An Android Application to Predict Human Activity using a Deep Learning LSTM Model

by

Debabrata Sikder
19366011

A project submitted to the Department of Computer Science and Engineering in partial fulfillment of the requirements for the degree of Master of Engineering (M.Engg.) in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
September 2023

Declaration

It is hereby declared that

1. The project submitted is our own original work while completing degree at Brac University.
2. The project does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The project does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Debabrata Sikder
19366011

Approval

The project titled "An Android Application to Predict Human Activity using a Deep Learning LSTM Model" submitted by

1. 1. Debabrata Sikder (19366011)

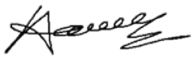
Of Summer, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Master of Engineering (M.Engg.) in Computer Science and Engineering September 21, 2023.

Examining Committee:

Supervisor:
(Member)

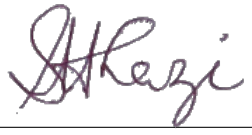
Dr. Md. Golam Rabiul Alam, PhD
Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)



Dr. Amitabha Chakrabarty, PhD
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)



Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement

I will remain curious, objective, and open minded about other people's ideas, choices, and beliefs.

Abstract

The machine learning approach to estimate human activity using smartphone sensor data is challenging. In this project, the HAR approach is conducted based on the LSTM model and can recognize six different behaviors, i.e., Downstairs, Jogging, Sitting, Standing, Upstairs, and Walking. To achieve the best potential result, various machine learning and statistical approaches were explored. The long short-term memory (LSTM) is a recurrent neural networks (RNNs) capable of learning long-term dependencies, especially in sequence prediction problems. This LSTM model was applied in this project, to obtain the desired result. This model shows 97% test accuracy. Finally, the model was exported and deployed in the Android application, which has an user interface that could provide a user-friendly experience.

Keywords: Machine Learning (ML), Deep Learning, Human activity (HA) Human activity recognition (HAR), TensorFlow, Recurrent neural networks (RNNs), Long short-term memory networks (LSTM).

Dedication

This endeavor is dedicated to all researchers who generate unique ideas by working hard to solve critical problems.

Acknowledgement

In the beginning, I sincerely recall the immense power of nature for helping me to complete this task. In addition, I want to express my profound gratitude and respect to my supervisor, Dr. Md. Golam Rabiul Alam, Professor, Department of Computer Science and Engineering at BRAC University, for his informative ideas, scholarly suggestions, significant support, and compassionate cooperation throughout the entire progress of this project. Finally, I want to thank my parents, who always encourage me to achieve my goal.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

GUI Graphical User Interface

HAR Human Activity Recognition

$LSTM$ Long Short-term Memory

PA Physical Activity

$PASE$ Physical Activity Scale for the Elderly

PB Protocol Buffers

RNN Recurrent Neural Network

$WISDM$ Wireless Sensor Data Mining

Chapter 1

Introduction

Human activities are functions, tasks, or works done by human beings over time for achieving certain objectives. Some primary human activities are walking, jogging, standing, sitting, and walking upstairs and downstairs. We can detect these activities by our senses easily. But, machines can not detect our activities as we do by our senses, so we need to build a machine learning system to detect human activities properly. Sometimes, we need to detect these types of human activities for specific purposes, i.e., game analysis and health care. In this case, we need to continuously detect activities for a certain period and store the data for analysis. But, a manual approach to detecting activities continuously for a long time is very difficult to do. Moreover, the usage of a manual system is expensive and can create a lot of human errors. An automatic artificial intelligence or machine learning-based system can solve this problem. So, we need to construct a system that can detect human activity continuously and efficiently.

Human activity recognition (HAR) can be referred to as the mechanism of identifying and labeling human activities using Machine Learning (ML) approach and raw human activity data collected from sources like Wearable sensors [30], electronic device sensors such as smartphone sensors [25], camera devices such as Kinect [28], closed-circuit television (CCTV) [26]. Smartphones are essential communication devices with modern technologies that give smart support to consumers daily. Nowadays, smartphones have different types of sensors, i.e., Accelerometer, Gyroscope, Proximity sensor, Magnetometer, etc. These sensors data can be used to detect human activities. The Human Activity Recognition (HAR)[21] framework takes raw data from sensors and analyzes human activity using deep learning algorithms [4]. As we want to use smartphone sensor data, we must make a system directly connected to the smartphone. Android applications are very popular these days. Because it is an open-source operating system, a vast pool of programmers can provide their best and improve Android. There are around 200000 Android applications available, with over 3 billion downloads. Therefore, if we develop an Android application for the HAR system, it will be very convenient for the users. In this case, we decided to make an Android application to detect human activities. This application will identify human activity using the deep learning model and real-time sensor data. The output of the Android application is an extensive system that identifies and labels different types of human activities. This HAR uses the LSTM deep learning technique and, smartphone sensors dataset and real-time

smartphone sensor data to recognize and classify human activities and motions. It can potentially transform various sectors, including healthcare, sports performance analysis, gaming, intelligent monitoring, and human-computer interface. Through improved classification methods and an improved dataset, the project tried to create a graphical user interface that can detect human activities and be easy to understand for a general user.

1.1 Motivation

This project aims to improve the effectiveness and efficiency of human activity recognition by developing an Android application which uses a deep learning model. The difficulties of human operators in real-time human activity recognition emphasize the need for automated solutions. By adopting deep learning methodologies, this project intends to improve the effectiveness and efficiency of human activity recognition. This will be accomplished with a dataset that contains more relevant smartphone sensor data and an advanced deep-learning model. The fundamental goal of this project is to contribute by improving the accuracy of human activity recognition using advanced algorithms for automated recognition processes. There are several HAR methods that are based on sensor types, i.e., cameras, wearable devices, and smartphone sensors. Camera-based HAR was popular previously. In a dark situation, camera performance degrades rapidly. Having cameras installed in certain areas, such as beds and bathrooms, will dramatically violate human privacy and cause moral and legal issues [29]. As a result, regular cameras have been abandoned as HAR sensors in recent years. Wearable technologies [35] such as smartwatches and fitness trackers are expensive and have a short battery life. But most of us have a smartphone, and we carry our smartphone a big part of the day. So, we need to make a HAR system to reduce the cost of extra devices like the Apple and Samsung Galaxy watches. The use of various sources makes HAR important for a wide range of usage, including healthcare [30], surveillance ([22], remote care to older people living alone [27]. This systemm can be used for sports performance analysis, gaming, intelligent monitoring, and human-computer interface. The broad deployment of HAR have a lot of benefits for human security and quality of life [24].

1.2 Problem Statement

The usage of human activity detection is increasing all around the world, and implementation of the HAR systems into an Android application ensures automated human activity recognition. However, this system's dependency on human operators limits its success. Furthermore, a computerized system can perform better performance and reduce cost. We have explored numerous related works and methodologies, highlighting the significance of precise human activity detection. The purpose of activity recognition is to identify human activities in real-world situations. Human activity recognition (HAR) is essential in surveillance, smart environments, and man-machine communication. Because human activity is dynamic and diverse, accurate activity identification is difficult. In the past, questionnaires like the Physical Activity Scale for the Elderly (PASE) were commonly used to assess Physical Activity (PA)[2]. Such questionnaires are limited because they cannot give accurate

continuous predictions. Therefore, we must build a system that can face continuous sensor data and detect human activity. Moreover, if we can develop an Application for smartphones that can detect human activity, we won't have to buy wearable devices. Consequently, it will reduce costs. Our main goal was to develop an Android application that can detect HA using the LSTM deep-learning approach. We searched for applications like this in the Google Play store, but they all use only sensors to record the data. None of them used a deep-learning approach to detect HA. We found some projects from GitHub, but most were old and could not be installed on the latest smartphones. Their accuracy was poor, and they were confused to detect differences between non-moving activities like sitting and standing. This project aims to provide a solution by developing a new human activity detection system. Our main goal is to enhance the accuracy of HAR systems and develop an Android application which can detect human activities. We collected a precise dataset to achieve better results. Then, combining a deep learning algorithm for automated recognition processes has been used. This project strives to build a system which emphasises creating a human activity recognition Android application.

1.3 Project Challenges

The first problem was to find a dataset that would fit the model. We found the WISDM Lab Dataset from Fordham University's website. However, the dataset had null values. We removed it using Python programming. There was an imbalance in the dataset; some rows were overlapped with another. So, we separate them so the model can consider them a unique value. For the developing section, sometimes we took help from different websites; many websites provided old techniques, which are not compatible with the recent updates of the high-level languages. So, we learned the new techniques and connected the old techniques together. During the model training, it took too long because Google Colab doesn't give as much computational speed as we want. So, we can not run as many epochs as we want.

1.4 Project Objectives

The primary objective of this project is to create an effective human activity recognition system by using a deep learning technique which is implemented in an Android application later. We focused on improving the identification and classification of different human activities to get increased accuracy in recognition. Moreover, we have done a thorough analysis of current procedures for detecting human activities. Finally, we perform careful testing in real-life situations and evaluation of the performance of the HAR system. We also compare its accuracy and efficiency to other methods and algorithms. In the future, the project can potentially improve human tracking, health and game analysis.
The aim is to prepare a deep learning LSTM [3] model first, to predict activities of a human. We used Wireless Sensor Data Mining (WISDM) dataset [20] from Fordham University, Bronx, NY. After that, we create a graphical user interface (GUI) for smartphones to detect a person's activities.Therefore, in summary, the key objectives of this project are as follows:

- To use a deep-learning model to recognize human activity using smartphone sensors dataset.
- After that to transform the deep-learning model into an Android application as general users can use it.

