MAR ATHANASIUS COLLEGE OF ENGINEERING
(Affiliated to APJ Abdul Kalam Technological University, TVM)
KOTHAMANGALAM

Department of Computer Applications

Main Project Report

HUMAN ACTIVITY RECOGNITION

Done by

Anlin Albert
Reg No: MAC20MCA-2003

Under the guidance of
Prof. Sonia Abraham

2020-2022

MAR ATHANASIUS COLLEGE OF ENGINEERING
(Affiliated to APJ Abdul Kalam Technological University, TVM)
KOTHAMANGALAM

CERTIFICATE

HUMAN ACTIVITY RECOGNITION

Certified that this is the bonafide record of project work done by

Anlin Albert
Reg No: MAC20MCA-2003

During the academic year 2020-2022, in partial fulfillment of requirements for the award of the degree,

Master of Computer Applications
of
APJ Abdul Kalam Technological University
Thiruvananthapuram

Faculty Guide	**Head of Department**
Prof. Sonia Abraham	Prof. Biju Skaria
Project Coordinator	**External Examiner**
Prof. Biju Skaria	

ACKNOWLEDGEMENT

First and foremost, I thank God Almighty for his divine grace and blessings in making all this possible. May he continue to lead me in the years to come.

I am also grateful to Prof. Biju Skaria, Head of Computer Applications Department and Project Coordinator, for his valuable guidance as well as timely advice which helped me a lot during the preparation of the project.

I would like to express my special gratitude and thanks to main project guide Prof. Sonia Abraham, Department of Computer Applications for the guidance and constant supervision as well as for providing necessary information regarding the main project & also for the support.

I profusely thank other Professors in the department and all other staff of MACE, for their guidance and inspiration throughout my course of study. No words can express my humble gratitude to my beloved parents who have been guiding me in all walks of my journey. My thanks and appreciations also go to my friends and people who have willingly helped me out with their abilities.

ABSTRACT

Inertial motion data-based Human Activity Recognition (HAR) has seen significant growth in recent years in both academic and commercial settings. From an abstract standpoint, this has been caused by a speeding up of the development of intelligent and smart surroundings and systems that span every element of human existence, including healthcare, sports, manufacturing, and commerce. Such environments and systems demand and encompass activity recognition, which identifies one or more individuals' behaviours, traits, and objectives from a temporal stream of observations streamed from one or more sensors.

Human Activity Recognition (HAR) is classifying the activity of a person using responsive sensors that are affected by human movement. Both users and capabilities(sensors) of smartphones increase and users usually carry their smartphones with them. These facts make HAR more important and popular.

With the ageing of the population in many nations today, older people are increasingly likely to live alone and are frequently unable to receive assistance from family members. Elderly people are known to be vulnerable to falls and mishaps when engaging in daily activities. Through the Internet of Things (IoT), smart home technology has been designed to recognise the daily actions of elders, assisting lone elderly to live safely and pleasantly.

Since smartphones and their privacy are highly safeguarded due to the development of pervasive computer and sensor automation, sensor-based HAR is being utilised in smart devices more frequently. The focus of this project is therefore smartphone sensor-based HAR. This work focuses on the recognition of human activity using smartphone sensors using different machine learning classification approaches. Data retrieved from smartphones' accelerometer sensor which is classified to recognize human activity. Results of the approaches used are compared in terms of efficiency and precision.

List of Tables

List of Figures

Contents

1. Introduction

Human activity recognition (HAR) is an ability to interpret human body gestures or motion via sensors and determine human activity or action. It is based on an inertial measurement unit (IMU) that has become the de facto method for continuously monitoring not only what human beings are up to but also in monitoring the activities of devices, machine parts, pets, and others. This has made HAR based on IMU sensors a hot area for research. Not to mention that these maintain high levels of privacy and comfort for the user. To understand human behavior and intrinsically anticipate human intentions, research into human activity recognition HAR) using sensors in wearable and handheld devices has intensified. The ability of a system to use as few resources as possible to recognize a user's activity from raw data is what many researchers are striving for attention.

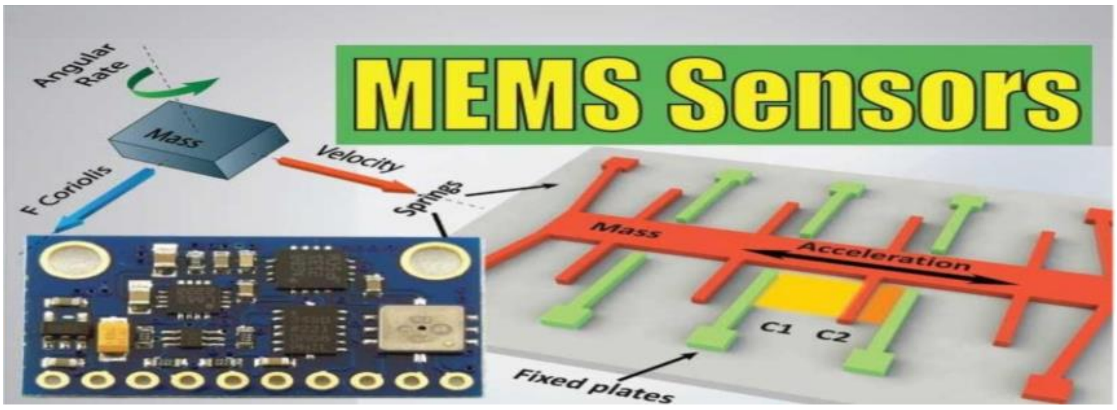
Human activity analysis is one of the most important problems that has received considerable attention from the computer vision community in recent years. It has various applications, spanning from activity understanding for intelligent surveillance systems to improving human-computer interactions. Recent approaches have demonstrated great performance in recognizing individual actions. However, in reality, human activity can involve multiple people, and to recognize such group activities and their interactions would require information more than the motion of individuals

Most human daily tasks can be simplified or automated if they can be recognized via the HAR system. Typically, the HAR system can be either supervised or unsupervised. A supervised HAR system required some prior training with dedicated datasets while an unsupervised HAR system is configured with a set of rules during development. HAR is considered an important component in various scientific research contexts i.e. surveillance, healthcare, and human-computer interaction (HCI).

Accelerometer

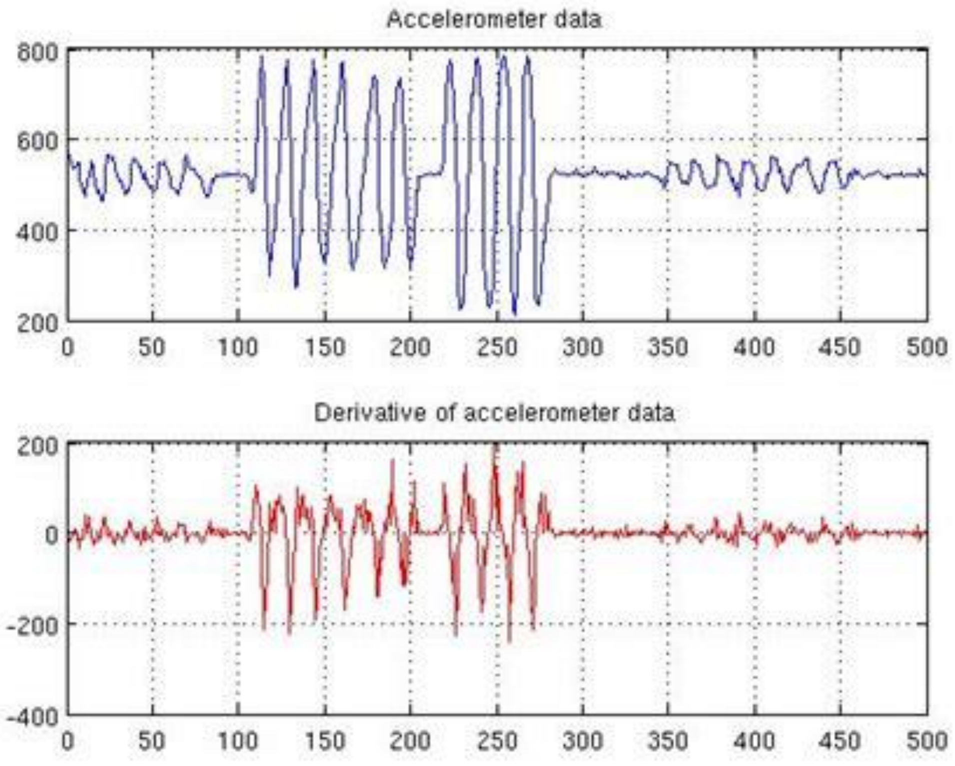
The accelerometer is an instrument that measures the experienced physical acceleration of an object. It has been employed for several applications in science, medicine, engineering, and industry such as for measuring vibrations in machinery, acceleration in high-speed vehicles, and moving loads on bridges.

Figure 1: Accelerometer



Its principle of operation generally consists of a seismic mass that is displaced by the acceleration it is subjected to. This displacement can then be transduced into a measurable electrical signal. This phenomenon has been applied to the development of microelectromechanical systems (MEMS) sensors. Their technology allows the creation of nano-scale devices fabricated with semiconductors. They are advantageous against other sensor technologies because it is possible to produce them on large scale and with low manufacturing costs. Most common MEMS accelerometers work as a captivating sensor composed of a cantilever beam with a proof mass whose deflection is correlated with the acceleration experienced by the sensor.

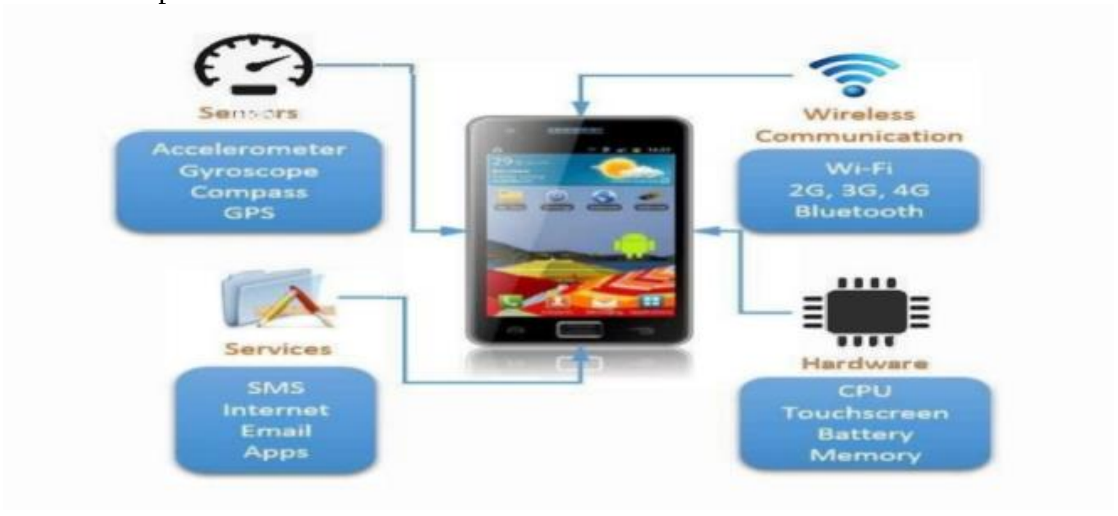
Figure 2: Accelerometer data plot



Smartphones as Wearable sensors

Wearable technologies comprise all the devices that are body-worn and allow to gather and process information from all the users and their interaction with the environment. In this project, we use smartphones as a wearable devices given that they are now provided with numerous internal sensors, some of which can be exploited for motion sensing and are thus appropriate for the identification of human activities.

Figure 3: Smartphones and their features



We have selected one of them: an accelerometer. This provides information about the user's linear acceleration and angular velocity when used as a wearable sensor and is not highly affected by external factors such as bad indoor signal reception in GPS or electromagnetic noise in compass. However, accelerometer measurements are always influenced by the gravity component in the detection of the body motion acceleration. Similarly, we work with acceleration and angular velocity signals which are directly read from these sensors and avoid their integration to obtain position or orientation information given the known drift due to noise found in this type of inertial sensor.

2. Supporting Literature

2.1. Literature Review

[1] Bulbul, E., Cetin, A., & Dogru, I. A. (2018). Human Activity Recognition Using Smartphones. 2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT).

Introduction - Smartphones are the most useful tools in our daily life and with the advancing technology they get more capable day by day to meet customer needs and expectations. An accelerometer has been standard hardware for almost all smartphone manufacturers. Since there is a meaningful difference in characteristics between data retrieved from these sensors, many features could be generated from these sensors' data to determine the activity of the person that is carrying the device. In this study, a dataset consisting of signals from the accelerometer and gyroscope of a smartphone carried by different men and women volunteers while doing different activities are classified using different machine learning approaches.

Dataset - Dataset consists of signals from a smartphone carried by 9 individuals performing 6 different activities. Activities performed are listed below with their corresponding codes.

- WALKING
- CLIMBING UP THE STAIRS
- CLIMBING DOWN THE STAIRS
- SITTING
- STANDING
- LAYING

Signals are recorded with a sampling rate of 50Hz and stored as time series for each dimension so 6 different signals were obtained (3 are from the accelerometer and the other 3 are from the gyroscope). The noise was filtered using median and 20Hz Butterworth filters to get more precise results. A second 3hz Butterworth filtering was applied to eliminate the effect of gravity in accelerometer signals. Values then normalized to (-1,1) interval. Euclid magnitudes of the values of 3 dimensions were calculated to merge 3-dimensional signals into one dataset. Finally, class codes (activity codes) given above for each row are added at the end of them among with the number that is given to each individual. In the end, the dataset consists of 2947 records with 561 features.

Proposed method - Supervised machine learning is used to recognize activity from dataset records. Different supervised machine learning models are designed using different classification approaches. Designed models first trained with training data that consists of 80% of the total dataset and then tested with the rest. Classification precision of models is tested and observed using 5-fold cross-validation.

Methods used for classification are as follows:

- Decision Trees
- Support Vector Machines
- K-nearest neighbors (KNN)
- Ensemble classification methods
- Boosting
- Bagging

[2] Wang, H., Zhao, J., Li, J., Tian, L., Tu, P., Cao, T., … Li, S. (2020). Wearable Sensor-Based Human Activity Recognition Using Hybrid Deep Learning Techniques. Security and Communication Networks, 2020, 1–12.

Introduction - Human behavior recognition (HAR) is the detection, interpretation, and recognition of human behaviors, which can use smart health care to actively assist users according to their needs. Human behavior recognition has wide application prospects, such as monitoring in smart homes, sports, game controls, health care, elderly patients care, bad habits detection, and identification. It plays a significant role in in-depth study and can make our daily life smarter, safer, and more convenient. This work proposes a deep learning-based scheme that can recognize both specific activities and the transitions between two different activities of short duration and low frequency for health care applications.

Dataset - This paper adopts the international standard Data Set, Smartphone-Based Recognition of Human Activities, and Postural Transitions Data Set to conduct an experiment, which is abbreviated as HAPT Data Set. The data set is an updated version of the UCI Human Activity Recognition Using popularity Data set. It provides raw data from smartphone sensors rather than preprocessed data and collects data from accelerometer and gyroscope sensors. In addition, the action category has been expanded to include transition actions. The HAPT data set contains twelve types of actions. A total of 815,614 valid pieces of data are used here.

Proposed method - The overall architecture diagram of the method proposed in this paper contains three parts. The first part is the preprocessing and transformation of the original data, which combines the original data such as acceleration and gyroscope into an image-like two-dimensional array. The second part is to input the composite image into a three-layer CNN network that can automatically extract the motion features from the activity image and abstract the features, then map them into the feature map. The third part is to input the feature vector into the LSTM model, establish a relationship between time and action sequence, and finally introduce the full connection layer to achieve the fusion of multiple features. In addition, Batch Normalization (BN) is introduced, in which BN can normalize the data in each layer and finally send it to the Softmax layer for action classification.

[3] Agarwal, P., & Alam, M. (2020). A Lightweight Deep Learning Model for Human Activity Recognition on Edge Devices. Procedia Computer Science, 167, 2364–2373.

Introduction - Here the architecture for the proposed Lightweight model is developed using Shallow Recurrent Neural Network (RNN) combined with Long Short-Term Memory (LSTM) deep learning algorithm. then the model is trained and tested for six HAR activities on resource-constrained edge devices like RaspberryPi3, using optimized parameters. The experiment is conducted to evaluate the efficiency of the proposed model on the WISDM dataset containing sensor data of 29 participants performing six daily activities: Jogging, Walking, Standing, Sitting, Upstairs, and Downstairs. And lastly, the performance of the model is measured in terms of accuracy, precision, recall, f-measure, and confusion matrix and is compared with certain previously developed models.

Dataset - Here Android smartphone having an inbuilt accelerometer is used to capture tri-axial data. The dataset consists of six activities performed by 29 subjects. These activities include walking, upstairs, downstairs, jogging, standing, and sitting. Each subject performed different activities by carrying a cell phone in the front leg pocket. A constant Sampling rate of 20 Hz was set for the accelerometer sensor. A detailed description of the dataset is given in table 1 below.

- Total no of samples: 1,098,207
- Total no of subjects: 29

• Activity	Samples:	Percentage
• Walking	4,24,400	38.6%
• Jogging	3,42,177	31.2%
• Upstairs	1,22,869	11.2%
• Downstairs	1,00,427	9.1%
• Sitting	59,939	5.5%
• Standing	48,397	4.4%

Proposed method - The working of the Lightweight RNN-LSTM-based HAR system for edge devices. The accelerometer reading is partitioned into fixed window size T. The input to the model is a set of readings (x1, x2, x3,…….,xT-1, xT) captured in time T, where xt is the reading captured at any time instance t. These segmented window readings are then fed to the Lightweight RNN-LSTM model. The model uses the sum of rule and combine output from different states using a softmax classifier to one final output of that particular window.

