume can be calculated. It might clog the rotor if used with any drinks other than water. Also, using hard water might affect the rotation because of calcium deposition on it. There are others like Trego[2], which uses ultrasonic sensors in the bottle's cap and costs around 50$. Using these sensors, firstly, restricts the type of bottles that can be used. Secondly, it is not universal or interchangeable for different kinds of bottles users use. The products available are manufactured with proprietary bottles and sold at high cost. Further, none

[2]http://www.hydracoach.com/
[2]http://trago.co/

of these systems can account for spilled liquid or drinking by another user.

3.6 Conclusion

Table 3.1 provides a comprehensive summary of various studies focused on sensor-based activity recognition and their applications. It categorizes the studies by the types of sensors used, specific actions detected, devices employed, algorithms applied, mobility context, and domain of application. Key highlights include the diversity of sensors capturing motion data and the range of detected actions, from generic activities like walking and smoking to specific actions such as gait detection and handwriting.

The table also indicates the variety of devices used, showcasing the diversity in hardware utilized for activity recognition. The review covers a range of algorithms, illustrating the computational techniques used to process sensor data. Human body movement contexts span from free arm and full body movements to restricted arm movements, highlighting different scenarios for activity recognition. The applications cover multiple domains, such as healthcare, fitness, and transport, emphasizing the broad impact and utility of sensor-based activity recognition.

In conclusion, the literature review chapter provides a thorough overview of advancements in HAR, with a specific focus on the use of motion sensors in smartphones and smartwatches. The review begins by discussing the wide range of HAR applications across various domains, highlighting the transformative potential of these technologies in enhancing everyday experiences and improving quality of life.

Furthermore, the chapter explores different categories of sensor deployment and their implications for activity recognition. It discusses the effectiveness of various sensor locations for detecting different activities. The review also examines various inference algorithms used in HAR, including both machine learning and deep learning techniques, analyzing their strengths and limitations. Lastly, the literature review presents relevant studies and research addressing the problems tackled by the thesis. Overall, this chapter provides a comprehensive understanding of current advancements in HAR, sensor deployment strategies, inference algorithms, and relevant studies in the field.

CHAPTER 4

Understanding Handwriting Micro-events Using Motion Sensor in Smartwatch

4.1 Introduction

Handwriting, or pen-paper writing, is a traditional and widely used method of capturing information despite the proliferation of computers and other digital technologies. Handwriting remains an essential skill in elementary teaching-learning for the following reasons. Handwriting engages the brain differently from typing on the keyboard [92] and helps children be more creative and improve their motor [97] and even reading skills [74]. This widely practiced skill among students holds a pool of information about the student's interactiveness, comprehension, and expression. The objective of this chapter is to develop a solution for detecting handwriting micro-events, such as "write," "shift," and "newline," using motion sensor data collected from commercially available smartwatches which have become ubiquitous at this day and age. Most earlier work on handwriting has focused on recognizing written content or using handwriting as an authentication technique. These include image-based methods for analyzing spatial patterns in handwritten text [49, 134], and content detection with specialized devices like smartpens [133, 166] or smartsurfaces [171, 125, 87]. This work differs from existing approaches in two key aspects. First, it leverages commercially available smartwatches, providing a more accessible and cost-effective alternative to specialized hardware, such as tablets or digital pens. Second, while some prior works have used smartwatches to analyze handwriting content, our approach shifts the focus to detecting and classifying handwriting micro-events rather than interpreting content. This is significant for applications where understanding handwriting behavior is more valuable than extracting content, such as behavior monitoring, handwriting skill evaluation, and fatigue detection.

This work presents an initial step for building a handwriting tracking system akin to health tracking applications that track heart rate, step counts, sleep quality, and nu-

tritional intake using a smartwatch. We introduce the idea of writing micro-events, defined as transient events that frequently occur during a writing episode. We identified three writing micro-events, *write*, *shift*, and *newline*. A *write* micro-event occurs when a writer makes a mark on the paper with a pen. A *shift* happens when the writer moves her hand from left to right to make space for writing. A *newline* is when the writer lifts her hand from the end of a line and brings it to the start of the following line. A handwriting episode consists of a sequence of writing micro-events. Furthermore, we consider variations of these three writing micro-events in varied writing scenarios a student may encounter in a teaching-learning environment. The scenarios include copying existing material, writing from memory, and writing their thoughts. Using the writing micro-events, this study attempts to answer the question – Can motion sensors in wrist-worn and commercially available smartwatches be used to detect writing micro-events?

While watches are conventionally worn on the non-dominant hand, this habit is adaptable when users perceive clear benefits from new technologies. For example, despite any initial discomfort, users are likely to adjust to wearing a smartwatch on either hand, particularly if it facilitates handwriting pattern detection. To explore this potential, we used a commercially available smartwatch for its practicality and accessibility in research. Future iterations could enhance naturalness by focusing on slimmer wrist-worn devices, such as fitness bands or compact smartwatches, which minimize interference with daily activities.

To this end, we aim to answer a set of questions, including determining the optimal data preparation method for supervised learning and identifying which writing micro-events are detectable through the approach. Furthermore, we compare the performance of human-engineered feature-based machine learning models to deep learning models that learn from raw time series sensor data. We also investigate the relative performance of user-specific and user-independent models and determine the amount of data needed to train the model. With the help of these analyses, we hope to contribute to developing an effective and accurate supervised learning model for writing micro-event detection. The ability to track writing micro-events will facilitate analysis and an understanding of students' behavior, collectively and individually. For example, tracking writing micro-events can help teachers assess the level of interactions in the classroom during lectures that typically involve taking notes. Likely, the total amount of time an attentive student spends on *write* micro-event during a lecture will be more than or equal to the class aver-

age. Throughout the academic session, we have an opportunity to examine the writing micro-events of the students. Analyzing this data from students in the same group, along with their quantitative assessment scores, can assist us in identifying trends, both positive and negative.

4.1.1 Challenges

It is difficult to manually tag the writing micro-events in real-time. We video-recorded the writing episodes and later did the tagging. We precisely tag the start and end time of writing micro-events. Although using a digital surface would increase tagging accuracy, it goes against our aim of preserving the natural writing pattern of the users on the paper. Additionally, it is impossible to tag the *shift* micro-event while writing on a digital surface.

There are three main challenges in detecting the writing micro-events. (i) The duration, the difference between the start and end times, of the writing micro-events vary significantly. Additionally, there is variability in the duration of the same micro-event between users. The variability in duration makes it challenging to decide on a fixed window size for detecting the micro-event and, consequently, how to assign a tag to the window while preparing the data for supervised learning. (ii) Due to the inherent nature of handwriting episodes, *write* is the most predominant micro-event leading to a class imbalance. (iii) Different motion sensors in the smartwatch capture the movement of the hand differently. Each sensor may be more suitable for capturing certain micro-events than others; therefore, selecting a single sensor for all micro-events is difficult.

4.1.2 Key Contributions

The main contributions of our work are as follows.

- To the best of our knowledge, our study is the first to investigate tracking of everyday writing activity using a wrist-mounted smartwatch containing motion sensors.
- We implement a prototype system using a consumer-grade smartwatch that detects the three writing micro-events: *write*, *shift*, and *newline*.
- We propose a *Tagging Threshold Assignment* approach that allows shorter-duration micro-events, *shift* and *newline*, to be tagged while maintaining a large enough

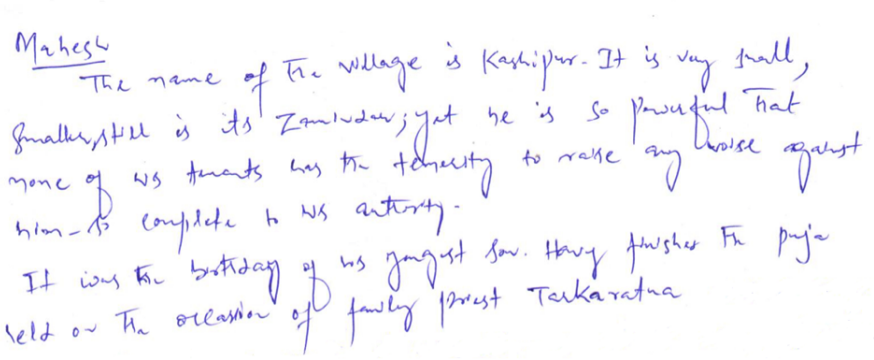


Figure 4.1: An image of handwriting sample from our data collection.

Figure 4.2: Pen Grips of different users

window size to detect the long-duration *write* micro-event. The most significant increase in F1 score is 0.26, achieved by lowering the tagging threshold value from 50% to 20%.

- We comprehensively assessed different configurations, including the type of motion sensor type, different window sizes, tagging threshold values, and classifiers based on human-engineered features and time series sensor data. We also evaluated user-independent and user-specific models, which reached the highest F1 scores of 0.79 and 0.84, respectively.

4.2 Data Collection

We use the LG W100 smartwatch to collect sensor data at 100Hz. We collected data from 10 users – 5 men and 5 women: The participants wrote while sitting on a chair. The writer used a non-metallic ballpoint pen and unruled A4-sized papers supported by a writing clipboard at the bottom. The data collection exercises were all recorded on video for ground-truth tagging. Although the data collection was conducted in the

lab, participants were allowed to move their hands freely to adjust the pen, paper, and writing board, mimicking natural movements to better reflect real-world scenarios. We consider three writing scenarios: *Copying*, *Memory*, and *Original*. In *Copying*, the subject copies word to word from a given material, such as a children's storybook. In *Memory*, the subject remembers a song or a poem of their choice and writes. In the *Original*, the subject creates new content while writing on a topic. The subjects wrote on the following topics: a) 'Describe your childhood.' b) 'Where do you see yourself after ten years?' c) 'Describe your experience in the institute.' We can assume that *Copying* has the most negligible cognitive load, whereas, in *Memory*, it could be from medium to high, depending on how well the person remembers. In the *Original*, the cognitive load would also range between medium and high depending on the subject's writing skills in the English language. By collecting our data using these three scenarios, we ensure that our data consists of writings that differ in both the writing content and the cognitive load on the subjects while covering most of the writing we do in our day-to-day lives.

To eliminate fatigue and capture intra-user variability, each of the ten users performed writing in all three scenarios in a day, spending approximately ten minutes on each activity. The users performed the same exercise on three different days, spending thirty minutes per day. So, we have around ninety minutes of data from each user and thirty minutes for each writing scenario. In total, fifteen hours of data, approximately, were collected from ten users.

We used ELAN (EUDICO Linguistic Annotator) [87] software to annotate the data collection video recordings. ELAN is a manual annotation tool that allows users to define time-aligned segments on a media timeline by selecting precise start and end points for each event. In our study, we labeled three specific writing micro-events using ELAN: *writing*, *shift*, and *newline*. These annotated segments capture the sequence and duration of each micro-event, providing the ground-truth data for our study.

We have thirteen and a half hours of tagged/labeled ground truth. The discrepancy between the total data collection time and the amount of annotated ground truth data is due to the presence of activities unrelated to the writing. Of the labeled ten hours, twenty-six minutes are *write*, followed by one hour forty-nine minutes of *shift*, and thirty-three minutes of *newline*.

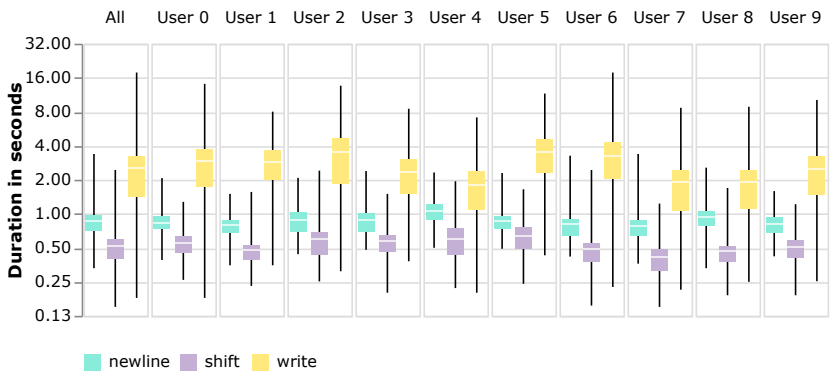


Figure 4.3: A box plot of ground truth duration of different writing micro-event, highlighting the challenge of determining an appropriate window size for supervised learning.

The duration of a writing micro-event is the time for which the writing micro-event lasts, that is, the difference between the end and the start of the micro-event. Figure 4.3 shows the box plot of the duration of the writing micro-events for each user and all users combined. The *write* micro-event has the most extended duration, followed by *newline* and *shift*. We see a significant variation across the duration of the different micro-events and also within the same micro-event. The difference in the micro-event duration creates a challenge in developing a supervised learning system, as we cannot create an optimum window size to determine the micro-event.

We use three motion sensors – an accelerometer, a gyroscope, and a rotation vector. The accelerometer and the gyroscope are physical sensors that measure acceleration and angular velocity, respectively. A rotation vector sensor is a type of sensor used to measure a device's orientation relative to a frame of reference. It is not a physical sensor as it does not have a specific and tangible component in the device. Instead, it is a virtual sensor that combines an accelerometer, magnetometer, and gyroscope output using a sensor fusion algorithm. This sensor produces a Quaternion, $q = w + xi + yj + zk$. It consists of two parts – an imaginary part, a three-dimensional vector specifying the rotation axis, and the real part, which specifies the angle by which to rotate about the axis by the imaginary part. In Figure 4.4, we see how the axes of the motion sensors are aligned when the user is writing, while Figure 4.5 shows a sample of raw data captured by the sensors during writing.

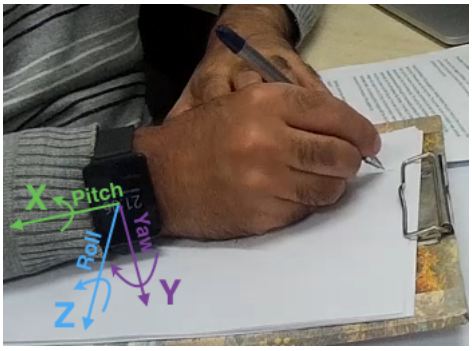


Figure 4.4: The spatial alignment of the 3D axes of the motion sensors on the smartwatch during writing.

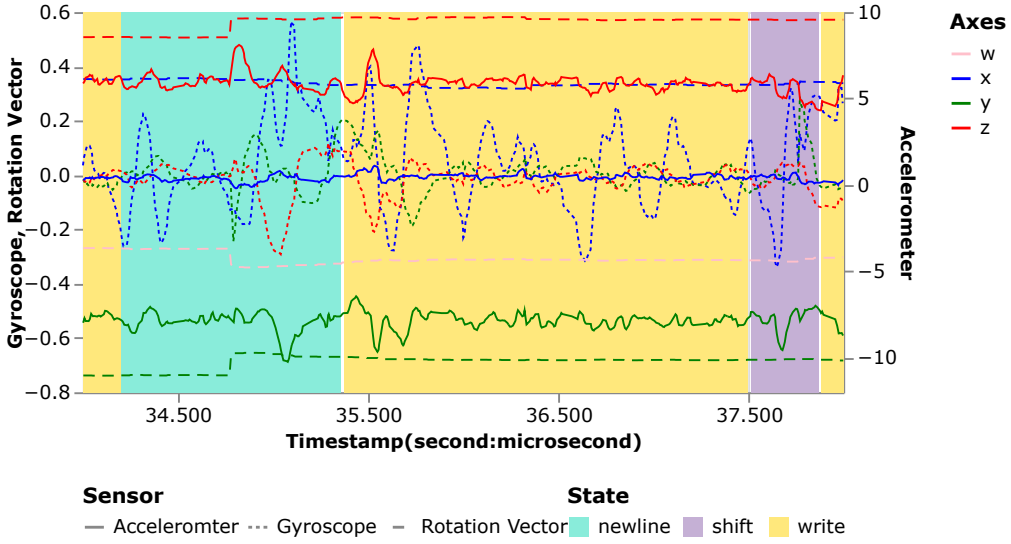


Figure 4.5: A sample 4-second raw sensor data showing how different sensors capture hand movement during writing. The accelerometer data is represented on the Y-axis on the right, while the gyroscope and rotation vector are shown on the left. The background color indicates the ground truth writing micro-event.

4.3 Proposed Solution

The objective of this section is to detect handwriting micro-events (*write*, *shift*, and *newline*) using motion sensor data from a smartwatch. The inputs include accelerometer, gyroscope, and rotation vector data, capturing acceleration, angular velocity, and orientation changes during handwriting. The system processes raw sensor data through four stages, as shown in Figure 4.6. First, we apply a frequency filter to remove noise from the raw sensor data. Next, we segment the time series data into fixed-sized windows. We then assign micro-event tags to each of these windows. Finally, we extract relevant features from the data and perform training and classification of the writing micro-events. Each of these steps is explained in detail in the following subsections.

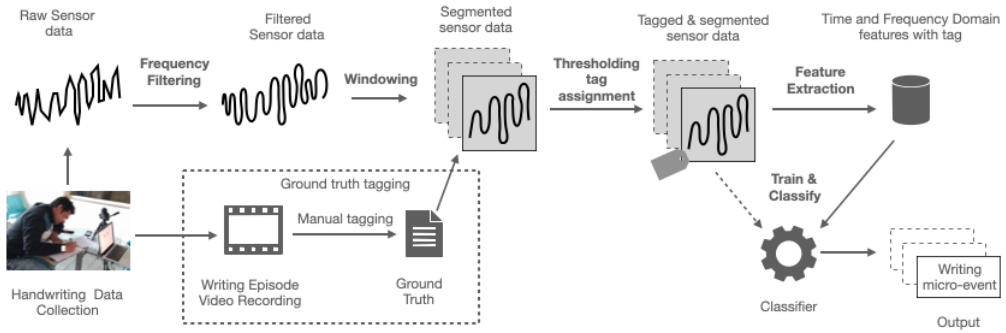


Figure 4.6: Our design of writing micro-event detection system.

4.3.1 Frequency Filtering

We collect motion sensor data at a sampling rate of 100Hz. However, the sampling rate is not constant as it fluctuates on the smartwatch. We fill in the gaps using cubic-spline interpolation to ensure that the sampling rate is 100 Hz for our FFT analysis. Figure 4.7 presents box plots of accelerometer data in terms of frequency. In these plots, the horizontal axis represents frequency, and the vertical axis indicates amplitude. A key finding from this analysis is that amplitude variations become stable after reaching 20Hz. This implies that frequencies higher than 20Hz mainly consist of noise, lacking significant information about the writing states we are interested in. This same pattern was observed in the data from the other two sensors, namely the gyroscope and rotation vector. To address this, we apply a low pass filter with a 20Hz cutoff during data processing. This step effectively removes high-frequency noise, ensuring our analysis is concentrated on the most relevant frequency range for understanding the activities related to writing states.

The plots provided in Figure 4.8 offer a detailed analysis of writing states in relation to frequency and time. The top plot presents frequency data, with the Y-axis indicating FFT frequency steps and color denoting amplitude. The bottom plot, representing time domain ground truth, divides a shared 200-sample window on the X-axis into 10 smaller segments of 20 samples each, shown on the Y-axis. Here, color highlights the predominant writing state in each segment. Visually, there is a correlation between the 'newline' state and frequencies in the 0-1.15 range. Additionally, the ground truth plot reveals that the 'writing' state often encases other states. This observation suggests that a conventional majority voting tagging approach might inaccurately categorize most data points as 'writing', overlooking the actual occurrence of other states within these

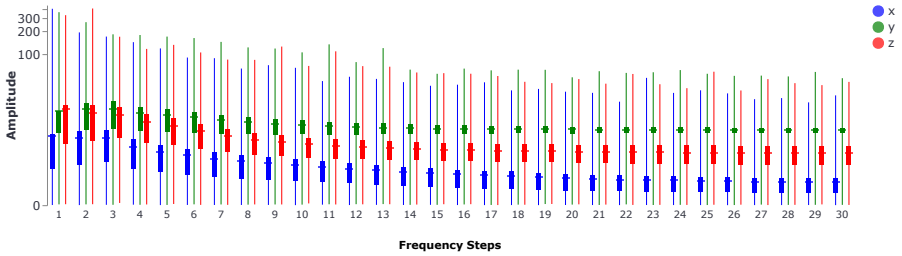


Figure 4.7: Box plots of accelerometer data highlighting frequency and amplitude: Identifying relevant frequency range for writing micro-event analysis.

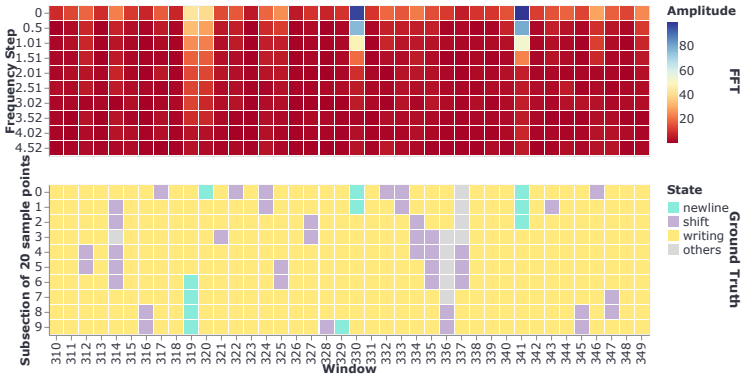


Figure 4.8: Correlating writing micro-states with frequency and time: Joint representation of frequency data and time domain ground truth.

windows.

4.3.2 Windowing

After applying the 20Hz low-pass filter to the sensor data, the next step is to segment the time series sensory data into fixed-sized windows, a commonly used technique in HAR.

4.3.3 Thresholding Tag Assignment

The windows of time-series sensor data are assigned a writing micro-event tag using ground-truth data to make the writing micro-event detection a supervised classification problem. Figure 4.9 shows a sample time series of ground truth. The top row represents a time series segment, while the following rows are 2-second windows on the segment with 50% overlap, namely windows 1 to 4. The bar colors correspond to ground-truth labels. A window can contain data points from different micro-events but can only be assigned one tag. Using a majority-voting approach, window 1 is assigned *newline* tag,

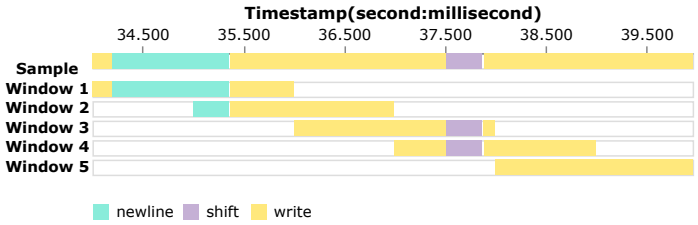


Figure 4.9: This figure illustrates the impact of windowing on group truth, which can potentially lead to shorter duration micro-events of *shift* occurrences to be overlooked as window tag.

and windows 2, 3, and 4 are assigned *write* tag. However, this method fails to capture smaller micro-events, such as *shift* in windows 3 and 4. The window size should be sufficiently small to allow the short-duration micro-events to emerge as the majority for tagging. Due to the difference in durations of micro-events, seen in Figure 4.3, using a smaller window size can result in insufficient data for detecting *write*. In conclusion, the majority-voting approach to assign tags is appropriate for detecting *shift* and *newline* when used with small windows. In contrast, large windows are better suited for detecting *write*.

To address the challenge of balancing the window size between *write* and the other two micro-events (*shift* and *newline*), we propose a *Thresholding Tag Assignment* approach. We use a threshold of less than or equal to 50% as the tagging threshold to determine a window's tag. The least frequent micro-event gets priority when more than one micro-event exceeds the threshold. The frequency of occurrence of the micro-event in decreasing order is `write`, `shift`, and `newline`. This annotation will be used throughout the rest of the chapter to represent the tag assigned via the thresholding tag assignment method. In our previous example in Figure 4.9, where each window is 200 data points, a 20% tagging threshold means the qualifying criteria is 40 or more data points. If 40 or more data points belong to *shift*, the windows get tagged as `shift`. If 40 or more points belong to *newline* in the same window, the tag will be a `newline`. Our approach works even for writing micro-events with shorter duration and lower occurrence frequency while maintaining a large enough window size to detect *write*.

