Human Activity Recognition using a single ankle mounted accelerometer

Agrawal, Mitali

Award date:
2021

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Uncertainty in Artificial Intelligence Research Group (UAI/e)
Postbus 513, 5600 MB Eindhoven
The Netherlands
www.tue.nl

Author
Mitali Agrawal
m.agrawal1@student.tue.nl
agrawal.mitali@gmail.com

Supervisors
Dr. Ir. Erik Quaeghebeur (TU/e)
Stefan De Vries (Mentech Innovation)

Date
October 4, 2021

Human Activity Recognition using a single ankle mounted accelerometer

Thesis Report

Master's in Data Science graduation project,
Mentech Innovation B.V.

Abstract

This work provides details for solving Human Activity Recognition (HAR) problem, using multiple and single wireless Inertial Measurement Unit (IMU) sensors. The HAR problem is seen from different perspectives in this project; firstly how HAR is being performed using a network multiple IMU sensors collecting multiple inertial signals for classifying multiple Activities of Daily Life (ADLs), secondly how HAR is performed using a single IMU sensor collecting multiple inertial signals for ADL classification, and lastly, how HAR is being performed using just single IMU sensor collecting only acceleration signal. This project utilizes the explored literature to solve the problem of HAR, focused upon reducing the sitting/standing confusion often faced by Machine learning (ML) and Deep learning (DL) models.

To implement the proposed solutions, an online available dataset PAMAP2 is used. An in-house dataset was also collected at Mentech with an aim of collecting a diverse dataset with minimum motion-based noise possible. Since some walking instances were recorded in sitting-standing data-classes for Mentech dataset during data-collection, both the datasets were pre-processed and cleaned to remove any noise recorded before feeding them into the proposed classifiers. Data cleaning was done using standard deviation applied non-overlapping windows of 10 samples, with a threshold value of '0.3'. Using this filter the data-signals for sitting and standing activity class were processed to remove noise, while no filter was applied for walking data signals. Both the datasets were also combined to create a new combined dataset, named Megabase in this report. The data-fragmentation was done using different input window sizes, and acceleration components (like three acceleration values, orientation features). 3D tensors were created to do the required data-manipulation steps, to prepare the windowed data-inputs for DL models.

The proposed models for achieving high classification accuracy, specially for sitting and standing activity classes are, 1D Convolutional Neural Network (CNN) and 2D CNN. The proposed architecture uses longer convolutional kernels, which leads to significant improvement in model accuracy. The use of smaller input window-size in data-fragmentation also improves the model performance. Best observed accuracies for all the (filtered) datasets was achieved using 3 orientation features and window-size of lengths less than 1.28 seconds. For PAMAP2 dataset a final accuracy of $\approx 94\%$ was achieved, while for Mentech and Megabase datasets, a final accuracy of $\approx 85\%$ was achieved.

Technische Universiteit
Eindhoven
University of Technology

Table of contents

Title
Human Activity Recognition using a single ankle mounted accelerometer

Technische Universiteit
Eindhoven
University of Technology

Table of contents

Title
Human Activity Recognition using a single ankle mounted accelerometer

1 Introduction

As life expectancy is increasing, people are living longer lives. This increase means more older people will require reliable health care in the coming future. However, there are too few caregivers for the given demand. This discrepancy can be expected to increase over time. With this imbalance, a single caregiver takes care of a more significant number of patients. This increase in workload can lead to stress for caregivers. The nursing staff often finds it stressful to document every patient condition correctly and take care of them simultaneously.

People who have dementia or mental disabilities often struggle to express their feelings and emotions. An ideal caregiver should be skilled and experienced in dealing with such patients correctly to identify their emotional states. Such caregivers are less available. Thus, nursing staff working with these patients often exhaust themselves and develop burnout symptoms (Looff 2019). It is hence crucial to support such caregivers and lessen their burden.

Mentech Innovation develops a wearable sock called 'SentiSock' to aid such healthcare staff. SentiSock is powered by the HUME (an emotion recognition platform), an AI-based solution providing insights into a person's emotional state. It is specially designed to notify the caregiver if any of their patients are under stress, using real-time data analysis. It is developed to act as a reliable early warning system to prevent stress escalations and give more accurate emotion predictions compared to less experienced medical staff.

Physiological data like heart rate and skin conductance can be used to identify stress and fatigue in a person (Khanade et al. 2017). Such data is measured and processed using machine learning algorithms to calculate stress levels in real time. In the coming future, HUME aims to utilize just one sensor capable of collecting different physiological signals, including an Electro Dermal Activity (EDA) to measure stress levels. It uses a 2-dimensional emotion space to translate the extracted physiological parameters into emotional states, called the circumplex model (POSNER et al. 2005). There are four quadrants of the arousal and valence plane to classify the positive and negative nature of emotions. An overview of emotion classes and their relation with the Arousal and Valence axis can be seen in the Figure 1.1.

EDA or more commonly known as Skin Conductance (SC) and Galvanic Skin Response (GSR), is a measure of electrical characteristics of the skin. Human SC increases if there is more sweat creation (Darrow 1964). Whenever a person undergoes stress, sweat glands are activated, and EDA increases. In our body, the palms and soles of our feet have the most densely distributed sweat glands. This is why the electrodes that measure the skin conductance variation are located on the base of the SentiSock. This signal is then transferred to the sensor housing, present at the ankle of the sock. An overview of the SentiSock product and HUME dashboard for monitoring real-time stress can be seen in the Figure 1.2. The HUME dashboard enables the user to gain insight into the subject's emotional state, which caregivers can use to monitor multiple patients efficiently.

To improve HUME's real-time accuracy, it is essential to consider the human posture and physical activity of the patient while developing stress monitoring algorithms. Many physiological data like Heart Rate (HR) and SC are sensitive to both anxiety and exercise (Sun et al. 2012). So when a person undergoes physical load, EDA is increased as a result of increased sweating rate (Posada-Quintero et al. 2018). Schumm et al. 2008, verified this after measuring EDA variability and provoking it by different walking speeds in an experiment. Similarly, the HR (which is a feature that can be extracted from ECG signal) increases for a person when they standing-up (Hallman et al. 2019). This creates a possibility for the model to confuse the high SC and HR values activated by physical activity, with highly stressful moments. This leads to the creation of false alarms. To avoid these wrong escalations, it is essential to know the real-time activity and posture of the patient.

The vital importance of activity recognition is a widely researched topic. Every activity a person performs is associated with different physical and physiological responses, which show unique characteristics. The use of multiple inertial sensors like a gyroscope and an accelerometer have proven to yield

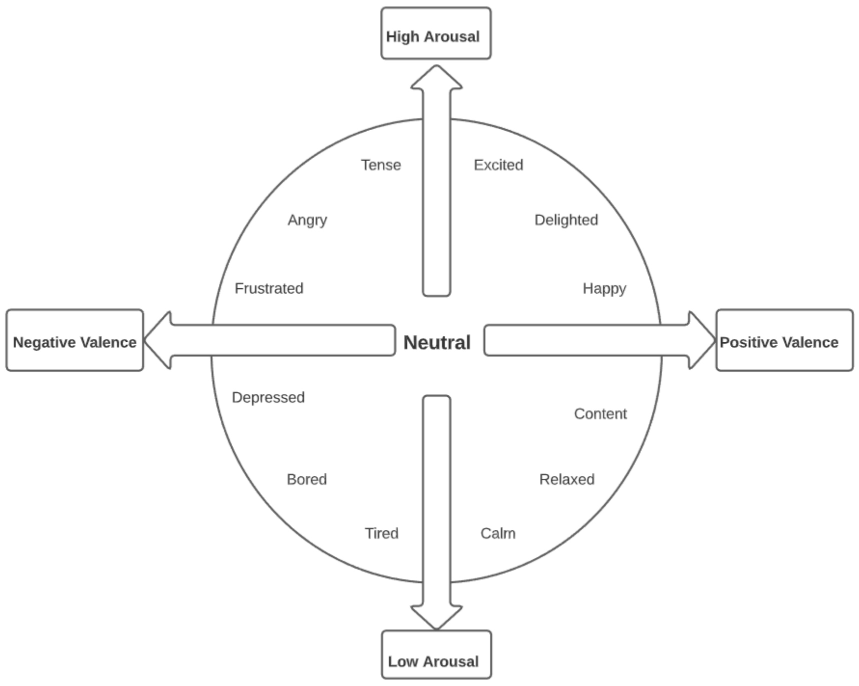


Figure 1.1: Circumplex Model using Valence-Arousal Axis for Emotion classification

promising results in possessing these unique characteristics to classify human activity accurately. However, the use of multiple wearable sensors results in movement obstruction and hence is not a practical solution for long-term wearable devices. This is why more research is being done nowadays that focuses on using only one accelerometer to perform the activity classification tasks. This single sensor can be located at a person's wrist, hip (or smartphone placement), thigh, upper arm, or ankle. The sensor housing found at the ankle of SentiSock contains an accelerometer, which will be used for activity detection.

The accuracy of activity classification depends upon the quality of features extracted from the raw signal. These features can be hand-crafted or can be automatically extracted based upon the model architecture. Hand-crafted features include meaning, correlation, signal transforms, entropy, standard deviation, etc. These are usually followed by classification techniques like support vector machines, decision trees, multi-layer perceptron neural networks, random forests, and K-nearest-neighbours. For automatic feature extraction, Convolutional Neural Networks (ConvNet/CNN), Long Short-Term Memory (LSTM) are being researched. These approaches are discussed in more detail in Chapter 2.

The restriction of using a single acceleration sensor positioned at the human ankle makes the activity recognition problem more complex and challenging. The hardest part for classification models is to distinguish between the sitting and standing positions of a human. This is because the acceleration signals show very similar feature properties when it comes to these two classes. In this study, we create a classification model to distinguish between three classes; sitting, standing, and walking. To make the model robust against these instances, a well-designed dataset was collected, covering a variety of human sitting and standing positions. More details for data collection are elaborated in Chapter 3.

To summarize, healthcare industries are suffering from decreasing availability of highly skilled caregivers. Mentech's product SentiSock (with HUME) aims to aid the healthcare industry by providing insights into a person's emotions. To achieve this, they utilize sweat to calculate the EDA signal, which the HUME processes to provide stress levels in real-time. The sweat, however, can be stress-induced or because of some physical activity. Since the SentiSock consists of just one accelerometer, it is difficult to correctly identify human activity using data-driven approaches like ML/DL models. This study will

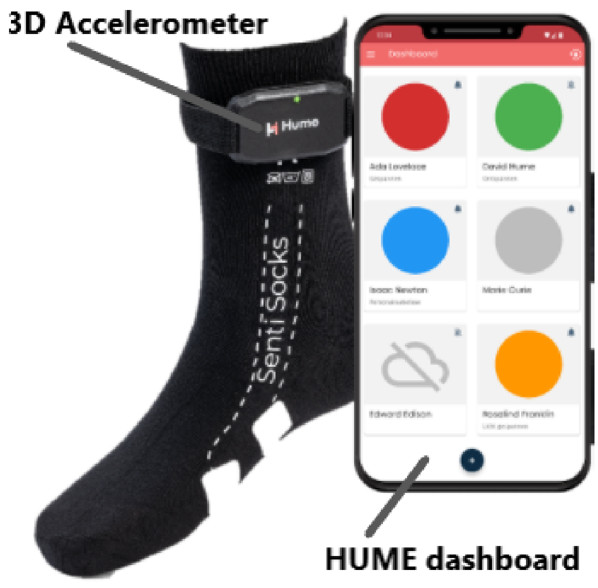


Figure 1.2: SentiSock with HUME dashboard

explore literature for existing human activity classification techniques and implement selected architectures as per project requirements (/HUME specifications). This means that the input data will be tri-axial (i.e., values along planar and vertical planes) acceleration values, being collected from an accelerometer placed at the ankle of a person. It will be a 3-class classification problem, focusing on accurately classifying the sitting and standing classes.

1.1 Research Questions

The aim and direction of this research are based on the research questions that follow. The study is broadly done on two levels; The first level is the Human Activity Recognition (HAR) model, without any restrictions. On this level, state-of-art methods and the latest HAR techniques utilizing accelerometer data from wearable sensors are researched. The second is the HUME level, which sets specifications like sensor placement and sensor configuration parameters. This defines the scope of the model. Since no one yet researches this second level, the solution development process will involve exploring a combination of the most promising approaches available to create a novel one.

The Research questions at the HAR level are:

1. What are the traditional approaches and state-of-art methods used for developing a HAR model using inertial signals?
2. What are the limitations of the explored approaches?
3. What are the most important and commonly used signals for HAR?
4. What pre-processing steps should be applied to filter the input signal for removing noises as well as preserving essential features?
5. What are the essential features that can be extracted from accelerometer signal data?

The Research questions at the HUME/SentiSock level are:

1. Which HAR model architecture is most suitable, keeping in mind the HUME specifications?

2. How can we make the model perform efficiently in real-time?
3. What are the requirements for collecting a good quality dataset (i.e., having important physiological features and least possible noise) from ankle mounted accelerometer for classification models?
4. How can we integrate the final model with HUME?

1.2 Thesis outline

1. **Chapter Two - Literature review:**
   At first essential HAR tools are discussed; wearable hardware sensors and datasets (online and collection). Then traditional approaches to solving HAR are discussed, followed by a more focused study with project constraints.
2. **Chapter Three - Dataset:**
   At first, datasets used for this project are discussed; PAMAP2, Mentech, and a combined dataset of these two (Megabase). Then the implemented data analysis and pre-processing techniques are discussed.
3. **Chapter Four - Methodology:**
   Combinations of CNN and LSTM architectures have been proposed, illustrated, and discussed.
4. **Chapter Five - Results:**
   Classification results are discussed and compared, mainly for; proposed models with different input window sizes with hand-crafted features working on different datasets .
5. **Chapter Six - Conclusions:**
   Insights gained from the model classification results and corresponding limitations are discussed.
6. **Chapter Seven - Future Recommendations:**
   Based on the limitations observed, the following line of research is suggested.

2 Theoretical background and Literature review

Before going through the literature ideas, let us first see how our input data looks, what signals can be recorded from wearable inertial sensors and the common points where these sensors are placed on a human body.

2.1 Sensor signals

A sensor is a device that converts physical action into an electrical equivalent. These electrical signals can then be transferred to servers and processed. When these sensors are integrated with wearable material, they are called wearable sensors to collect the physiological signals directly from the human body. The standard signals collected by such sensors include Electrocardiogram (ECG), Electroencephalogram (EEG), Electrodermal Activity (EDA), acceleration, angular velocity, temperature, Electromyography (EMG), Radio-frequency identification (RFID), Motion, Pressure, etc.

An inertial sensor, or Inertial Measurement Unit (IMU), collects specific 3D data such as acceleration, angular velocity, tilt or orientation, and magnetic field at the object. IMU hence is usually comprised of an accelerometer, gyrometer, and magnetometer, as shown in Figure 2.1, collecting a tri-variate time-series data for every signal. IMU collects multiple axes data to achieve an output of six or nine degrees of freedom. For example, if an IMU sensor comprises an accelerometer and gyrometer, it will be called a 6-Axis inertial sensor with six degrees of freedom. IMU can also include a barometer, pressure sensor, orientation sensor, etc.

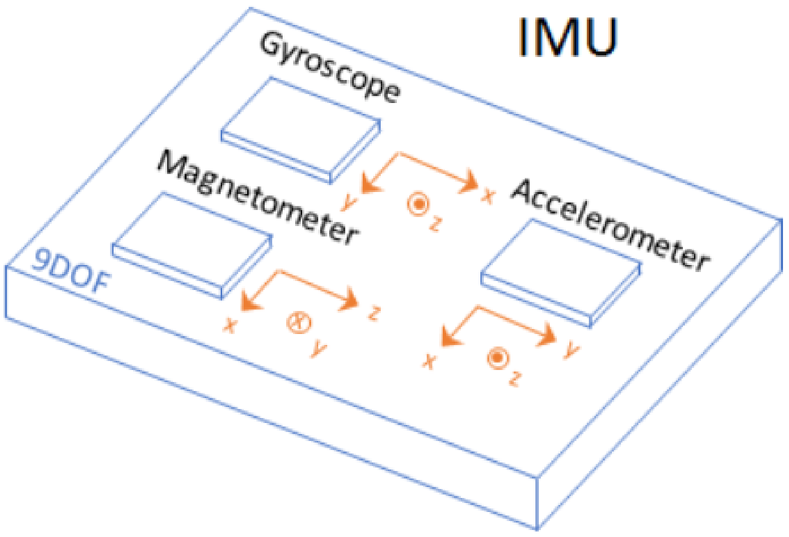


Figure 2.1: A 9-Axis IMU sensor (Mathworks n.d.)

2.1.1 Accelerometer

An accelerometer is a device that measures the acceleration or rate of change of velocity of an object (K. Chen et al. 2021). As per the International System of Units (SI), its measuring unit is meters per squared second ($^{m}/_{s^2}$). Acceleration of any object at rest, when measured on Earth's surface, will have a default acceleration due to Earth's gravity. By definition, this gravitational force, $g \approx 9.81\,^{m}/_{s^2}$ acts along the vertical Z-axis (Corke 2017; Ha et al. 2016). Common accelerometer applications include monitoring vibrations in mechanical machines (Albarbar et al. 2009), for stabilizing flights (Corda et al. 2002), correcting noise created by hand-shaking in digital mobile cameras and auto-rotating visual content on smartphone screens (ensuring the upright position of content w.r.t. real-time device orientation) (Mark et al. 2014), etc.

2.1.2 Gyroscope or Gyrometer

A gyroscope is a device that measures and maintains the angular velocity and orientation of an object. The unit of measured angular velocity is in degrees per second (°/s). A Gyroscope is helpful to measure the rate of '*tilt*', where the axis of rotation is free to assume any fixed point of reference, i.e., while rotating the object, the orientation of this axis remains unchanged. Gyroscopes are often referred to as Gyrometers when they are microchip-packaged and used in industries. Like accelerometers, gyrometers are used for stabilizing devices undergoing severe vibrations in industrial machines, ships navigation (Bennett 1984), aviation (Banning et al. 2020), mobiles (Mark et al. 2014), satellites (Zharkov et al. 2020) etc. Apart from this, combinations of gyrometers and accelerometers are also used in navigation systems, replacing the traditional magnetic compass (J.-H. Chen et al. 1994; Bristeau et al. 2011).

2.1.3 Magnetometer

A magnetometer is a device that measures the strength, direction, and change of magnetic field at a particular point. Its measuring unit is Tesla (T). They are used to identify fingerprints in a smartphone (Park et al. 2020), detect human gestures (Fang et al. 2018), aircraft aviation (Banning et al. 2020), detect submarines (M. Wang et al. 2019) etc. They are also widely integrated with accelerometers and gyrometers in IMU sensors for activity recognition.

2.1.4 Common sensor placements

Let us assume we have one IMU sensor, which collects tri-variate time-series data from an accelerometer, gyrometer, and magnetometer. This sensor will have nine degrees of freedom. The sampling frequency of these signals lies in the range of tens to hundreds of Hz. This sensor can now be mounted on various human body parts like the waist, arm, ankle, wrist, hip, chest, etc., as illustrated in Figure 2.2. We can also use multiple IMU sensors with the same or different specifications and mount them on other body parts. This technique provides an opportunity to align the time-series data collected by various IMU sensors and analyze the combined data. This process of aligning multi-axes data, coming from a network of sensors, is called **sensor fusion**.

2.2 Commonly used datasets

For Human Activity Recognition, we will focus on data collected from IMU sensors, i.e., focused on capturing data related to acceleration, angular velocity, tilt/orientation, magnetic field changes around a body part (since we are talking about wearable sensors). Next, we will discuss some popular online-available datasets collected by the University of California Irvine (UCI) in their Machine Learning Repository that were collected specifically to research the problem of HAR. Researchers use these datasets widely to implement different Machine Learning (ML), Deep Learning (DL) architectures to solve HAR problems and generate benchmark results. The details of these UCI datasets and some other popular online datasets are summarized in Table 2.1.

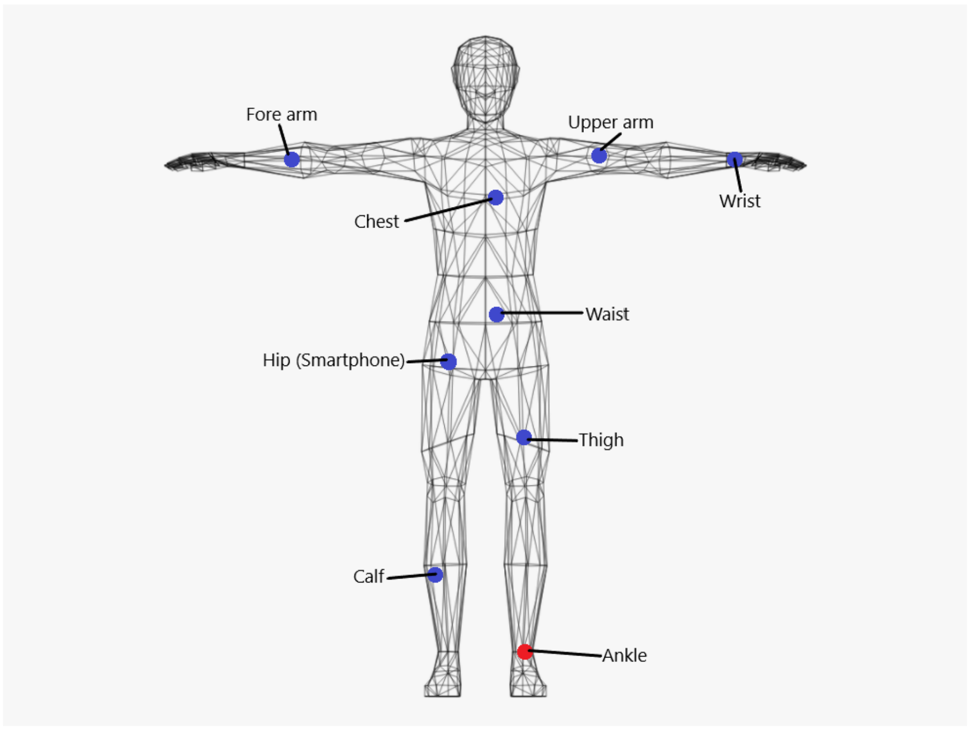


Figure 2.2: Commonly used body placements for wearable sensor are shown with blue dots, while red dot signifies the sensor placement used for this project

DATASET	ACTIVITIES	SENSORS	SAMPLING RATE	VOLUNTEERS
UCI-HAR (Reyes-Ortiz et al. 2012)	6	A, G	50 Hz	30
WISDM (Weiss 2019)	6	A	20 Hz	36
OPPORTUNITY (Roggen et al. 2012)	17	A, G, M, O, AM	30 Hz	4
MHEALTH (Banos 2014)	12	A, G, M	50 Hz	10
WHARF (Bruno et al. 2012)	12	A (at-wrist)	32 Hz	17
PAMAP2 (Reiss 2012)	18	A, G, M	100 Hz	8

A = accelerometer, G = gyroscope, M = magnetometer,
O = object sensor, AM = ambient sensor

Table 2.1: Details of the commonly available online datasets for HAR

2.2.1 MHEALTH

The MHEALTH (Mobile-HEALTH) dataset by UCI was collected to study the ECG signal with different daily human activities (Banos et al. 2014; Baños et al. 2015). Multiple sensors were used to diversify the body parts involved in measuring the activity-motion. For this 3D accelerometer, 3D gyroscope and 3D magnetometer were used to collect 9-Axis inertial data. These sensors were placed on the chest, right wrist, and left ankle of 10 subjects. The chest sensor also obtained the ECG measurements. Every volunteer performed 12 daily living activities like cycling, sitting, running, etc. All sensor data were collected at a sampling rate of 50 Hz.

2.2.2 UCI-HAR using smartphone sensors

The Human Activity Recognition database was built by UCI from the recordings of 30 subjects within the age range of 19-48 years (Reyes-Ortiz et al. 2012). Each subject performed Activities of Daily Living (ADL), including six activities (walking, sitting, standing, lying, walking up and down the stairs). 6-Axis inertial data was collected from the 3D accelerometer and 3D gyroscope embedded in a smartphone at a sampling rate of 50 Hz. The smartphone was waist-mounted for real-life activity scenarios. The collected sensor signals (accelerometer and gyroscope) were pre-processed by applying noise filters. The acceleration signal was passed through a Butterworth low-pass filter to separate body acceleration from the gravitational component. Along with signal-activity labels, this dataset also provides vectors of a total of 561 features calculated from the time and frequency domain (like frequency-bands energy, entropy, skewness, signal magnitude, phase angle, etc.).

2.2.3 WISDM

The WISDM dataset by UCI includes data collected from smartphones and smartwatches simultaneously (Weiss 2019). It was collected from 51 test subjects performing 18 different ADLs. The phone and the watch include a 3D accelerometer and a 3D gyroscope collecting 6-Axis inertial data simultaneously. The smartphone was placed in the subject's pocket, while the smartwatch was worn on the dominant hand's wrist of the subject. The sensor data was collected at a sampling rate of 20 Hz.