[4] Das, S., Partha, S. B., & Imtiaz Hasan, K. N. (2020). Sentence Generation using LSTM Based Deep Learning. 2020 IEEE Region 10 Symposium (TENSYMP).
Using the Long Short-Term Memory (LSTM) architecture creates a phrase creation system here. The fundamentals of word embedding are generally followed by the system, where words from the dataset are tokenized and transformed into vector shapes.
Following processing, a layer of long short-term memory is used to store these vectors. After each repetition, the system generates a new set of words. As a result of this process, a

sentence or passage will eventually be formed using pertinent words. In comparison to other existing approaches, the system's results are fairly compelling.

Conclusion: This study presents the idea of developing a deep learning-based model that can produce novel sentences. Word embedding and the Long Short-Term Memory (LSTM) architecture, a modified version of recurrent neural networks, serve as the foundation for all of the methodologies (RNN). The text data are transformed into vector form using word embedding since the neural network finds it easier to operate with numerical data. LSTM and Bidirectional LSTM networks are used to preserve the correct context. Because it can work on both the present and the past at once, the bidirectional LSTM is a crucial layer for the model. The model was effectively trained, and the outcomes were carefully examined and contrasted with those from other models.

[5] Ullah, M., Ullah, H., Khan, S. D., & Cheikh, F. A. (2019). Stacked Lstm Network for Human Activity Recognition Using Smartphone Data. 2019 8th European Workshop on Visual Information Processing (EUVIP).
Proposed method – The method mainly consists of two parts i.e. a single layer neural network and a network of stacked LSTM cells. Initially, sensor data is obtained from the smartphone that is worn by a human subject. Here two types of sensor data i.e. accelerometer and the gyroscope. The raw sensor data is passed through a single-layer neural network which acts as a pre-processing and normalized input data for the succeeding network. The normalization is achieved through a linear discriminant function and ReLU activation. After that, the data is fed to the stacked LSTM network. The network consists of five LSTM cells that have learned the temporal dependencies of the sensor sequential data. The output of the stacked LSTM network is given to a six-way softmax which gives the individual probability of the six human behavior i.e. (walking, walking upstairs, walking downstairs, sitting, standing, lying).

[6] Sun, B., Liu, M., Zheng, R., & Zhang, S. (2019). Attention-based LSTM Network for Wearable Human Activity Recognition. 2019 Chinese Control Conference (CCC).
In this paper, an LSTM network with an attention mechanism, which can automatically focus on the time series that has a decisive effect on classification, to capture the most important temporal dependencies from the input, without using extra handcrafted features and human domain knowledge. The attention mechanism allows a model to learn a set of weights over raw sensor data, which is used to leverage the weight of the temporal context.
Proposed method - The model contains four components - Input layer: input sensor data to this model, LSTM layer: utilize LSTM to get high-level features, Attention layer: produce a weight vector, and merge features from each time step into a temporal feature vector to find relevant temporal context by multiplying the weight vector and Output layer: the temporal feature is finally used for activity recognition.

[7] Alemayoh, T. T., Hoon Lee, J., & Okamoto, S. (2019). Deep Learning-Based Real-time Daily Human Activity Recognition and Its Implementation in a Smartphone. 2019 16th International Conference on Ubiquitous Robots (UR).

Introduction - Human activity recognition is a broad area of study mainly concerned with identifying specific movement or action of a person based on given input data. Mostly, input data signals are obtained from videos, where video frames are taken for analysis or multi-axis time-series IMU devices. Comparably, wearable IMU sensors became a popular and convenient way of data collection mechanisms without an extensive installation of equipment and privacy issues.
The processed version of the data was used to fit statistical and machine learning models such as SVM as in Anguita et al. Deep learning methods have shown the capability and even achieve state-of-the-art results by automatically learning high-level and meaningful features from raw data. In large-scale data classification, CNN is competent to extract features from signals and it has demonstrated excellent performance in image classification, speech recognition, and sentence classification.

Dataset – The smartphone used was attached tightly to the waist of the subjects. Out of the various motion-related sensors of a smartphone, the 3-axis Acc and 3-axis Gyro were chosen for a better result. Motion data of eight activities were collected. The activities are: walking, jumping, running, bicycle riding, stairs ascending, and descending, laying down, and still.

Proposed method - A CNN is applied to the activities' one-dimensional virtual images prepared. Each convolutional layer performs a 2D convolution on its inputs followed by a non-linear activation function, ReLU (Rectifier Linear Unit). To reduce the effect of internal covariance shift of activations, batch normalization was utilized, which forces each mini-batch input of a layer to have similar distribution throughout the hidden layers. Besides, it allows the use of larger learning rates to speed up the optimization process. After the output of the second pooling is flattened to form a long 1D feature map vector, the classification is decided by the probability distribution of an eight-class softmax layer. All the parameters of the network are updated by Adam optimizer using back optimization.

[8] Deep, S., & Zheng, X. (2019). Hybrid Model Featuring CNN and LSTM Architecture for Human Activity Recognition on Smartphone Sensor Data. 2019 20th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT).

Introduction - The proliferation of smartphones with various embedded sensors have eased the method of gathering human activity data in recent time. With the development of unprecedented characteristics of sensors such as accelerometers and gyroscopes, sensor-based human activity recognition has received extensive concerns. In wearable-based HAR, sensors or other external devices are attached to the human body. HAR is a method of predicting activities from the data obtained from sensors. The process involves extracting motion features and classifying the activities into different categories. The data collected from the sensors are a sequence of time series data and traditional machine learning algorithms may not exploit the temporal correlations of input data.

In this paper, a combination of CNN and LSTM for HAR is used. Furthermore, it is also possible to apply LSTM for activity recognition tasks in the same dataset and compare the results with the CNN-LSTM model.

Dataset - To evaluate the effectiveness of the CNN-LSTM model, experiment on the UCI HAR dataset. The dataset consists of time series data collected from 30 volunteers of the 19-48 age group. Each volunteer performed six activities (walking, walking upstairs, walking downstairs, sitting, standing, laying) with a smartphone attached to their waist. The 3-axial linear acceleration (tAcc-XYZ) from the accelerometer and 3-axial angular velocity (tGyro-XYZ) from gyroscope data were collected. The data were collected with a constant sampling rate of 50Hz. The activities were video-recorded for ground truth and data were manually labeled. The dataset is randomly divided into 70% training and 30%
testing data.

Proposed method – Here, a kernel size of 6 and several filters 128 for both the convolutional layers. The output data size is then passed through the dropout layer. This output data is further used to pass through the LSTM layer. The output 3D data is fed to the LSTM layer.

The LSTM layer is used in this hybrid architecture because it works well with the time series data and is designed to handle time dependence problems. Each LSTM layer in this architecture produces hidden cell information. The LSTM layer is followed by a dense layer, hyperbolic tangent activation, and a soft-max layer at the end. It is then passed through a dense layer with hyperbolic tangent activation and used Adam optimizer which ends the LSTM networks in this hybrid model.

2.2. Findings and Proposals

The study has shown that this recurrent system is capable of handling a wide range of issues, including sentiment analysis, computer vision, time series forecasting, text recognition, natural language processing, picture and video captioning, and text recognition. It was discovered that combining LSTM with other architectures helps to achieve the best performance is a typical strategy when modeling the majority of these issues.

Convolution and pooling layers were utilized in such hybrid models to drastically eliminate representational redundancy while reducing the problem's dimensionality. Additional architecture customization might always be used to increase precision.

Based on the study, the learning rate is the most significant hyperparameter in the backpropagation algorithm, while the forget gate and output transfer function are the most crucial parts of the LSTM block. Therefore, additional research into these elements may result in LSTM variants with enhanced prediction skills. Another equally important study area discusses less computationally intensive learning techniques to modify the parameters that can be learned.

3. System Analysis

3.1. Analysis of Dataset

3.1.1. About the Dataset

Table 1: Dataset source

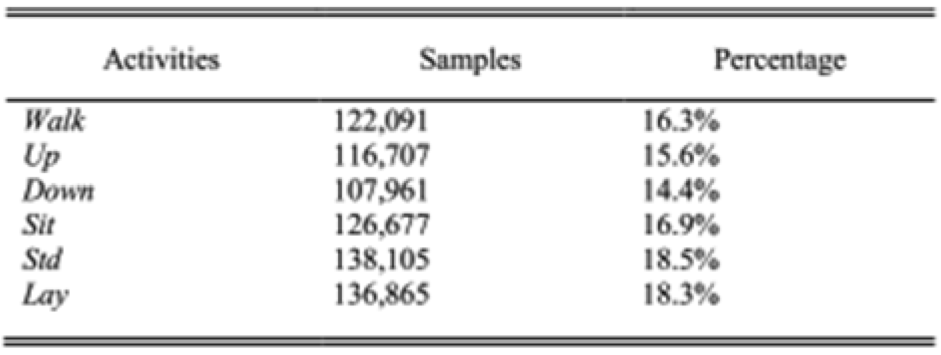
Name	Source
UCI-HAR	https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones
WISDM	https://www.cis.fordham.edu/wisdm/dataset.php

3.1.2. Explore the Dataset

a. UCI-HAR

The UCI-HAR dataset was built from the recordings of 30 subjects aged 19-48 years. During the recording, all subjects were instructed to follow an activity protocol. And they wore a smartphone (Samsung Galaxy S II) with embedded inertial sensors around their waist. The six activities of daily living are standing (Std), laying (Lay), walking (Walk), walking downstairs (Down), and walking upstairs (Up). In addition, this dataset also includes postural transitions that occur between the static postures: standing to sitting, sitting to standing, sitting to laying, laying to sitting, standing to laying and laying to standing. Specifically, in this paper, only six basic activities were selected as input samples due to the percentage of postural transitions being small. The experiments had been video-recorded to manually label the data. Finally, the researchers captured 3-axial acceleration and 3-axial angular velocity data at a constant rate of 50Hz. According to statistics, the number of samples in this dataset is 748406, and the detailed information is shown below.

Figure 4: UCI-HAR

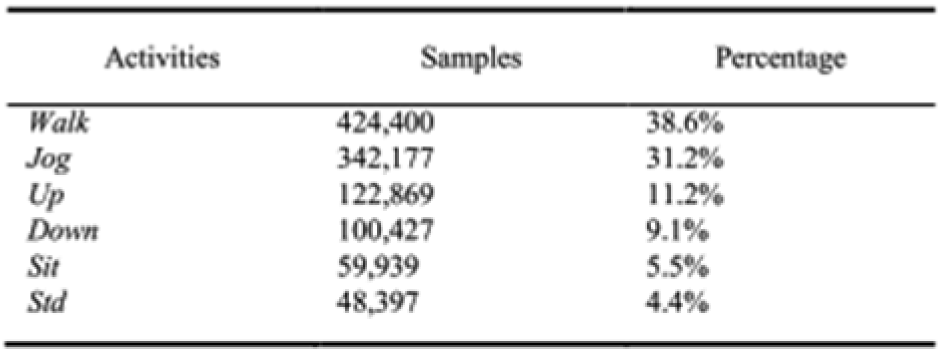


Activities	Samples	Percentage
Walk	122,091	16.3%
Up	116,707	15.6%
Down	107,961	14.4%
Sit	126,677	16.9%
Std	138,105	18.5%
Lay	136,865	18.3%

b. WISDM

The WISDM dataset has a total of 1098209 samples, and the percentage of the total samples associated with each activity was shown in Table 3. It can be seen that WISDM is an unbalanced dataset. Activity walking takes up the most, reaching 38.6% while standing only accounts for 4.4%. Its experimental object consists of 36 subjects. These subjects performed certain daily activities with an Android phone in their front leg pockets. The sensor used is an accelerometer with a sampling frequency of 20 Hz. It is also a built-in motion sensor of the smartphone. Six activities were recorded: standing (Std), sitting (Sit), walking (Walk), upstairs (Up), downstairs (Down), and jogging (Jog). The data collection was supervised by a dedicated person to ensure the quality of data.

Figure 5: WISDM



Activities	Samples	Percentage
Walk	424,400	38.6%
Jog	342,177	31.2%
Up	122,869	11.2%
Down	100,427	9.1%
Sit	59,939	5.5%
Std	48,397	4.4%

3.2. Data Pre-Processing

3.2.1. Data Cleaning

Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabelled. If data is incorrect, outcomes and algorithms are unreliable, even though they may look correct. There is no one absolute way to prescribe the exact steps in the data cleaning process because the processes will vary from dataset to dataset. But it is crucial to establish a template for your data cleaning process so you know you are doing it the right way every time.

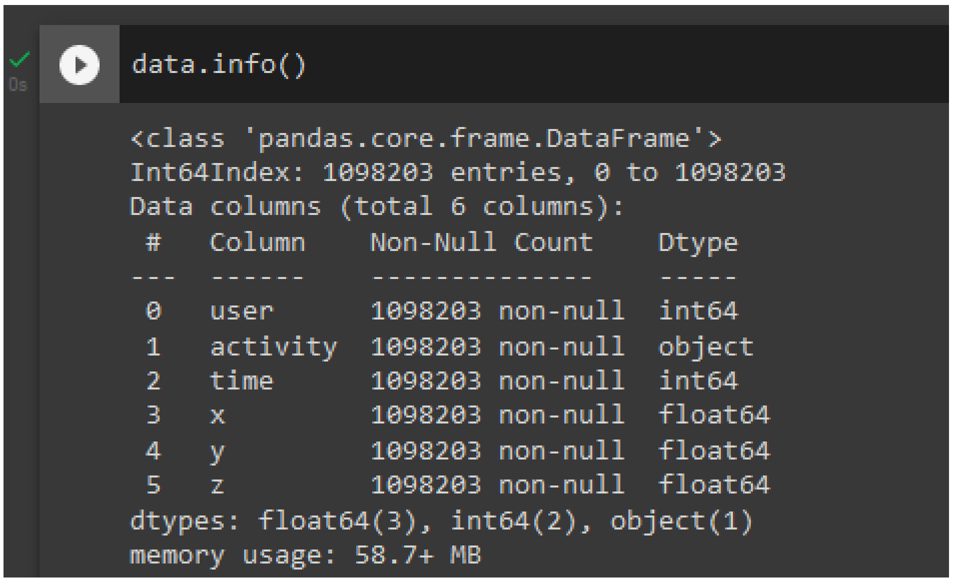
In this project, two datasets are used. The datasets contain categorical data and may also contain missing values.

Encoding data: The dataset contains categorical data in the attribute's 'activity'. This column contains the six activities: walking, walking upstairs, walking downstairs, sitting, jogging and standing. We have changed these values to numeric data using pandas 'get_dummies'. Which was further converted to an array using 'asarray'.

Missing values: Missing values are handled using the method ‘dropna’. This will drop every row that contains missing values.

3.2.2. Analysis of Feature Variable

Figure 6: Feature Variables



```
data.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1098203 entries, 0 to 1098203
Data columns (total 6 columns):
 #   Column    Non-Null Count    Dtype
---  ------    --------------    -----
 0   user      1098203 non-null  int64
 1   activity  1098203 non-null  object
 2   time      1098203 non-null  int64
 3   x         1098203 non-null  float64
 4   y         1098203 non-null  float64
 5   z         1098203 non-null  float64
dtypes: float64(3), int64(2), object(1)
memory usage: 58.7+ MB
```

There are six feature variables: user, activity, time, x, y, and z. Out of these five variables we have only taken three variables: x, y, and z which contain the accelerometer values.

3.2.3. Analysis of Class Variable

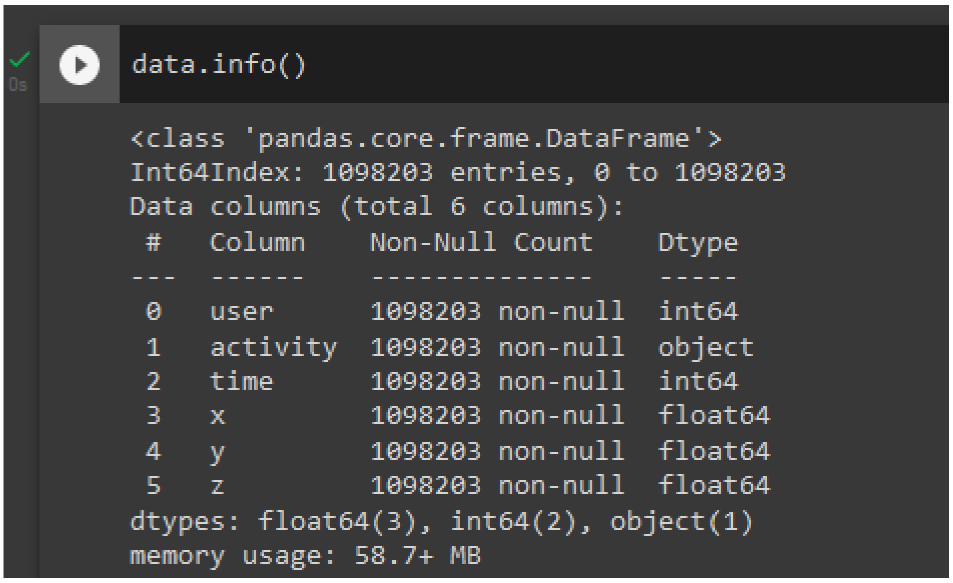
The class variables mainly consist of six variables:

- Walking
- Walking upstairs
- Walking downstairs
- Sitting
- Standing
- Jogging

3.3. Data Visualizations

This is the summary of the dataset used in this project. This dataset contains 6 activities having nearly 11 lakhs records.

