Annexure – 1

Identifying Activity of Humans using CNN with Auto-Encoders and LSTM

A Major Project Report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE & ENGINEERING

By

1. K. Lovely Srenika	-	19B01A0583
2. M. Alekhya	-	19B01A0594
3. V. Harshita Sai	–	19B01A05A2
4. N. Lasya	–	19B01A05B9
5. N. Tulasi	–	19B01A05C2

Under the esteemed guidance of
Dr. T. Gayathri Ph.D
(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN(A)
(Approved by AICTE, Accredited by NBA & NAAC, Affiliated to JNTU Kakinada) BHIMAVARAM – 534 202

2022 – 2023

Annexure – 2

SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN (A)
(Approved by AICTE, Accredited by NBA & NAAC,
Affiliated to JNTU Kakinada)
BHIMAVARAM – 534 202

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the Major Project entitled "**Identifying Activity of Humans using CNN with Auto-Encoders and LSTM**", is being submitted by **K.Lovely Srenika**, **M.Alekhya, V.Harshita Sai, N.Lasya, N.Tulasi** bearing the **Regd. No. 19B01A05A83, 19B01A0594, 19B01A05A2, 19B01A05B9, 19B01A5C2** in partial fulfillment of the requirements for the award of the degree of "**Bachelor of Technology** in **Computer Science & Engineering**" is a record of bonafide work carried out by them under my guidance and supervision during the academic year 2022–2023 and it has been found worthy of acceptance according to the requirements of the university.

Internal Guide **Head of the Department**

ACKNOWLEDGEMENTS

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant encouragement and guidance has been a source of inspiration throughout the course of this project. I take this opportunity to express our gratitude to all those who have helped me in this project.

I wish to place my deep sense of gratitude to **Sri. K. V. Vishnu Raju, chairman of SVES**, for his constant support on each and every progressive work of mine.

My deep sense of gratitude and sincere thanks to **Dr. G. Srinivasa Rao, Principal of SVECW** for being a source of an inspirational constant encouragement.

My deep sense of gratitude and sincere thanks to **Dr. P. Srinivasa Raju, VicePrincipal of SVECW** for being a source of an inspirational constant encouragement.

I am deeply indebted and sincere thanks to **Dr. P. Kiran Sree, Head of the Department**, for his valuable advice in completing this project successfully.

I am extremely thankful and indebted to **Dr. T. Gayathri, Assistant Professor,** Department of Computer Science and Engineering, SVECW for her constant guidance, encouragement and moral support throughout the project.

My deep sense of gratitude and sincere thanks for his unflinching devotion and valuable suggestions throughout my project.

Project Associates

1. K. Lovely Srenika	**– 19B01A0583**
2. M. Alekhya	**– 19B01A0594**
3. V. Harshita Sai	**– 19B01A05A2**
4. N. Lasya	**– 19B01A05B9**
5. N. Tulasi	**– 19B01A05C2**

Abstract

Recognizing a person's activities has been a critical Time series classification problem that includes predicting Human movements based on sensor data. The conventional approach using Machine learning algorithms requires in-depth domain knowledge for signal processing to the correct construction of features from raw data. This paper proposes deep learning approaches for Human activity recognition with data from sensors such as accelerometers and gyroscopes.

The Convolutional Neural Networks(CNNs) with the ability of automatic feature extraction, autoencoders(AEs) used for dimensionality reduction, and Long short-term memory(LSTM), which are good at temporal modeling, complement each other. In this work, we would like to leverage the complementarity of CNNs, AEs, and LSTMs by integrating them into a unified architecture.

CONTENTS

INTRODUCTION

1. Introduction

1.1 Theoretical Background

Efficient, Accurate, and Faster Human activity recognition(HAR) can have a wide range of applications. For Instance, in healthcare, particularly in the case of elderly people, HAR, when used with other technologies such as the Internet of Things (IoT), can be leveraged to initiate a quick response to any contingency as the Human activity is being observed, thereby reducing loss of lives. HAR involves recognizing a variety of human activities, including walking, running, sitting, sleeping, standing, showering, cooking, driving, and unusual activities. We can use wearable sensors or accelerometers and video frames to collect data. The crime rate can also be monitored by HAR, and recognizing human activity during vehicle driving can also lead to safe travel.