1.5 Contributions

Our project contributions include improved datasets, innovative system design, improved preprocessing approaches, and the implementation of an Android application. We significantly improve the accuracy of the system and build a human activity detection Android application. Furthermore, our effort provides a solid framework for future research in the domain. The datasets were processed, removing null values [30] before being fed into the deep-learning model. This method produced more accurate results in identifying human activities in real-life. Our primary goal was to design and develop a HAR system. There is much research on HAR. However, there was no graphical user interface (GUI) to show HA in real time. That is why we decided to build a GUI for the HAR system. We developed an Android application that is compatible with most of the latest smartphones. In summary, our contributions are illustrated bellow:

- This project began with the careful collection of an enriched dataset named WISDM Dataset. This new dataset has 1,098,207 rows, which have six different labels: Walking: 424,400 (38.6%), Jogging: 342,177 (31.2%), Upstairs: 122,869 (11.2%), Downstairs: 100,427 (9.1%), Sitting: 59,939 (5.5%) and Standing: 48,395 (4.4%). Notably, this dataset was made by collecting smartphone accelerometer sensors. We used the accelerometer x,y, and z-axis values to generate a typical pattern associated with the specific activities. After collecting the dataset, we removed the null values and separated some overlapped values.
- The following contribution of our project was to compare some machine learning methods and select the best method that shows the best accuracy for human activity recognition. We found the LSTM shows the best accuracy, and its execution time was the fastest. We developed a HAR system by integrating a preprocessed dataset and LATM deep learning model.
- The final contribution was to build a graphical user interface that would show a user-friendly experience for general users. We developed an Android application to detect human activities. The deep learning model is written in Python and exported as a Protocol buffers (PB) file so the Android application can understand and use the deep learning model for implementation purposes. Next, we developed the graphical user interface, an Android application called "Activity Recognition."

1.6 Organization of the Report

The report is constructed in the following manner - Chapter I demonstrates the introduction, motivation, problem statement, project objectives, and contribution. Chapter II explains the related works associated with HAR. Chapter III will methodology of the project. Then, in Chapter IV, includes application development. Afterward, Chapter V will present the performance analysis. Finally, the conclusion and the future scopes are shown in Chapter VI.

Chapter 2

Related Work

Activity identification has recently become a research issue because of the increasing availability of accelerometers in consumer items such as cell phones and the numerous possible uses. Early work in accelerometer-based activity recognition concentrated on using many accelerometers placed on various regions of the user's body.

Bao and Intille [7] employed five biaxial accelerometers worn on the user's right hip, dominant wrist, non-dominant upper arm, dominant ankle, and non-dominant thigh to collect data from 20 users in one of the early types of research on this topic. They developed models to recognize twenty daily tasks using decision tables, instance-based learning, C4.5, and Nave Bayes classifiers. Their findings revealed that the accelerometer on the thigh was the most effective at discriminating between activities. This discovery backs up our decision to have our test subjects carry their phones in the handiest location—their jeans pockets. Krishnan et al. [16] used two accelerometers to collect data from three users to recognize five activities: walking, sitting, standing, running, and lying down. According to this article, data from a thigh accelerometer is insufficient for identifying activities, including sitting, lying down, walking, and running. As a result, several accelerometers were required.

Tapia et al. [11] collected data from five accelerometers placed on various body sites for twenty-one individuals and utilized it to build a real-time system that recognized thirty gymnasium activities. Incorporating data from a heart monitor in addition to accelerometer data resulted in a slight improvement in performance. Mannini and Sabitini [19] identified twenty behaviors from thirteen users using five tri-axial accelerometers mounted to the hip, wrist, arm, ankle, and thigh. Three postures (sleeping, sitting, and standing) and five movements (walking, stair climbing, running, and cycling) were identified using a variety of learning strategies. For activity recognition, Foerster and Fahrenberg [5] used data from five accelerometers in one set of studies and two accelerometers in another. The study included 31 male individuals. A hierarchical classification model was developed to discriminate between postures, such as sitting and lying at specific angles, and actions, such as walking and climbing stairs, at different rates. To achieve activity recognition, researchers used a combination of accelerometers and other sensors.Parkka et al. [10] used twenty different types of sensors to create a system that recognizes activities such as lying, standing, walking, running, football, swinging, croquet, playing ball, and using the toilet in specific locations.

By combining a sensor module worn in the pocket with a digital compass worn at the user's waist, Lee and Mase [6] created a system that recognizes a user's location and actions, such as sitting, standing, walking on level ground, walking upstairs, and walking downhill. Some studies have focused on combining multiple types of sensors for activity recognition in addition to accelerometers.

Choudhury et al. [13] used a multimodal sensor device with seven different types of sensors to recognize activities such as walking, sitting, standing, ascending and descending stairs, elevator up and down, and brushing one's teeth. Cho et al. [12] identified nine activities using a single tri-axial accelerometer and an embedded image sensor worn at the user's waist. Although these multi-sensor systems demonstrate the tremendous potential of mobile sensor data as additional types of sensors are integrated into devices, our approach demonstrates that only one type of sensor—an accelerometer—is required to recognize most daily activities. As a result, our solution provides a simple and easy-to-implement approach to doing this work.

Other research, including ours, have concentrated on the use of a single accelerometer for activity identification. Long, Yin, and Aarts [17] used a triaxial accelerometer worn at the user's waist without regard for orientation to collect accelerometer data from twenty-four individuals. To distinguish walking, jogging, running, cycling, and sports, data was collected naturally, and decision trees, as well as a Bayes classifier mixed with a Parzen window estimator, were utilized. Several academics have proposed using readily available mobile devices, such as cell phones, to address the challenge of activity recognition. However, previous techniques did not take advantage of the sensors built into mobile devices themselves.

Gyorbiro et al. [23], for example, employed "MotionBands" affixed to each subject's dominant wrist, hip, and ankle to discern between six different motion patterns. A tri-axial accelerometer, magnetometer, and gyroscope were included in each MotionBand. The data obtained by the MotionBand was then communicated to a smart phone carried by the user for storage. Ravi et al. [8] obtained data from two users wearing a single accelerometer-based device, which was subsequently transferred to the user's HP iPAQ mobile device. Researchers compared the performance of eighteen different classifiers for activity recognition using this data.

Lester et al. [9] recognized eight everyday activities from a small group of users using accelerometer data, audio data, and barometer sensor data. While this research could have generated accelerometer data using a cell phone, they did not. Instead, the data was created using distinct accelerometer-based devices worn by the person and then stored on a cell phone. A few studies, including ours, did collect data for activity recognition using an actual commercial mobile device. These systems have an advantage over other accelerometer-based systems in that they are unobtrusive and require no additional equipment for data collecting and correct recognition.

Miluzzo et al. [14] investigated the use of commercial smartphone sensors (such as a microphone, accelerometer, GPS, and camera) for activity identification and mobile social networking apps. To handle the activity recognition task, they gathered ac-

celerometer data from ten users and used J48 to create an activity recognition model for walking, running, sitting, and standing. This model struggled to discriminate between sitting and standing activities, a task that our models quickly accomplished.

Yang et al. [18] used the Nokia N95 phone to create an activity identification system that distinguishes between sitting, standing, walking, running, driving, and bicycling. This research also looked into the usage of an activity recognition model to create physical activity diaries for users. Although the study obtained rather high prediction accuracies, stair climbing was not taken into account, and the system was trained and validated using just data from four users.

Brezmes et al. [15] developed a real-time system for distinguishing six user behaviors using the Nokia N95 phone. Because their technology trains an activity recognition model for each user, there is no universal model that can be applied to new users with no training data. This restriction does not apply to our models.

Mekruksavanich et al. [36] proved that the usage of microelectronics mechanical systems sensor technology in smart wearable devices has substantially increased HAR capabilities through the use of machine learning and deep learning approaches. On the other hand, processing lengthy dependent sequence samples with noisy data remains a substantial difficulty, affecting classification time and accuracy. They examined noise-tolerant deep learning models for detecting physical activities from noisy smartphone data. They also examined the performance of the five most common deep learning networks for HAR under various amounts of Gaussian noise. The results showed that the BiGRU network was remarkably resistant to Gaussian noise at various levels.

S. Al Farshi Oman et al. [38] introduce a branch CNN and LSTM structure for identifying human activity that produces cutting-edge results in their research. Due to the noisy sensor data, domain analysis and signal processing are required to extract features from the raw data and fit them into machine learning models. The latest breakthrough in Deep Learning Models allows features to be learned automatically rather than by hand. Deep learning techniques such as CNN and RNN are heavily used in this field. They conducted experiments on the SHOAIB Al and UCI HAR datasets, which yielded superior results than the previous method.

S. Mekruksavanich et al. [37] investigate how motion sensors respond during the activity recognition process by employing deep learning (DL) algorithms to identify physical movements based on smartphone sensor data. The DL models are evaluated using a dataset from 10 persons doing eight activities while carrying cell phones in various settings. Their experimental results show that, except for the magnetometer, each sensor can play a leading role in HAR to detect human activity. Furthermore, they discovered that combining sensors enhances overall recognition interpretation only when their performances are poor.

A. N. Tarekegn [39] provided a novel S-HAR approach based on ensemble deep learning with sensor nodes attached to the waist, chest, leg, and arm. Three deep learning networks are implemented and trained utilizing a publicly available dataset

that includes wearable sensors from eight human actions. The results showed that the suggested Ens-ResNeXt model outperforms existing strategies regarding accuracy and F-score.

