

A
Major Project
On
HUMAN ACTIVITY RECOGNITION
(Submitted in partial fulfillment of the requirements for the award of Degree)
BACHELOR OF TECHNOLOGY

In
COMPUTER SCIENCE AND ENGINEERING

By
M Mahitha Choudary (177R1A0539)
G Sri Vallika (177R1A0516)
G Abhignya (167R1A0529)

Under the Guidance of
NAJEEMA AFRIN
(Assistant Professor)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CMR TECHNICAL CAMPUS
UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by
AICTE, New Delhi)

Recognized Under Section 2(f) & 12(B) of the UGC Act.1956,
Kandlakoya (V), Medchal Road, Hyderabad-501401.

2017-21

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that the project entitled “Human Activity Recognition” being submitted by M Mahitha Choudary (177R1A0539), G Sri Vallika (177R1A0516) and G Abhignya (167R1A0529) in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by him/her under our guidance and supervision during the year 2020-21.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Najeema Afrin
Assistant Professor
INTERNAL GUIDE

Dr. A. Raji Reddy
DIRECTOR

Dr. K. Srujan Raju
HoD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on

ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project. We take this opportunity to express my profound gratitude and deep regard to my guide

Najeema Afrin, Assistant Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to Project Review Committee (PRC) Coordinators: **Mr. J. Narasimha Rao, Mr. Dr. Somashekar, Mr. K. Murali, Dr. Suwarna Gothane and Mr. B. Ramji** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to Dr. K. Srujan Raju, Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to Dr. A. Raji Reddy, Director for being cooperative throughout the course of this project. We would like to express our sincere gratitude to Sri. Ch. Gopal Reddy, Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of CMR Technical Campus who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

M Mahitha Choudary (177R1A0539)

G Sri Vallika (177R1A0516)

G Abhignya (167R1A0529)

ABSTRACT

Human activity recognition, or HAR for short, is a broad field of study concerned with identifying the specific movement or action of a person based on sensor data. The sensor data may be remotely recorded, such as video, radar, or other wireless methods. It contains data generated from accelerometer, gyroscope and other sensors of Smart phone to train supervised predictive models using machine learning techniques like SVM , Random forest and decision tree to generate a model. Which can be used to predict the kind of movement being carried out by the person which is divided into six categories walking, walking upstairs, walking down-stairs, sitting, standing and laying. MLM and SVM achieved accuracy of more than 99.2% in the original data set and 98.1% using new feature selection method. Results show that the proposed feature selection approach is a promising alternative to activity recognition on smartphones.

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
Figure 4.1	Project Architecture	19
Figure 4.2	Use case diagram	20
Figure 4.3	Class diagram	21
Figure 4.4	Sequence diagram	22
Figure 4.5	Activity diagram	23

LIST OF SCREENSHOTS

SCREENSHOT NO.	SCREENSHOT NAME	PAGE NO.
Screenshot 6.1	Data Sets	33
Screenshot 6.2	Training and Testing	33
Screenshot 6.3	LSTM	34
Screenshot 6.4	Epochs	34
Screenshot 6.5	Loss and Accuracy	35
Screenshot 6.6	Confusion Matrix	35

TABLE OF CONTENTS

ABSTRACT	i
LIST OF FIGURES	ii
LIST OF SCREENSHOTS	iii
1. INTRODUCTION	1
1.1 PROJECT SCOPE	1
1.2 PROJECT PURPOSE	1
1.3 PROJECT FEATURES	4
2. LITREATURE SURVEY	5
2.1 RELATED WORK	5
2.2 TECHNOLOGY	10
3. SYSTEM ANALYSIS	15
3.1 PROBLEM DEFINITION	15
3.2 EXISTING SYSTEM	15
3.2.1 LIMITATIONS OF THE EXISTING SYSTEM	16
3.3 PROPOSED SYSTEM	16
3.3.1 ADVANTAGES OF PROPOSED SYSTEM	16
3.4 FEASIBILITY STUDY	16
3.4.1 ECONOMIC FESIBILITY	17
3.4.2 TECHNICAL FEASIBILITY	17
3.4.3 SOCIAL FEASIBILITY	17
3.5 HARDWARE & SOFTWARE REQUIREMENTS	18
3.5.1 HARDWARE REQUIREMENTS	18
3.5.2 SOFTWARE REQUIREMENTS	18
4. ARCHITECTURE	19
4.1 PROJECT ARCHITECTURE	19
4.2 DESCRIPTION	19
4.3 USECASE DIAGRAM	20
4.4 CLASS DIAGRAM	21
4.5 SEQUENCE DIAGRAM	22
4.6 ACTIVITY DIAGRAM	23
5. IMPLEMENTATION	24
5.1 SAMPLE CODE	24

6. SCREENSHOTS	33
7. TESTING	37
7.1 INTRODUCTION TO TESTING	37
7.2 TYPES OF TESTING	39
7.2.1 UNIT TESTING	39
7.2.2 INTEGRATION TESTING	40
7.2.3 FUNCTIONAL TESTING	40
7.3 TEST CASES	41
7.3.1 UPLOADING DATA	41
7.3.2 CLASSIFICATION	42
8. CONCLUSION & FUTURE SCOPE	43
8.1 PROJECT CONCLUSION	43
8.2 FUTURE SCOPE	43
9. REFERENCES	44
9.1 REFERENCES	44
9.2 WEBSITES	45

1. INTRODUCTION

1. INTRODUCTION

1.1 PROJECT SCOPE

we can determine what is normal and what is abnormal activity for them therefore indicating whether they require attention from facility staff. Innovative approaches to recognize activities of daily living (ADL) is essential input part for development of more interactive human-computer applications. Methods for understanding Human Activity Recognition (HAR) are developed by interpreting attributes derived from motion, location, physiological signals and environmental information.

1.2 PROJECT PURPOSE

The purpose of being able to classify what activity a person is undergoing at a given time is to allow computers to help and guidance to a person prior to or while undertaking a task. The difficulty lies in how diverse our movements are as we perform our day-to-day tasks. There have been many attempts to use the various machine learning algorithms to accurately classify a person's activity, so much so that Google have created an Activity Recognition API for developers to embed into their creation of mobile applications.

1.3 PROJECT FEATURES

Human activity recognition (HAR) aims to classify a person's actions from a series of measurements captured by sensors. Nowadays, collecting this type of data is not a hard task. With the growth of the Internet of Things, almost everyone has some gadget that monitors their movements. It can be a smartwatch, a pulsometer, or even a smartphone. Usually, this is performed by following a fixed-length sliding window approach for the features extraction were two parameters have to be fixed: the size of the window and the shift.

These are some of the data you could use:

Body acceleration.

Gravity acceleration.

Body angular speed.

Body angular acceleration.

The machine learning model used for activity recognition is built on top of the devices' available sensors. However, due to the complexity of human activities and the existing differences between two individuals, analyzing this data can be a big challenge.

To recognize, detect and classify the activity of the human many applications have been developed with human centered monitoring and researchers have proposed different solutions. Human activity recognition is one of the important technology to monitor the dynamism of a person and this can be attained with the support of Machine learning techniques. Threshold-based algorithm is simpler and faster which is often applied to recognize the human activity. But Machine algorithm provides the reliable result. Numerous sensors have been deployed to observe the human dynamic characteristics. This paper intends to measure the effectiveness of various machine learning classification algorithms. Low cost and commercial smartphones are used as sensors to record the activities of the human. Different studies have been conducted in the intelligent environment to observe the activities of the human. We developed AI Models for "Human Activity Recognition using smartphones Data set" from UCI online storehouse. The motivation behind our work is to implement machine learning algorithms in real world datasets so that their accuracy can be studied and effective conclusions can be drawn.

Recognizing human activities by means of sensors attached on the body has been widely studied. Common activity and functional performance level of a person can be determined by the capability to record and identify distinct daily activities. In the health care field, initial detection of diseases could be obtained through long term inquiry of human activity or to stimulate people to be physically active and improve their exercise opportunities. As far as physiotherapy, it can help to comprehend if an exercise is been correctly accomplished or even to monitor possible disorders. Security and entertainment are others fields impacted by investigation of human behavior through mobile phone data. Mobile phones were used in police investigations to track suspects and victims. Regarding to entertainment, Badawi and Saddik, in [5], proposed a mobile application that provides users with physical activity recommendations on a daily basis. Motion capture video systems have become an important research topic in the monitor human activity. Dubois and Charpillet, in, proposed a method to detect

fall using a system made up of RGB-Depth cameras.

Human beings can realize others' psychological state and personality by observing their daily activities. Following this pattern, the researchers want to predict human behavior using machines, and Human Behavior Recognition (HAR) comes as an active research topic. This has become one of the important research topics in machine learning and computer vision. Though motion data collection was hard in previous days, current technological developments help researchers capture the data as they can now use portable devices, including smartphones, music players, smartwatches, or smart home sensors. Especially, motion sensor embedded smartphones, like accelerometers, gyroscope, etc. bring a new era for activity recognition.

HAR is the problem of classifying day-to-day human activity using data collected from smartphone sensors. Data are continuously generated from the accelerometer and gyroscope, and these data are instrumental in predicting our activities such as walking or standing. There are lots of datasets and ongoing research on this topic. In, the authors discuss wearable sensor data and related works of predictions with machine learning techniques. Wearable devices can predict an extensive range of activities using data from various sensors. Deep Learning models are also being used to predict various human activities. Nowadays, people use smartphones almost all the time and use many wearable devices. Through these devices, physical and mental health can be monitored by predicting human activity without specialized and costly medical equipment, and nowadays, it is an efficient, cheap, and safe way to do this as the COVID-19 (Coronavirus disease 2019) pandemic is ongoing.

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 (Weinland et al., 2011; Tran et al., 2012). 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. This remains a challenging research

topic largely due to the tremendous intra-class variation of human activities attributed to the visual appearance differences, subject motion variabilities, and viewpoint changes.

To solve these challenges, previous approaches in human activity recognition have focused on information about context. Context can be defined as information that is not directly related to the human activity itself, but it can be utilized to improve the traditional target-centered activity recognition (Wang and Ji, 2015). Lan et. al. (Lan et al., 2012) proposes action context to encode the human interactions among multiple people. Choi et. al. (Choi et al., 2011) uses spatio-temporal volume descriptor to capture nearby person actions.

Most existing approaches for human activity recognition mainly use people as context without richer context information, such as the scene information where the activity is performed, the location of person within the scene, etc.

2. LITREATURE SURVEY

2.LITERATURE SURVEY

2.1 RELATED WORK

Several investigations have considered the use of widely available mobile devices. Ravi et. al. collected data from only two users wearing a single accelerometer-based device and then transmitted this data to the phone carried by the user (Ravi et al.,2005). Lester et. al. used accelerometer data from a small set of users along with audio and barometric sensor data to recognize eight daily activities (Lester et al., 2006). However, the data was generated using distinct accelerometer-based devices worn by the user and then sent to the phone for storage.