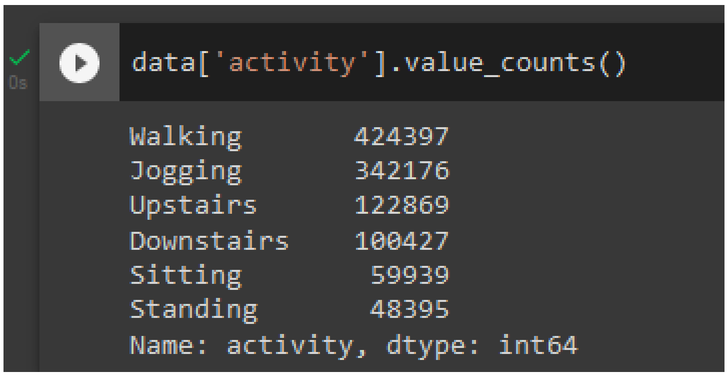
Figure 7: Feature variables



```
data.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1098203 entries, 0 to 1098203
Data columns (total 6 columns):
 #   Column    Non-Null Count    Dtype  
---  ------    --------------    -----  
 0   user      1098203 non-null  int64  
 1   activity  1098203 non-null  object 
 2   time      1098203 non-null  int64  
 3   x         1098203 non-null  float64
 4   y         1098203 non-null  float64
 5   z         1098203 non-null  float64
dtypes: float64(3), int64(2), object(1)
memory usage: 58.7+ MB
```

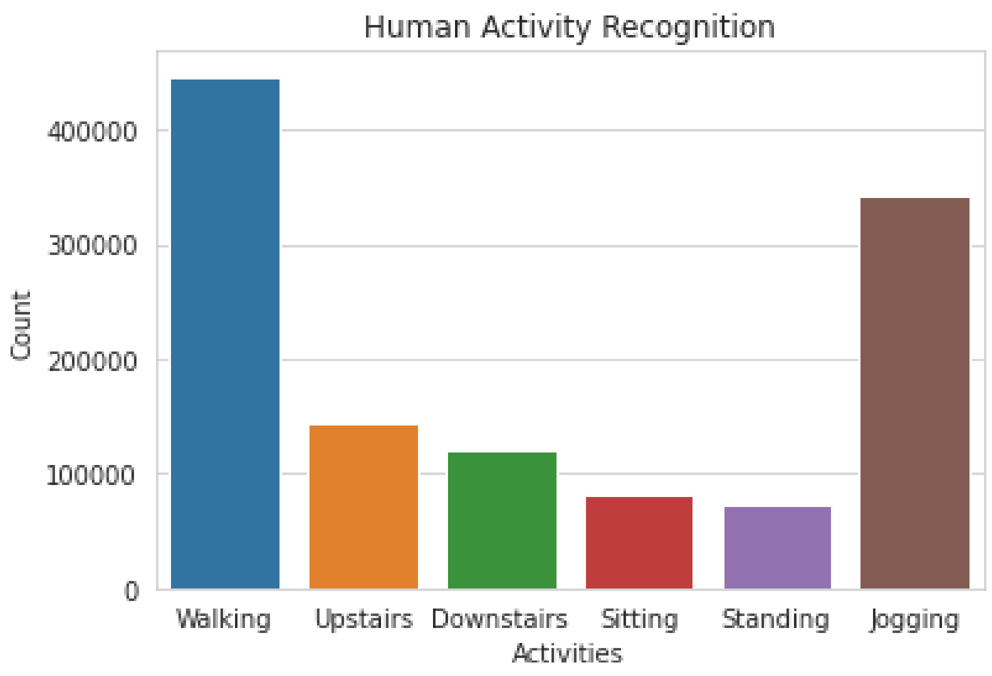
From the below snapshot, we can see the entire contents of each item in the dataset. Means here walking and jogging have more no of records i.e. 424397 and 342176 records respectively while standing has 48395 records only.

Figure 8: Activities



```
data['activity'].value_counts()

Walking       424397
Jogging       342176
Upstairs      122869
Downstairs    100427
Sitting        59939
Standing       48395
Name: activity, dtype: int64
```

Figure 9: Data items



The image shown below represents the accelerometer values for timestamp 200sec so that we can see how the accelerometer data looks visually for each activity. Because each activity follows a specific pattern and by looking at these patterns, we can classify which accelerometer values belong to which class.

Figure 10: Signal (Walking)

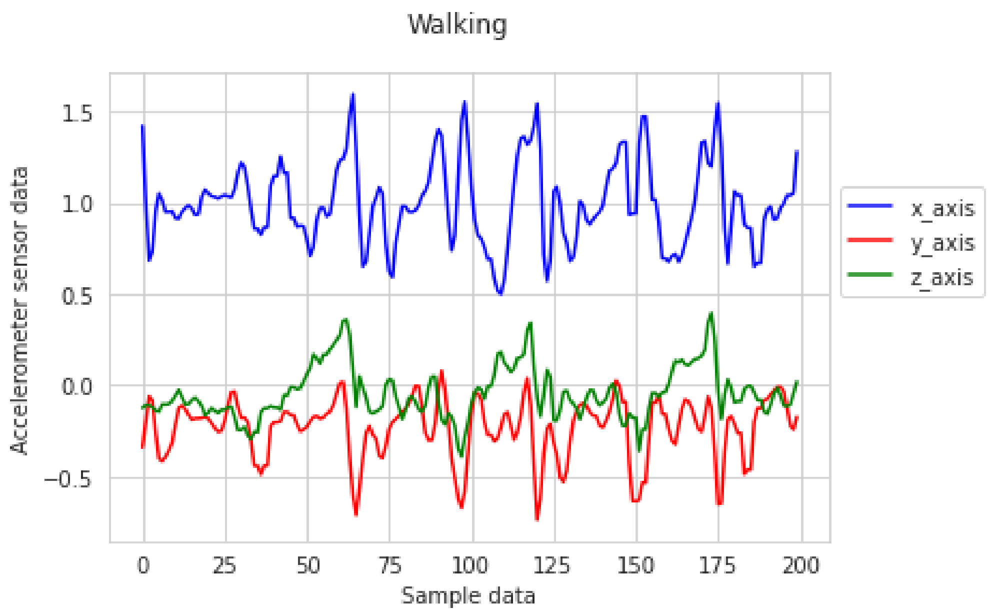


Figure 11: Signal (Jogging)

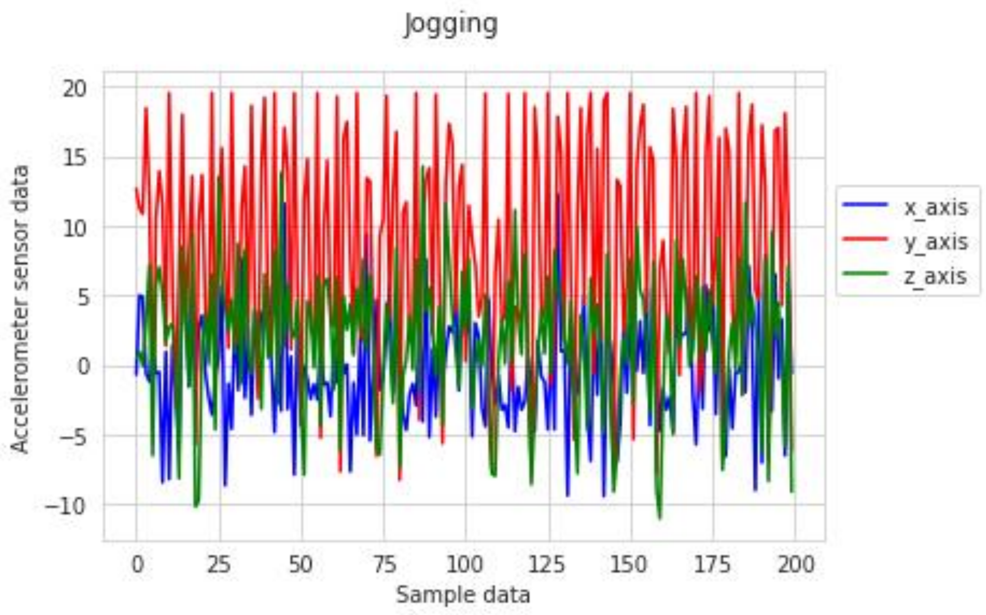


Figure 12: Signal (Sitting)

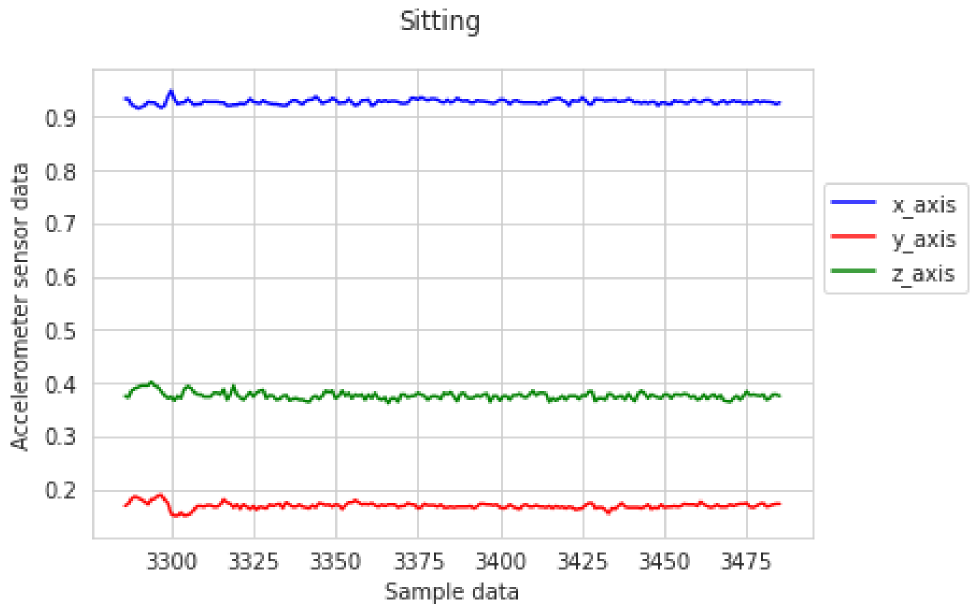


Figure 13: Signal (Standing)

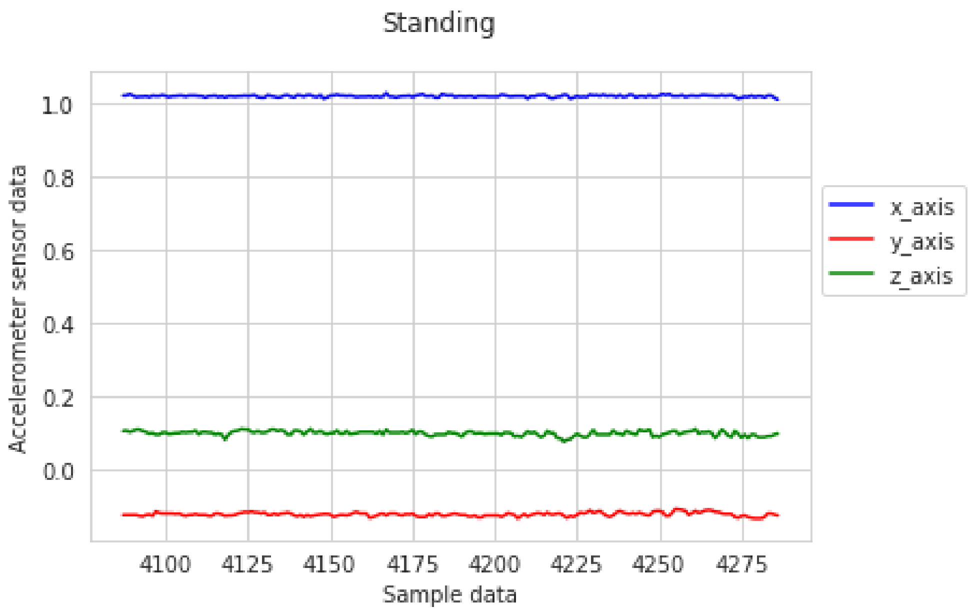


Figure 14: Signal (Upstairs)

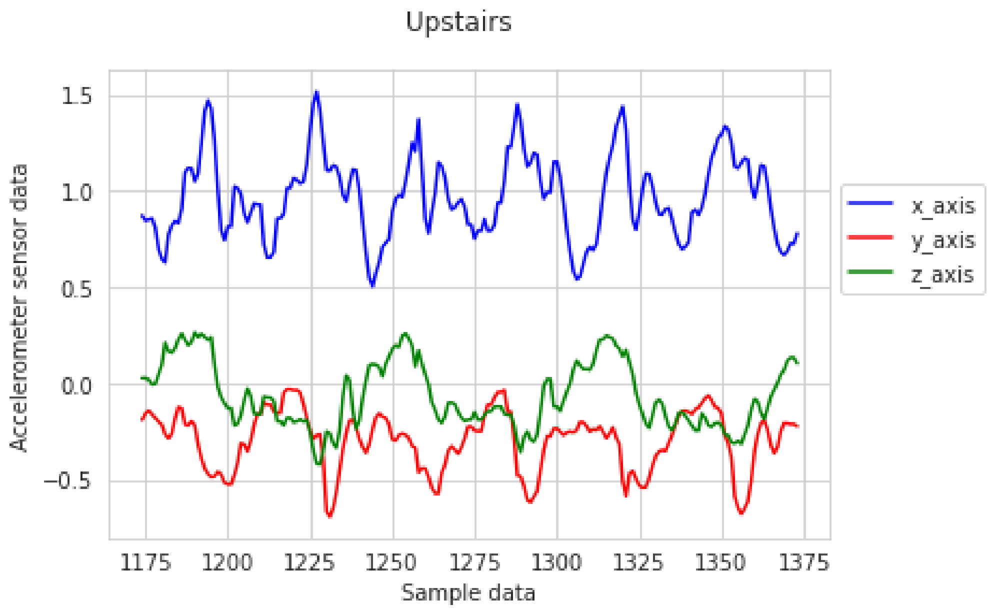
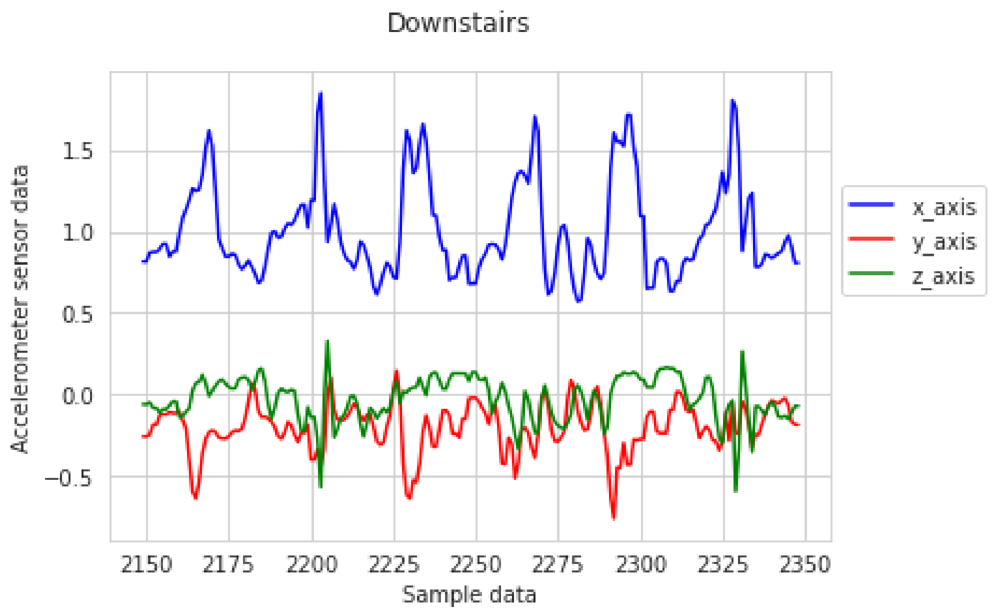


Figure 15: Signal (Downstairs)

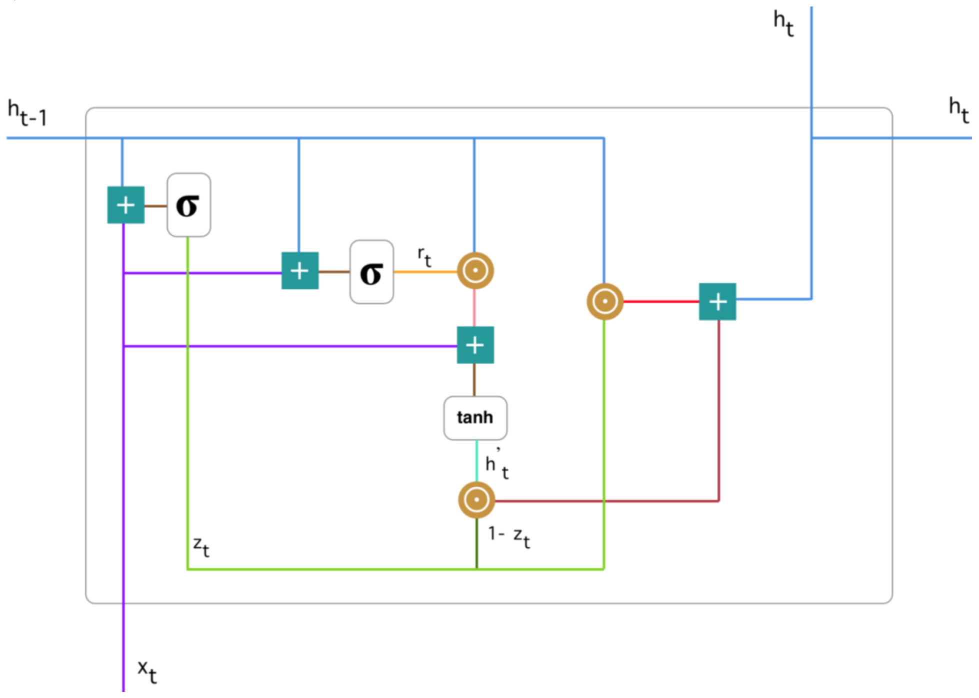


3.4. Analysis of Architecture

3.4.1. Detailed Study of Architecture

Gated Recurrent Unit (GRU)
GRU is a simplified version of the LSTM (Long Short-Term Memory) recurrent neural network model. GRU uses only one state vector and two gate vectors, reset gate and update gate.