2.2.4 Data collection

An in-house dataset was collected at Mentech Lab for the HAR problem, mainly focused on sitting and standing classes. With the motivation to assemble a high-quality dataset, best practices to do so were researched. The aim was (i) to obtain a diverse dataset capturing different activity positions (for sitting, standing, and walking), and (ii) to avoid introducing noise of any kind, which can harm and confuse DL models. Firstly, the data collection details provided by the available online datasets were studied to achieve these aims. As explained in the 'README' file for the MHEALTH dataset (Banos et al. 2014; Baños et al. 2015), to capture a diverse dataset, the main factors considered during the data collection of different activity classes are:

1. To include a diversity of different body parts involved during the execution of an activity. For example, wrist movements, knee bending, the elevation of arms and legs, etc.
2. To include activities of different intensities and speeds such as cycling, walking, sitting, lying, etc.

In the paper Janssen et al. 2002, a general study was conducted to detect sit-to-stand (STS) movements or transitions for a variety of input data-streams used for motion analysis (like force plates, accelerometers, pressure sensors, etc). In the Table 2 of Janssen et al. 2002, a paper by Goulart et al. 1999 is referred, who studied the determinants of sit-to-stand movement using piezoelectric accelerometer. The found determinants focused upon the use of different human muscles induced by different sitting postures and chair-support being used, as shown in Figure 2.3. This essence is discussed in depth in the paper, where the importance of chair-specific features like height, armrests and, backrests are discussed (Janssen et al. 2002). Different sitting postures and foot positioning were also found to be useful aspects in detecting STS movements.

Different sitting postures, with the varied sitting environment, therefore lead to different sets of body muscles being used, affecting the acceleration signals. Most datasets included different speed settings for dynamic activity classes like walking and cycling to have intensities of activities. Defining these factors for data collection is usually called Environment setup.

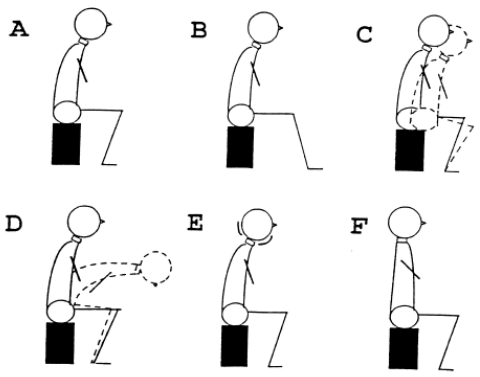


Figure 2.3: Schematic representation of the six experimental conditions used in the study by Goulart et al. 1999, to detect the sit-to-stand movement. Here, Posture A is used as the Reference condition; Posture B is the feet forward sitting; Posture C is the knees first sitting; Posture D is sitting with flexion of the trunk; Posture E is the head with support; Posture F is the straight trunk.

To ensure that the least amount of noise is recorded during data collection, we must look at different types of noises in wearable sensors. Wearable sensor noise can either be motion-induced or sensor-intrinsic (Vijayan et al. 2021). Sensor-intrinsic noises are hardware-components-based abnormalities, which are out-of-scope for this project. Motion-induced noises are caused by human motion while data-recording, which does not correspond to the recorded class.

To understand the motion-induced noises, we must revisit the activity classes for this project. Sitting and Standing classes represent the 'Static activities' or those activities where the subject is steady with respect to the sensor in the environment setup. In contrast, the Walking class represents the 'dynamic class' where the subject is moving continuously (Shelke et al. 2019). Although, this 'sensor environment setup' can differ from project to project. For example, some online datasets allow posture change in static classes, especially sitting, while some add different types of sitting classes w.r.t. predefined postures. Another typical environmental setup for the sitting class can be using a 'rocking chair', which differs from the regular human sitting and lying postures. Due to this dynamic environment setup which exists in real-life for the sitting class, various researches debate if the 'sitting' class should be treated as static or dynamic (Riener et al. 2007).

Various researchers also define any 'interruptions' encountered during recording an activity class (static or dynamic) as noise. Hence, they should be avoided or removed. In a paper by Anzanpour et al. 2019, researchers prioritized to minimize the interruptions caused by replacing IMU sensor batteries (by caregivers) while the subject was 'sleeping' at night. In another paper Tabrizi et al. 2020, table tennis forehand strokes were studied. While collecting the required data for such a dynamic class, every time a player paused or waited to re-position or re-grip the tennis racquet, they were called interruptions. To avoid recording such noise while real-time labeling of data, the activities were performed under the surveillance of three high-ranked coaches, who collected the labeled data for every player simultaneously. In a study by Riener et al. 2007, different car-sitting environment setups were proposed with varying sitting postures to detect the driver's activity. The idea was to use fixed sitting posture while collecting the corresponding class label data to avoid recording any noise or interruptions (which is a change in sitting posture in this scenario).

Hence, it is essential to define the environment setup of any recorded activity label to describe the noise corresponding to that activity. For 'static' sitting and standing classes, any continuous motion should be considered noise. For a 'dynamic' walking class, any interruptions caused during data-recording leading the subject to pause the walking activity should be regarded as noise.

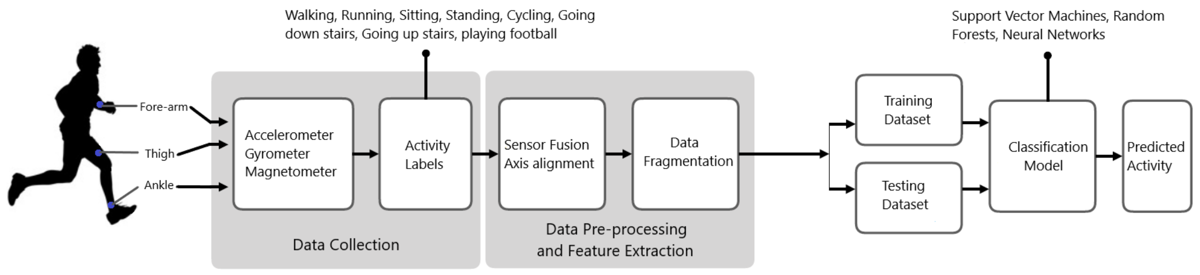

Walking, Running, Sitting, Standing, Cycling, Going down stairs, Going up stairs, playing football
Support Vector Machines, Random Forests, Neural Networks
Fore-arm
Thigh
Ankle
Accelerometer Gyrometer Magnetometer
Activity Labels
Sensor Fusion Axis alignment
Data Fragmentation
Training Dataset
Testing Dataset
Classification Model
Predicted Activity
Data Collection
Data Pre-processing and Feature Extraction

Figure 2.4: Generic HAR pipeline using multiple inertial sensors

2.3 Traditional approaches for HAR

The traditional way to perform HAR is to (i) extract meaningful information like hand-crafted features from sensor data and (ii) feed them to a classifier to predict the activity (Jordao et al. 2018). Various classifying algorithms researched in previous works for HAR include Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Multi-Layer Perceptron neural networks (MLP), Hidden Markov Model (HMM), Random Forest (RF), and Decision Trees (DT) (K. Chen et al. 2021; Ha et al. 2016; Gholamrezaii et al. 2021; Sukor et al. 2018; Z. Chen et al. 2018). These models compare the sensor input data during their training phase to produce individual activity template models. The main advantage of using such ML models (using extracted features) is that they help in reducing the model sensitivity towards the noisy real-life environment and imbalanced datasets (Sukor et al. 2018; Riboni et al. 2009)[page 26]. On the downside, these methods highly rely on the hand-crafted features extracted in the first step.

Unlike other ML methods, Deep Learning (DL) models auto-extract important features from signals instead of hand-crafted features. Although, some state-of-art techniques use a combination of DL methods and selected hand-crafted features to improve classification performance (Z. Chen et al. 2018; Ignatov 2018). This increase in performance emphasizes the importance of hand-crafted features in performing HAR, using both ML and DL approaches.

2.3.1 Hand-crafted features

The main focus of extracting meaningful features or of doing **feature engineering** is to generate data representations that capture even the most negligible differences among activity classes (Sukor et al. 2018; Z. Chen et al. 2018). Often a network of IMU and other wearable sensors are utilized to collect different 3D signals. The Figure 2.4 depicts the typical architecture used to collect multi-sensor data for HAR, pre-process it and perform classification on it (Badawi et al. 2018). The data signals are pre-processed to calculate statistical features like mean, standard deviation, frequency energy-bands, kurtosis, the correlation coefficient between two signals, etc. These features help better classification performance, even when using shallow algorithms like KNN, SVM, and RF. The advantage of using this structure (feature engineering followed by a classifier) is that the classifier model can find relevant features in a dynamic real-time environment (Sukor et al. 2018; Riboni et al. 2009).

The hand-crafted statistical features are either derived from the time-domain or frequency-domain. Both these domain features contain valuable properties helpful in differentiating activities (Ha et al. 2016; Gholamrezaii et al. 2021; Z. Chen et al. 2018). For example, the frequency domain can provide insights to classify activities with different frequencies, like walking, running, brushing teeth. The time-domain feature like ’the mean value’ provides the magnitude of the signal, which can be used to classify static and dynamic activities. Hence with proper domain knowledge, hand-crafted features can be designed by specialists. The Table 2.2 summarizes important, commonly used features in both the domains (Z. Chen et al. 2018; Sukor et al. 2018).

FEATURE TYPE	METHODS
Time-Domain	Mean, Standard deviation, Maximum, Minimum, Signal magnitude area, Average sum of squares, Signal vector magnitude, Signal entropy, Autoregression coefficients, Correlation coefficient
Frequency-Domain	Largest frequency component, Weighted average, Skewness, Kurtosis, Energy of a frequency spectrum, Phase angle

Table 2.2: Commonly used Time and Frequency domain features

2.3.2 Limitations

Even if domain specialists carefully engineer the hand-crafted features, there exists no single feature extraction approach that can be applied universally for effective human activity classification (K. Chen et al. 2021). Another disadvantage of using these features is the high computational cost. The pre-processing of 3D signal data for extracting a list of hand-crafted features can take a lot of processing time. Machine Learning (ML) algorithms that rely on these features are hence computationally costly. By training shallow ML models on these specific features, some significant inherent signal information can also get lost (Z. Chen et al. 2018). This data loss makes it difficult to distinguish between sitting, standing, lying, walking, running, and going down the stairs.

In the paper Saez et al. 2016, researchers use PAMAP2 dataset and apply different ML techniques like RF, Extra randomized trees (Extra trees), principal component analysis (PCA), SVM, etc. for HAR. Top performing approaches were Extra trees and PCA with average accuracies of 96.1% and 92.5% respectively. Although sitting and standing accuracies were 88% and 83% for Extra trees, and 78% and 74% for PCA. Similar low-accuracy result pattern was also observed by other researchers, where although the overall accuracy was $\sim$95%, the accuracy for sitting and standing classes remained low (Z. Chen et al. 2018; Ignatov 2018). This led to the exploration of more complex solutions for performing Human Activity Recognition, like Deep Learning algorithms.

2.4 State-of-art approaches for HAR

To eliminate the computational overhead and complexity of calculating hand-crafted features, Deep Learning (DL) models were explored as they learn relevant features directly from the raw signal data. Convolutional neural networks (Convnet, long short-term memory (LSTM) networks) and combinations of these two models are being actively researched. Among all the Deep Neural Network (DNN) approaches, Convolutional Neural Networks (CNNs) have repeatedly proved to be a powerful tool for HAR (Ha et al. 2016; Gholamrezaii et al. 2021). This is because CNN models capture the local dependencies in the input signal along the temporal dimension (Ha et al. 2016). State-of-art methods hence utilize 1D or 2D convolutional layers for feature extraction and activity classification.

2.4.1 Convolutional Neural Network (CNN) Architecture

Convolutional Neural Networks (CNNs) is a type of Artificial Neural Networks (ANNs) that initially gained popularity in the field of image and video processing (Rafegas et al. 2020). Its unique architecture enabled it to pick important patterns from input and convert them into meaningful information (Konstantinidis et al. 2020). The image data is composed of RGB components, often called *channels*. For example, if we have an image of length l and breadth b, the CNN input will be a $l \times b$ matrix, each element having three values (red, green, and blue). This concept is extended for signal processing, where the time-series data from a sensor is divided into equal parts, called *windows*. The multi-axis values of sensors (X, Y, and Z-axis values) can be compared to the RGB channels of the image data.

CNN's are hierarchical feedforward neural networks consisting of an input layer, an output layer, and one or more hidden layer(s) (Ignatov 2018). Hidden layers contain combinations of convolutional and pooling layers, which are a special type of layers found in CNN architecture (Erdaş et al. 2021). The **Convolutional layer** is the most important layer in CNNs they transform the input data using convolution operations (Xia et al. 2020). A convolutional *kernel* (also called *filter*) slides over the input data, processing or transforming it to create a *feature map*. The values of this kernel or filter are also called **weights**. Let vector **f** be the convolutional filter which slides over vector **x**, which is the sub-section of input data. Then the convolution vector **c** is the scalar product computed between the vectors **x** and **f** at each step or sub-region (Xia et al. 2020). A feature map is made by stacking the computed vector **c**. This process is presented in Figure 2.5.

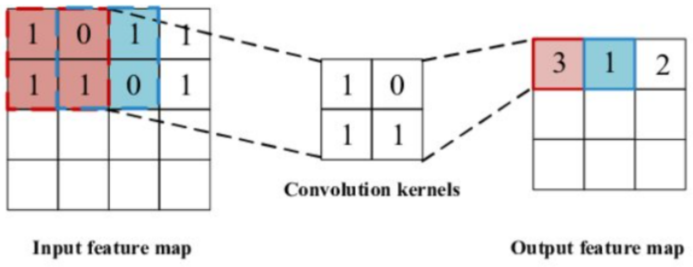


Figure 2.5: Working of a 2D convolutional kernel (Liang et al. 2020)

This transformed data or feature map is typically followed by an activation function. To learn the classification boundaries, non-linear activation functions are used (Ignatov 2018). Commonly used functions are sigmoid, hyperbolic tangent, and ReLU. The most commonly used function is the Rectified Linear Unit (ReLU) defined as a thresholding operation,s $ReLU(x) = \max(0, x)$, which converts negative values to zero while the positive values remain unchanged (Xia et al. 2020; Erdaş et al. 2021). This output is sent as input to the **Pooling layer**, where the information is further 'summarized' or, more specifically, the dimensionality of the input vector is reduced. This process is also called *subsampling* or *downsampling*. It is traditionally done by taking average, maximum, or sum of the sub-sections of input data (Xia et al. 2020).

After having multiple convolutional and pooling layers, the final output is flattened to create a one-dimensional vector used for classification (Ignatov 2018). This is sent to a fully connected layer called **Dense layer**. In this layer, every neuron receives an input from each neuron of the previous layer (Erdaş et al. 2021). A CNN architecture can have one or more dense layers to perform the classification. The final layer in CNN architecture is the **Softmax layer**, which computes the probability distribution over every element of the output vector, which are the predicted classes. A 1D CNN architecture is illustrated in Figure 2.6, whose input data comes from 6 channels (for example, 3D accelerometer and 3D gyrometer), and window size is 128 values.

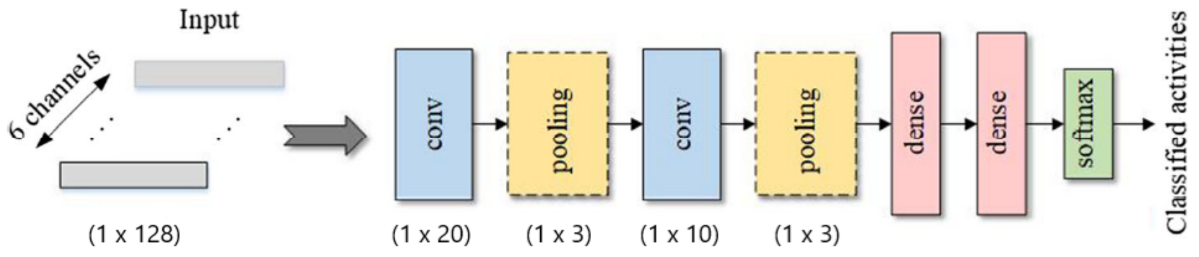


Figure 2.6: 1D CNN architecture for HAR (Gholamrezaii et al. 2021)

2.4.2 Long Short-Term Memory (LSTM) Architecture

The Long Short-Term Memory (LSTM) is a special kind of Neural Network which yields good performance in long time-series data (Erdaş et al. 2021). Some deep Neural models suffer from the problem of vanishing/exploding gradient, which hinders the network's ability to capture the meaningful information required to map raw sensor data and human activity classes in long time-context windows (L. Wang et al. 2020; Xia et al. 2020). LSTM structure overcomes this limitation by having special memory units called **memory cells** (Rafegas et al. 2020; Xia et al. 2020; Z. Chen et al. 2018). These memory cells have a gate structure, which enables the learning of when to 'forget' and/or 'update' previous hidden state information (Z. Chen et al. 2018; Erdaş et al. 2021). It contains an input gate to control the read operation, an output gate to control the write operation, and a forget gate to control the reset operation (Erdaş et al. 2021). This memory cell structure of LSTM is seen in Figure 2.7.

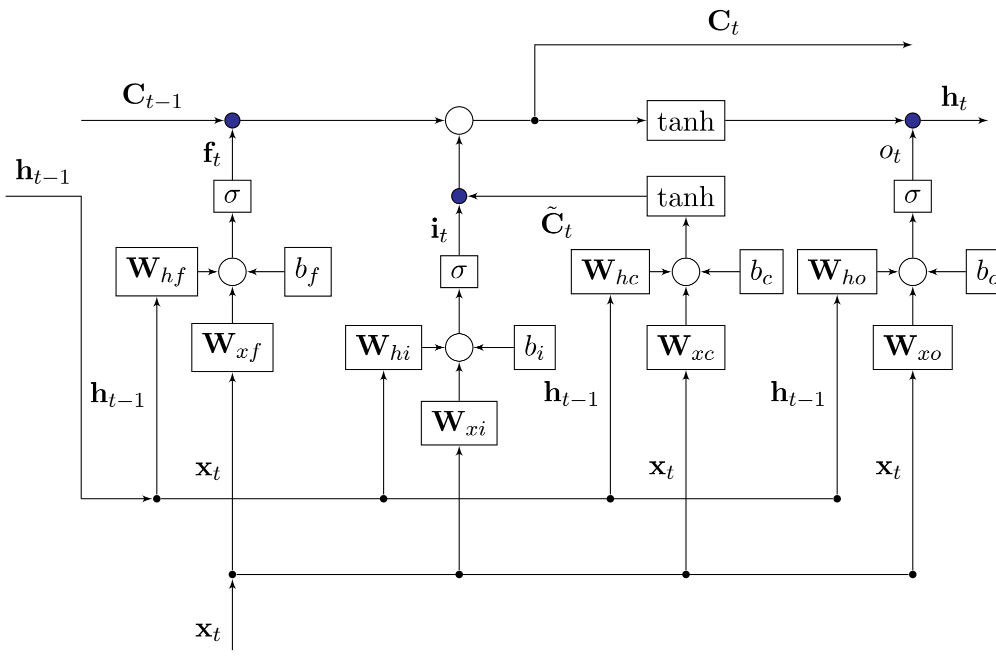


Figure 2.7: LSTM node architecture consisting of element-wise product is shown as blue dots, sigmoid function is shown as σ, hyperbolic tangent function is shown as $tanh$, addition is shown as a circular junction (denoted by empty circular dots). The different weights and biases are multiplied with respective signals of the $t-1$ time instance to calculate the t time instance.

The weights and bias of the LSTM network can be defined by different gate parameters, which are updated during the training phase (Erdaş et al. 2021). Let LSTM network update these parameters at time $\mathbf{t}$ based on its signal input $\mathbf{x}_t$, previous hidden state $\mathbf{h}_{t-1}$, and previous memory gate $\mathbf{C}_{t-1}$ using

(Z. Chen et al. 2018) below steps :

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + b_i\right) \tag{2.1a}$$
$$\mathbf{f}_t = \sigma\left(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + b_f\right) \tag{2.1b}$$
$$\mathbf{o}_t = \sigma\left(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + b_o\right) \tag{2.1c}$$
$$\tilde{\mathbf{C}}_t = \tanh\left(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + b_C\right) \tag{2.1d}$$
$$\mathbf{C}_t = \mathbf{f}_t \cdot \mathbf{C}_{t-1} + \mathbf{i}_t \cdot \tilde{\mathbf{C}}_t \tag{2.1e}$$
$$\mathbf{h}_t = \mathbf{o}_t \cdot \tanh\left(\mathbf{C}_t\right) \tag{2.1f}$$

Where:

$\mathbf{h}_t$	:	are the hidden units
$\mathbf{i}_t,\ \mathbf{f}_t,\ \mathbf{o}_t$	:	represent input, forget and output gate respectively
$\tilde{\mathbf{C}}_t,\ \mathbf{C}_t$	:	is the input modulation gate and the memory gate
$\mathbf{W}_{xf},\ \mathbf{W}_{xi},\ \mathbf{W}_{xc},\ \mathbf{W}_{xo}$	:	are the weights
$\mathbf{W}_{hf},\ \mathbf{W}_{hi},\ \mathbf{W}_{hc},\ \mathbf{W}_{ho}$	:	are the weights
$b_f,\ b_i,\ b_C,\ b_o$	:	is the bias

Using:

$m \cdot n$	:	is the element-wise product of m and n
$\sigma(\mathrm{k}) = \left(1 + e^{-\mathrm{k}}\right)^{-1}$	:	is the non-linear sigmoid function
$\tanh(\mathrm{k}) = \frac{e^{\mathrm{k}} - e^{-\mathrm{k}}}{e^{\mathrm{k}} + e^{-\mathrm{k}}} = 2\sigma(2\mathrm{k} - 1)$	:	is the non-linear hyperbolic tangent function

The input gate $\mathbf{i}_t$ and forget gate $\mathbf{f}_t$ are sigmoid in nature, i.e. they squash real-time inputs into a range of $[0, 1]$ (Z. Chen et al. 2018). The input gate is used to tune the influence of a current data $\mathbf{x}_t$ in a memory cell unit value C_t. In contrast, the forget gate is used to limit the influence of historical data $\mathbf{h}_{t-1}$ in memory cell unit (L. Wang et al. 2020). This special gate structure acts as knobs that LSTM learns to either forget the previous information or to focus upon current input data (Z. Chen et al. 2018).

To summarize, the input gate $\mathbf{i}_t$ in LSTM structure consists of 3 parts; the current input data $\mathbf{x}_t$, the output value of previous hidden state $\mathbf{h}_{t-1}$, and the memory cell unit value C_t. This C_t further is determined by two inputs; the previous memory cell unit C_{t-1} which is modulated by forget gate $\mathbf{f}_t$, and the $\tilde{\mathrm{C}}_t$, which is modulated by input gate $\mathbf{i}_t$ (Z. Chen et al. 2018). The output gate $\mathbf{o}_t$ controls the transfer of information from the memory cell unit to the hidden states. It is used to control the final output generated by the memory cell unit (L. Wang et al. 2020). So with the help of these three gates, LSTM alters the current state of its memory cell through internal updates. This structure enables LSTM to learn complex features from the space-time domain, making them efficient in working with longer input-data sizes or windows. This is why LSTM is seen as a potential solution for handling raw multi-dimensional time-series data for HAR.

2.4.3 Challenges of real-time systems

The downside of such a Deep Learning approach is that they require a large dataset to train on and can still lead to poor performance if there is noise in the training data (Z. Chen et al. 2018). Improving model accuracy in ML/DL techniques comes at the expense of computational cost (Gholamrezaii et al. 2021). Since computational time and performance accuracy of the solution are both critical factors in real-time systems, it is wise to create a good balance between them. For achieving this balance, it is essential to think about the following questions:

- How many layers a DL model should have?

REFERENCE	METHODS	ACCURACY %age point	DATASET	SIGNALS
Ha et al. 2016	1D CNN	0.43 pp	MHEALTH	3xA, 2xG
Ha et al. 2016	2D CNN	-0.86 pp	MHEALTH	3xA, 2xG
Ha et al. 2016	CNN-ppf	1.94 pp	MHEALTH	3xA, 2xG
Gholamrezaii et al. 2021	2D CNN	7.82 pp	MHEALTH	3xA, 2xG
Gholamrezaii et al. 2021	2D CNN + FFT	9.20 pp	MHEALTH	3xA, 2xG
Ignatov 2018	2D CNN + Stat. Features	3.32 pp	WISDM	2xA, 2xG, 2xM
Xia et al. 2020	LSTM + CNN	5.75 pp	WISDM	2xA, 2xG, 2xM
Xia et al. 2020	LSTM + CNN	2.71 pp	OPPORTUNITY	2xA, 2xG, 2xM

A = accelerometer, G = gyroscope, M = magnetometer
(All sensors are 3-Dimensional)

Table 2.3: Performance comparison of different architectures using multiple inertial signals from multiple 9-Axis or 6-Axis inertial sensors

- How much complex model architecture is required (CNN or LSTM)?
- Which features can contribute most to distinguish sitting and standing activity signals?

This motivates the literature direction explored in the following Sections, where we first focus on DL methods being researched for a network of wearable sensors, collecting multiple inertial signals. Then we gradually added the constraints of this project, reducing the literature to talk about the HAR problem using only acceleration signal being collected from a single sensor.