Some studies took advantage of the sensors incorporated into the phones themselves. Yang developed an activity recognition system using a smartphone to distinguish between various activities (Yang, 2009). However, stair climbing was not considered, and their system was trained and tested using data from only four users. Brezmes et. al. developed a real-time system for recognizing six user activities (Brezmes et al., 2009). In their system, an activity recognition model is trained for each user, i.e., there is no universal model that can be applied to new users for whom no training data exists. Bayat et al. gathered acceleration data from only four participants, performing six activities. (Bayat et al., 2014) Shoaib et al. evaluated different classifiers by collecting data of smartphone accelerometer, gyroscope, and magnetometer for four subjects, performing six activities. (Shoaib et al., 2013).

In human activity recognition systems, various lowlevel features are introduced to describe the activity observation. Schuldt et. al. (Schuldt et al., 2004) proposed a local space-time feature to represent the human movement observed in a video, and integrated such representations with SVM classification schemes for recognition. Laptev et. al. (Laptev et al., 2008a) proposed space-time feature point (STIP) and spatio-temporal bag-of-features as the descriptor for human motion. Tran et. al. (Tran et al., 2012) presented a framework for human action recognition based on modeling the motion of human body parts. They utilized a descriptor that combines both local and global representations of human motion, encoding the motion information as well

as being robust to local appearance changes. The mentioned activity recognition methods mainly focus on recognizing the individual action. Their frameworks are difficult to scale to address real-world scenarios where multiple people activity and interaction are involved. Our approach represents the motion information using STIP feature similar to (Laptev et al., 2008a), but combines the rich context information that we extract from the video. By using the deep model, our method is able to: capture the extensive information about people motion and interactions; scale to recognize activity of each individual in the scene; and improve the accuracy of the overall activity recognition task. Context based Activity Recognition. Context information is widely utilized in many video analysis applications (Wei and Shah, 2015; Wei and Shah, 2016).

In the topic of human activity recognition, many approaches integrate contextual information by proposing new feature descriptors extracted from an individual and its surrounding area. Lan et. al. (Lan et al., 2012) proposed Action Context (AC) descriptor capturing the activity of the focal person and the behavior of other persons nearby. The AC descriptor is concatenating the focal person action probability vector with context action vectors that captures the nearby people action. Choi et. al. (Choi et al., 2009) propose Spatio-Temporal Volume (STV) descriptor, which captures spatial distribution of pose and motion of individuals in the scene to analyze group activity. STV descriptor centered on a person of interest is used to classify centered person's group activity. SVM with pyramid kernel is used for classification. The same descriptor is leveraged in (Choi et al., 2011), however, the random forest classification is used for group activity analysis. In (Lan et al., 2012; Choi et al., 2009; Choi et al., 2011), the nearby person that serves as context are selected according to the distance to the centered target. This does not necessarily ensure the existence of interactions among the selected persons. To address this issue, Tran et. al. (Tran et al., 2015) proposed group context activity descriptor similar to (Lan et al., 2012), but the people are first clustered into groups by modeling the social interaction among the persons. However, due to the noisy observation in videos, the group detection might not be robust or stable. Therefore, our approach utilizes the social interaction region to select the contextual people without a clustering process. Besides focusing on people as context, our approach also introduces scene information as

context for the first time. The scene context describes the environment around the center target at the local and global levels. We utilize the existing place recognition method (Zhou et al., 2014) to provide scene context features that have semantic meanings.

Deep Model for Activity Recognition. In recent years, deep models including deep neural networks, convolution neural networks, and auto-encoders have been used in many applications. For human activity recognition (Ji et al., 2013; Karpathy et al., 2014), convolution neural networks and auto-encoder approaches (Hasan and Roy-Chowdhury, 2014) have been developed. However, these action/activity deep models are generally target-centered and do not consider any context information, which is important for human activity that involves multiple people. Comparatively, Wang et al. (Wang and Ji, 2015) proposed an event recognition framework, which is a hierarchical context model that captures the context information in multiple levels. Similarly, our approach uses a deep-structure model that be trained using the contextual information extracted from human groups and video scene. However, our approach focuses on a different problem, which is recognizing the activity of each individual appearing in the scene, other than an overall event of the entire scene (Wang and Ji, 2015).

Since HAR becomes an important research topic, the researchers use different Machine Learning methods, such as Naive Bayes, Logistic Regression, Decision Trees, Support Vector Machines, Nearest Neighbor, Hidden Markov Model, CNN, RNN (Recurrent Neural Network), etc. for recognizing human activities. D. Singh et al. used RNN in their work for human activity recognition. They use smart home sensors to collect data and applied a Long Short Term Memory (LSTM) RNN. ANN is used for prototyping data acquisition module [11]. A Anjum et al. [12] use smartphone sensors over detectable physical activities, including walking, running, climbing, cycling, driving, etc. They compare Naive Bayes, Decision Tree, KNN (K Nearest Neighbours), and SVM to calculate accuracy. They achieve a greater than 95% true positive rate and less than 1.5% false positives rate.

Z Chen et al. [13] propose a robust HAR system based on coordinate transformation and PCA (CT-Principal Component Analysis) and online SVM. CT-

PCA scheme is used to reduce the effect of orientation variations. Their presented OSVM is independent, which uses a tiny portion of data from the unseen placement. Sun et al. [14] propose a HAR approach with varying position and orientation through smartphone accelerometer data. They use their acceleration magnitude as the fourth data dimension. They also use generic SVM and location-specific SVM in the model.

Deep Learning approaches require a considerable amount of training data; therefore, many researchers mainly use generic machine learning approaches [15]. Hence, in our paper, we focus on the generic approaches to analyze and detect human activity.

An viable alternative to video systems is motion sensors. They are small, have low cost, and are able to record motion signals within wearable and unobtrusive systems. Accelerometers and gyroscopes, the two major motion sensors, were used in research for fall detection [19], analysis of energy expenditure in physical activities [8] and daily activity monitoring [9]. Chen et al., in [7], used information of the motion sensor signals to identify the exercise type and classify if their postures are proper or not. Olivares et al., in [3], proposed an approach of inertial magnitude-based detectors and compares with well known others. Bayat et al., in [6], showed how to identify some human physical activities using a smartphone with accelerometer. In [3], Shoaib et al. compared the effect of accelerometer, gyroscope and linear acceleration sensor at both wrist and pocket positions using smartphone and wrist-worn motion sensors. Essentially, activity recognition requires a math model which enables the identification of interest. In general, models can be created using the knowledge of the specialist about the phenomenon which needs shape or by techniques of machine learning (statistical and neural). A machine learning model, among other benefits, allows us build models based on with little or no prior knowledge about the task of interest. In this regard, examples of pattern should be captured or available [7]. The aim of this paper is to evaluate a new approach of feature selection along with the PCA technique and compare the performance of several machine learning techniques for activity monitoring. The learning algorithms used in the comparison are k-Nearest Neighbors (kNN), Artificial Neural Network Multilayer Perceptron (MLP), Support Vector Machines (SVM), Bayes classifier, Minimal Learning Machine (MLM) and Minimal Learning Machine

with Near-est Neighbors (MLM-NN). For the recently proposed Minimal Learning Machine, we evaluate the impact of using different distance metrics. We propose a simple and effective approach of feature selection. Our method significantly reduces the number of features and therefore increases the speed of sensing over the human activity recognition task. With this approach the time step of the acquisition and classification is decreased, then the final application, which consists exclusively of these two steps, will become much faster in general.

2 Machine Learning methods

In this section, the six machine learning techniques used in our comparison are defined. Minimal Learning Machine (MLM) is a method for supervised learning which assumes a mapping between points in the input and output space.

Shahroudy et al. (2016) examined late methodologies top to bottom based human movement examination accomplished exceptional execution and demonstrated the viability of 3D portrayal for arrangement of activity classes. As of now accessible profundity based and RGB+D-based activity acknowledgment benchmarks have various restrictions, including the absence of preparing tests, unmistakable class names, camera perspectives and assortment of subjects. In this paper presented a vast scale dataset for human activity acknowledgment of 56,000 video tests and 4 million edges, gathered from 40 particular subjects. It contains sixty diverse activity classes including day by day, shared, and wellbeing related activities.

Moreover, another repetitive neural system structure is proposed to demonstrate the long haul transient connection of the highlights for all the body parts, and use them for proper activity characterization. Finally demonstrated the benefits of applying profound learning strategies over cutting edge hand that includes cross-subject and cross assessment criteria for the chosen dataset. [2]

Oyelade et al. (2010) considered the capacity to screen the advancement of understudies' scholarly execution is a basic issue to the scholastic network of higher learning. A framework for breaking down understudy's outcomes dependent on group examination and utilizations standard measurable calculations to mastermind their scores information as indicated by the dimension of their execution is depicted. In order to assess the academic performance of the students, cluster analysis and standard

statistical algorithms are used to the considered student dataset containing student score of one particular semester. The number of clusters to be obtained is given as input for the chosen random samples.

Anguita et al. (2012) demonstrated how activities of human are recognized by exploiting dissimilar sensors so as to give adjustment to exogenous registering assets. At the point when these sensors are joined to the subject's body, they license nonstop checking of various physiological signs. He has presented a framework to recognize human physical activities using inertial navigation system. As these cell phones are constrained as far as vitality and processing power, equipment benevolent methodology is proposed for multiclass characterization. This strategy adjusts the standard Support Vector Machine (SVM) and endeavors fixed-point number juggling for computational cost decrease. An examination with the customary SVM demonstrates a noteworthy improvement as far as computational expenses while keeping up comparative precision, which can add to grow increasingly maintainable frameworks for AmI.[1]

2.2 Python

Python is a programming language, which means it's a language both people and computers can understand. Python was developed by a Dutch software engineer named Guido van Rossum, who created the language to solve some problems he saw in computer languages of the time. 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, and a syntax that allows programmers to express concepts in fewer lines of code, 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. C Python, the reference

implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. C Python is managed by the non-profit Python Software Foundation.

One significant advantage of learning Python is that it's a general-purpose language that can be applied in a large variety of projects. Below are just some of the most common fields where Python has found its use:

- Data science
- Scientific and mathematical computing
- Web development
- Computer graphics
- Basic game development
- Mapping and geography (GIS software)

Python's ecosystem is growing over the years and it's more and more capable of the statistical analysis. It's the best compromise between scale and sophistication (in terms of data processing). Python emphasizes productivity and readability. Python is used by programmers that want to delve into data analysis or apply statistical techniques (and by devs that turn to data science) There are plenty of Python scientific packages for data visualization, machine learning, natural language processing, complex data analysis and more. All of these factors make Python a great tool for scientific computing and a solid alternative for commercial packages such as MatLab. The most popular libraries and tools for data science are:

Pandas: a library for data manipulation and analysis. The library provides data structures and operations for manipulating numerical tables and time series.