The nature of writing exercises leads to a class imbalance among the three writing micro-events in the ground truth, with the majority of instances being the *write* class, followed by *shift* and *newline*. The tagging threshold's value affects assigned tags' distribution, where a lower value increases the likelihood of assigning a tag to less fre-

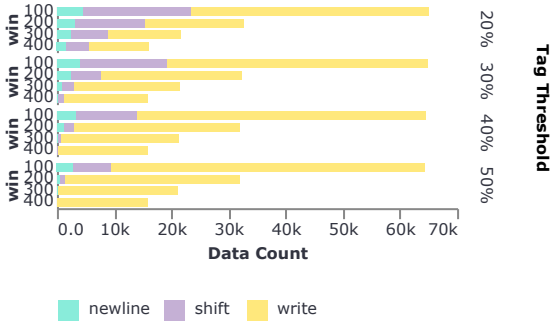


Figure 4.10: This figure depicts the relationship between tagging threshold and data count, demonstrating that decreasing tagging threshold values increase `shift` and `newline` data points and a decrease in writing data points, with window size kept constant.

quent micro-events. Figure 4.10 displays the stacked data count with tagging thresholds of 20%, 30%, 40%, and 50%. The Y-axis shows the different window sizes, while the X-axis represents the count of the micro-event tags, representing each micro-event tag with unique colors. As the tagging threshold value decreases, the count for `shift` and `newline` increases while that of `write` decreases. Additionally, the `shift` and `newline` have higher counts at the lower window sizes for the same threshold value. In subsection 4.4.1, we test and compare different tagging thresholds to determine the most suitable one.

4.3.4 Feature Extraction and Classification

We experimented with two types of classifiers based on the input data. One uses human-engineered features, while the other directly utilizes temporal sensor data. We do this to understand how simpler models, i.e., ones using human-engineered features, perform as compared to more complex models, which can process sensor signals directly without requiring domain-specific knowledge.

Human-Engineered Features-based Classifiers (HEFC)

We train these classifiers on human-engineered features commonly used in the field of HAR. For each data window, we extract the features mentioned in Table 4.1, categorized as statistical and frequency-based. We use either one or a combination of both to train our classifiers. We employ classifiers that have demonstrated effectiveness in the do-

main of HAR [136, 143, 89] namely, Decision Tree, Random Forest, SVM, Multilayer Perceptron, and XGBoost.

To handle the class imbalance problem seen in Figure 4.10, we use three popular approaches – Random Under Sampling, Synthetic Minority Oversampling Technique (SMOTE), and SMOTE-Tomek Link. Random Under Sampling randomly removes samples from the majority class to match the minority class. SMOTE generates synthetic data from the minority class using k-NN. SMOTE-Tomek Link combines SMOTE with removing data points from the majority class closest to the minority class using Tomek Links.

Temporal Sensor Data-based Classifiers (TSDC)

We feed the windowed sensor data directly to Long Short-Term Memory (LSTM) [111], a Deep Learning model, to learn temporal patterns from the time series data. We use ten LSTM models with random combinations of hyper-parameters. The hyperparameters tested were optimizer, number of LSTM layers, hidden layer size, epochs, learning rate, batch size, and input scaling. We included 32, 64, and 128 hidden layer sizes on a two-layer LSTM model. Epochs vary at 100, 150, and 200, the learning rate ranges from 0.01 to 0.00001, and batch size is selected from a logarithmic distribution between 16 and 64. Furthermore, we used min-max, standard, and robust scaling methods for input scaling. With these models, we assess whether LSTM captures any temporal patterns that human-engineered features might have overlooked. In TSDC models, we use class weight assignments to handle the class imbalance in these classifiers. It assigns higher weights to the minority class during training, thereby giving it more importance. The reasons why we do not use the same methods used in HEFC are as follows. Firstly, random under-sampling cannot be used for deep learning as it reduces the number of data points. Secondly, the remaining two approaches for managing class imbalance are intended explicitly for 2D data and are, therefore, unsuitable for 3D temporal sensor data utilized by TSDC. Converting the 3D sensor data into 2D sensor data to apply these techniques may lead to critical information loss, including the relationships between different axes.

Table 4.1: Table of features used in detecting writing micro-events through human-engineered feature-based supervised learning.

Type	Features
Statistical	mean, median, standard deviation, mean absolute difference, interquartile range, maximum, minimum, correlation, entropy, kurtosis, signal magnitude area, signal vector magnitude, zero crossings
Frequency-based	mean, median, standard deviation, mean absolute difference, interquartile range, maximum, minimum, average absolute difference, min-max difference, values above the mean, skewness, kurtosis, energy

4.4 Evaluation

We use the macro-average F1 score of 5-fold cross-validation as the performance metric. We evaluate four stages. First, we evaluate the performance of every possible combination of data set parameters to comprehend a general trend. Second, we determine the maximum F1 score to compare writing micro-events. Third, we compare the performance of user-independent and user-specific models. Finally, we examine the impact of data size on performance. Except for the user-specific performance, we evaluate performance utilizing user-independent data, i.e., the data comprising all ten users.

Number of Data Sets The data preparation process outlined in subsection 4.3 involves creating a data set by choosing a sensor, a window size, and a tagging threshold. These three choices, collectively known as the data set parameters, play an important role in the performance of our classification. Table 4.2 lists the data set parameter values we use to evaluate the performance. We form 48 data sets by combining three sensors, four window sizes, and four tagging thresholds. We derive the input to the classifiers from these data sets. For the three writing micro-events, we train classifiers to compare each micro-event with the other two micro-events resulting in three pairwise comparisons, `shift-newline` (`SN`), `shift-write` (`SW`), and `newline-write` (`NW`). We also train our classifiers to compare all three micro-events, i.e., `shift-newline-write` (`SNW`), with each other. In summary, there are four micro-event comparisons, referred to as the comparison modes, in the subsequent sections of the chapter. HEFC model includes five classifiers, three feature types, and three methods for handling class imbalance, resulting in a total of 45 models per data set, or 45 x 48 x 4 = 8640 HEFC models in total when considering all 48 data sets and four comparison modes. On the

Table 4.2: The list of evaluated values for each data set parameter.

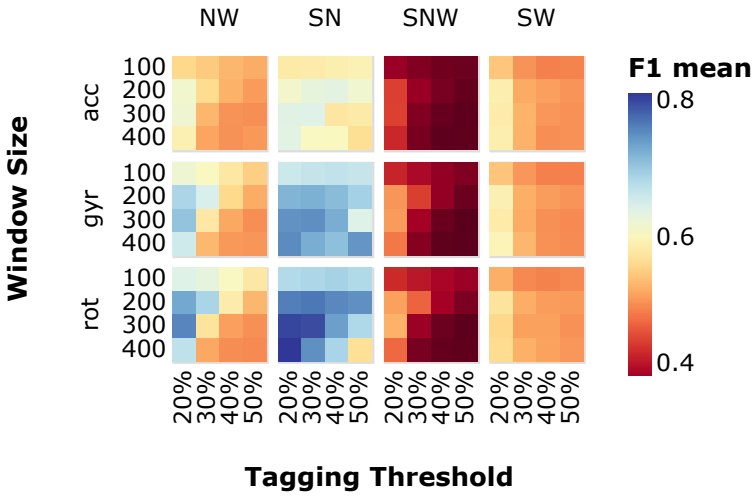
Notation	Description	Options
sen	Sensor	acc, gyr, rot
win	Window Size	100, 200, 300, 400
tt	Tagging Threshold	20%,30%,40%,50%

Table 4.3: List of abbreviations used in the evaluation.

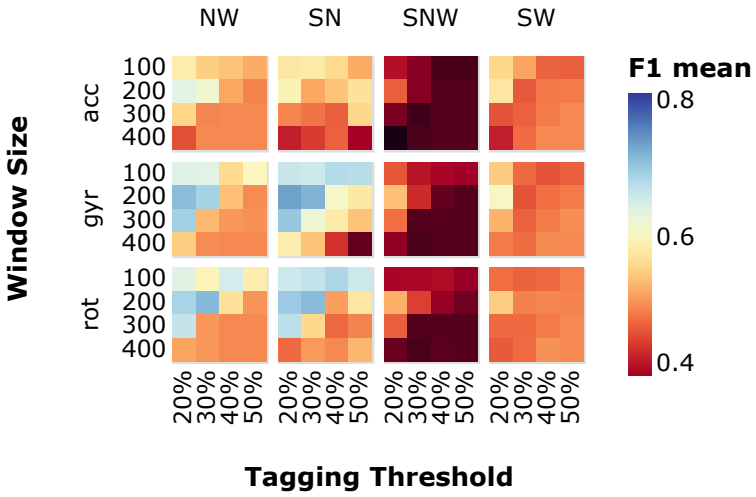
Type	Abbreviation	Description
Sensor (sen)	acc	Accelerometer
	gyr	Gyroscope
	rot	Rotation Vector
Micro-event Comparison Mode (mode)	SNW	shift vs newline vs write
	SN	shift vs newline
	SW	shift vs write
	NW	newline vs write
Feature Type (fea)	S	Statistical Features
	F	Frequency-based Features
	S&F	Time & Frequency Features
Imbalance class handling (imbh)	OVRS	SMOTE Over Sampling
	UNDS	Random Under Sampling
	HYB	Tomek-Smote Hybrid Sampling
Classifier Type	HEFC	Human-engineered Features Based
	TSDC	Temporal Sensor Data Based
Classifier (clf)	MLP	Multilayer Perceptron
	DT	Decision Tree
	XGB	XtremeBoost
	RF	Random Forest
	SVM	Support Vector Machine

other hand, the TSDC model directly uses sensor data and only one method of addressing the class imbalance, yielding ten models per data set, or 10 x 48 x 4 = 1920 TSDC models across all 48 data sets and 4 comparison modes. Table 4.3 summarizes all the abbreviations used for our evaluation.

In the preliminary phase of our study, we developed ten distinct models, each designed for a specific user, to ensure detailed and accurate handling of individual data sets. Subsequently, we introduced a user-independent model in the next step of our evaluation. This model was constructed based on insights gained from the configurations of the user-specific models. We chose this approach because running the full range of potential data set parameter combinations across a large dataset is resource-intensive. Initially, focusing on individual models allowed us to understand the impact of these parameters on a per-user basis before generalizing our findings.



(a) Using human-engineered features-based classifiers (HEFC).



(b) Using temporal-sensor data-based classifiers (TSDC).

Figure 4.11: The heat maps in the figure display the average F1 score for various data set parameter configurations, which reveals that using the rotation vector and gyroscope sensor with medium-size windows and low tagging threshold yields better performance across different comparison modes.

4.4.1 Influence of Data Set Parameters on Performance of the Classifiers

We assess the effect of changing the value of the data set parameter on the model's performance by computing the mean F1 score of all models. Figure 4.11 presents the mean F1 score for the dataset options listed in Table 4.2 as heat maps. The performance of HEFC models, with 45 models per cell, is depicted in Figure 4.11a, while Figure 4.11b represents the performance of TSDC models, with 10 models per cell.

The primary aim of Figures 4.11a and 4.11b is to provide a broad overview of the

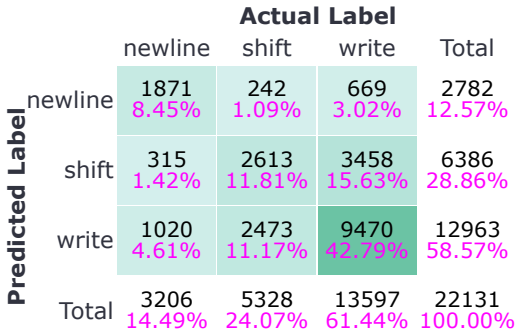


Figure 4.12: A confusion matrix of best performing `SNW` mode classification that highlights the problem in differentiating `shift` and `write` micro-events.

performance of TSDC and HEFC models across different sensors, window sizes, and tagging thresholds. HEFC employs 45 models per dataset, combining five classifiers, three feature types (statistical, frequency-based, and combined), and three methods for handling class imbalance ($5 \times 3 \times 3 = 45$). Each parameter combination corresponds to one "cell" in the heatmap. For TSDC, 10 models per cell were used, generated by varying deep learning hyperparameters such as optimizer type, number of LSTM layers, hidden layer size, epochs, learning rate, batch size, and input scaling. These diverse configurations ensure robust results and account for variability in deep learning training.

The rows represent sensors, and the columns represent comparison modes. The X and Y axes represent tagging thresholds and window sizes, respectively. The observations we make from these two figures are as follows. First, we notice that modes that involve comparison between `shift` and `write`, `SNW` and `SW`, show poor performance on both HEFC and TSDC models. Upon closer examination with the help of the confusion matrix displayed in Figure 4.12, we observe that the classifiers are not good at differentiating the `write` and `shift` micro-events. This poor performance is primarily due to the similarity in wrist movements between `writing` and `shift` micro-activity, as recorded by a wrist-worn sensor. The motion features extracted from the sensor are insufficiently distinctive, making it challenging for the classifiers to separate these two micro-events accurately. Moreover, the collected motion data may lack fine-grained details, such as subtle pressure changes or precise finger movements, that could help in distinguishing writing from shift. Second, a lower tagging threshold value, combined with a medium window size of 200 or 300, gives higher F1 scores in all comparison modes. It verifies the effectiveness of the thresholding tag assignment method,

which enables the comparison of longer-duration micro-events with shorter-duration ones. Moreover, we observe in Figures 4.11a and 4.11b that the HEFC and TSDC show similar levels of efficiency across different data sets. Nevertheless, a slight disparity is evident in the SN mode. SN consists of two micro-events, namely *shift* and *newline*, with shorter durations and limited data points. As the window size and tagging threshold values increase, these micro-events' data points decrease, as previously seen in Figure 4.10. The HEFC's machine learning models can learn with smaller data set sizes, resulting in relatively uniform F1 scores across various window and tagging thresholds, as depicted in Figure 8a. Conversely, the TSDC, which relies on deep learning, exhibits a higher F1 score with more data points at smaller window sizes and smaller tagging threshold values, as seen in Figure 4.11b.

After exploring various combinations of data set parameters, we observe that using a gyroscope or rotation sensor with a window size of 200 or 300 and a tagging threshold of 20% or 30% consistently results in the highest F1 scores across all comparison modes. Moreover, another observation we make from this evaluation is that TSDC and HEFC models exhibit similar levels of efficiency in detecting writing micro-events. These two insights are instrumental.

4.4.2 Optimising Performance for Writing Micro-events

We identify the best models for each writing micro-event comparison mode based on the results from the previous subsection. We select three models with the highest F1 scores for each micro-event comparison mode. The selected models further undergo randomized hyper-parameter tuning. The F1 scores of the tuned modes are shown in Tables 4.4 for HEFC models and Tables 4.5 for TSDC models. By pooling the results from both HEFC and TSDC modes, the best F1 scores are NW=0.79, SN=0.85, SW=0.65, and SNW=0.59. We note that all the models with the best F1 scores are HEFC models.

4.4.3 Comparing User-Independent and User-Specific Models

In this subsection, we use the insights gained from studying user-independent models in the previous two subsections to compare them to user-specific models. In order to make a fair comparison, we select a single data set for detecting all micro-events. This way,

mode	sen	win	tt	fea	imbh	clf	F1	P	R
	rot	300	20	S&F	OVRS	RF	0.79	0.78	0.80
NW	rot	300	20	S	HYB	RF	0.79	0.81	0.77
	rot	300	20	S	OVRS	RF	0.79	0.78	0.80
	gyr	400	50	F	OVRS	SVM	0.85	0.92	0.84
SN	gyr	400	50	F	HYB	SVM	0.85	0.92	0.84
	rot	400	20	S	HYB	RF	0.85	0.88	0.83
	acc	400	20	S	OVRS	RF	0.65	0.65	0.64
SW	gyr	400	20	S&F	OVRS	RF	0.64	0.65	0.64
	acc	400	20	S&F	OVRS	RF	0.64	0.64	0.63
	rot	300	20	S	OVRS	RF	0.59	0.59	0.60
SNW	rot	300	20	S&F	OVRS	RF	0.59	0.58	0.60
	rot	300	20	S	OVRS	SVM	0.58	0.57	0.61

Table 4.4: Models with the best performance for each comparison mode using human-engineered feature-based classifiers (HEFC).

mode	sen	win	tt	F1	P	R
	rot	300	20	0.78	0.82	0.75
NW	rot	300	20	0.76	0.82	0.73
	rot	200	20	0.76	0.83	0.72
	rot	300	20	0.83	0.88	0.81
SN	rot	300	20	0.81	0.88	0.79
	rot	200	20	0.81	0.87	0.79
	gyr	200	20	0.63	0.65	0.63
SW	gyr	200	20	0.62	0.64	0.62
	gyr	200	20	0.61	0.64	0.62
	rot	200	20	0.56	0.63	0.54
SNW	rot	300	20	0.56	0.63	0.54
	rot	200	20	0.55	0.60	0.54

Table 4.5: Models with the best performance for each comparison mode using time-series sensor data-based classifiers (TSDC).

we also avoid the need to process different sensors and possibly synchronize different window sizes. We have identified a single data set that delivers good performance across all modes by utilizing the insights gained from the previous sections. The chosen data set consists of the rotation vector sensor data with a window size of 300 and a tagging threshold of 20%. As TSDCs are deep learning models that rely on large data sets, we exclude them from consideration as the user-specific data sets are typically small. Based on Table 4.4, we select the Random Forest as the classifier with statistical and frequency features and SMOTE oversampling for handling class imbalance. We perform randomized hyperparameter tuning on the Random Forest classifier using the selected data set parameters for both user-specific and user-independent data. For the user-independent models, the NW and SNW modes do not require hyperparameter tuning

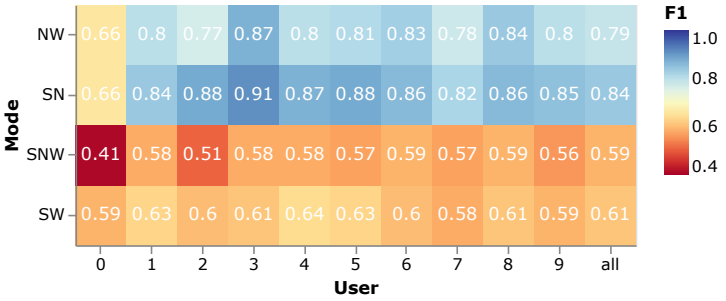


Figure 4.13: A comparison of the F1 scores of user-specific and user-independent models, revealing that user-specific models generally outperform user-independent models.

since we have optimized them in the previous subsection, as seen in Table 4.4.

The F1 scores for both user-specific models, with users ranging from 0 to 9, and user-independent models labeled as “all" are presented in Figure 4.13. We observe that the user-specific models perform better than user-independent models. Despite the inherent disadvantage of reduced training data, user-specific models can still offer improved performance over user-independent models. A reason for this is their ability to exploit user idiosyncrasies. On the contrary, user-independent models must overcome user-specific idiosyncrasies to learn the writing micro-events that can be generalized across users. In contrast, user-specific models can leverage them to their advantage. Even though the user-specific model results in better performance, collecting data from individual users may not always be feasible. In such cases, a user-independent can be used. In summary, the choice of approach depends on the specific situation and data availability.

4.4.4 Effect of Data Set’s Size on Performance

User-independent models are the preferred approach when user-specific training is not viable. In this subsection, we investigate the minimum amount of data required to train writing micro-event detection models that are user-independent. We evaluate this using the same user-independent data set in the previous subsection. Figure 4.14 shows the relationship between the data set size and the F1 scores of the models for the four comparison modes. For `SW` and `SNW` modes, the F1 scores remain low despite increasing data points. On the other hand, the `SN` and `NW` micro-events comparison modes achieve F1 scores close to the optimal values of 0.84 for `SN` and 0.76 for `NW`, with as little as

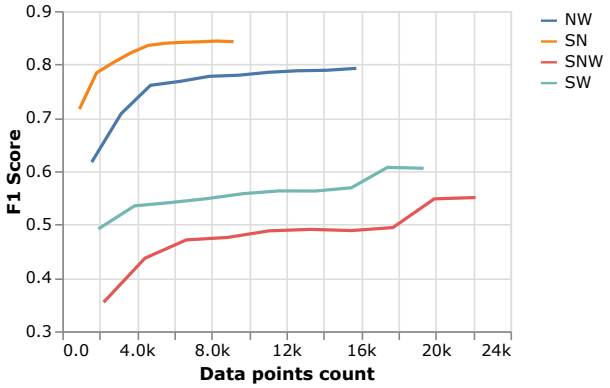


Figure 4.14: The correlation between the size of the user-independent training data and the F1 score for four micro-event comparison models. It highlights that `SN` and `NW` can be trained with fewer data, while `SW` and `SNW` struggle to achieve higher F1 scores even with larger data sets.

4.2k data points, which amounts to less than two hours of data.

4.5 Conclusion

In this chapter, we introduce the idea of tracking handwriting activity using writing micro-events and demonstrate a prototype classification system in the smartwatch using motion sensors. Our findings indicate that the proposed Thresholding Tag Assignment approach allows us to distinguish *shift-newline* and *write-newline* but struggled while distinguishing *write-shift*. We observe that human-engineered feature-based machine learning models perform better than time series sensor data-based deep learning models for our data. The results reveal that a Random Forest classifier trained on a rotation vector sensor data with a medium-sized window and a low tagging threshold works optimally in various comparison modes. Furthermore, user-specific models perform better than user-independent models. For instances where collecting and training user-specific models may not be viable, a user-independent model that can be trained using relatively small training data of approximately two hours is a reasonable alternative. Despite the limitations in differentiating between *write* and *shift* micro-events, our study's findings contribute to developing an effective and accurate supervised learning model for detecting writing micro-events.

CHAPTER 5

Detecting and Localizing Parking and Un-parking Event in Indoor Parking Garages

5.1 Introduction

Figure 5.1: Representative picture of an indoor parking facility showcasing alphanumeric markings on pillars, serving as parking spot identifiers.

Shopping malls in India, serving as major shopping and entertainment hubs, attract approximately 50 million visitors each month [1]. Many of these malls feature expansive underground parking facilities. However, finding free parking spots in these areas can be challenging for drivers due to limited visibility and the vast size of the lots. Additionally, the adoption of advanced parking systems in India has been slow, hindered by practical concerns such as high costs, maintenance challenges, and the potential for mischievous users tampering with sensors. In such parking facilities, marked pillars are often the only aid available for retrieving parked cars, as shown in Figure 5.1. Many times, users fail to observe the numbers on the pillar next to their parking space, causing difficulty when they attempt to locate their car afterward. In this work, we propose *BluePark*, a smart parking system for indoor parking facilities that can detect when and where a car is parked and un-parked using smartphone sensing in real-time, without any user input. While parking location information helps users retrace their vehicles,

[1] http://outdoormediaplan.com/malls-media-kit.php

un-parking event information helps other users find empty parking spots. Furthermore, the parking administration can use the parking and un-parking information to monitor the occupancy state of the indoor parking space.

Real-time parking systems today are mainly divided into two categories: infrastructure-based and crowdsource-based. Infrastructure-based systems rely on additional hardware like occupancy sensors and wireless transceivers, which are installed either in parking spots [149, 82] or on vehicles [99]. The significant costs for procurement, installation, and maintenance of these systems have hindered their widespread adoption. Crowdsource-based systems, conversely, gather information from users via web applications and smartphones [105, 118]. These systems tend to be more adaptable and can significantly reduce the need for extensive infrastructure development. However, their effectiveness largely depends on consistent user participation and the regular input of data. There are two types of crowdsource-based methods: manual and automatic reporting. In manual reporting systems [52], users manually input data about empty parking spots through a mobile or web application. This information is then sent to a central server, where other drivers can access it to locate recently freed parking spots. Despite incentives [105], the requirement for manual input remains a major drawback of this method. Moreover, there's a risk of users either withholding information or deliberately providing false data to keep others away from desirable parking areas [77]. Automatic reporting in crowdsource-based systems eliminates the need for incentives. Here, smartphone sensors automatically detect and report the occupancy status to a central system, allowing drivers to find parking spaces without the need to report their status manually. This approach removes the burden of manual reporting and enhances user convenience. Our research falls within the realm of automated crowdsourced strategies based on smartphone sensors, where we leverage the inherent capabilities of these devices for efficient detection and reporting of parking-related events.