2.5 Network of 9-Axis/6-Axis inertial sensors

This Section reviews literature that uses a network of IMU sensors, collecting at least two sets of tri-axial signal data. Although we will be looking at a more specific problem later (HAR using a single sensor and focus on sit/stand/walk activity), to build a scalable solution (which can handle the complexity of adding more classes at a future stage), it is essential to look at solutions which involve the complexity of multiple sensors collecting multiple 3D-signal data. Various 1D-CNN, 2D-CNN, and LSTM architectures have been researched to handle the complexity of such input data using online datasets like WISDOM and MHEALTH. The model accuracy here-forward is defined as percentage point (pp), the percentage value of model accuracy above 90 or below.

An overview of results achieved in this literature direction is summarized in Table 2.3, while interesting architectural decisions are discussed below:

1. **CNN-pff :** Ha et al. 2016 in their study utilizes the concept of partial and full weight sharing in convolutional layers to learn the independent channel-specific features and the common features for all the input channels. The idea is to collect local characteristics from multi-model sensor data in the lower convolutional layer and gradually aggregate this learned information in the upper convolution layer. To achieve this they implement **CNN-pff** using the MHEALTH dataset, **p**artial and **f**ull-weight sharing in 1st 2D-convolution layer, followed by **f**ull-weight sharing in 2nd 2D-convolutional layer. The classification accuracy of CNN-pff increases by 1.94 percentage points (pp), while 1D and 2D CNN are at the expected ≈90%.

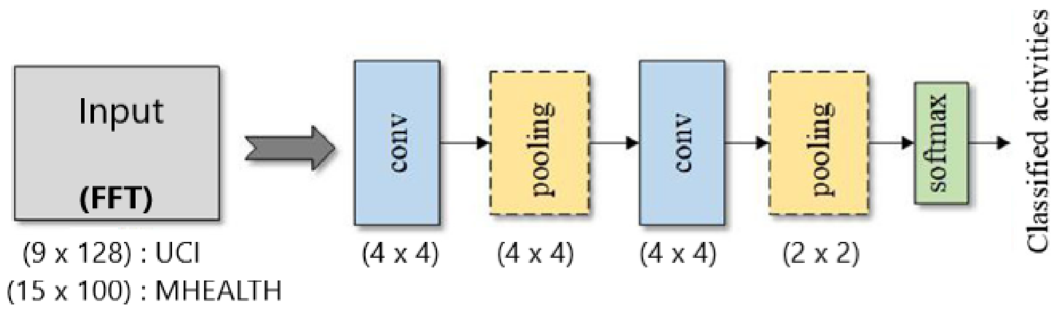


Figure 2.8: 2D CNN architecture for HAR with Fast Fourier Transform (FFT) applied as a pre-processing step to input signal (Gholamrezaii et al. 2021)

2. **Fast Fourier Transform :** Gholamrezaii et al. 2021 implemented 2D CNN architecture and to further improve its performance they studied the impact of applying Fast Fourier Transform (FFT) to the raw data before feeding it to the model. This approach, illustrated in Figure 2.8, resulted in increased classification accuracy of 1.38 pp when implemented on the MHEALTH dataset.

3. **Feature fusion :** Ignatov 2018 implemented a shallow (fewer layers) 2D CNN architecture using the WISDM dataset. They used one convolution layer, one max-pooling layer, and one dense layer. The output of the pooling layer was first flatted, and then statistical features were stacked to this 1D vector before feeding it to the dense layer for classification.

4. **Longer window-size :** Gholamrezaii et al. 2021 also studied the effect of different input window-size on classification performance. An increment of 0.42 pp was observed in accuracy after augmenting statistical features with 2D-CNN (as discussed in FFT step). The overall accuracy percentage point increased by 3 pp when window-size was increased from 2.5 s (or 50 samples) to 10 s (Ignatov 2018). It is interesting to note that the observed sitting and standing class accuracy percentage points were -7.37 pp (below 90%) and 3.33 pp, respectively.

5. **Data centering :** Gholamrezaii et al. 2021 further tests the effect of performing data centering or removing the offset in the raw input signal. This step standardizes the input signal, making the classification job easier for the CNN classifier. This step was done in combination with Feature fusion to prevent the loss of any important information. This technique increased the accuracy by 1.5pp.

6. **LSTM + CNN :** A combination of LSTM and CNN architecture was proposed by Xia et al. 2020, contains eight layers in total. The first two LSTM layers extract temporal features, followed by two convolutional layers for extracting spatial features. Every convolution layer was followed by a max-pooling layer. The Global Average Pooling (GAP) layer and Batch Normalization (BN) layer was added in the end. The authors tested the model on the WISDM and OPPORTUNITY datasets, and the overall accuracy percentage point was 5.75 pp and 2.71 pp, respectively.

2.6 Single 9-Axis/6-Axis inertial sensor

In this Section, the literature review covered for HAR will be focused upon using just one wearable sensor, utilizing at least two 3D signals being collected. All the papers discussed in this Section utilize smartphone data, more specifically the UCI-HAR dataset except for Z. Chen et al. 2018 and Ha et al. 2016, who use an in-house dataset (collected using smartphone) in their study. As seen in the previous

REFERENCE	METHODS	ACCURACY %age point	DATASET	SIGNALS
Ha et al. 2016	CNN-ppf	9.66 pp	In-house dataset (smartphone)	1xA, 1xG
Xia et al. 2020	LSTM + CNN	5.8 pp	UCI-HAR	1xA, 1xG
L. Wang et al. 2020	LSTM + FFT + Stat. Features	1.65 pp	UCI-HAR	1xA, 1xG
Mekruksavanich et al. 2021	2 layer CNN + LSTM	8.49 pp	UCI-HAR	1xA, 1xG
Mekruksavanich et al. 2021	4 layer CNN + LSTM	9.39 pp	UCI-HAR	1xA, 1xG
Z. Chen et al. 2018	SLFN	1.20 pp	In-house dataset (smartphone)	1xA(l), 1xA(t), 1xG
Z. Chen et al. 2018	SLFN + Hand-crafted features	6.50 pp	In-house dataset (smartphone)	1xA(l), 1xA(t), 1xG
Gholamrezaii et al. 2021	1D CNN + FFT (without pooling)	0.51 pp	UCI-HAR	1xA(l), 1xA(t), 1xG
Gholamrezaii et al. 2021	2D CNN + FFT (without pooling)	5.69 pp	UCI-HAR	1xA(l), 1xA(t), 1xG
Ignatov 2018	2D CNN + Stat. Features + data centering	7.63 pp	UCI-HAR	1xA, 1xG

$\mathrm{A(l)}$ = linear accelerometer, $\mathrm{A(t)}$ = total accelerometer,
G = gyroscope, M = magnetometer
(All sensors are 3-Dimensional)

Table 2.4: Performance comparison of different architectures using multiple inertial signals from a Single 9-Axis inertial sensor

Section, the 1D and 2D CNN architectures 2.5, was also implemented using the UCI-HAR dataset by some researchers. These approaches and many more are summarized in the Table 2.4.

The crucial findings from the architectural models implemented by different researchers in specified literature scope are:

1. **Effect of removing pooling layers in Convolutional Networks :** Gholamrezaii et al. 2021 in addition to studying the effect of applying FFT to raw input data (discussed in Section 2.5), we also studied the effect of removing pooling layers. More specifically, they studied the computational time taken by 1D and 2D CNN models to perform the classification task with and without pooling layers. The pooling layer was rather replaced by increasing the *stride size* of the previous convolutional layer. The implemented approach resulted in a noticeable improvement of $0.91\,\mathrm{s}$ in 2D CNN architecture and of $0.69\,\mathrm{s}$ in 1D CNN architecture. An important thing to note here is that removing the pooling layer and adding strides did not compromise the classification accuracy; rather, it was increased by 1 pp for 2D CNN.

2. **LSTM :** L. Wang et al. 2020 used a 2-layered LSTM model with selected hand-crafted features to perform activity classification. They use FFT on raw input data to capture time and frequency domain features, followed by calculating specific time-domain feature vectors (namely, kurtosis, skewness, mean absolute deviation, standard deviation, and root mean square). An accuracy of 1.65 pp was reported, while the change in percentage points of sitting and standing classes was -22.6 pp and 8.1 pp, respectively.

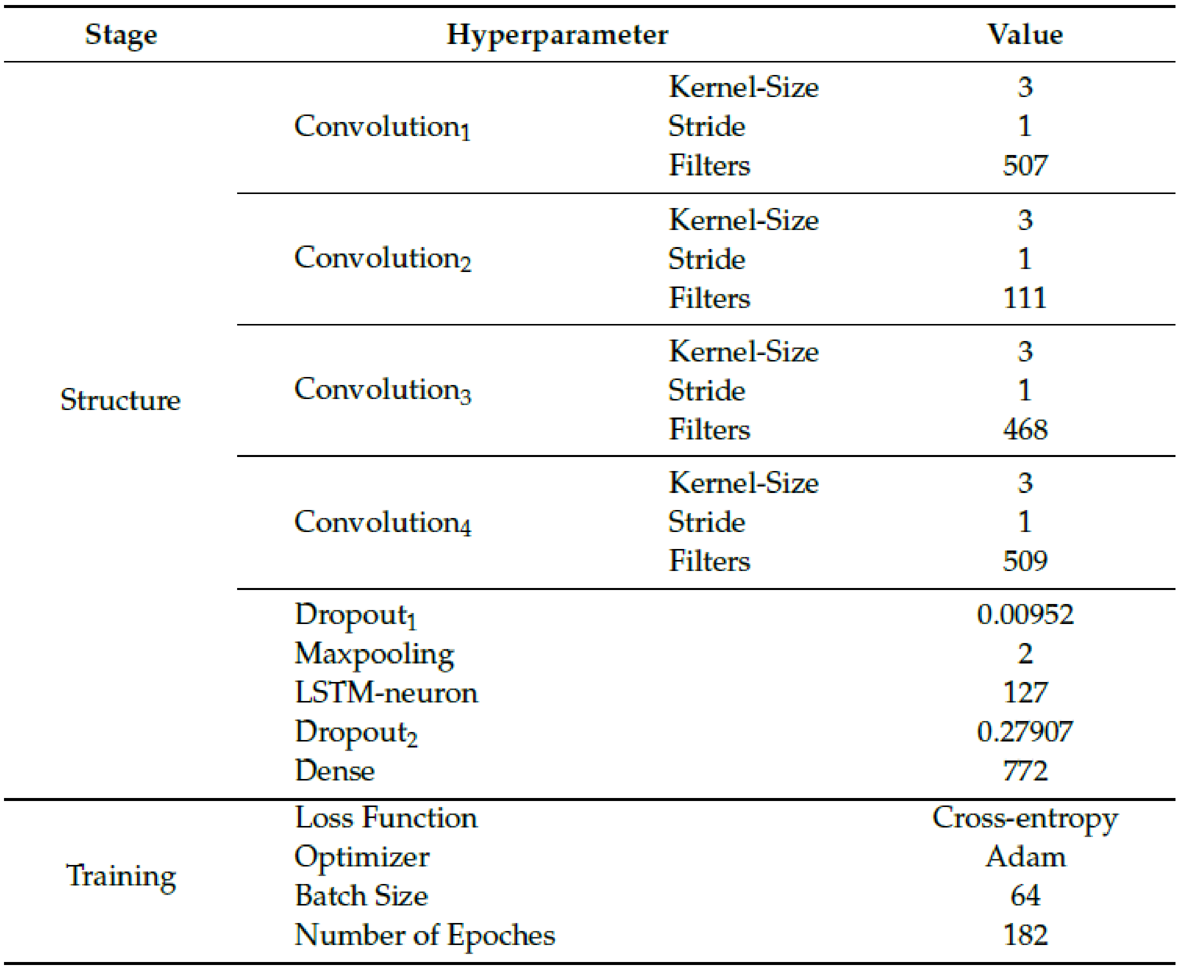
3. **CNN + LSTM :** Mekruksavanich et al. 2021 explored different variations of the CNN-LSTM network and achieved the best performance by using a 10-fold cross-validation protocol with 50% over-lapping windows. The Figure 2.9 shows the architecture of a 4-layered CNN-LSTM network, while 2-layered architecture can be visualized similarly with only two convolutional layers followed by a single LSTM layer.

4. **LSTM + CNN :** The LSTM-CNN architecture proposed by Xia et al. 2020 discussed in Section 2.5, was also implemented using the UCI dataset. The main difference between LSTM+CNN and CNN+LSTM model architectures is whether LSTM is used for feature extraction followed by CNN for classification or vice-versa. It is interesting to note that the sitting and standing accuracies observed were 3.32 pp and 2.18 pp, respectively. The confusion matrix results discussed in this paper show that the model's sitting and standing classes were the most confused classes.

5. **Teacher-Student LSTM model :** Z. Chen et al. 2018 focuses on **hard targets** in their study, i.e., activity classes which are likely to have higher chances of being misclassified. Activity classes such as {'sitting' , 'standing'} and {'walking upstairs' , 'walking downstairs'} are examples of such hard targets. Authors propose training 2 separate models, to extract the knowledge of handcrafted-features from one model and impart that knowledge to a second model which is training on raw input data to perform final classification. A deep LSTM model, called the *Teacher model* is trained on different features to carry rich domain knowledge. The second deep LSTM model, called the *Student model*, is kept deeper (with 128 neurons) than the teacher network (with 100 neurons). This enforces the second model, to generalize the input information in a similar way done by the teacher network. This is done by modifying the temperature T variable, in the final *Softmax layer* (Z. Chen et al. 2018).

2.7 Single 3-Axis inertial sensor collecting acceleration signal

This Section narrows down the literature closer to this project's use-case, i.e., research papers that discuss the HAR problem using only one wearable sensor, collecting only acceleration signal. Both 2D



Stage	Hyperparameter		Value
Structure	Convolution_1	Kernel-Size	3
		Stride	1
		Filters	507
	Convolution_2	Kernel-Size	3
		Stride	1
		Filters	111
	Convolution_3	Kernel-Size	3
		Stride	1
		Filters	468
	Convolution_4	Kernel-Size	3
		Stride	1
		Filters	509
	Dropout_1		0.00952
	Maxpooling		2
	LSTM-neuron		127
	Dropout_2		0.27907
	Dense		772
Training	Loss Function		Cross-entropy
	Optimizer		Adam
	Batch Size		64
	Number of Epoches		182

Figure 2.9: Hyperparameters summary for four-layer CNN-LSTM architecture (Mekruksavanich et al. 2021)

REFERENCE	METHODS	ACCURACY %age point	DATASET	SIGNALS
Mannini et al. 2017	SVM + Hand-crafted features	2.4 pp	Inhouse dataset	1xA (ankle)
Mentech Innovation	1D CNN	9.99 pp	Smartphone	1xA (hip)
Jordao et al. 2018	CNN^{+}	-10 pp	WHARF	1xA (wrist)
Erdaş et al. 2021	3D CNN fed by ConvLSTM	3.69 pp	Smartphone	1xA (chest)

A = 3-Dimensional accelerometer

Table 2.5: Performance comparison of different architectures using 3D acceleration signal from a Single 3-Axis inertial sensor

CNN and LSTM architectures have proved to be promising approaches for HAR in previous Sections. Although less research is available in the current scope, the model performance of implemented architectures is detailed in Table 2.5. The critical design choices and observations which were explored are discussed below:

1. **SVM :** Mannini et al. 2017 implemented SVM classifier using an in-house (youth) database (of age range 13 $\pm$ 1.3 years). This dataset was collected to study and compare the acceleration signal for HAR between two specific sensor placements, wrist, and ankle. They proposed using **Fragmentation** or input data-window features (where 1 data-fragment = 1 window of input-data), which were found to be informative for both the sensor placements. They also used selected hand-crafted features and SVM classifier and obtained an overall accuracy of $\sim$2 pp for the Ankle sensor. Another interesting but out-of-scope information here is that the wrist sensor's observed accuracy was lower than the ankle sensor ($\sim$ -8 pp).

2. **1D CNN :** The initial HAR model at Mentech consisted of 1D CNN architecture and was implemented using two sensors collecting 3D accelerometer data. The sensors were placed on the human chest and human hip (smartphone in pocket configuration). The model had a very high accuracy of $\approx$99%(9 pp). The accuracy remained unchanged even when the chest sensor was removed. It was found that the high accuracy (even for classes like sitting and standing) was due to the noticeable orientation change of the mobile phone. Smartphone placed in back pocket stays vertical in standing position and lays horizontal when a person is sitting. This leads gravity to act upon an entirely different acceleration-signal axis for these two positions. This orientation pattern is easily picked up by the model, leading to high model performance.

3. **CNN^{+} :** Different 1D and 2D CNN architectures (similar to as illustrated in Figures 2.6 and 2.8) were studied Jordao et al. 2018. It is important to note that this dataset is focused on transition-based activities like *pouring water into a glass*, *going down the stairs*, *brushing teeth*, etc. The authors pointed out that most Convolutional Networks fail to capture long temporal patterns, compromising their overall accuracy. Instead of exploring complex DL structures to improve results, they proposed CNN and CNN^{+} architectures:

   (a) **Longer convolutional kernel-size (CNN) :**, To solve the complex HAR problem, the authors saw a potential more straightforward solution of using appropriate convolutional kernel size. For time-series data, convolutional kernel length controls temporal data-range that captures time-domain patterns (Jordao et al. 2018). Similarly, convolutional kernel width captures inter-axis patterns for neighboring accelerometer axes. The CNN architecture is illustrated in Figure 2.10) with proposed kernel (longer) dimensions.

   (b) **Orientation features (CNN^{+}) :** Authors also calculated orientation angles like *Pitch* and *Roll*, which provides estimated orientation w.r.t. original sensor orientation. They improved $\sim$8.5 pp for classes *sitting down on a chair* and *standing up from a chair*.

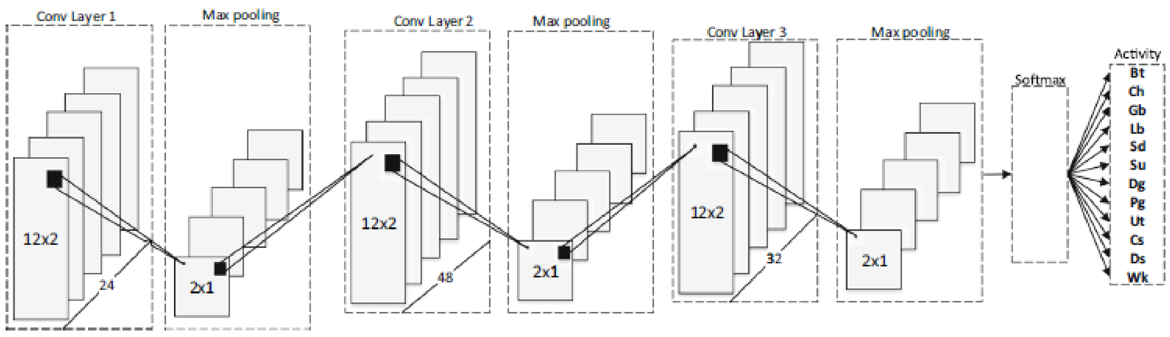


Figure 2.10: 2D CNN architecture with long kernel (Jordao et al. 2018); The convolutional kernels or filters capture the temporal patterns (by length or height) and the patterns among adjacent accelerometer-axis pairs (by width)

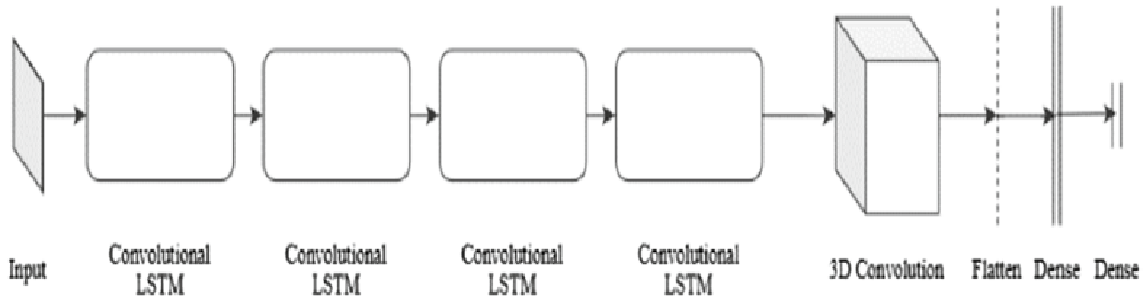


Figure 2.11: 3D CNN fed by ConvLSTM layers (Erdaş et al. 2021)

4. **Convolutional LSTM :** Erdaş et al. 2021 proposed to combine the advantages of CNN and LSTM structures. They use Convolutional LSTM (ConvLSTM) instead of CNN+LSTM architectures, as explored earlier. In ConvLSTM architecture, a convolutional structure is followed by an LSTM structure, which can be applied sequentially multiple times. This is different from CNN+LSTM architecture, where various CNN structures can be followed by multiple LSTM structures. The Figure 2.11 shows the proposed architecture, where four layers of ConvLSTM are used, and their 3D tensor output is fed to a 3D convolution layer.

2.8 Hand-crafted features: Use-case specific

As seen in the above Sections, it is essential to use information-rich features to improve the performance of HAR. Hand-crafted features are aimed to capture specific characteristics associated uniquely with each activity class (Jordao et al. 2018). With the same aim, some essential features used for HAR (in general and from a use-case viewpoint) are detailed below.

2.8.1 Signal Magnitude Features

1. *Signal Magnitude Area (SMA):* Signal Magnitude Area measures how acceleration signal magnitude varies with time. For the time intervals i, SMA is calculated using (Sukor et al. 2018):

$$\text{SMA} = \frac{1}{i}\left(\sum_{u=1}^{i}|x_u| + \sum_{u=1}^{i}|y_u| + \sum_{u=1}^{i}|z_u|\right) \tag{2.2}$$

Where x_u, y_u and z_u are the data components of the tri-axial accelerometer signal.

2. *Total Acceleration (Total Acc.):* Whenever a wearable sensor like accelerometer is used for monitoring human activity, the original sensor placement changes as a result of physical activity being performed by the subject. This means acceleration due to gravity often gets distributed over two or three signal axes. This makes it challenging to analyze the acceleration peaks with changing human position. For this, Total Acceleration, also known as Signal Magnitude Vector, is calculated over all three acceleration axes using (Sukor et al. 2018; Mannini et al. 2017):

$$\text{TotalAcc.} = \sqrt{x_i^2 + {y_i}^2 + {z_i}^2} \tag{2.3}$$

where, x_i, y_i and z_i are the i^{th} samples of tri-axial accelerometer signal.

2.8.2 Pitch, Roll, and Tilt Angle

1. *Pitch and Roll Angles:* Orientation features were calculated by Jordao et al. 2018, specifically for wrist-based HAR. To remove random noise and quantization errors from raw accelerometer data, a low-pass Butterworth filter was applied in the study producing filtered signal-data; a_x^f, a_y^f, and a_z^f. The wrist orientation features, Roll (ϕ) and Pitch (θ), are computed using:

$$\phi = \text{atan2}\left(-a_x^f, -a_z^f\right) \tag{2.4}$$

$$\theta = \text{atan2}\left(-a_y^f, \sqrt{\left(a_x^f\right)^2 + \left(a_z^f\right)^2}\right) \tag{2.5}$$

2. *Tilt Angle:* Tilt angle or the inclination of vertical axes (y) from the gravity, is calculated using (Sukor et al. 2018):

$$\text{TA} = \text{asin}\left(\frac{y_i}{\sqrt{{x_i}^2 + {y_i}^2 + {z_i}^2}}\right) \tag{2.6}$$

where, x_i, y_i and z_i are the i^{th} samples of tri-axial accelerometer signal.

2.8.3 Fragmentation Features

Window specific features were applied by Mannini et al. 2017. They defined a threshold value ($Th = 0.2 \times g$, where gravity $g = 1.96\,{}^{\text{m}}\!/\!{}_{\text{s}^2}$), using which static and dynamic instances were identified. This threshold value was used to identify which samples are 'active' or belong to dynamic activity. For every (unfiltered) sample, total acceleration (using equation 2.3) was calculated. If this value is lower than $0.4\,{}^{\text{m}}\!/\!{}_{\text{s}^2}$, the sample was considered as static otherwise dynamic. This way, Andrea et al., calculated the number of active samples for one input window, threshold crosses in one window, etc. These Fragmentation features were found to be useful for ankle-mounted acceleration sensors.

2.9 Limitations of the existing research

This Section discusses limitations of the state-of-art research available for this project's use-case, i.e., single ankle-mounted tri-axial accelerometer. As seen in previous Sections, very little literature provides meaningful insights for ankle-mounted accelerometer HAR. The unavailability of research and datasets leads to a lack of comparable benchmark results and implementations. Apart from having incomparable datasets, the difference in sampling rate is also an important thing to consider. Some essential hand-crafted features, especially frequency-domain features, require a high sampling rate. Most research results are based on available online datasets, having a sampling frequency of 50 Hz, which is much higher than 26 Hz of in-house collected data. This may provide DL approaches the advantage of having more information; however, further domain-specific research is required to make sure if a higher sampling rate provides meaningful information for HAR or not. This part was not explored in this project.

Most of the single-sensor-based state-of-art methods are researched on the UCI-HAR dataset, which consists of smartphone data. This dataset is very different from the signal from the ankle-mounted accelerometer versus the waist or hip-mounted accelerometer, which has much restricted movement than the human ankle. It is also essential to consider the possibility of sitting and standing classes being correctly classified with high accuracy due to the position-specific smartphone orientation feature (as discussed for old Mentech architecture in Section 2.7). The confusion matrix of existing research clearly shows that sitting and standing classes are the most confused activity classes. Since this kind of activity-specific orientation change will not occur in ankle-mounted sensors, it can be expected that the models explored might fail for our use case.