Figure 16: GRU



At each timestamp t, it takes an input Xt and the hidden state Ht-1 from the previous timestamp t-1. Later it outputs a new hidden state Ht which is again passed to the next timestamp.

Now there are primarily two gates in a GRU as opposed to three gates in an LSTM cell. The first gate is the Reset gate and the other one is the update gate.

- Reset gate
  The Reset Gate is responsible for the short-term memory of the network i.e the hidden tate (Ht). Here is the equation of the Reset gate.
  $$r_t = \sigma\left(W^{(r)}x_t + U^{(r)}h_{t-1}\right)$$

- Update gate
  Similarly, we have an Update gate for long-term memory and the equation of the gate is shown below.
  $$z_t = \sigma\left(W^{(z)}x_t + U^{(z)}h_{t-1}\right)$$

To find the Hidden state Ht in GRU, it follows a two-step process. The first step is to generate what is known as the candidate hidden state. As shown below:

- Candidate hidden state
  It takes in the input and the hidden state from the previous timestamp t-1 which is multiplied by the reset gate output rt. Later passed this entire information to the tanh function, the resultant value is the candidate's hidden state.

  $$\acute{h}_{\mathrm{t}} = \tanh(Wx_t + r_t \odot \ Uh_{t-1})$$

  The most important part of this equation is how we are using the value of the reset gate to control how much influence the previous hidden state can have on the candidate state.

  If the value of rt is equal to 1 then it means the entire information from the previous hidden state Ht-1 is being considered. Likewise, if the value of rt is 0 then that means the information from the previous hidden state is completely ignored.

- Hidden state
  Once we have the candidate state, it is used to generate the current hidden state Ht. It is where the Update gate comes into the picture. Now, this is a very interesting equation, instead of using a separate gate like in LSTM in GRU we use a single update gate to control both the historical information which is Ht-1 as well as the new information which comes from the candidate state.

  $$h_{\mathrm{t}} = z_{\mathrm{t}} \odot h_{t-1} + (1 - \ z_t) \odot \ \acute{h}_{\mathrm{t}}$$

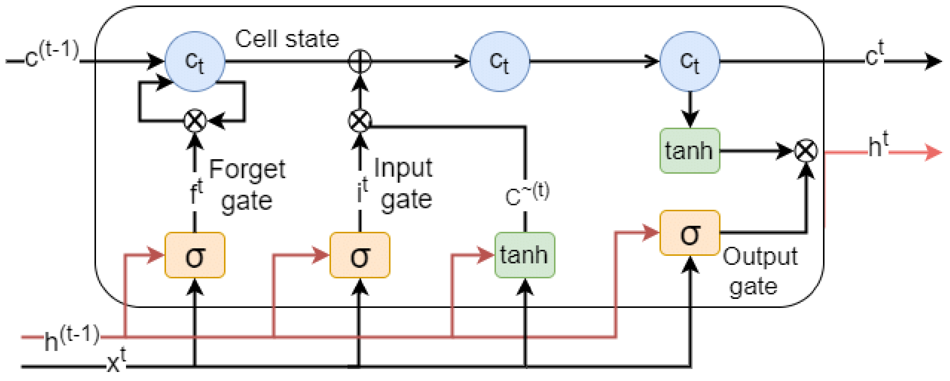
  Now assume the value of ut is around 0 then the first term in the equation will vanish which means the new hidden state will not have much information from the previous hidden state. On the other hand, the second part becomes almost one which essentially means the hidden state at the current timestamp will consist of the information from the candidate state only.

  Similarly, if the value of ut is on the second term will become entirely 0 and the current hidden state will entirely depend on the first term i.e the information from the hidden state at the previous timestamp t-1. Hence, we can conclude that the value of ut is very critical in this equation and it can range from 0 to 1.

Long-Short Term Memory (LSTM)

Long Short-Term Memory is a kind of recurrent neural network. In RNN output from the last step is fed as input in the current step. LSTM was designed by Hochreiter & Schmidhuber. It tackled the problem of long-term dependencies of RNN in which the RNN cannot predict the word stored in the long-term memory but can give more accurate predictions from the recent information. As the gap length increases RNN does not give an efficient performance. LSTM can by default retain the information for a long time. It is used for processing, predicting, and classifying based on time-series data.

Figure 17: LSTM



Forget Gate: The information that is no longer useful in the cell state is removed with the forget gate. Two inputs x_t (input at the particular time) and h_t-1 (previous cell output) are fed to the gate and multiplied with weight matrices followed by the addition of bias. The resultant is passed through an activation function which gives a binary output. If for a particular cell state the output is 0, the piece of information is forgotten and for output 1, the information is retained for future use.

Input gate: The addition of useful information to the cell state is done by the input gate. First, the information is regulated using the sigmoid function and filters the values to be remembered similar to the forget gate using inputs h_t-1 and x_t. Then, a vector is created using the tanh function that gives an output from -1 to +1, which contains all the possible values from h_t-1 and x_t. At last, the values of the vector and the regulated values are multiplied to obtain useful information.

Output gate: The task of extracting useful information from the current cell state to be presented as output is done by the output gate. First, a vector is generated by applying the tanh function to the cell. Then, the information is regulated using the sigmoid function and filtered by the values to be remembered using inputs h_t-1 and x_t. At last, the values of the vector and the regulated values are multiplied to be sent as an output and input to the next cell.

Sigmoid

Gates contains sigmoid activations. A sigmoid activation is similar to the tanh activation. Instead of squishing values between -1 and 1, it squishes values between 0 and 1. That is helpful to update or forget data because any number getting multiplied by 0 is 0, causing values to disappears or be “forgotten.” Any number multiplied by 1 is the same value therefore that value stay’s the same or is “kept.” The network can learn which data is not important therefore can be forgotten or which data is important to keep.

$$S(x) = \frac{1}{1 + e^{-x}}$$

$$S(x) = sigmoid\, function$$

$$e^{-x} = Euler's\, number$$

Tanh

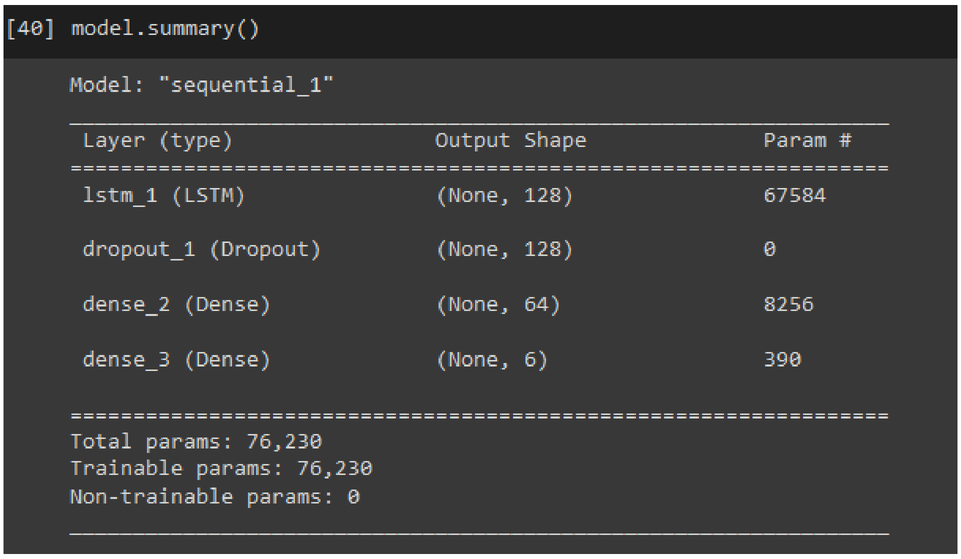
The tanh activation is used to help regulate the values flowing through the network. The tanh function squishes values to always be between -1 and 1. When vectors are flowing through a neural network, it undergoes many transformations due to various math operations. So imagine a value that continues to be multiplied by let’s say 3. You can see how some values can explode and become astronomical, causing other values to seem insignificant.

A tanh function ensures that the values stay between -1 and 1, thus regulating the output of the neural network. You can see how the same values from above remain between the boundaries allowed by the tanh function.

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

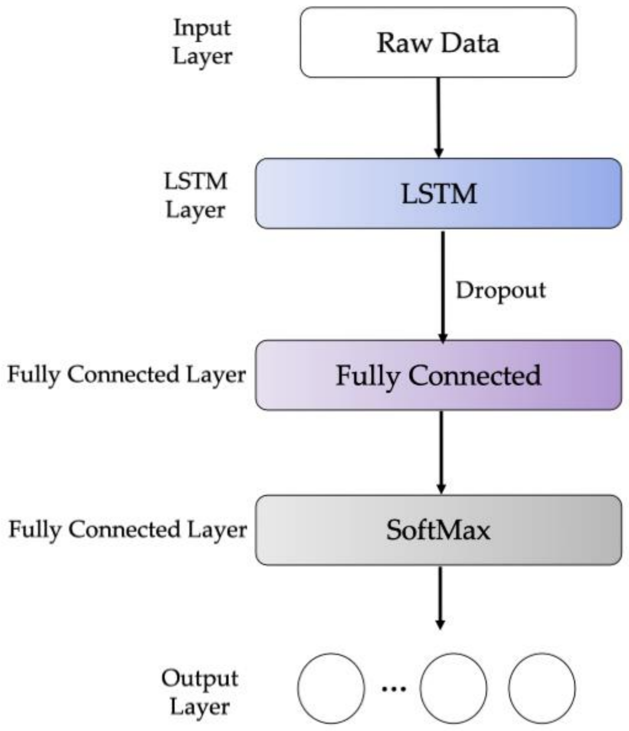
Dimension Table of LSTM

Figure 18: LSTM dimension table



```
[40] model.summary()

     Model: "sequential_1"
     _________________________________________________________________
      Layer (type)                Output Shape              Param #
     =================================================================
      lstm_1 (LSTM)               (None, 128)               67584

      dropout_1 (Dropout)         (None, 128)               0

      dense_2 (Dense)             (None, 64)                8256

      dense_3 (Dense)             (None, 6)                 390

     =================================================================
     Total params: 76,230
     Trainable params: 76,230
     Non-trainable params: 0
     _________________________________________________________________
```

Figure 19: Vanilla LSTM



3.5. Project Pipeline

A machine learning pipeline is a way to codify and automate the workflow it takes to produce a machine learning model. The pipelines consist of multiple sequential steps that do everything from data extraction and pre-processing to model training and deployment.

The pipeline is used to help automate machine learning workflows. They operate by enabling a sequence of data to be transformed and correlated together in a model that can be tested and evaluated to achieve an outcome, whether positive or negative.

Pipelines consist of several steps to train a model. The pipelines are iterative as every step is repeated to continuously improve the accuracy of the model and achieve a successful algorithm. To build better machine learning models, and get the most value from them, accessible, scalable and durable storage solutions are imperative, paving the way for on-premises object storage.

Now-a-days Data has become a modern-day currency. Tremendous value and intelligence is being extracted from large, captured datasets (Big data) that has led to actionable insights through today’s world. It’s not just about storing data any longer, but capturing, preserving, accessing and transforming it to take advantage of its possibilities and the value it can deliver.

1. The main objective of having a proper pipeline for any ML model is to exercise control over it. A well-organised pipeline makes the implementation more flexible. It is like having an exploded view of a computer where you can pick the faulty pieces and replace it- in our case, replacing a chunk of code.

2. The term ML model refers to the model that is created by the training process.

3. The learning algorithm finds patterns in the training data that map the input data attributes to the target (the answer to be predicted), and it outputs an ML model that captures these patterns.

4. A model can have many dependencies and to store all the components to make sure all features available both offline and online for deployment, all the information is stored in a central repository.

5. A pipeline consists of a sequence of components which are a compilation of computations.

Data is sent through these components and is manipulated with the help of computation.
Pipelines are not one-way flows. They are cyclic in nature and enable iteration to improve the scores of the machine learning algorithms and make the model scalable.

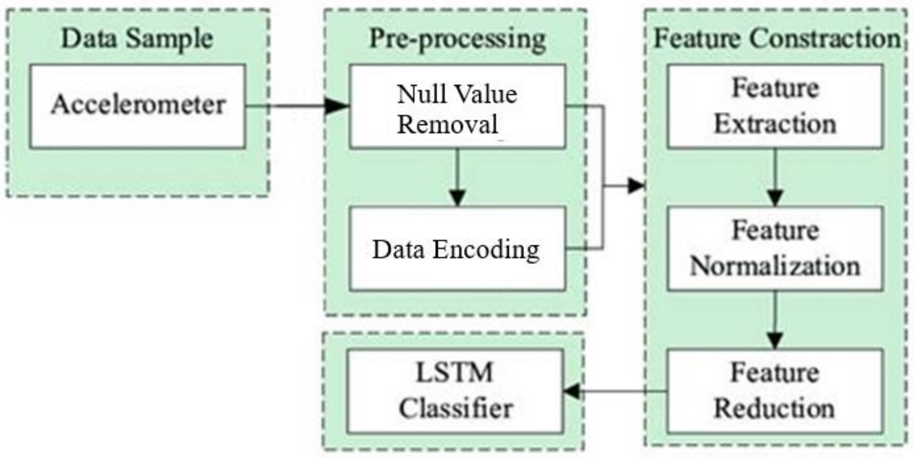
A typical machine learning pipeline would consist of the following processes:

- Data collection
- Data cleaning
- Feature extraction (labelling and dimensionality reduction)
- Model validation
- Visualisation

Data collection and cleaning are the primary tasks of any machine learning engineer who wants to make meaning out of data. But getting data and especially getting the right data is an uphill task in itself. Data quality and its accessibility are two main challenges one will come across in the initial stages of building a pipeline. The captured data should be pulled and put together and the benefits of collection should outweigh the costs of collection and analysis. For this purpose, a data lake is recommended for every organisation. A data lake is a centralised repository that allows the user to store both structured and unstructured data at any scale. It also enables ad-hoc analysis by applying schemas to read, not write. In this way, the user can apply multiple analytics and processing frameworks to the same data.

3.5.1. Project Pipeline

Figure 20: LSTM pipeline



3.6. Feasibility Analysis

The feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort, and time that spend on it. A feasibility study lets the developer foresee the future of the project and its usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet its user needs, and effective use of resources. Thus, when a new application is proposed it normally goes through a feasibility study before it is approved for development. There are three aspects in the feasibility study portion of the preliminary investigation

- Technical Feasibility
- Economic Feasibility
- Operational Feasibility

Technical Feasibility

A technical study is a study of hardware and software requirements. The application “Human Activity Recognition” is technically feasible because all the technical resources required for the development and working of the application are easily available and reliable. The hardware requirement of this application is an android device.

The codes are written in Google Colab. The advantage of Colab is that we can create a separate environment for our project with the required libraries installed. The interface is developed using Android Studio which is simple to understand.

There is no need to develop any hardware to use this system. These requirements are easily available, and reliable, and will make the system more time-saving and require less manpower. So, there is no need to install any bulk software for using this application and there is no need to use any special equipment, thus the system is technically feasible and non-intrusive.

Economic Feasibility

Economic analysis is the most frequently used method for evaluating the effectiveness of a candidate system. The Human Activity Recognition system is cost-effective and has budgetary constraints, it is cheap and quick to implement. The cost to manage the system will be less. This system will require only an android device for working. The development of the system will not require a huge amount of money since there isn’t any extra requirement or peripherals or software for the development of the system as it can be completed with the available resources. So, it is economically feasible.

Operational Feasibility

Operational feasibility is the measure of how well a proposed system solves the problems and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. Human Activity Recognition is easy to operate because it only uses simple steps to classify human activity. The developed system is completely driven and user-friendly since the code is written in Google Colab which has a separate environment. The application is simple for the user to use and hence the system is not complicated. So, it is feasible.

3.7. System Environment

The system environment specifies the hardware and software configuration of the new system. Regardless of how the requirement phase proceeds, it ultimately ends with the software requirement specification. A good SRS contains all the system requirements to a level of detail sufficient to enable designers to design a system that satisfies those requirements. The system specified in the SRS will assist the potential users to determine if the system meets their needs or how the system must be modified to meet their needs.

3.7.1. Software Environment

Front End: Android
Back End: Python

Python: Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented imperative, functional and procedural, and has a large and comprehensive standard library. Python interpreters are available for many operating systems. It has a wide range of applications from Web development (like Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D). Python is a widely used high-level programming language for general-purpose programming. Apart from being an open-source programming language, python is a great object-oriented, interpreted, and interactive programming language. Python combines remarkable power with very learn syntax. It has modules, classes, exceptions, very high-level dynamic data types, and dynamic typing. There are interfaces to many systems calls and libraries, as well as to various windowing systems. New built-in modules are easily written in C or C++ (or other languages, depending on the chosen implementation). Python is also usable as an extension language for applications written in other languages that need easy-to-use scripting or automation interfaces. Python is also usable as an extension language for applications written in other languages that need easy-to-use scripting or automation interfaces.

A few simple reasons are:

- It’s simple to learn. As compared to C, C++, and Java the syntax is simpler and Python also consists of a lot of code libraries for ease of use.

- Though it is slower than some of the other languages, the data handling capacity is great.

- Open Source! – Python along with R is gaining momentum and popularity in the Analytics domain since both of these languages are open-source Capability of interacting with almost all third-party languages and platforms.

Github: Git is an open-source version control system that was started by Linus Torvalds the same person who created Linux. Git is similar to other version control systems Subversion, CVS, and Mercurial to name a few. Version control systems keep these revisions straight, storing the modifications in a central repository. This allows developers to easily collaborate, as they can download a new version of the software, make changes, and upload the newest revision. Every developer can see these new changes, download them, and contribute. Git is the preferred version control system of most developers since it has multiple advantages over the other systems available. It stores file changes more efficiently and ensures file integrity better. If you’re interested in knowing the details, the Git Basics page has a thorough explanation of how Git works. The social networking aspect of GitHub is probably its most powerful feature, allowing projects to grow more than just about any of the other features offered. Project revisions can be discussed publicly, so a mass of experts can contribute knowledge and collaborate to advance a project forward.