Building on the concept of automated crowdsourced strategies using smartphone sensors, our work specifically focuses on the detection of parking and un-parking events as changes in a user's locomotive state [106]. These events are essentially transitions between different states of movement: parking events represent a shift from a *drive* state to a *walk* state, while un-parking events indicate the reverse. By accurately and promptly detecting these transitions, our system effectively captures the location of the parking or un-parking. The precision of localizing these events is heavily dependent

on the timely detection of the transition; any significant delay can lead to incorrect identification of the event's location. For outdoor parking scenarios, GPS technology is typically employed to pinpoint the vehicle's location and to determine if the user is in a *drive* state, as evidenced by the rate of GPS location change. However, the challenge intensifies in indoor parking environments where GPS signals are weak or non-existent. In such situations, an alternative approach is required not only for the accurate localization of the car but also for effectively detecting the *drive* state. This is where our research makes a crucial intervention, proposing a novel method tailored to the unique conditions of indoor parking scenarios.

We propose a novel approach of combining accelerometer and WiFi AP data to effectively distinguish between *walk* and *drive* states within indoor parking environments. We have developed a streamlined algorithm to rapidly identify state transitions, correlating these changes to parking and un-parking events. This algorithm leverages the user's state history, ensuring high precision and minimal latency in event detection. In addition to recognizing state changes, our system utilizes WiFi signals at each event to accurately locate where vehicles are parked. To conserve power, our system avoids continuous monitoring of parking. Instead, we employ Google's Geo-fencing API to activate the event detection module only when the user enters a pre-defined indoor parking zone. This approach is not only energy-efficient but it also effectively reduces false positives by ignoring state transitions similar to parking events that occur outside the designated indoor parking area. The algorithms for detecting states and their transitions operate exclusively on the smartphone, ensuring a streamlined and efficient process. The system communicates with an external server for localization purposes, but this involves only a one-time request, minimizing power consumption. The entire BluePark system is designed to function automatically, eliminating the need for user intervention. We implemented BluePark in the indoor parking area of a well-known shopping mall in New Delhi. This deployment took advantage of the already existing WiFi APs within the facility, meaning no additional infrastructure was needed for the system to operate effectively.

The principal contributions of this study are as follows:

1. Development of an algorithm tailored to accurately detect $drive$ and $walk$ states in indoor parking areas. Specifically:
   (a) A rule-based integration of WiFi and accelerometer data enhances the differentiation between $drive$ and $walk$ states. Our $drive$ detection methodology

outperforms the Google Activity Recognition API by a margin exceeding 20% in accuracy.

(b) Introduction of *Adaptive DBSCAN*, an evolution of the conventional DBSCAN algorithm, which facilitates unsupervised, real-time state detection employing accelerometer data.

(c) A novel WiFi-based metric adept at distinguishing between mobile and stationary states in indoor contexts.

2. Leveraging our state-detection algorithm, a mechanism for detecting and localizing parking and un-parking events. This mechanism boasts:
   (a) 100% precision and recall at a sampling frequency of 15Hz.
   (b) Low detection latency of 20 seconds with a probability of 0.9.
   (c) Good parking localization accuracy of 10 meters.

The sections that follow are structured in the following manner: Section 5.2 provides a detailed problem description, further breaking it down into specific components. In Section 5.3, we outline the system architecture of BluePark and elaborate on the adopted methodologies. Section 5.6 presents a comparative analysis, highlighting the performance of BluePark relative to other approaches. Lastly, Section 5.7 offers concluding remarks on the topic.

5.2 Problem Statement

Locating a parking spot and retracing a parked vehicle in bustling indoor parking facilities present a substantial challenge, exacerbated by the limitations of traditional GPS-based systems in such settings. The core issue lies in accurately detecting parking and un-parking events and subsequently localizing these events within the complex and often GPS-obscured indoor environment.

Our primary goal is to develop a system that overcomes these obstacles using smartphone sensors to autonomously and accurately detect parking and un-parking events in real-time, without requiring user intervention. This task involves three crucial components: state detection, event detection, and event localization, each presenting unique challenges in an indoor parking context.

1. **State Detection:** Traditional locomotive state detection solutions, such as the Google Activity Recognition API, fail to deliver accurate results in indoor parking scenarios, particularly when vehicles move at low speeds or perform subtle maneuvers typical of such environments. Our system aims to distinguish the *walk* and *drive* states more effectively, leveraging accelerometer and WiFi data.

2. **Event Detection:** Following state recognition, the next challenge is accurately identifying the exact moments of parking and un-parking, discerning them from similar but irrelevant activities. The efficiency of this detection impacts the system's overall reliability, necessitating a method that integrates both real-time and historical data to minimize false positives.

3. **Event Localization:** Upon successful event detection, the system must precisely localize these events. Indoor parking scenarios demand alternative methods like WiFi-based localization, given the ineffectiveness of GPS signals. Our approach aims to counteract the inherent challenges of WiFi-based methods, such as signal fluctuation due to environmental factors, ensuring more accurate and reliable localization.

In addressing these challenges, our system, named BluePark, aspires to redefine the experience of finding and retracing vehicles in indoor parking lots, presenting an efficient, user-friendly solution adaptable to the intricacies of modern urban parking infrastructures.

5.3 Approach

This section details the framework and methodology of our BluePark parking and un-parking detection system. We begin by discussing the System Architecture in subsection 5.3.1, which establishes the structure for the subsequent modules. Subsection 5.3.2 details the techniques used to identify different user states essential for detecting parking-related activities. In the subsection 5.3.3, we elaborate on the methods employed to recognize specific parking and un-parking events based on the identified user states. Finally, in the subsection 5.3.4, we outline the strategies used to determine the location of these events.

5.3.1 System Architecture

Figure 5.2 illustrates the BluePark system architecture, which consists of two principal components: the mobile client and the cloud service. The mobile client developed as an Android application, performs state detection using a hybrid accelerometer and WiFi-based sensing approach. This hybrid method enhances the accuracy of identifying the user's state in indoor parking environments. Additionally, the mobile client incorporates an innovative change detection algorithm in its event detection module. This algorithm

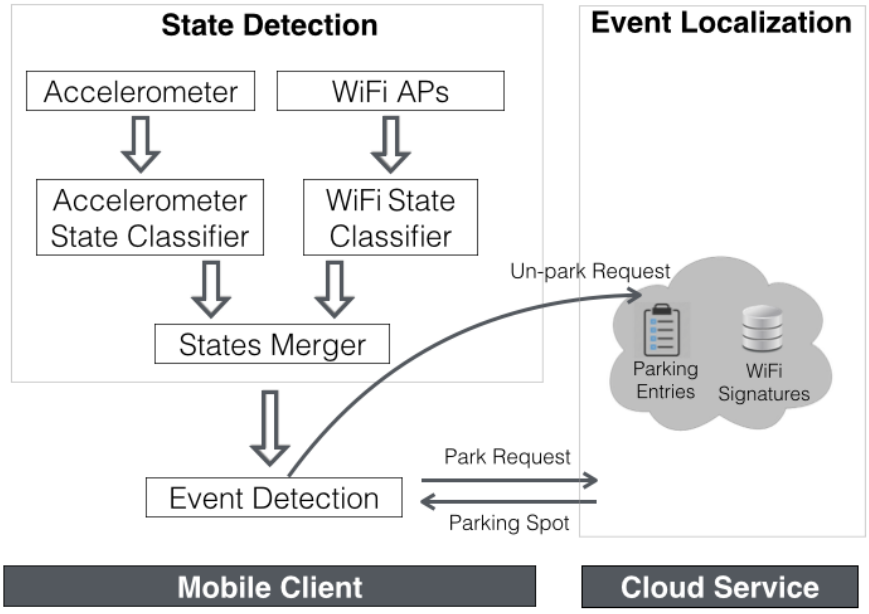


Figure 5.2: System Architecture of BluePark

efficiently identifies transitions indicative of parking and un-parking events, thereby minimizing latency.

Upon identifying a state transition, the mobile client captures the corresponding WiFi signature and relays it to the cloud service for processing. The cloud service, in turn, fulfills dual roles. Firstly, it executes a localization algorithm on the received WiFi signature to ascertain the most probable parking or un-parking location. Secondly, it manages parking occupancy status by maintaining a database of WiFi fingerprints corresponding to various parking spots. The mobile client communicates parking and un-parking events to the cloud service, which updates a Parking Entries log. This keeps track of the parking spots' occupancy status.

The following segments delve into a comprehensive examination of our state detection, event detection, and event localization methods.

5.3.2 State Detection

Our system's effectiveness in indoor parking areas relies on accurately identifying the locomotive states of the user, including differentiating low-speed driving from loitering behavior. We use a dual-sensor approach that combines accelerometer and WiFi data to achieve this. This hybrid method efficiently leverages both sensors, with WiFi sig-

nals adding minimal computational load, a significant consideration given the system's reliance on WiFi for event localization.

The State Detection module comprises three components: the Accelerometer State Classifier, the WiFi State Classifier, and the State Merger. The Accelerometer State Classifier utilizes accelerometer data to differentiate the *walk* state from other states, while the WiFi State Classifier employs WiFi signal analysis to identify the *drive* state. The State Merger consolidates the outputs from both classifiers, blending them with the historical user state data. This integrated result provides the current merged state of the user, which is then interpreted as the user's locomotive state.

Notation and Defintions

For clarity and ease of understanding, we have defined the following time series to represent the user's states:

1. $A(1), A(2), \ldots, A(t)$ is a time series resulting from the Accelerometer State Classifier, where $A(i)$ is the accelerometer based user state at i^{th} interval. $A(i)$ has three possible classes: a_{walk}, $a_{\neg walk}$ and $a_{unknown}$.
2. $W(1), W(2), \ldots, W(t)$ is a time series resulting from the WiFi State Classifier, where $W(i)$ is the WiFi-based user state at i^{th} interval. $W(i)$ has two possible classes: w_{moving} and $w_{\neg moving}$.
3. $C(1), C(2), \ldots C(t)$ is a time series resulting from the State Merger, where $C(i)$ is the state of the user at time t derived by combining $A(i)$ and $W(i)$. $C(i)$ has four possible classes: *walk*, *still*, *drive*, and *unknown*.

Accelerometer State Classifier

The Accelerometer State Classifier utilizes a real-time unsupervised learning methodology to determine whether a user is walking based on accelerometer data from the smartphone. We introduce a modified variant of the DBSCAN [43] algorithm called *Adaptive DBSCAN*.

In selecting a algorithm for walking detection, several criteria were considered: real-time processing capability, computational efficiency, robustness against noise, flexibility in handling cluster shapes, and no requirement to pre-specify the number of clusters. Existing methods such as K-means, Mean Shift, Affinity Propagation, Spectral Clustering, and Gaussian Mixture Models (GMM) present various limitations. K-means

Algorithm 2 AdaptiveDBScan

INPUT: Accelerometer feature values, $V_i = \{v_1, v_2, ...v_n\}$
INITIALIZE: $NeighborPts \leftarrow \{\}$, list of potential cluster points
procedure ADAPTIVEDBSCAN(v_i)
  if $NeighborPts$ is **empty then**
    append v_i to $NeighborPts$;
    $\alpha \leftarrow v_i$
  else
    if REGIONQUERY(v_i, α) **then**
      append v_i to $NeighborPts$;
      $\alpha \leftarrow mean$(NeighborPts);
      if $sizeof(NeighborPts) \geq MinPts$ **then**
        $A(i) \leftarrow getState(\alpha)$;
        declare $A(i)$;
    else
      if $sizeof(NeighborPts) < MinPts$ **then**
        $A(i) \leftarrow unknown$;
        declare $A(i)$;
      empty $NeighborPts$;
      append v_i to $NeighborPts$;
      $\alpha \leftarrow v_i$;
procedure REGIONQUERY(v_j,α)
  if $v_j > \tau$ **then** $\epsilon \leftarrow \theta * \alpha$;
  else
    $\epsilon \leftarrow \epsilon_0$;
  if $|v_j - \alpha| < \epsilon$ **then return** true;
  else
    return false;

requires the entire dataset for processing and assumes spherical clusters, making it unsuitable for dynamic and noisy event detection scenarios. Mean Shift and Affinity Propagation, while effective in some contexts, are computationally intensive due to their iterative nature, rendering them impractical for real-time applications on resource-constrained devices like smartphones. Spectral Clustering's high computational complexity and GMM's iterative expectation-maximization steps further limit their applicability in real-time settings. Additionally, K-means and GMM necessitate a predefined number of clusters, which is impractical for environments where the number of events is unknown a priori.

Standard DBSCAN offers several advantages: it can identify clusters of arbitrary shapes, handle noise by classifying low-density points as outliers, and does not require pre-specifying the number of clusters. However, despite these strengths, standard DBSCAN has limitations when applied to real-time event detection on smartphones,

particularly due to its static ϵ parameter. To overcome these limitations, we propose *Adaptive DBSCAN*.

The Adaptive DBSCAN algorithm, given in Algorithm 2, performs real-time clustering of accelerometer data points and dynamically adjusts the clustering threshold ϵ based on the density of incoming data. The input sequence for Adaptive DBSCAN consists of feature values $V_i = \{v_1, v_2, ...v_n\}$, computed from non-overlapping time windows of accelerometer data. Each v_i corresponds to a numerical feature that quantifies the variation in energy distribution across different frequency components within a single time window.

The FFT is applied to the accelerometer signal to extract frequency-domain characteristics, which are crucial for identifying periodic patterns associated with locomotive states such as walking. These patterns occur at characteristic frequencies, and FFT helps capture their energy distributions effectively. The variance of FFT energy provides a robust measure of how the energy is distributed across frequencies during the time window. Taking the average variance smooths out short-term noise and highlights persistent frequency-domain patterns, which are critical for accurate clustering and distinguishing between movement states.

This dynamic adjustment of the clustering threshold ϵ allows the algorithm to intuitively discriminate between different locomotive states, such as walking and slow driving. Walking generates periodic accelerometer signals with relatively higher frequencies due to regular steps, producing feature values v_i with greater variance. In contrast, slow driving exhibits lower frequency components or near-flat patterns because of reduced vibrations and movement variations, resulting in feature values v_i that are less variable and form denser clusters. The algorithm leverages these distinct cluster dynamics to differentiate between slow driving and walking effectively.

The algorithm utilizes a dynamic list, $NeighborPts$, to track potential cluster points alongside an adaptive threshold ϵ. The function `REGIONQUERY` assesses whether an incoming data point qualifies to be part of the current cluster by verifying if the data point is within the distance ϵ from the cluster's centroid α. A data point is part of the cluster if its distance from α is less than ϵ. If the count of points in $NeighborPts$ falls below a minimum limit MinPts, the algorithm labels the current state as $c_{unknown}$. Otherwise, the *getState* function returns a_{walk} or $a_{\neg walk}$ output using a threshold on

the value of α. The value of α is intrinsically linked to the two parameters: θ and $MinPts$. Tweaking θ influences the span of the ϵ-neighborhood when $v_j > \tau$. A larger θ means a more inclusive boundary, while a smaller θ tightens it. Concurrently, $MinPts$ designates the minimal data points for a valid cluster, with a higher $MinPts$ value categorizing an increased subset of data as outliers.

Unlike traditional DBSCAN, which uses a static ϵ, *Adaptive DBSCAN* recalibrates the value of ϵ based on incoming data. When a new data point v_j exceeds a preset threshold τ, the algorithm modifies ϵ by multiplying it with α and a fraction θ, which lies between 0 and 1. For points less than or equal to τ, a default ϵ_0 is applied. This enables *Adaptive DBSCAN* to work on real-time incoming data and helps to accommodate large density differences among different user locomotive states.

WiFi State Classifier

The WiFi State Classifier determines whether a user has moved or not moved solely based on the observable WiFi access points. The *moving* state encompasses both walking and driving activities. This approach, while not novel in tracking movement, faces unique challenges in the indoor parking context compared to outdoor environments.

Previous studies, such as ParkSense [107], have employed the Jaccard Index, $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$, to gauge motion in outdoor settings. The Jaccard Index quantifies the similarity between two sets of WiFi APs, focusing on set membership without considering the frequency of AP appearances. In outdoor scenarios, the assumption that a WiFi signal from an AP disappears when moving away from it is generally valid due to open spaces and fewer obstructions.

However, indoor parking facilities present a different scenario. These areas have multiple reflective surfaces, like walls, ceilings, and pillars, which can cause a WiFi signal from a distant AP to reach the receiver through multi-path effects. As a result, relying solely on a set membership of visible APs for location change estimation can be misleading. Additionally, in indoor parking, where driving speeds are slower, and driving lanes are restricted, the same AP may be visible multiple times within the same scanning window. These factors necessitate a refined approach for accurately determining movement rate in such environments, considering both the presence and frequency

of APs in successive scans.

In indoor environments, accurately assessing movement and location using WiFi signals requires more than just identifying the set of visible Access Points (APs). It is crucial to consider the frequency of these AP occurrences, primarily because of multi-path propagation. In such settings, WiFi signals can reflect off various surfaces, allowing signals from distant APs to reach the receiver. These reflected signals often have a longer travel path than direct signals from nearer APs. As a result, APs that are closer and in direct line of sight to the user's smartphone tend to appear more frequently in scan results compared to those farther away.

We propose a modified version of the Jaccard Index to capture this dynamic, which we denote as β. This new metric accounts for the frequency of AP occurrences across consecutive *WiFi scan windows*. In this context, the term *WiFi scan window* refers to a specific time interval of set length Δ, during which a series of WiFi scans are systematically conducted. Each scan extracts the Service Set Identifier (SSID) of visible APs. An SSID uniquely identifies a WiFi network.

Let $S(t)$ represent the set of unique SSIDs detected in the WiFi scan window starting at time t, where s_i is the SSID associated with a unique WiFi AP. The frequency of an AP, denoted as $f_t(s)$, refers to how often that AP appears within the scan window at time t.

The modified Jaccard Index, $\beta(t)$, for successive WiFi scan windows at times $(t-\Delta)$ and t, is defined as follows:

$$\beta(t) = \frac{\sum\limits_{s \in S(t-\Delta) \cup S(t)} \min\{f_t(s), f_{t-\Delta}(s)\}}{\sum\limits_{s \in S(t-\Delta) \cup S(t)} \max\{f_t(s), f_{t-\Delta}(s)\}}$$

This equation calculates β as the ratio of the minimum frequency of each AP occurring in both windows to the maximum frequency of the same APs across the two windows. The value of β ranges from 0 to 1. A value of 0 indicates no overlap, suggesting significant movement or location change. A value of 1 implies complete overlap, indicating minimal or no movement. Intermediate values show varying degrees of change in the WiFi environment, with values closer to 1 suggesting smaller movements and those nearer to 0 indicating more substantial location shifts. This metric is particularly

useful in indoor settings for assessing motion states, such as walking or driving, based on WiFi landscape variations over time.

While using WiFi signal strengths to compute β is theoretically possible, it was excluded due to the unreliability of signal strength in indoor environments. Factors such as multipath effects, where signals bounce off walls and obstacles, and environmental interference, such as device placement and user movement, cause significant fluctuations that do not reliably correlate with distance. By focusing solely on the appearance and disappearance of APs, our approach avoids these inconsistencies and ensures a more robust and accurate movement detection system.

To validate our proposed approach, we utilize histogram plots. Figure 5.3 presents histograms illustrating the Jaccard Index of WiFi APs for both *walk* and *drive* states within an indoor parking facility. Notably, the index values for these states are dispersed across the full range from 0 to 1. This widespread distribution starkly contrasts findings from outdoor environments [107], underscoring the inappropriateness of the Jaccard Index for distinguishing between states in indoor settings. Additionally, Figure 5.4 offers a comparative analysis of the histogram of β values for *walk* and *drive* states in similar indoor conditions. The histogram for the *drive* state demonstrates a distinct left-skewed distribution, indicating a significant difference in location displacement during driving compared to walking.

The peak at $\beta \approx 0.9$–1.0 for the drive state reflects the constrained and stable nature of driving in indoor parking facilities. Slow driving along predefined paths results in minimal changes to the WiFi environment, with most APs remaining visible and their frequencies similar across successive scan windows. Multi-path effects further ensure consistent visibility of even distant APs, contributing to higher β values.

Conversely, low β values ($\beta \approx 0$) are rare in driving because significant differences in AP frequencies are unlikely in such structured, low-speed scenarios. These values are more typical in rapid movement settings, such as outdoor driving, where users quickly transition between non-overlapping WiFi zones. In contrast, walking produces more variation in AP frequencies due to irregular movement, resulting in a broader distribution of β values. This left-skewed histogram for the drive state highlights β's effectiveness in distinguishing between walking and driving in indoor environments.

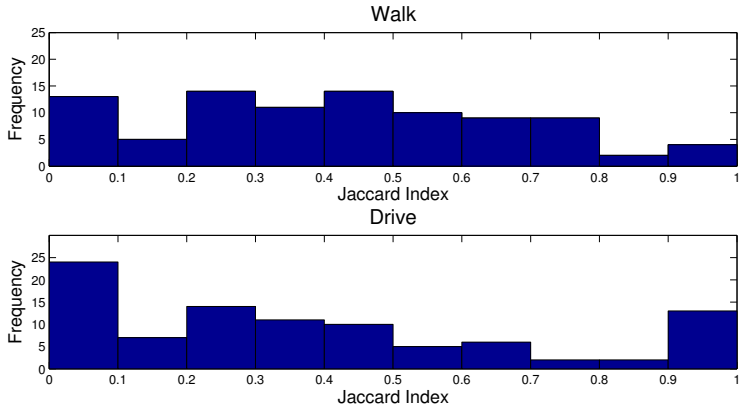


Figure 5.3: Histogram of Jaccard Index for walking and driving in the indoor parking facility. Using the Jaccard Index, it is not easy to distinguish *drive* state from *walk* state.

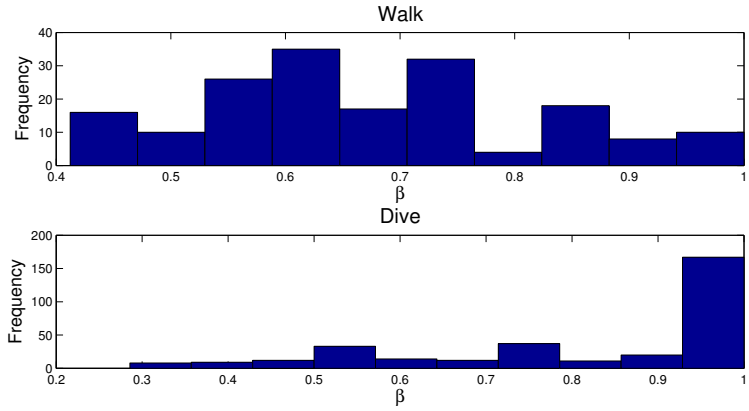


Figure 5.4: Histogram of β for walking and driving in the indoor parking facility. The left skewed histogram of *drive* state shows that β is a better metric to distinguish *drive* from *walk* state.

Utilizing a predetermined threshold for β values, we categorize WiFi-based states as either w_{moving} or $w_{\neg\text{moving}}$, which are subsequently integrated into the States Merger module for further analysis.

States Merger

The State Merger plays a pivotal role as the fusion module, integrating the asynchronous outputs from both the Accelerometer State Classifier and the WiFi State Classifier to ascertain the user's current locomotive state. At any given time t, the State Merger computes the user's combined state $C(t)$, which is derived from the current states $A(t)$ from Accelerometer State Classifier and $W(t)$ from WiFi State Classifier, as well as the preceding combined state $C(t-1)$. The possible classes for $C(t)$ are $walk, drive, still$,

and $unknown$, which collectively represent all potential locomotive states of a user within an indoor parking facility.

Due to the varying time windows of Accelerometer State Classifier and WiFi State Classifier, the State Merger employs a 'wait-and-proceed' strategy to align these asynchronous states. It looks for temporal overlaps between incoming $A(t)$ and $W(t)$ states to accurately compute $C(t)$ for intervals where such overlaps occur. The overlap period is defined as the interval starting from the later of the two start times and ending at the earlier of the two end times of the respective states.