Various HAR frameworks have been proposed based on audio/video data frameworks, wearable sensor-based, and smartphone sensor-based, which cites different pros and cons according to the ease with which the information is gathered or by measuring the utility of the sensors in use. For example, wearable and audio/video-based sensors can be uncomfortable to the users and require complex

signal processing. Moreover, these sensors are additional baggage to the users and are prone to physical obstacles. On the contrary, portability and intraoperative properties of the smartphone have been leveraged to build smartphone-based physical HAR systems. Data can be collected continuously while carrying out any physical activity through a smartphone. Furthermore, the range of built-in smartphone sensors makes mobile health-related data monitoring more elegant and accurate.

Sensor data obtained from smartphones contain multivariate time-series data. One of the intrinsic features of time-series data is Local dependency. Further, human activity signals have hierarchical characteristics, and it is translation-invariant. Physical activities have a few distinct characteristics. As a result, HAR faces a number of methodological obstacles, including unbalanced datasets, the multiclass window problem, etc

1.2 Motivation

Smartphone-based HAR systems that use traditional machine learning algorithms or deep learning techniques are becoming increasingly popular. Because it extracts

the relevant characteristics that are responsible for discriminating diverse activity patterns, feature engineering is a dominant phase in classical ML approaches. The feature extraction of raw signals has a significant impact on the accuracy of HAR solutions. The characteristics are then input into classifiers, which identify human behaviors. Conventional classifiers fail to recognize physical activities competently and correctly without a proper feature engineering approach. As a result, extensive data preparation techniques are necessary to deliver sensory data in a suitable format, and handmade characteristics are derived from the collected sensory data based on expert domain knowledge.

Finally, to recognize distinct human physical activities, the handmade feature vector is passed to traditional classifiers. However, previous research has shown that while some of these handmade traits are effective at identifying one action from another, they are not as good at recognizing others. Furthermore, to properly address classification issues, different study topics necessitate distinct handmade feature vectors.

1.3 Aim of the Proposed Work

The difficulty of predicting what a person is doing based on a trail of their movement using sensors is known as Human Activity Recognition (HAR). Standing, sitting, leaping, and moving up and down stairs are some examples of movements. Sensors, such as those found on a smartphone or a vest, are frequently used to gather accelerometer data in three dimensions (x, y, z).

1.4 Objective(s) of the proposed work

In this project, we would like to propose and build advanced methodologies and compare them with the previous methods. To build a robust system, we would like to consider datasets where data is collected from various sensors, images, accelerometers, and gyroscopes. We will study various deep neural network

architectures based on convolutional neural networks, recurrent neural networks to solve this problem and propose a novel architecture that can better perform the task. The primary public dataset that will be used in this project is the "Human Activity Recognition Using Smartphones" dataset introduced by Davide Anguita et al. To collect the data; The experiment was conducted in a group of 30 volunteers within the 19-48 year age group. Each person wore a smartphone on their waist
and perform six different activities (walking,walking_downstairs,walking_upstairs, standing, sitting, laying). With a built-in accelerometer and gyroscope in the smartphone, recorded 3-axis linear acceleration and 3-axis angular velocity at a constant frequency of 50Hz. The experiment was recorded on video for manual labeling of the data.