Google Colab: Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser and is especially well suited to machine learning, data analysis, and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs. Colab is free of charge to use.

Android: Android is an open source and Linux-based Operating System for mobile devices such as smartphones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies.

Android offers a unified approach to application development for mobile devices which means developers need only develop for Android, and their applications should be able to run on different devices powered by Android. The source code for Android is available under free and open source software licenses. Google publishes most of the code under the Apache License version 2.0 and the rest, Linux kernel changes, under the GNU General Public License version 2.

Android applications are usually developed in the Java language using the Android Software Development Kit. Once developed, Android applications can be packaged easily and sold out either through a store such as Google Play, SlideME, Opera Mobile Store, Mobango, F-droid and the Amazon Appstore.

Android powers hundreds of millions of mobile devices in more than 190 countries around the world. It's the largest installed base of any mobile platform and growing fast. Every day more than 1 million new Android devices are activated worldwide.

3.7.2. Hardware Environment

RAM: 2 GB or more
Storage: 1GB or more
Sensor: Accelerometer

4. System Design

Despite many solutions that have been recently proposed, there are still open challenges in creating a user-friendly application concerning Human Activity Recognition. The solution proposed here aims to solve these limitations, by developing a user-friendly application that considers the parameters from various test results. The main objective is to predict human activity by reading accelerometer values from the smartphone.

4.1. Model Planning

After the comparison of architectures: LSTM & GRU, it is decided that this model will be developed using LSTM because of its speed and efficiency. The input to the LSTM model is the combination of two datasets: WISDM & UCI-HAR. These datasets once combined are then split into training and testing sets to train and evaluate the model. The model will use the 'adam' optimizer and the loss parameter will be set to 'categorical_crossentropy' because the model tries to predict multiclass classification.

4.2. Training

Training data is the initial dataset we use to teach a deep learning application to recognize a pattern. A training dataset is used to train the model, to get the correct prediction by the model.

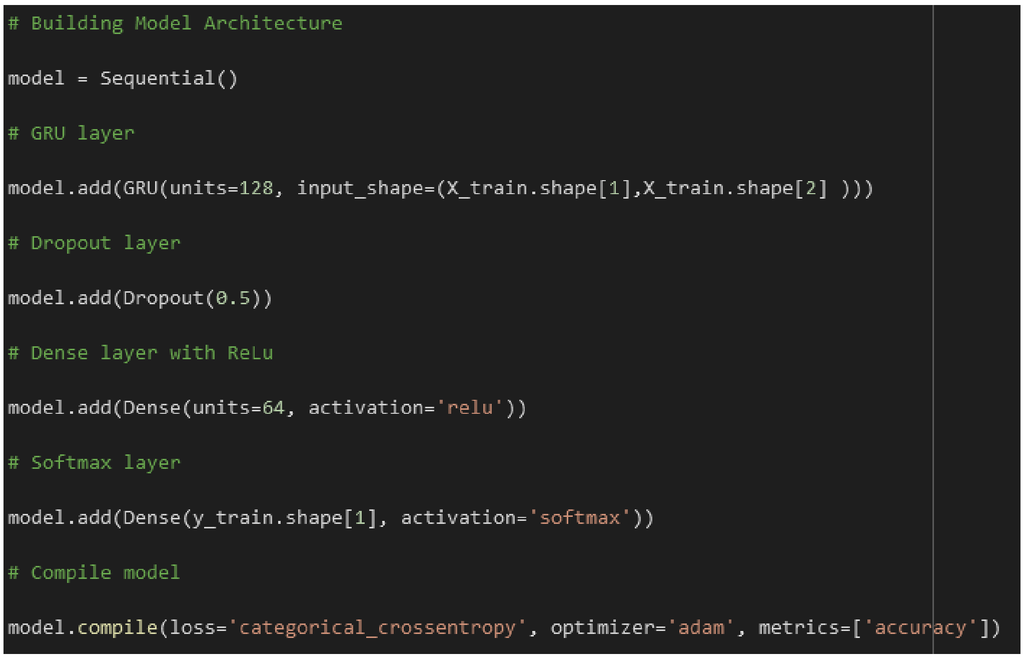
All the models are set to run 100 epochs but will automatically stop training when the model validation loss becomes a constant or when there is no significant change in the readings.

GRU

Here we use the GRU architecture which was developed using Tensorflow and Keras libraries. In this model, the sequential layer is called first which is followed by an LSTM layer. A dropout layer of value 0.5 is included in this model as a regularization method to avoid overfitting and improve the robustness.

After the dropout two dense layers are added with activation functions ReLU and Softmax respectively, this is done to improve the generalization and to regularize their output to prevent vanishing gradient.

Figure 21: GRU model

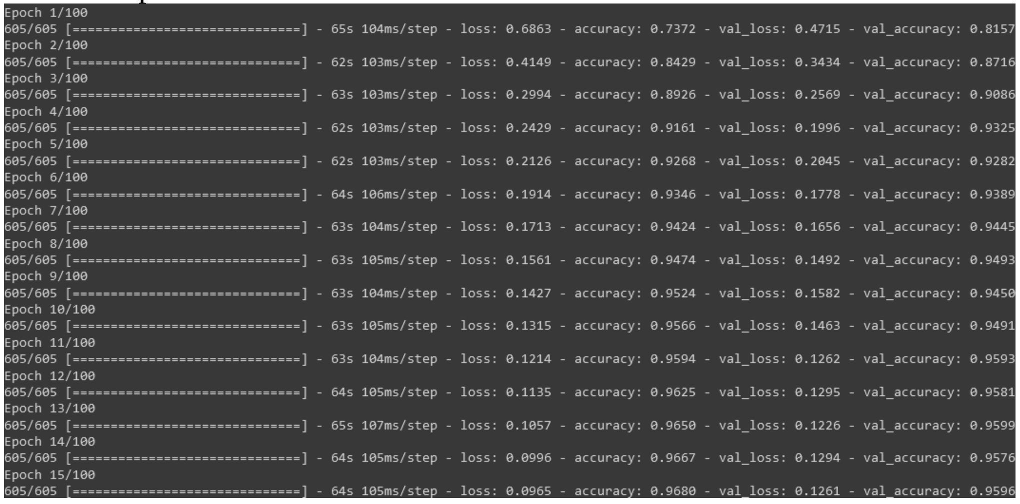


```
# Building Model Architecture

model = Sequential()

# GRU layer

model.add(GRU(units=128, input_shape=(X_train.shape[1],X_train.shape[2] )))

# Dropout layer

model.add(Dropout(0.5))

# Dense layer with ReLu

model.add(Dense(units=64, activation='relu'))

# Softmax layer

model.add(Dense(y_train.shape[1], activation='softmax'))

# Compile model

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Table 2: GRU hyperparameters

Phase	Hyperparameters	Value
Training	Loss function	Categorical cross entropy
	Optimizer	Adam
	Learning rate	0.0025
	Batch size	128
	Epochs	100

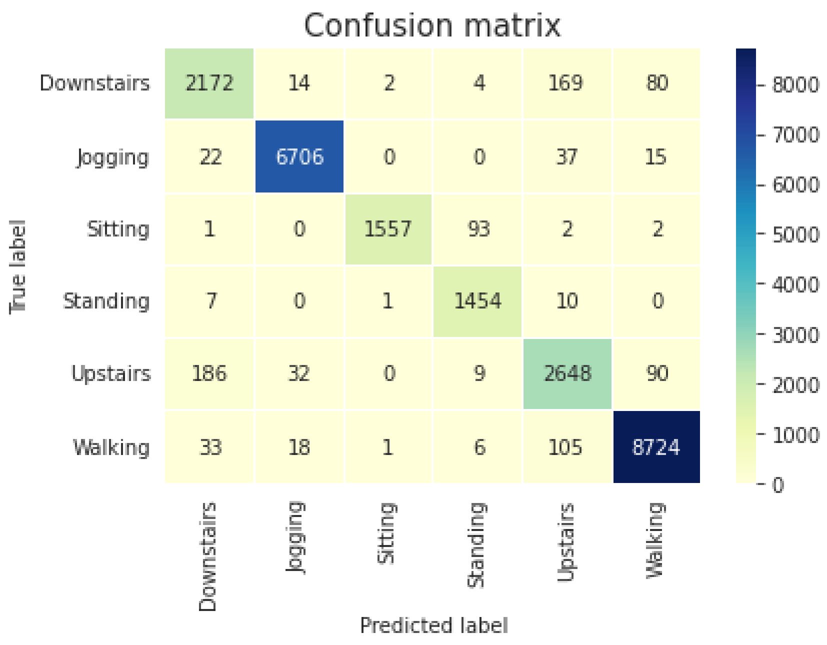
Figure 22: GRU epochs



```
Epoch 1/100
605/605 [==============================] - 65s 104ms/step - loss: 0.6863 - accuracy: 0.7372 - val_loss: 0.4715 - val_accuracy: 0.8157
Epoch 2/100
605/605 [==============================] - 62s 103ms/step - loss: 0.4149 - accuracy: 0.8429 - val_loss: 0.3434 - val_accuracy: 0.8716
Epoch 3/100
605/605 [==============================] - 63s 103ms/step - loss: 0.2994 - accuracy: 0.8926 - val_loss: 0.2569 - val_accuracy: 0.9086
Epoch 4/100
605/605 [==============================] - 62s 103ms/step - loss: 0.2429 - accuracy: 0.9161 - val_loss: 0.1996 - val_accuracy: 0.9325
Epoch 5/100
605/605 [==============================] - 62s 103ms/step - loss: 0.2126 - accuracy: 0.9268 - val_loss: 0.2045 - val_accuracy: 0.9282
Epoch 6/100
605/605 [==============================] - 64s 106ms/step - loss: 0.1914 - accuracy: 0.9346 - val_loss: 0.1778 - val_accuracy: 0.9389
Epoch 7/100
605/605 [==============================] - 63s 104ms/step - loss: 0.1713 - accuracy: 0.9424 - val_loss: 0.1656 - val_accuracy: 0.9445
Epoch 8/100
605/605 [==============================] - 63s 105ms/step - loss: 0.1561 - accuracy: 0.9474 - val_loss: 0.1492 - val_accuracy: 0.9493
Epoch 9/100
605/605 [==============================] - 63s 104ms/step - loss: 0.1427 - accuracy: 0.9524 - val_loss: 0.1582 - val_accuracy: 0.9450
Epoch 10/100
605/605 [==============================] - 63s 105ms/step - loss: 0.1315 - accuracy: 0.9566 - val_loss: 0.1463 - val_accuracy: 0.9491
Epoch 11/100
605/605 [==============================] - 63s 104ms/step - loss: 0.1214 - accuracy: 0.9594 - val_loss: 0.1262 - val_accuracy: 0.9593
Epoch 12/100
605/605 [==============================] - 64s 105ms/step - loss: 0.1135 - accuracy: 0.9625 - val_loss: 0.1295 - val_accuracy: 0.9581
Epoch 13/100
605/605 [==============================] - 65s 107ms/step - loss: 0.1057 - accuracy: 0.9650 - val_loss: 0.1226 - val_accuracy: 0.9599
Epoch 14/100
605/605 [==============================] - 64s 105ms/step - loss: 0.0996 - accuracy: 0.9667 - val_loss: 0.1294 - val_accuracy: 0.9576
Epoch 15/100
605/605 [==============================] - 64s 105ms/step - loss: 0.0965 - accuracy: 0.9680 - val_loss: 0.1261 - val_accuracy: 0.9596
```

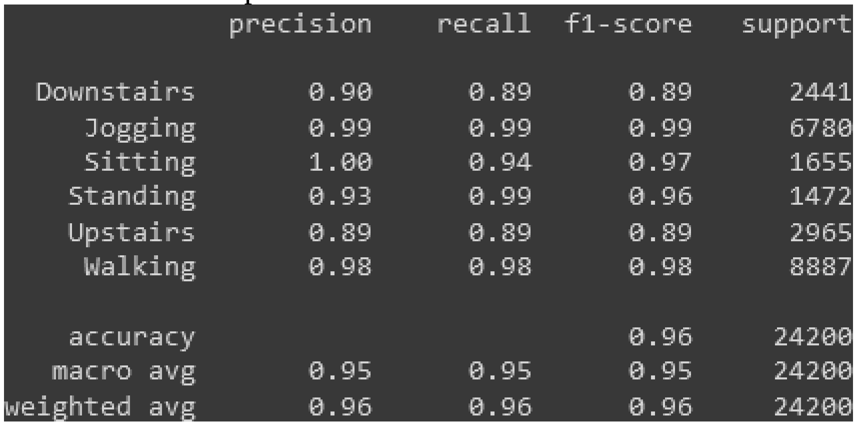
The training accuracy in the 1st epoch is 73.72% which increased to 96.8% in the last epoch (15th). Here, in the 15th epoch, the model training is automatically stopped because there is no reduction in the validation loss of the model.

Figure 23: GRU confusion matrix



	Downstairs	Jogging	Sitting	Standing	Upstairs	Walking
Downstairs	2172	14	2	4	169	80
Jogging	22	6706	0	0	37	15
Sitting	1	0	1557	93	2	2
Standing	7	0	1	1454	10	0
Upstairs	186	32	0	9	2648	90
Walking	33	18	1	6	105	8724

Figure 24: GRU classification report



```
              precision    recall  f1-score   support

  Downstairs       0.90      0.89      0.89      2441
     Jogging       0.99      0.99      0.99      6780
     Sitting       1.00      0.94      0.97      1655
    Standing       0.93      0.99      0.96      1472
    Upstairs       0.89      0.89      0.89      2965
     Walking       0.98      0.98      0.98      8887

    accuracy                           0.96     24200
   macro avg       0.95      0.95      0.95     24200
weighted avg       0.96      0.96      0.96     24200
```

Figure 25: GRU model accuracy

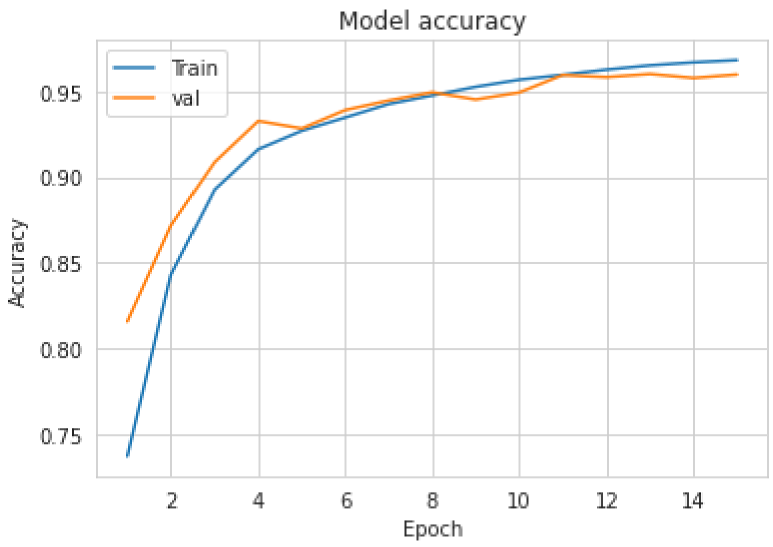
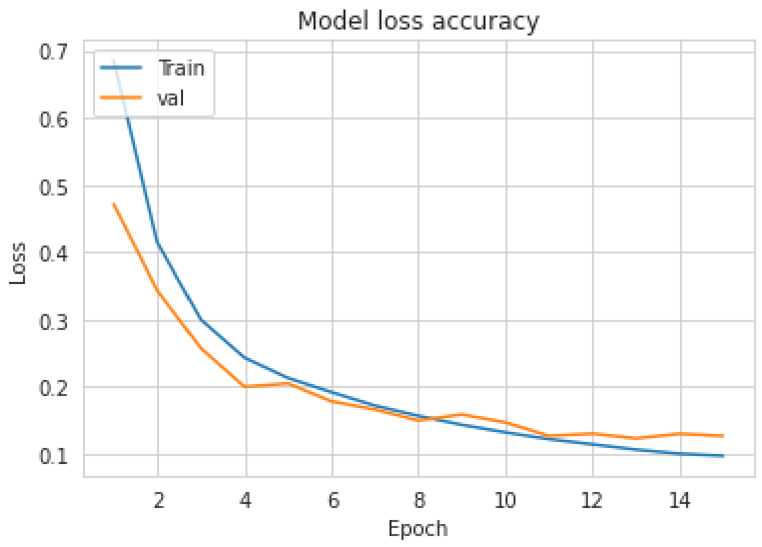


Figure 26: GRU model loss

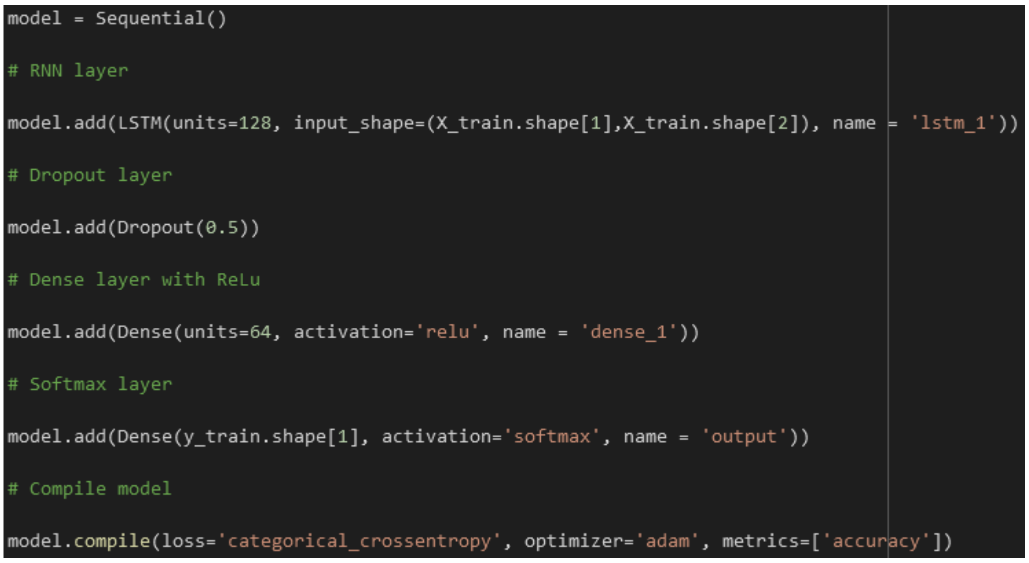


LSTM

Here we use the LSTM architecture which was developed using Tensorflow and Keras libraries. In this model, the sequential layer is called first which is followed by an LSTM layer. A dropout layer of value 0.5 is included in this model as a regularization method to avoid overfitting and improve the robustness.

After the dropout two dense layers are added with activation functions ReLU and Softmax respectively, this is done to improve the generalization and to regularize their output to prevent vanishing gradient.

Figure 27: LSTM model

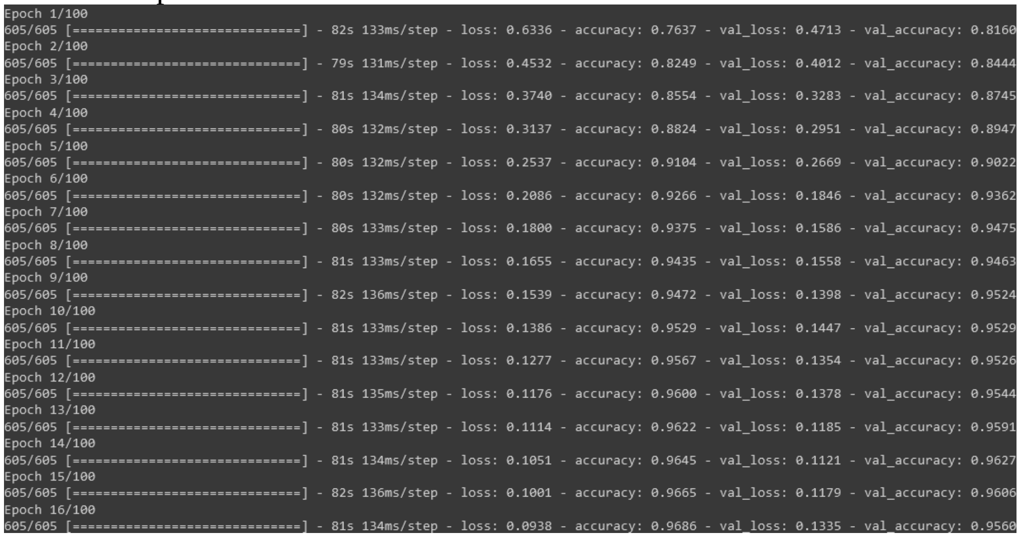