NumPy: the fundamental package for scientific computing with Python, adding support for large, multi-dimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays.

SciPy: a library used by scientists, analysts, and engineers doing scientific computing and technical computing.

Being a free, cross-platform, general-purpose and high-level programming language, Python has been widely adopted by the scientific community. Scientists value Python for its precise and efficient syntax, relatively flat learning curve and the fact that it

integrates well with other languages (e.g. C/C++). As a result of this popularity there are plenty of Python scientific packages for data visualization, machine learning, natural language processing, complex data analysis and more. All of these factors make Python a great tool for scientific computing and a solid alternative for commercial packages such as MatLab.

Matplotlib : Matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib allows you to generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, and more.

NumPy : NumPy is the fundamental package for scientific computing with Python, adding support for large, multi-dimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays.

NetworkX : NetworkX is a library for studying graphs which helps you create, manipulate, and study the structure, dynamics, and functions of complex networks.

TomoPy : TomoPy is an open-sourced Python toolbox to perform tomographic data processing and image reconstruction tasks. TomoPy provides a collaborative framework for the analysis of synchrotron tomographic data with the goal to unify the effort of different facilities and beamlines performing similar tasks.

Theano : Theano is a numerical computation Python library. Theano allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.

SymPy : SymPy is a library for symbolic computation and includes features ranging from basic symbolic arithmetic to calculus, algebra, discrete mathematics and quantum physics. It provides computer algebra capabilities either as a standalone application, as a library to other applications, or live on the web.

SciPy : SciPy is a library used by scientists, analysts, and engineers doing scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

Scikit-learn : Scikit-learn is a machine learning library. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Scikit-image : Scikit-image is a image processing library. It includes algorithms for segmentation, geometric transformations, color space manipulation, analysis, filtering, morphology, feature detection, and more.

ScientificPython : ScientificPython is a collection of modules for scientific computing. It contains support for geometry, mathematical functions, statistics, physical units, IO, visualization, and parallelization.

SageMath : SageMath is mathematical software with features covering many aspects of mathematics, including algebra, combinatorics, numerical mathematics, number theory, and calculus. SageMath uses the Python, supporting procedural, functional and object-oriented constructs.

Veusz : Veusz is a scientific plotting and graphing package designed to produce publication-quality plots in popular vector formats, including PDF, PostScript and SVG.

Graph-tool : Graph-tool is a module for the manipulation and statistical analysis of graphs.

SunPy : SunPy is a data-analysis environment specializing in providing the software necessary to analyze solar and heliospheric data in Python.

Bokeh : Bokeh is a Python interactive visualization library that targets modern web browsers for presentation. Bokeh can help anyone who would like to quickly and easily create interactive plots, dashboards, and data applications. Its goal is to provide elegant, concise construction of novel graphics in the style of D3.js, but also deliver this capability with high-performance interactivity over very large or streaming datasets.

TensorFlow : TensorFlow is an open source software library for machine learning across a range of tasks, developed by Google to meet their needs for systems capable of building and training neural networks to detect and decipher patterns and correlations, analogous to the learning and reasoning which humans use. It is currently used for both research and production at Google products, often replacing the role of its closed-source predecessor, DistBelief.

Nilearn : Nilearn is a Python module for fast and easy statistical learning on NeuroImaging data. Nilearn makes it easy to use many advanced machine learning, pattern recognition and multivariate statistical techniques on neuroimaging data for applications such as MVPA (Mutli-Voxel Pattern Analysis), decoding, predictive modelling, functional connectivity, brain parcellations, connectomes.

Dmelt : DataMelt, or DMelt, is a software for numeric computation, statistics, analysis of large data volumes ("big data") and scientific visualization. The program can be used in many areas, such as natural sciences, engineering, modeling and analysis of financial markets. DMelt can be used with several scripting languages including Python/Jython, BeanShell, Groovy, Ruby, as well as with Java.

Python-weka-wrapper : Weka is a suite of machine learning software written in Java, developed at the University of Waikato, New Zealand. It contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to these functions. The python-weka-wrapper package makes it easy to run Weka algorithms and filters from within Python.

Dask : Dask is a flexible parallel computing library for analytic computing composed of two components: 1) dynamic task scheduling optimized for computation, optimized for interactive computational workloads, and 2) Big Data collections like parallel arrays, dataframes, and lists that extend common interfaces like NumPy, Pandas, or Python iterators to larger-than-memory or distributed environments.

Python Saves Time : Even the classic “Hello, world” program illustrates this point:

```
print("Hello, world")
```

For comparison, this is what the same program looks like in Java:

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world");
    }
}
```


3. SYSTEM ANALYSIS

3.SYSTEM ANALYSIS

SYSTEM ANALYSIS

The prediction, analysis and cause of flight delays have been a major problem for air traffic control, decision making by airlines and ground delay response programs. Studies are conducted on the delay propagation of the sequence. Also, studying the predictive model of arrival delay and departure delay with meteorological features is encouraged.

3.1 PROBLEM DEFINITION

The purpose of being able to classify what activity a person is undergoing at a given time is to allow computers to help and guidance to a person prior to or while undertaking a task. The difficulty lies in how diverse our movements are as we perform our day-to-day tasks.

There have been many attempts to use the various machine learning algorithms to accurately classify a person's activity, so much so that Google have created an Activity Recognition API for developers to embed into their creation of mobile applications.

3.2 EXISTING SYSTEM

Several investigations have considered the use of widely available mobile devices. Ravi et. al. collected data from only two users wearing a single accelerometer-based device and then transmitted this data to the phone carried by the user (Ravi et al.,2005). Lester et. al. used accelerometer data from a small set of users along with audio and barometric sensor data to recognize eight daily activities (Lester et al., 2006). However, the data was generated using distinct accelerometer-based devices worn by the user and then sent to the phone for storage.

Some studies took advantage of the sensors incorporated into the phones themselves. Yang developed an activity recognition system using a smart-phone to distinguish between various activities (Yang, 2009). However, stair climbing was not considered and their system was trained and tested using data from only four users.

3.2.1 LIMITATIONS OF EXISTING SYSTEM

- Data set if heavy data mining concepts doeskin have a goof scopes for the analysis.
- Prediction model always wont let us the accurate values.
- No proper resolution of how the accidents delay has been happening.
- Impacts of the delay on next location and time schedules

3.3 PROPOSED SYSTEM

The purpose of being able to classify what activity a person is undergoing at a given time is to allow computers to provide assistance and guidance to a person prior to or while undertaking a task. The difficulty lies in how diverse our movements are as we perform our day-to-day tasks. There have been many attempts to use the various machine learning algorithms to accurately classify a person's activity, so much so that Google have created an Activity Recognition API for developers to embed into their creation of mobile applications.

3.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

- This model helps in predicting the delays of the flight under different circumstances like weather and technical problems.
- In this model analysis helps us to overcome other flight delays and management issues.
- Prevention or avoidance of accidents and can know the major accident prone areas.
- Accident prediction final result is to find the percentage of accident in particular area

3.4 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. Three key considerations involved in the feasibility analysis are

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

3.4.1 ECONOMIC FEASIBILITY

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during preliminary investigation:

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. Also all the resources are already available, it give an indication of the system is economically possible for development.

3.4.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.4.3 BEHAVIORAL FEASIBILITY

This includes the following questions:

- Is there sufficient support for the users?
- Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioral aspects are considered carefully and conclude that the project is behaviorally feasible.

3.5 HARDWARE & SOFTWARE REQUIREMENTS

3.5.1 HARDWARE REQUIREMENTS:

Hardware interfaces specifies the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

- Processor : Intel Dual Core@ CPU 2.90GHz.
- Hard disk : 500GB and Above.
- RAM : 4GB and Above.
- Monitor : 5 inches or above.

3.5.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements,

- Operating system : Windows 8, 10
- Languages : Python
- Backend : Mysql
- IDE : Jupyter
- Front End : HTML

4. ARCHITECTURE

4. ARCHITECTURE

4.1 PROJECT ARCHITECTURE

This project architecture shows the procedure followed for flight delay using machine learning, starting from input to final prediction.

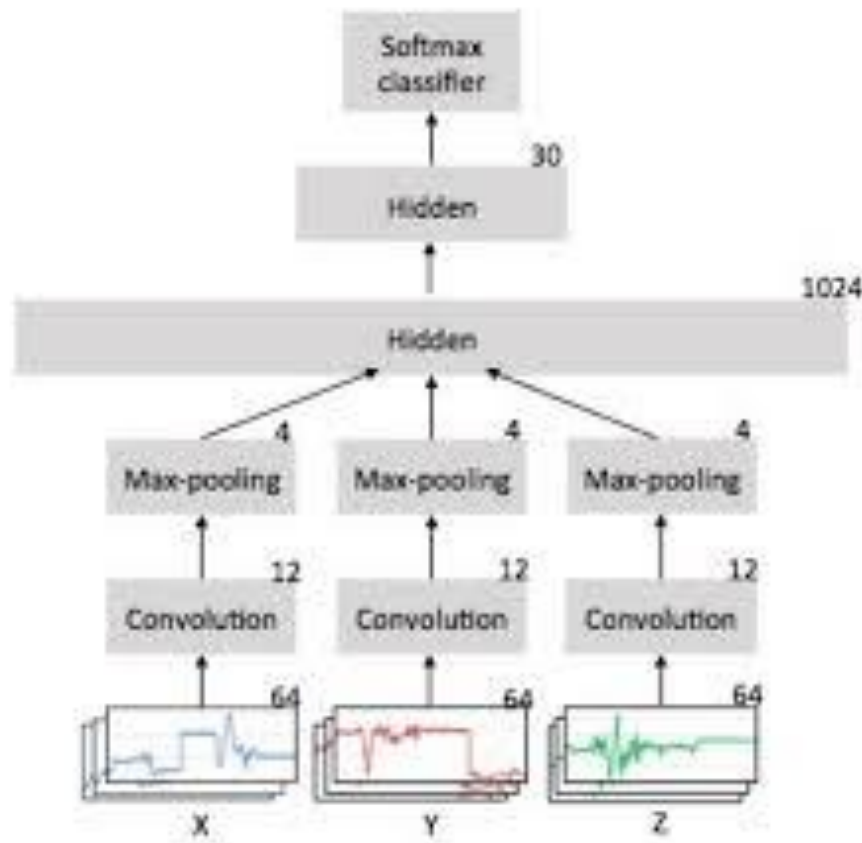


Figure 4.1: Project Architecture

4.2 DESCRIPTION

Input Data: Input data is generally in csv format where the data is fetched and mapped in the data framed from the source columns.

Reading Data: pandas sklearn library is used to read the data into the data frame.

Separating Features: In this following step we are going to separate the features which

we take to train the model by giving the target value i.e. 1/0 for the particular of features.

Normalization: Normalization is a very important step while we are dealing with the large values in the features as the higher bit integers will cost high computational power and time. To achieve the efficiency in computation we are going to normalize the data values.

Training and test data: Training data is passed to the MLP classifier to train the model. Test data is used to test the trained model whether it is making correct predictions or not.

4.3 USE CASE DIAGRAM

The use case graph is for demonstrating the direct of the structure. This chart contains the course of action of use cases, performing pros and their relationship. This chart might be utilized to address the static perspective of the structure.

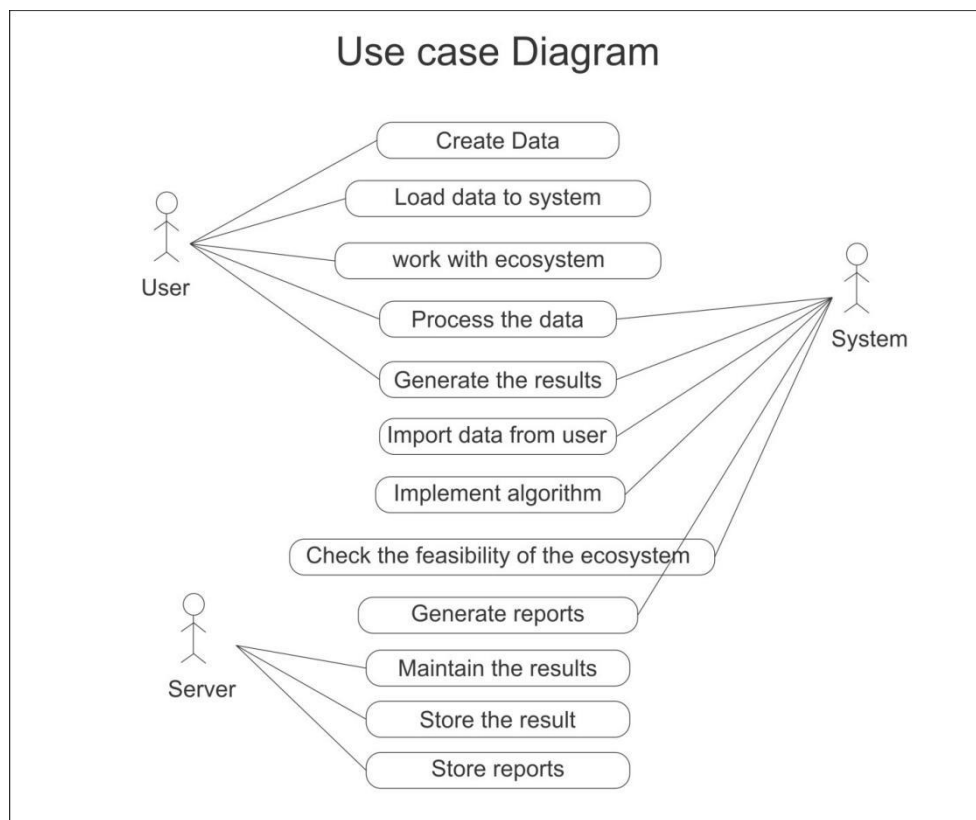


Figure 4.2: Use Case Diagram

4.4 CLASS DIAGRAM

The class graph is the most normally pulled in layout UML. It addresses the static course of action perspective of the structure. It solidifies the strategy of classes, interfaces, joint attempts and their affiliations.

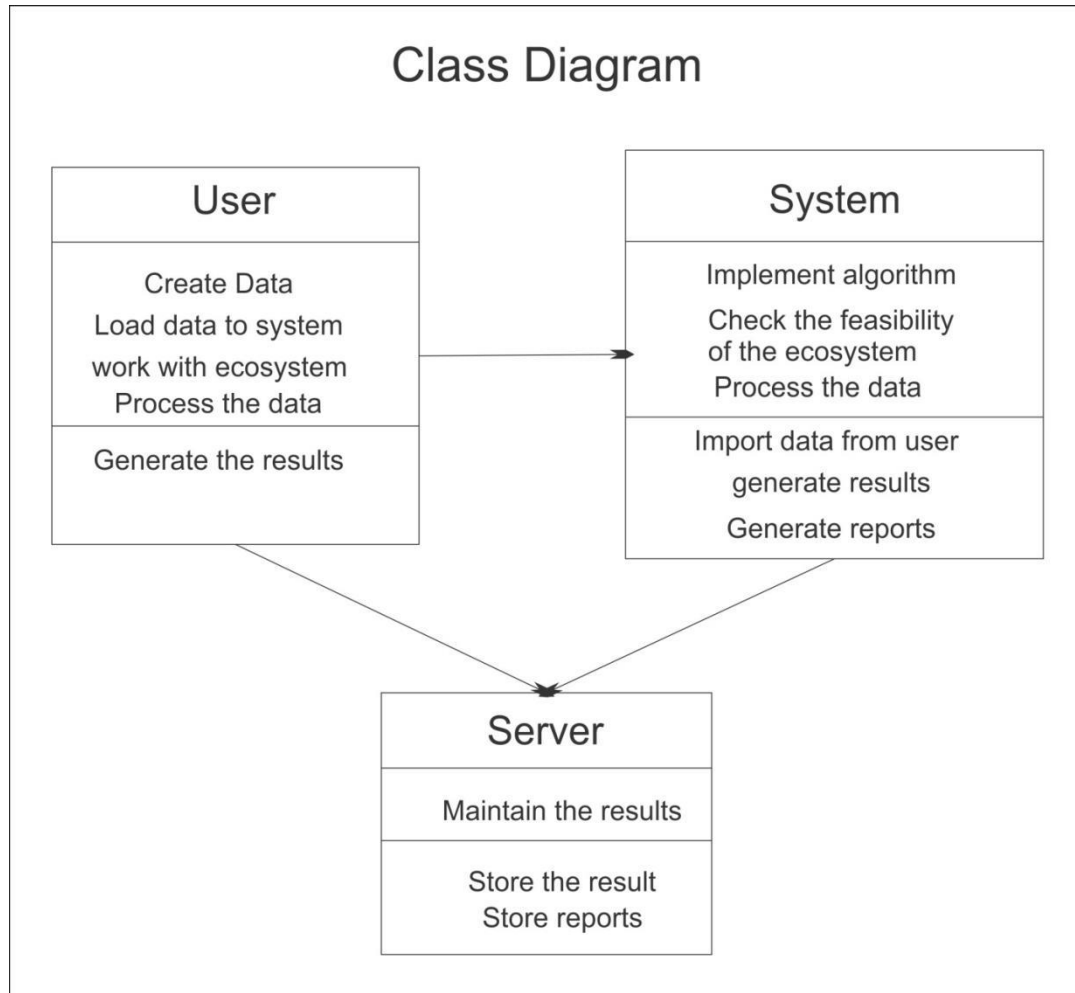


Figure 4.3: Class Diagram

4.5 SEQUENCE DIAGRAM

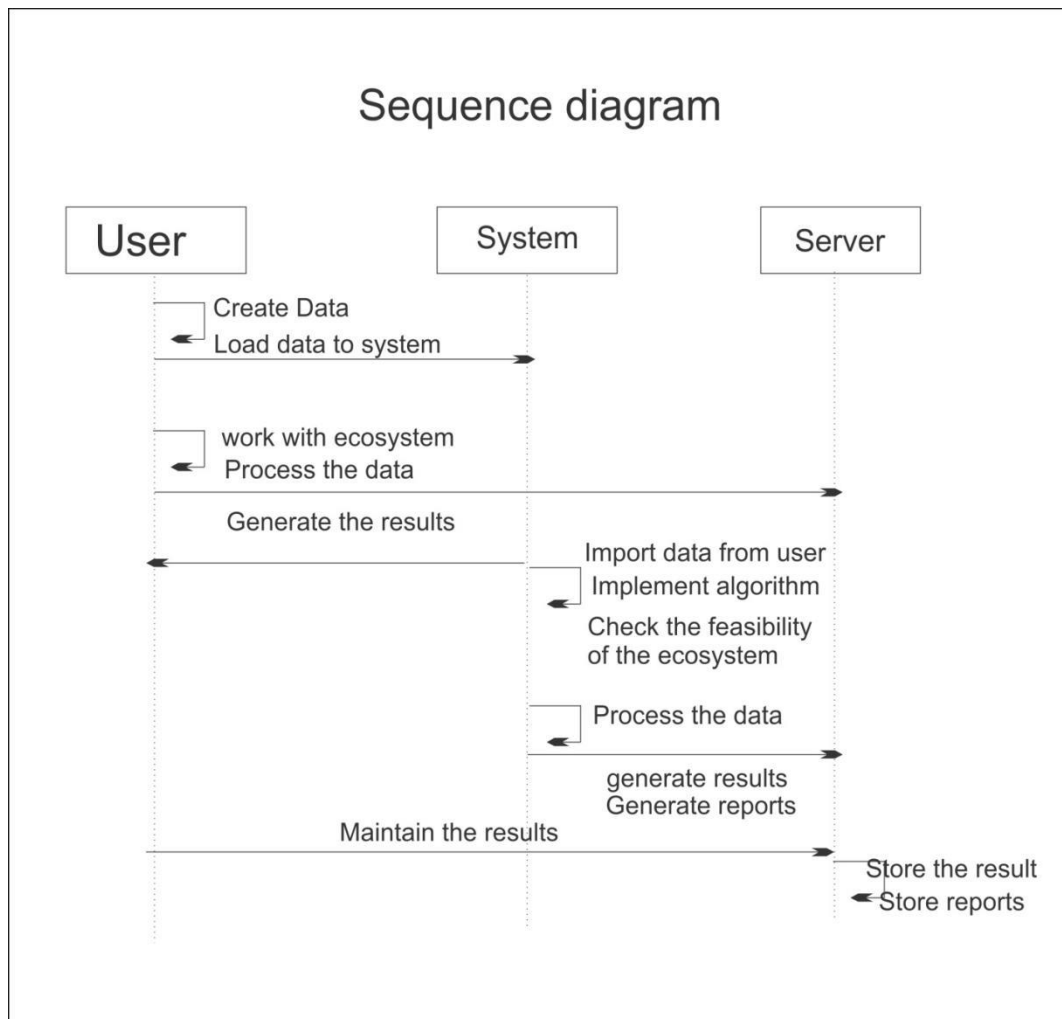


Figure 4.4: Sequence Diagram

4.6 ACTIVITY DIAGRAM

It describes about flow of activity states.

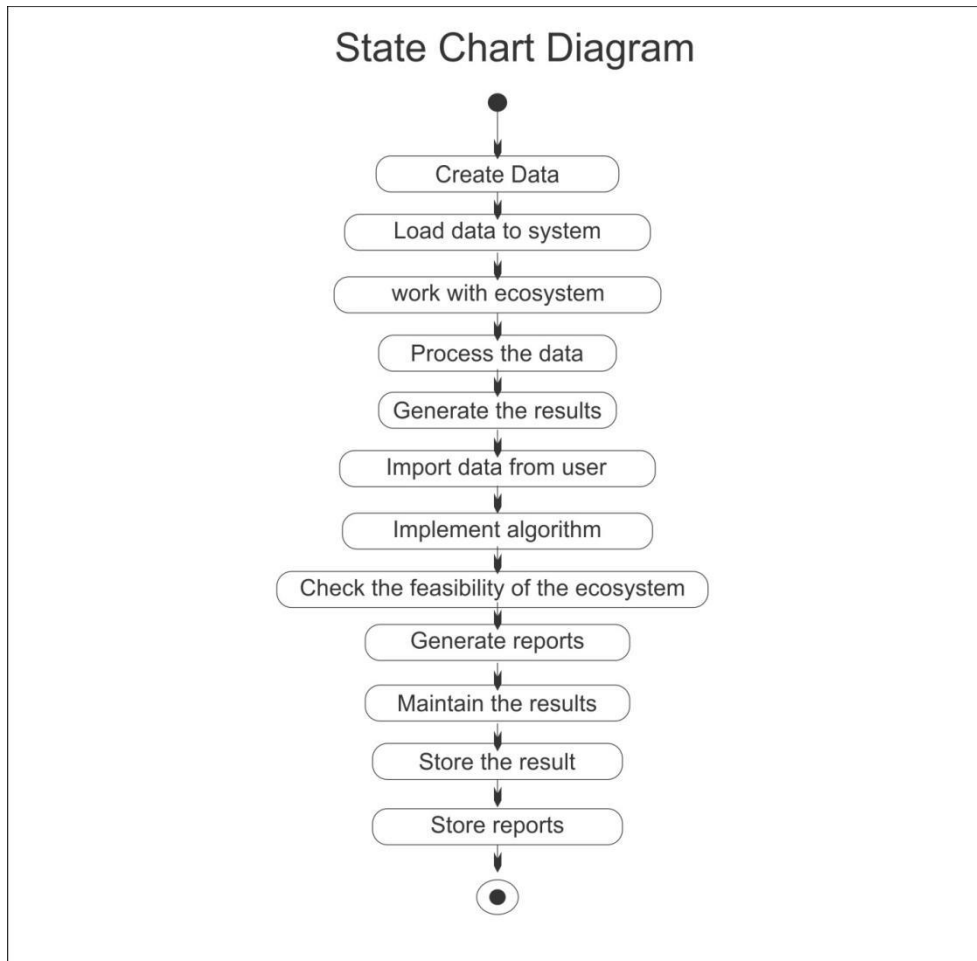


Figure 4.5: Activity Diagram

5. IMPLEMENTATION

5.IMPLEMENTATION

5.1 SAMPLE CODE

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import tensorflow as tf # Version 1.0.0 (some previous versions are used in past
commits)
from sklearn import metrics

import os

# Useful Constants

# Those are separate normalised input features for the neural network
INPUT_SIGNAL_TYPES = [
    "body_acc_x_",
    "body_acc_y_",
    "body_acc_z_",
    "body_gyro_x_",
    "body_gyro_y_",
    "body_gyro_z_",
    "total_acc_x_",
    "total_acc_y_",
    "total_acc_z_"
]

# Output classes to learn how to classify
LABELS = [
    "WALKING",
    "WALKING_UPSTAIRS",
    "WALKING_DOWNSTAIRS",
```

```

        "SITTING",
        "STANDING",
        "LAYING"
    ]
    DATA_PATH = "data/"

    !pwd && ls
    os.chdir(DATA_PATH)
    !pwd && ls

    !python download_dataset.py

    !pwd && ls
    os.chdir("..")
    !pwd && ls

    DATASET_PATH = DATA_PATH + "UCI HAR Dataset/"
    print("\n" + "Dataset is now located at: " + DATASET_PATH)

    TRAIN = "train/"
    TEST = "test/"

    # Load "X" (the neural network's training and testing inputs)

    def load_X(X_signals_paths):
        X_signals = []

        for signal_type_path in X_signals_paths:
            file = open(signal_type_path, 'r')
            # Read dataset from disk, dealing with text files' syntax
            X_signals.append(