Human Actions Recognition, or HAR, is a growing research field with applications in healthcare, human-computer interaction, assistive learning, and many other fields. Despite significant progress in this subject over the last few years, there is still a need to construct robust hybrid machine learning models for human action identification that must match the application's objectives, have a reliable prediction, and have a high recognition rate. A small amount of work has been done on the detection of anomalies while completing an action. We thoroughly reviewed the related works, and examined several tools and methodologies for human activity recognition, such as machine learning algorithms and neural network techniques. The approaches were tested on various datasets and yielded diverse results depending on the ambient circumstances and type of data used, such as accelerometer data, other sensor data, sensor placement, and implementation methods. These strategies are contrasted in terms of context as well as computational difficulties. Finally, difficulties in recognizing human behaviour are discussed. We conclude from the previous work that there is no single way that is best for recognizing any activity; so, in order to select a specific method for the desired application, numerous variables must be considered. We can say, that researchers have been working on HAR for a long time. However, it is still challenging because researchers have found that every human has a unique activity style, so the pattern match may be difficult. Moreover, the models get confused when the users do a no-moving activity, such as lying or sitting. However, most of the models show comparatively better results with moving objects. In our study, we tried to find out the activities using smartphone accelerometer data, and we made an Android application to represent the outputs in real-time. So, even though there are multiple methods, some difficulties remain unresolved, and the implementation of a HAR system into an Android application is still very inadequate. So, we need more research and real-life implementation in the human activity recognition domain.

Chapter 3

Methodology

Long short-term memory (LSTM) [3] is used in deep learning. It is able to learn long-term dependencies and sequence prediction. Due to its ability to understand long-term connections between data time steps, LSTM is frequently used to learn, analyze, and classify sequential data. We used LSTM to built our model.

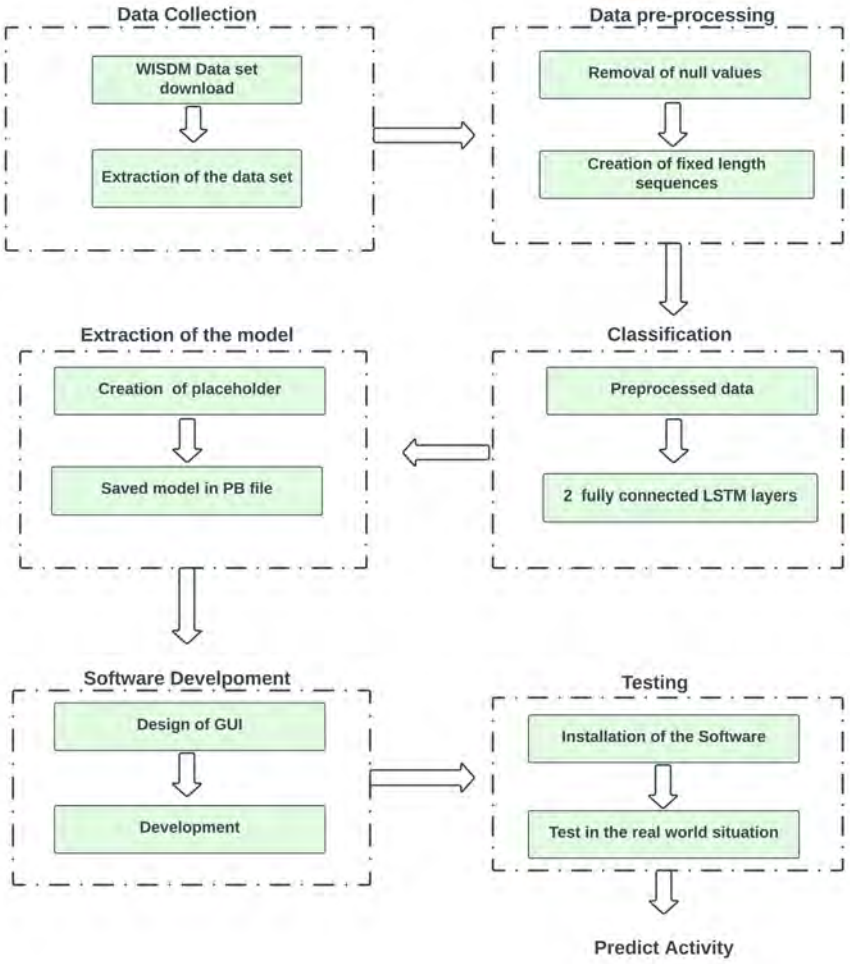


Figure 3.1: Top level overview of the HAR system

First, we collected the dataset from Fordham University's website. The dataset was compressed into a zip file, so we extracted the file and got a raw text file. The dataset was stored in the text file. In the data-preprocessing module We checked

the null values and removed them. We create fixed-length sequences; each generated sequence contains 200 training examples. After that, 2 fully connected and 2 LSTM layers were used for classification. We create placeholders, which are a particular type of tensor used to supply real data to the model during its execution. Next, we save our trained model in a Protobuffer(PB) file, as the Android application can use the deep learning model. We used Android Studio as an IDE and Java as the programming language in the development module. In the end, we installed the application on a smartphone and test the application in real-world situations.

3.1 Data Collection

Human activity recognition is helpful for different analytical approaches like public health, human-machine interaction and game analysis. Applying Artificial Intelligence (Al) and Deep Learning (ML) models can effectively enrich the performance of the HAR system. This approach can identify human activities correctly. We need a proper dataset and an effective deep-learning model to enhance the model's performance. To classify the different activities correctly, collecting a dataset containing a diverse range of real-world human activities is essential. The dataset should have features that have attributes of every activity. The performance of the deep-learning model depends on the quality and quantity of the training dataset they are fed. We have collected relevant datasets from many online sources named the 'WISDM' dataset. This dataset contains the smartphone's accelerometer's x,y, and z-axis values. This dataset was selected because it shows a significant accuracy in detecting human activity recognition. Real-world situation is complex and challenging because scientists have discovered every person has a unique activity style. This variety can reduce the performance of the system. Therefore, we need a dataset with a good number of examples, as our model can generate a typical pattern associated with different activities. We used data from the Wireless Sensor Data Mining (WISDM) Lab [20]. It is available on the Fordham University website. The data was gathered in a controlled laboratory environment. This dataset is made up of 1,098,207 rows and six columns. There are no values that are missing. We will try to identify six activities: walking, jogging, upstairs, downstairs, sitting, and standing. Let's take a deeper look at the data.

Raw Time Series Data

- Number of examples: 1,098,207
- Number of attributes: 6
- Walking: 424,400 (38.6%)
- Jogging: 342,177 (31.2%)
- Upstairs: 122,869 (11.2%)
- Downstairs: 100,427 (9.1%)
- Sitting: 59,939 (5.5%)
- Standing: 48,395 (4.4%)

Table 3.1: Data sample

User	Activity	Timestamp	X-axis	Y-axis	Z-axis
33	Jogging	49105962326000	-0.6946377	12.680544	0.50395286
18	Upstairs	33338202227000	4.99	5.41	-4.1814466
15	Walking	1224722240000	10.65	8.81	-3.173541
16	Sitting	1002862346000	5.56	2.72	8.16
24	Standing	16157221475000	0.65	9.34	0.46
24	Downstairs	15977701486000	-9.11	10.65	12.53

The table 3.1 shows the first column is the user's ID number; the second column is the activity name; the next is the timestamp, then the three-axis values of the accelerometer, i.e., x, y, and z, respectively.

3.2 Data Preprocessing

This LSTM model expects fixed-length sequences as training data. We used a familiar method for generating these. Each generated sequence contains 200 training examples and we visualize the data based on different features here. We can see there

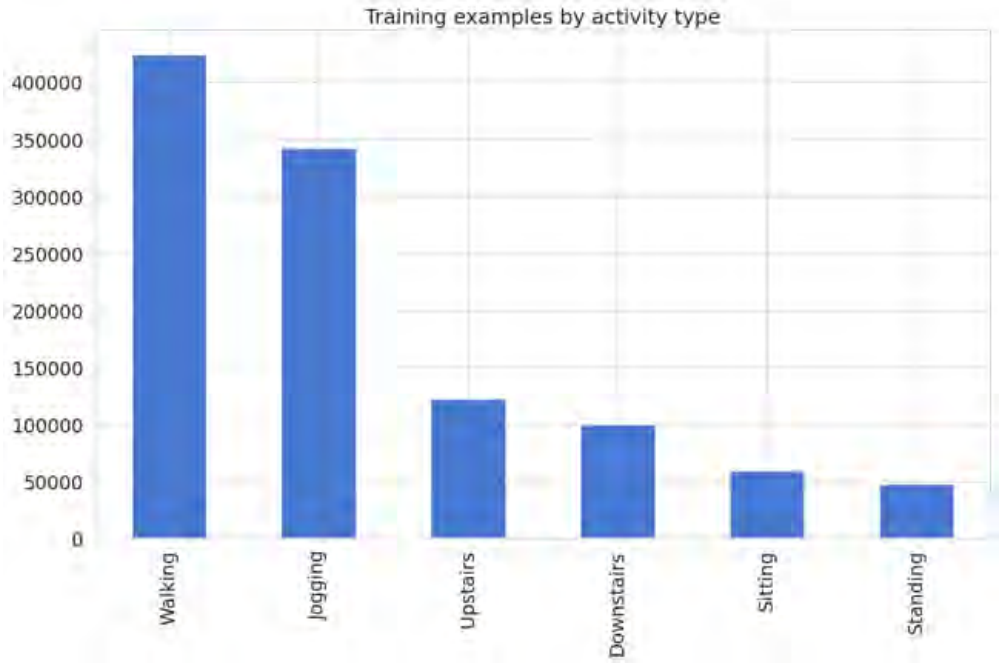


Figure 3.2: Training examples by activity type

are six activities have been demonstrated in the figure. Walking activity has the highest training sample where Standing activity has the lowest training sample.
We have total 36 user and their given data from their smartphone has been illustrated in Figure 3.3. Now we showed the acceleorometer's X, Y and Z axies signal based on the activity in the Figure 3.4, Figure 3.5, Figure 3.6 and Figure 3.7 bellow. We can see that there are clear differences among the axis based on different activities. These differences create a pattern for an activity and using this pattern our model can face real time data to detect a user's activity.