SYSTEM ANALYSIS

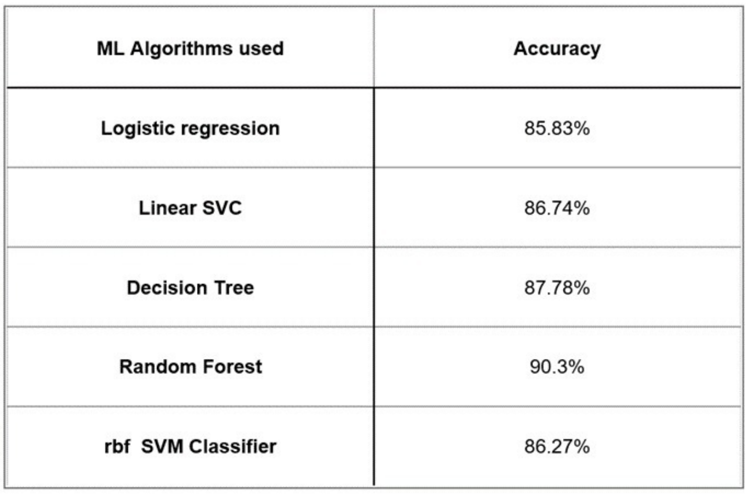
2.System Analysis

2.1 Existing System

Machine learning methods with handcrafted features are the dominant methods used in previous works. We tested various ML methods and found the Average validation accuracies.

Deep learning Methods such as CNN, LSTM and other combinations resulted in the test accuracy of around 90% with a validation accuracy of around 95% in the previous research works.

The Fig 2.1 describes the accuracy of different Machine Learning algorithms in implementing Human Activity Recognition systems.



ML Algorithms used	Accuracy
Logistic regression	85.83%
Linear SVC	86.74%
Decision Tree	87.78%
Random Forest	90.3%
rbf SVM Classifier	86.27%

Fig 2.1 Accuracy of Human Activity Recognition with various ML algorithms

Primary Dataset Description

In this research work, Primarily UCI-HAR dataset is being used as this is used as a benchmark dataset in previous research works.

The experiment was conducted in a group of 30 volunteers in the 1948 age group. Each subject wore a smartphone (Samsung Galaxy S II) on his waist and performed 6 activities (WALK, WALK_UP, WALK_DOWN, SIT, STAND, LAY).

Using an embedded gyroscope and accelerometer, we sampled 3-axial angular velocity and 3-axis linear acceleration at a constant 50Hz rate.

The experiment was videotaped to manually label the data. The resulting dataset was randomly split into two sets, 70% of the volunteers were selected to generate training data and 30% were selected as test data.

2.2 Proposed System

The proposed model consists of three different modules, the first module is a convolutional AE, containing a convolutional layer, a pooling layer and a deconvolutional layer. The output of the convolution AE passes through the flattened layer and is converted to the LSTM input format. The LSTM output goes through a fully connected layer to get a high level representation. Finally, the softmax (using the softmax function) layer is used as the final step in detecting human physical activity.

A flattened layer is added after the deconvolutional layer of the convolution AE, and the feature data of the LSTM layer is formatted. This is because the data format of the convolution layer is different from the input data format of the LSTM layer.

The time-sharing wrapper provided by Python's Keras library takes a slice as an argument and applies a convolution to the signal while maintaining the temporal integrity of the LSTM slice.

Since the time division layer operates in 3D data format, it is necessary to convert the input signal from 128 time frames using the exact number of signals. A total of 128 timeframes are divided into 4 slices of 32 timeframes each.

2.3 Feasibility Study

An important outcome of preliminary investigation is the determination that the system request is feasible. This is possible only if it is feasible within limited resources and time. The different feasibilities that have to be analyzed are

1.Economic Feasibility

2.Technical Feasibility

Economic Feasibility:

Economic feasibility or Cost-benefit is an assessment of the economic justification for the computer based project. Here we don't need any hardware for this project .Only software is required that is basically present in every pc.

Technical Feasibility:

Technical Feasibility is the process of validating the technology assumptions, architecture and design of a product or project. This project is planned to run using Python language and Visual Studio Code editor.

SYSTEM REQUIREMENTS SPECIFICATION

3. System Requirements Specification

3.1 Software Requirements

IDE : Google Colab

Language : Python

Libraries :

- ➔ Tensorflow
- ➔ Keras
- ➔ Pandas
- ➔ Numpy

3.2 Hardware Requirements

Processor : Tesla P 100

RAM : 12GB

Space on Hard Disk : 500GB

3.3 Functional Requirements

In Software Engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, behavior, outputs. Functional requirements may be calculations, technical details, data manipulation and supposed to accomplish. The plan for implementing functional requirements is detailed in the system design. In requirements engineering, functional requirements specify particular results of a system. Functional requirements drive the application architecture of a system.