Previous Sections also show that even after using state-of-art DL techniques for HAR, to improve accuracy further, hand-crafted features are required. This requires prior domain knowledge to extract meaningful features from raw data (Ha et al. 2016). Hand-crafted features also come at the computational cost of real-time feature calculations (Gholamrezaii et al. 2021). A balance between accuracy and computational resources spent should be maintained for real-time applications. DL methods are also susceptible to any noise found in the training dataset (Z. Chen et al. 2018). This makes data collection for HAR a problematic task.

Last but not least, for all discussed literature (relatable to the project's use-case), low accuracy for sitting and standing classes (≈83%) remained an open problem.

2.10 Discussion

Based on the literature review, there are many aspects to consider while designing a (scalable) solution for a given HAR problem. Before we have a closer look at the design choices made, let's briefly revisit the Research Questions discussed in Section 1.1:

1. **What are the traditional approaches and state-of-art methods used for developing a HAR model using inertial signals?**
   [Discussed in Sections 2.3.1, 2.4, 2.5, 2.6, 2.7]
   Traditional approaches to solve HAR are often made by first calculating the hand-crafted features (like time-domain, frequency-domain, statistical, orientation, etc.) and then feeding this information to the classifiers (like SVM, KNN, RF, DT, etc.). State-of-art methods for HAR are based on utilizing DL architectures like CNN, LSTM and their combinations.

2. **What are the limitations of the explored approaches?**
   [Discussed in Sections 2.3.2, 2.9]
   Limitations of Traditional approaches are listed below:
   (a) Rely heavily on hand-crafted features, whose calculation takes lots of data-processing time and domain knowledge.
   (b) The use of hand-crafted features followed by shallow ML architectures can lead to loss of inherent signal information.
   (c) Low sitting/standing accuracy (≈75%-80%).

Limitations of state-of-art approaches are listed below:

(a) DL models require a large dataset to train upon.

(b) Even the most negligible noise present in the training data can mislead DL models, leading to poor performance.

(c) Low sitting/standing accuracy ($\approx$85%).

3. **What are the most important and commonly used signals for HAR?**
   [Discussed in Sections 2.1, 2.2, 2.8.2]
   3D inertial signals from Accelerometer, Magnetometer and Gyrometer combined with physiological signals like ECG and Heart-rate were most widely used for HAR by researchers. Other potential signals apart from those discussed in Chapter 2 are Pressure and Orientation signals. In the paper Merry et al. 2019, the authors studied the planar pressure distributions for classes sitting, standing, and walking. They used shoe insole Pressure-measurements to collect and study the different regions of the human foot by creating a footprint. Orientation of the sensor can also provide meaningful features to perform HAR, but multiple inertial sensors are required to do so correctly. Even with the required 9-Axis sensors and signals, calculating the correct orientation of the sensor in real-time can be computationally costly if not done correctly. A smart sensor by 'Bosch Sensortec' does precisely the same to provide real-time sensor orientation. Since these require significant hardware changes in SentiSock, these approaches were skipped for this project.

4. **What pre-processing steps should be applied to filter the input signal to remove noise as well as preserve essential features?**
   [Discussed in Sections 2.2, 3.3]
   The online available HAR datasets use a low-pass Butterworth filter to remove high-frequency noise expected in an acceleration signal. The same was used by several other researchers who collected an in-house dataset. It is equally important to avoid any kind of noise during the data-collection procedure. This is because DL models (this project's selected approach) rely on labeled input data for finding important features (Z. Chen et al. 2018). Since datasets for HAR are usually small, the quality of the dataset is even more crucial. Data pre-processing done for in-house data collection is discussed in Chapter 3.

5. **What are the essential features that can be extracted from accelerometer signal data?**
   [Discussed in Sections 2.3.1, 2.8]
   Essential features for HAR using multiple sensors and signals are summarized in Table 2.2. Selected hand-crafted features meaningful from this project's perspective are discussed in Section 2.8.

6. **Which HAR model architecture is most suitable, keeping in mind the HUME specifications?**
   [Discussed in Sections 2.9, 4.2]
   For this project, Deep Learning architectures are selected since (i) Calculating many hand-crafted features in real-time is computationally costly, (ii) The accuracy bottleneck observed for sitting/-standing classes. The proposed architecture utilizes Convolutional Neural Network architectures which are discussed further in Chapter 4.

7. **How can we make the model perform efficiently in real-time?**
   [Discussed in Section 4.2]
   Limiting the hand-crafted features only to include orientation features reduces the run-time complexity due to reduced data pre-processing. Other architectural choices made (like the use of long convolutional kernels, replacing max-pooling layers with a stride of 1 in convolutional networks) ensures that the model architecture is efficient. To further improve the real-time efficiency of the model, concepts of Data-Engineering (using Spark) can be applied, which are discussed more in Chapter 7.

8. **What are the requirements for collecting a good quality dataset (i.e., having important physiological features and least possible noise) from an ankle-mounted accelerometer for classification models?**
   [Discussed in Sections 2.2.4, 3.2]
   To collect a quality in-house dataset, essential steps were taken, which are discussed further in Chapter 3. The main requirements found are listed below:

   (a) Inclusion of different environments while collecting the dataset: It was found that including various kinds of chairs, postures, walking pace, etc., can improve the quality of the dataset being collected.

   (b) Avoid any possible noise that can be expected during the data-collection process: It is important to note here that, for any moment where the input signal does not belong to the activity being labeled/collected, it is termed as **noise**. This means, every time a subject moves during the data-collection of static activities, it will lead to the appearance of noise in the acceleration signal.

9. **How can we integrate the final model with HUME?**
   [Discussed in Section 4.1]
   HUME currently uses a Binary model for static and dynamic activity classification, where sitting and standing classes are grouped into a single static activity class. At the same time, walking is a part of the dynamic class. Either the proposed model architecture can replace the current static and dynamic model. For this, it is recommended to use smaller window sizes in a real-time setting. To keep the current static and dynamic architecture, the same proposed model can be used for binary classification of sitting and standing classes after the HUME model has made static-activity decisions. Since the HUME implementation uses older TensorFlow and sklearn, the current model code needs to be downgraded accordingly to make it compatible with HUME.

3 Datasets

Deep learning (DL) methods require a large training dataset to achieve their best capable performance. This data should be carefully collected and labeled to avoid the introduction of any noise. This is important because DL methods extensively try to find meaningful features from available data, including noise, if present, which can lead to poor performance (Z. Chen et al. 2018). This makes collecting datasets for HAR time-consuming. If any noise is introduced during the collection process, it is tedious to remove it (since handling 3D tensors are difficult). Online available datasets are hence used by researchers as they act as a reliable data source (i.e., with minimal noise). However, there are only a few online datasets that have used ankle-mounted accelerometers for HAR data collection.

In this project, the PAMAP2 dataset collected by the University of California Irvine (UCI) is used (Reiss 2012). It contains 18 activities, but we will talk about only three; sitting, standing, and walking. A total of 9 volunteers participated in this data collection experiment, but for only eight volunteers, labeled data (for sitting, standing, walking classes) was available. These eight subjects were selected for this project. An in-house database was also collected at Mentech lab for these three activity classes. Eleven volunteers participated in the data collection procedure, but only 5 participants' data was available and used for this project due to corrupted signals received for some sessions. These datasets were also integrated to study the performance of the classification model on the combined, larger dataset (called Megabase in this report). Details for these datasets, collecting data from ankle mounted 3D accelerometer, for three activity classes are summarized in the Table 3.1.

DATASET	#VOLUNTEERS	SAMPLING RATE	TOTAL #SAMPLES
PAMAP2	8	100 Hz	610200
Mentech	5	26 Hz	477710
Megabase	13	26 Hz	630255

Table 3.1: Specifications of all the datasets used for this project

3.1 The PAMAP2 dataset

In the paper Reiss 2012, the PAMAP2 Physical Activity Monitoring dataset by UCI contains data collected from 3 inertial measurement units (3D acceleration, 3D gyroscope, 3D magnetometer) and a heart rate monitor. All the subjects wore wireless Inertial Measurement Unit (IMU) sensors over their wrist on the dominant arm, chest, and the dominant side's ankle. The heart rate monitor was placed alongside the chest IMU, with a sampling rate of about 9 Hz. Eighteen different Activities of Daily Living (ADL), such as walking, cycling, playing soccer, etc., were performed by subjects during this data collection.

Ankle data from the PAMAP2 dataset was selected to generate results of the proposed solution architecture for HAR. This makes it possible to replicate the model performance achieved in this project, providing benchmark results for future research (Saez et al. 2016), which is not possible using in-house collected datasets. Subject-wise details and the total number of data samples collected per subject for every activity class can be found in Table 3.2. The Figure 3.1 shows how the 3D acceleration signal changes for different activity classes. Notice how x-y-z signals differ among themselves for the same activity class and other classes.

ID	SEX	AGE	DOMINANT SIDE	#SAMPLES		
				SIT	STAND	WALK
1	Male	27	right	23475	21642	21955
2	Female	25	right	22323	25436	32113
3	Male	31	right	28752	20435	28637
4	Male	24	right	25487	24680	31497
5	Male	26	right	26808	22089	31487
6	Male	26	right	22823	24114	25511
7	Male	23	right	12280	25737	33407
8	Male	32	left	22920	25137	31047

Table 3.2: Details of all subjects from Mentech dataset

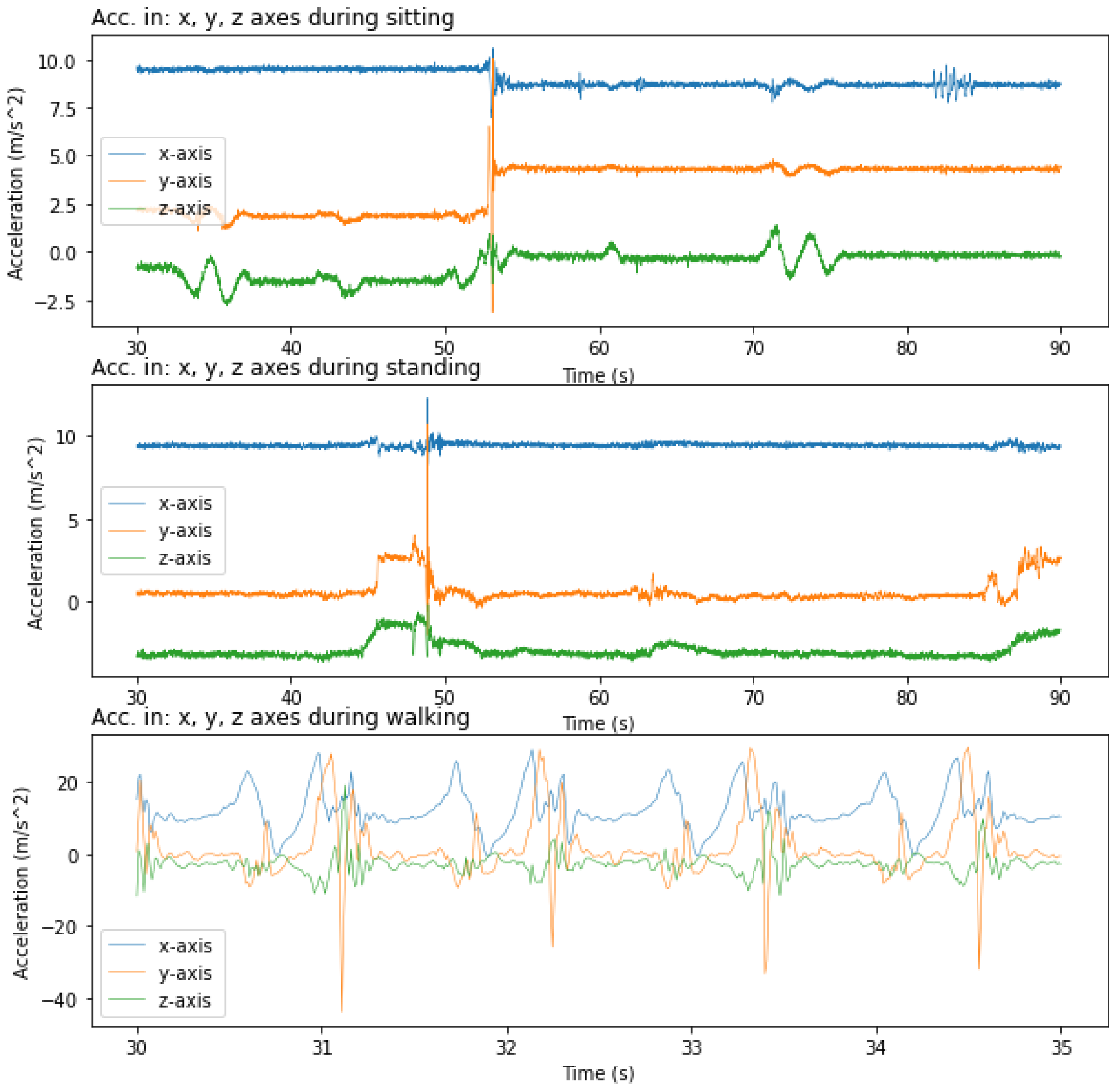


Figure 3.1: Comparison of 3D acceleration signal for Sitting, Standing and Walking activities for one subject from PAMAP2 dataset

ID	SEX	AGE	DOMINANT SIDE	#SAMPLES		
				SIT	STAND	WALK
1	Male	23	left	59240	21740	18720
2	Female	18	right	68490	17500	14550
3	Male	21	right	61640	19590	15400
5	Male	20	right	59670	18130	14920
6	Female	27	right	57310	18620	12190

Table 3.3: Details of all subjects from Mentech dataset

3.2 Mentech dataset (in-house dataset)

The in-house dataset was collected at Mentech lab for activity recognition. Since other wearable sensors are also being explored at Mentech, two wearable sensors collect 3D acceleration data. One sensor was placed at the subject's dominant wrist and the second at the subject's dominant ankle; however, only an ankle sensor was selected for this project.

Motivated by the ideas discussed in Section 2.2.4, various postures and chair styles were considered to affect different human-ankle positions. Particular focus was on finding the ankle-specific (sitting/standing) positions that can lead to noticeable changes in acceleration across different axes. The aim was to collect a diverse database, including various sitting, standing, and walking positions. Various research also suggests that noise can be introduced in collected data if 'interruptions' of any kind are caused while live-recording of an activity label (discussed in Section 2.2.4). For example, a volunteer while walking is interrupted by someone for a talk; volunteer stops to open a door while walking, volunteer starts walking due to interruption caused while recording standing label, etc. Labels recorded for such instances were discarded on the spot. To record the time-stamps for activity labels, an android based application, *aTimeLogger*, was used.

Since this data was collected by Data Analyst from Mentech Lab, a document was created capturing the essential aspects to consider while collecting this data for HAR. This document was also provided as a pamphlet to the volunteer to make them comfortable with the data-collection process, ensuring easy onboarding. This was also done to make sure volunteers also know how to contribute to avoid noise-collection in the process. Different positions for sitting, standing, and walking were also proposed, and for every varied position, labeled data of 1 minute was collected. This means that the labeled data does not include transitions from one position to another. This document containing the overall experiment setup used for in-house data collection can be found in Appendix A. Patient-wise details per activity class can be found in Table 3.3, and the 3D acceleration signal for different activity classes can be seen in Figure 3.2.

The PAMAP2 dataset can act as a reference for how a clean acceleration signal looks like. If one compares Figure 3.1 and Figure 3.2, it can be noticed that the Mentech dataset has more (random) peaks for sitting and standing classes than compared to the PAMAP2 dataset. To investigate the root cause of this, data from both datasets were further processed and analyzed.

3.3 Data Pre-processing

Despite taking many precautions in recording a noise-free in-house database, an error was made while recording time-stamps for labels. The Android app, which was used to record labels (or activity time-stamps), was used with default time settings to collect minute-specific data. This led to the loss of 'second' information from recorded time-stamps for activity labels. This creates a high possibility of noise in recorded labels; for example, a sitting label (recorded for one minute using a timer) can include instances of the person walking to prepare for their next activity). As discussed in the literature, DL

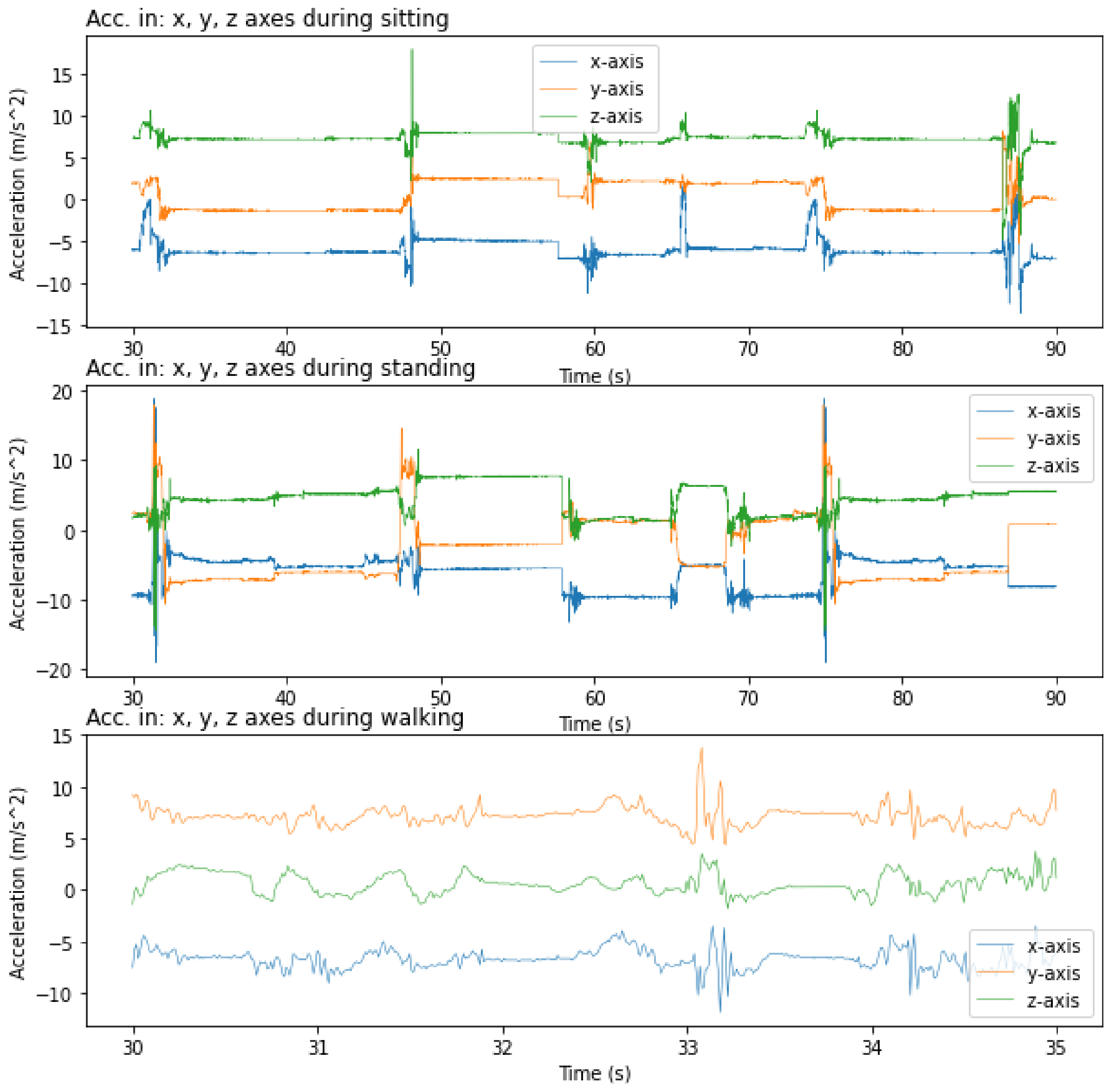


Figure 3.2: Comparison of 3D acceleration signal for Sitting, Standing and Walking activities for one subject from Mentech dataset

algorithms suffer a lot from even small instances of noise introduced in data (Z. Chen et al. 2018). Hence, it is essential to understand how a situation like this can lead to a DL model misclassifying *sitting* as *walking*. Mannini et al. 2017 Also faced similar issues, where the model was misclassifying, for example, *room cleaning* with *sedentary* because there were instances in room-cleaning labels, where the participant was standing for a while.

To overcome this possibility and maintain the quality of the collected dataset, extensive data cleaning is required. Before data cleaning, it is necessary to visualize the data signals and understand which parts of the signal can potentially be noise. Data pre-processing steps relevant for selected datasets are mentioned below, which were dealt with before data-plotting:

1. PAMAP2 dataset:
    - A subset from the original dataset was created, containing only ankle measurements. Most of the columns in the data file (per subject) were dropped, and only five values were kept; time-stamp, label, and ankle acceleration values (x,y,z).
    - This subset was further processed to filter out rows of data that are not labeled as either sitting, standing, or walking.
2. Mentech Dataset:
    - The time-stamp labels (start and end time-recordings of the activity class collected using the *aTimeLogger* app) were extracted for every variation of the recorded activity. Then the collected data signals (from ankle and wrist-mounted sensors) were processed to keep the activity-specific data, using the extracted timestamp-label information. This means only the labeled data was maintained, while the instances where the subject is relaxing or preparing for the next activity were removed in this step.
    - Then, both the sensor signals were visualized to study and compare the *sitting* activity class for all the patients. This was done to distinguish between the ankle and wrist data signals as the filenames given during the recording sessions were not informative enough. Then the relevant ankle-specific acceleration data files were kept, and the wrist-specific data files were dropped.

3.3.1 Data plotting

For data analysis that will be done using data-plotting, it is vital to finalize what we want to compare. The aim is to finalize the variables (to the plot) and plot them, which can help answer only one crucial question: *'Precisely which parts of the acceleration signal should be treated as noise?'*. These parameters can be found below:

1. **Subject-wise comparison:** It is important to compare how ankle-acceleration data differs for every human. For example, an individual might walk differently due to different health conditions.
2. **Acceleration-signal component comparison:** 3D acceleration signal has three components which are the values across the x, y, and z-axis. In addition to these, total acceleration (calculated using Equation (2.8.1.2.3)) was also selected.
3. **Activity-wise comparison:** It is important to compare how the acceleration components differ for sitting, standing, and walking activity classes.
4. **Dataset-wise comparison:** Since we know some noise is introduced in the Mentech dataset, the PAMAP2 is treated as a data source having 'less' noise. However, it is essential to realize that the PAMAP2 dataset can also have some noise.

From the points above, it can be estimated that at least 168 ($= 14 \times 4 \times 3$) plots are required to find out the answer to our question. This is the 'tedious' part of removing noise mentioned at the start of this Chapter. These plots were further grouped and combined to make the required comparison as and when needed. An example plot can be seen in Figures 3.3 and 3.4, where two subjects' data are compared simultaneously for sitting, standing, and walking classes. Summarizing data this way helps in efficient data comparison.

It can be noticed in Figures 3.3 and 3.4, that the acceleration signal for the dynamic-activity class (walking) shows a pattern having high-frequency peaks, which is very different from static classes like sitting and standing. It can further be seen why classes like sitting and standing are tough to distinguish as they show similar patterns.

3.3.2 Data analysis

It is crucial to understand the kind of noises identified during the signal plotting step to clean the data. For both datasets, no hardware-specific noise was found. Other main observations are listed below:

1. **Small 'walking' like instances in 'Sitting' and 'Standing' classes:**
   For mostly all subjects of the PAMAP2 dataset, it was observed that some noise similar to that of a slow-paced walk is present at the starting of 'Sitting' and 'Standing' classes. Similar noise was observed in the Mentech dataset for both the classes; however, noise instances lasted longer and repeatedly occurred in the data signals. This was because of how the in-house dataset was collected, in combination with the label error made.

2. **Sudden movements or 'jerks' all classes:**
   For both the datasets, sudden high peaks were observed, which can be a result of some sudden ankle-specific movement. For the 'Sitting' and the 'Standing' classes, these peaks were often surrounded by clusters of smaller peaks identified as noise. The tricky part about this type of noise is that they occur in random patterns, and the maximum peak value differs a lot per subject. Another tricky part here is that for the 'Walking' class, if such jerks occur, they are almost impossible to identify using visual signal-plotting. This is because, for the *walking* activity, the signal pattern is very similar to that of a *jerk* instance.

3. **Less data for some subjects:**
   For some subjects, there are fewer samples for the 'Sitting' and the 'Standing' classes. Some subjects can also follow a particular motion pattern, which might not be a jerk but just how a subject performs the activity. Such instances should be identified as 'outlier' subjects (with imbalanced activity samples or showing particular activity patterns). In DL models, often data is divided into training and testing sets subject-wise to study the model performance on subjects it has not been trained upon. If the 'outlier' subject is kept in the testing dataset (especially while using a small-sized dataset with few subjects), it can lead to poor model performance. Hence, it is essential to ensure that some parts of such subjects are also included in the training dataset.

When a combination of the above-discussed noise types (that do not have a fixed pattern) occurs in the input signal, it is very time-consuming to remove them by visually identifying the start/end time-stamps for the noises. Deleting noise this way also poses a risk of throwing away some meaningful information specific to that activity. The concept of standard deviation was utilized to ensure that essential features were not lost during data cleaning.