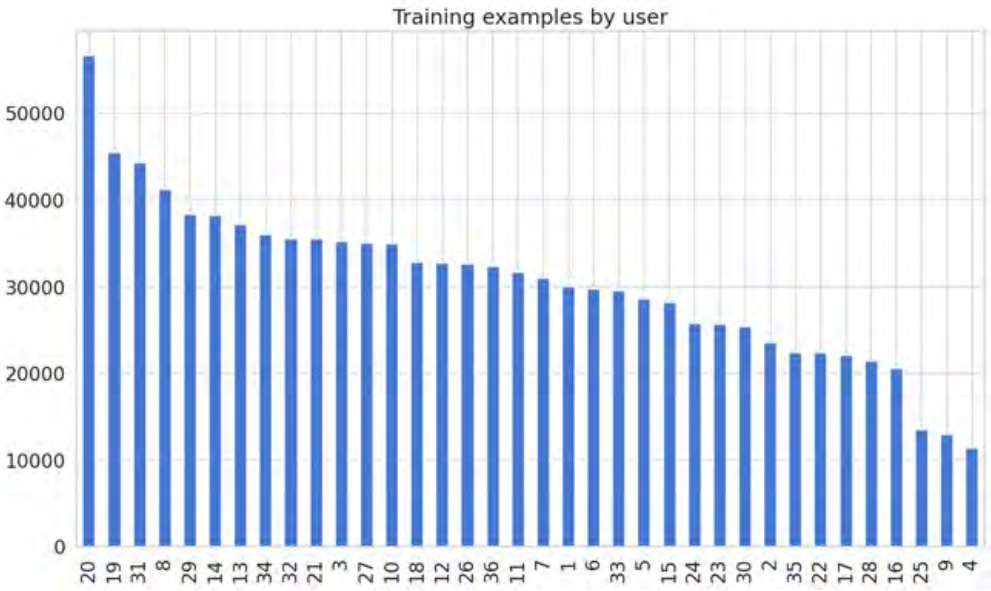


Figure 3.3: Training examples by user

Figure 3.4: Sitting

Figure 3.5: Standing

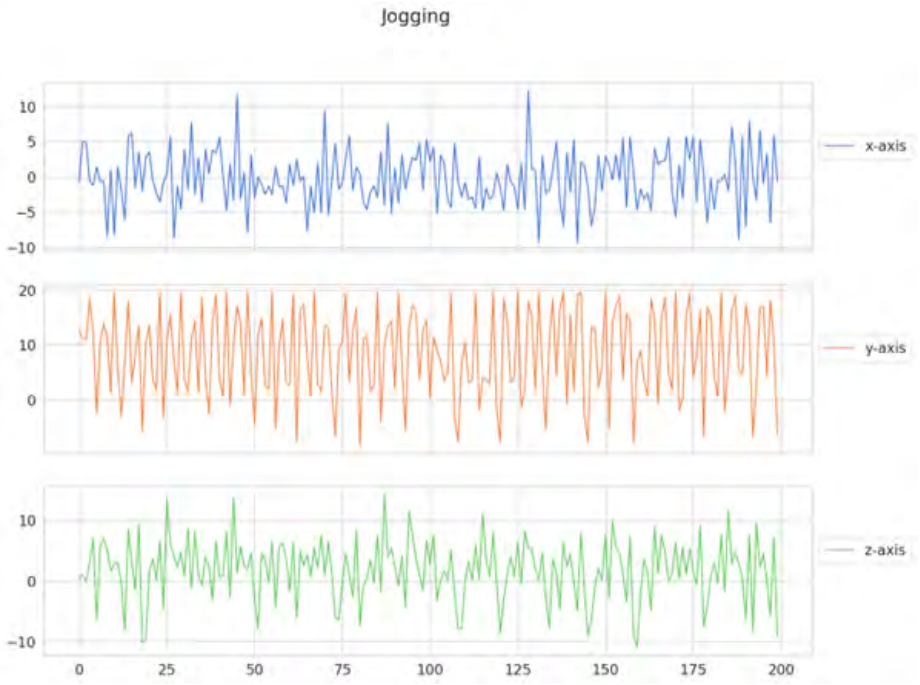


Figure 3.6: Jogging

Figure 3.7: Walking

3.3 Model Specification

Our human activity recognition system combines the Long Short-Term Memory (LSTM) [3] and the WISDM dataset[21]. Long Short-Term Memory (LSTM) is a Recurrent Neural Network (RNN) [1] form designed to handle sequential data such as time series, audio, and text. Because LSTM networks can learn long-term dependencies in sequential data, they are well- suited for applications such as language translation, speech recognition, and time series forecasting. This comprehensive system shows excellent performance in identifying human activities in real-life situations. Combining the deep-learning LSTM model and fine-tuned WISDM dataset enhances the accuracy and precision of the system. This good relationship between computational efficiency and performance encourages us to implement the system into an Android application.

3.3.1 Long Short-Term Memory (LSTM)

LSTM is a type of recurrent neural network[33]. It can understand order dependence in sequence prediction challenges. LSTMs are a complicated field of deep learning. LSTM has various advantages, making it an effective tool for modelling sequential data. The advantages are as follows:

- LSTM networks can catch long-term dependencies in sequential data, which typical neural networks cannot.
- LSTM cells use input, output, and forget gates to recall or forget information selectively. This allows the network to keep only valuable data while filtering out noise.

- Because LSTM can accept variable-length input and output sequences, it is suitable for various tasks, such as voice recognition, auto-translation, and image labelling.
- LSTM can tolerate noisy data and missing values in the input sequence, which is beneficial in real-world applications where data may be incomplete or noisy.
- LSTM can be efficiently taught via back-propagation through time (BPTT) and readily parallelised, making it scalable to enormous datasets and computationally efficient.

Overall, LSTM is a powerful tool for modelling sequential data that has been successfully used for various tasks, including natural language processing, voice identification, image captioning, and others. We are dealing with sequential time series data, and LSTM performs well on these datasets, so we selected LSTM to build our model. We will discuss the comparative study later.

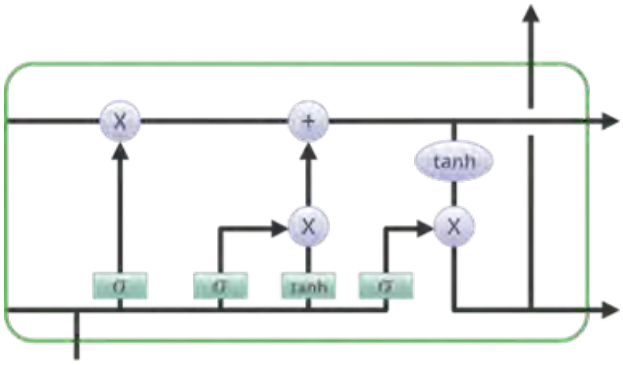


Figure 3.8: LSTM Structure

The LSTM has a chain structure that includes four neural networks and various memory blocks known as cells. The cells store information, and the gates perform memory modifications. There are three entrances.

3.3.2 Forget Gate

The forget gate removes information no longer helpful in the cell state. Two inputs, time input x_t and prior cell output h_{t-1} are sent into the gate and multiplied using weight matrices before bias is added. The result is processed via an activation function, which produces a binary output. If the output for a certain cell state is 0, the information is forgotten; if the output is 1, the information is saved for future use. The forget gate's equation is:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.1}$$

Here,

- W_f stands for the weight matrix associated with the forget gate.
- $[h_{t-1}, x_t]$ represents the concatenation of the current input and the prior hidden state.
- σ is the sigmoid activation function.

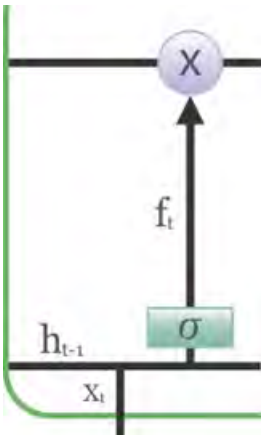


Figure 3.9: Forget Gate

3.3.3 Input Gate

The input gate is responsible for adding important information to the cell state. First, the information is regulated using the sigmoid function, and the values to be remembered are filtered using the $[h_{t\text{-}1}]$ and x_t inputs, similar to the forget gate. The tanh function is then used to generate a vector with values ranging from -1 to +1 that encompasses all of the possible values from $[h_{t\text{-}1}]$ and x_t. Finally, the vector and controlled values are multiplied to acquire the valuable information. The input gate equation is as follows:

$$i_t = \sigma(W_i \cdot [h_{t\text{-}1}, x_t] + b_i) \tag{3.2}$$

$$\widehat{C}_t = \tanh\left(W_c \cdot [h_{t\text{-}1}, x_t] + b_c\right) \tag{3.3}$$

$$C_t = f_t \odot C_{t\text{-}1} + i_t \odot \widehat{C}_t \tag{3.4}$$

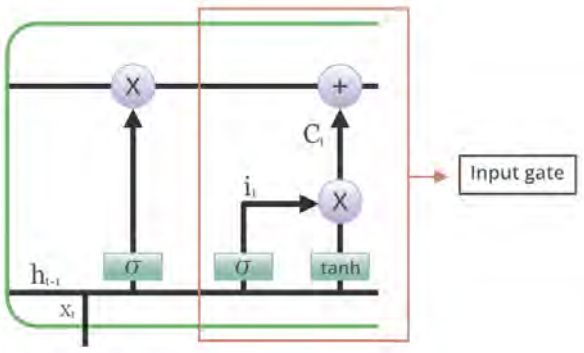


Figure 3.10: Input Gate

Here,

- $\odot$ means element-wise multiplication.
- tanh is the activation function.
- C_t means cell state.
- $\widehat{C}_t$ means candidate for cell state at timestamp (t).