```
model = Sequential()

# RNN layer

model.add(LSTM(units=128, input_shape=(X_train.shape[1],X_train.shape[2]), name = 'lstm_1'))

# Dropout layer

model.add(Dropout(0.5))

# Dense layer with ReLu

model.add(Dense(units=64, activation='relu', name = 'dense_1'))

# Softmax layer

model.add(Dense(y_train.shape[1], activation='softmax', name = 'output'))

# Compile model

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Table 3: LSTM hyperparametes

Phase	Hyperparameters	Value
Training	Loss function	Categorical cross entropy
	Optimizer	Adam
	Learning rate	0.0025
	Batch size	128
	Epochs	100

Figure 28: LSTM epochs



```
Epoch 1/100
605/605 [==============================] - 82s 133ms/step - loss: 0.6336 - accuracy: 0.7637 - val_loss: 0.4713 - val_accuracy: 0.8160
Epoch 2/100
605/605 [==============================] - 79s 131ms/step - loss: 0.4532 - accuracy: 0.8249 - val_loss: 0.4012 - val_accuracy: 0.8444
Epoch 3/100
605/605 [==============================] - 81s 134ms/step - loss: 0.3740 - accuracy: 0.8554 - val_loss: 0.3283 - val_accuracy: 0.8745
Epoch 4/100
605/605 [==============================] - 80s 132ms/step - loss: 0.3137 - accuracy: 0.8824 - val_loss: 0.2951 - val_accuracy: 0.8947
Epoch 5/100
605/605 [==============================] - 80s 132ms/step - loss: 0.2537 - accuracy: 0.9104 - val_loss: 0.2669 - val_accuracy: 0.9022
Epoch 6/100
605/605 [==============================] - 80s 132ms/step - loss: 0.2086 - accuracy: 0.9266 - val_loss: 0.1846 - val_accuracy: 0.9362
Epoch 7/100
605/605 [==============================] - 80s 133ms/step - loss: 0.1800 - accuracy: 0.9375 - val_loss: 0.1586 - val_accuracy: 0.9475
Epoch 8/100
605/605 [==============================] - 81s 133ms/step - loss: 0.1655 - accuracy: 0.9435 - val_loss: 0.1558 - val_accuracy: 0.9463
Epoch 9/100
605/605 [==============================] - 82s 136ms/step - loss: 0.1539 - accuracy: 0.9472 - val_loss: 0.1398 - val_accuracy: 0.9524
Epoch 10/100
605/605 [==============================] - 81s 133ms/step - loss: 0.1386 - accuracy: 0.9529 - val_loss: 0.1447 - val_accuracy: 0.9529
Epoch 11/100
605/605 [==============================] - 81s 133ms/step - loss: 0.1277 - accuracy: 0.9567 - val_loss: 0.1354 - val_accuracy: 0.9526
Epoch 12/100
605/605 [==============================] - 81s 135ms/step - loss: 0.1176 - accuracy: 0.9600 - val_loss: 0.1378 - val_accuracy: 0.9544
Epoch 13/100
605/605 [==============================] - 81s 133ms/step - loss: 0.1114 - accuracy: 0.9622 - val_loss: 0.1185 - val_accuracy: 0.9591
Epoch 14/100
605/605 [==============================] - 81s 134ms/step - loss: 0.1051 - accuracy: 0.9645 - val_loss: 0.1121 - val_accuracy: 0.9627
Epoch 15/100
605/605 [==============================] - 82s 136ms/step - loss: 0.1001 - accuracy: 0.9665 - val_loss: 0.1179 - val_accuracy: 0.9606
Epoch 16/100
605/605 [==============================] - 81s 134ms/step - loss: 0.0938 - accuracy: 0.9686 - val_loss: 0.1335 - val_accuracy: 0.9560
```

Figure 29: LSTM confusion matrix

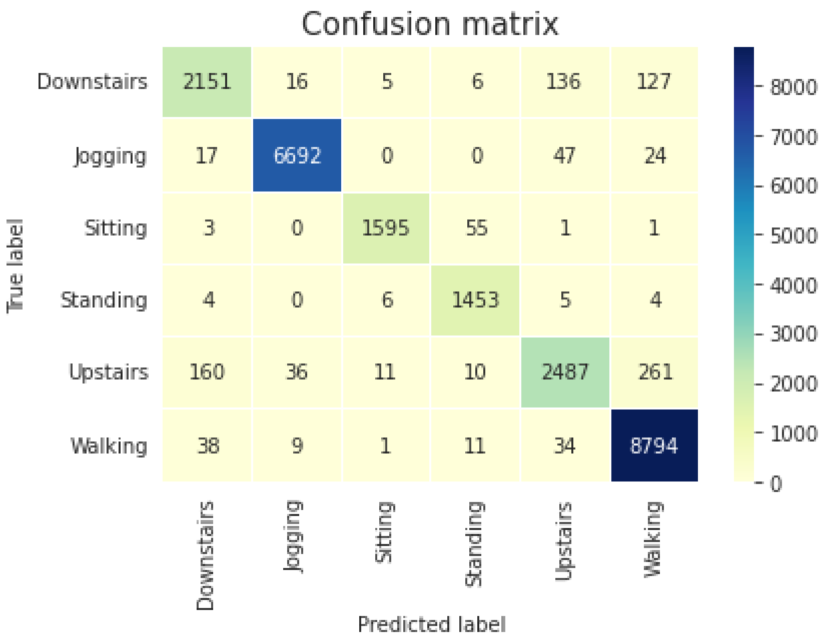
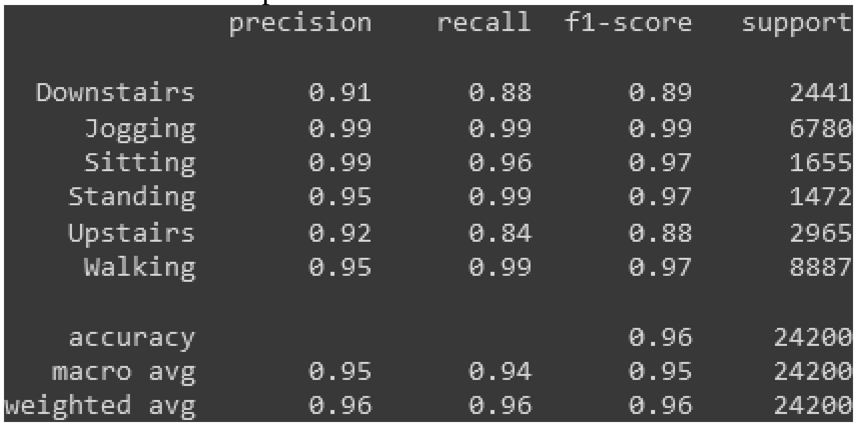


Figure 30: LSTM classification report



```
              precision    recall  f1-score   support

  Downstairs       0.91      0.88      0.89      2441
     Jogging       0.99      0.99      0.99      6780
     Sitting       0.99      0.96      0.97      1655
    Standing       0.95      0.99      0.97      1472
    Upstairs       0.92      0.84      0.88      2965
     Walking       0.95      0.99      0.97      8887

    accuracy                           0.96     24200
   macro avg       0.95      0.94      0.95     24200
weighted avg       0.96      0.96      0.96     24200
```

Figure 31: LSTM model accuracy

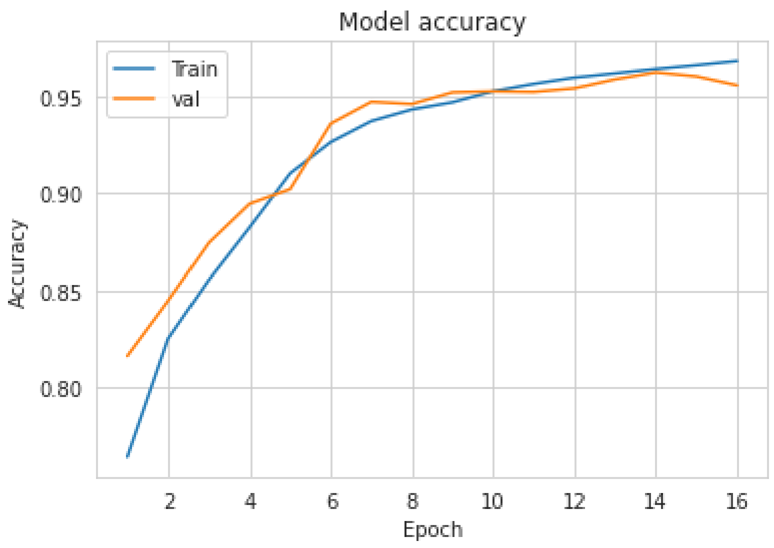


Figure 32: LSTM model loss

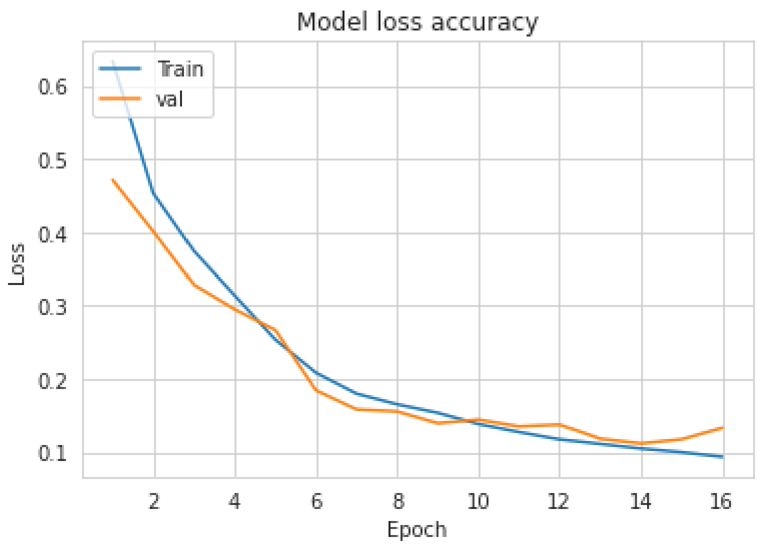


Table 4: LSTM VS GRU

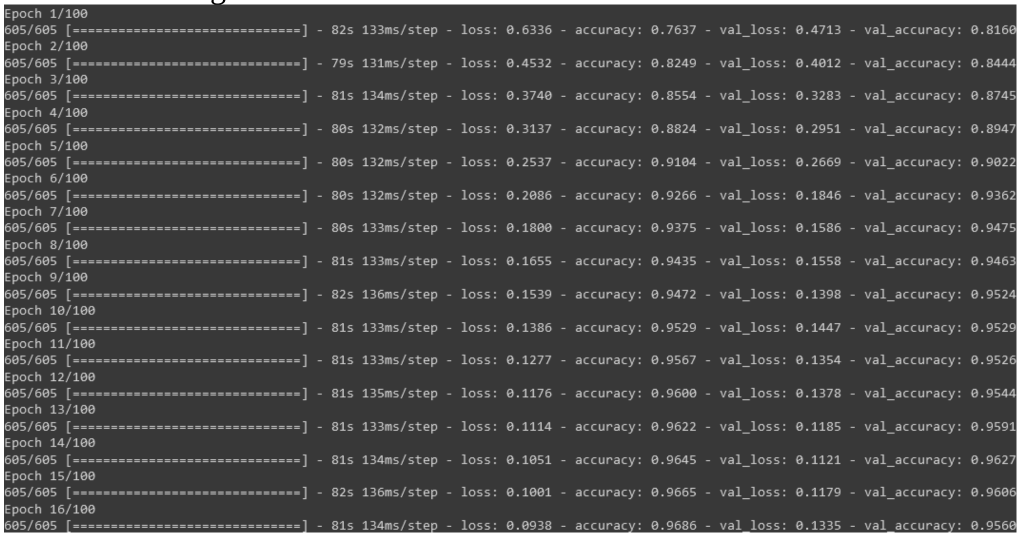
LSTM	GRU
High performance	Low performance
Similar accuracy	
Better to handle large sequence of inputs	Better to handle small sequence of inputs

4.3. Testing

Software testing is a critical element of software quality assurance and represents an ultimate view of specification, design, and code generation. Once the source code has been generated the program should be executed before the customer gets it with the specific intention of finding and removing all errors, the test must be conducted systematically and the test must be designed using disciplined techniques.

Testing or validation data is used to evaluate the model's accuracy. To check whether the application can correctly predict the output. Here, LSTM network was selected and 20% of the dataset is used for testing the model. The model is tested along with training. At each epoch, the model tries to predict the unseen data. The accuracy of the test dataset changes with each epoch and the best model with maximum accuracy and minimum loss is selected.

Figure 33: Model testing



```
Epoch 1/100
605/605 [==============================] - 82s 133ms/step - loss: 0.6336 - accuracy: 0.7637 - val_loss: 0.4713 - val_accuracy: 0.8160
Epoch 2/100
605/605 [==============================] - 79s 131ms/step - loss: 0.4532 - accuracy: 0.8249 - val_loss: 0.4012 - val_accuracy: 0.8444
Epoch 3/100
605/605 [==============================] - 81s 134ms/step - loss: 0.3740 - accuracy: 0.8554 - val_loss: 0.3283 - val_accuracy: 0.8745
Epoch 4/100
605/605 [==============================] - 80s 132ms/step - loss: 0.3137 - accuracy: 0.8824 - val_loss: 0.2951 - val_accuracy: 0.8947
Epoch 5/100
605/605 [==============================] - 80s 132ms/step - loss: 0.2537 - accuracy: 0.9104 - val_loss: 0.2669 - val_accuracy: 0.9022
Epoch 6/100
605/605 [==============================] - 80s 132ms/step - loss: 0.2086 - accuracy: 0.9266 - val_loss: 0.1846 - val_accuracy: 0.9362
Epoch 7/100
605/605 [==============================] - 80s 133ms/step - loss: 0.1800 - accuracy: 0.9375 - val_loss: 0.1586 - val_accuracy: 0.9475
Epoch 8/100
605/605 [==============================] - 81s 133ms/step - loss: 0.1655 - accuracy: 0.9435 - val_loss: 0.1558 - val_accuracy: 0.9463
Epoch 9/100
605/605 [==============================] - 82s 136ms/step - loss: 0.1539 - accuracy: 0.9472 - val_loss: 0.1398 - val_accuracy: 0.9524
Epoch 10/100
605/605 [==============================] - 81s 133ms/step - loss: 0.1386 - accuracy: 0.9529 - val_loss: 0.1447 - val_accuracy: 0.9529
Epoch 11/100
605/605 [==============================] - 81s 133ms/step - loss: 0.1277 - accuracy: 0.9567 - val_loss: 0.1354 - val_accuracy: 0.9526
Epoch 12/100
605/605 [==============================] - 81s 135ms/step - loss: 0.1176 - accuracy: 0.9600 - val_loss: 0.1378 - val_accuracy: 0.9544
Epoch 13/100
605/605 [==============================] - 81s 133ms/step - loss: 0.1114 - accuracy: 0.9622 - val_loss: 0.1185 - val_accuracy: 0.9591
Epoch 14/100
605/605 [==============================] - 81s 134ms/step - loss: 0.1051 - accuracy: 0.9645 - val_loss: 0.1121 - val_accuracy: 0.9627
Epoch 15/100
605/605 [==============================] - 82s 136ms/step - loss: 0.1001 - accuracy: 0.9665 - val_loss: 0.1179 - val_accuracy: 0.9606
Epoch 16/100
605/605 [==============================] - 81s 134ms/step - loss: 0.0938 - accuracy: 0.9686 - val_loss: 0.1335 - val_accuracy: 0.9560
```

Maximum validation accuracy of 96.86% and minimum validation loss of 0.1333 is achieved in the 16th epoch.