3.4 Non – Functional Requirements

Reliability : The reliability will be depending on the accuracy of the data. To collect the data we use the Yahoo finance api which will help to get the data dynamically.

Efficiency : We will maintain the possible highest accuracy in closing date of the stock prices.

Performance : We will use the LSTM model in prediction the stock prices which is the highest accurate model and the Yahoo finance api will generate the data whenever required which makes the data updated regularly.

SYSTEM DESIGN

4. System Design

4.1 Introduction

Design is the first step in the development phase of an engineering product or system. Design is the place where quality is considered in software development. Design is the only way that we can accurately translate user requirements into finished software products or systems. Software design serves as the foundation for all the software engineers and software maintenance that steps follow. Without design we risk building an unstable design, one that will fail when small changes are made, one that may be difficult to test and one whose quantity cannot be assessed until late in the software engineering process.

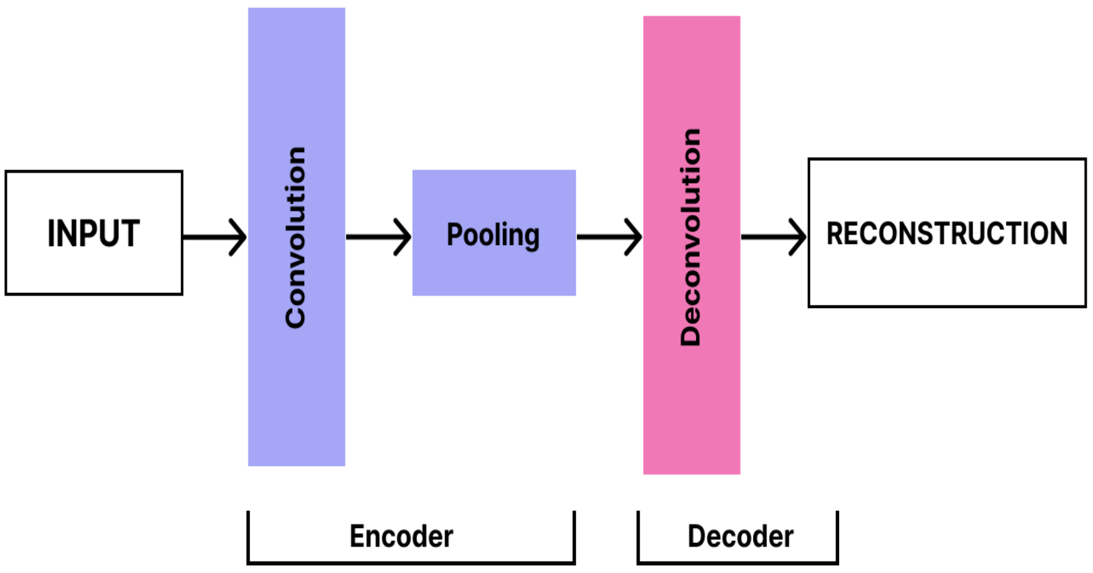


Fig 4.1.1 System flow

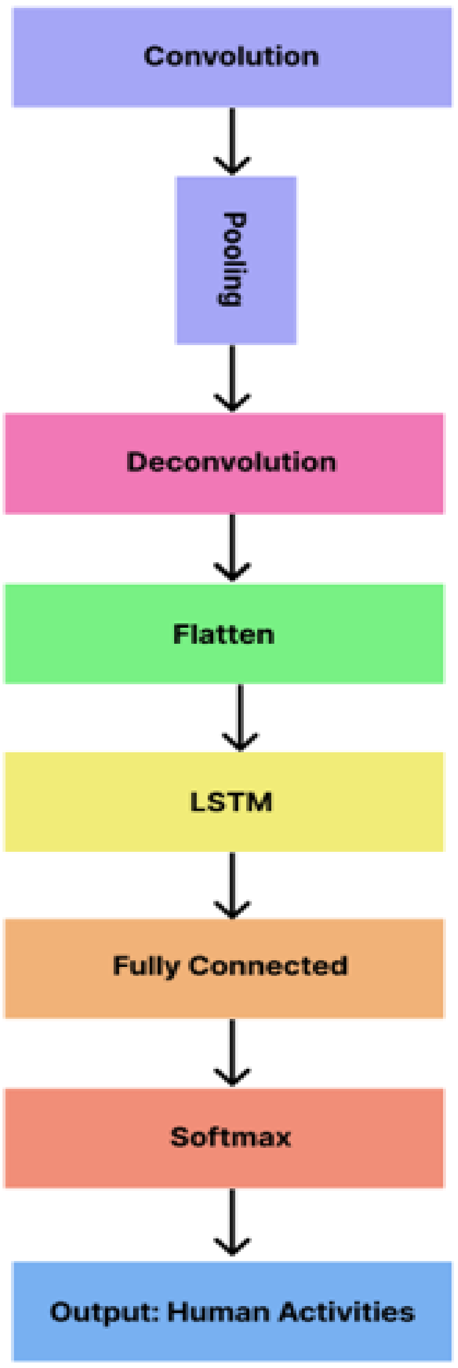


Fig 4.1.2 System flow

The different steps involved in system flow are:

- **Convolution**

  Convolution is a mathematical operation that allows the merging of two sets of information. In the case of CNN, convolution is applied to the input data to filter the information and produce a feature map. This filter is also called a kernel, or feature detector.

  Convolution is an orderly procedure where two sources of information are intertwined; it's an operation that changes a function into something else. Convolutions have been used for a long time typically in image processing to blur and sharpen images, but also to perform other operations. (e.g. enhance edges and emboss) CNNs enforce a local connectivity pattern between neurons of adjacent layers.

- **Pooling**

  Pooling is nothing other than down sampling of an image. The most common pooling layer filter is of size 2x2, which discards three forth of the activations. Role of the pooling layer is to reduce the resolution of the feature map but retain features of the map required for classification through translational and rotational invariants. In addition to spatial invariance robustness, pooling will reduce the computation cost by a great deal.