```

```

        [np.array(serie, dtype=np.float32) for serie in [
            row.replace(' ', ' ').strip().split(' ') for row in file
        ]]
    )
    file.close()

    return np.transpose(np.array(X_signals), (1, 2, 0))
X_train_signals_paths = [
    DATASET_PATH + TRAIN + "Inertial Signals/" + signal + "train.txt" for signal in
INPUT_SIGNAL_TYPES
]
X_test_signals_paths = [
    DATASET_PATH + TEST + "Inertial Signals/" + signal + "test.txt" for signal in
INPUT_SIGNAL_TYPES
]

X_train = load_X(X_train_signals_paths)
X_test = load_X(X_test_signals_paths)

# Load "y" (the neural network's training and testing outputs)

def load_y(y_path):
    file = open(y_path, 'r')
    # Read dataset from disk, dealing with text file's syntax
    y_ = np.array(
        [elem for elem in [
            row.replace(' ', ' ').strip().split(' ') for row in file
        ]],
        dtype=np.int32
    )
    file.close()

```

```
# Subtract 1 to each output class for friendly 0-based indexing
return y_ - 1

y_train_path = DATASET_PATH + TRAIN + "y_train.txt"
y_test_path = DATASET_PATH + TEST + "y_test.txt"

y_train = load_y(y_train_path)
y_test = load_y(y_test_path)

# Input Data

training_data_count = len(X_train) # 7352 training series (with 50% overlap between
each serie)
test_data_count = len(X_test) # 2947 testing series
n_steps = len(X_train[0]) # 128 timesteps per series
n_input = len(X_train[0][0]) # 9 input parameters per timestep

# LSTM Neural Network's internal structure

n_hidden = 32 # Hidden layer num of features
n_classes = 6 # Total classes (should go up, or should go down)

# Training

learning_rate = 0.0025
lambda_loss_amount = 0.0015
training_iters = training_data_count * 300 # Loop 300 times on the dataset
batch_size = 1500
display_iter = 30000 # To show test set accuracy during training
```



```
# Some debugging info
```

```
print("Some useful info to get an insight on dataset's shape and normalisation:")
print("(X shape, y shape, every X's mean, every X's standard deviation)")
print(X_test.shape, y_test.shape, np.mean(X_test), np.std(X_test))
print("The dataset is therefore properly normalised, as expected, but not yet one-hot
encoded.")
```

```
def LSTM_RNN(_X, _weights, _biases):
```

```
    # Function returns a tensorflow LSTM (RNN) artificial neural network from given
    parameters.
```

```
    # Moreover, two LSTM cells are stacked which adds deepness to the neural
    network.
```

```
    # Note, some code of this notebook is inspired from an slightly different
```

```
    # RNN architecture used on another dataset, some of the credits goes to
```

```
    # "aymericdamien" under the MIT license.
```

```
    # (NOTE: This step could be greatly optimised by shaping the dataset once
```

```
    # input shape: (batch_size, n_steps, n_input)
```

```
    _X = tf.transpose(_X, [1, 0, 2]) # permute n_steps and batch_size
```

```
    # Reshape to prepare input to hidden activation
```

```
    _X = tf.reshape(_X, [-1, n_input])
```

```
    # new shape: (n_steps*batch_size, n_input)
```

```
    # ReLU activation, thanks to Yu Zhao for adding this improvement here:
```

```
    _X = tf.nn.relu(tf.matmul(_X, _weights['hidden']) + _biases['hidden'])
```

```
    # Split data because rnn cell needs a list of inputs for the RNN inner loop
```

```
    _X = tf.split(_X, n_steps, 0)
```

```
    # new shape: n_steps * (batch_size, n_hidden)
```

```
    # Define two stacked LSTM cells (two recurrent layers deep) with tensorflow
```

```
    lstm_cell_1 = tf.contrib.rnn.BasicLSTMCell(n_hidden, forget_bias=1.0,
```

```

state_is_tuple=True)

lstm_cell_2 = tf.contrib.rnn.BasicLSTMCell(n_hidden, forget_bias=1.0,
state_is_tuple=True)

lstm_cells = tf.contrib.rnn.MultiRNNCell([lstm_cell_1, lstm_cell_2],
state_is_tuple=True)

# Get LSTM cell output
outputs, states = tf.contrib.rnn.static_rnn(lstm_cells, _X, dtype=tf.float32)

# Get last time step's output feature for a "many-to-one" style classifier,
# as in the image describing RNNs at the top of this page
lstm_last_output = outputs[-1]

# Linear activation
return tf.matmul(lstm_last_output, _weights['out']) + _biases['out']

def extract_batch_size(_train, step, batch_size):
    # Function to fetch a "batch_size" amount of data from "(X|y)_train" data.

    shape = list(_train.shape)
    shape[0] = batch_size
    batch_s = np.empty(shape)

    for i in range(batch_size):
        # Loop index
        index = ((step-1)*batch_size + i) % len(_train)
        batch_s[i] = _train[index]

    return batch_s

def one_hot(y_, n_classes=n_classes):
    # Function to encode neural one-hot output labels from number indexes

```

```

# e.g.:
# one_hot(y_=[[5], [0], [3]], n_classes=6):
#     return [[0, 0, 0, 0, 0, 1], [1, 0, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0]]

y_ = y_.reshape(len(y_))
return np.eye(n_classes)[np.array(y_, dtype=np.int32)] # Returns FLOATS


# Graph input/output
x = tf.placeholder(tf.float32, [None, n_steps, n_input])
y = tf.placeholder(tf.float32, [None, n_classes])


# Graph weights
weights = {
    'hidden': tf.Variable(tf.random_normal([n_input, n_hidden])), # Hidden layer
weights
    'out': tf.Variable(tf.random_normal([n_hidden, n_classes], mean=1.0))
}
biases = {
    'hidden': tf.Variable(tf.random_normal([n_hidden])),
    'out': tf.Variable(tf.random_normal([n_classes]))
}
pred = LSTM_RNN(x, weights, biases)


# Loss, optimizer and evaluation
l2 = lambda_loss_amount * sum(
    tf.nn.l2_loss(tf_var) for tf_var in tf.trainable_variables()
) # L2 loss prevents this overkill neural network to overfit the data
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=y,
logits=pred)) + l2 # Softmax loss
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost) #
Adam Optimizer