The State Merger's rule-based approach combines $A(t)$, $W(t)$, and $C(t-1)$, as outlined in Table 5.1, where '*' signifies any state. This table is instrumental in differentiating between $walk$ and $drive$ states, the core objective of the state detection module. If the WiFi State Classifier suggests a $moving$ state but the Accelerometer State Classifier does not indicate a $walk$ state, the inferred state is $drive$. Conversely, a $still$ state is determined when neither $moving$ nor $walk$ states are detected. A $walk$ state is deduced whenever the Accelerometer State Classifier identifies a $walk$ state, regardless of other states. In instances where the Accelerometer-based state is $unknown$, the State Merger reverts to the previous state $C(t-1)$. In contrast to more complex methods like HMM, this rule-based system offers a streamlined approach to state detection, crucial for minimizing latency and enhancing the precision in state transition detection and event localization.

The latency in the initial state detection for the Accelerometer State Classifier is influenced by two key parameters: *MinPts* and the window size. On the other hand, the initial detection time for the WiFi State Classifier is primarily determined by the duration of the *WiFi Scan Window*. The specific values of these parameters used in our experiments are detailed in Table 5.2, and the rationale behind their selection is further expounded upon in subsection 5.4.

5.3.3 Event Detection

The core objective of the Event Detection module is to pinpoint state transitions corresponding to parking and un-parking events. As outlined in Algorithm 3, this module analyzes transitions between *walk* and *drive* states. Detecting *walk-to-drive* and *drive-*

$A(t)$	$C(t-1)$	$W(t)$	$C(t)$
a_{walk}	*	*	$walk$
$a_{\neg walk}$	*	$w_{\neg moving}$	$still$
		w_{moving}	$drive$
$a_{unknown}$	$unknown$	*	$unknown$
	$walk$	$w_{\neg moving}$	$unknown$
		w_{moving}	$walk$
	$drive$	$w_{\neg moving}$	$unknown$
		w_{moving}	$drive$
	$still$	$w_{\neg moving}$	$still$
		w_{moving}	$unknown$

Table 5.1: Rule for combining Accelerometer and WiFi-based states to obtain the final state.

to-walk transitions is crucial for identifying parking and un-parking events, as these transitions represent the user's state change before and after vehicle usage. Specifically, a parking event corresponds to a transition from *drive* to *walk*, indicating the user has exited the vehicle, while an un-parking event involves a transition from *walk* to *drive*, signaling the vehicle is being used again.

The parameter ν ensures that detected transitions persist for at least 10 seconds, filtering out brief, transient state changes that could lead to false positives. This value of 10 seconds was empirically chosen based on extensive experiments, as it provides the optimal balance between filtering noise and capturing meaningful transitions. By stabilizing the detection process, ν focuses on sustained transitions while excluding irrelevant states such as *still* or *unknown*.

5.3.4 Event Localization

Upon successfully detecting a parking or un-parking event, the mobile client communicates with the cloud service via two distinct request types. The first type, a Park Request, involves relaying a detected parking event along with the corresponding WiFi signature captured at the time of the event. The cloud service, upon receipt of this data, analyzes the WiFi signature to deduce the parking location identifier L_{id} and communicates this back to the mobile client while concurrently updating the status of L_{id} as occupied. The second type, an Un-park Request, entails transmitting the L_{id} to signify an un-parking event. The cloud service, in turn, updates the status of L_{id} to reflect its availability. These interactions ensure real-time and accurate tracking of parking space

Algorithm 3 State transition detection algorithm

INPUT: Current state $C(k) = \{type, duration, endTime\}$
INITIALIZE: $sL \leftarrow \{\}$, a list of merged state
procedure DETECTSTATETRANSITION($C(k)$)
  $matchFound \leftarrow$ **false**
  if sL is **not** empty **then** *// Check history*
    for L_i in sL **do**
      if $L_i.type = C(k).type$ **then**
        remove L_i from sL
        $C(k).dur \leftarrow C(k).dur + L_i.dur$
        append $C(k)$ to sL
        $matchFound$ = ***true***
  if not $matchFound$ **then**
    append $C(k)$ to sL
  if $sizeof(sL) = 2$ **then**
    CHECKTRANSITION(sL(0),sL(1));
    if $(sL(1).dur \geq \nu)$ **then**
      // Keep only the last stable state
      remove $sL(0)$ from sL;
procedure CHECKTRANSITION(prevState,curState)
  if $(prevState.dur \geq \nu)$ **and** $(curState.dur \geq \nu)$ **then**
    declare $prevState.type \rightarrow curState.type$
    transition at time $prevState.endTime$

occupancy.

The Event Localization module is tasked with identifying the precise parking location, denoted as L_{id}, utilizing the WiFi signature transmitted by the mobile client. For this purpose, we integrate a WiFi-based indoor localization system within our cloud service, employing the Horus method, a renowned Bayesian Inference technique [173]. We chose a WiFi-based approach for event localization due to its several advantages. Firstly, WiFi infrastructure is widespread and cost-effective. Secondly, since our BluePark system already uses WiFi to ascertain the user's locomotive state, leveraging the same data for event localization streamlines the process, negating the need for extra sensors and reducing complexity.

In implementing the Horus WiFi localization, our initial step involved the collection of WiFi fingerprints from different locations of the parking facility. This was particularly focused on the area depicted in Figure 5.7. These fingerprints, once gathered, were matched with the WiFi data obtained during parking and un-parking events to determine the precise event location. For practical identification of parking locations, we utilized the alphanumeric markings on pillars adjacent to the parking spots.

We noted uneven WiFi signal distribution during the data collection phase in the parking facility. This irregularity primarily stemmed from the scattered positioning of WiFi APs installed by various retailers on the floors above the parking area. To mitigate this challenge and ensure the consistent accuracy of our localization system, we meticulously fine-tuned it. This allowed us to accommodate and counteract the nuances of signal variation and non-uniformity within the parking area.

5.4 Tuning of Parameter

In this section, we detail the parameter selection process for our Accelerometer State Classifier and WiFi State Classifier. To define these parameters accurately, we gathered extensive accelerometer and WiFi data, representing various user states, including $walk$, $drive$, and $\neg moving$.

For the $drive$ data collection, we operated under the assumption that the smartphone remains stationary relative to the vehicle's movement. This approach meant that the phone's on-body position was not a factor in our $drive$ data collection. In total, we amassed 43 minutes of driving data within the parking area, ensuring a representative sample of the $drive$ state.

In contrast, the $walk$ data required considering the phone's position relative to the user's body, as this can significantly impact the accelerometer readings. To capture a comprehensive dataset, we recorded continuous walking data over 43 minutes for each of the three most common phone placements: in a trouser pocket, held in hand, and carried in a backpack. This varied data collection aimed to encapsulate the range of real-world scenarios users may encounter while walking, thus providing a robust basis for our classifiers' parameter determination.

5.4.1 Accelerometer State Classifier Parameters

For optimization purposes, our analysis focused on fine-tuning the threshold value (θ) and the minimum points ($MinPts$) for our BluePark system. We examined θ values ranging from 0.2 to 0.5 and ultimately selected 0.35 as the optimal threshold. Similarly, we explored $MinPts$ values between 2 to 5, concluding that $MinPts$ = 2 was most

effective for our system.

With these settings established, we calculated the α values for the collected accelerometer data. The cumulative distribution of α for both $walk$ and $drive$ states is illustrated in Figure 5.5. This plot distinguishes between $walk$ states, which remain consistent across various on-body phone positions, and $\neg walk$ states (encompassing both driving and loitering activities). It's worth noting that while $walk$ states exhibit α values ranging between 2 and 20, $drive$ states are characterized by an α lower than 2 with a probability of 0.99. This disparity in α values underpins the Accelerometer State Classifier's ability to effectively differentiate between $walk$ and $\neg walk$ states, setting the stage for accurate state classification in our BluePark system.

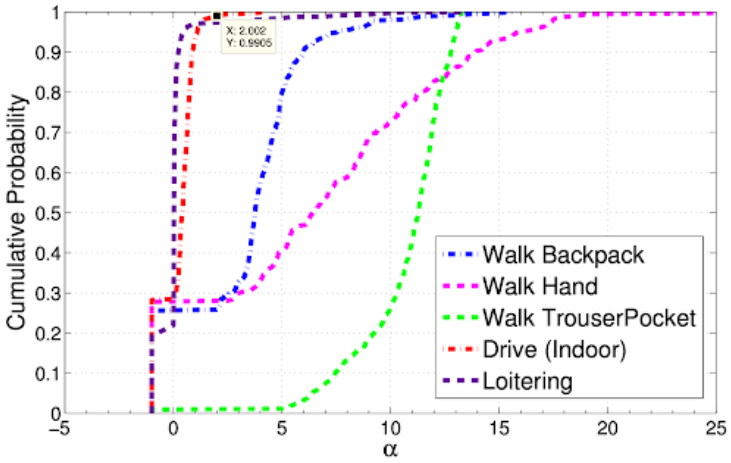


Figure 5.5: Cumulative distribution of α for walk and drive states. $walk$ state at all phone positions are distinguishable whereas $drive$ and $loitering$ states are not distinguishable from each other.

5.4.2 WiFi State Classifier Parameters

In the WiFi State Classifier of BluePark, we established a scanning window of 20 seconds with scans initiated every 5 seconds to capture WiFi signal data. The cumulative distribution function of the β values for different user states – $walk$, $drive$, and $\neg moving$ – is depicted in Figure 5.6. This distribution effectively illustrates that while $walk$ and $drive$ states present similar β values, the $\neg moving$ state is distinctly identifiable. Specifically, a β value not exceeding 0.34 indicates a $\neg moving$ state with a 90% probability, as opposed to less than a 15% probability for both $walk$ and $drive$ states. This distinction in β values is crucial for the WiFi State Classifier to accurately differentiate between the $moving$ states (walking and driving) and the $\neg moving$ state.

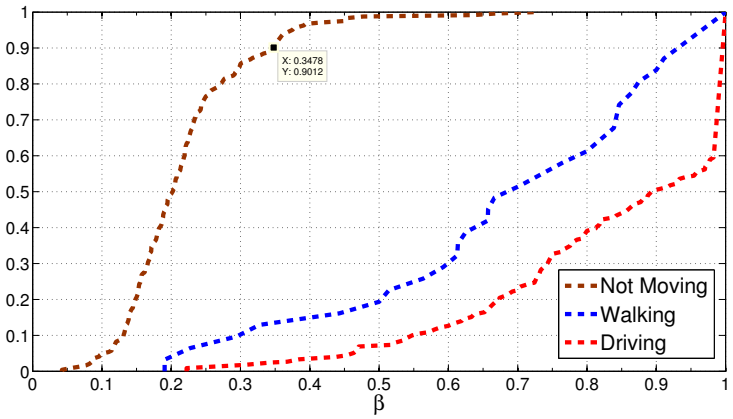


Figure 5.6: Cumulative distribution of β values for $walk$, $drive$, and $\neg moving$ states.

Module	Parameters	Classifier Threshold
Accelerometer State Classifier	window size = 6s $MinPts = 2$ $\theta = 0.35$	$A_{state} = \begin{cases} a_{\neg walk} & \text{if } \alpha < 2 \\ a_{walk} & \text{if } 2 \leq \alpha \leq 20 \\ a_{unknown} & \text{if } \alpha > 19 \end{cases}$
WiFi State Classifier	sampling interval = 5s window size = 20s	$W_{state} = \begin{cases} w_{moving} & \text{if } \beta \geq 0.34 \\ w_{\neg moving} & \text{if } \beta < 0.34 \end{cases}$
Event Detection	$\nu = 10s$	NA

Table 5.2: Comprehensive list of parameter configurations for BluePark's assessment.

For reference and clarity, a detailed table of all the parameters employed in the BluePark system, including those for the WiFi State Classifier, is provided in Table 5.2.

5.5 Data Collection

Our experimental setup was conducted in the underground parking facility of a bustling shopping mall in New Delhi. The layout of this specific parking area, along with the zones targeted for WiFi fingerprinting, is illustrated in Figure 5.7. Additionally, Figure 5.1 visually represents the facility, showcasing unique alphanumeric codes marked on each parking pillar. These codes are crucial as they serve as location identifiers within the BluePark system. To evaluate localization accuracy, we considered the standard dimensions of each parking spot (2.5 meters x 5 meters) and the overall area of the parking facility (172.5 meters x 165 meters), focusing only on those spots where WiFi fingerprinting was conducted.

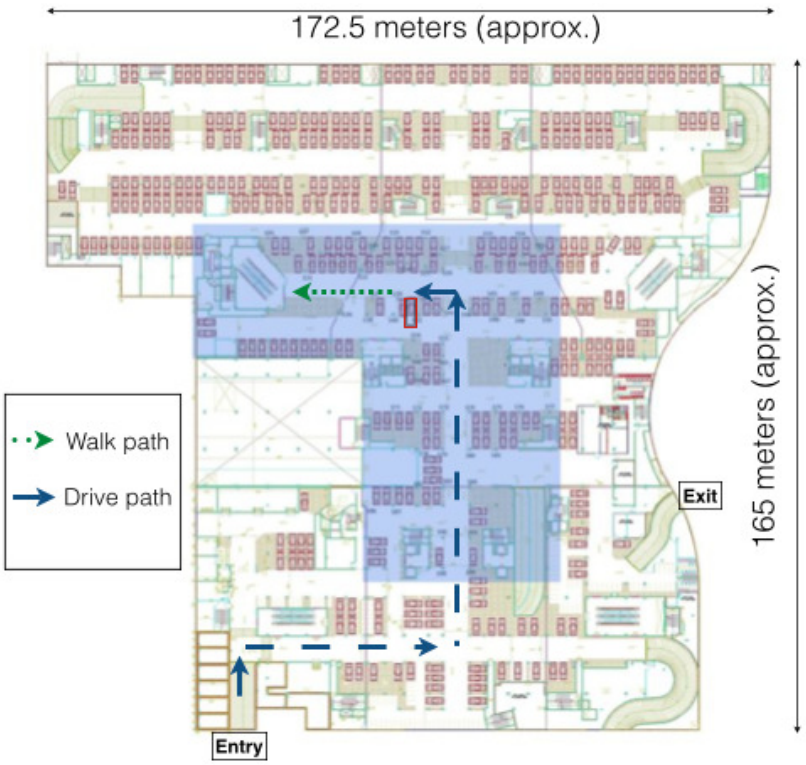


Figure 5.7: Blueprint of the indoor parking area where the BluePark system was deployed. The Wi-Fi fingerprinted areas are highlighted in blue.

The data collection process was divided into two main phases. The initial phase involved creating an extensive WiFi fingerprint map. This task entailed collecting data from 56 different parking spots, during which 53 unique WiFi APs were identified. However, this does not imply a one-to-one correspondence between parking spots and APs. Many APs have overlapping coverage and are visible from multiple parking spots, while each parking spot can also detect signals from several APs. The 53 APs represent the total number of unique access points detected across all parking spots. The second phase focused on gathering real-world data during actual parking and un-parking events. This involved using the Mobile Client, an Android application detailed in Section 5.3.1, designed to accumulate and analyze both WiFi and accelerometer data. To test the versatility of our system, we conducted experiments across various accelerometer sampling rates (5Hz, 10Hz, 15Hz, 50Hz, and 99Hz) and evaluated the system's performance for three commonly encountered phone placements: handheld, in a trouser pocket, and inside a backpack. An exhaustive summary of these experiments, encompassing a range of parking and un-parking scenarios across different phone positions and sampling rates, is methodically laid out in Table 5.3.

Table 5.3: Enumeration of parking and un-parking experiment pairs, differentiated by the phone's position and the chosen sampling frequency.

Phone Position	Frequency	Number of Instances
Trouser pocket	99 Hz	5
	50 Hz	5
	15 Hz	5
On hand	99 Hz	5
	50 Hz	5
	15 Hz	5
Back Pack	99 Hz	5
	50 Hz	5
	15 Hz	5

5.6 Evaluation

For our evaluation, we selected Google's Activity Recognition API (hereafter referred to as GARA) [53] as the benchmark to evaluate the effectiveness of our BluePark system in state detection. GARA is a widely recognized activity detection service on the Android platform, proficient in identifying various states such as STILL, ON_FOOT, IN_VEHICLE, TILTING, ON_BICYCLE, and UNKNOWN. For a fair comparison, we replaced BluePark's original State Detection method with GARA. This substitution enables us to underscore the enhanced precision of BluePark's State Detection mechanism, which is specifically optimized for indoor parking environments, as opposed to GARA's general state detection function.

GARA's sampling frequency is not publicly disclosed, prompting us to test BluePark across a spectrum of accelerometer sampling frequencies, namely 5Hz, 10Hz, 15Hz, 50Hz, and 90Hz. The comparative assessment between these two systems focuses on four critical aspects: the accuracy of State Detection, the accuracy of Event Detection, the latency in detecting events, and the precision in Event Localization. When integrating Google Play Services for activity detection, developers are required to specify an update interval. In our experimental setup, we utilized version 6.1 of the Activity Recognition API and selected a 3-second update interval, aiming to minimize detection delays. This 3-second interval was maintained consistently throughout our evaluation, except in specific cases where a deviation is explicitly mentioned. For a comprehensive understanding of the parameter settings utilized in the BluePark system during this comparative analysis, refer to Table 5.2.

5.6.1 Accuracy of State Detection

GARA	BluePark				
	5Hz	10Hz	15Hz	50Hz	99Hz
72.10%	93.38%	93.90%	95.27%	95.08%	95.98%

Table 5.4: Accuracy of $drive$ detection using BluePark and Google Activity Recognition API (GARA). BluePark detects $drive$ state better than GARA.

We consider GARA's IN_VEHICLE as *drive* and ON_FOOT state as *walk*. Table 5.4 presents the accuracy percentages of both methods for the $drive$ state. Correspondingly, Table 5.5 offers a comparative view of the $walk$ state detection in the three on-body phone positions.

Table 5.4 demonstrates that BluePark outperformed GARA in accurately detecting the $drive$ state at each of the evaluated accelerometer sampling frequencies. Specifically, even at its lowest performance, BluePark yielded an accuracy of 93.38% at 5Hz, which is significantly higher than GARA's 72.10%. Moreover, we observe an intriguing trend in BluePark's performance across the sampling frequencies. Its accuracy witnessed an increase as the frequency escalated from 5Hz to 15Hz. However, a subsequent marginal decline is observed from 15Hz to 50Hz, with a slight augmentation at 99Hz. Furthermore, the difference in accuracy between BluePark's lowest (5Hz) and highest (99Hz) frequencies is a mere 2.6%. These results suggest that beyond a certain threshold (15Hz in this context), further increments in the sampling rate do not substantially enhance accuracy. A moderate frequency, like 15Hz, could offer nearly equivalent accuracy while potentially being more resource-efficient. Given the notable disparity in performance, it's evident that GARA might not be the optimal solution for applications necessitating precise $drive$ state detection, especially when contrasted with BluePark. While GARA does not disclose the sampling frequency, the potential of BluePark to offer high accuracy at lower frequencies suggests an avenue for balancing performance with resource conservation, such as battery lifespan and computational overload.

Phone position	GARA	BluePark					
		5Hz	10Hz	15Hz	50Hz	99Hz	Average
Trouser Pocket	100.00%	98.60%	98.14%	98.78%	98.56%	98.55%	98.52%
Hand	96.45%	98.05%	98.72%	98.93%	98.95%	98.74%	98.68%
Backpack	99.57%	85.00%	96.75%	96.86%	98.21%	98.21%	95.07%
Average	98.65%	94.41%	98.10%	98.48%	98.65%	98.56%	98.00%

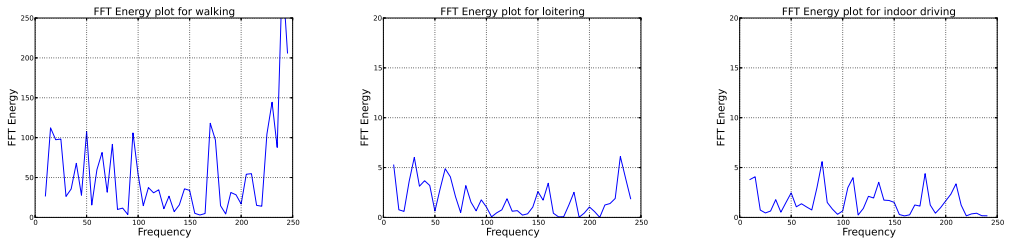
Table 5.5: Accuracy of $walk$ using BluePark and Google Activity Recognition API (GARA)

Table 5.5 presents a comparison of $walk$ detection accuracy between BluePark and GARA, offering insightful observations. On average, BluePark achieves a commendable accuracy of approximately 98.00% across varying phone positions, closely paralleling GARA's 98.65%.

The placement of the phone is a crucial factor influencing detection accuracy. In scenarios where the phone is stored in a trouser pocket, BluePark demonstrates an accuracy of 98.52%, slightly trailing GARA's optimal accuracy of 100%. Contrastingly, when the phone is held in hand, BluePark surpasses GARA with an accuracy of 98.68%, compared to GARA's 96.45%. A notable disparity is observed when the phone is placed in a backpack. Here, GARA maintains robust accuracy at 99.57%. However, BluePark experiences a dip at a 5Hz sampling frequency, registering an accuracy of only 85.00%. Notably, as the sampling frequency escalates, BluePark shows marked improvements, averaging 95.07% accuracy in the backpack position. It becomes evident that increased sampling frequencies generally enhance BluePark's efficacy, particularly around the 15Hz or 50Hz marks. Beyond 15Hz, however, the incremental gains in accuracy become marginal. Excluding the backpack position at a 5Hz frequency, BluePark consistently attains high accuracy levels, often exceeding 96%, irrespective of the phone's position or the chosen frequency.

To accurately identify parking and un-parking events, the system needs to distinguish between the $walk$ and $drive$ states proficiently. Based on our findings, BluePark performs better in detecting the $drive$ state and equally well in the $walk$ state, making it a better choice than GARA for indoor parking situations.

The superior performance of BluePark can be attributed to differences in the underlying methodologies. While GARA does not disclose its internal workings and likely relies solely on accelerometer data for activity detection, BluePark enhances detection by combining accelerometer and WiFi data, allowing it to identify subtle motion patterns more effectively. Slow driving in indoor parking environments often generates weak accelerometer signals that resemble stationary states, which GARA struggles to differentiate. In contrast, BluePark's hybrid approach leverages WiFi signal dynamics to enhance detection accuracy, particularly for the *drive* state.



(a) Walking (b) Loitering (c) Indoor Driving

Figure 5.8: FFT Energy plot of accelerometer in different locomotive states

Exploring the Causes of GARA's Reduced Accuracy in *drive* Detection

Traditionally, in HAR, accelerometers have been a popular choice for detecting user locomotive states. It is reasonable to assume that GARA also relies solely on accelerometer data. Figure 5.8 shows the FFT energy of accelerometer data over a 5-second window for different user states within an indoor parking facility. In the figure, the FFT energy plot for walking (Figure 5.8a) shows significant peaks across various frequencies, indicating consistent and repetitive motion patterns characteristic of walking. In contrast, loitering (Figure 5.8b), defined as sporadic movement without continuous walking, exhibits lower and more irregular FFT energy peaks compared to walking, reflecting its less structured motion. Indoor driving (Figure 5.8c) displays relatively low energy levels with less frequent and more dispersed peaks, similar to loitering but with even less regularity.

Walking is clearly distinguishable from the other two states due to its higher and more consistent energy peaks. However, loitering and driving display relatively similar energy distributions, which, although not identical, might contribute to the challenges GARA faces in accurately detecting driving within indoor parking facilities. This similarity, albeit not very pronounced, could be a contributing factor to GARA's reduced performance in distinguishing the driving state from loitering in such environments. Moreover, loitering is a common behavior in indoor parking spaces, yet it is not one of the classes for which GARA has been trained. This lack of training on loitering further complicates GARA's ability to accurately differentiate between loitering and driving, leading to potential misclassifications and reduced detection accuracy in such settings. In contrast, our system, which leverages a combination of WiFi and accelerometer data, is capable of classifying loitering as walking.