Standard deviation (std. dev.) is a measure of how the data-values deviate from their mean value. Let x_i be the i^{th} sample-value of the signal, σ be the standard deviation, μ be the mean, and N be the total number of samples in the signal, then σ^2 represents the variance or power of these data-values. It is calculated as follows:

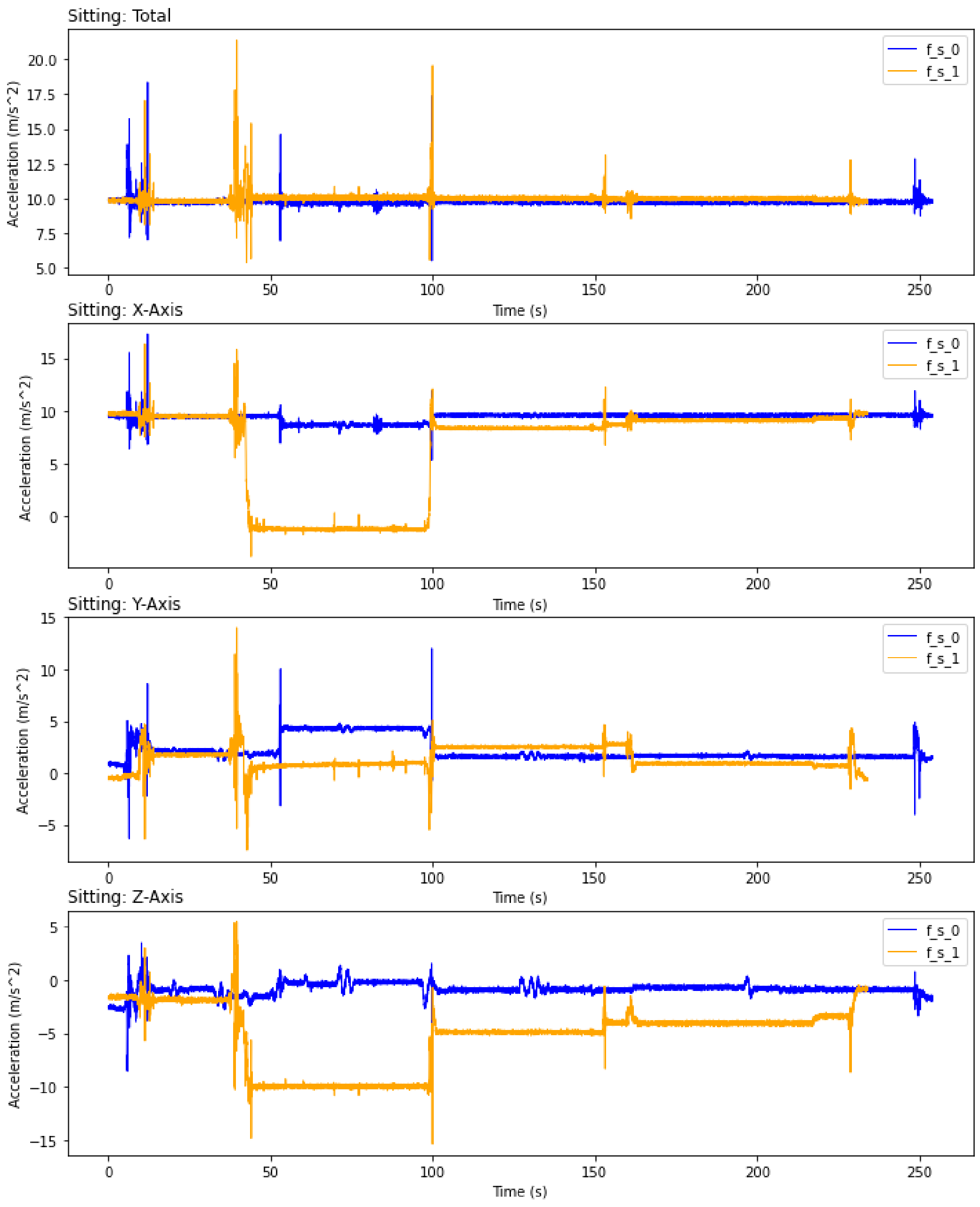


Figure 3.3: Comparison of acceleration-signal components for Sitting activity class of 2 subjects (named f_s_0 and f_s_1) from PAMAP2 dataset. The first row shows the Total acceleration values, followed by the X, Y, and Z axes values.

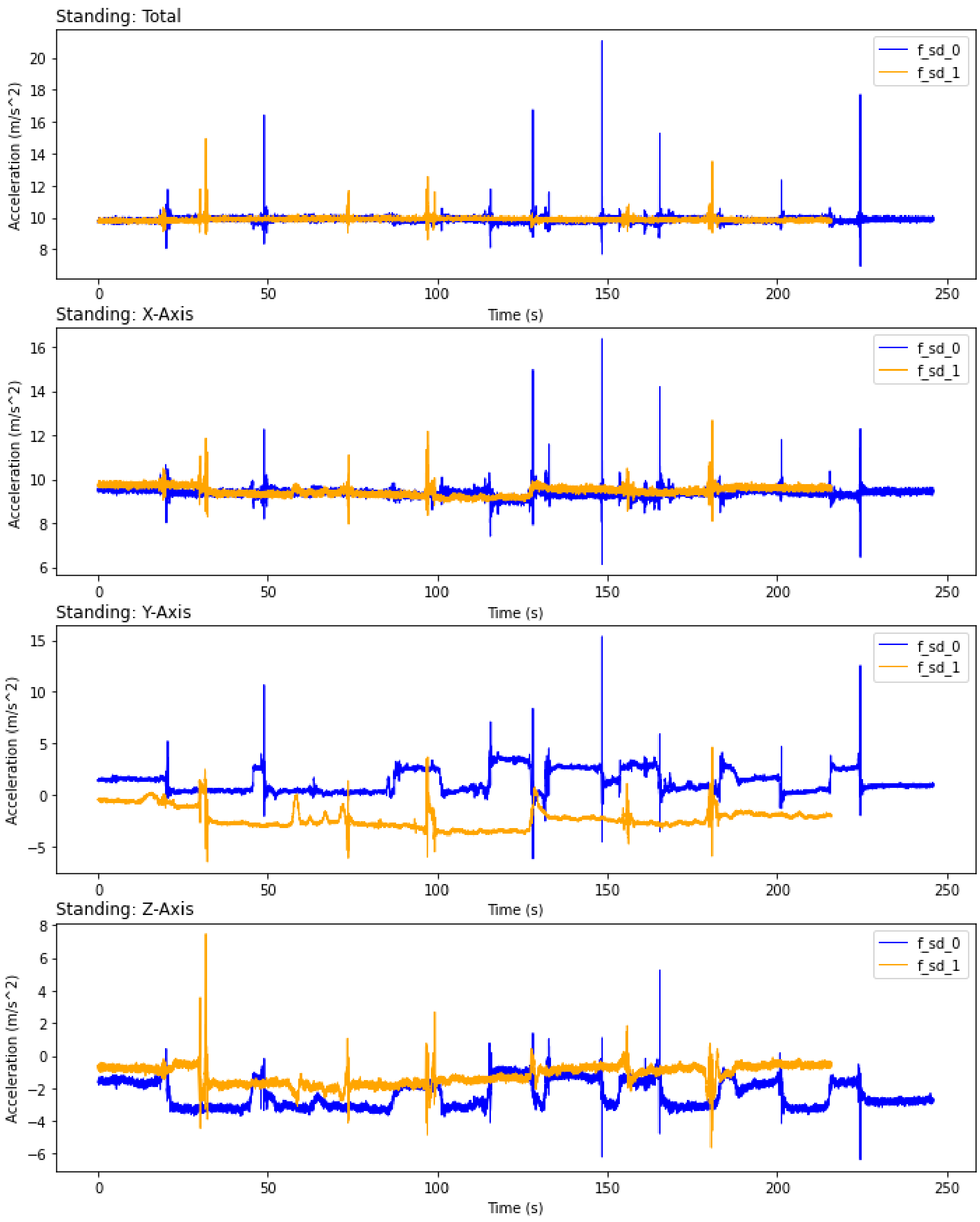


Figure 3.4: Comparison of acceleration-signal components for Standing activity class of 2 subjects (named f_sd_0 and f_sd_1) from PAMAP2 dataset. The first row shows the Total acceleration values, followed by the X, Y, and Z axes values.

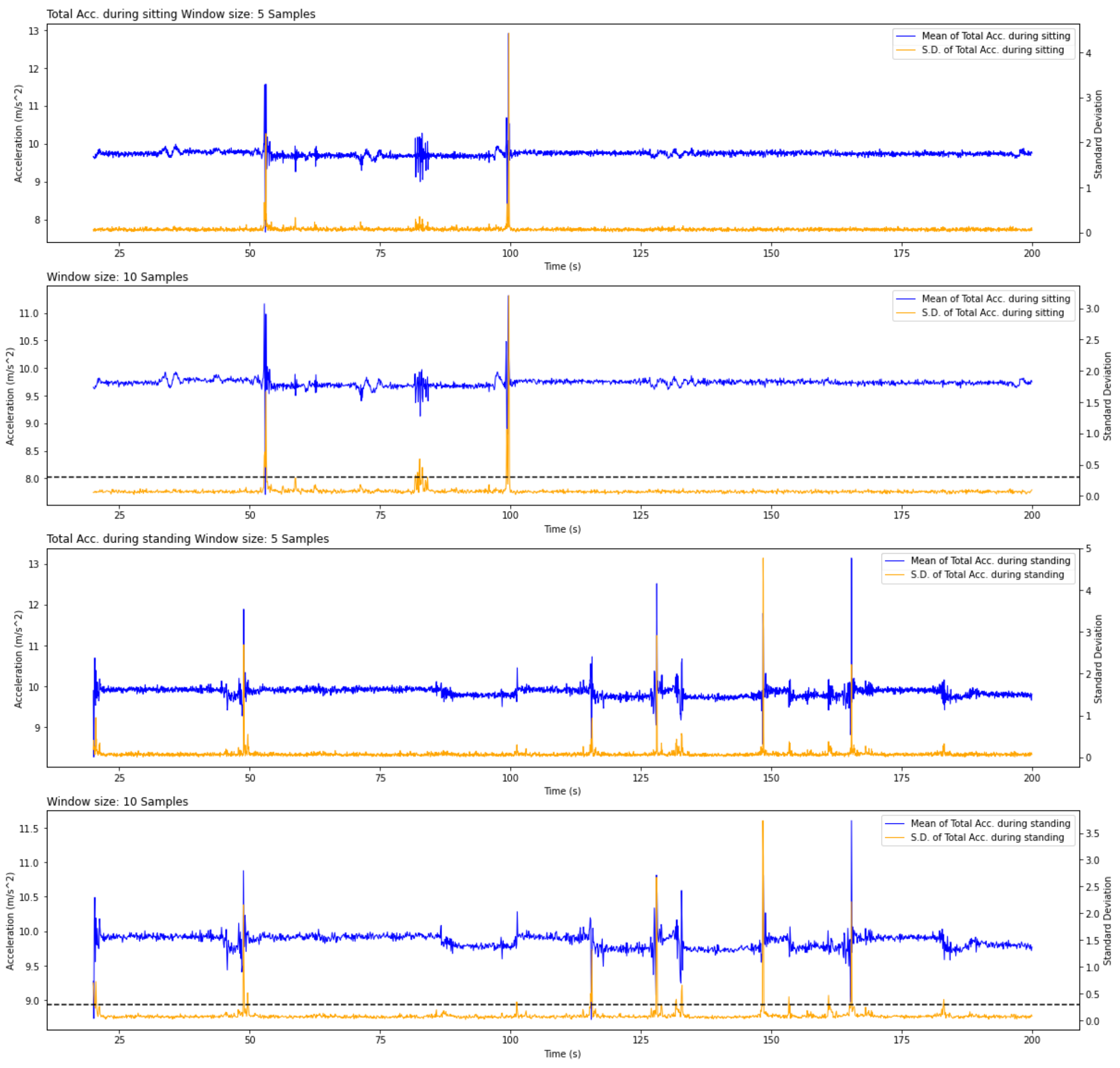


Figure 3.5: Plots used for Data analysis using Standard Deviation and Mean of Total acceleration signal using different window lengths (PAMAP2 dataset). The dotted line for window-size 10 shows the standard deviation threshold value of 0.3, used for data-cleaning.

$$\sigma^2 = \frac{1}{N-1}\sum_{i=0}^{N-1}(x_i - \mu)^2 \tag{3.1}$$

$$\mu = \frac{1}{N}\sum_{i=0}^{N-1} x_i \tag{3.2}$$

Using the acceleration (x, y, z) data-values contained in one sample, Total acceleration (discussed in Subsection 2.3) was calculated. To use the above concepts, the total acceleration signal was processed in the form of non-overlapping mini windows in time. For every window, mean and standard deviation values were calculated and stored. The data-plotting implementation was further used for extensive signal-plotting to compare the standard deviation outputs using different window lengths. Such example can be seen in Figures 3.5 and 3.6, where windows of lengths 5 and 10 are compared.

Notice in the Figures 3.5 and 3.6, how the standard deviation is higher for the signal where noise in-

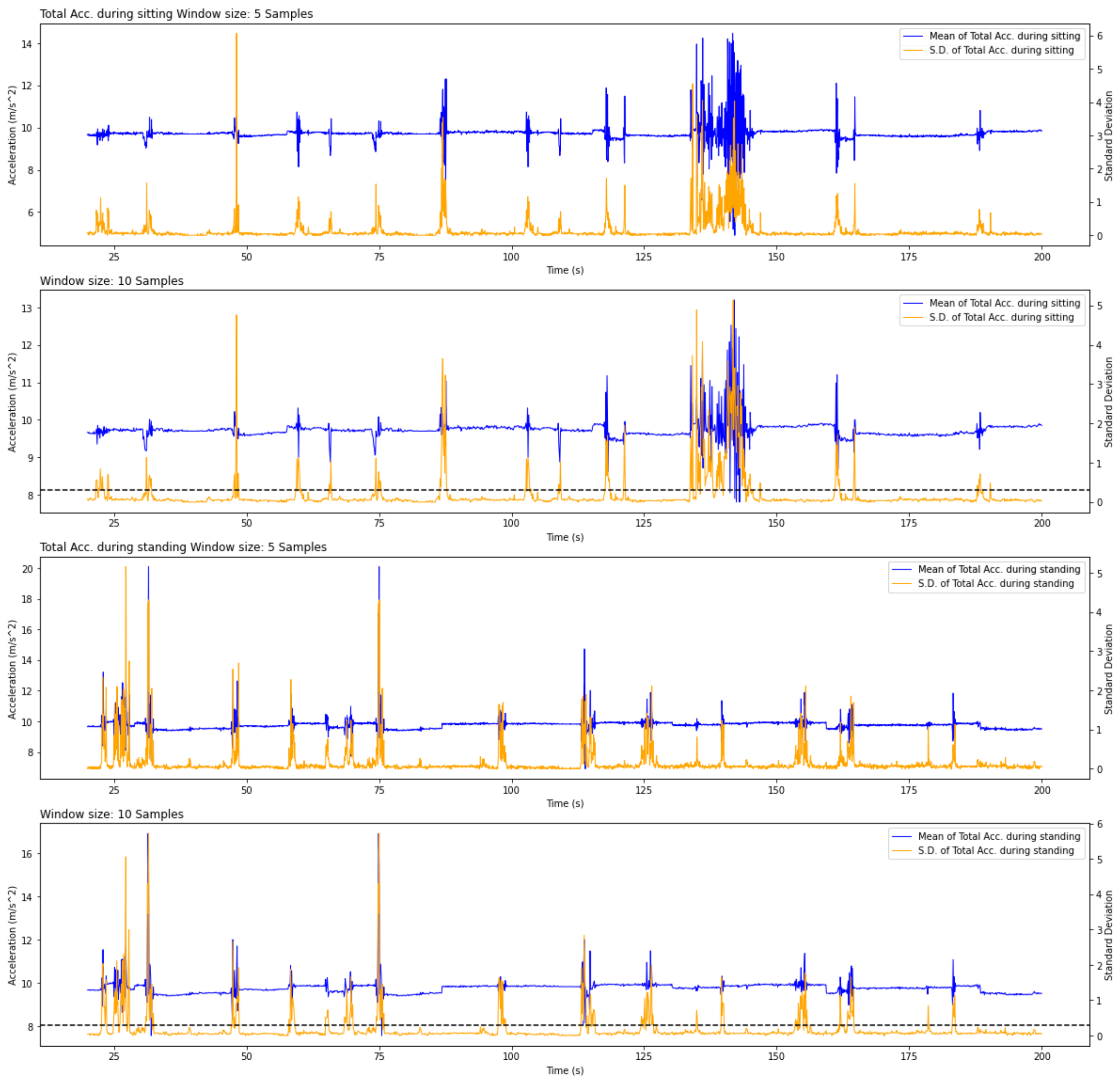


Figure 3.6: Plots used for Data analysis using Standard Deviation and Mean of Total acceleration signal using different window lengths (Mentech dataset). The dotted line for window-size 10 shows the standard deviation threshold value of 0.3, used for data-cleaning.

stances occur. It confirms that these mini-windows can be used to process the signal and remove random noise from it. These standard deviation-applied mini-windows will now be referred to as **filters**, as they will filter out the unwanted data.

3.3.3 Data cleaning

Filters of different lengths were implemented, plotted, and compared to see how well the higher values of standard deviation overlaps with the noise found in the signal. Special attention was given to find the appropriate **threshold** values for standard deviation, indicating the decision of which samples to keep in the signal and which to discard. These thresholds were set differently for every activity class but were aimed to be generic for all subjects and datasets. After comparison, a filter with a length of 10 samples was selected with a threshold value '0.3' for sitting and standing classes, while no filter was applied to the walking class. Figures 3.5 and 3.6 also shows how this filter works and how the threshold value of 0.3 (with window size 10) is used to identify random noise instances. The original and final signals can be compared in Figures 3.7 and 3.8 for PAMAP2 and Mentech datasets, respectively.

3.3.4 Data manipulation

Different sensor hardware can have different default axis values defined for the acceleration signal. This configuration plays a vital role in calculating hand-crafted features that are gravity-dependent. The filtered acceleration data was further used to analyze the gravitational component of acceleration data. It was studied how this component varies for the different axis of acceleration for a periodic activity such as walking. Using this information, sensor orientation or the original x-y-z-axis of acceleration signal was identified. Original sensor orientation for both datasets is shown in Figure 3.9. Necessary data-manipulation steps used for this project are listed below:

1. **Axis alignment:** When two datasets are combined to create a larger dataset, it is important to align their x-y-z-axis to ensure that the model does not confuse inter-axis features. While creating Megabase, the PAMAP2 axis was altered to make it compatible with the Mentech database.

2. **Downsampling:** Since we are working with time-series data, it is crucial to maintain the same sampling rate at which input signal data is arranged. While creating Megabase, the PAMAP2 signal was downsampled to 26 Hz.

3. **hand-crafted features:** Three angles required for calculating the orientation-related features as discussed in the Subsection 2.8.2 were selected and implemented for this project.

4. **Window size:** Before feeding the input data into classification models like CNN and LSTM, they are further divided into smaller chunks of data with respect to the time domain. In most of the literature explored, smaller window-lengths were used, the most common one being 2.56 seconds (Gholamrezaii et al. 2021; Ignatov 2018). To understand this, one should check how long one instance of human activity (being classified) lasts. For *sitting*, *standing*, and *walking* activity classes, valid instances can last anywhere from 1 to 3 seconds. So the window size of a dataset should be equal to the sampling rate of the dataset multiplied by the activity-span duration. Different activity time-spans that were used and compared for this project are:

   - PAMAP2 dataset : 0.64, 1.28, 2.56, and 5.12 seconds or 64, 128, 256, and 512 samples.
   - Mentech dataset : 0.64, 1.28, 2.56, 5.12, 10.24, and 20.48 seconds or 16, 32, 64, 128, 256, and 512 samples.

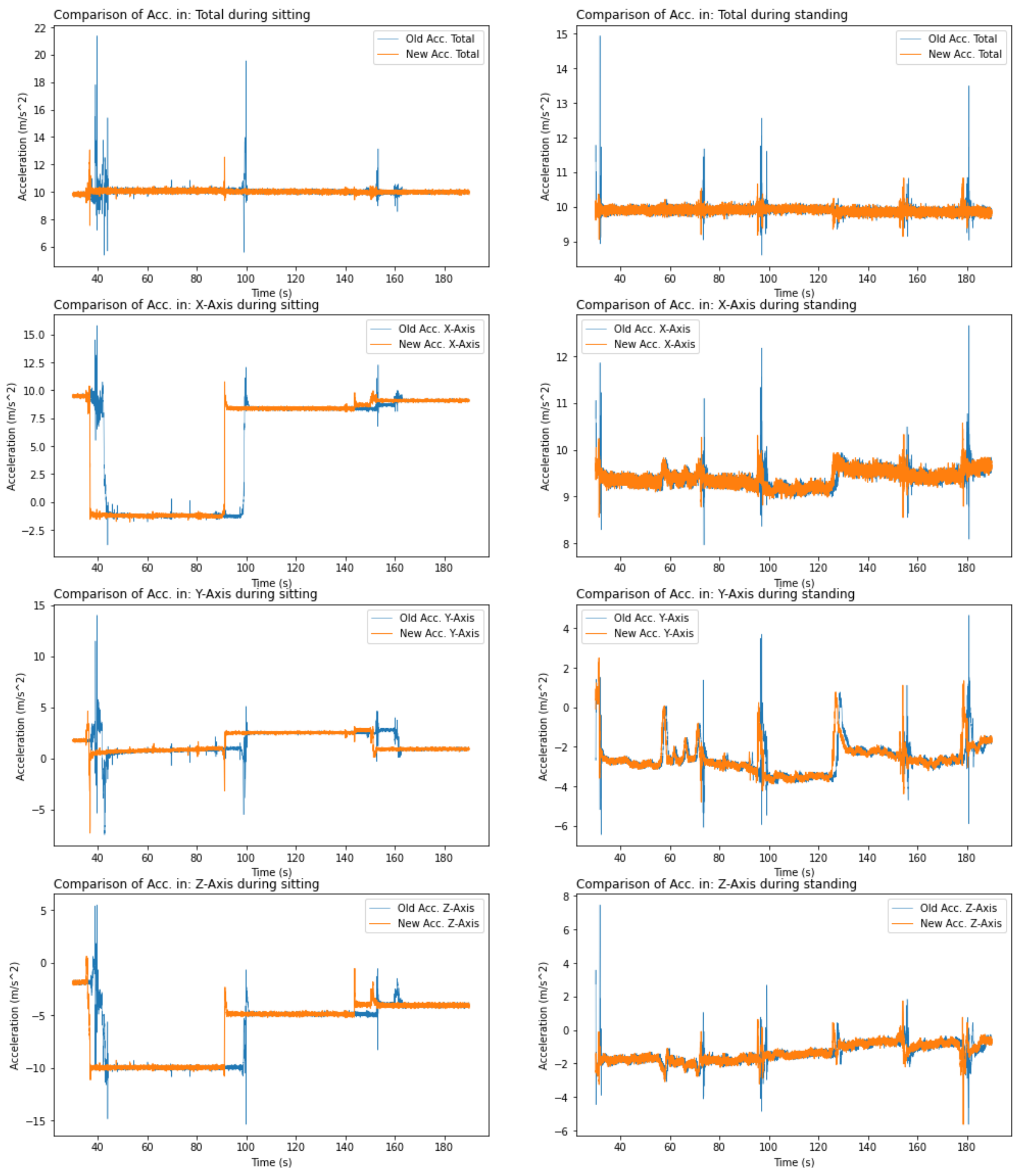


Figure 3.7: All plots compare the Original (old) and the Filtered (new) acceleration signals, for one subject from PAMAP2 dataset. The first row shows the Total acceleration values, followed by the X, Y, and Z axes values. The first column is for Sitting class, while second is for Standing class.