  Backpropagation is used for training of pooling operations. It again helps the processor to process things faster.

- **Deconvolution**

  Deconvolution is a quantitative approach that uses the picture as an estimate of the real specimen intensity and conducts the mathematical inverse of the imaging process to generate an improved estimate of the image intensity using a formula for the point spread function.

  The deconvolution operation is an upsampling procedure that both upsamples feature maps and keeps the connectivity pattern. The deconvolutional layers essentially increase and densify the input by employing convolution-like procedures with numerous filters. Deconvolution, unlike previous scaling algorithms, has trainable parameters. During network training, the weights of deconvolutional layers are constantly updated and refined. It is accomplished by inserting zeros between the consecutive neurons in the receptive field on the input side, and then one convolution kernel with a unit stride is used on top.

- **Flatten**

Flattening involves taking the pooled feature map that is generated in the pooling step and transforming it into a one-dimensional vector. The reason why we transform the pooled feature map into a one-dimensional vector is because this vector will now be fed into an artificial neural network.

Once the pooled featured map is obtained, the next step is to flatten it. Flattening involves transforming the entire pooled feature map matrix into a single column which is then fed to the neural network for processing.

- **Long / Short Term Memory**

Schmidhuber and Hochreiter in 1997 built a neural network which is called long short term memory networks (LSTMs). Its main goal is to remember things for a long time in a memory cell that is explicitly defined. Previous values are stored in the memory cell unless told to forget the values by "forget gate". New stuff is added through the "input gate" to the memory cell, and it is passed to the next hidden state from the cell along the vectors which is decided by the "output gate". Composition of primitive music, writing like Shakespeare, or learning complex sequences are some of the applications of LSTMs.

Temporal features play an important role in modeling human movements. In recent years, LSTMs have performed impressively in HAR and various other domains. The temporal features are extracted from time sensory signals by the LSTM architecture because of its long-term dependencies and temporal characteristics.