3.3.4 Output Gate

The output gate obtains valuable information from the current cell state and presents it as output. A vector is created by applying the *tanh* function on the cell. The information is then regulated using the sigmoid function and filtered based on the values to be remembered via inputs $[h_{t-1}]$ and x_t. Finally, the vector and the controlled values are multiplied and sent as output and input to the following cell. The output gate equation is as follows:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3.5}$$

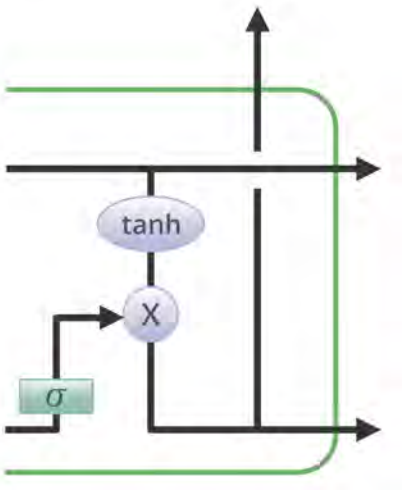


Figure 3.11: Output Gate

Chapter 4

Application Development

The process of conceiving, specifying, designing, programming, documenting, testing, and bug fixing that goes into building and maintaining applications, frameworks, or other software components is known as software development [31]. Software development entails developing and maintaining source code, but it also encompasses all steps from the conception of intended software through its final manifestation, often in a planned and disciplined process. In the development phase, we first determine how our application works, and based on these criteria, we divide the development into two sections i.e., design the GUI and the coding. We use Android Studio for both designing purpose and coding. Android Studio is Google's official integrated development environment for the Android operating system. It is based on JetBrains' IntelliJ IDEA software and is specifically developed for Android software development.

4.1 Permission and Dependencies

Android permissions offer tools for enhancing user awareness and limiting an app's access to private information. As we want to use the accelerometer data so, we need to get permission from the Android. We edit our Android manifest file. The Android manifest file assists in declaring the permissions that an app must have in order to access data from other apps. The Android manifest file also defines the package name of the app, which aids the Android SDK in constructing the app. We enable the permission of the accelerometer as our application can read the accelerometer sensor's data of device. Dependencies are external elements created by other developers that are acceptable to Android for performing a specific task. Dependencies enable the inclusion of external libraries or other library modules into an Android project. For example, if we want to display an image in our project app, we can use the Picasso or Glide libraries. So, dependencies are useful since they simplify our work by allowing us to just include it and begin using the classes and functions without having to think about how it was constructed. We used TensorFlow in our project so, we need to use TensorFlow dependencies in the build.gradle file. We edit our build.gradle file. This file is the foundation of our build process, containing all of the instructions required to compile an Android app from source. We set implementation of TensorFlow version to 1.8.0.

4.2 Designing

A user interface (UI) and a user experience (UX) are combined in application design. User interface Design is concerned with the entire design of the program, such as colors, fonts, and the overall appearance and feel. The user experience, on the other hand, will always prioritize functionality and usability. Application design is also a major factor in determining which applications people will use on a regular basis and which applications they will avoid in the future. We're discussing application design. We must talk about both process sections, as well as user interface and user experience. User interface design will concentrate on what users must do to ensure that the interface contains the features that make it easy to access, understand, and utilize, as well as to facilitate those tasks. User interface design will merge interaction design, visual design, and information architecture concepts. Understanding a user experience is accomplished by better understanding a user's "journey" when interacting with an application. Users must understand what information has the most value , where they anticipate to find it, how to obtain it, how to engage with it, and other factors. They look for the flaws and locations that make the experience unsatisfactory or less than ideal. They then make suggestions for improvement. People will abandon the application if it is hard to use and confusing. It may not be as effective in the long run. We carefully designed the application interface and tried to find user experiences during designing process. We came to the conclusion that people would prefer a table view layout for the human activity recognition application. Moreover, their user experience says that the application size must be smaller in size and compatible with most smartphones, so we consider these criteria. Android UI design involves developing the graphical user interface for our applications using prebuilt Android UI components such as structured layout elements and UI controllers. XML is an abbreviation for extensible Markup Language, which is a text-based data description language. Because XML is extensible and extremely adaptable, it is utilized for a variety of purposes, including creating the UI layout of Android apps. For the designing purpose we used the xml file. In the application the name of the file was activity_main.xml. In this file we performed coding for the designing. All the activity i.e., Walking, Sitting, Jogging, Standing, Downstairs, and Upstairs should be shown in a table view, therefore we set the table view layout for showing the activities.
We set all the activities name left of the screen and their associated probability was shown to the right. We define the layout, text size, text alignment, text style, id of an activity and padding. After the designing we have our desired view of the application which is shown below.

4.3 Development

In the development, coding is used by computer programmers to connect with computers and instruct them on what to perform. Coding is the process of writing with programming languages. In this section we used java language for programming purpose. We created tow java class then first one is MainActivity.java and the second one is TensorFlowClassifier.java.

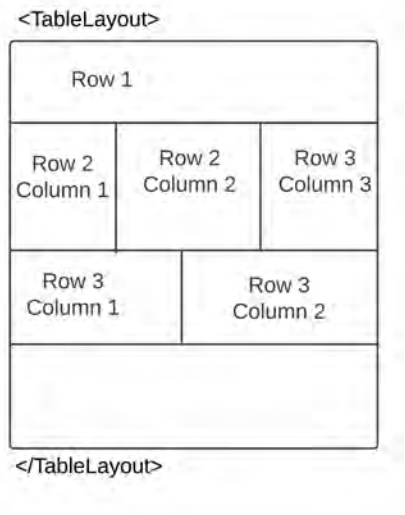


Figure 4.1: TableView layout

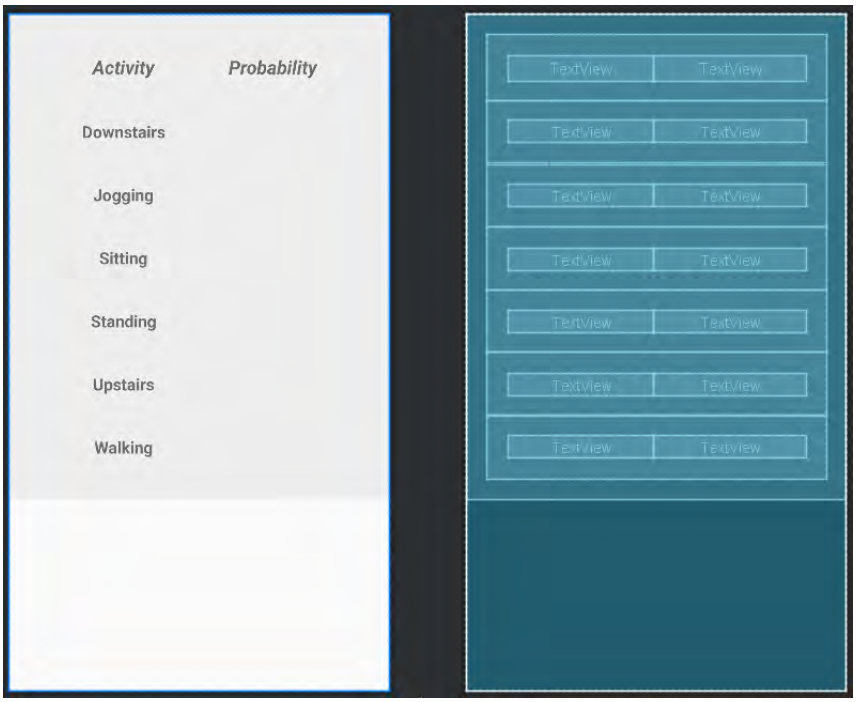


Figure 4.2: The application outlook

4.3.1 Importing Necessary Libraries

An Android library has the same structure as an Android app module. It includes source code, resource files, and an Android manifest, as well as everything else required to develop an app. Furthermore, Support Libraries include convenience classes and functionality not available in the core Framework API enabling faster development and maintenance across a wider range of devices. The Android Support Library developed from a single binary library for apps to a suite of libraries for app development. We import the necessary "TensorFlowInference" which is a Java interface for using a pre-trained TensorFlow model in an Android application. It allows us to easily import a stored "TensorFlow" model and conduct inference on it from an Android app. Then we define the TensorFlowClassifier first which is public class.

4.3.2 TensorFlowClassifier Class

This java class is responsible to take the sensor data from the smartphone and put the data as an input into our model and perform the estimation of the activity of the user. We load the tensorflow inference library inside the class. This library helps to run a TensorFlow model on a device to create predictions based on input data. To infer with a TensorFlow model, we must first run it via an interpreter. The TensorFlow interpreter is intended to be lightweight and quick. To achieve minimal load, startup, and execution time, the interpreter employs static graph ordering and a specialized memory allocator. Durinng the deep-learning LSTM model creation we used Google colab to write code for the model. We write code in python and used TensorFlow, which is a free and open-source machine learning and artificial intelligence software library. It can be used for a variety of applications, but it focuses on deep neural network training and inference. After export the model in a protocol buffer file (PB) we must use that file in our android application. To use it we must use the TensorFlow inference library. So, we called the System.loadLibrary method inside the class to use the TensorFlow inference library. Then we call the "frozen_model.pb", the PB file name, and assign its location as the android asset folder. We assign the input nodes and output nodes. We also declarer input size and output size in the TensorFlow class. Finally, we create a method named "predictProbabilities", which feeds the trained deep-learning LSTM model the real-time data from smartphones and predicts the probability of human activity.