```

```

correct_pred = tf.equal(tf.argmax(pred,1), tf.argmax(y,1))
accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))

# To keep track of training's performance
test_losses = []
test_accuracies = []
train_losses = []
train_accuracies = []

# Launch the graph
sess = tf.InteractiveSession(config=tf.ConfigProto(log_device_placement=True))
init = tf.global_variables_initializer()
sess.run(init)

# Perform Training steps with "batch_size" amount of example data at each loop
step = 1
while step * batch_size <= training_iters:
    batch_xs =      extract_batch_size(X_train, step, batch_size)
    batch_ys = one_hot(extract_batch_size(y_train, step, batch_size))

    # Fit training using batch data
    _, loss, acc = sess.run(
        [optimizer, cost, accuracy],
        feed_dict={
            x: batch_xs,
            y: batch_ys
        }
    )
    train_losses.append(loss)
    train_accuracies.append(acc)
test_losses.append(final_loss)

```

```

test_accuracies.append(accuracy)

print("FINAL RESULT: " + \
      "Batch Loss = {}".format(final_loss) + \
      ", Accuracy = {}".format(accuracy))

# (Inline plots: )
%matplotlib inline

font = {
    'family' : 'Bitstream Vera Sans',
    'weight' : 'bold',
    'size' : 18
}
matplotlib.rc('font', **font)

width = 12
height = 12
plt.figure(figsize=(width, height))

indep_train_axis = np.array(range(batch_size, (len(train_losses)+1)*batch_size,
batch_size))

plt.plot(indep_train_axis, np.array(train_losses), "b--", label="Train losses")
plt.plot(indep_train_axis, np.array(train_accuracies), "g--", label="Train accuracies")

indep_test_axis = np.append(
    np.array(range(batch_size, len(test_losses)*display_iter, display_iter)[: -1]),
    [training_iters]
)

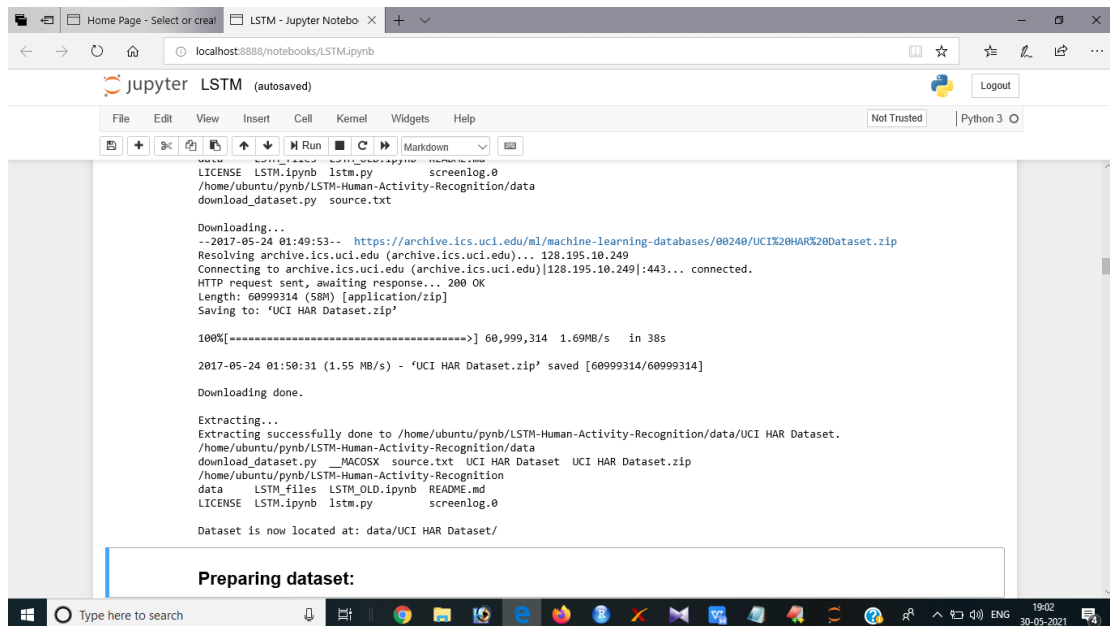
plt.plot(indep_test_axis, np.array(test_losses), "b-", label="Test losses")
plt.plot(indep_test_axis, np.array(test_accuracies), "g-", label="Test accuracies")

```

6. SCREENSHOTS

6 SCREEN SHOTS

6.1 Preparing the data set



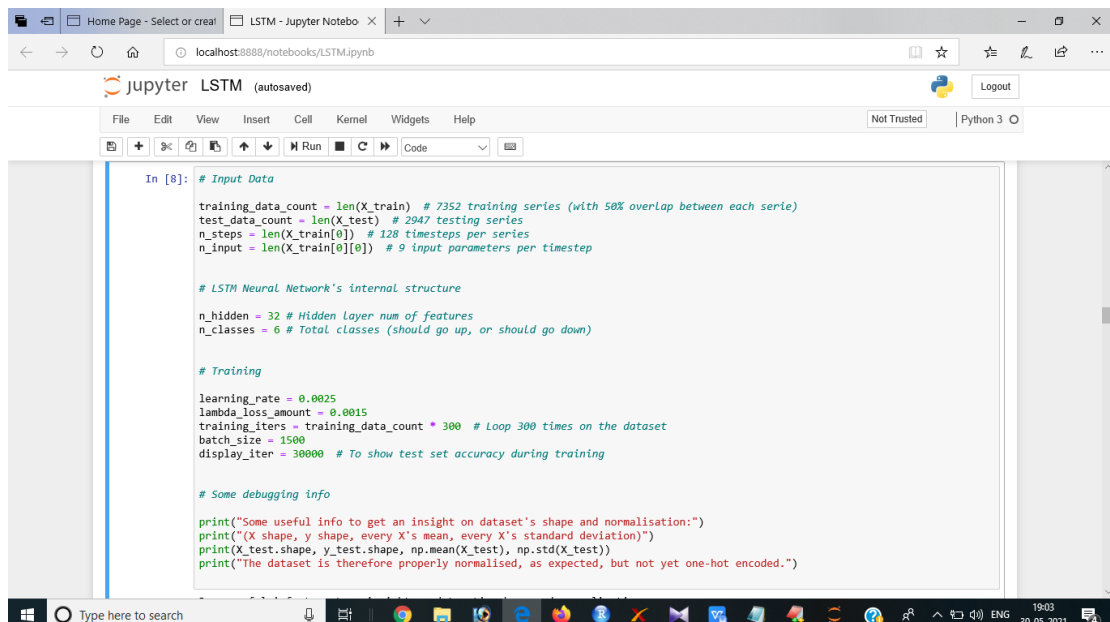
The screenshot shows a Jupyter Notebook interface with a single code cell. The code in the cell performs the following steps:

- Imports necessary modules: `os, urllib, zipfile, tarfile, shutil`.
- Defines the local path for the dataset: `data_path = '/home/ubuntu/pynb/LSTM-Human-Activity-Recognition/data'`.
- Downloads the dataset from the UCI archive: `urllib.urlretrieve('https://archive.ics.uci.edu/ml/machine-learning-databases/00240/UCI%20HAR%20Dataset.zip', 'UCI HAR Dataset.zip')`.
- Extracts the dataset: `zipfile.ZipFile('UCI HAR Dataset.zip').extractall(data_path)`.
- Prints the dataset location: `print('Dataset is now located at: data/UCI HAR Dataset/')`.

The output of the code cell shows the download progress and the successful extraction of the dataset files into the specified directory.

Screenshot 6.1: Preparing the data set

6.2 Training and testing Data

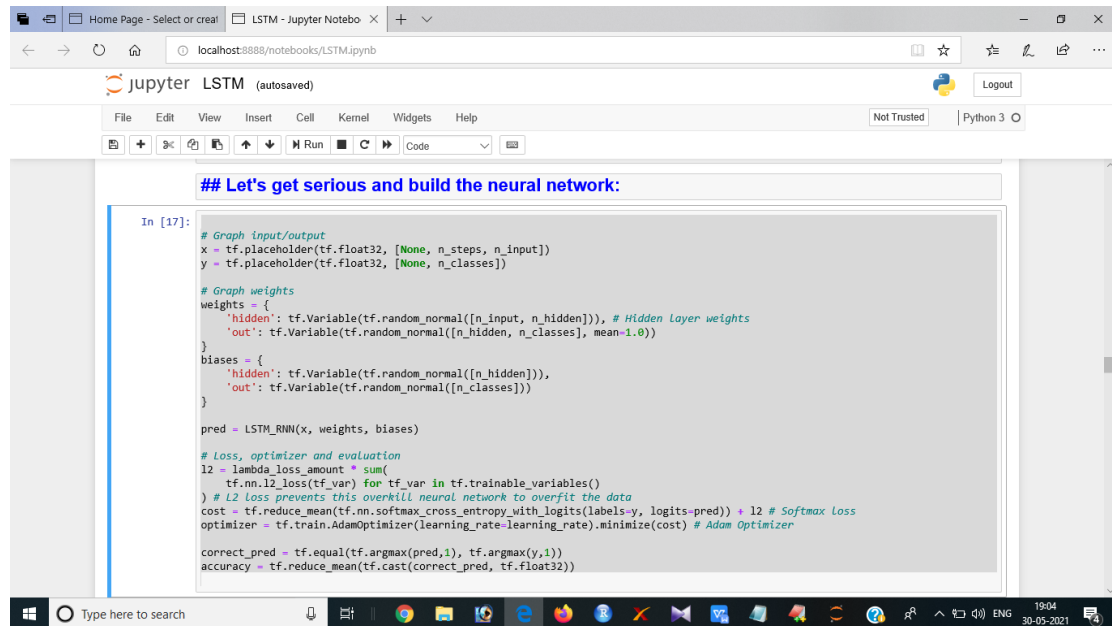


The screenshot shows a Jupyter Notebook interface with a single code cell. The code in the cell configures the training and testing parameters for an LSTM model:

- Defines the training and testing data counts: `training_data_count = len(X_train)` and `test_data_count = len(X_test)`.
- Defines the number of steps and input parameters: `n_steps = len(X_train[0])` and `n_input = len(X_train[0][0])`.
- Defines the LSTM neural network's internal structure: `n_hidden = 32` and `n_classes = 6`.
- Defines the training parameters: `learning_rate = 0.0025`, `lambda_loss_amount = 0.0015`, `training_iters = training_data_count * 300`, `batch_size = 1500`, and `display_iter = 30000`.
- Prints debugging information: `print('Some useful info to get an insight on dataset's shape and normalisation:')`, `print('X shape, y shape, every X's mean, every X's standard deviation:')`, `print(X_test.shape, y_test.shape, np.mean(X_test), np.std(X_test))`, and `print('The dataset is therefore properly normalised, as expected, but not yet one-hot encoded.')`.