In our proposed architecture as explained earlier, convolutional AE is followed by a LSTM model. The results of the convolutional AE are passed as inputs to the LSTM to deduce the latent temporal interactions across the timeframes.

- **Fully connected Layer**

  A fully connected layer refers to a neural network in which each neuron applies a linear transformation to the input vector through a weights matrix. As a result, all possible connections layer-to-layer are present, meaning every input of the input vector influences every output of the output vector.

- **Softmax**

  Softmax is implemented through a neural network layer just before the output layer. The Softmax layer must have the same number of nodes as the output layer. Full Softmax is the Softmax we've been discussing; that is, Softmax calculates a probability for every possible class. Candidate sampling means that Softmax calculates a probability for all the positive labels but only for a random sample of negative labels. For example, if we are interested in determining whether an input image is a beagle or a bloodhound, we don't have to provide probabilities for every non-doggy example.

  Full Softmax is fairly cheap when the number of classes is small but becomes prohibitively expensive when the number of classes climbs. Candidate sampling can improve efficiency in problems having a large number of classes.

4.2 Methodology Adapted

Various human activities are identified using multivariate time series data which are collected from smartphone based sensors such as gyroscopes and triaxial accelerometers. The sensory data collected through smartphones are noisy and are not appropriate for recognising fundamental patterns. Conventional filtering techniques such as median filtering, low pass and high pass are used to remove the noise from the raw sensory data. After the noise has been removed, feature engineering is used to extract the features which are used as an input to the classifier. However, traditional noise removal and classification requires constant human intervention and expertise and also they fail to reveal the temporal independence of data.

Temporal window based data segmentation is used to assign a class for an activity in Machine learning and deep learning architectures. Extracting features from labeled data requires an approximated size of the sliding window over the sensory data streams. This might incur loss of important information. Moreover, the accuracy of the activity recognition can be improved by increasing the size of the window but a large window size causes response time delays in real time HAR. Therefore, AE which is an unsupervised feature learning approach is used where we do not require labeled data.

The model proposed consists of three modules, the first being the convolutional AE consisting of a convolutional layer, a pooling layer and a deconvolutional layer. The output of this layer is passed through a flattened layer which serves as a desired input for the LSTM layer. The LSTM output goes through a fully connected layer to get a high level representation.

A. Convolutional AE

Convolutional AE is a variation of AE where a Convolutional layer replaces a fully connected layer. Convolutional AE has the advantages of both an AE with unsupervised pre-training capabilities and a convolutional layer. The difference between an AE and a Convolutional AE is that instead of having a fully connected layer in the decoder, Convolutional AE contains deconvolutional layers in the decoder and convolutional layers in the encoder. The proposed Convolutional AE comprises a convolution, pooling and deconvolution layers as shown in the Figure[]. Encoder includes a convolutional layer and a Maxpool layer whereas Decoder comprises a deconvolutional layer. The results from the convolutional layer are encoded with a Maxpool layer that allows high-layer representations that doesn’t alter with small changes in the inputs which reduces the overall computational cost. The Activation function of the proposed convolution deconvolution layer is represented below

$$h^k = \sigma\left(\sum_{l \in L} x^l \otimes w^k + b^k\right)$$

Equation 1

Where,

- h^k = latent representation of the k^{th} feature map of the current layer
- σ = activation function
- x^l = l^{th} feature map of the group of feature maps L obtained from the previous layer
- $\otimes$ = a 2D convolution operation

- w^k = weights of the k^{th} feature map of the current layer
- b^k = bias of the k^{th} feature map of the current layer

B. LSTM

Temporal features play an important role in modeling human movements. In recent years, LSTMs have performed impressively in HAR and various other domains. The temporal features are extracted from time sensory signals by the LSTM architecture because of its long-term dependencies and temporal characteristics.

Schmidhuber and Hochreiter in 1997 built a neural network which is called long short term memory networks (LSTMs). Its main goal is to remember things for a long time in a memory cell that is explicitly defined. Previous values are stored in the memory cell unless told to forget the values by "forget gate". New stuff is added through the "input gate" to the memory cell, and it is passed to the next hidden state from the cell along the vectors which is decided by the "output gate". Composition of primitive music, writing like Shakespeare, or learning complex sequences are some of the applications of LSTMs.