5.6.2 Accuracy of Event Detection

Event	Phone position	GARA		BluePark									
				5Hz		10Hz		15Hz		50Hz		99Hz	
		P	R	P	R	P	R	P	R	P	R	P	R
Parking	Trouser Pocket	100	100	100	100	100	100	100	100	100	100	100	100
	Hand	100	100	100	100	**80**	100	100	100	100	100	100	100
	BackPack	100	100	**80**	100	100	100	100	100	100	100	100	100
Unparking	Trouser Pocket	100	100	**83.33**	100	100	100	100	100	100	100	100	100
	Hand	100	100	100	100	**80**	100	100	100	100	100	100	100
	BackPack	100	100	**80**	100	100	100	100	100	100	100	100	100

Table 5.6: Comparative analysis of Precision (P) and Recall (R) for Parking and Unparking Event Detection between BluePark and Google Activity Recognition API (GARA) across various sensor frequencies and phone placements.

Table 5.6 presents a comparative analysis of BluePark's and GARA's efficacy in detecting parking and un-parking events, taking into account various phone placements and sampling rates. In this comparison, BluePark exhibits some instances of false positives at lower sampling rates of 5Hz and 10Hz, leading to precision and recall values below 100%. However, at a sampling rate of 15Hz or above, BluePark consistently achieves 100% precision and recall. After reviewing the results, we have selected a sampling rate of at least 15Hz for further evaluation.On the other hand, GARA consistently maintains a high standard with 100% precision and recall across all scenarios.

It is important to note that while precision and recall metrics are indicative of the accuracy with which events are detected, they do not encompass the system's responsiveness. Responsiveness, or the system's reaction time, is a critical factor when it comes to accurately pinpointing the location of an event or verifying the authenticity of parking information. The subsequent section delves into an examination of this aspect, exploring the temporal efficiency and responsiveness of the system in real-world scenarios.

5.6.3 Latency in Event Detection

Latency in event detection represents the time gap between when an event actually happens and when it gets detected by the system. During data collection, we manually noted down the actual event times for both BluePark and GARA. Figure 5.9 portrays the cumulative distribution function of this latency. BluePark identifies parking transitions within a mere 20-second delay 90% of the time. In contrast, GARA often requires up to 30 seconds for comparable accuracy. When it comes to un-parking detection, BluePark

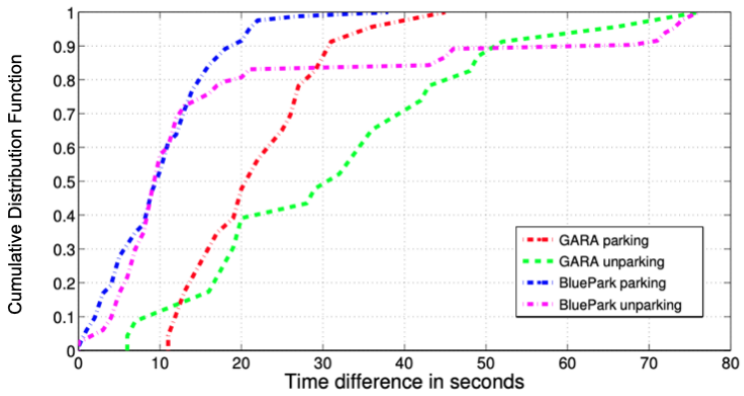


Figure 5.9: Cumulative distribution of time difference between ground truth and detected time for parking and un-parking activity. BluePark detects parking within 20 seconds with a probability of 0.9, while GARA takes around 30 seconds for the same probability.

requires only 20 seconds 80% of the time, whereas GARA lags slightly behind, taking 28 seconds. A contributing factor to GARA's longer latency is its lesser accuracy in detecting the drive state, as seen earlier in Table 5.4.

The latency of event detection influences the WiFi signature recorded at the moment of parking. An extended latency might lead to recording an inaccurate WiFi signature, blurring the clarity of the parking spot's location. The ramifications of this delay on localization accuracy are explored in the next subsection.

5.6.4 Accuracy of Event Localization

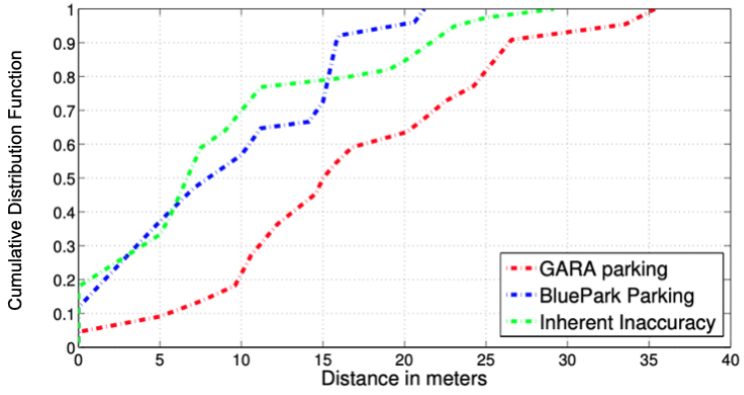


Figure 5.10: Cumulative distribution of location error in terms of meters. BluePark closely follows the inherent latency, whereas GARA is less accurate due to higher detection latency.

In Event Localization, the precise determination of a vehicle's parking spot is cru-

cial. This accuracy is influenced by event detection latency and the inherent constraints of the chosen localization method. We present a comparative visual analysis of location errors for both BluePark and GARA in Figure 5.10. This figure also includes inherent error data to ensure a comprehensive and fair comparison.

In over 90% of the evaluated cases, BluePark's location inaccuracy was less than 15.81 meters, which translates to a deviation of approximately six parking spots from the actual location. In contrast, GARA showed a location inaccuracy of 26.7 meters for the same 90% confidence level, resulting in a deviation of about 10 parking spots. When looking at median location errors, BluePark pinpointed an error of 10 meters, while GARA presented a 15.4 meter discrepancy.

BluePark achieves better location accuracy than the baseline GARA primarily due to its superior drive state detection. BluePark leverages a combination of accelerometer and WiFi data, enabling more accurate detection of parking and un-parking events. This improved detection leads to reduced latency in identifying *walk-to-drive and drive-to-walk* transitions. The lower latency directly impacts localization accuracy by ensuring that the WiFi signature used for localization corresponds closely to the actual transition point. In contrast, GARA's less accurate drive state detection introduces higher latency, causing mismatches between the detected event and the associated WiFi signature, ultimately reducing localization accuracy.

5.7 Conclusion

This chapter presented the BluePark system, a novel real-time sensing framework designed specifically for the autonomous detection and localization of parking and un-parking events in indoor environments. BluePark skillfully utilizes a combination of accelerometer and WiFi data to distinguish between *drive* and *walk* states. The system is enhanced by a state transition detection algorithm that operates with low latency, reliably identifying shifts between these states. These transitions serve as critical indicators of parking and un-parking events and are detected without requiring user intervention.

The integration of the Horus WiFi Localization system further empowers the framework, enabling precise event localization. Our extensive evaluations in a busy underground parking lot of a shopping mall in Delhi have demonstrated BluePark's supe-

rior performance in state detection, particularly when compared to the Google Activity Recognition API. Notably, BluePark exhibits reduced detection latency, even at lower sampling rates. This is a significant advantage, considering our system relies on an unsupervised learning model, thus eliminating the need for user-specific calibration.

While the Google Activity Recognition API may have higher overall precision and recall, BluePark compensates with considerably lower latency in state detection. This contributes to its enhanced accuracy in event localization, surpassing other solutions that depend on the Google API. Our tests also confirmed BluePark's robustness across various common on-body phone placements, indicating its versatility and effectiveness. In summary, BluePark emerges as a comprehensive solution, setting a new benchmark in real-time indoor parking detection and localization.

CHAPTER 6

Using an Arduino and a Smartwatch to Measure Liquid Consumed from any Container

6.1 Introduction

Proper hydration is fundamental for maintaining physical fitness, overall health, weight management, and cognitive function [120]. Unfortunately, busy schedules often lead to irregular hydration patterns, with many people depending on thirst as their primary cue to drink water. However, relying solely on thirst is problematic, as it is actually an indicator of existing dehydration. This delayed response can result in individuals feeling fatigued before recognizing the need to hydrate. Furthermore, the situation is complicated by the fact that thirst mechanisms weaken with age. Older adults often confuse thirst with hunger, leading to inappropriate eating when they actually need to hydrate. This confusion can contribute to both obesity and dehydration. In fact, it is estimated that 75% of the US population suffers from chronic dehydration [172], which can have serious long-term health consequences. Given these challenges, there is a clear benefit to tracking a person's liquid intake and providing timely reminders to hydrate. Such proactive hydration management could help maintain a healthy hydration level, thereby mitigating the risks associated with chronic dehydration and its related health issues.

Various methods [30, 38, 6] [1] [2] have been developed to monitor an individual's liquid intake, each with its own set of advantages and limitations. One common approach is using mobile applications that require users to input their liquid consumption data manually. Based on user inputs and predefined hydration goals, these apps send reminders to encourage more frequent drinking. Although the ubiquity of smartphones makes such apps accessible to most people, the reliance on manual data entry poses a significant drawback. This requirement often becomes burdensome, particularly for

[1] http://www.hydracoach.com
[2] http://trago.co

individuals with busy schedules, leading to inconsistent usage and, in many cases, a failure to hydrate regularly.

Specialized smart bottles such as HydraCoach[1] and Trago[2] have been introduced as an alternative. These smart bottles come equipped with built-in sensors designed to directly track and monitor liquid consumption, thereby greatly reducing manual input and providing a more streamlined and user-friendly tracking experience. A primary drawback of these smart bottles is their restriction to a single, specific container, which can lead to underestimating total hydration if users consume liquids from other sources. Additionally, these bottles typically do not account for factors such as spillage or the possibility of the bottle being used by multiple individuals, which can lead to inaccuracies in tracking actual liquid intake. Furthermore, a significant barrier to widespread adoption is the cost; these smart bottles are often priced substantially higher than traditional bottles, making them a less accessible option for many consumers.

In designing an effective and user-friendly liquid intake tracking system, it is crucial to consider potential users' varied preferences and lifestyles. An ideal system should seamlessly integrate into daily routines and be adaptable and capable of tracking hydration from multiple sources, not just a single bottle or container. This multi-source tracking capability is essential to accurately reflect an individual's total fluid intake, accommodating various drinking habits and scenarios. Additionally, affordability is a significant consideration, ensuring that effective hydration monitoring is accessible and practical for a wide range of users.

With these key factors in mind, this chapter introduces an innovative automated liquid intake tracking system. This system uniquely combines a detachable base with a commercially available smartwatch, offering a comprehensive and versatile solution to overcome the limitations of existing methods. Our proposed system enhances user convenience and maintains affordability, making hydration monitoring both practical and accessible. Moreover, it is designed to account for liquid consumption from various containers, providing a more accurate and holistic view of an individual's hydration status. This approach ensures a user-friendly, cost-effective, and comprehensive solution for accurately tracking hydration, catering to the diverse needs of individuals in their daily lives.

The remaining sections of this chapter are organized as follows. In Section 6.2, we

provide an overview of the proposed system's functionality and detailed descriptions of each component. Following that, Section 6.3 presents the experimental procedure and evaluation of the system's performance. Additionally, in Section 6.3.4, we explore potential future research directions and areas for improvement in the proposed system. Finally, in Section 6.4, we conclude this chapter by summarizing our study's key findings and contributions.

6.2 Proposed System

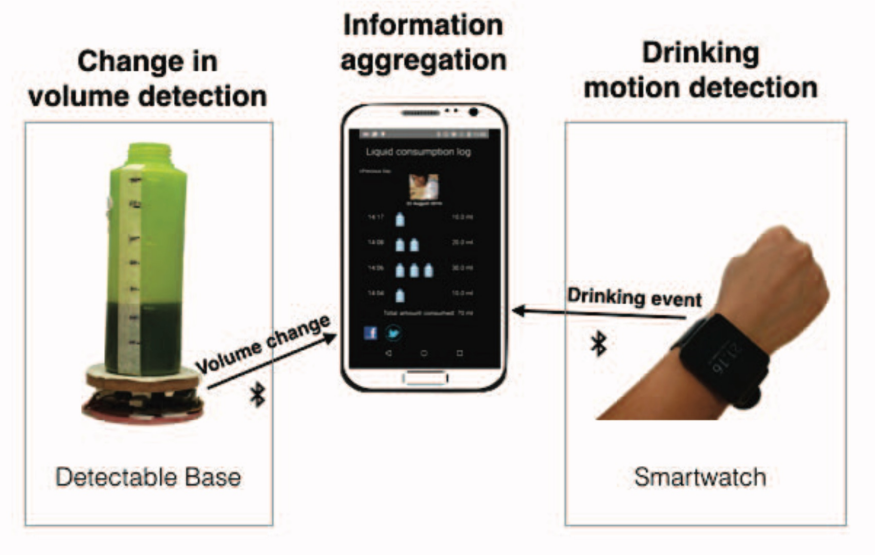
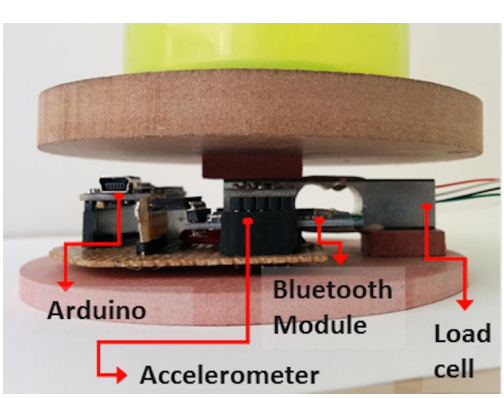


Figure 6.1: System architecture of the proposed liquid intake tracking system featuring detachable base and smartwatch.

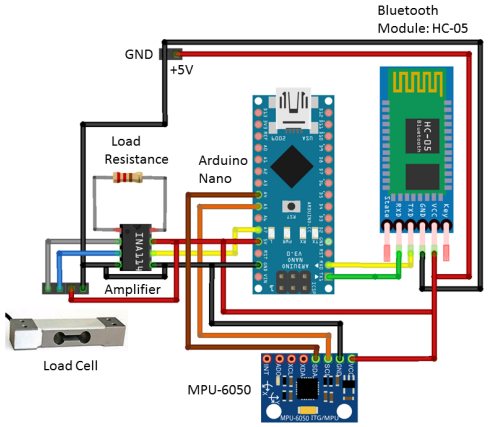
Figure 6.1 presents the architecture of our innovative liquid intake tracking system. The system comprises two primary components: a detachable base and a smartwatch, both of which interface with the user's smartphone via Bluetooth. The detachable base is designed to be easily attached to commonly used drink containers and includes a sensing module equipped with a load cell. The load cell accurately measures the changes in liquid quantity within the container after each sip, facilitating real-time tracking of liquid consumption volumes. Additionally, the smartwatch, equipped with an accelerometer sensor and worn on the user's wrist, plays a crucial role in gesture recognition. It captures motion data to detect the user's drinking gestures, utilizing machine learning to identify drinking events. Data from both the detachable base and the smartwatch are transmitted to the user's smartphone via Bluetooth. The detachable base provides information about the volume changes in the container, while the smartwatch sends data

pertaining to detected drinking gestures. We integrate the information from these two sources to enhance the system's accuracy. This integration is crucial in distinguishing genuine water intake events, verified by the simultaneous occurrence of a volume change in the container and a drinking gesture detected by the smartwatch. This approach ensures the smartphone receives comprehensive data, combining it to give users a detailed timeline report of their liquid consumption. By correlating volume changes with drinking gestures, our system effectively accounts for discrepancies caused by actions other than consumption, such as spillage or use by others, thereby enhancing the reliability of hydration tracking. The following subsections explain how the detachable container base detects liquid volume changes and how the smartwatch identifies drinking events.

6.2.1 Liquid Volume Detector



(a) Functional prototype of the detachable base showcasing integrated components.



(b) Load Sensing Module circuit diagram depicting component connections.

Figure 6.2: Detachable base design for accurately tracking container liquid volume variations.

Figure 6.2a shows the detachable base equipped with key components: a load cell, an accelerometer, a Bluetooth module, and an Arduino powered by batteries. The circuit diagram in Figure 6.2b details how these components are interconnected. Our prototype features an Arduino Nano microcontroller chosen for its compact size, energy efficiency, and cost-effectiveness. The microcontroller's 8-bit processor, operating at 16MHz, effectively meets our project's needs. Its on-chip Analog-to-Digital Converter measures resistance changes in the load cell, while the digital accelerometer tracks any shifts in the bottle's position. For communication purposes, a serial Bluetooth module

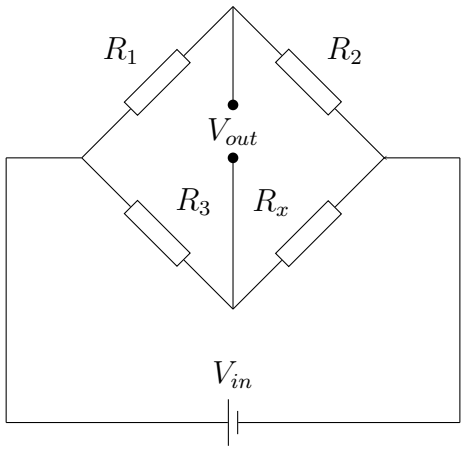


Figure 6.3: Schematic diagram of a Wheatstone Bridge Circuit, highlighting resistive arms R_1, R_2, R_3, and R_x, with V_{in} as the power supply and V_{out} as the output voltage

is integrated. Additionally, the accelerometer connects through the SPI bus for efficient data transmission.

The load cell in our system functions by measuring mechanical strain using the Wheatstone bridge principle applied to a strain gauge. A strain gauge, a small sensor affixed to the load cell, changes its electrical resistance in response to strain. When the load cell is under load, it slightly deforms, altering the strain gauge's resistance. The Wheatstone bridge circuit, depicted in Figure 6.3, consists of four resistors (R_1, R_2, R_3, and R_x), and measures this resistance change. R_x represents the strain gauge's resistance. When balanced, the Wheatstone bridge ensures a stable output voltage (V_{out}), which alters with any deformation in the load cell.

The electrical output from the load cell, typically in the millivolt range, is amplified for accuracy. The Arduino microcontroller processes this data to calculate weight changes. These changes are then transmitted to a mobile app via Bluetooth, enabling real-time monitoring of liquid intake.

In designing our system, we carefully selected the load cell's sensitivity, which is crucial for detecting the smallest measurable load change. We calibrated our system to detect a minimum change of 30 mL, aligning with the average sip volume identified in our research. This sensitivity level can be adjusted using various load cells and amplification techniques.

To address the limitation of the load cell in accurately measuring liquid volume when the bottle is in motion or tilted, we integrated the MPU-6050 accelerometer

module. The accelerometer detects changes in orientation and movement, identifying whether the bottle is stationary, upright, or tilted. If the MPU-6050 determines that the detachable base is not stationary or the bottle is tilted, any weight measurements from the load cell are disregarded. This ensures that only valid readings are reported to the smartphone app, maintaining reliable volume tracking when the bottle is upright and stationary on a flat surface.

Calibration of the Container: To utilize the load sensor effectively, we need to calibrate the detachable base for each container and liquid combination. We take two readings for calibration: when the bottle is empty and when it is completely filled with the particular liquid. By calculating the difference between these upper and lower readings and dividing it by the container's capacity, we obtain a constant α. We multiply the load sensor's subsequent readings by α, which yields the amount of liquid in the container. It is important to note that α varies for different container and liquid combinations. Once calibrated for a specific combination, the values can be shared among users.

6.2.2 Drinking Motion Detector

To detect the hand motion of drinking, we utilized tri-axial time series accelerometer data collected from a smartwatch. We adopt a time window-based segmentation approach on the accelerometer data with a sliding window having 50% overlap to extract features. Several window sizes were experimented with, and the most effective one was found to be 6 seconds. A set of 48 features was extracted from each segment, including statistical and frequency domain characteristics listed in Table 6.1, enabling a comprehensive representation of the accelerometer data.

Type	Features
Statistical	mean, median, standard deviation, mean absolute difference, interquartile range, maximum, minimum, correlation, kurtosis
Frequency	DC component, energy, information entropy

Table 6.1: Features extracted from accelerometer data for detecting the hand motion of drinking.

The accelerometer data features were used to train four distinct classifiers: Naive Bayes, Decision Tree, Support Vector Machine (SVM), and Random Forest. These

classifiers are commonly employed in human activity recognition tasks. Naive Bayes and Decision Tree are simple classifiers that offer computational efficiency and low cost. On the other hand, SVM and Random Forest are relatively more complex and computationally expensive due to their intricate algorithms. Utilizing this diverse set of classifiers allowed us to gain comprehensive insights into their respective strengths and limitations for the specific task of hand gesture recognition.

6.3 Experiment and Evaluation

In this section, we outline our experimental setup and the procedure employed to assess the efficiency of the proposed liquid intake tracking system. Subsequently, we provide an analysis of the results obtained.

6.3.1 Determining Amount of Liquid Consumed

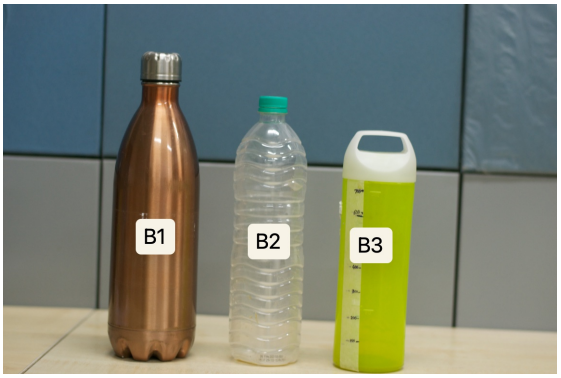


Figure 6.4: The containers used for Testing; **B1**: A Thermos, **B2**: A disposable plastic bottle and **B3**: Reusable Plastic Bottle.

Figure 6.5: Illustration for the testing process with Soda as the liquid being tested

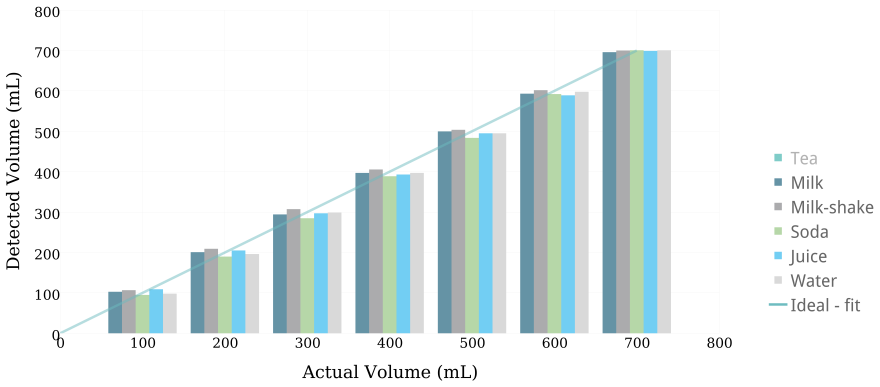


Figure 6.6: A Comparison of Detected Volume versus Actual Volume for All Liquids in a Disposable Bottle (B3).

Type of Container	Water	Milk	Tea	Juice	Soda	Thick Milk-shake
B1	99.98	99.52	98.19	98.16	97.78	99.02
B2	98.42	98.08	98.01	98.32	96.71	99.21
B3	99.20	99.04	97.45	97.84	97.15	97.98

Table 6.2: Accuracy of Volume Detection by Detachable Base for Different Types of Liquids.

We conducted a step-by-step analysis for each combination of container and liquid to assess the effectiveness of liquid volume detection. Initially, we calibrated the sensing module following the procedure outlined in subsection 6.2.1 to determine the multiplying factor (α). Subsequently, we filled the container with 700 mL of the liquid and successively removed 100 mL increments while recording the corresponding predicted volume at each step until the container was empty. An illustration for the same is shown in Figure 6.5.

Figure 6.6 compares the reusable plastic bottle's predicted volume and ground truth during the 100 mL reductions. Additionally, Table 6.2 presents the averaged accuracy for all combinations of container-liquid pairs. It is worth noting that the milkshake, which possesses higher viscosity among the six tested liquids, tends to be slightly over-predicted. In contrast, the soda tends to be slightly lower than its actual values. The accuracy values for each type of liquid-container combination fall within the range of approximately 96.71% to 99.98%. Overall, the system consistently demonstrates reliable performance across all containers and liquid compositions, exhibiting accuracy values above 96% for all cases. These results affirm the efficacy of the proposed system in accurately tracking liquid intake volume measurements.