4.3.3 Pre-trained Model in the Android Application

So, we initiate the INPUT_NODE as "inputs" and OUTPUT_NODE as "y_." We set the INPUT_SIZE, which specify the shape of the input tensor. The input tensor must have the form of a 3D tensor with dimensions (batch_size, time_steps, input_size), where: batch_size: The number of samples in each training data batch. time_steps: The number of time steps in the sequence of input data. input_size: The number of features in the input data. The shape of a tensor describes how many dimensions the tensor has, and how many elements are in each dimension. Here, we have a tensor with shape (1, 200, 3), this means that the tensor has three dimensions, the first dimension has 1 element, the second dimension has 200 elements, and the third

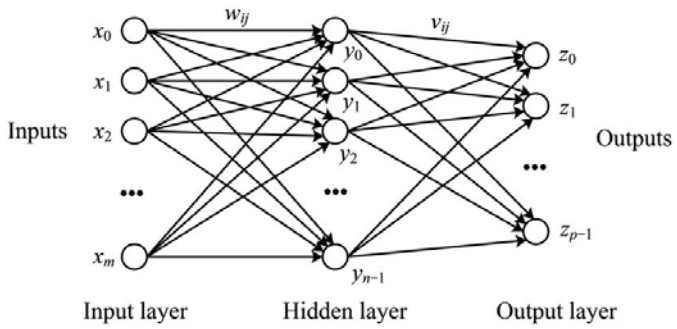


Figure 4.3: Nodes of a deep-learning model

dimension has 3 elements. As we have six activities so we set the OUTPUT_SIZE value to 6.

4.3.4 Prediction Estimation

Predictive modeling is at the heart of predictive analytics. It's more of a strategy than a method. Because predictive models often involve a machine learning algorithm, predictive analytics and machine learning go hand in hand. These models can be trained to adapt to new data or values over time, producing the outputs that the business requires. Predictive modeling is closely related to machine learning. Predictive models are classified into two categories. Classification models determine class membership, while regression models predict a number. Algorithms are then used to construct these models. Data mining and statistical analysis are carried out by algorithms, which identify trends and patterns in data. Predictive analytics software solutions will have built-in algorithms for creating predictive models. The algorithms are referred to as 'classifiers' since they determine which set of categories data belongs to. We create a "predictProbabilities" method which is responsible for predicting the human activities. In this method we defined an array named "results", which will store the probability of each activity. We feed the input data to input nodes, run output nodes and fetch the output node. After that we store all the values in the" results" array.

4.3.5 MainActivity Class

An activity is a single, well-defined function that your user can take. We have an activity that displays the activity labels with the prediction probabilities, as well as a text-to-speech feature. Activities are often associated with a single screen and are created in Java. The appearance of the screen is described by a layout. Layouts are XML files that tell Android how to arrange the various screen elements. The device starts an app and creates an activity object. The layout is specified by the activity object. The activity instructs Android to show the layout on the screen. The layout presented on the device is interacted with by the user. In response to these interactions, the activity executes application code. The action refreshes the screen that the user sees on the device. The main Activity handles the user interface from sensors and feeds them into the activity classifier. We define the sample size and set its value to 200. The main activity we have created for the

Android application displays the layout on the smartphone screen. The users can see all the activity's names on the screen. The main activity collects the predicted results from the TensorFlow class and shows the results on the screen by reacting to the codes written in Java. It also provides a text-to-speech feature that indicates the user's current activity.

4.3.6 Storing the Accelerometer Data

A Java List is an ordered collection. Java List is an interface that extends the collection interface. Java List allows us to specify where we want to insert an element. We can access elements by index and also search for elements in the list. The List interface is used to hold the ordered collection. It is a collection's child interface. It is an ordered collection of objects that can store duplicate values. Because List keeps the insertion order, it allows both positional access and element insertion. We have to keep the obtained data from the accelerometer. The accelerometer has three axes; so we create three lists, i.e., x, y, and z, for storing accelerometer data. Each variable can be used to share data across instances of that class. The data type is float. When sensor values are changed, they are automatically updated by the method "onSensorChanged " and stored in the three lists. In this method, we call another method named "activityPrediction()" which use these three axes values and combine them to predict an activity.

4.3.7 Text View of the Main Activity

A TextView a simple widget which shows text to the user and, if needed, allows them to alter it. A TextView is a full-featured text editor, but the base class is set to prevent editing. It should be used for uneditable material that the user want to read but not modify. By this widget, we represent the text view for each activity labeling, i.e., joggingTextView, sittingTextView, standingTextView, etc. This text view also shows the probabilities for each activity on the left side of the screen. Each text view element is set as the row of the table view. The text view of the main activity is basically a table view with activity label names on the left side, and we have the estimated probabilities on the right side. The title "Activity" and "Probability" is shown in the bold italic format, and its text size is 20sp. The activity values are shown in standard bold format, and the text size is 18sp.

4.3.8 Showing the Layout

The structure of a user interface in an application, such as an activity, is defined by a layout. The layout's elements are all built utilizing a hierarchy of View and ViewGroup objects. A View usually refers to something that the user can see and interact with. We call "findViewById" a method for locating the view in the layout resource file associated with the current Activity of the application. So, when we create the table view, we define the TableRow. In TableRow, we assign the TableRow id as title_row, which is the id of the TableRow. We define layout width as match_parent. Match_parent is an attribute that specifies that the width or height of a view should be the same as the width or height of its parent view. This signifies that the view will occupy the same amount of space as its parent. This is commonly

used when designing a layout with numerous views because it assures that the views are all the same size. We set the layout height as match_parent. Padding is the space between the content we want to display and its border. Padding is used to generate extra space within the content. We can use padding to offset the view's content by an exact number of pixels. For example, a left padding of two pushes the view's content two pixels to the right of the left edge. Any of the specific sides can be used to provide padding to a widget. We add padding as 22dp. Text views are used by many Android applications to show text within them. Within our XML layout, the text view is aligned in several ways. We cannot apply margins from all sides to align TextView for different screen widths. For different screen widths, we must align the text view centrally to vertically and centrally to horizontally of the overall height and breadth of the Android device screen. We set text alignment as center.

4.3.9 Storing Each Activity Probability

An array is a container object that holds a fixed number of single-type values. The length of an array is determined when it is constructed. Its length is fixed after it is created. An array's items are referred to as elements, and each element is accessible by its numerical index. To store each activity probability, we define an array named "results." This array holds all the predicted activity values in float format. It always shows the updated values, which were returned as an array from the "predictProbabilities" method of the TensorFlowClassifier class. We call this "results" array in the "activityPrediction", which is a method of the main activity. The return type of the "result" array is float type.

4.3.10 Showing the Human Activities

In this developing section, we want to show the activity labels and their probabilities in a text-view format. Basically, the text view is shown in the application's main activity. So, we write all the necessary logic in the main activity class. We create a method "activityPrediction" which is responsible for showing the probability of each activity. But if we want to show it on the main activity, we need to convert it to string. So, we use a built-in method, "Float.toString." Because the "setText" method is responsible for showing the text view on the main activity and doesn't support float data type, we convert it to string. Each value we get are float data type so we need to convert it to string if we want to show it as a text view. So, we call "Float.toString" method. When we get the probability convert it to two decimal places calling the "round" method. Then we call "setText" method to display the activities along with their probability values. Each time the accelerometer sensor value changes, the data is stored in the associated list, i.e., x, y, and z. Then we call the "activityPrediction" method, which sets the probability of all the activities after facing the sensor data. Again, the "setText" method displays the updated value of the main activity of our application. In this way, we can see the real-time view of the predicted human activities of the user. A higher value of a labelled activity indicates the most possible activity done by the user.

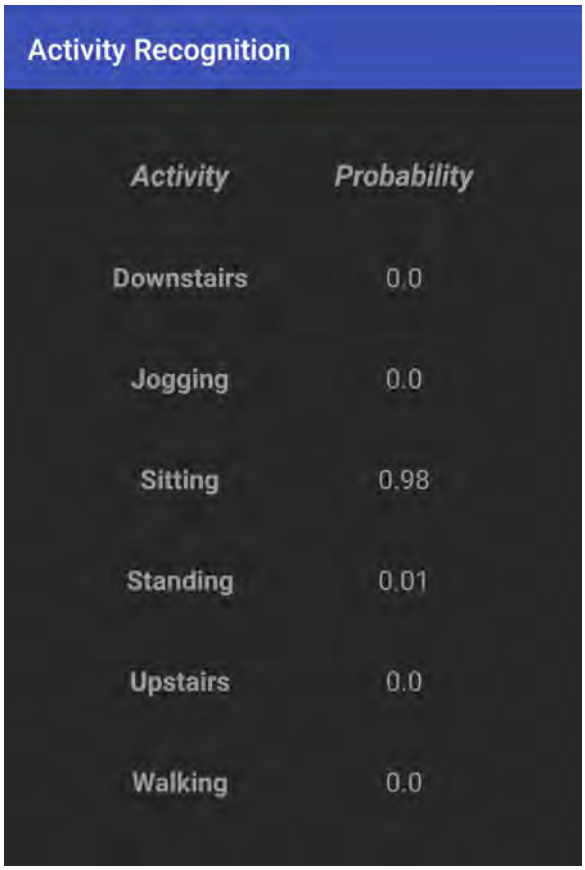


Figure 4.4: Probability shown in float format

4.3.11 Text-to-Speech

Text-to-speech (TTS) library, sometimes known as the "read-aloud" library, has created how humans interact with printed text. Text-to-speech (TTS) library translates text into spoken words using a computer-generated voice, which may be sped up or slowed down in most situations, making them tools with a wide range of uses available to adults, business professionals, children, and students. We want to make our application more user-friendly. Therefore, we used a text-to-speak feature, which converts the activity names to voice. With the voice feature, users can identify any activity even if they don't look at the screen. They can put the smartphone in their pockets, run the application, and recognize the voice. When this feature generates the voice "Walking", that means the user is walking. Then, we set the logic for the text to speech in the "onInit" method. When an activity has the highest probability, the "Text to Speech" library converts the activity name into voice. We used "Text to Speech" library that provides Text-to-Speech functionality for our Android application, which converts the text written on the screen to audio. So, the user can listen to their Activity from the application in real time.