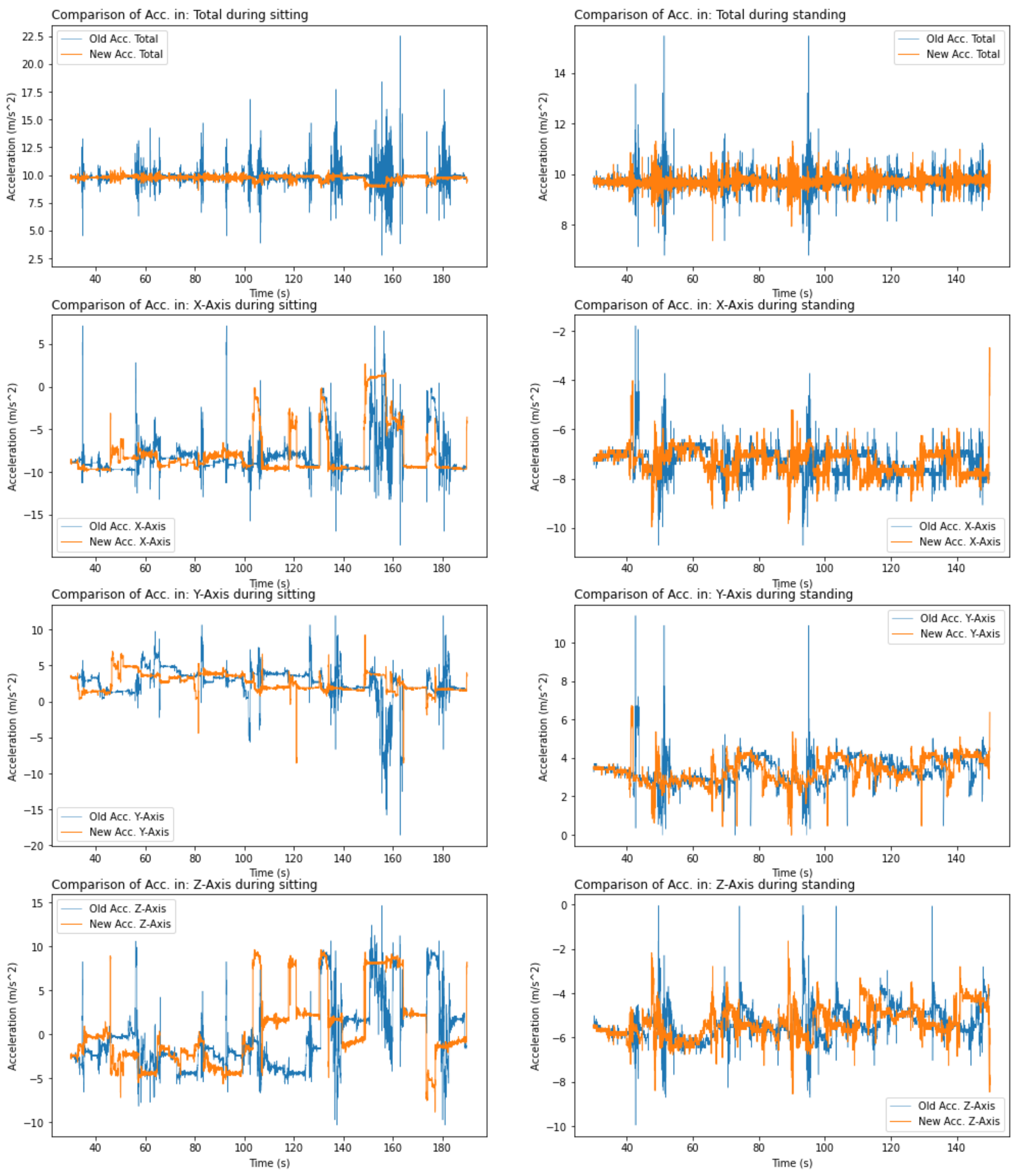


Figure 3.8: All plots compare the Original (old) and the Filtered (new) acceleration signals, for one subject from Mentech dataset. The first row shows the Total acceleration values, followed by the X, Y, and Z axes values. The first column is for Sitting class, while second is for Standing class.

.

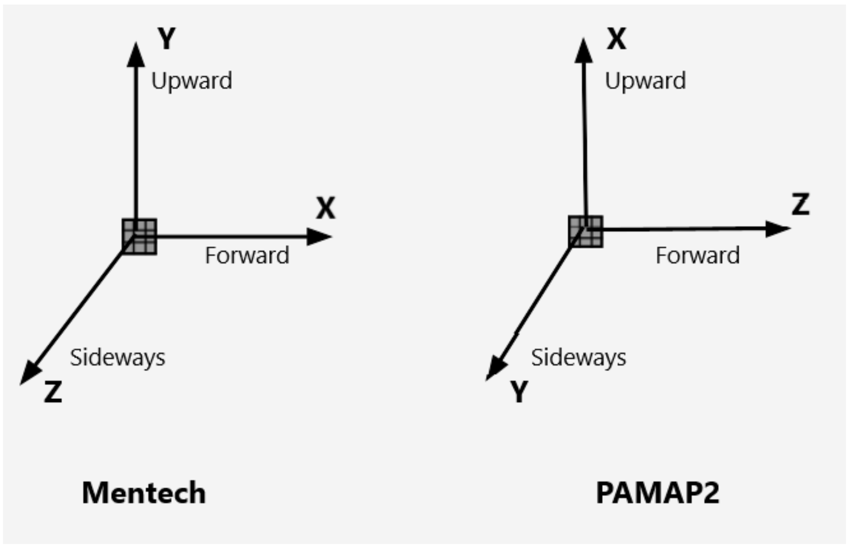


Figure 3.9: Sensor orientation for Mentech and PAMAP2 ankle accelerometers

3.4 Data Fragmentation

Once data pre-processing is completed, based on the state-of-art methods discussed in Chapter 2, it is best to prepare the input signal data to feed into the Convolutional network. Once the signal data is clean from noise and features are calculated, it can be reshaped as per the input specifications of the classification model. This process is called **Data Fragmentation**. Conv2D and Conv3D layers from an open-source software library (TensorFlow) are being used for this project.

The Conv2D layers takes a 4D Tensor as input of shape: (batch_size, height, width, channels), where for this project, batch_size = 1, height = window size, width = number of acceleration components (like x, y, z, orientation feature1, feature2, feature3), and channels = 1 (because only acceleration channel is present). To use this Conv2D layer, a **3D Tensor** is created of shape (N, height, width) where N is the total number of input batches, as illustrated in Figure 3.10. This 3D Tensor was converted to create the input 4D Tensor with degenerate dimension, i.e., shape (N, height, width,1). It is useful to know that TensorFlow Conv2D/Conv3D layer provides 2D/3D kernels, which can process the multivariate time-series data (as discussed in Subsection 2.4.1). The kernel size of these layers further signifies which part of the 3D input Tensor the model will work, or more specifically. For example, the 2D kernel of the Conv2D layer slides over (a part of) the 3D input data or 3D Tensor. Although the input data dimension is 3D, the 2D convolutional filter can only move in two directions along with the height and width of input data.

This way, by using 3D Tensors combined with different convolutional kernel-size, we can implement 1D, 2D, and 3D CNN architectures without modifying this 3D input tensor. For example, if we use 3D CNN architecture and set the kernel size as (3,1,1), we are working on three samples of only one acceleration-axis and one channel at a time (hence a *1D CNN* implementation). Similarly, if we set the kernel size as (3,3,1), we work on all three acceleration-axes for one channel (hence a *2D CNN* implementation). The 9-Axis inertial data can be thus handled by a kernel-size of (3,3,3). Although 3D tensors are complicated to work with, they offer many benefits:

1. They save implementation time required for data preparation of different (input-shape specific) CNN architectures. This provides the flexibility to explore different architectures without excessive data manipulations to achieve compatibility with CNN structures.

2. They provide an appropriate data format, enabling easy data-handling to calculate many data-fragment-related features (discussed in Subsection 2.8.3).

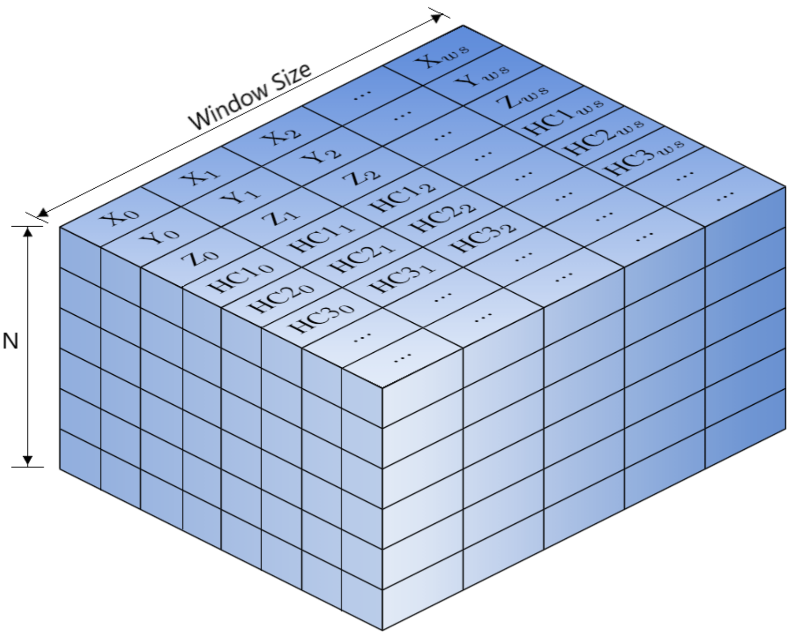

Window Size
N
X_0 X_1 X_2 ... X_{ws}
Y_0 Y_1 Y_2 ... Y_{ws}
Z_0 Z_1 Z_2 ... Z_{ws}
$HC1_0$ $HC1_1$ $HC1_2$ ... $HC1_{ws}$
$HC2_0$ $HC2_1$ $HC2_2$ ... $HC2_{ws}$
$HC3_0$ $HC3_1$ $HC3_2$ ... $HC3_{ws}$

Figure 3.10: 3D Input Tensor

3. 3D tensor structure is scalable to add important hand-crafted features. For example, a 3D tensor of size (N,256,3) (i.e., (number-of-batches, window-size, axes)) can be extended along the x-y-z axes of data to add orientation related features (discussed in Subsection 2.8.2) where the new 3D Tensor will have a size of (N,256,6).

Hence for this project, the data signal was fragmented to create 3D tensors of shape (N, window_size, 6) if hand-crafted features are being used, otherwise (N, window_size, 3). These were then converted into 4D input Tensors with degenerate dimensions to feed into Conv2D and Conv3D layers.

3.5 Implementation architecture

After gaining knowledge about *what* data pre-processing needs to be done, it is essential to gain insights into *how* it can be done. This Section discusses the implementation details used to handle data plotting and analysis and was further scaled up to perform data-manipulation tasks. Figures 3.11 and 3.12 shows the end-to-end architecture used for data pre-processing and model classification. To do extensive data-plotting and analysis, it is essential to create, map, and access the subject-wise meta-data like subject ID, subject data-label, data-length for every activity label, etc. Such meta-data was extracted and utilized to process the data signals. Meaningful choices are discussed below:

1. **Numpy ndarrays:** To handle multivariate time series data, numpy ndarrays were selected for base implementation. This includes: extracting data from *.csv* signal files, processing data in a subject-wise manner, extracting activity-wise data for every subject, and performing data fragmentation once the noise is removed from the signal.

2. **User-interactive interface:** Implementation code was kept interactive, prompting the user to make selections from pre-defined sets of choices to provide them the flexibility to train and compare model results in different parameter settings. The parameters or choices which were offered to the user are:

- choice of the dataset (s) to work upon
- choice of using filtered or non-filtered dataset (using data cleaning procedure)
- choice to include or exclude orientation features
- choices of different input window-size for data-fragmentation
- choice of classification model to train

3. **Data structures:** It was found that numpy ndarray has a limitation that 'length of elements for all rows, columns, etc., should be the same'. This is an issue since all subjects have a different total number of data samples for various activities. For this reason, python data structure, *Nested lists* were utilized. Since they allow index-based access and data manipulation, they are highly efficient. Another essential data structure used was *Dictionary*. Dictionaries are the key-value pairs that are convenient to map all subject IDs with their corresponding signal length. Combining the above data structures (and meta-data) makes it possible to make the implementation independent of the type of dataset selected by the user.

4. **Python function *globals()* versus *eval()*:** Data analysis requires efficient access to the individual subject lists and nested lists using the meta-data stored for that subject. Managing a large number of subjects with many activity labels for data analysis can result in complex code. One is often tempted to use Python's *eval()* function to fetch the individual subject lists by giving their 'list-string-name' or subject ID meta-data to avoid this complexity. If not done correctly, this is a bad practice and can expose the system to security risks and memory leaks. An easy alternate solution to this problem is creating global variables and using the *globals()* function to access them using their string-name meta-data.

In this Chapter we talk about two datasets, one online available PAMAP2 dataset and a second in-house collected Mentech dataset. Then we discuss the data pre-processing techniques used to plot, compare, and analyze the subject-wise input data. Data analysis was used to perform data-cleaning to remove the noise instances collected during the data collection of the Mentech dataset. Standard deviation applied windows were used with a Threshold value of $0.3\,\mathrm{Hz}$ to only keep the static instances from sitting and standing labeled data. A combined dataset, called Megabase, was created by down-sampling the PAMAP2 dataset to $25\,\mathrm{Hz}$, and aligning its axis to match the sensor orientation of the Mentech dataset. Once data is pre-processed, three hand-crafted orientation features (Pitch, Roll, and Tilt angle) were calculated. Next, Data Fragmentation was performed to create 3D Tensors from acceleration components, further converted into 4D Tensors with one degenerate dimension.

These steps make the final 4D input sensor required for 2D Convolutional Neural Networks. The proposed model architectures to further process this created input and perform Human Activity classification are discussed in the next Chapter 4.

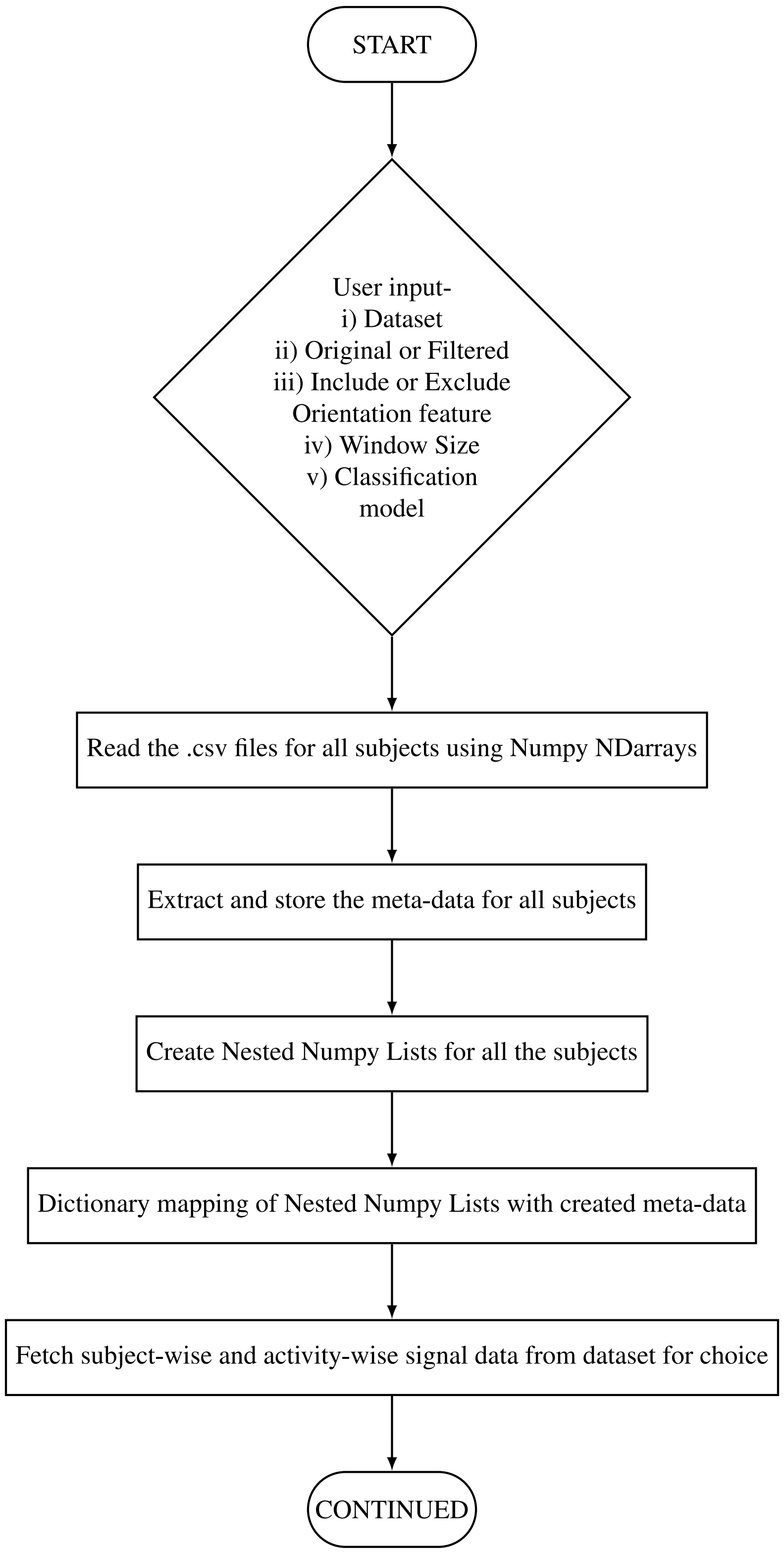

START
User input-
i) Dataset
ii) Original or Filtered
iii) Include or Exclude Orientation feature
iv) Window Size
v) Classification model
Read the .csv files for all subjects using Numpy NDarrays
Extract and store the meta-data for all subjects
Create Nested Numpy Lists for all the subjects
Dictionary mapping of Nested Numpy Lists with created meta-data
Fetch subject-wise and activity-wise signal data from dataset for choice
CONTINUED

Figure 3.11: Process flowchart for proposed solution Part-I

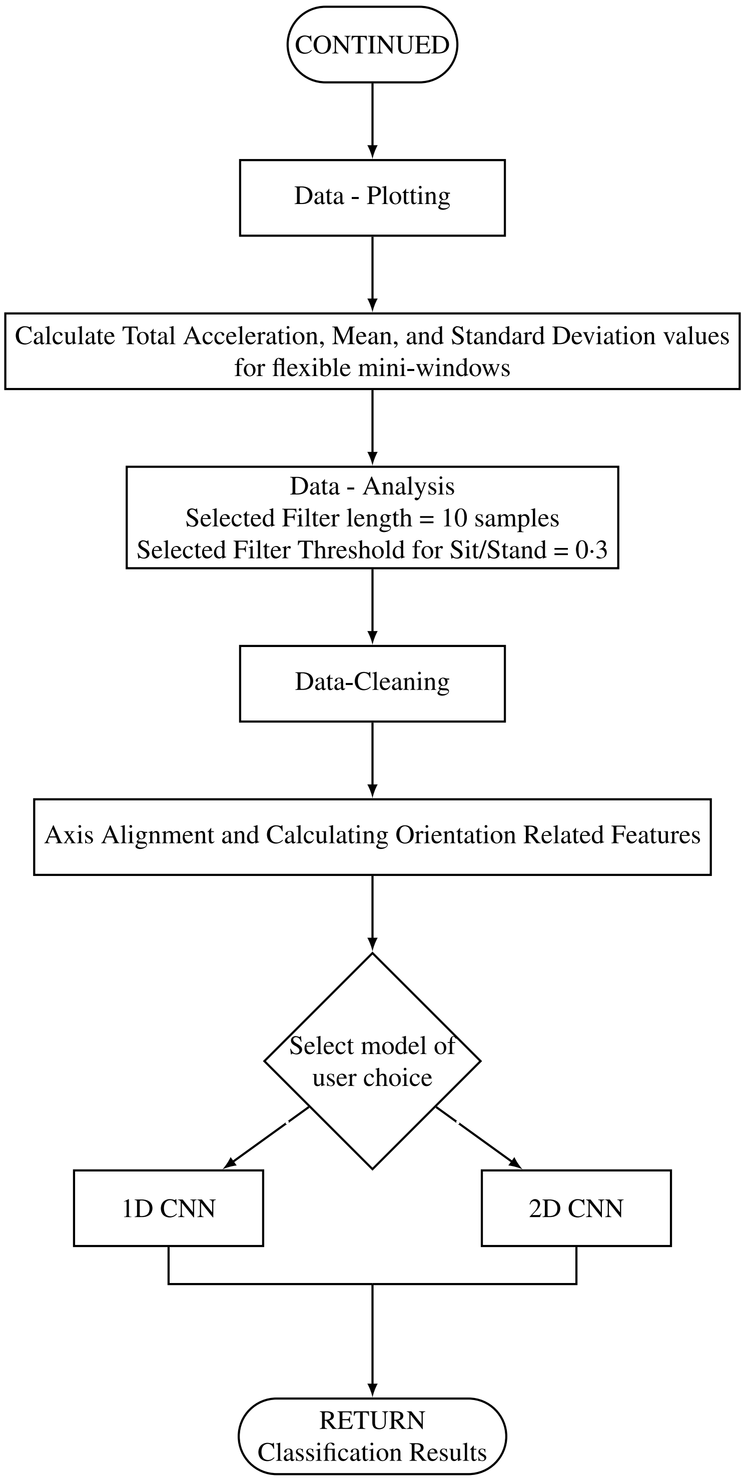

CONTINUED
Data - Plotting
Calculate Total Acceleration, Mean, and Standard Deviation values for flexible mini-windows
Data - Analysis
Selected Filter length = 10 samples
Selected Filter Threshold for Sit/Stand = 0·3
Data-Cleaning
Axis Alignment and Calculating Orientation Related Features
Select model of user choice
1D CNN
2D CNN
RETURN Classification Results

Figure 3.12: Process flowchart for proposed solution Part-II

4 Model Architecture

This Chapter discusses the proposed solution for HAR, keeping in mind the literature's observations explored in the Chapter 2. The Deep learning techniques found most promising for HAR are variations and combinations of Convolutional Neural Networks (CNN) architectures. The 1D and 2D CNN architectures were selected and researched further in this project. Special attention was given to using a minimum number of layers and hand-crafted features in the proposed solution to reduce the complexity of architecture, reducing the real-time computational cost of the solution. Before discussing the proposed model, it is essential to understand the existing HUME architecture and its integration with the current HAR model. The proposed solution's architectural and integration details are discussed in the following Sections.

4.1 Current HUME architecture

A high-level architecture diagram of HUME is given in Figure 4.1, where the HAR is performed using 1D CNN in the pre-processing step. The existing HAR model is a binary classification model for discriminating static and dynamic activities. Using this model, the subject's real-time (static-dynamic) position is estimated, which defines whether Static or Dynamic stress models are activated in real-time. HUME uses the output of Arousal levels for stress classification. Every time a position change is identified, further decision making to process the input signals as per the changed activity to provide more accurate Arousal levels, thus improving overall HUME performance.

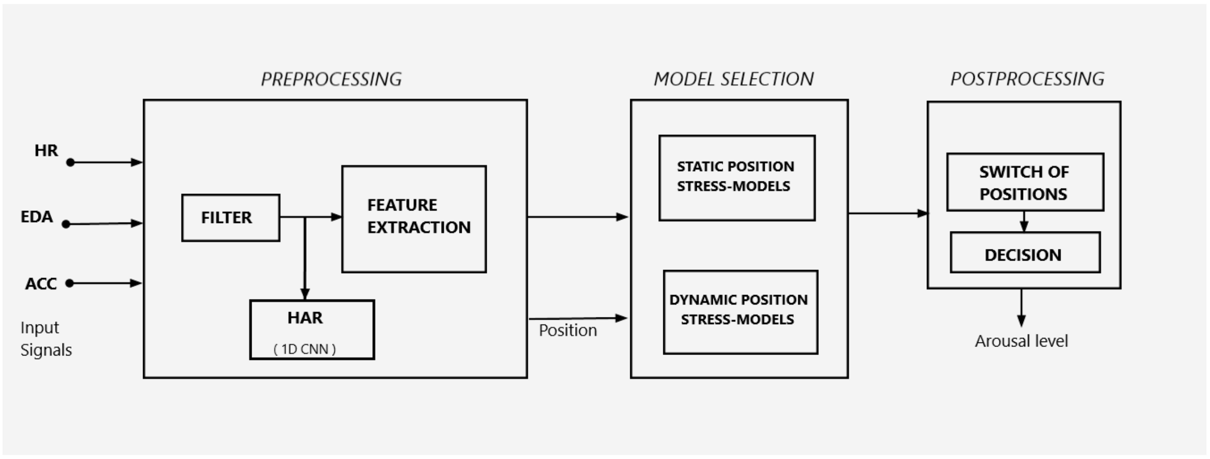


Figure 4.1: High-level HUME architecture with existing static-dynamic HAR model

The studied DL architectures are capable of distinguishing between Static-Dynamic classes efficiently ($\sim 95\%$ accuracy). The HAR problem becomes more complex as we add new activity classes to it. The three classes, i.e., sitting, standing, and walking, are the most common static-dynamic activities a human performs in their daily life. To study the effect of sensor orientation on the model performance, three orientation features, namely, Pitch, roll, and tilt angle (discussed in Section 2.8.2) are calculated before feeding the input signal to the classification model.

Irrespective of the final model being used, a 3-class classification architecture can be integrated with HUME to provide further information about the static activity, i.e., whether a person is sitting or standing. As discussed in Chapter 1, sitting and standing positions can deliver meaningful insights to deciding whether the person is mentally stressed or physically active. The post-processing step can be modified accordingly and is not discussed further in this report to utilize this additional (sitting/standing) information in producing the 'Arousal Levels' for stress detection. Since the HUME implementation uses older Tensorflow and sklearn, the current model code needs to be downgraded accordingly to make it compatible with HUME. The proposed HAR model can be integrated with HUME in the following ways:

1. The proposed 3-class classification model can be used to replace the 2-class classification model. For this, the walking class will represent the dynamic class, while sitting and standing will represent the static class.
2. The proposed 3-class classification model can also be used as a 2-class classification model to classify static activities like sitting and standing. Once the decision is made by the static/dynamic HAR model decides that it is a static activity, the input signal can further be processed to remove the 'dynamic' instances from it, using a low threshold filter similar to the one used in the data-cleaning process (Section 3.3.3). This filtered input can then be fed to the proposed HAR classifiers to identify the sitting and standing classes. This integration flow can also be seen in Figure 4.2.