5. Results and Discussion

The project aimed to develop an application that will classify six human activities: walking, walking upstairs, walking downstairs, standing, jogging and sitting, with the help of a smartphone-based on the readings from its accelerometer readings. The project has been completed and is developed using LSTM architecture.

Accuracy and loss are the metrics used in the training of the model. Accuracy is a measure of observational error. This model showed better accuracy with an increase in each epoch during the training.

The model at the 16th epoch was selected because it had an accuracy of 96.86% and the loss at this stage was minimum which was recorded at 0.1335.

Learning Curves

Generally, a learning curve is a plot that shows time or experience on the x-axis and learning or improvement on the y-axis. The metric used to evaluate learning could be maximizing, meaning that better scores (larger numbers) indicate more learning. An example would be classification accuracy. During the training of a machine learning model, the current state of the model at each step of the training algorithm can be evaluated. It can be evaluated on the training dataset to give an idea of how well the model is "learning." It can also be evaluated on a hold-out validation dataset that is not part of the training dataset. Evaluation of the validation dataset gives an idea of how well the model is "generalizing."

- Train Learning Curve: The learning curve calculated from the training dataset gives an idea of how well the model is learning.

- Validation Learning Curve: The learning curve is calculated from a hold-out validation dataset that gives an idea of how well the model is generalizing.

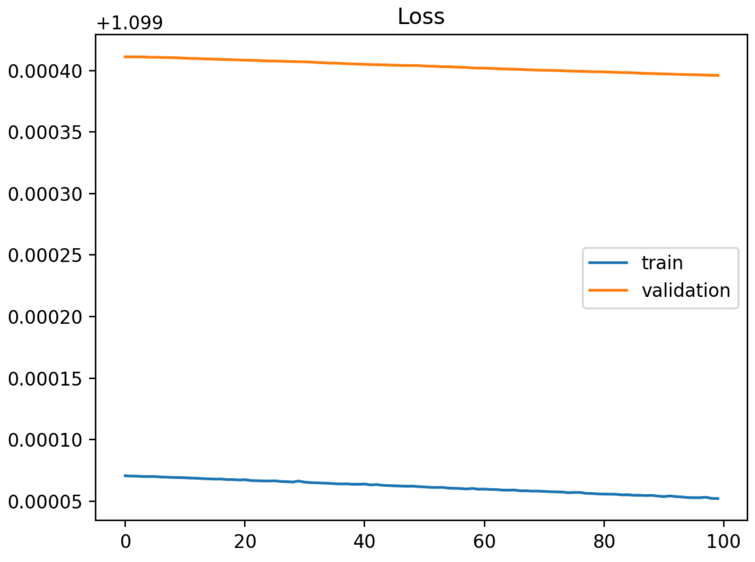
The shape and dynamics of a learning curve can be used to diagnose the behavior of a machine learning model and in turn, perhaps suggest the type of configuration changes that may be made to improve learning and/or performance. There are three common dynamics that you are likely to observe in learning curves; they are:

- Underfit.
- Overfit.
- Good Fit.

Underfit Learning Curves

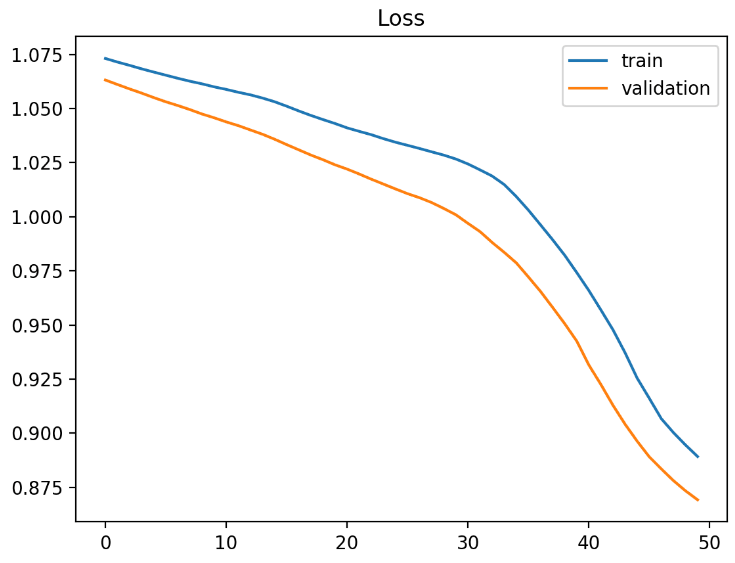
An underfit model can be identified from the learning curve of the training loss only. It may show a flat line or noisy values of relatively high loss, indicating that the model was unable to learn the training dataset at all. An example of this is provided below and is common when the model does not have a suitable capacity for the complexity of the dataset.

Figure 34: Underfit (e.g. 1)



An underfit model may also be identified by a training loss that is decreasing and continues to decrease at the end of the plot. This indicates that the model is capable of further learning and possible further improvements and that the training process was halted prematurely.

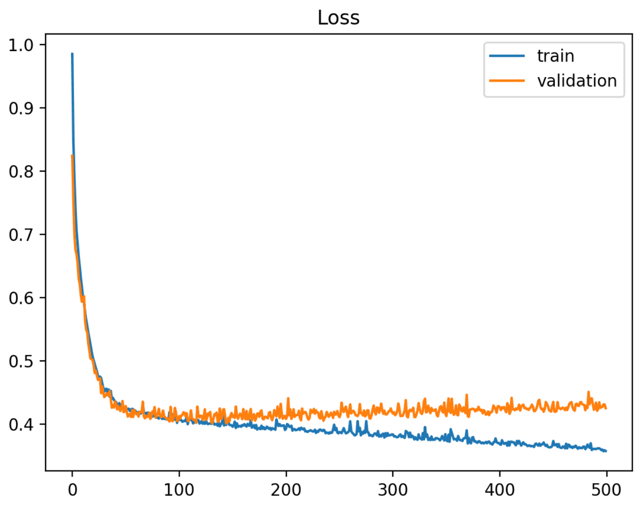
Figure 35: Underfit (e.g. 2)



Overfit Learning Curves

Overfitting refers to a model that has learned the training dataset too well, including the statistical noise or random fluctuations in the training dataset. The problem with overfitting is that the more specialized the model becomes for training data, the less well it can generalize to new data, increasing generalization error. This increase in generalization error can be measured by the performance of the model on the validation dataset.

Figure 36: Overfit

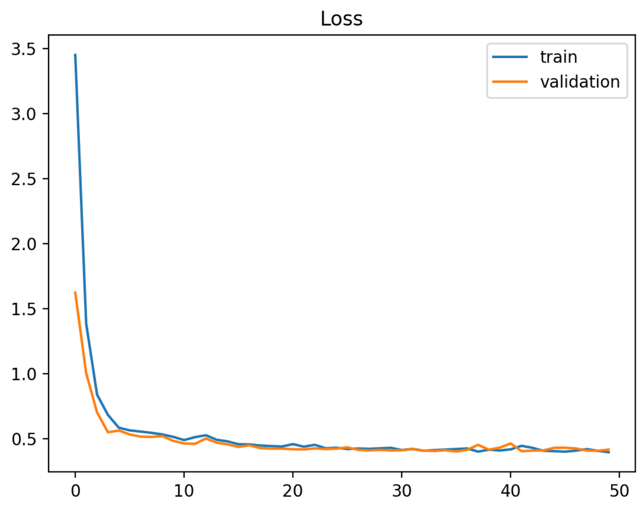


Good Fit Learning Curves

A good fit is the goal of the learning algorithm and exists between an overfit and underfit model. A good fit is identified by a training and validation loss that decreases to a point of stability with a minimal gap between the two final loss values.

The loss of the model will almost always be lower on the training dataset than on the validation dataset. This means that we should expect some gap between the train and validation loss learning curves. This gap is referred to as the “generalization gap.”

Figure 37: Good fit



The snapshots below represent the learning curves for 16 epochs. 1st graph is plotted between no of epochs and the accuracy of the model. Then the 2nd graph is plotted between no of epochs and loss of a model. Here we got quite a good accuracy. As validation loss is less than training loss, we can say that the model is neither overfitting nor underfitting.

Figure 38: Model accuracy

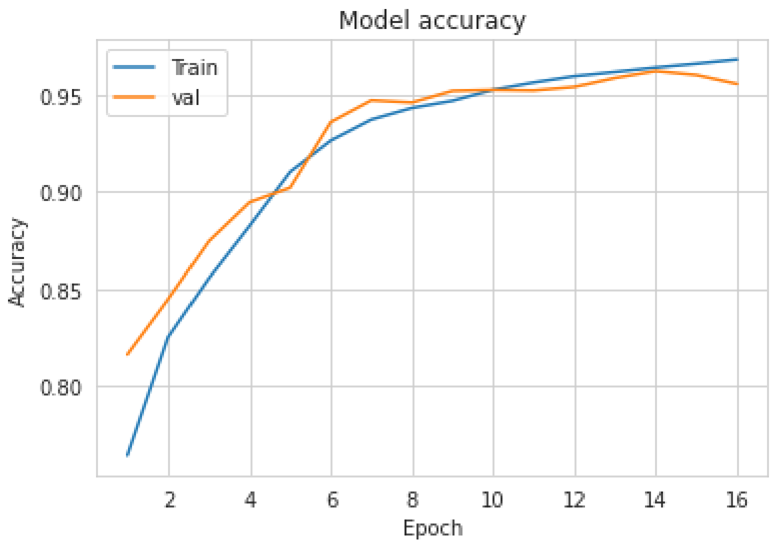
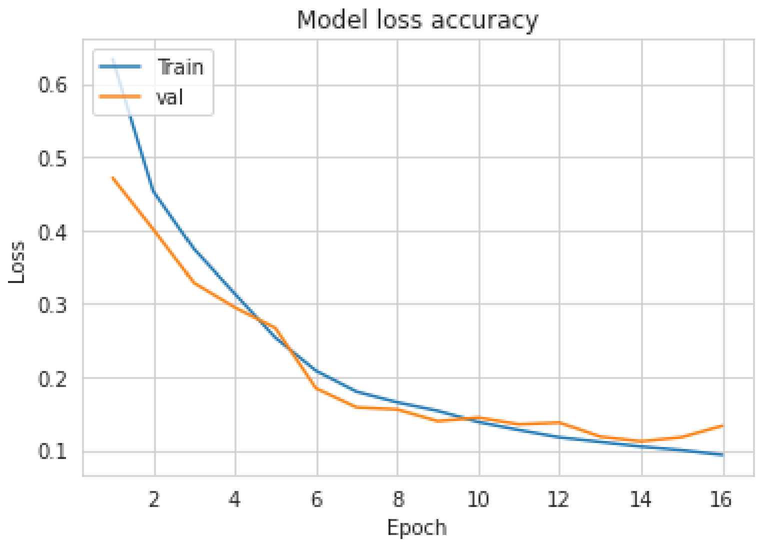


Figure 39: Model loss



6. Model Deployment

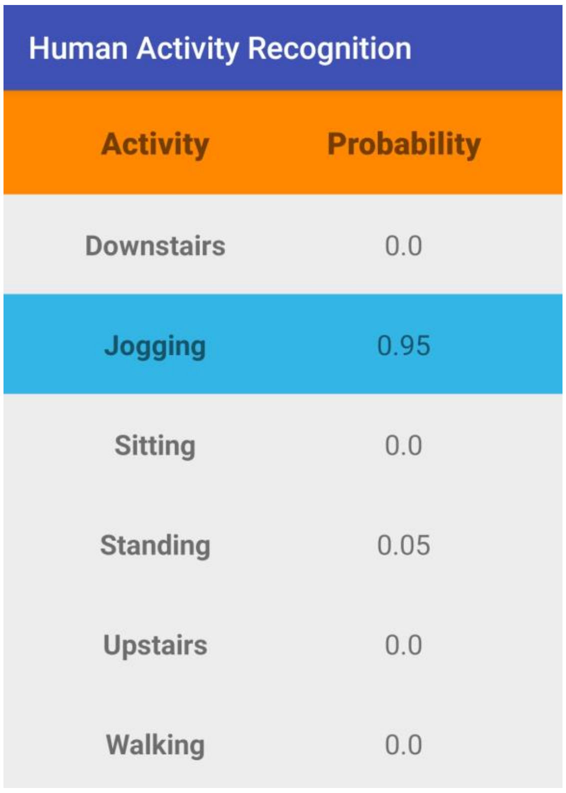
The implementation simply means carrying out the activities described in the requirement. After testing, the system “Human Activity Recognition” is ready for implementation. Implementation is the stage of the project when the theoretical design is turned into a working system. Implementation is the process of bringing a newly developed system or revised into an operational one. The new system and its components are to be tested in a structured and planned manner. There are some challenges faced while implementing the software.

The implementation stage of a project is often very complex and time-consuming. This involves careful planning, investigation of the current system and constraints of implementation, and training the operating users in the changeover procedures before the system is set up and running. So, “Human Activity Recognition” is easy to implement and would be very easy to run also.

The Human Activity Recognition system is implemented successfully. The system predicts human activity using various test results and predicts the stage using raw accelerometer values from the mentioned datasets.

UI Design

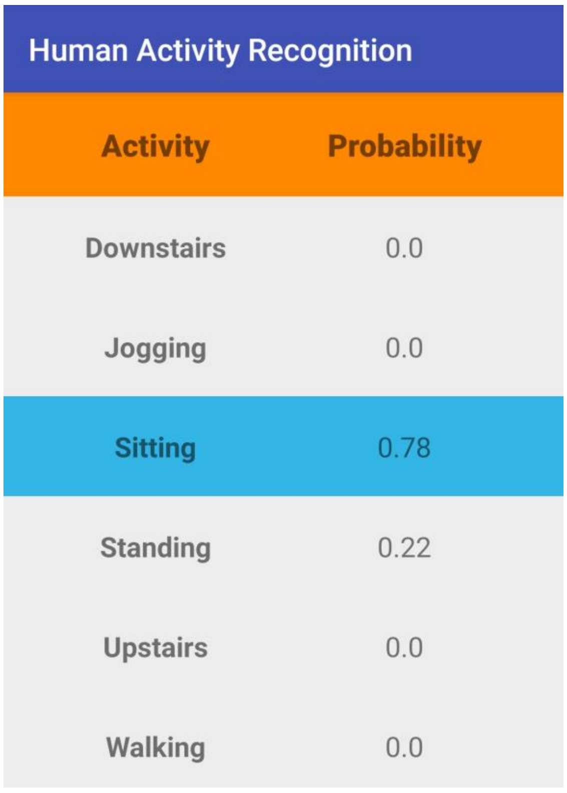
Figure 40: UI (Jogging)



Human Activity Recognition

Activity	Probability
Downstairs	0.0
Jogging	0.95
Sitting	0.0
Standing	0.05
Upstairs	0.0
Walking	0.0

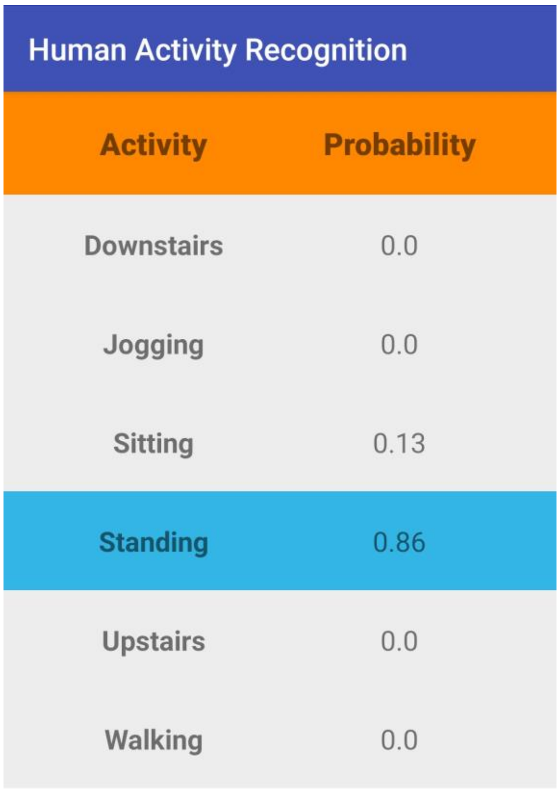
Figure 41: UI (Sitting)



Human Activity Recognition

Activity	Probability
Downstairs	0.0
Jogging	0.0
Sitting	0.78
Standing	0.22
Upstairs	0.0
Walking	0.0

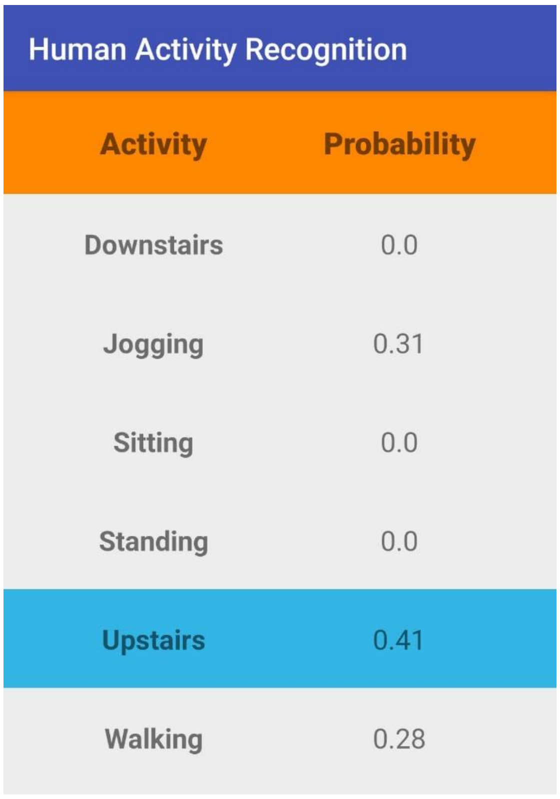
Figure 42: UI (Standing)



Human Activity Recognition

Activity	Probability
Downstairs	0.0
Jogging	0.0
Sitting	0.13
Standing	0.86
Upstairs	0.0
Walking	0.0

Figure 43: UI (Upstairs)



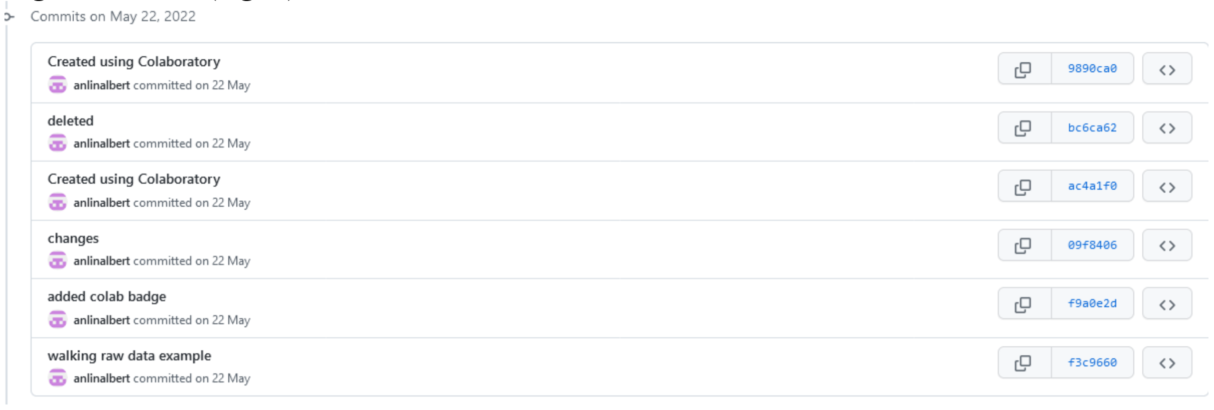
Human Activity Recognition

Activity	Probability
Downstairs	0.0
Jogging	0.31
Sitting	0.0
Standing	0.0
Upstairs	0.41
Walking	0.28

7. GIT History

There is about a total 86 commits made to the git repository in total during the entire working of the project.