Chapter 5

Performance Evaluation

5.1 Performance Metrics

To determine the performance of our HAR system we utilize some performance measuring mathematical methods [32]. The results were assessed using well-known performance metrics such as confusion matrix, accuracy [34], sensitivity, specificity, and F-score. A confusion matrix is a table that is used to define a classification algorithm's performance. A confusion matrix visualizes and summarizes a classification algorithm's performance. It is calculated by number of true positive (TP), false positive (FP), true negative (TN), and false negative (FN) values. Accuracy is defined as the ratio of correctly classified instances to the total number of cases under evaluation.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \times 100 \tag{5.1}$$

The F1-score is a machine learning evaluation metric that determines the accuracy of a model. It combines a model's Precision and Recall scores. F-1 score makes use of precision and recall to estimate accuracy of classification.

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

$$F - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5.4}$$

A classifier's sensitivity is its capacity to recognize positive class patterns. It is possible to compute it as follows:

$$Sensitivity = \frac{TP}{TP + FN} \tag{5.5}$$

A classifier's specificity is its capacity to recognize negative class patterns. It is possible to compute it as follows:

$$Specificity = \frac{TN}{TN + FP} \tag{5.6}$$

5.2 Result Analysis

We will discuss all performance metrics to monitor and measure the performance of our model.

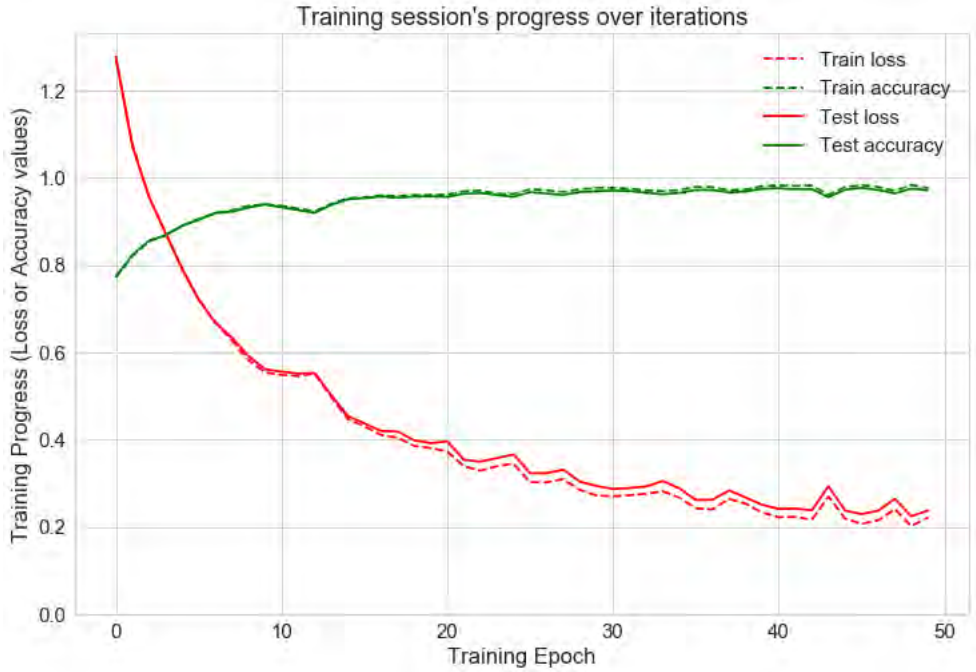


Figure 5.1: Training sessions progress over iterations

In Figure 5.1, we have taken the Training Epoch along the horizontal axis and the Training Progress along the vertical axis. Here, we want to measure the Train loss, Train accuracy, Test loss, and Test accuracy over the iterations. We run 50 epochs for this purpose.

Train loss measures the model's error on the training set during the training of the model. At the beginning of the training, the Train loss (indicated by the broken red line) was pretty high, which is around 1. It means our model was making errors at the beginning. But, over the iterations, our model reduced the Train loss to 0.2. This measurement states that our model made a few errors at the end of the training session.

Train accuracy measures how well it performs during training. It is the percentage of successfully predicted labels over the total number of training samples. During the first stage of the training, the Train accuracy (indicated by the broken green line) was below 0.8. But when the training was finished, the Train accuracy increased to 0.97. It means our model was precisely predicting labels at the end of the training.

Test loss is used to estimate the errors or loss of a deep learning model on the test set during testing of the model. When we started to test the model, the Test loss (indicated by the red line) was high, around 1. But by the end of the testing, the Test loss was reduced to around 0.25. It indicates our model made a few mistakes at the end of the testing.

Test accuracy of a model measures how well it performs while testing the model. At the beginning of the testing, the Test accuracy (indicated by the green line) was below 0.8. But when the testing was finished, the Test accuracy increased to around 0.97. It means our model could predict labels efficiently at the end of the testing.

Figure 5.2: Confusion matrix

We visualize the model's performance using a confusion matrix. A confusion matrix shows a table of all a classifier's predicted and actual values. It presents a table arrangement of the different outcomes of the prediction and results of a classification task. We take the Predicted Label along the horizontal axis and the True Label along the vertical axis. All the correctly predicted activities are shown diagonally in the confusion matrix. For example, if we consider the second column of the matrix, the model predicted Jogging, but it was Downstairs; this situation happened 116 times. The model predicted Jogging, and it was really Jogging; this situation occurred 3380 times. The model never confuses Jogging and Sitting. The model was also never confused between Jogging and Standing. The model predicted Jogging, but it was Upstairs; this situation happened 116 times. The model wrongly predicted Jogging 265 times, which was actually Walking.
From the confusion matrix, we can see that Walking has the highest correctly predicted number, 3820. Then, Jogging has the second position as the correctly predicted number is 3380. It means our model can predict Jogging as Walking very precisely. But for Downstairs, the model shows relatively low performance, as the correctly predicted value is 140. The model shows a regular result for other activities, i.e., Standing, Sitting, and Upstairs.

We run 50 epochs, showing the accuracy changes after every ten-epoch difference.

Table 5.1: Accuracy of the model

Epoch	**Accuracy**	**Loss**
1	0.7736998796463013	1.2773654460906982
10	0.9388942122459412	0.5612533092498779
20	0.9574717283248901	0.3916512429714203
30	0.9693103432655334	0.2935260236263275
40	0.9747744202613831	0.2502188980579376

We can see from Table 5.1, the model's accuracy was the lowest at the first epoch, which is 0.77, and the loss was high, which is 1.27. After running 10 epochs, the model started to improve. At the epoch number 10, the accuracy and loss were 0.93 and 0.56, respectively. The model achieved 0.97 accuracy, and the loss was 0.25. This table illustrates the model performance increases if we run a significant number of epochs.

Table 5.2: F – 1 scores of the model's classes

Activity Label	**Precision**	**Recall**	**F-1 score**	**Support**
Downstairs	0.46	0.11	0.18	1058
Jogging	0.82	0.92	0.87	3447
Sitting	0.94	0.88	0.91	576
Standing	0.86	0.93	0.89	436
Upstairs	0.51	0.44	0.47	1209
Walking	0.76	0.86	0.81	4255

Table 5.2 shows that we can achieve a good F-score in most circumstances. We typically attain more than 0.8 F-score for the two common activities, Jogging and Walking. Jogging is easier to distinguish than Walking because Jogging has more sudden changes in acceleration. The two stair-climbing activities, i.e., Upstairs and Downstairs, are more challenging to determine because they show similar patterns to the accelerometer. The two non-moving activities, i.e., Sitting and Standing, have the highest F-scores, which are 0.91 and 0.89, respectively.

Here, in Table 5.3, we show the sensitivity and specificity of the model for different instances.

Table 5.3: Sensitivity and Specificity of the model

Activity	**Sensitivity**	**Specificity**
Downstairs	0.27	0.97
Jogging	0.99	0.93
Sitting	0.84	1.00
Standing	0.96	0.99
Upstairs	0.33	0.97
Walking	0.88	0.85

Sensitivity determines how well a machine learning model can recognize the true positives of each available category. It is also known as the true positive rate (TPR). Downstairs and Upstairs have 0.27 and 0.33 sensitivity scores, respectively. As the scores are low, our model has limitations in determining Downstairs and Upstairs. But other activities like Jogging, Sitting, Standing, and Walking have high sensitivity scores, around 0.90, which indicates our model can predict true positive instances in most cases.
Specificity determines how well a machine learning model can recognize the true negatives of each available category. It is also known as the true negative rate (TNR). The table shows that all activities have high specificity scores (more than 0.90 in most cases), indicating our model can predict true negative instances precisely.

5.3 Comparative Study

We apply different machine learning techniques to the dataset and compare the models' accuracies.

Table 5.4: Comparative Study

Model Name	**Accuracy**
Support Vector Machine	0.20
Logistic Regression	0.48
Multi-layer Perceptron Classifier	0.54
Gaussian Naive Bayes	0.57
Decision Trees	0.61
Long Short-term Memory	0.97

We can see that the Support vector machine (SVM) has the lowest accuracy, i.e., around 0.2. The other models, like Logistic Regression, Multi-layer Perceptron classifier, Gaussian Naive Bayes, and Decision Trees, have medium-level accuracy, which is around 0.5 to 0.6. But we can see that Long Short-term Memory (LSTM) has the highest accuracy value, 0.97. Therefore, we chose LSTM as our proposed model.