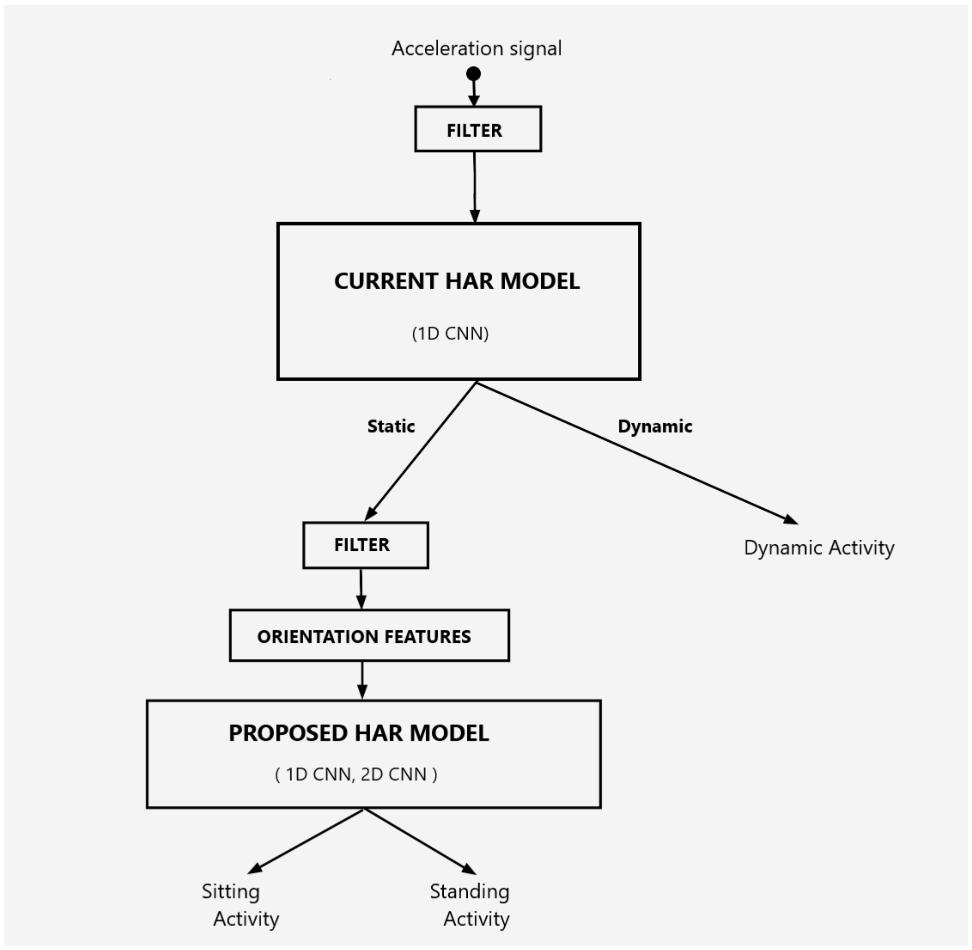


Figure 4.2: Integration of existing static-dynamic (1D CNN) HAR model, with proposed HAR model being used as a binary classifier for sitting and standing classes

4.2 Convolutional Neural Network Architecture

According to the literature review done in Chapter 2, Convolutional Neural Network (CNN) often fails to capture temporal patterns. This is why many researchers explored LSTM since they can learn long-term time dependencies, overcoming CNN's limitation (Jordao et al. 2018). To overcome this limitation,

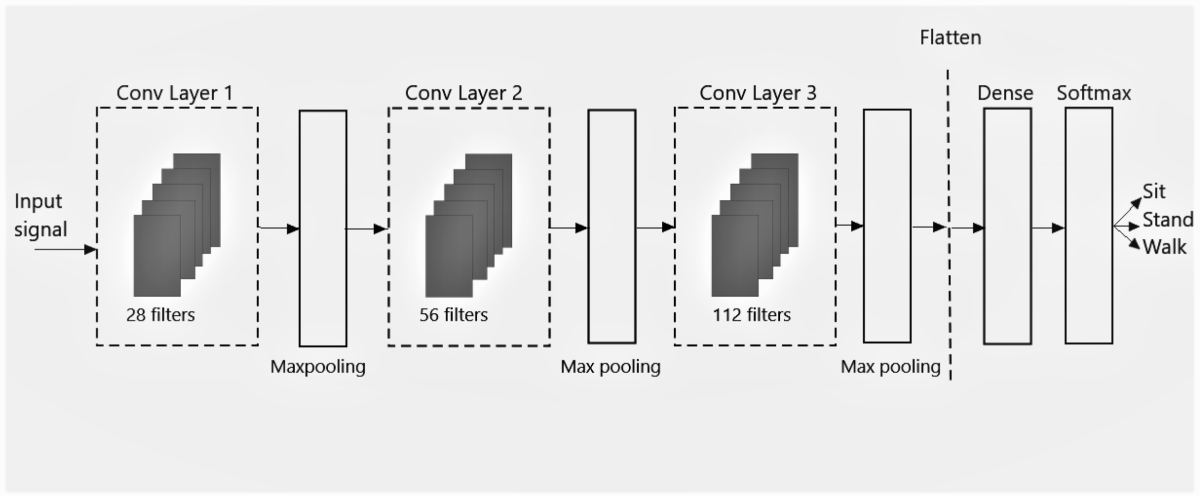


Figure 4.3: Proposed 2D CNN architecture

Jordao et al. 2018 used convolutional kernels of larger size each in their implementation (also discussed in Section 2.7). Using the same motivation, more extended convolutional kernel sizes are used in the proposed CNN architectures.

A 2D CNN is proposed consisting of 3 convolutional layers, each followed by a Maxpooling layer. Different 2D convolutional kernels are used for every convolution layer, as illustrated in Figure 4.3. The first convolutional layer consists of 28 filters of kernel-size (15,n), where n = 2 if hand-crafted features are not selected, otherwise n = 5. This means that the convolutional kernel works on 15 input data values at a time for 'n' input acceleration values. The number of filters is increased to 56 in the second convolutional layer, with a kernel size of (25,3). A more significant number of filters and kernel size are used to ensure that the features extracted from the first convolutional layer are properly analyzed by making more extensive combined feature patterns. The third convolutional layer consists of 112 convolutional filters of size (15,3). A Batch-normalization layer was also used just before the second and third convolutional layers. The last convolutional layer is followed by one fully connected dense layer and a softmax layer at the end. The layer-wise model architecture can be seen in Figure 4.3. These parameters are derived from the study in Jordao et al. 2018, and are modified or fine-tuned as discussed in Chapter 5.

The same architecture, as shown in Figure 4.3, is also utilized to achieve a 1D CNN architecture. The 2D convolutional layers are used with a modified kernel-size of (m,1), where m is the same kernel dimension-value used for 2D CNN architecture, i.e., 15, 25, and 15 for the three convolutional layers, respectively. Notice here that '1' in the kernel-size (m,1) enforces 1D CNN implementation because the convolutional kernel will work on 'm' input data-values (in time) for only one input acceleration value at a time.

5 Results

Two model architectures were implemented and tested for this project are: a 1D Convolutional Neural Network (CNN) and a 2D CNN. These architectures were trained, and classification accuracies were compared for three datasets (PAMAP2, Mentech in-house dataset, and Megabase), with and without using the filter used for data cleaning.

In the exploratory model-training phase, the selected models were trained for longer epochs (200-300 range) to study their performance and learning trend. One interesting general observation is captured in Figure 5.1, which shows the apparent confusion between 'sitting' and 'standing' classes faced by the 2D CNN model. Notice that the validation accuracy was initially terrible ($\approx 55\%$) until the first 50 epochs. Then the model can be seen to 'identify' the difference between 'siting' and 'standing' between 50-80 epochs. The model finally 'learning' the difference between these classes leading to high validation accuracy after that.

To further 'fast-forward' the above discussed model-training process, different 2D kernel sizes of 2D CNN were studied. The potential of the convolutional kernel was also discussed in Section 2.7, where Jordao et al. 2018 used convolutional kernels of size (12,2) each in their implementation (here, 12 stands for data-samples in time, while 2 represents two acceleration axes). The intention for using longer-sized convolution kernels was to test whether they can capture the required features (both local and temporal) 'quickly'. If 2D CNN architecture indeed learns features (both local and temporal) faster, it will be reflected in the model's reduced 'learning' phase (in terms of epochs), as shown in Figure 5.1. It was found that the 2D CNN model with a larger kernel size achieved higher accuracy within $\approx$ 20 epochs. Other Hyper-parameters which were tested along with kernel size are described in the next Section.

5.1 Hyper-parameter settings

2D CNN Model performance (for 256 window-size, using PAMAP2 dataset) was studied using different convolutional kernel sizes, learning rates, and decay rates. This step was performed to find the hyper-parameter settings used for Convolutional Models. The final settings used were learning-rate "0.001", with a decay of "0.001", and kernel-sizes as specified in Section 4.2. Since we are working with a three-class classification problem, categorical cross-entropy was used as the loss function. The model optimization was done using the Adam optimizer, and the activation function used was ReLU (Ignatov 2018; Xia et al. 2020). These specifications ensured that the model learns essential features reasonably while also not over-fitting on the training data.

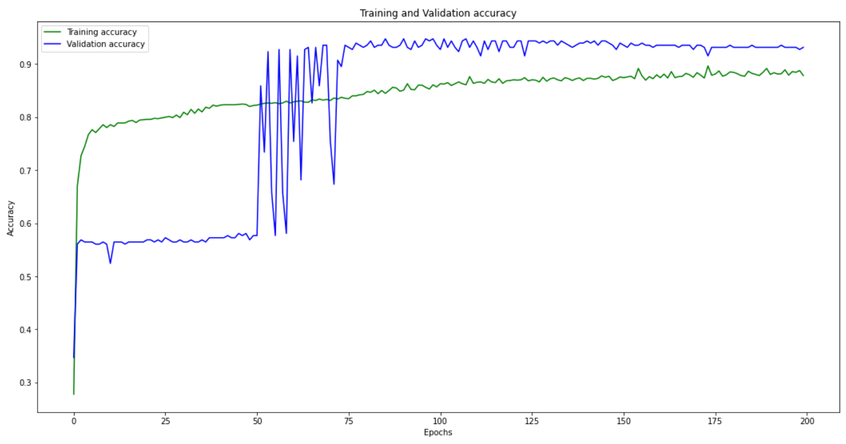


Figure 5.1: Training and validation accuracies for 200 epochs for 2D CNN model (using PAMAP2 dataset)

MODEL	PAMAP2		MENTECH		MEGABASE	
	Original	Filtered	Original	Filtered	Original	Filtered
1D CNN	92.2	93.8	69.9	76.6	78.1	84.3
2D CNN	91.8	93.4	63.7	81.4	72.3	80.7

Table 5.1: Model balanced accuracies using original and filtered data signals using 256 window-size

In the final model-training phase (i.e., Sections 5.2, 5.3, and 5.4), the selected models were trained for 100 epochs, with an early stopping having the patience of 20 epochs. Since initially, models were trained for longer epochs ($\approx$ 200 epochs), the patience of 20 epochs was used to study the model performance (even if bad or over-fitting). The two proposed CNN model architectures (with finalized hyper-parameters) were further tested to see the effect of applied data-cleaning, adding handcrafted features, and different input window sizes.

For evaluating the overall balanced model performances, a balanced accuracy metric is used, which is calculated using:

$$\text{Balanced Accuracy } = (((TP/(TP+FN) + (TN/(TN+FP)))/2 \tag{5.1}$$

where, TP = True Positives, TN = True Negatives, FN = False Negatives, and TN = True Negatives.

5.2 Effect of performing data-cleaning

This Section compares the model performances using the original acceleration signal and the filtered signal after applying the data-cleaning filter (as discussed in Subsection 3.3.3). All three models were trained on all three datasets using the input window size of 256 samples. The balanced accuracies obtained are summarized in Table 5.1.

For the PAMAP2 dataset, model performance did not improve much after applying data-cleaning ($\approx$ 2 pp), while for the Mentech dataset, significant improvements in balanced accuracy ($\approx$ 7-18pp) were observed. These results are as expected since, for the PAMAP2 dataset, there was not much data that was deleted, leading to similar model performance. While for Mentech and Megabase datasets, removing noise increased the model performance significantly. Another important observation is that the 1D CNN model architecture performed and 2D CNN model (sometimes even better). This shows the potential of using a simpler model for solving this HAR problem. For all the following Sections 5.3 and 5.4, only filtered datasets are used for model training.

5.3 Effect of using a different input window size

This Section compares different input window sizes used for data fragmentation. Different input sizes were selected based on the input-time used for testing the model performance (as mentioned in Subsection 3.3.4). The balanced accuracies obtained after training the models using filtered signals are summarized in Tables 5.2, 5.3 and 5.4, for PAMAP2, Mentech, and Megabase datasets, respectively.

In general, models perform best for smaller window sizes. Since the sampling rate of PAMAP2 is 100 Hz, the ideal window size is 256 samples or less (i.e., 2.56 seconds or less). For Mentech and Megabase datasets, the sampling rate is 25 Hz, hence 64 samples or less (i.e., again 2.56 seconds or less). For all the datasets, model performance decreases for a more extended window size of more than 2.56 seconds. Hence for the next Section 5.4, the three window-size selected for all datasets is 0.64, 1.28, and 2.56 seconds.

MODEL	Window size (# samples)			
	64	128	256	512
1D CNN	93.6	92.3	93.8	88.2
2D CNN	93.1	92.9	93.4	89.1

Table 5.2: Model balanced accuracies using three different input window sizes for the PAMAP2 dataset

MODEL	Window size (# samples)					
	16	32	64	128	256	512
1D CNN	84.9	86.4	85.8	80.5	76.6	63.1
2D CNN	85.1	85.4	85.5	81.4	81.4	69.5

Table 5.3: Model balanced accuracies using three different input window sizes for Mentech dataset

5.4 Effect of adding handcrafted features

This Section compares the model performances by adding three orientation features for every database; Pitch, Roll and Tilt angle. Models were trained on filtered datasets, with selected window sizes from the previous Section. These results are summarized in Table 5.5. The addition of orientation features helped increase the accuracy for the PAMAP2 dataset by 1pp for the 2D CNN model, while for the 1D CNN model, 93% accuracy was observed. On the other hand, the addition of these features neither did improve nor degrade the performance for Mentech and Megabase datasets. The best performance for all the datasets was found for a window size of 0.64 and 1.28 seconds. Confusion matrix for 1.28 seconds window for all the datasets is shown in Figures 5.2, 5.3 and 5.6. The accuracy and loss over the epochs for model training, and validation are shown in Figures 5.4, 5.5 and 5.7.

As seen in Figure 5.2, all three classes are being classified with high confidence for the PAMAP2 dataset. For Mentech and Megabase datasets, models can be seen confusing between sitting and standing classes the most. The Figures 5.5 and 5.7 show diverging training and validation loss and accuracies. This means the 2D CNN model is overfitting for the Mentech and Megabase datasets. This can be either due noise present in datasets, or due to lack of model hyper-parameters fine-tuning for data with low sampling rate. The addition of orientation features improved the classification accuracy by $\approx$1 pp.

MODEL	Window size (# samples)					
	16	32	64	128	256	512
1D CNN	87.4	89.3	87.7	84.9	80.8	73.2
2D CNN	87.8	87.6	85.4	83.8	84.7	70.8

Table 5.4: Model balanced accuracies using three different input window sizes for the Megabase dataset

MODEL	PAMAP2			MENTECH			MEGABASE		
	64	128	256	16	32	64	16	32	64
1D CNN	93.8	93.5	93.0	83.6	86.4	80.9	86.7	89.1	87.7
2D CNN	94.4	94.0	93.4	85.9	85.5	83.9	88.8	88.2	86.3

Table 5.5: Model balanced accuracies after adding orientation features using three different window sizes.

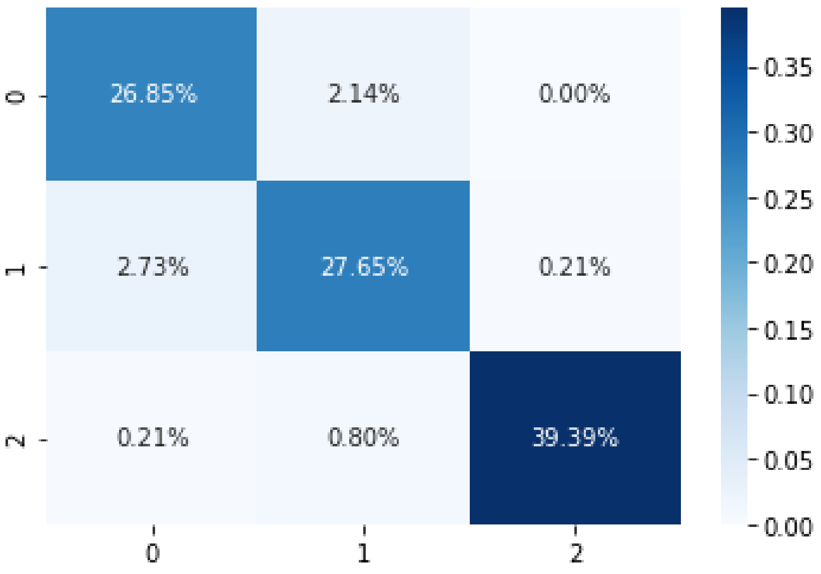


Figure 5.2: Confusion Matrix for 2D CNN model, with orientation features, with window-size 1.28 seconds (for 100 epochs with early stopping) using PAMAP2 dataset. Here 0 is the Sitting class; 1 is the Standing class; 2 is the Walking class.

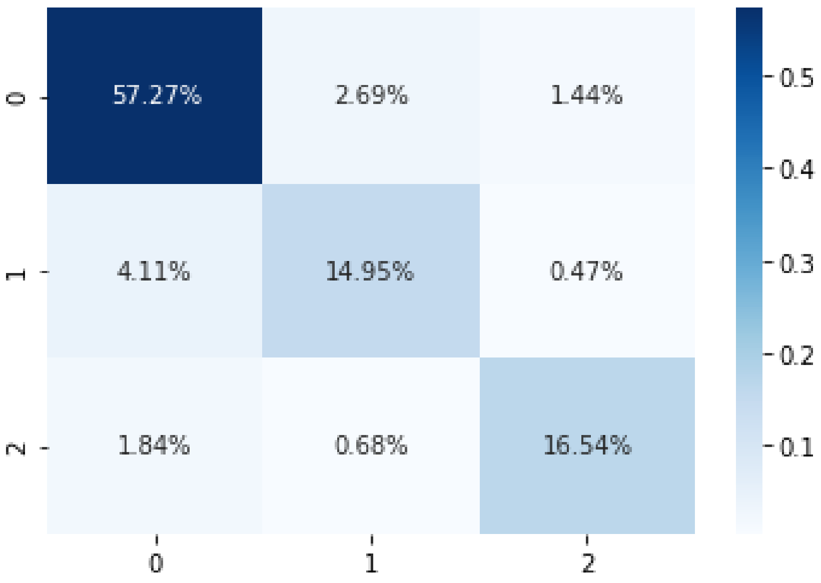
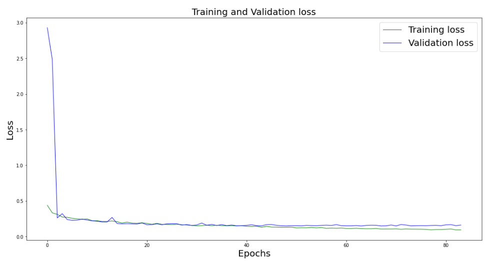
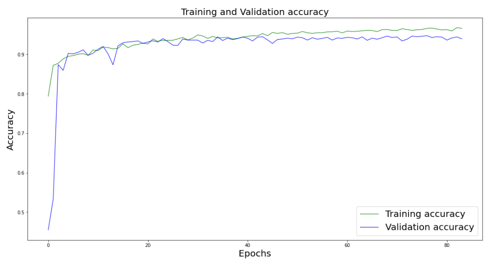


Figure 5.3: Confusion Matrix for 2D CNN model, with orientation features, with window-size 1.28 seconds (for 100 epochs with early stopping) using Mentech dataset. Here 0 is the Sitting class; 1 is the Standing class; 2 is the Walking class.

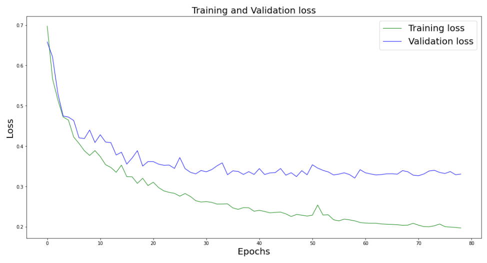


(a) Training and Validation loss graph

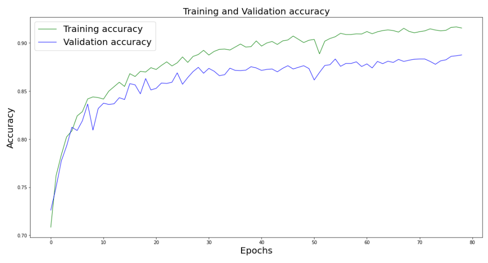


(b) Training and Validation accuracy graph

Figure 5.4: Loss and Accuracy graphs for 2D CNN model, with orientation features, with window-size 1.28 seconds (for 100 epochs with early stopping) using PAMAP2 dataset.



(a) Training and Validation loss graph



(b) Training and Validation accuracy graph

Figure 5.5: Loss and Accuracy graphs for 2D CNN model, with orientation features, with window-size 1.28 seconds (for 100 epochs with early stopping) using Mentech dataset.

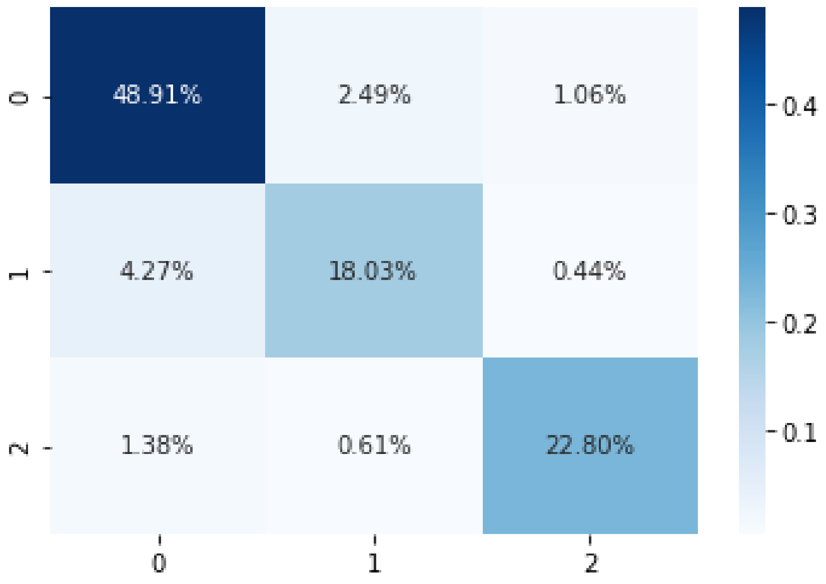
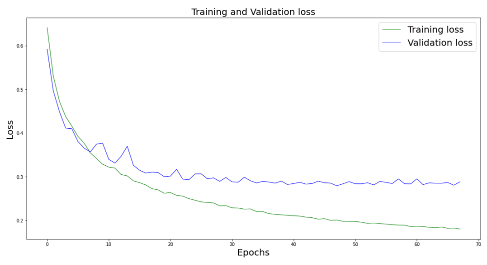
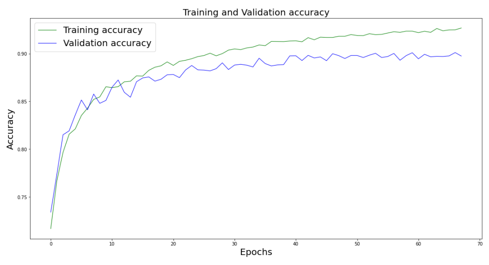


Figure 5.6: Confusion Matrix for 2D CNN model, with orientation features, with window-size 1.28 seconds (for 100 epochs with early stopping) using Megabase dataset. Here 0 is the Sitting class; 1 is the Standing class; 2 is the Walking class.



(a) Training and Validation loss graph



(b) Training and Validation accuracy graph

Figure 5.7: Loss and Accuracy graphs for 2D CNN model, with orientation features, with window-size 1.28 seconds (for 100 epochs with early stopping) using Megabase dataset.

6 Conclusion

This work explores the research methods available for solving the Human Activity Recognition (HAR) problem using multiple IMU sensors. The most commonly used signals used for HAR found in the research were extracted from an accelerometer, magnetometer, and gyrometer. While orientation and pressure sensors were also good signals for HAR, they were not considered for implementation in this project. Literature shows the importance of using a noise-free dataset for DL approaches. A low-pass butter-worth filter was found to be sufficient to filter the high-frequency noise from acceleration signal data. Although for in-house data collection, ensuring that motion-induced noise is not recorded while collecting labeled data was a priority for many researchers. Including varied environment setups for data-collection, like different activity intensities (for walking, running, cycling), chairs with varying heights for sitting, different sitting/standing postures, etc., were also found to help collect a good-quality dataset.