Figure 44: GIT (e.g. 1)



Commits on May 22, 2022

Created using Colaboratory
anlinalbert committed on 22 May
9890ca0

deleted
anlinalbert committed on 22 May
bc6ca62

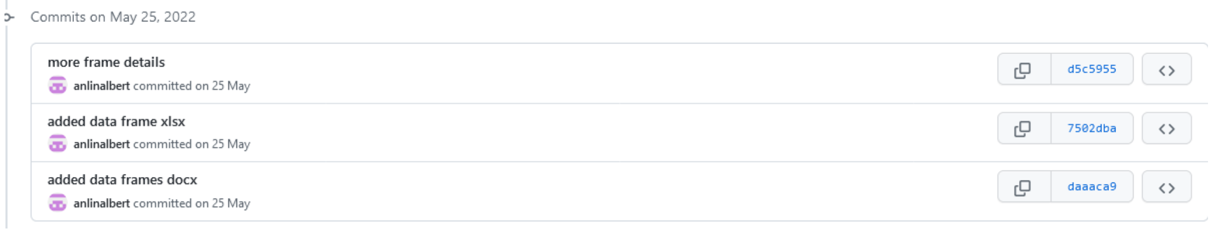
Created using Colaboratory
anlinalbert committed on 22 May
ac4a1f0

changes
anlinalbert committed on 22 May
09f8406

added colab badge
anlinalbert committed on 22 May
f9a0e2d

walking raw data example
anlinalbert committed on 22 May
f3c9660

Figure 45: GIT (e.g. 2)



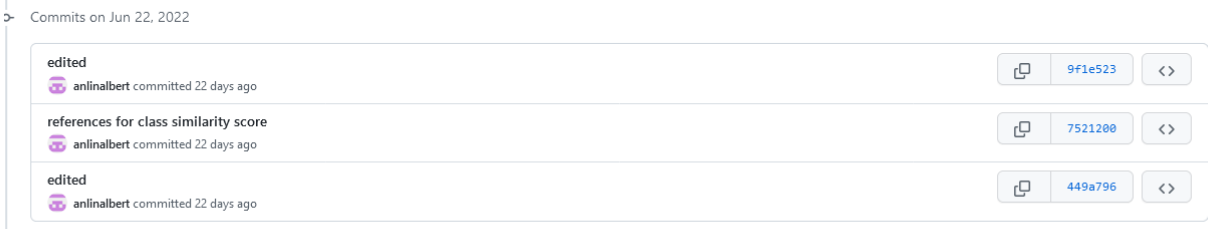
Commits on May 25, 2022

more frame details
anlinalbert committed on 25 May
d5c5955

added data frame xlsx
anlinalbert committed on 25 May
7502dba

added data frames docx
anlinalbert committed on 25 May
daaaca9

Figure 46: GIT (e.g. 3)



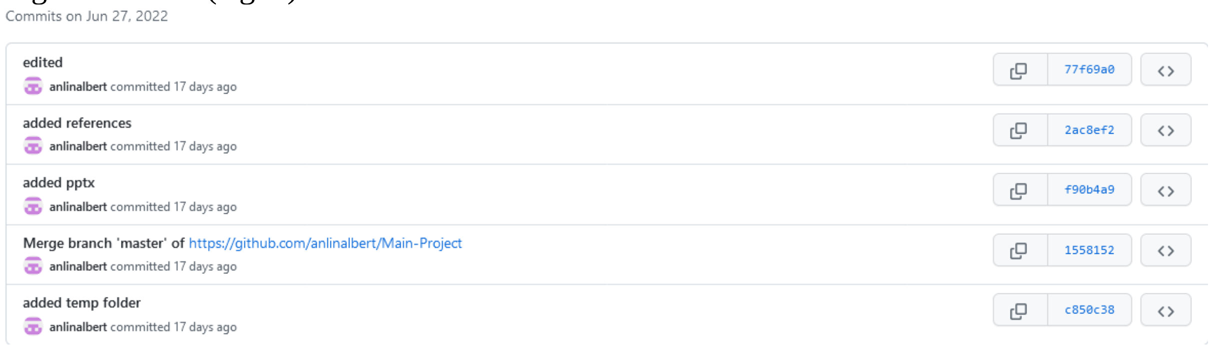
Commits on Jun 22, 2022

edited
anlinalbert committed 22 days ago
9f1e523

references for class similarity score
anlinalbert committed 22 days ago
7521200

edited
anlinalbert committed 22 days ago
449a796

Figure 47: GIT (e.g. 4)



Commits on Jun 27, 2022

edited
anlinalbert committed 17 days ago
77f69a0

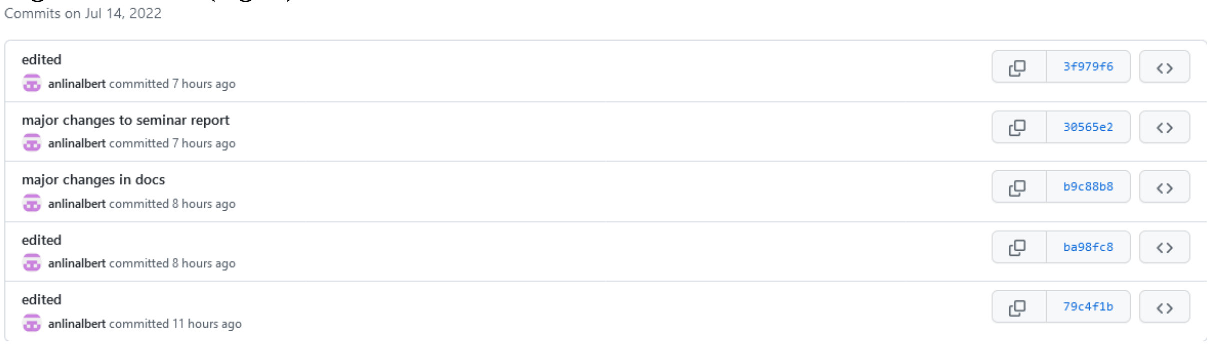
added references
anlinalbert committed 17 days ago
2ac8ef2

added pptx
anlinalbert committed 17 days ago
f90b4a9

Merge branch 'master' of https://github.com/anlinalbert/Main-Project
anlinalbert committed 17 days ago
1558152

added temp folder
anlinalbert committed 17 days ago
c850c38

Figure 48: GIT (e.g. 5)



Commits on Jul 14, 2022

edited
anlinalbert committed 7 hours ago — 3f979f6

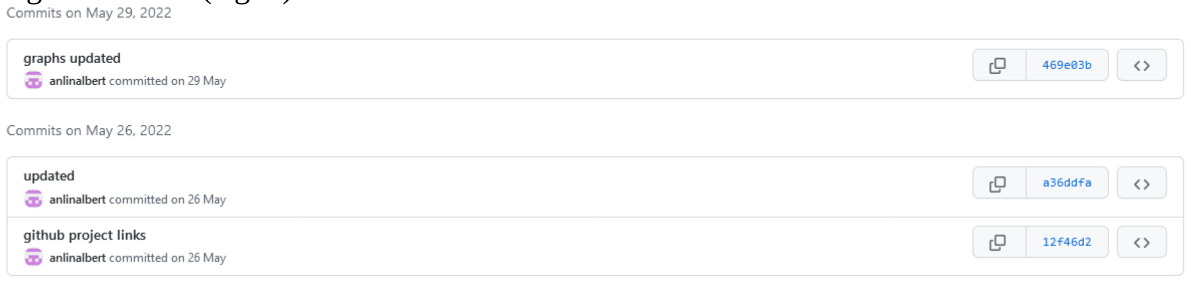
major changes to seminar report
anlinalbert committed 7 hours ago — 30565e2

major changes in docs
anlinalbert committed 8 hours ago — b9c88b8

edited
anlinalbert committed 8 hours ago — ba98fc8

edited
anlinalbert committed 11 hours ago — 79c4f1b

Figure 49: GIT (e.g. 6)



Commits on May 29, 2022

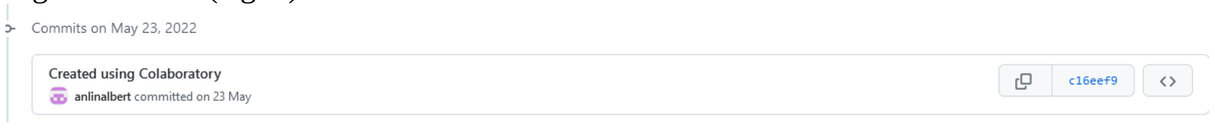
graphs updated
anlinalbert committed on 29 May — 469e03b

Commits on May 26, 2022

updated
anlinalbert committed on 26 May — a36ddfa

github project links
anlinalbert committed on 26 May — 12f46d2

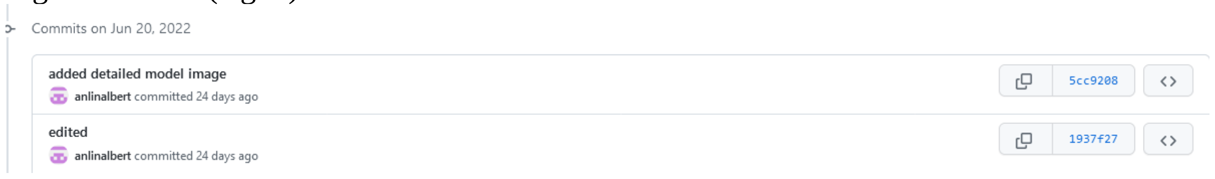
Figure 50: GIT (e.g. 7)



Commits on May 23, 2022

Created using Colaboratory
anlinalbert committed on 23 May — c16eef9

Figure 51: GIT (e.g. 8)



Commits on Jun 20, 2022

added detailed model image
anlinalbert committed 24 days ago — 5cc9208

edited
anlinalbert committed 24 days ago — 1937f27

8. Conclusion

Human activity recognition has broad applications in medical research and human survey system. In the project, we designed a smartphone-based recognition system that recognizes six human activities: walking, walking upstairs, walking downstairs, sitting, jogging, and standing. We used a combination of two datasets known as WISDM and UCI as input to the system which contains time-series signals which were recorded using an accelerometer. The activity data were trained and tested using LSTM and GRU architecture to reduce the feature dimensionality and improve the performance.

Out of all these architectures, LSTM was selected. This was because the LSTM network provided high performance and was more efficient than all the other architectures compared to it. The LSTM layers take full advantage of the temporal dependency to significantly improve the extraction features of HAR. The LSTM network was also evaluated by considering predictive accuracy and other performance metrics such as precision, recall, F1-score, and AUC.

The best classification rate in the experiment was 96.86% and the lowest loss of 0.1335 which is achieved by the LSTM architecture. Classification performance is robust to the orientation and the position of smartphones.

We’ve successfully built an LSTM model that can predict human activity with approx. 97% accuracy on the test set. The model was exported and used in an Android app. The app uses the text-to-speech Android API to tell you what the model predicts at some interval and includes our pre-trained model.

Future work may consider more activities and other query strategies such as variance reduction, and density-weighted methods may be investigated to enhance the performance of the model implemented here. Also, it could involve the further development of LSTM models using various hyperparameters, including regularization, learning rate, batch size, and others.

9. Future Work

There is still room for improvement in our work which can be addressed from two different perspectives.

i. By solving the current limitations of our proposed model
ii. By extending our achievements through complementary and novel applications

In the first case, some issues have arisen such as a limited number of activities the system can deal with and the adoption of novel approaches to deal with users with distinct differences in their motion patterns (people with walking difficulties) in the system. In the second case, new ideas about how to exploit HAR information to provide new services can be explored. These include the development of context-aware apps for health and sports monitoring, elderly care, and understanding interaction between users using similar systems. In this section, we focus on some of these aspects and propose them as future research dimensions.

- For a new user, the performance of the proposed HAR system can be improved if his motion data is integrated into the learned model. Although this can be done, for instance, through retraining after following a controlled sequence of activities, this process can be tedious for the user. But considering that during a normal day it is possible to gather large amounts of data, we can explore semi-supervised strategies which can allow combining this unlabelled data with already existing labeled trained data to produce considerable improvements to the system learning accuracy. This can bring advantages to new users, especially those with particular conditions such as very slow motion or physical disabilities which are normally difficult to incorporate into the training and the ML generalization capability is not sufficient to include them.

- We should explore if it is possible to place the device in different body parts such as a shirt or back pockets and even around the waist.

- Now that it is becoming increasingly popular the use of wearable devices such as smart watches with embedded inertial sensors, it is interesting to investigate how to combine them with the current smartphone-based system to improve recognition or to add new activities that involve upper limbs such as typing, brushing teeth, and writing.

- The outcome of the presented HAR system can be used in higher-level context-aware applications. For example, when combined with location-based services such as GPS or home presence sensors for achieving indoor and outdoor activity detection. Also, if activity information is merged with vital sign sensors, it is possible to develop apps for medical diagnostic and monitoring of ill patients during their daily life without third-party supervision. Likewise, it is possible to merge activity information with smartphone connectivity status such as incoming calls/messages, to control the smartphone behavior when some specific activities are occurring. On the other hand, we also propose the study of the interaction between two or more users earing the HAR recognition system to infer collective behavior such as chatting, dancing, playing sports, etc. In conclusion, there is a wide range of new possibilities and services that need to be explored where HAR can contribute to their development.

10. Appendix

10.1. Minimum Software Requirements

Operating System: Android

10.2. Minimum Hardware Requirements

Storage: 1GB
Ram: 2GB
Sensor: Accelerometer

11. References

1. Bulbul, E., Cetin, A., & Dogru, I. A. (2018). Human Activity Recognition Using Smartphones. 2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT).

2. Wang, H., Zhao, J., Li, J., Tian, L., Tu, P., Cao, T., … Li, S. (2020). Wearable Sensor-Based Human Activity Recognition Using Hybrid Deep Learning Techniques. Security and Communication Networks, 2020, 1–12.

3. Agarwal, P., & Alam, M. (2020). A Lightweight Deep Learning Model for Human Activity Recognition on Edge Devices. Procedia Computer Science, 167, 2364–2373.

4. Das, S., Partha, S. B., & Imtiaz Hasan, K. N. (2020). Sentence Generation using LSTM Based Deep Learning. 2020 IEEE Region 10 Symposium (TENSYMP).

5. Ullah, M., Ullah, H., Khan, S. D., & Cheikh, F. A. (2019). Stacked Lstm Network for Human Activity Recognition Using Smartphone Data. 2019 8th European Workshop on Visual Information Processing (EUVIP).

6. Sun, B., Liu, M., Zheng, R., & Zhang, S. (2019). Attention-based LSTM Network for Wearable Human Activity Recognition. 2019 Chinese Control Conference (CCC).

7. Alemayoh, T. T., Hoon Lee, J., & Okamoto, S. (2019). Deep Learning-Based Real-time Daily Human Activity Recognition and Its Implementation in a Smartphone. 2019 16th International Conference on Ubiquitous Robots (UR).

8. Deep, S., & Zheng, X. (2019). Hybrid Model Featuring CNN and LSTM Architecture for Human Activity Recognition on Smartphone Sensor Data. 2019 20th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT).

9. Brownlee, Jason. "How to use Learning Curves to Diagnose Machine Learning Model Performance". Machine Learning Mastery, February 27, 2019. https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/.