Chapter 6

Conclusion

This study proposed a unique method for automatically identifying human activity, addressing the critical need for enhanced public health, game analysis, and human tracking systems. The project provided here exhibits the use of a deep learning methodology and its real-world application. On the WISDM datasets, we tested six machine-learning algorithms in this study. However, the LSTM has the highest accuracy on the dataset, nearing 97%. The other machine-learning algorithms, on the other hand, had disappointing outcomes. This project was converted into an Android application. As a result, it is able to detect considerable human activity by using a smartphone. The project's efficiency is demonstrated through practical proof in the form of graphical representation and comparative evaluation. This initiative has the potential to significantly improve the domain of human activity recognition systems by enabling automatic identification.

6.1 Future Works

In human activity recognition, myriad potential areas for additional study have evolved. We still have some confusion about classifying the activities. For the improvement of the accuracy of our project, it is crucial to investigate advanced deep-learning models and combine some approaches. The WISDM dataset was created in the controlled lab environment. As a result, we may create a dataset based on real-life scenarios. In a real-world scenario, this new dataset may show greater compatibility. Furthermore, we can add gyroscope sensor data to the dataset to improve the identification process. We may enhance our Android app by adding additional features. We can calculate an estimated calorie burn based on the duration of a user's activity. We can include more particular actions such as lying, playing, driving, limping, falling, and so on. By adding more resources, we may also improve the design of our application. We can also turn this program into tracking software, which can run in the background on a smartphone and display the results on another device.

Bibliography

[1] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *et al.*, *Learning internal representations by error propagation*, 1985.

[2] R. A. Washburn, K. W. Smith, A. M. Jette, and C. A. Janney, "The physical activity scale for the elderly (pase): Development and evaluation," *Journal of clinical epidemiology*, vol. 46, no. 2, pp. 153–162, 1993.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] G. M. Weiss and H. Hirsh, "Learning to predict rare events in event sequences.," in *KDD*, vol. 98, 1998, pp. 359–363.

[5] F. Foerster and J. Fahrenberg, "Motion pattern and posture: Correctly assessed by calibrated accelerometers," *Behavior research methods, instruments, & computers*, vol. 32, no. 3, pp. 450–457, 2000.

[6] S.-W. Lee and K. Mase, "Activity and location recognition using wearable sensors," *IEEE pervasive computing*, vol. 1, no. 3, pp. 24–32, 2002.

[7] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *International conference on pervasive computing*, Springer, 2004, pp. 1–17.

[8] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, "Activity recognition from accelerometer data," in *Aaai*, Pittsburgh, PA, vol. 5, 2005, pp. 1541–1546.

[9] J. Lester, T. Choudhury, and G. Borriello, "A practical approach to recognizing physical activities," in *International conference on pervasive computing*, Springer, 2006, pp. 1–16.

[10] J. Parkka, M. Ermes, P. Korpipaa, J. Mantyjarvi, J. Peltola, and I. Korhonen, "Activity classification using realistic data from wearable sensors," *IEEE Transactions on information technology in biomedicine*, vol. 10, no. 1, pp. 119–128, 2006.

[11] E. M. Tapia, S. S. Intille, W. Haskell, *et al.*, "Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor," in *2007 11th IEEE international symposium on wearable computers*, IEEE, 2007, pp. 37–40.

[12] Y. Cho, Y. Nam, Y.-J. Choi, and W.-D. Cho, "Smartbuckle: Human activity recognition using a 3-axis accelerometer and a wearable camera," in *Proceedings of the 2nd international workshop on systems and networking support for health care and assisted living environments*, 2008, pp. 1–3.

[13] T. Choudhury, G. Borriello, S. Consolvo, *et al.*, "The mobile sensing platform: An embedded activity recognition system," *IEEE Pervasive Computing*, vol. 7, no. 2, pp. 32–41, 2008.

[14] E. Miluzzo, N. D. Lane, K. Fodor, *et al.*, "Sensing meets mobile social networks: The design, implementation and evaluation of the cenceme application," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*, 2008, pp. 337–350.

[15] T. Brezmes, J.-L. Gorricho, and J. Cotrina, "Activity recognition from accelerometer data on a mobile phone," in *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living: 10th International Work-Conference on Artificial Neural Networks, IWANN 2009 Workshops, Salamanca, Spain, June 10-12, 2009. Proceedings, Part II 10*, Springer, 2009, pp. 796–799.

[16] N. C. Krishnan, C. Juillard, D. Colbry, and S. Panchanathan, "Recognition of hand movements using wearable accelerometers," *Journal of Ambient Intelligence and Smart Environments*, vol. 1, no. 2, pp. 143–155, 2009.

[17] X. Long, B. Yin, and R. M. Aarts, "Single-accelerometer-based daily physical activity classification," in *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, IEEE, 2009, pp. 6107–6110.

[18] J. Yang, "Toward physical activity diary: Motion recognition using simple acceleration features with mobile phones," in *Proceedings of the 1st international workshop on Interactive multimedia for consumer electronics*, 2009, pp. 1–10.

[19] A. Mannini and A. M. Sabatini, "Machine learning methods for classifying human physical activity from on-body accelerometers," *Sensors*, vol. 10, no. 2, pp. 1154–1175, 2010.

[20] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SigKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2011.

[21] G. Weiss, *Wisdm (wireless sensor data mining) project. fordham university, department of computer and info. science*, 2011.

[22] M. Thida, H.-L. Eng, and P. Remagnino, "Laplacian eigenmap with temporal constraints for local abnormality detection in crowded scenes," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 2147–2156, 2013.

[23] Y.-S. Lee and S.-B. Cho, "Activity recognition with android phone using mixture-of-experts co-trained with labeled and unlabeled data," *Neurocomputing*, vol. 126, pp. 106–115, 2014.

[24] H. Ding, L. Shangguan, Z. Yang, *et al.*, "Femo: A platform for free-weight exercise monitoring with rfids," in *Proceedings of the 13th ACM conference on embedded networked sensor systems*, 2015, pp. 141–154.

[25] J. Qi, Z. Wang, X. Lin, and C. Li, "Learning complex spatio-temporal configurations of body joints for online activity recognition," *IEEE Transactions on Human-Machine Systems*, vol. 48, no. 6, pp. 637–647, 2018.

[26] Y. Du, Y. Lim, and Y. Tan, "A novel human activity recognition and prediction in smart home based on interaction," *Sensors*, vol. 19, no. 20, p. 4474, 2019.

[27] C. N. Phyo, T. T. Zin, and P. Tin, "Deep learning for recognizing human activities using motions of skeletal joints," *IEEE Transactions on Consumer Electronics*, vol. 65, no. 2, pp. 243–252, 2019.

[28] K. Wang, J. He, and L. Zhang, "Attention-based convolutional neural network for weakly labeled human activities' recognition with wearable sensors," *IEEE Sensors Journal*, vol. 19, no. 17, pp. 7598–7604, 2019.

[29] G. Grossi, R. Lanzarotti, P. Napoletano, N. Noceti, and F. Odone, "Positive technology for elderly well-being: A review," *Pattern Recognition Letters*, vol. 137, pp. 61–70, 2020.

[30] C. Pham, S. Nguyen-Thai, H. Tran-Quang, *et al.*, "Senscapsnet: Deep neural network for non-obtrusive sensing based human activity recognition," *IEEE Access*, vol. 8, pp. 86 934–86 946, 2020.

[31] V. Sihag, A. Mitharwal, M. Vardhan, and P. Singh, "Opcode n-gram based malware classification in android," in *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, IEEE, 2020, pp. 645–650.

[32] M. Gong, "A novel performance measure for machine learning classification," *International Journal of Managing Information Technology (IJMIT) Vol*, vol. 13, 2021.

[33] D. Kim, H. Han, W. Wang, Y. Kang, H. Lee, and H. S. Kim, "Application of deep learning models and network method for comprehensive air-quality index prediction," *Applied Sciences*, vol. 12, no. 13, p. 6699, 2022.

[34] C. Sweeney, E. Ennis, M. Mulvenna, R. Bond, and S. O'Neill, "How machine learning classification accuracy changes in a happiness dataset with different demographic groups," *Computers*, vol. 11, no. 5, p. 83, 2022.

[35] Y. Zhao, H. Zhou, S. Lu, Y. Liu, X. An, and Q. Liu, "Human activity recognition based on non-contact radar data and improved pca method," *Applied Sciences*, vol. 12, no. 14, p. 7124, 2022.

[36] S. Mekruksavanich and A. Jitpattanakul, "A comparative study of deep learning robustness for sensor-based human activity recognition," in *2023 46th International Conference on Telecommunications and Signal Processing (TSP)*, IEEE, 2023, pp. 87–90.

[37] S. Mekruksavanich and A. Jitpattanakul, "Position-aware human activity recognition with smartphone sensors based on deep learning approaches," in *2023 46th International Conference on Telecommunications and Signal Processing (TSP)*, IEEE, 2023, pp. 43–46.

[38] S. A. F. Oman, M. N. Jamil, and S. T. U. Raju, "Bcl: A branched cnn-lstm architecture for human activity recognition using smartphone sensors," in *2023 International Conference on Next-Generation Computing, IoT and Machine Learning (NCIM)*, IEEE, 2023, pp. 1–5.

[39] A. N. Tarekegn, M. Ullah, F. A. Cheikh, and M. Sajjad, "Enhancing human activity recognition through sensor fusion and hybrid deep learning model," in *2023 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*, IEEE, 2023, pp. 1–5.