Component	Current Draw (mA)
Arduino	25
Accelerometer and Gyroscope	28
Load cell	30
Bluetooth (pairing)	40
Bluetooth (transmitting data)	25
Bluetooth (idle)	3

Table 6.3: Power Consumption by Components within the Detachable Base.

Battery Optimization of the Detachable Base: For our detachable base module to be usable in real life, it should have sufficient battery life. To evaluate the power consumption, we connected the Arduino to each component and measured the current drawn by each circuit. The results are summarized in Table 6.3, indicating a total current draw ranging from 60 to 70 mA. Our prototype is equipped with a 500 mAh battery, providing approximately 8 hours of operation as a proof of concept. However, we can implement a power-saving strategy to optimize battery usage further. We can significantly reduce power consumption by programming the system to read data from sensors at regular intervals rather than continuously and establishing Bluetooth connections only when necessary for data transmission. For instance, employing a 15-minute interval for data readings would extend the system's runtime to approximately 20 hours. This optimization approach allows for longer operation times between battery charges, enhancing the practicality and usability of the detachable base system in real-world scenarios.

6.3.2 Detecting the Hand Gesture of Drinking

We collected data using an LG W100 smartwatch and a Moto G smartphone. The tri-axial accelerometer data was sampled at 186 Hz, gathered from the smartwatch worn on the dominant hand of each of the 11 volunteers (four male and seven female), whose ages ranged from 21 to 45. All volunteers were right-handed.

During the data collection, each volunteer was asked to drink water while working at their desk within a controlled laboratory setup. Alarms were set at 30-second intervals on the Moto G to prompt the volunteers to drink water from a mug. Between the alarms, the volunteers were allowed to perform their regular work, such as writing, browsing the internet, reading, or walking inside the room. Timestamps of the alarms were recorded as the ground truth for drinking and non-drinking events. We collected

a total of fifteen drinking instances per day for each user on three different days. Additionally, we collected an equivalent number of eating instances to demonstrate the distinguishability between these similar hand gestures. We obtained 45 drinking and 45 eating instances from each user. These account for 990 instances (495 drinking and 495 eating) collected from 11 users.

Based on the gathered data, we formed two types of datasets: (1) a user-dependent dataset, where the training and testing data belonged to the same individual, and (2) a user-independent dataset, which involved training and testing data combined from all 11 users without any user-based segregation. The reported accuracy of our approach is the average performance obtained from 10 iterations of 10-fold cross-validation.

Dataset	Comparison	NB	DT	SVM	RF
User-dependent	drink vs not drink	92.21	90.47	92.67	95.97
	drink vs eat	94.38	92.74	95.96	97.54
User-independent	drink vs not drink	87.78	87.97	89.91	93.53
	drink vs eat	87.05	88.14	90.53	92.38

Table 6.4: Classifier performance in drinking gesture recognition across user-dependent and user-independent datasets

Table 6.4 presents a comparative analysis of different machine learning classifiers across the two datasets. The classifiers compared include Decision Tree, Naive Bayes, Support Vector Machine, and Random Forest. For the User-dependent dataset, which distinguishes between drinking and not drinking activities, the Random Forest classifier shows the highest accuracy at 95.97%. The SVM classifier also performs well, with an accuracy of 92.67%. In comparing drinking versus eating activities within this dataset, the Random Forest classifier again leads with a remarkable accuracy of 97.54%, closely followed by SVM at 95.96%. In the User-independent dataset, which is more challenging due to the lack of personalization, the Random Forest classifier still outperforms others with 93.53% accuracy in distinguishing between drinking and not drinking, and 92.38% in differentiating drinking from eating activities. SVM also shows robust performance with 89.91% and 90.53% accuracy in these respective categories. Overall, the Random Forest classifier consistently achieves the highest accuracy across all categories, demonstrating its effectiveness in recognizing drinking gestures in both user-dependent and independent contexts.

6.3.3 Overall Liquid Intake Evaluation

Combining data from the detachable base and hand gesture recognition for drinking illustrates high efficacy. In an illustrative scenario where an individual consumes 25 ml per sip and completes 40 sips to ingest 1 liter of liquid, our user-dependent classifier exhibits an impressive accuracy of 95.97%, accurately discerning 38 out of the 40 sips. Concurrently, the precision of volume detection is noteworthy, standing at 98.34%. This level of accuracy allows the application to quantify each sip at 24.58 ml, slightly deviating from the theoretical value of 25 ml. Accumulatively, for the 38 sips identified with precision, the total calculated volume is 934 ml. Thus, in instances where 1 liter of liquid is consumed, the system, employing a user-dependent model, reports a consumption of 934 ml. In contrast, the user-independent classifier, also demonstrating robust performance, records a consumption volume of 910 ml for the equivalent intake of 1 liter, affirming the system's consistency and reliability across different user profiles.

6.3.4 Limitations and Future Work

The primary goal of this work is to track the volume of liquid consumed in a robust and user-friendly manner. This system is not intended to estimate total hydration levels. It should be considered a component of a broader hydration estimation system that accounts for factors like temperature, humidity, exertion, and personal attributes such as age, gender, weight, and height.

A fundamental limitation of our system is its dependence on the detachable base being placed on a stable surface for precise volume measurement. Consumption that occurs during container movement is logged only when the base is stabilized, which may introduce inaccuracies if the container is not returned to a stable surface.

Additionally, the current requirement of calibrating for each unique liquid-container combination could be streamlined. For example, QR codes on beverage cans or NFC tags on container bases could store and retrieve calibration values for frequent use. The system does not automatically identify the liquid type. Instead, users must manually select or calibrate for each combination. Since most individuals consume a limited variety of liquids and use a small number of containers, this limitation has minimal practical impact.

We acknowledge that the system may underestimate liquid intake. The extent of this underestimation has been quantified, providing users with a predictable margin of error. Finally, for our system to function correctly, we assume a non-adversarial use and rely on users genuinely aiming to track their consumption accurately.

To address these limitations, future work will explore the use of prior liquid consumption behavior to predict the volume consumed, potentially eliminating the need for a load base. This will involve analyzing parameters such as liquid temperature, container type, and the duration of drinking actions. Another area for improvement is taking a deep dive into optimizing the smartwatch accelerometer's sampling frequency. If we could reduce the accelerometer sampling rates and activate data sensing on the watch only upon detecting motion on the load base, we could potentially minimize unnecessary data collection, extend battery life, and enhance overall system usability.

6.4 Conclusion

We have introduced an automated liquid intake tracking solution that eliminates the need for recurrent user input. By leveraging accelerometer data from a smartwatch, our system detects the user's drinking gesture, while a load sensor-equipped detachable base accurately measures the volume of liquid consumed with each sip. In our experiment involving 11 users, our drinking motion detection achieved user-independent and user-dependent accuracies of 93.53% and 95.97%, respectively. Furthermore, our proposed approach demonstrated an average accuracy of 98.34% in measuring the quantity of consumed liquid. Notably, this system is cost-effective, with a construction cost of approximately $25, and it is adaptable for use with various types of containers and liquids. Additionally, it exhibits robustness against accidental spills and enables multiple users to share the container without compromising performance. Furthermore, we have extensively studied the current consumption of the detachable base, confirming its feasibility for real-world applications. By incorporating context information such as temperature, humidity, physical activity levels, gender, and age, our system can offer users highly accurate liquid intake tracking and personalized drinking recommendations.

CHAPTER 7

Future Research and Conclusion

7.1 Future Research

To further enhance the robustness and utility of our proposed system, we outline several key areas for future research.

7.1.1 Handwriting Micro-Event Detection

In our future work, we plan to explore finger-worn ring sensors that could capture more detailed finger movements during handwriting. Because the index finger plays such a critical role, wearing a ring sensor on this finger might help overcome the current challenge of distinguishing between shift and writing micro-events. This setup could also allow us to detect additional events like pauses and crossing out text, giving us a more thorough understanding of handwriting behavior. Nevertheless, practical considerations—such as user comfort, battery life, and device availability—must be addressed to ensure that these sensors can be adopted widely. Given the prevalence of smartwatches and their seamless fit into everyday routines, we also see potential in hybrid approaches that combine smartwatch data with ring sensors. For example, the smartwatch could signal the ring sensor to operate at a low sampling rate, which may help maintain accuracy while improving overall efficiency.

To build greater robustness into our model, we intend to include handwriting from various scripts and languages in future studies, particularly under single-subject conditions. This expanded scope could reveal how well the model adapts and performs across a wider array of handwriting styles. In this work, we focused on three types of tasks—copying, writing from memory, and spontaneous writing—to ensure a systematic and reproducible approach across participants. Future research, however, may involve more realistic scenarios, such as taking lecture notes, enabling us to better understand how handwriting micro-event detection systems operate in everyday contexts.

Moreover, it may be beneficial to investigate user-specific factors, including the choice of tagging thresholds and window sizes, to account for individual handwriting characteristics. A promising strategy would be to start with a model trained on a pooled dataset from all participants and then progressively fine-tune it with data from individual users. Such a "hybrid" model could combine the strengths of both aggregated and personalized approaches, potentially leading to a more adaptable and effective solution.

We also acknowledge the importance of examining how cognitive load affects handwriting micro-events—including writing, shifts, and newline transitions. In this work, we could not probe this aspect thoroughly because of the difficulty in reliably distinguishing *shift* events from *writing*. Addressing these methodological hurdles in future research could contribute to building a more comprehensive handwriting event detection framework.

Finally, continuous sampling of smartwatch motion sensors can drain the battery quickly, hindering long-term use. To mitigate this issue, an event-triggered strategy could be employed: the system would only be activated under specific, likely writing-related conditions. For instance, it might record data during scheduled class times or in designated areas like classrooms and offices, or when other sensors indicate that the user is seated and ready to write. Giving users the option to manually enable or disable the handwriting detection feature could further conserve battery life and offer greater control.

7.1.2 Indoor Parking and Unparking Detection

Determining the optimal number of Wi-Fi APs remains a key challenge for any indoor localization system, including BluePark. In complex indoor environments, such as multi-level or large garages, variations in visible APs and fluctuations in signal strength can significantly affect both detection accuracy and overall reliability. Consequently, investigating how these variations influence system performance—along with establishing guidelines for the strategic deployment of APs—warrants further study.

Another promising avenue involves probabilistic modeling to estimate parking spot availability in real-time. This capability would enable more efficient allocation of parking resources, as users could be guided to available spots based on both sensor data

and predictive algorithms. Integrating such a model into BluePark would offer a more holistic platform for monitoring, managing, and forecasting parking conditions.

Although the current study primarily targets cars, extending experiments to include two-wheeler vehicles could provide valuable insights into how smartphone placement (e.g., in backpacks or on vehicles themselves) and different movement patterns affect detection performance. In addition, diversifying test environments—spanning underground garages, open-sided structures, or multi-building complexes—would help validate the system's scalability and robustness under varied conditions. These expansions are integral to fully assessing BluePark's potential for widespread adoption across diverse indoor parking scenarios.

7.1.3 Liquid Intake Tracking

One important step toward improving the proposed liquid intake tracked system is to minimize the effort of manually calibrating each container–liquid combination. We could do this by integrating features like QR codes or NFC tags that automatically store and retrieve calibration details. This streamlining would not only make it more appealing to users who switch between different beverages or container types but also encourage them to keep using it consistently.

A second area of focus involves accommodating a wider range of drinking circumstances. Real-world usage often involves multiple people sharing a single container or individual users varying their drinking styles—such as sipping instead of gulping—and these scenarios can complicate tracking. Differentiating among various users and their drinking behaviors can help boost the system's accuracy in social settings.

Another promising route is to reduce strict dependence on the detachable base. By applying predictive modeling that harnesses the data collected when the detachable base is in use, the system could build individual consumption profiles and better estimate how much each sip holds. This would give users the freedom to rely on the detachable base only in convenient situations, making the overall setup more practical in dynamic environments.

Adding real-time feedback, such as notifications for inadequate or excessive fluid intake, would further enhance the system's utility—particularly if combined with a more

comprehensive hydration monitoring framework. Accounting for factors like physical activity, ambient temperature, or even body temperature would make these alerts much more tailored and useful, especially in clinical and fitness contexts.

Finally, optimizing the battery is critical to ensure the system's practicality over the long term. Continuous sampling can quickly drain a device's power supply. Implementing dynamic sampling rates or activating sensors only when the watch detects relevant drinking motions would make the system far more energy-efficient. Along with cutting down on frequent recharging, this would allow broader deployment scenarios, including large-scale studies and extended usage periods. The resulting gains in accessibility and convenience could greatly expand the impact and applicability of this liquid intake tracking system.

7.2 Unifying Theme of the Thesis

Over the course of three problems, this thesis presents a natural progression of techniques and approaches for HAR—from detecting drinking to parking and un-parking, and finally to handwriting micro-events. Each problem domain posed unique challenges that shaped the size of data windows, feature engineering strategies, and the choice of machine learning models, ensuring robust performance in each context. The complexity increases in the following order of the problems.

7.2.1 Liquid Intake Tracking

Here, the task was to monitor liquid intake using a smartwatch and a detachable base. To capture the average sip duration and keep latency low, we used relatively simple features derived from accelerometer data, with Random Forest proving effective. Because drinking gestures are fairly consistent, straightforward statistical and frequency-based features were sufficient to achieve solid accuracy.

7.2.2 Indoor Parking and Unparking Detection

This scenario added complexity by removing reliable GPS signals in indoor environments. We paired accelerometer data with WiFi signals to pinpoint user location and detect the transitions between walking and driving states. Adjusting the data window to cover these transitions was critical. We used adaptive DBSCAN, an unsupervised method, for real-time detection, while rule-based techniques helped refine the final results. Overall, this approach highlighted the value of blending multiple sensing modalities and dynamically sized data windows for handling unpredictable indoor conditions.

7.2.3 Handwriting Micro-Event Recognition

This was the most intricate domain, requiring a more granular analysis of activities like writing, shifting, and starting new lines. Because these micro-events vary in duration, we needed smaller windows for short events and bigger windows for longer tasks. To manage this balance, we introduced a Thresholding Tag Assignment approach. While Random Forest and other feature-engineered models performed well, we also explored time-series deep learning models such as LSTMs. Although deep learning showed promise for learning intricate temporal patterns, it demanded larger datasets and faced challenges with class imbalance compared to feature-based methods.

Overall, the journey moves from simpler statistical methods for drinking detection to more complex hybrid multi-sensor and unsupervised approaches for parking, and finally, to a mix of human-engineered features and deep learning for handwriting. This progression underscores the increasing complexity in feature spaces and modeling techniques for HAR, driven by the granularity and variety of events in each domain.

7.3 Conclusion

In the course of this thesis, we have embarked on an intricate journey through the multifaceted realm of HAR, meticulously examining the interplay between sensor deployment, body motion, and application domains. This exploration has been underpinned by three parallel themes that have served as the guiding pillars of our research.

The first theme, sensor deployment, has been manifested in three distinct forms: environmental, object-embedded, and wearable. Each form has offered unique insights into HAR systems' potential applications and limitations. As demonstrated by the BluePark system, environmental sensors have proven to be robust in monitoring large-scale activities, achieving over 20% higher accuracy in drive detection compared to the Google Activity Recognition API. As illustrated by the automated liquid intake tracking solution, object-embedded sensors have shown their precision in tracking specific activities, providing detailed measurements of liquid consumption with an average accuracy of 98.34%. Wearable sensors, as employed in the handwriting activity tracking system, have demonstrated their capacity to capture nuanced, individual-specific data, offering a granular view of human activity with user-specific models reaching an F1 score of 0.84.

The second theme, body motion, has been explored through the lens of full-body, free-arm, and restricted-arm movements. Each type of motion has provided a different perspective on human activity. Full body movement, as observed in the parking and unparking events, has shown how large-scale motions can be effectively monitored and managed, with our state-detection algorithm achieving 100% precision and recall at a 15Hz sampling frequency. Free arm movement, as seen in the liquid intake tracking, has demonstrated the potential of HAR in tracking everyday activities, achieving user-dependent and independent accuracies of 95.97% and 93.53%, respectively. Restricted arm movement, as examined in handwriting activity tracking, has revealed the subtleties of fine-grained activities and the challenges associated with their recognition, where our approach led to a significant increase in F1 score by 0.26.

The third theme, application domains, has encompassed transport, education, and healthcare. Each domain has presented unique challenges and opportunities for HAR. In the transport domain, we have seen how HAR can contribute to efficient parking management, with our algorithm demonstrating a low detection latency of 20 seconds with a probability of 0.9 and good parking localization accuracy of 10 meters. In the education domain, we have explored the potential of HAR in providing detailed insights into the writing process, with our system detecting three writing micro-events: 'write', 'shift', and 'newline'. In the healthcare domain, we have demonstrated how HAR can automate health monitoring tasks, reducing the burden on individuals and potentially improving health outcomes, as evidenced by the high accuracy and cost-effectiveness

of our liquid intake tracking system. The user-dependent model calculates 934 ml out of a theoretical 1 liter, while the user-independent model estimates 910 ml, showcasing the system's reliability in both personalized and general use cases.

In conclusion, this thesis has conducted an in-depth exploration of the various aspects of HAR, demonstrating the versatility and adaptability of this field. The interplay between sensor deployment, body motion, and application domains has shaped the design and effectiveness of HAR systems, offering valuable insights for future research. As we move forward, it is evident that the continued exploration of these themes will be instrumental in pushing the boundaries of HAR, enhancing our understanding of human behavior and our capacity to improve it.

APPENDIX A

APPENDIX

A.1 Horus WiFi Localization

The Horus WLAN Location Determination System is a wireless localization system [173] that leverages signal strength information from access points within a WLAN (Wireless Local Area Network) to infer a user's location. The system operates in two distinct phases: the offline phase and the online phase.

In the offline phase, the system constructs a comprehensive radio map of the area of interest. This map tabulates the signal strength received from access points at selected locations. The signal strength distributions from the access points are stored in the radio map. The online determination phase involves several steps for estimating the user's location based on signal strength measurements. Firstly, the mobile client measures the signal strength of nearby access points (APs) by scanning and recording the received signal strength indicator (RSSI) values. Next, these measured values are compared with the signal strength values saved in the radio map to find the closest match. The Horus system employs a K-nearest approach, selecting the K-closest signal strength values from the radio map. Once the K nearest values are identified, the system estimates the user's location using pattern matching (PM) or triangulation, mapping, and interpolation (TMI). In the PM approach, the system compares the measured and stored signal strength values to identify patterns and estimate the user's location. In the TMI technique, the physical positions of APs are known, and a function maps signal strength to distances. Interpolation of training data allows the algorithm to generate contours and determine the user's position. Finally, the system provides the estimated location output, which can be in the form of coordinates or a location identifier, indicating the user's position within the area of interest.

The Horus system distinguishes between temporal and spatial variations to address challenges posed by wireless channel variations, employing different techniques for each. For temporal variations, the system utilizes an autoregressive model to capture the correlation between consecutive samples, resulting in more precise location estimates. Spatial variations are handled through two approaches: treating radio map locations as objects in physical space, weighted by normalized probabilities, and using a time-average window to smooth location estimates.

To handle small-scale variations, the Horus system incorporates a technique known

as Perturbation. It detects small-scale variations by comparing the estimated location with the previous user location. If the distance exceeds a threshold, small-scale variations are considered. The system compensates for these variations using techniques such as the Center of Mass and Time-Averaging. These techniques improve accuracy by considering weighted averages and averaging signal strength values over time.

In summary, the Horus system is designed to provide high accuracy and low computational requirements for WLAN location determination. It addresses the different causes of wireless channel variations and uses various techniques to estimate the user's location in both discrete and continuous space.

REFERENCES

[1] **Ahmed, N.**, **R. Banerjee**, **A. Ghose**, and **A. Sinharay**, Feasibility analysis for estimation of blood pressure and heart rate using a smart eye wear. *In Proceedings of the 2015 Workshop on Wearable Systems and Applications*, WearSys '15. Association for Computing Machinery, New York, NY, USA, 2015. ISBN 9781450335003. URL https://doi.org/10.1145/2753509.2753511.

[2] **Alaoui, F. T.**, **H. Fourati**, **A. Kibangou**, **B. Robu**, and **N. Vuillerme** (2022). Urban transportation mode detection from inertial and barometric data in pedestrian mobility. *IEEE Sensors Journal*, **22**(6), 4772–4780.

[3] **Alazrai, R.**, **M. Hababeh**, **B. A. Alsaify**, **M. Z. Ali**, and **M. I. Daoud** (2020). An end-to-end deep learning framework for recognizing human-to-human interactions using wi-fi signals. *IEEE Access*, **8**, 197695–197710. ISSN 2169-3536.

[4] **Altini, M.** and **H. Kinnunen** (2021). The promise of sleep: A multi-sensor approach for accurate sleep stage detection using the oura ring. *Sensors*, **21**(13). ISSN 1424-8220. URL https://www.mdpi.com/1424-8220/21/13/4302.

[5] **Amft, O.**, **D. Bannach**, **G. Pirkl**, **M. Kreil**, and **P. Lukowicz**, Towards wearable sensing-based assessment of fluid intake. *In 2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. IEEE, 2010. ISBN 978-1-4244-6605-4. URL http://ieeexplore.ieee.org/document/5470653/.

[6] **Amft, O.**, **D. Bannach**, **G. Pirkl**, **M. Kreil**, and **P. Lukowicz**, Towards wearable sensing-based assessment of fluid intake. *In 2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. IEEE, 2010.

[7] **Amma, C.**, **M. Georgi**, and **T. Schultz** (2014). Airwriting: A wearable handwriting recognition system. *Personal Ubiquitous Comput.*, **18**(1), 191–203. ISSN 1617-4909.

[8] **Anand, A.**, **M. Sharma**, **R. Srivastava**, **L. Kaligounder**, and **D. Prakash**, Wearable motion sensor based analysis of swing sports. *In 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 2017.

[9] **Anguita, D.**, **A. Ghio**, **L. Oneto**, **X. Parra**, and **J. L. Reyes-Ortiz**, Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. *In* **J. Bravo**, **R. Hervás**, and **M. Rodríguez** (eds.), *Ambient Assisted Living and Home Care*, Lecture Notes in Computer Science. Springer, 2012.

[10] **Ardüser, L.**, **P. Bissig**, **P. Brandes**, and **R. Wattenhofer**, Recognizing text using motion data from a smartwatch. *In Pervasive Computing and Communication Workshops (PerCom Workshops), 2016 IEEE International Conference on*. IEEE, 2016.

[11] **Arista, F. S.** and **H. Kuswanto** (2018). Virtual physics laboratory application based on the android smartphone to improve learning independence and conceptual understanding. *International Journal of Instruction*, **11**(1), 1–16.

[12] **Avilés-Cruz, C.** and **J. Villegas-Cortez** (2019). A smartphone-based augmented reality system for university students for learning digital electronics. *Computer Applications in Engineering Education*, **27**(3), 615–630.

[13] **Azadi, B.**, **M. Haslgrübler**, **B. Anzengruber-Tanase**, **S. Grünberger**, and **A. Ferscha** (2022). Alpine skiing activity recognition using smartphone's imus. *Sensors*, **22**(15), 5922. ISSN 1424-8220. URL https://www.mdpi.com/1424-8220/22/15/5922.

[14] **Baca, A.**, **P. Dabnichki**, **M. Heller**, and **P. Kornfeind** (2009). Ubiquitous computing in sports: A review and analysis. *Journal of Sports Sciences*, **27**(12), 1335–1346. ISSN 0264-0414. URL https://doi.org/10.1080/02640410903277427.

[15] **Badarna, M.**, **I. Shimshoni**, **G. Luria**, and **S. Rosenblum** (2017). The importance of pen motion pattern groups for semi-automatic classification of handwriting into mental workload classes. *Cognitive Computation*. ISSN 1866-9964.

[16] **Bao, L.** and **S. S. Intille**, Activity recognition from user-annotated acceleration data. *In* **A. Ferscha** and **F. Mattern** (eds.), *Pervasive Computing*, Lecture Notes in Computer Science. Springer, 2004. ISBN 978-3-540-24646-6.

[17] **Bavan, L.**, **K. Surmacz**, **D. Beard**, **S. Mellon**, and **J. Rees** (2019). Adherence monitoring of rehabilitation exercise with inertial sensors: A clinical validation study. *Gait & Posture*, **70**, 211–217. ISSN 1879-2219.