Screenshot 6.2: Training and Testing of the data

6.3 LSTM method



The screenshot shows a Jupyter Notebook titled "LSTM - Jupyter Notebo" with the following code in a cell:

```
## Let's get serious and build the neural network:

In [17]:
# Graph input/output
x = tf.placeholder(tf.float32, [None, n_steps, n_input])
y = tf.placeholder(tf.float32, [None, n_classes])

# Graph weights
weights = {
    'hidden': tf.Variable(tf.random_normal([n_input, n_hidden])), # Hidden Layer weights
    'out': tf.Variable(tf.random_normal([n_hidden, n_classes], mean=1.0))
}
biases = {
    'hidden': tf.Variable(tf.random_normal([n_hidden])),
    'out': tf.Variable(tf.random_normal([n_classes]))
}

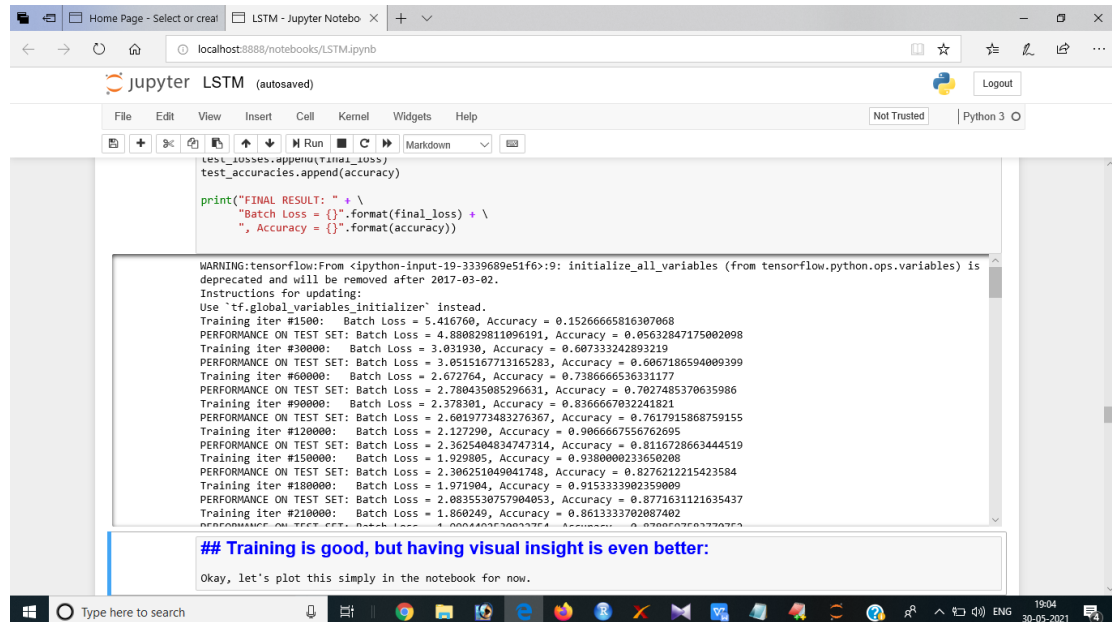
pred = LSTM_RNN(x, weights, biases)

# Loss, optimizer and evaluation
l2 = lambda loss_amount: sum(
    tf.nn.l2_loss(tf_var) for tf_var in tf.trainable_variables()
) # L2 loss prevents this overkill neural network to overfit the data
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=y, logits=pred)) + l2 # Softmax Loss
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost) # Adam Optimizer

correct_pred = tf.equal(tf.argmax(pred,1), tf.argmax(y,1))
accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))
```

Screenshot 6.3: LSTM method

6.4 Running the Epochs



The screenshot shows a Jupyter Notebook titled "LSTM - Jupyter Notebo" with the following code in a cell:

```
test_losses.append(train_loss)
test_accuracies.append(accuracy)

print("FINAL RESULT: " + \
      "Batch Loss = {}".format(final_loss) + \
      ", Accuracy = {}".format(accuracy))
```

The output of the code shows the following results:

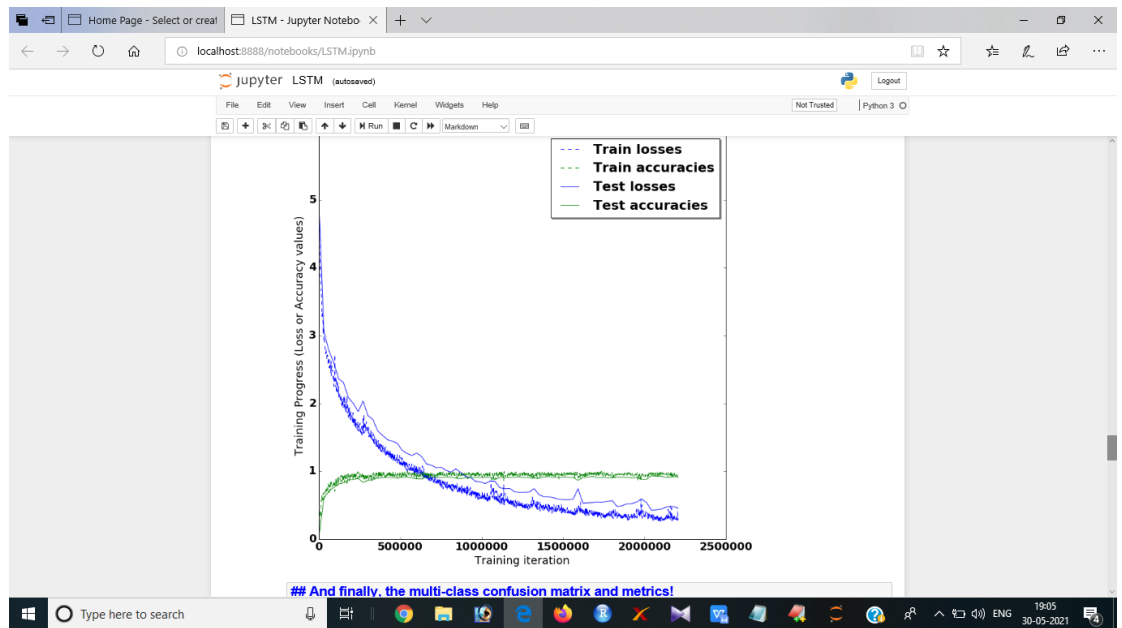
```
WARNING:tensorflow:From <ipython-input-19-3339689e51f6>:9: initialize_all_variables (from tensorflow.python.ops.variables) is deprecated and will be removed after 2017-03-02.
Instructions for updating:
Use 'tf.global_variables_initializer' instead.
Training iter #1500: Batch Loss = 5.416760, Accuracy = 0.15266665816307068
PERFORMANCE ON TEST SET: Batch Loss = 4.880829811096191, Accuracy = 0.05632847175002098
Training iter #30000: Batch Loss = 3.031930, Accuracy = 0.607333242893219
PERFORMANCE ON TEST SET: Batch Loss = 3.0515167713165283, Accuracy = 0.6067186594009399
Training iter #60000: Batch Loss = 2.672764, Accuracy = 0.7386666536331177
PERFORMANCE ON TEST SET: Batch Loss = 2.788435085296631, Accuracy = 0.7027485370635986
Training iter #90000: Batch Loss = 2.378301, Accuracy = 0.8366667032241821
PERFORMANCE ON TEST SET: Batch Loss = 2.6819773483276367, Accuracy = 0.7617915868759155
Training iter #120000: Batch Loss = 2.127290, Accuracy = 0.9066667556762695
PERFORMANCE ON TEST SET: Batch Loss = 2.3625404834747314, Accuracy = 0.8116728663444519
Training iter #150000: Batch Loss = 1.929805, Accuracy = 0.9300000233650208
PERFORMANCE ON TEST SET: Batch Loss = 2.306251049041748, Accuracy = 0.8276212215423584
Training iter #180000: Batch Loss = 1.971904, Accuracy = 0.9153333902359009
PERFORMANCE ON TEST SET: Batch Loss = 2.0835530757904053, Accuracy = 0.8771631121635437
Training iter #210000: Batch Loss = 1.868249, Accuracy = 0.8613333702087402
PERFORMANCE ON TEST SET: Batch Loss = 1.8004403630033754, Accuracy = 0.8308667033333333

## Training is good, but having visual insight is even better:

Okay, let's plot this simply in the notebook for now.
```

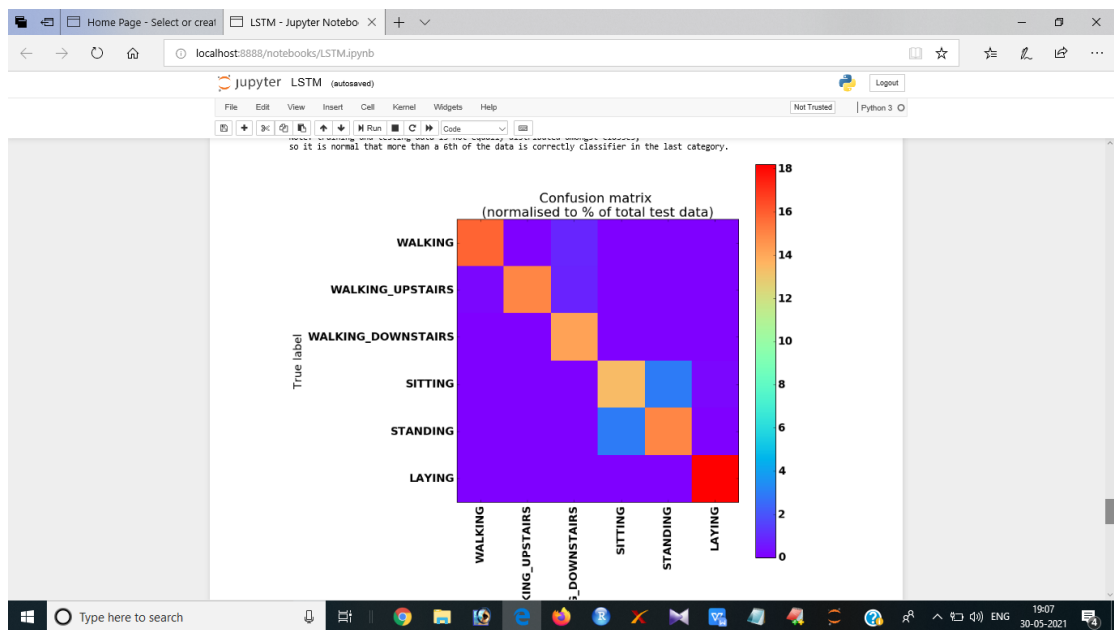
Screenshot 6.4: Running the epochs

6.5 Training and Testing Error and Accuracy



Screenshot 6.5: Error and Accuracy

6.6 LSTM confusion Matrix



Screenshot 6.6: Confusion Matrix

7. TESTING

7.TESTING

7.1 INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

Testing is a procedure, which uncovers blunders in the program. Programming testing is a basic component of programming quality affirmation and speaks to a definitive audit of determination, outline and coding. The expanding perceivability of programming as a framework component and chaperon costs related with a product disappointment are propelling variables for we arranged, through testing. Testing is the way toward executing a program with the plan of finding a mistake. The plan of tests for programming and other built items can be as trying as the underlying outline of the item itself It is the significant quality measure utilized amid programming improvement. Amid testing, the program is executed with an arrangement of experiments and the yield of the program for the experiments is assessed to decide whether the program is executing as it is relied upon to perform.