Literature shows that the State-of-art approaches to solving HAR problems use Deep Learning (DL) architectures with selected handcrafted features. These approaches show a good history of higher Activity classification accuracy compared to the traditional Machine learning (ML) approaches. However, a low accuracy bottleneck was observed for both these approaches, for sitting and standing classes ($\approx$ 75-85%). The gained knowledge was applied to solve HAR using only a single accelerometer, focusing on achieving high classification accuracy for the sitting and standing classes using DL models like 1D Convolutional Neural Network (CNN), and 2D CNN. Results were compared for an online available PAMAP2 dataset, an in-house collected (Mentech) dataset, and a combination of these two datasets (Megabase). The main findings from the comparison of the results, with their limitations, are:

- **Hyper-parameters:** Results show that larger convolutional kernel sizes were able to learn the essential features in a lower number of training epochs while also speeding up the model training procedure. The total number of convolutional filters and learning rate play a crucial role in achieving higher model performance. Although, increasing these parameters (convolutional kernel size, number of filters) or reducing the learning rate can lead to model over-fitting and should be done cautiously.

- **Data cleaning:** Results show that the used data filter (with standard deviation threshold of 0.3 for sitting and standing data signals), to reduce such noisy instances from data-signal improves the accuracy for Mentech and Megabase datasets significantly. While for the PAMAP2 dataset, not much data was deleted during data-cleaning, nor did it affect the model performance much.

- **Orientation-features:** Orientation-features were a promising addition as per the literature review done; however, they did not affect the performance much. Feature selection was not made due to time constraints. Still, it can provide more insights into what time-domain, frequency-domain, or orientation-related features can help the most in model performance.

- **Window size:** Results show that all DL models performed best with smaller input window size (in seconds), typically of 1.28 s or less. Due to the limitation of time, the model hyper-parameters were not fine-tuned for different input window sizes (or for different sampling rates of the datasets) to study the general behavior of other models.

- **Overfitting model performance:** Results show that the final trained models are overfitting for Mentech and Megabase datasets, with current hyper-parameter settings. This can be either due to low sampling rate used for these datasets, or the model can overfit on the noise introduced during data-collection of in-house Mentech dataset.

The final obtained accuracy for the PAMAP2 dataset is $\approx$ 94%, for the Mentech dataset is $\approx$ 85%, and for the Megabase dataset is $\approx$ 88% using 2D CNN model architecture. The confusion between standing and walking classes was high for the Mentech and Megabase datasets due to a data-collection labeling error which led to the recording of small walking-transition instances in static classes (sitting and standing). Best model performance was achieved using an input window size of 1.28 seconds or less, combined with the three orientation features for both datasets. Steps can be taken to study the model performance further to achieve even higher performance.

7 Future Recommendations

High classification accuracy was achieved by the proposed models on the PAMAP2 dataset, while performance was comparatively low for Mentech and Megabase datasets. However, by overcoming the limitations of the work done, model performance can be further improved. Recommendations in the order of decreasing priority are enlisted below:

1. It is advised to finalize or select the most suitable input window size and model per the project requirements. The hyper-parameters (like convolutional kernel size, number of filters, learning rate) should be fine-tuned again accordingly for the final selected classifier model.
2. Setting up more data-collection campaigns are advised to collect a large, correctly labeled activity-specific dataset. The removal of any noise in the training dataset will ideally lead to increased model performance. To minimize recording of noise during the data-collection procedure, further steps can be taken:
    - For all activity classes, especially static class, ' noise' must be defined. Accordingly, if the static classes can include small dynamic instances (like position-change or transitions), the value of the data-cleaning filter threshold should be increased to reflect the new allowed total acceleration for the static activities.
    - In the case of data-collection using multiple sensors, the addition of meta-data (like sensor ID, initial sensor orientation) in the raw signal-files can help in saving the pre-processing data time.
3. The Fragmentation features (discussed in Section 2.8.3) can be used to improve further the performance of the current (or existing) static and dynamic HAR model.
4. The proposed solution can be further scaled to add more classification classes like cycling, walking up or down the stairs, lying, car-driving activities, etc. To do so, it is essential to note that according to the literature survey done in Chapter 2, including selected handcrafted features can improve the model performance. Since dynamic activities can contain combinations of various human activities, it will also be reflected in the frequency-domain and time-domain features for these data signals. Handcrafted features can be selected using appropriate feature selection techniques, such as Chi-Square Test, Pearson Correlation Coefficient, Random Forest feature importance, Principal Component Analysis, etc.
5. Since calculating handcrafted features can be computationally costly; two approaches can be undertaken to make the model (with calculated features) real-time efficient:
    - The concept of the Teacher-Student LSTM model (discussed in Section 2.6) can be utilized. The teacher model can be the initial selected CNN model, trained on handcrafted features to learn the information-rich domain-specific features. The resulting probability distribution from the softmax layer of the Teacher model can be used as ground truth labels for a separate Student model (2D CNN or LSTM), which contains more convolutional filters (or neurons for LSTM) than the teacher model. This way, the learnings of one model can be transferred to another model, helping the student model to generalize in the same way as the teacher model. This method was found helpful in classifying hard targets like sitting and standing (Z. Chen et al. 2018).
    - To make the HAR solution real-time efficient, Apache Spark's framework can be used to start the data pre-processing required for HAR as soon as the data is collected. Spark Streaming efficiently handles live input data streams and provides high-level functions like 'map' and 'reduce' to perform important aggregation tasks. This can significantly reduce the time taken by a model to calculate selected handcrafted features in real-time.

Bibliography

Albarbar, A., A. Badri, Jyoti Sinha, and A. Starr (2009). “Performance evaluation of MEMS accelerometers”. In: vol. 42. 5, pp. 790–795. DOI: `https://doi.org/10.1016/j.measurement.2008.12.002`.

Anzanpour, Arman, Humayun Rashid, Amir M. Rahmani, Axel Jantsch, Nikil Dutt, and Pasi Liljeberg (2019). “Energy-efficient and Reliable Wearable Internet-of-Things through Fog-Assisted Dynamic Goal Management”. In: *Procedia Computer Science* 151, pp. 493–500. ISSN: 1877-0509. DOI: `https://doi.org/10.1016/j.procs.2019.04.067`.

Badawi, Abeer A., Ahmad Al-Kabbany, and Heba Shaban (2018). “Multimodal Human Activity Recognition From Wearable Inertial Sensors Using Machine Learning”. In: *2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, pp. 402–407. DOI: `10.1109/IECBES.2018.8626737`.

Banning, Bart, Anthony MacKay, and Paul Hickley (2020). “Changing from Magnetic to True Tracks in Aviation”. In: *2020 European Navigation Conference (ENC)*, pp. 1–8. DOI: `10.23919/ENC48637.2020.9317517`.

Banos, Oresti (2014). *MHEALTH Dataset Data Set*. UCI Machine Learning Repository. URL: `http://archive.ics.uci.edu/ml/datasets/mhealth+dataset`.

Banos, Oresti, Rafael Garcia, Juan A. Holgado-Terriza, Miguel Damas, Hector Pomares, Ignacio Rojas, Alejandro Saez, and Claudia Villalonga (2014). “mHealthDroid: A Novel Framework for Agile Development of Mobile Health Applications”. In: *Ambient Assisted Living and Daily Activities*. Ed. by Leandro Pecchia, Liming Luke Chen, Chris Nugent, and José Bravo. Cham: Springer International Publishing, pp. 91–98. ISBN: 978-3-319-13105-4.

Baños, O., C. Villalonga, Rafael García, Alejandro Saez, M. Damas, J. A. Holgado-Terriza, Sungyong Lee, H. Pomares, and I. Rojas (2015). “Design, implementation and validation of a novel open framework for agile development of mobile health applications”. In: *BioMedical Engineering OnLine* 14, S6–S6.

Bennett, S. (1984). “Nicholas Minorsky and the automatic steering of ships”. In: *IEEE Control Systems Magazine* 4.4, pp. 10–15. DOI: `10.1109/MCS.1984.1104827`.

Bristeau, Pierre-Jean, François Callou, David Vissière, and Nicolas Petit (2011). “The Navigation and Control technology inside the AR.Drone micro UAV”. In: *IFAC Proceedings Volumes* 44.1. 18th IFAC World Congress, pp. 1477–1484. ISSN: 1474-6670. DOI: `https://doi.org/10.3182/20110828-6-IT-1002.02327`.

Bruno, B., F. Mastrogiovanni, A. Sgorbissa, T. Vernazza, and R Zaccaria (2012). *WHARF - Wearable Human Activity Recognition Folder*. University of Genoa. URL: `https://github.com/fulviomas/WHARF`.

Chen, Jeng-Heng, Sou-Chen Lee, and Daniel B. DeBra (1994). “Gyroscope free strapdown inertial measurement unit by six linear accelerometers”. In: *Journal of Guidance, Control, and Dynamics* 17.2, pp. 286–290. DOI: `10.2514/3.21195`.

Chen, Kaixuan, Dalin Zhang, Lina Yao, Bin Guo, Zhiwen Yu, and Yunhao Liu (Jan. 22, 2021). “Deep Learning for Sensor-based Human Activity Recognition: Overview, Challenges and Opportunities”. In: *arXiv:2001.07416 [cs]*. arXiv: `2001.07416`.

Chen, Zhenghua, Le Zhang, Zhiguang Cao, and Jing Guo (Oct. 2018). “Distilling the Knowledge From Handcrafted Features for Human Activity Recognition”. In: *IEEE Transactions on Industrial Informatics* 14.10, pp. 4334–4342. ISSN: 1551-3203, 1941-0050. DOI: `10.1109/TII.2018.2789925`.

Corda, Stephen, Russell J. Franz, James N. Blanton, M. Jake Vachon, and James B. DeBoer (2002). *In-flight vibration environment of the nasa F-15B flight test fixture*. National Aeronautics and Space Administration, Dryden Flight Research Center.

Corke, Peter (2017). *Robotics, Vision and Control: Fundamental Algorithms In MATLAB*. second, completely revised, extended and updated ed. Extract of page 83. ISBN: 978-3-319-54413-7.

Darrow, Chester W. (1964). “The rationale for treating the change in galvanic skin response as a change in conductance”. In: *Psychophysiology* 1.1, pp. 31–38. DOI: https://doi.org/10.1111/j.1469-8986.1964.tb02618.x.

Erdaş, Çağatay Berke and Selda Güney (June 2021). “Human Activity Recognition by Using Different Deep Learning Approaches for Wearable Sensors”. In: *Neural Processing Letters* 53.3, pp. 1795–1809. ISSN: 1370-4621, 1573-773X. DOI: 10.1007/s11063-021-10448-3.

Fang, Bin, Fuchun Sun, Huaping Liu, and Chunfang Liu (2018). “3D human gesture capturing and recognition by the IMMU-based data glove”. In: *Neurocomputing* 277. Hierarchical Extreme Learning Machines, pp. 198–207. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2017.02.101.

Gholamrezaii, Marjan and Smt AlModarresi (May 2021). “A time-efficient convolutional neural network model in human activity recognition”. In: *Multimedia Tools and Applications* 80.13, pp. 19361–19376. ISSN: 1380-7501, 1573-7721. DOI: 10.1007/s11042-020-10435-1.

Goulart, Fátima Rodrigues-de-Paula and Josep Valls-Solé (1999). “Patterned electromyographic activity in the sit-to-stand movement”. In: *Clinical Neurophysiology* 110.9, pp. 1634–1640. ISSN: 1388-2457. DOI: https://doi.org/10.1016/S1388-2457(99)00109-1.

Ha, Sojeong and Seungjin Choi (July 2016). “Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors”. In: *2016 International Joint Conference on Neural Networks (IJCNN)*. Vancouver, BC, Canada: IEEE, pp. 381–388. ISBN: 978-1-5090-0620-5. DOI: 10.1109/IJCNN.2016.7727224.

Hallman, David M., Niklas Krause, Magnus Thorsten Jensen, Nidhi Gupta, Marie Birk Jørgensen, and Andreas Holtermann (2019). “Objectively Measured Sitting and Standing in Workers: Cross-Sectional Relationship with Autonomic Cardiac Modulation”. In: *International Journal of Environmental Research and Public Health* 16.4. ISSN: 1660-4601. DOI: 10.3390/ijerph16040650.

Ignatov, Andrey (Jan. 2018). “Real-time human activity recognition from accelerometer data using Convolutional Neural Networks”. In: *Applied Soft Computing* 62, pp. 915–922. ISSN: 15684946. DOI: 10.1016/j.asoc.2017.09.027.

Janssen, Wim GM, Hans BJ Bussmann, and Henk J Stam (Sept. 1, 2002). “Determinants of the Sit-to-Stand Movement: A Review”. In: *Physical Therapy* 82.9, pp. 866–879. ISSN: 0031-9023, 1538-6724. DOI: 10.1093/ptj/82.9.866.

Jordao, Artur, Leonardo Antônio Borges Torres, and William Robson Schwartz (Oct. 2018). “Novel approaches to human activity recognition based on accelerometer data”. In: *Signal, Image and Video Processing* 12.7, pp. 1387–1394. ISSN: 1863-1703, 1863-1711. DOI: 10.1007/s11760-018-1293-x.

Khanade, Kunal and Farzan Sasangohar (2017). “Stress, Fatigue, and Workload in Intensive Care Nursing: A Scoping Literature Review”. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 61.1, pp. 686–690. DOI: 10.1177/1541931213601658.

Konstantinidis, Dimitrios, Vasileios Argyriou, Tania Stathaki, and Nikolaos Grammalidis (2020). “A modular CNN-based building detector for remote sensing images”. In: *Computer Networks* 168, p. 107034. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2019.107034.

Liang, Jiefeng, Tianjun Jing, Huanna Niu, and Jiangbo Wang (2020). “Two-Terminal Fault Location Method of Distribution Network Based on Adaptive Convolution Neural Network”. In: *IEEE Access* 8, pp. 54035–54043. DOI: 10.1109/ACCESS.2020.2980573.

Looff, Peter de (2019). *Aggressive behaviour, burnout and physiology: Predictors of aggression in patients and burnout symptoms in nursing staff: Biosensors*. URL: https://hdl.handle.net/2066/202974.

Mannini, Andrea, Mary Rosenberger, William L. Haskell, Angelo M. Sabatini, and Stephen S. Intille (Apr. 2017). “Activity Recognition in Youth Using Single Accelerometer Placed at Wrist or Ankle”. In: *Medicine & Science in Sports & Exercise* 49.4, pp. 801–812. ISSN: 0195-9131. DOI: 10.1249/MSS.0000000000001144.

Mark, David, Jack Nutting, Kim Topley, Fredrik Olsson, and Jeff LaMarche (2014). "Whee! Gyro and Accelerometer!" In: *Beginning iPhone Development: Exploring the iOS SDK*. Berkeley, CA: Apress, pp. 699–726. ISBN: 978-1-4842-0199-2. DOI: `10.1007/978-1-4842-0199-2_20`.

Mathworks (n.d.). *model imu gps and insgps*. URL: `https://www.mathworks.com/help/fusion/gs/model-imu-gps-and-insgps.html`.

Mekruksavanich, Sakorn and Anuchit Jitpattanakul (Feb. 26, 2021). "LSTM Networks Using Smartphone Data for Sensor-Based Human Activity Recognition in Smart Homes". In: *Sensors* 21.5, p. 1636. ISSN: 1424-8220. DOI: `10.3390/s21051636`.

Merry, Kohle J, M. MacPherson, E. Macdonald, Michael Ryan, E. Park, and Carolyn J. Sparrey (2019). "Differentiating Sitting, Standing and Walking Through Regional Plantar Pressure Characteristics." In: *Journal of biomechanical engineering*.

Park, Keunwoo, Daehwa Kim, Seongkook Heo, and Geehyuk Lee (2020). "MagTouch: Robust Finger Identification for a Smartwatch Using a Magnet Ring and a Built-in Magnetometer". In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, pp. 1–13. ISBN: 9781450367080. URL: `https://doi.org/10.1145/3313831.3376234`.

Posada-Quintero, H. F., N. Reljin, C. Mills, I. Mills, J. P. Florian, J. L. VanHeest, and K. H. Chon (Jan. 2018). "Time-varying analysis of electrodermal activity during exercise". In: DOI: `https://doi.org/10.1371/journal.pone.0198328`.

POSNER, JONATHAN, JAMES A. RUSSELL, and BRADLEY S. PETERSON (2005). "The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology". In: *Development and Psychopathology* 17.3, pp. 715–734. DOI: `10.1017/S0954579405050340`.

Rafegas, Ivet, Maria Vanrell, Luís A. Alexandre, and Guillem Arias (2020). "Understanding trained CNNs by indexing neuron selectivity". In: *Pattern Recognition Letters* 136, pp. 318–325. ISSN: 0167-8655. DOI: `https://doi.org/10.1016/j.patrec.2019.10.013`.

Raja, Sudeep (n.d.). *FNNs ,RNNs ,LSTM and BLSTM*. URL: `http://cse.iitkgp.ac.in/~psraja/FNNs%5C%20,RNNs%5C%20,LSTM%5C%20and%5C%20BLSTM.pdf`.

Reiss, Attila (2012). *PAMAP2 Physical Activity Monitoring*. UCI Machine Learning Repository. URL: `https://archive.ics.uci.edu/ml/datasets/PAMAP2+Physical+Activity+Monitoring`.

Reyes-Ortiz, Jorge, Davide Anguita, Alessandro Ghio, Luca Oneto, and Xavier Parra (2012). *Human Activity Recognition Using Smartphones*. UCI Machine Learning Repository. URL: `https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones`.

Riboni, Daniele and Claudio Bettini (2009). "Context-Aware Activity Recognition through a Combination of Ontological and Statistical Reasoning". In: *Ubiquitous Intelligence and Computing*. Ed. by Daqing Zhang, Marius Portmann, Ah-Hwee Tan, and Jadwiga Indulska. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 39–53. ISBN: 978-3-642-02830-4.

Riener, Andreas and Alois Ferscha (2007). "Driver Activity Recognition from Sitting Postures". In: *Mensch & Computer 2007 Workshopband*. Ed. by Thilo Paul-Stueve. Weimar: Verlag der Bauhaus-Universität Weimar, pp. 55–62.

Roggen, Daniel, Alberto Calatroni, and Long-Van Nguyen-Dinh (2012). *OPPORTUNITY Activity Recognition Data Set*. UCI Machine Learning Repository. URL: `https://archive.ics.uci.edu/ml/datasets/opportunity+activity+recognition`.

Saez, Yago, Alejandro Baldominos, and Pedro Isasi (Dec. 30, 2016). "A Comparison Study of Classifier Algorithms for Cross-Person Physical Activity Recognition". In: *Sensors* 17.12, p. 66. ISSN: 1424-8220. DOI: `10.3390/s17010066`.

Sarkar, Sudeshna (n.d.). *Convolutional Neural Network*. URL: `http://cse.iitkgp.ac.in/~sudeshna/courses/DL17/CNN-22mar2017.pdf`.

Schumm, J., M. Bachlin, C. Setz, B. Arnrich, D. Roggen, and G. Troster (2008). "Effect of movements on the electrodermal response after a startle event". In: *2008 Second International Conference on Pervasive Computing Technologies for Healthcare*, pp. 315–318. DOI: 10.1109/PCTHEALTH.2008.4571101.

Shelke, Sagar and Baris Aksanli (2019). "Static and Dynamic Activity Detection with Ambient Sensors in Smart Spaces". In: *Sensors* 19.4. ISSN: 1424-8220. DOI: 10.3390/s19040804.

Sukor, A. S. Abdull, A. Zakaria, and N. Abdul Rahim (Mar. 2018). "Activity recognition using accelerometer sensor and machine learning classifiers". In: *2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA)*. 2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA). Batu Feringghi: IEEE, pp. 233–238. ISBN: 978-1-5386-0389-5. DOI: 10.1109/CSPA.2018.8368718.

Sun, Feng-Tso, Cynthia Kuo, Heng-Tze Cheng, Senaka Buthpitiya, Patricia Collins, and Martin Griss (Jan. 2012). "Activity-Aware Mental Stress Detection Using Physiological Sensors". In: vol. 23. ISBN: 978-3-642-29335-1. DOI: 10.1007/978-3-642-29336-8_12.

Tabrizi, Sahar S., Saeid Pashazadeh, and Vajiheh Javani (2020). "Data acquired by a single object sensor for the detection and quality evaluation of table tennis forehand strokes". In: *Data in Brief* 33, p. 106504. ISSN: 2352-3409. DOI: https://doi.org/10.1016/j.dib.2020.106504.

towardsdatascience, Kunlun Bai @ (n.d.). *A Comprehensive Introduction to Different Types of Convolutions in Deep Learning*. URL: https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215.

Vijayan, Vini, James P. Connolly, Joan Condell, Nigel McKelvey, and Philip Gardiner (2021). "Review of Wearable Devices and Data Collection Considerations for Connected Health". In: *Sensors* 21.16. ISSN: 1424-8220. DOI: 10.3390/s21165589.

Wang, LuKun and RuYue Liu (Feb. 2020). "Human Activity Recognition Based on Wearable Sensor Using Hierarchical Deep LSTM Networks". In: *Circuits, Systems, and Signal Processing* 39.2, pp. 837–856. ISSN: 0278-081X, 1531-5878. DOI: 10.1007/s00034-019-01116-y.

The Magnetic Array Study of Effective Detection and Location for Submarine Pipeline (June 2019). Vol. All Days. International Ocean and Polar Engineering Conference. ISOPE-I-19-639. eprint: https://onepetro.org/ISOPEIOPEC/proceedings-pdf/ISOPE19/All-ISOPE19/ISOPE-I-19-639/1129713/isope-i-19-639.pdf.

Weiss, Gary (2019). *WISDM Smartphone and Smartwatch Activity and Biometrics Dataset*. UCI Machine Learning Repository. URL: https://archive.ics.uci.edu/ml/datasets/WISDM+Smartphone+and+Smartwatch+Activity+and+Biometrics+Dataset+.

Weytjens, Hans and Jochen De Weerdt (2020). "Process Outcome Prediction: CNN vs. LSTM (with Attention)". In: *Lecture Notes in Business Information Processing*, pp. 321–333. ISSN: 1865-1356. DOI: 10.1007/978-3-030-66498-5_24.

Xia, Kun, Jianguang Huang, and Hanyu Wang (2020). "LSTM-CNN Architecture for Human Activity Recognition". In: *IEEE Access* 8, pp. 56855–56866. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2982225.

Zharkov, M V, K K Veremeenko, I M Kuznetsov, and A N Pronkin (Sept. 2020). "Multiple antenna gyro-GNSS attitude determination system". In: *IOP Conference Series: Materials Science and Engineering* 927, p. 012073. DOI: 10.1088/1757-899x/927/1/012073.

A Appendix

Human Activity Recognition data collection document - @Mentech, Mitali Agrawal, 10th April 2021

This data is being collected to train a model for classifying human activities: sitting, standing, and walking. The aim is to collect the data to cover all the relaxed postures and trends we perform during these activities. The best way to do that is to be comfortable and be yourself. Remember, the sensor (signal) which we are working on will be located on your ankle. That means we want to study the difference in your foot movement (precisely your ankle) while performing these activities. To make a good dataset, we must cover all the different ways we can sit, stand, and walk. To help you get started and give you some ideas, the below table shows a routine that can be followed in the lab, detailing the detail, aim, and time duration for every activity type.

NOTE: make sure you wear the sensor on your dominant foot. The dominant foot is usually the same as one's dominant hand.

There will be a separate recording for every class and posture. The recording and labeling will be started once you indicate you are ready and are already in your position. During every recording, make sure you take a posture, maintain it for at least 1 minute, think about your next posture (from the idea column). After this change to your next posture, be comfortable, indicate start of activity and hold it for the next 1 minute. Continue with this loop unless the time mentioned below table runs out.

CLASS CODE	ACTIVITY TO RECORD (CLASS)	TIME (in minutes)	AIM & IDEAS
sd1	Standing: without support	8	• Read a book, Work on white-board • Talk on mobile, face-2-face • Stand cross-legged • Shift weight to different leg • Remove your shoes and maybe repeat a few of the above postures.
sd2	Standing: with support	8	
st1	Sitting: chair on wheels	5	• Interested sitting: browse the internet on a laptop, type something, be attentive, read a book. • Lazy sitting, tired, relaxing postures. • Crossed leg sitting. • Change different foot positions. • Remove your shoes and maybe repeat a few of the above postures.
st2	Sitting: chair without back support	5	
st3	Sitting: low height chair	3	
st4	Sitting: high height chair, bar chair	3	
st5	Sitting: Sofa	5	
w1	Walking: slow pace (Treadmill)	5	• Feel free to multitask while walking. Talk, chat, listen to music or walk for a while. • Be safe, don't fall.
w2	Walking: medium pace (hallway/normal walk)	5	
w3	Walking: high pace (Treadmill)	3	
w4	Walking: inclination (Treadmill) (comfortable)	3	
t1	Transition: Sitting to Standing	4 or more times	• Try different chair varieties, heights, sofa to do this. • Try to cover the transition movement/pattern you follow during this process. • The label marking should be started/ended separately for every transition.
t2	Transition: Standing to Sitting	4 or more times	
t3	Transition: Sitting to Walking	4 or more times	
t4	Transition: Walking to Sitting	4 or more times	

Discard data with interruptions or wherever you feel the activity has deviated from the aim of the class being recorded. Remember, recording data with noises is very poisonous for the data model. Also, make a note of participants Age, Sex, and Dominant leg.