[18] **Behboodi, A.**, **N. Zahradka**, **H. Wright**, **J. Alesi**, and **S. C. K. Lee** (2019). Real-time detection of seven phases of gait in children with cerebral palsy using two gyroscopes. *Sensors*, **19**(11), 2517. ISSN 1424-8220. URL https://www.mdpi.com/1424-8220/19/11/2517. Number: 11 Publisher: Multidisciplinary Digital Publishing Institute.

[19] **Bi, C.**, **J. Huang**, **G. Xing**, **L. Jiang**, **X. Liu**, and **M. Chen**, Safewatch: A wearable hand motion tracking system for improving driving safety. *In Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*, IoTDI '17. Association for Computing Machinery, 2017. ISBN 978-1-4503-4966-6. URL https://dl.acm.org/doi/10.1145/3054977.3054979.

[20] **Bi, C.**, **J. Huang**, **G. Xing**, **L. Jiang**, **X. Liu**, and **M. Chen** (2019). Safewatch: A wearable hand motion tracking system for improving driving safety. *ACM Transactions on Cyber-Physical Systems*, **4**(1), 13:1–13:21. ISSN 2378-962X. URL https://doi.org/10.1145/3360323.

[21] **Bleda, A. L.**, **F. J. Fernández-Luque**, **A. Rosa**, **J. Zapata**, and **R. Maestre** (2017). Smart sensory furniture based on wsn for ambient assisted living. *IEEE Sensors Journal*, **17**(17), 5626–5636.

[22] **Bouten, C. V.**, **K. T. Koekkoek**, **M. Verduin**, **R. Kodde**, and **J. D. Janssen** (1997). A triaxial accelerometer and portable data processing unit for the assessment of daily physical activity. *IEEE transactions on bio-medical engineering*, **44**(3), 136–147. ISSN 0018-9294.

[23] **Bulling, A.**, **U. Blanke**, and **B. Schiele** (2014). A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys*, **46**(3), 1–33. ISSN 0360-0300, 1557-7341. URL https://dl.acm.org/doi/10.1145/2499621.

[24] **Cavazza, M.**, **O. Martin**, **F. Charles**, **S. J. Mead**, and **X. Marichal**, Interacting with virtual agents in mixed reality interactive storytelling. *In* **T. Rist**, **R. S. Aylett**, **D. Ballin**, and **J. Rickel** (eds.), *Intelligent Virtual Agents*, volume 2792. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-20003-1 978-3-540-39396-2, 231–235. URL http://link.springer.com/10.1007/978-3-540-39396-2_39.

[25] **Chang, L.**, **J. Lu**, **J. Wang**, **X. Chen**, **D. Fang**, **Z. Tang**, **P. Nurmi**, and **Z. Wang** (2018). Sleepguard: Capturing rich sleep information using smartwatch sensing data. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, **2**(3). URL https://doi.org/10.1145/3264908.

[26] **Chaquet, J. M.**, **E. J. Carmona**, and **A. Fernández-Caballero** (2013). A survey of video datasets for human action and activity recognition. *Computer Vision and Image Understanding*, **117**(6), 633–659. ISSN 1077-3142. URL https://www.sciencedirect.com/science/article/pii/S1077314213000295.

[27] **Chatzaki, C.**, **V. Skaramagkas**, **N. Tachos**, **G. Christodoulakis**, **E. Maniadi**, **Z. Kefalopoulou**, **D. I. Fotiadis**, and **M. Tsiknakis** (2021). The smart-insole dataset: Gait analysis using wearable sensors with a focus on elderly and parkinson's patients. *Sensors*, **21**(8), 2821. ISSN 1424-8220. URL https://www.mdpi.com/1424-8220/21/8/2821.

[28] **Chen, C.-H.**, **C.-C. Wang**, and **Y.-Z. Chen** (2021). Intelligent brushing monitoring using a smart toothbrush with recurrent probabilistic neural network. *Sensors*, **21**(4), 1238. ISSN 1424-8220. URL https://www.mdpi.com/1424-8220/21/4/1238. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute.

[29] **Chen, C.-H.**, **C.-C. Wang**, and **Y.-Z. Chen** (2021). Intelligent brushing monitoring using a smart toothbrush with recurrent probabilistic neural network. *Sensors*, **21**(4), 1238. ISSN 1424-8220. URL https://www.mdpi.com/1424-8220/21/4/1238.

[30] **Chiu, M.-C.**, **S.-P. Chang**, **Y.-C. Chang**, **H.-H. Chu**, **C. C.-H. Chen**, **F.-H. Hsiao**, and **J.-C. Ko**, Playful bottle: A mobile social persuasion system to motivate healthy water intake. *In Proceedings of the 11th International Conference*

on Ubiquitous Computing, UbiComp '09. Association for Computing Machinery, 2009. ISBN 978-1-60558-431-7. URL https://doi.org/10.1145/1620545.1620574.

[31] **Choudhury, R. R.**, Earable computing: A new area to think about. *In Proceedings of the 22nd International Workshop on Mobile Computing Systems and Applications*, HotMobile '21. Association for Computing Machinery, New York, NY, USA, 2021. ISBN 9781450383233. URL https://doi.org/10.1145/3446382.3450216.

[32] **Cole, C. A.**, **S. Powers**, **R. L. Tomko**, **B. Froeliger**, and **H. Valafar** (2021). Quantification of smoking characteristics using smartwatch technology: Pilot feasibility study of new technology. *JMIR Formative Research*, **5**(2). ISSN 2561-326X. URL https://formative.jmir.org/2021/2/e20464.

[33] **Constant, N.**, **O. Douglas-Prawl**, **S. Johnson**, and **K. Mankodiya**, Pulseglasses: An unobtrusive, wearable hr monitor with internet-of-things functionality. *In 2015 IEEE 12th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*. 2015.

[34] **Csizmadia, G.**, **K. Liszkai-Peres**, **B. Ferdinandy**, **A. Miklosi**, and **V. Konok** (2022). Human activity recognition of children with wearable devices using lightgbm machine learning. *Scientific Reports*, **12**(1), 5472. ISSN 2045-2322. URL https://www.nature.com/articles/s41598-022-09521-1.

[35] **Deselaers, T.**, **D. Keysers**, **J. Hosang**, and **H. A. Rowley** (2015). Gyropen: Gyroscopes for pen-input with mobile phones. *IEEE Transactions on Human-Machine Systems*, **45**(2), 263–271.

[36] **Dhar, A.**, **A. Nittala**, and **K. Yadav**, Tactback: Vibrotactile braille output using smartphone and smartwatch for visually impaired. *In Proceedings of the 13th International Web for All Conference*, W4A '16. Association for Computing Machinery, New York, NY, USA, 2016. ISBN 9781450341387. URL https://doi.org/10.1145/2899475.2899514.

[37] **Dong, D.** and **Z. Li** (2021). Smartphone sensing of road surface condition and defect detection. *Sensors*, **21**(16), 5433. ISSN 1424-8220. URL https://www.mdpi.com/1424-8220/21/16/5433.

[38] **Dong, Y.**, **J. Scisco**, **M. Wilson**, **E. Muth**, and **A. Hoover** (2014). Detecting periods of eating during free-living by tracking wrist motion. *Biomedical and Health Informatics, IEEE Journal of*, **18**(4), 1253–1260. ISSN 2168-2194.

[39] **Doukas, C.** and **I. Maglogiannis**, Advanced patient or elder fall detection based on movement and sound data. *In 2008 Second International Conference on Pervasive Computing Technologies for Healthcare*. 2008. ISSN 2153-1641.

[40] **Dudukcu, H. V.**, **M. Taskiran**, and **M. Z. Dudukcu**, Physiotherapy activities classification with deep neural networks. *In 2022 Innovations in Intelligent Systems and Applications Conference (ASYU)*. 2022.

[41] **Dunlop, M. D.**, **M. Roper**, **M. Elliot**, **R. McCartan**, and **B. McGregor**, Using smartphones in cities to crowdsource dangerous road sections and give effective in-car warnings. *In Proceedings of the SEACHI 2016 on Smart Cities for Better Living with HCI and UX*, SEACHI 2016. Association for Computing Machinery, 2016. ISBN 978-1-4503-4194-3. URL https://doi.org/10.1145/2898365.2899796.

[42] **Epps, J.** and **S. Chen** (2018). Automatic task analysis: Toward wearable behaviometrics. *IEEE Systems, Man, and Cybernetics Magazine*, **4**(4), 15–20.

[43] **Ester, M.**, **H.-P. Kriegel**, **J. Sander**, and **X. Xu**, A density-based algorithm for discovering clusters in large spatial databases with noise. *In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96. AAAI Press, 1996.

[44] **Fang, S.-H.**, **Y.-X. Fei**, **Z. Xu**, and **Y. Tsao** (2017). Learning transportation modes from smartphone sensors based on deep neural network. *IEEE Sensors Journal*, **17**(18), 6111–6118.

[45] **Faurholt-Jepsen, M.**, **M. Vinberg**, **M. Frost**, **E. M. Christensen**, **J. E. Bardram**, and **L. V. Kessing** (2015). Smartphone data as an electronic biomarker of illness activity in bipolar disorder. *Bipolar Disorders*, **17**(7), 715–728. ISSN 1399-5618.

[46] **Foerster, F.** and **J. Fahrenberg** (2000). Motion pattern and posture: Correctly assessed by calibrated accelerometers. *Behavior Research Methods, Instruments, & Computers*, **32**(3), 450–457. ISSN 1532-5970. URL https://doi.org/10.3758/BF03200815.

[47] **Foerster, F.**, **M. Smeja**, and **J. Fahrenberg** (1999). Detection of posture and motion by accelerometry: A validation study in ambulatory monitoring. *Computers in Human Behavior*, **15**(5), 571–583. ISSN 0747-5632. URL https://www.sciencedirect.com/science/article/pii/S0747563299000370.

[48] **Garcia-Ceja, E.**, **V. Osmani**, and **O. Mayora** (2016). Automatic stress detection in working environments from smartphones' accelerometer data: A first step. *IEEE journal of biomedical and health informatics*, **20**(4), 1053–1060. ISSN 2168-2208.

[49] **Gavrilescu, M.** and **N. Vizireanu** (2018). Predicting the big five personality traits from handwriting. *EURASIP Journal on Image and Video Processing*, **2018**.

[50] **Gerina, F.**, **S. M. Massa**, **F. Moi**, **D. Reforgiato Recupero**, and **D. Riboni** (2020). Recognition of cooking activities through air quality sensor data for supporting food journaling. *Human-centric Computing and Information Sciences*, **10**(1), 1–26.

[51] **González-Landero, F.**, **I. García-Magariño**, **R. Amariglio**, and **R. Lacuesta** (2019). Smart cupboard for assessing memory in home environment. *Sensors*, **19**(11). ISSN 1424-8220. URL https://www.mdpi.com/1424-8220/19/11/2552.

[52] **Google** (). Google openspot. Website. http://techcrunch.com/2010/07/09/google-parking-open-spot/ Accessed: 2015-02-12.

[53] **Google** (2014). Recognizing the user's current activity. http://goo.gl/5fm314 Accessed : 2014-08-16.

[54] **Griswold-Steiner, I.**, **R. Matovu**, and **A. Serwadda** (2019). Wearables-driven freeform handwriting authentication. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, **1**(3), 152–164.

[55] **Gu, W.**, **Z. Yang**, **L. Shangguan**, **W. Sun**, **K. Jin**, and **Y. Liu**, Intelligent sleep stage mining service with smartphones. *In Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '14. Association for Computing Machinery, New York, NY, USA, 2014. ISBN 9781450329682. URL https://doi.org/10.1145/2632048.2632084.

[56] **Guo, X.**, **J. Liu**, and **Y. Chen**, Fitcoach: Virtual fitness coach empowered by wearable mobile devices. *In IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*. 2017.

[57] **Gupta, A.** and **H. P. Gupta** (2021). Yogahelp: Leveraging motion sensors for learning correct execution of yoga with feedback. *IEEE Transactions on Artificial Intelligence*, **2**(4), 362–371. ISSN 2691-4581.

[58] **Gupta, A.**, **C. Ji**, **H.-S. Yeo**, **A. Quigley**, and **D. Vogel**, Rotoswype: Word-gesture typing using a ring. *In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19. Association for Computing Machinery, 2019. URL https://doi.org/10.1145/3290605.3300244.

[59] **Hamatani, T.**, **M. Elhamshary**, **A. Uchiyama**, and **T. Higashino** (2018). Fluidmeter: Gauging the human daily fluid intake using smartwatches. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, **2**(3), 113:1–113:25. URL https://doi.org/10.1145/3264923.

[60] **Hao, T.**, **G. Xing**, and **G. Zhou**, Isleep: Unobtrusive sleep quality monitoring using smartphones. *In Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13. Association for Computing Machinery, New York, NY, USA, 2013. ISBN 9781450320276. URL https://doi.org/10.1145/2517351.2517359.

[61] **Heinonen, H.**, **S. Siltanen**, and **P. Ahola** (2021). Information design for small screens: Toward smart glass use in guidance for industrial maintenance. *IEEE Transactions on Professional Communication*, **64**(4), 407–426.

[62] **Hemminki, S.**, **P. Nurmi**, and **S. Tarkoma**, Accelerometer-based transportation mode detection on smartphones. *In Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13. Association for Computing Machinery, 2013. URL https://doi.org/10.1145/2517351.2517367.

[63] **Hochreiter, S.** and **J. Schmidhuber** (1997). Long short-term memory. *Neural Computation*, **9**(8), 1735–1780. ISSN 0899-7667. URL https://doi.org/10.1162/neco.1997.9.8.1735.

[64] **Hoog, T. G.**, **L. M. Aufdembrink**, **N. J. Gaut**, **R.-J. Sung**, **K. P. Adamala**, and **A. E. Engelhart** (2020). Rapid deployment of smartphone-based augmented reality tools for field and online education in structural biology. *Biochemistry and Molecular Biology Education*, **48**(5), 448–451.

[65] **Hossain, T.**, **M. S. Islam**, **M. A. R. Ahad**, and **S. Inoue**, Human activity recognition using earable device. *In Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, UbiComp/ISWC '19 Adjunct. Association for Computing Machinery, New York, NY, USA, 2019. ISBN 9781450368698. URL https://doi.org/10.1145/3341162.3343822.

[66] **Hou, J.**, **X.-Y. Li**, **P. Zhu**, **Z. Wang**, **Y. Wang**, **J. Qian**, and **P. Yang**, Signspeaker: A real-time, high-precision smartwatch-based sign language translator. *In The 25th Annual International Conference on Mobile Computing and Networking*, MobiCom '19. Association for Computing Machinery, 2019. ISBN 978-1-4503-6169-9. URL https://dl.acm.org/doi/10.1145/3300061.3300117.

[67] **Hu, S.**, **L. Su**, **H. Liu**, **H. Wang**, and **T. F. Abdelzaher** (2015). Smartroad: Smartphone-based crowd sensing for traffic regulator detection and identification. *ACM Trans. Sen. Netw.*, **11**(4). ISSN 1550-4859. URL https://doi.org/10.1145/2770876.

[68] **Huang, H.**, **H. Chen**, and **S. Lin**, Magtrack: Enabling safe driving monitoring with wearable magnetics. *In Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '19. Association for Computing Machinery, 2019. ISBN 978-1-4503-6661-8. URL https://dl.acm.org/doi/10.1145/3307334.3326107.

[69] **Hussain, Z.**, **M. Sheng**, and **W. E. Zhang** (2020). Different approaches for human activity recognition: A survey. *Journal of Network and Computer Applications*, **167**, 102738. ISSN 10848045. URL http://arxiv.org/abs/1906.05074.

[70] **Huynh, T.**, **M. Fritz**, and **B. Schiele**, Discovery of activity patterns using topic models. *In Proceedings of the 10th International Conference on Ubiquitous Computing*, UbiComp '08. Association for Computing Machinery, 2008. URL https://doi.org/10.1145/1409635.1409638.

[71] **Huynh, T.** and **B. Schiele**, Towards less supervision in activity recognition from wearable sensors. *In 2006 10th IEEE International Symposium on Wearable Computers*. 2006.

[72] **Hwang, A.** and **E. Peli** (2014). Augmented edge enhancement on google glass for vision-impaired users. *Information Display*, **30**, 16–19.

[73] **Hwang, I.**, **H. N. Kim**, **M. Seong**, **S.-H. Lee**, **M. Kang**, **H. Yi**, **W. G. Bae**, **M. K. Kwak**, and **H. E. Jeong** (2018). Multifunctional smart skin adhesive patches for advanced health care. *Advanced healthcare materials*, **7**(15), 1800275.

[74] **James, K. H.** and **L. Engelhardt** (2012). The effects of handwriting experience on functional brain development in pre-literate children. *Trends in neuroscience and education*, **1**(1), 32–42. ISSN 2211-9493. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4274624/.

[75] **Jung, I.** (2020). A review of privacy-preserving human and human activity recognition. *International Journal on Smart Sensing and Intelligent Systems*, **13**, 1–13.

[76] **Khamis, A.**, **B. Kusy**, **C. T. Chou**, **M.-L. McLaws**, and **W. Hu**, Rfwash: A weakly supervised tracking of hand hygiene technique. *In Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, SenSys '20. Association for Computing Machinery, 2020. ISBN 978-1-4503-7590-0. URL https://doi.org/10.1145/3384419.3430733.

[77] **Kokolaki, E.**, **G. Kollias**, **M. Papadaki**, **M. Karaliopoulos**, and **I. Stavrakakis** (2013). Opportunistically-assisted parking search: A story of free riders, selfish liars and bona fide mules. *2013 10th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, 17–24. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6578315.

[78] **Kremser, W.**, **S. Kranzinger**, and **S. Bernhart** (2021). Design and implementation of a gesture-aided e-learning platform. *Sensors*, **21**(23), 8042. ISSN 1424-8220. URL https://www.mdpi.com/1424-8220/21/23/8042.

[79] **Kwapisz, J. R.**, **G. M. Weiss**, and **S. A. Moore** (2011). Activity recognition using cell phone accelerometers. *ACM SIGKDD Explorations Newsletter*, **12**(2), 74–82. ISSN 1931-0145, 1931-0153. URL https://dl.acm.org/doi/10.1145/1964897.1964918.

[80] **Lan, K.-C.** and **W.-Y. Shih** (2014). An intelligent driver location system for smart parking. *Expert Systems with Applications*, **41**(5), 2443–2456. ISSN 09574174. URL https://linkinghub.elsevier.com/retrieve/pii/S0957417413007987.

[81] **Laput, G.**, **R. Xiao**, and **C. Harrison**, Viband: High-fidelity bio-acoustic sensing using commodity smartwatch accelerometers. *In Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM, 2016. ISBN 978-1-4503-4189-9. URL https://dl.acm.org/doi/10.1145/2984511.2984582.

[82] **Lee, S.**, **D. Yoon**, and **A. Ghosh**, Intelligent parking lot application using wireless sensor networks. *In 2008 International Symposium on Collaborative Technologies and Systems*. 2008.

[83] **Levy, A.**, **B. Nassi**, **Y. Elovici**, and **E. Shmueli** (2018). Handwritten signature verification using wrist-worn devices. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, **2**(3).

[84] **Li, H.**, **C. Ye**, and **A. P. Sample**, Idsense: A human object interaction detection system based on passive uhf rfid. *In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15. Association for Computing Machinery, New York, NY, USA, 2015. ISBN 9781450331456. URL https://doi.org/10.1145/2702123.2702178.

[85] **Li, R. T.**, **S. R. Kling**, **M. J. Salata**, **S. A. Cupp**, **J. Sheehan**, and **J. E. Voos** (2016). Wearable performance devices in sports medicine. *Sports health*, **8**(1), 74–78.

[86] **Li, Y.**, **K. Yao**, and **G. Zweig**, Feedback-based handwriting recognition from inertial sensor data for wearable devices. *In 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2015. ISSN 1520-6149.

[87] **Likforman-Sulem, L.**, **A. Esposito**, **M. Faundez-Zanuy**, **S. Clemencon**, and **G. Cordasco** (2017). Emothaw: A novel database for emotional state recognition from handwriting and drawing. *IEEE Transactions on Human-Machine Systems*, **47**(2), 273–284. ISSN 2168-2291, 2168-2305. URL http://ieeexplore.ieee.org/document/7807324/.

[88] **Lin, F.**, **A. Wang**, **Y. Zhuang**, **M. Tomita**, and **W. Xu** (2016). Smart insole: A wearable sensor device for unobtrusive gait monitoring in daily life. *IEEE Transactions on Industrial Informatics*, **12**, 1551–3203.

[89] **Lin, X.**, **Y. Chen**, **X.-W. Chang**, **X. Liu**, and **X. Wang** (2018). Show: Smart handwriting on watches. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, **1**(4), 151:1–151:23. ISSN 2474-9567.

[90] **Liu, C.-F.** and **P.-Y. Chiang**, Smart glasses based intelligent trainer for factory new recruits. *In Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, MobileHCI '18. Association for Computing Machinery, New York, NY, USA, 2018. ISBN 9781450359412. URL https://doi.org/10.1145/3236112.3236174.

[91] **Liu, Y.**, **M. Pharr**, and **G. A. Salvatore** (2017). Lab-on-skin: a review of flexible and stretchable electronics for wearable health monitoring. *ACS nano*, **11**(10), 9614–9635.

[92] **Longcamp, M.**, **M.-T. Zerbato-Poudou**, and **J.-L. Velay** (2005). The influence of writing practice on letter recognition in preschool children: A comparison between handwriting and typing. *Acta Psychologica*, **119**(1), 67–79. ISSN 0001-6918.

[93] **Lopez, G.**, **S. Abe**, **K. Hashimoto**, and **A. Yokokubo**, On-site personal sport skill improvement support using only a smartwatch. *In 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. 2019.

[94] **Luna, M. M.**, **F. A. A. de M. N. Soares**, **H. A. D. Nascimento**, and **A. Quigley**, Braille text entry on smartwatches: An evaluation of methods for composing the braille cell. *In Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '19. Association for Computing Machinery, New York, NY, USA, 2019. ISBN 9781450367455. URL https://doi.org/10.1145/3319499.3328233.

[95] **Luo, C.**, **X. Feng**, **J. Chen**, **J. Li**, **W. Xu**, **W. Li**, **L. Zhang**, **Z. Tari**, and **A. Y. Zomaya**, Brush like a dentist: Accurate monitoring of toothbrushing via wrist-worn gesture sensing. *In IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. 2019. ISSN: 2641-9874.

[96] **Ma, Y.**, **S. Arshad**, **S. Muniraju**, **E. Torkildson**, **E. Rantala**, **K. Doppler**, and **G. Zhou** (2021). Location- and person-independent activity recognition with wifi, deep neural networks, and reinforcement learning. *ACM Transactions on Internet of Things*, **2**(1), 3:1–3:25. ISSN 2691-1914. URL https://doi.org/10.1145/3424739.

[97] **Maldarelli, J. E.**, **B. A. Kahrs**, **S. C. Hunt**, and **J. J. Lockman** (2015). Development of early handwriting: Visual-motor control during letter copying. *Developmental Psychology*, **51**(7), 879–888. ISSN 1939-0599.

[98] **Mandal, R.**, **P. Karmakar**, **S. Chatterjee**, **D. Das Spandan**, **S. Pradhan**, **S. Saha**, **S. Chakraborty**, and **S. Nandi** (2023). Exploiting multi-modal contextual sensing for city-bus's stay location characterization: Towards sub-60 seconds accurate arrival time prediction. *ACM Trans. Internet Things*, **4**(1). URL https://doi.org/10.1145/3549548.

[99] **Mathur, S.**, **S. Kaul**, **M. Gruteser**, and **W. Trappe**, Parknet: A mobile sensor network for harvesting real time vehicular parking information. *In Proceedings of the 2009 MobiHoc S3 Workshop on MobiHoc S3*, MobiHoc S3 '09. ACM, New York, NY, USA, 2009. ISBN 978-1-60558-521-5. URL http://doi.acm.org/10.1145/1540358.1540367.

[100] **Mekruksavanich, S.** and **A. Jitpattanakul**, Sensor-based complex human activity recognition from smartwatch data using hybrid deep learning network. *In 2021 36th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*. 2020.