A technique for programming testing coordinates the outline of programming experiments into an all around arranged arrangement of steps that outcome in fruitful improvement of the product. The procedure gives a guide that portrays the means to be taken, when, and how much exertion, time, and assets will be required. The procedure joins test arranging, experiment configuration, test execution, and test outcome gathering and assessment. The procedure gives direction to the specialist and an arrangement of points of reference for the chief. Due to time weights, advance must be quantifiable and issues must surface as ahead of schedule as would be prudent

Keeping in mind the end goal to ensure that the framework does not have blunders,

the distinctive levels of testing techniques that are connected at varying periods of programming improvement are here.

Framework testing includes in-house testing of the whole framework before conveyance to the client. Its point is to fulfill the client the framework meets all necessities of the customer's determinations. This testing assesses working of framework from client perspective, with the assistance of particular report. It doesn't require any inward learning of framework like plan or structure of code.

It contains utilitarian and non-useful zones of utilization/item. Framework Testing is known as a super arrangement of a wide range of testing as all the significant sorts of testing are shrouded in it. In spite of the fact that attention on sorts of testing may differ on the premise of item, association procedures, course of events and necessities. Framework Testing is the start of genuine testing where you test an item all in all and not a module/highlight.

Acknowledgment testing, a testing method performed to decide if the product framework has met the prerequisite particulars. The principle motivation behind this test is to assess the framework's consistence with the business necessities and check in the event that it is has met the required criteria for conveyance to end clients. It is a pre-conveyance testing in which whole framework is tried at customer's site on genuine information to discover blunders. The acknowledgment test bodies of evidence are executed against the test information or utilizing an acknowledgment test content and afterward the outcomes are contrasted and the normal ones.

The acknowledgment test exercises are completed in stages. Right off the bat, the essential tests are executed, and if the test outcomes are palatable then the execution of more intricate situations are done

Bottom up Approach : Testing can be performed beginning from littlest and most reduced level modules and continuing each one in turn. In this approach testing is directed from sub module to primary module, if the fundamental module is not built up a transitory program called DRIVERS is utilized to recreate the principle module. At the point when base level modules are tried consideration swings to those on the

following level that utilization the lower level ones they are tried exclusively and afterward connected with the already inspected bring down level modules.

Top down Approach : In this approach testing is directed from fundamental module to sub module. in the event that the sub module is not built up an impermanent program called STUB is utilized for mimic the sub module. This sort of testing begins from upper level modules. Since the nitty gritty exercises more often than not performed in the lower level schedules are not given stubs are composed. A stub is a module shell called by upper level module and that when achieved legitimately will restore a message to the calling module demonstrating that appropriate association happened.

7.2 TYPES OF TESTING

7.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results. Unit Testing is done on singular modules as they are finished and turned out to be executable. It is restricted just to the planner's prerequisites. It centers testing around the capacity or programming module. It Concentrates on the interior preparing rationale and information structures. It is rearranged when a module is composed with high union.

It is otherwise called Functional testing. A product testing strategy whereby the inward workings of the thing being tried are not known by the analyzer. For instance,

in a discovery test on a product outline the analyzer just knows the information sources and what the normal results ought to be and not how the program touches base at those yields. The analyzer does not ever inspect the programming code and does not require any further learning of the program other than its determinations. In this system some experiments are produced as information conditions that completely execute every single practical prerequisite for the program. This testing has been utilizations to discover mistakes in the accompanying classifications.

7.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

It is otherwise called Glass box, Structural, Clear box and Open box testing . A product testing procedure whereby express learning of the inner workings of the thing being tried are utilized to choose the test information. Not at all like discovery testing, white box testing utilizes particular learning of programming code to inspect yields. The test is precise just if the analyzer comprehends what the program should do. He or she would then be able to check whether the program veers from its expected objective. White box testing does not represent blunders caused by oversight, and all obvious code should likewise be discernable. For an entire programming examination, both white box and discovery tests are required.

In this the experiments are produced on the rationale of every module by drawing stream diagrams of that module and sensible choices are tried on every one of the cases. It has been utilizations to produce the experiments in the accompanying cases

7.2.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system

documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes.

7.3 TEST CASES

7.3.1 UPLOADING IMAGES

Test case ID	Test case name	Purpose	Test Case	Output
1	User uploads image	Use it for identification	The user uploads the a dog image	Uploaded successfully
2	User uploads 2 nd image	Use it for identification	The user uploads the a non-dogimage	Uploaded successfully

7.3.2 CLASSIFICATION

Test case ID	Test case name	Purpose	Input	Output
--------------	----------------	---------	-------	--------

HUMAN ACTIVITY RECOGNITION

1	Classification test 1	To check if the classifier performs its task	A dog image is given	Breed is predicted.
2	Classification test 2	To check if the classifier performs its task	A human image is given	It predicted as human.
3	Classification test 3	To check if the classifier performs its task	A frog image is given	Predicted as not a dog.

8. CONCLUSION & FUTURE SCOPE

8.CONCLUSION & FUTURE SCOPE

8.1 PROJECT CONCLUSION

In this paper, a platform to combine sensors of smartphones and smartwatches to classify various human activities was proposed. It recognizes activities in real-time. Moreover, this approach is light-weight, computationally inexpensive, and able to run on handheld devices. The results showed that there is no clear winner, but naive Bayes performs best in our experiment in both the classification accuracy and efficiency. The overall accuracy lies between 84.6% and 89.4%, at which the differences are negligible. Thus, this platform is able to recognize various human activities. However, all of the tested classifiers confused walking and using the stairs activities. The second conclusion is that adding the smartwatch's sensor data to the recognition system improves its accuracy with at least six percentage point. Finally, it is computations that the best sampling frequency is in the field of 10 Hz. Some questions still require to be answered.

8.2 FUTURE SCOPE

Most important is the conducting of larger experiments with more people in order to perform more robust evaluation to clarify if indeed one method is better than the other, or whether, any off-the-shelf method can do well in this classification task. This work could be further extended by incorporating more sensors (e.g. heart rate sensor), recognizing high-level activities (e.g. shopping or eating dinner) or extrapolating these trained classifiers to other people.

9. BIBILOGRAPHY

9. BIBLIOGRAPHY

9.1 REFERENCES

- [1] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, Jorge L. Reyes-Ortiz(2012). Human Activity Recognition on Smartphones Using a Multiclass Hardware-Friendly Support Vector Machine. Springer International Workshop on Ambient Assisted Living. Lecture notes in Computer Science. Vol(7657), pp 216- 223.
- [2] Jun Liu, Amir Shahroudy, Dong Xu, Gang Wang(2016). Spatio-Temporal LSTM with Trust Gates for 3D Human Action Recognition. European Conference on Computer Vision., pp 816-833. Vol(9907)
- Oyelade, Oladipupo, Obagbuwa(2010). Application of k-Means Clustering algorithm for prediction of Students' Academic Performance. International Journal of Computer Science and Information Security, 7(1). 292-295
- 4. Juha Vesanto(1999). SOM-Based Data Visualization Methods. Intelligent Data Analysis, Laboratory of Computer and Information Science, Helsinki University of Technology, P. O. Box 5400, FIN-02015 HUT, Finland vol. 3(2), pp. 111-126.
- [5] Ivan Viola(2010). Information Theory in Computer Graphics and Visualization. Proceeding in SA'11 SIGGRAPH Asia 2011 Courses .
- [6] Jiang, L., Zhang, H., & Cai, Z. (2009). A novel Bayes model: Hidden Naïve Bayes. IEEE
- [7] Vincenzi S., Zucchetta M., Franzoi P., Pellizzato M., Pranovi F., De Leo G.A., Torricelli P.(2011). Application of a Random Forest algorithm to predict spatial distribution of the potential yield of *Ruditapes philippinarum* in the Venice lagoon, Italy. Ecological Modelling. 222, 1471–1478.
- [8] Jie Hu, Yu Kong, Yun fu. (2013). Activity Recognition by Learning Structural and Pairwise Mid-level Features Using Random Forest . 2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG) Automatic Face and Gesture Recognition
- [9] J.K. Aggarwal. M.S. Ryoo.(2011). Human activity analysis: A review. ACM Computing Surveys. 43(3)
- [10] Enea Cipitelli, Ennio Gambi, Susanna Spinsante, Francisco Florez-Revuelta (2016). Human Action Recognition Based on Temporal Pyramid of Key Poses

Using RGB-D Sensors. Springer International, Conference, Advanced Concepts for Intelligent Vision Systems. PP 510-521.vol(10016)

[11] Qingzhong Liu, Zhaoxian Zhou, SarbagyaRatnaShakya, PrathyushaUduthalapally, MengyuQiao, and Andrew H. Sung(2018). “Smartphone Sensor-Based Activity Recognition by Using Machine Learning and Deep Learning Algorithms” International Journal of Machine Learning and Computing, Vol. 8, No. 2.

[12]Ms.S.Roobini, Ms.J.FenilaNaomi(2019). Smartphone Sensor Based Human Activity Recognition using Deep Learning Models. International Journal of Recent Technology and Engineering . ISSN: 2277-3878, Volume-8, Issue1.

[13] Ye , G. Stevenson, and S. Dobson, “Kcar: A knowledge-driven approach for concurrent activity recognition,”Pervasive and Mobile Computing, vol. 19 , pp. 47 – 70, 2015.

[14] O . D. Lara and M. A. Labrador, “A survey on human activity recognition using wearable sensors,”IEEE Communications Surveys and Tutorials,vol. 15, no. 3, pp. 1192–1209, 2013. 15. [15]M. ZiaeeFard and R. Berg evin, “Semantic human activity recognition: A literature review,”Pattern Recognition, vol. 48, no. 8, pp. 2329 – 2345,2015..

9.2 WEBSITES

[1] [https://web.stanford.edu/class/cs231a/prev_projects_2016/output%20\(1\).pdf](https://web.stanford.edu/class/cs231a/prev_projects_2016/output%20(1).pdf)

[2] <https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234->.