[101] **Mekruksavanich, S.** and **A. Jitpattanakul**, Smartwatch-based human activity recognition using hybrid lstm network. *In 2020 IEEE SENSORS*. 2020. ISSN 2168-9229.

[102] **Meyer, J.**, **A. Frank**, **T. Schlebusch**, and **E. Kasneci** (2022). U-har: A convolutional approach to human activity recognition combining head and eye movements for context-aware smart glasses. *Proc. ACM Hum.-Comput. Interact.*, **6**(ETRA). URL https://doi.org/10.1145/3530884.

[103] **Miller, A.**, **J. Malasig**, **B. Castro**, **V. L. Hanson**, **H. Nicolau**, and **A. Brandão**, The use of smart glasses for lecture comprehension by deaf and hard of hearing students. *In Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '17. Association for Computing Machinery, New York, NY, USA, 2017. ISBN 9781450346566. URL https://doi.org/10.1145/3027063.3053117.

[104] **Min, J.-K.**, **A. Doryab**, **J. Wiese**, **S. Amini**, **J. Zimmerman**, and **J. I. Hong**, Toss 'n' turn: Smartphone as sleep and sleep quality detector. *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14. Association for Computing Machinery, New York, NY, USA, 2014. ISBN 9781450324731. URL https://doi.org/10.1145/2556288.2557220.

[105] MonkeyParking (). Monkeyparking : on-street parking on demand. http://monkeyparking.strikingly.com.

[106] **Nandugudi, A.**, **T. Ki**, **C. Nuessle**, and **G. Challen**, Pocketparker: Pocketsourcing parking lot availability. *In Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '14. Association for Computing Machinery, New York, NY, USA, 2014. ISBN 9781450329682. URL https://doi.org/10.1145/2632048.2632098.

[107] **Nawaz, S.**, **C. Efstratiou**, and **C. Mascolo**, Parksense: A smartphone based sensing system for on-street parking. *In Proceedings of the 19th Annual International Conference on Mobile Computing & Networking - MobiCom '13*. ACM Press, 2013. ISBN 978-1-4503-1999-7. URL http://dl.acm.org/citation.cfm?doid=2500423.2500438.

[108] **Nemati, E.**, **Y. S. Suh**, **B. Motamed**, and **M. Sarrafzadeh**, Gait velocity estimation for a smartwatch platform using kalman filter peak recovery. *In 2016 IEEE 13th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*. 2016.

[109] **Nguyen, L. N. N.**, **D. Rodríguez-Martín**, **A. Català**, **C. Pérez-López**, **A. Samà**, and **A. Cavallaro**, Basketball activity recognition using wearable inertial measurement units. *In Proceedings of the XVI International Conference on Human Computer Interaction*, Interacción '15. Association for Computing Machinery, 2015. URL https://dl.acm.org/doi/10.1145/2829875.2829930.

[110] **Noor, M. H. M.**, **Z. Salcic**, and **K. I.-K. Wang** (2017). Adaptive sliding window segmentation for physical activity recognition using a single tri-axial accelerometer. *Pervasive and Mobile Computing*, **38**, 41–59. ISSN 1574-1192. URL https://www.sciencedirect.com/science/article/pii/S1574119216302280.

[111] **Odhiambo, C. O.**, **S. Saha**, **C. K. Martin**, and **H. Valafar** (2022). Human activity recognition on time series accelerometer sensor data using lstm recurrent neural networks. *arXiv preprint arXiv:2206.07654*.

[112] **Ohkawa, Y.**, **M. Kodama**, **Y. Konno**, **X. Zhao**, and **T. Mitsuishi**, A study on ui design of smartphone app for continuous blended language learning. *In 2018 5th International Conference on Business and Industrial Research (ICBIR)*. IEEE, 2018.

[113] **Ott, W. R.** (1999). Mathematical models for predicting indoor air quality from smoking activity. *Environmental Health Perspectives*, **107**(suppl 2), 375–381.

[114] **Pande, A.**, **J. Zhu**, **A. K. Das**, **Y. Zeng**, **P. Mohapatra**, and **J. J. Han** (2015). Using smartphone sensors for improving energy expenditure estimation. *IEEE Journal of Translational Engineering in Health and Medicine*, **3**, 1–12. ISSN 2168-2372.

[115] **Panzieri, S.**, **F. Pascucci**, and **G. Ulivi** (2002). An outdoor navigation system using gps and inertial platform. *IEEE/ASME transactions on Mechatronics*, **7**(2), 134–142.

[116] **Parate, A.**, **M.-C. Chiu**, **C. Chadowitz**, **D. Ganesan**, and **E. Kalogerakis** (2014). Risq: Recognizing smoking gestures with inertial sensors on a wristband. *MobiSys ...: The ... International Conference on Mobile Systems, Applications and Services. International Conference on Mobile Systems, Applications, and Services*, **2014**, 149–161.

[117] **Parkinson, A.**, **R. Kitchen**, **A.-D. Tudor**, **S. Minocha**, and **S. Tilling**, Role of smartphone-driven virtual reality field trips in inquiry-based learning. *In Geographical Association Annual Conference 2017*. 2017. URL https://oro.open.ac.uk/49749/.

[118] **Parkitekt** (2015). Parkitekt bangalore. https://goo.gl/oitu6f Accessed : 2015-02-12.

[119] **Pasadyn, S. R.**, **M. A. Soudan**, **M. Gillinov**, **P. L. Houghtaling**, **D. M. Phelan**, **N. Gillinov**, **B. Bittel**, and **M. Y. Desai** (2019). Accuracy of commercially available heart rate monitors in athletes: a prospective study. *Cardiovascular diagnosis and therapy*, **9 4**, 379–385. URL https://api.semanticscholar.org/CorpusID:198375042.

[120] **Popkin, B. M.** and **I. H. Rosenberg** (2011). Nih public access. *NIH Public Access*, **68**(8), 439–458.

[121] **Qiu, S.**, **J. Hu**, **T. Han**, **H. Osawa**, and **M. Rauterberg** (2020). Social glasses: simulating interactive gaze for visually impaired people in face-to-face communication. *International Journal of Human–Computer Interaction*, **36**(9), 839–855.

[122] **Quiroz, J. C.**, **M. H. Yong**, and **E. Geangu**, Emotion-recognition using smart watch accelerometer data: Preliminary findings. *In Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*, UbiComp '17. Association for Computing Machinery, 2017. ISBN 978-1-4503-5190-4. URL https://doi.org/10.1145/3123024.3125614.

[123] **Radhakrishnan, M.**, **D. Rathnayake**, **O. K. Han**, **I. Hwang**, and **A. Misra**, Erica: Enabling real-time mistake detection & corrective feedback for free-weights exercises. *In Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, SenSys '20. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450375900. URL https://doi.org/10.1145/3384419.3430732.

[124] **RajKumar, A.**, **C. Arora**, **B. Katz**, and **V. Kapila**, Wearable smart glasses for assessment of eye-contact behavior in children with autism. *In Frontiers in Biomedical Devices*, volume 41037. American Society of Mechanical Engineers, 2019.

[125] **Raman, A.**, **Y. Don**, **R. Khalid**, **F. Hussin**, **O. Fauzee**, **M. Sofian**, and **M. Ghani** (2014). Technology acceptance on smart board among teachers in terengganu using utaut model. *Asian Social Science*, **10**(11), 84–91.

[126] **Ranjan, J.** and **K. Whitehouse**, Object hallmarks: Identifying object users using wearable wrist sensors. *In Proceedings of the 2015 ACM International Joint*

Conference on Pervasive and Ubiquitous Computing, UbiComp '15. Association for Computing Machinery, 2015. ISBN 978-1-4503-3574-4. URL https://doi.org/10.1145/2750858.2804263.

[127] **Reddy, S.**, **M. Mun**, **J. Burke**, **D. Estrin**, **M. Hansen**, and **M. Srivastava** (2010). Using mobile phones to determine transportation modes. *ACM Trans. Sen. Netw.*, **6**(2), 13:1–13:27. ISSN 1550-4859. URL http://doi.acm.org/10.1145/1689239.1689243.

[128] **Remolar, I.**, **C. Rebollo**, and **J. A. Fernández-Moyano** (2021). Learning history using virtual and augmented reality. *Computers*, **10**(11). ISSN 2073-431X. URL https://www.mdpi.com/2073-431X/10/11/146.

[129] **Richoz, S.**, **L. Wang**, **P. Birch**, and **D. Roggen** (2020). Transportation mode recognition fusing wearable motion, sound, and vision sensors. *IEEE Sensors Journal*, **20**(16), 9314–9328.

[130] **Rispler, C.**, **G. Luria**, **A. Kahana**, and **S. Rosenblum** (2018). Mood impact on automaticity of performance: Handwriting as exemplar. *Cognitive Computation*. ISSN 1866-9964.

[131] **Saeb, S.**, **M. Zhang**, **C. Karr**, **S. Schueller**, **M. Corden**, **K. Kording**, and **D. Mohr** (2015). Mobile phone sensor correlates of depressive symptom severity in daily-life behavior: An exploratory study. *Journal of Medical Internet Research*, **17**.

[132] **Sato, T.**, **J. Sandars**, **J. Brown**, and **S. N. Rogers** (2021). Usefulness of smart glasses and point of view for suturing skills training in medical students: pilot study. *BMJ Simulation & Technology Enhanced Learning*, **7**(3), 173.

[133] **Schrapel, M.**, **M.-L. Stadler**, and **M. Rohs**, Pentelligence: Combining pen tip motion and writing sounds for handwritten digit recognition. *In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018.

[134] **Sen, A.** and **H. Shah**, Automated handwriting analysis system using principles of graphology and image processing. *In 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*. 2017.

[135] **Sen, S.**, **V. Subbaraju**, **A. Misra**, **R. Balan**, and **Y. Lee**, Annapurna: building a real-world smartwatch-based automated food journal. *In 2018 IEEE 19th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE, 2018.

[136] **Sen, S.**, **V. Subbaraju**, **A. Misra**, **R. K. Balan**, and **Youngki Lee**, The case for smartwatch-based diet monitoring. *In 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. IEEE, 2015. ISBN 978-1-4799-8425-1. URL http://ieeexplore.ieee.org/document/7134103/.

[137] **Shi, J.**, **S. Liu**, **L. Zhang**, **B. Yang**, **L. Shu**, **Y. Yang**, **M. Ren**, **Y. Wang**, **J. Chen**, **W. Chen**, *et al.* (2020). Smart textile-integrated microelectronic systems for wearable applications. *Advanced materials*, **32**(5), 1901958.

[138] **Shin, D.**, **D. Aliaga**, **B. Tunçer**, **S. M. Arisona**, **S. Kim**, **D. Zünd**, and **G. Schmitt** (2015). Urban sensing: Using smartphones for transportation mode classification. *Computers, Environment and Urban Systems*, **53**, 76–86. ISSN 0198-9715. URL https://www.sciencedirect.com/science/article/pii/S0198971514000921. Special Issue on Volunteered Geographic Information.

[139] **Shin, D.**, **S. Lee**, and **S. Hwang** (2021). Locomotion mode recognition algorithm based on gaussian mixture model using imu sensors. *Sensors*, **21**(8), 2785. ISSN 1424-8220. URL https://www.mdpi.com/1424-8220/21/8/2785.

[140] **Singh, S.** and **B. Aksanli** (2019). Non-intrusive presence detection and position tracking for multiple people using low-resolution thermal sensors. *Journal of Sensor and Actuator Networks*, **8**(3), 40.

[141] **Smith, J. R.**, **B. Snapp**, **S. Madar**, **J. R. Brown**, **J. Fowler**, **M. Andersen**, **C. D. Porter**, and **C. Orban** (2023). A smartphone-based virtual reality plotting system for stem education. *Primus*, **33**(1), 1–15.

[142] **Sogi, N. R.**, **P. Chatterjee**, **U. Nethra**, and **V. Suma**, Smarisa: A raspberry pi based smart ring for women safety using iot. *In 2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*. 2018.

[143] **Soubam, S.**, **M. Agrawal**, and **V. Naik**, Using an arduino and a smartwatch to measure liquid consumed from any container. *In 2017 9th International Conference on Communication Systems and Networks (COMSNETS)*. IEEE, 2017. ISBN 978-1-5090-4250-0. URL http://ieeexplore.ieee.org/document/7945434/.

[144] **Soubam, S.**, **D. Banerjee**, and **V. Naik**, Detecting writing micro-events using motion sensors in smartwatches. *In 2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*. 2023.

[145] **Soubam, S.**, **D. Banerjee**, **V. Naik**, and **D. Chakraborty** (2015). Bluepark: Tracking parking and un-parking events in indoor parking lot. Technical Report IIITD-TR-2015-009, Indraprastha Institute of Information Technology Delhi (IIIT-Delhi). URL https://repository.iiitd.edu.in/xmlui/bitstream/handle/123456789/335/IIITD-TR-2015-009.pdf?sequence=1.

[146] **Soubam, S.**, **D. Banerjee**, **V. Naik**, and **D. Chakraborty**, Bluepark: tracking parking and un-parking events in indoor garages. *In Proceedings of the 17th International Conference on Distributed Computing and Networking*, ICDCN '16. Association for Computing Machinery, New York, NY, USA, 2016. ISBN 9781450340328. URL https://doi.org/10.1145/2833312.2833458.

[147] **Spandonidis, C.**, **F. Giannopoulos**, **J. Agulló**, **N. S. Tachos**, **L. Frederiksen**, **A. Martín**, and **R. G. Carvajal** (2023). Development of visual sensors and augmented reality based smart glasses for enhancement of business sustainability and wellbeing of the aging workforce. *IEEE Instrumentation & Measurement Magazine*, **26**(6), 21–27.

[148] **Stankoski, S.**, **M. Jordan**, **H. Gjoreski**, and **M. Luštrek** (2021). Smartwatch-based eating detection: Data selection for machine learning from imbalanced data with imperfect labels. *Sensors*, **21**(5), 1902. ISSN 1424-8220. URL https://www.mdpi.com/1424-8220/21/5/1902.

[149] **Tang, V. W.**, **Y. Zheng**, and **J. Cao**, An intelligent car park management system based on wireless sensor networks. *In 2006 First International Symposium on Pervasive Computing and Applications*. 2006.

[150] **Tao, L.**, **T. Volonakis**, **B. Tan**, **Y. Jing**, **K. Chetty**, and **M. Smith** (2018). Home activity monitoring using low resolution infrared sensor. *arXiv preprint arXiv:1811.05416*.

[151] **Thomaz, E.**, **I. Essa**, and **G. D. Abowd**, A practical approach for recognizing eating moments with wrist-mounted inertial sensing. *In Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '15. ACM, New York, NY, USA, 2015. ISBN 978-1-4503-3574-4. URL http://doi.acm.org/10.1145/2750858.2807545.

[152] **Tizzano, G. R.**, **M. Spezialetti**, and **S. Rossi**, A deep learning approach for mood recognition from wearable data. *In 2020 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*. 2020.

[153] **Uva, A. E.**, **M. Gattullo**, **V. M. Manghisi**, **D. Spagnulo**, **G. L. Cascella**, and **M. Fiorentino** (2018). Evaluating the effectiveness of spatial augmented reality in smart manufacturing: a solution for manual working stations. *The International Journal of Advanced Manufacturing Technology*, **94**, 509–521.

[154] **Uysal, C.**, **A. Onat**, and **T. Filik** (2020). Non-contact respiratory rate estimation in real-time with modified joint unscented kalman filter. *IEEE access: practical innovations, open solutions*, **8**, 99445–99457. ISSN 2169-3536.

[155] **Verma, R.**, **S. Ghosh**, **M. Saketh**, **N. Ganguly**, **B. Mitra**, and **S. Chakraborty**, Comfride: A smartphone based system for comfortable public transport recommendation. *In Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18. Association for Computing Machinery, New York, NY, USA, 2018. ISBN 9781450359016. URL https://doi.org/10.1145/3240323.3240359.

[156] **Vinayaga-Sureshkanth, N.**, **A. Maiti**, **M. Jadliwala**, **K. Crager**, **J. He**, and **H. Rathore** (2018). A practical framework for preventing distracted pedestrian-related incidents using wrist wearables. *CoRR*, **abs/1811.04797**. URL http://arxiv.org/abs/1811.04797.

[157] **Walmsley-Eyre, L.** and **R. Cardell-Oliver**, Hierarchical classification of low resolution thermal images for occupancy estimation. *In 2017 IEEE 42nd conference on local computer networks workshops (LCN Workshops)*. IEEE, 2017.

[158] **Wang, G.**, **Q. Li**, **L. Wang**, **W. Wang**, **M. Wu**, and **T. Liu** (2018). Impact of sliding window length in indoor human motion modes and pose pattern recognition based on smartphone sensors. *Sensors*, **18**(6), 1965. ISSN 1424-8220. URL https://www.mdpi.com/1424-8220/18/6/1965.

[159] **Wang, H.** and **W. He** (2011). A reservation-based smart parking system. *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 690–695. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5928901.

[160] **Wang, H.**, **T. T.-T. Lai**, and **R. Roy Choudhury**, Mole: Motion leaks through smartwatch sensors. *In Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, MobiCom '15. Association for Computing Machinery, New York, NY, USA, 2015. ISBN 9781450336192. URL https://doi.org/10.1145/2789168.2790121.

[161] **Wang, J.-S.** and **F.-C. Chuang** (2012). An accelerometer-based digital pen with a trajectory recognition algorithm for handwritten digit and gesture recognition. *IEEE Transactions on Industrial Electronics*, **59**(7), 2998–3007.

[162] **Wang, R.**, **F. Chen**, **Z. Chen**, **T. Li**, **G. Harari**, **S. Tignor**, **X. Zhou**, **D. Ben-Zeev**, and **A. T. Campbell**, Studentlife: Assessing mental health, academic performance and behavioral trends of college students using smartphones. *In Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2014. ISBN 978-1-4503-2968-2. URL https://dl.acm.org/doi/10.1145/2632048.2632054.

[163] **Wang, R.**, **G. Harari**, **P. Hao**, **X. Zhou**, and **A. T. Campbell**, Smartgpa: how smartphones can assess and predict academic performance of college students. *In Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp '15*. ACM Press, Osaka, Japan, 2015. ISBN 978-1-4503-3574-4. URL http://dl.acm.org/citation.cfm?doid=2750858.2804251.

[164] **Wang, Z.**, **X. Shi**, **J. Wang**, **F. Gao**, **J. Li**, **M. Guo**, **H. Zhao**, and **S. Qiu**, Swimming motion analysis and posture recognition based on wearable inertial sensors. *In 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. 2019. ISSN 2577-1655.

[165] **Wang, Z.**, **T. Zhao**, **J. Ma**, **H. Chen**, **K. Liu**, **H. Shao**, **Q. Wang**, and **J. Ren** (2022). Hear sign language: A real-time end-to-end sign language recognition system. *IEEE Transactions on Mobile Computing*, **21**(7), 2398–2410.

[166] **Wehbi, M.**, **T. Hamann**, **J. Barth**, and **B. Eskofier**, Digitizing handwriting with a sensor pen: A writer-independent recognizer. *In 2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2020.

[167] **Wen, C.** and **J. Zhang** (2015). Design of a microlecture mobile learning system based on smartphone and web platforms. *IEEE Transactions on Education*, **58**(3), 203–207.

[168] **Wen, H.**, **J. Ramos Rojas**, and **A. K. Dey**, Serendipity: Finger gesture recognition using an off-the-shelf smartwatch. *In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 2016. ISBN 978-1-4503-3362-7. URL https://dl.acm.org/doi/10.1145/2858036.2858466.

[169] **Wu, S.**, **H. Duan**, **X. Min**, **D. Tu**, and **G. Zhai**, Accurate compensation makes the world more clear for the visually impaired. *In 2021 IEEE International Conference on Image Processing (ICIP)*. 2021.

[170] **Xie, W.**, **Q. Zhang**, and **J. Zhang** (2021). Acoustic-based upper facial action recognition for smart eyewear. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, **5**(2). URL https://doi.org/10.1145/3448105.

[171] **Xin, J. F.** and **F. X. Sutman** (2011). Using the smart board in teaching social stories to students with autism. *Teaching Exceptional Children*, **43**(4), 18–24.

[172] **Yoon, C. K.** (1998). Us drinking itself dry, study finds. *The New York Times*. URL https://www.nytimes.com/1998/06/16/science/us-drinking-itself-dry-study-finds.html.

[173] **Youssef, M.** and **A. Agrawala**, The horus wlan location determination system. *In Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services - MobiSys '05*. ACM Press, 2005. ISBN 978-1-931971-31-7. URL http://portal.acm.org/citation.cfm?doid=1067170.1067193.

[174] **Yu, S.**, **H. Chen**, and **R. A. Brown** (2018). Hidden markov model-based fall detection with motion sensor orientation calibration: A case for real-life home monitoring. *IEEE Journal of Biomedical and Health Informatics*, **22**(6), 1847–1853. ISSN 2168-2208.

[175] **Yue, S.**, **Y. Yang**, **H. Wang**, **H. Rahul**, and **D. Katabi** (2020). Bodycompass: Monitoring sleep posture with wireless signals. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, **4**(2), 66:1–66:25. URL https://dl.acm.org/doi/10.1145/3397311.

[176] **Yuh, A. H.** and **S. J. Kang**, Real-time sound event classification for human activity of daily living using deep neural network. *In 2021 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*. 2021.

[177] **Zafari, F.**, **A. Gkelias**, and **K. K. Leung** (2019). A survey of indoor localization systems and technologies. *IEEE Communications Surveys & Tutorials*, **21**(3), 2568–2599.

[178] **Zhang, C.**, **A. Bedri**, **G. Reyes**, **B. Bercik**, **O. T. Inan**, **T. E. Starner**, and **G. D. Abowd**, Tapskin: Recognizing on-skin input for smartwatches. *In Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces*, ISS '16. ACM, New York, NY, USA, 2016. ISBN 978-1-4503-4248-3.

[179] **Zhang, R.** and **O. Amft** (2017). Monitoring chewing and eating in free-living using smart eyeglasses. *IEEE journal of biomedical and health informatics*, **22**(1), 23–32.

[180] **Zhang, S.**, **M. Ang**, **W. Xiao**, and **C. Tham**, Detection of activities for daily life surveillance: Eating and drinking. *In e-health Networking, Applications and Services, 2008. HealthCom 2008. 10th International Conference on*. 2008.

[181] **Zhang, T.**, **J. Wang**, **L. Xu**, and **P. Liu**, Fall detection by wearable sensor and one-class svm algorithm. *In* **D.-S. Huang**, **K. Li**, and **G. W. Irwin** (eds.), *Intelligent Computing in Signal Processing and Pattern Recognition: International Conference on Intelligent Computing, ICIC 2006 Kunming, China, August 16–19, 2006*, Lecture Notes in Control and Information Sciences. Springer, 2006, 858–863. URL https://doi.org/10.1007/978-3-540-37258-5_104.

[182] **Zhang, W.**, **X. Zhao**, and **Z. Li** (2019). A comprehensive study of smartphone-based indoor activity recognition via xgboost. *IEEE Access*, **7**, 80027–80042. ISSN 2169-3536. URL https://ieeexplore.ieee.org/document/8736849/.

[183] **Zhou, C.**, **P. Dai**, **F. Wang**, and **Z. Zhang** (2016). Predicting the passenger demand on bus services for mobile users. *Pervasive and Mobile Computing*, **25**, 48–66. ISSN 1574-1192. URL https://www.sciencedirect.com/science/article/pii/S157411921500187X.

LIST OF PAPERS BASED ON THESIS

1. S. Soubam, D. Banerjee, V. Naik, "Detecting Writing Micro-Events using Motion Sensors in Smartwatches," *The 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things*, Volume, Page, 2023.

2. S. Soubam; M. Agrawal; V. Naik, "Using an Arduino and a smartwatch to measure liquid consumed from any container," *2017 9th International Conference on Communication Systems and Networks (COMSNETS)*, Pages 464-467, 2017.

3. S. Soubam, D. Banerjee, V. Naik, D. Chakraborty, "Bluepark: tracking parking and un-parking events in indoor garages," *Proceedings of the 17th International Conference on Distributed Computing and Networking*, Article No.: 33, Pages 1–4, 2016.