The architecture of a conventional LSTM is represented in Figure.

In our proposed architecture as explained earlier, convolutional AE is followed by a LSTM model. The results of the convolutional AE are passed as inputs to the LSTM to deduce the latent temporal interactions across the timeframes.

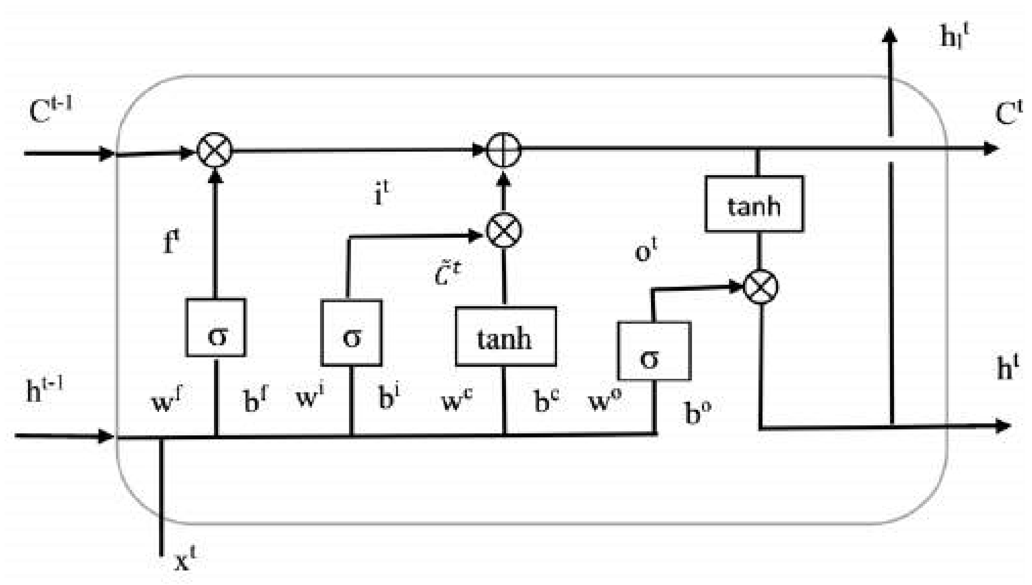


Fig 4.2.1 Architecture of conventional LSTM

4.3 Data flow diagrams (UML Diagrams)

Introduction to UML

A model is an abstract representation of a system, constructed to understand the system priority to building or modifying it. A model is a simplified representation of reality and it provides a means for conceptualization and communication of ideas in a precise and ambiguous form. We build models so that we can better understand the system we are developing. The elements are like components which can be associated in different ways to make a complete UML picture, which is known as a diagram. Thus, it is very important to understand the different diagrams to implement the knowledge in real life systems.

UML (Unified Modeling Language) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. It is a method for describing the system architecture in detail using the blueprint. We use UML diagrams to portray the behavior and structure of a system. This is the step while developing any product after analysis. The goal from this is to produce a model of the entities involved in the project which later need to be built. The representation of the entities that are to be used in the product being developed need to be designed.

There are various kinds of methods in software design:

- Use case Diagram
- Class Diagram
- Sequence Diagram
- Activity Diagram
- State Chart Diagram

4.3.1 Use Case Diagram

Use Case Diagrams are used to depict the functionality of a system or a part of a system. They are widely used to illustrate the functional requirements of the system and its interaction with external agents (actors).

A use case is basically a diagram representing different scenarios where the system can be used. A use case diagram gives us a high level view of what the system or a part of the system does without going into implementation details. When the initial task is complete, use case diagrams are modelled to present the outside view.

In brief, the purposes of use case diagrams can be said to be as follows

- Used to gather the requirements of a system.
- Used to get an outside view of a system.
- Identify the external and internal factors influencing the system.

Use case diagrams commonly contains

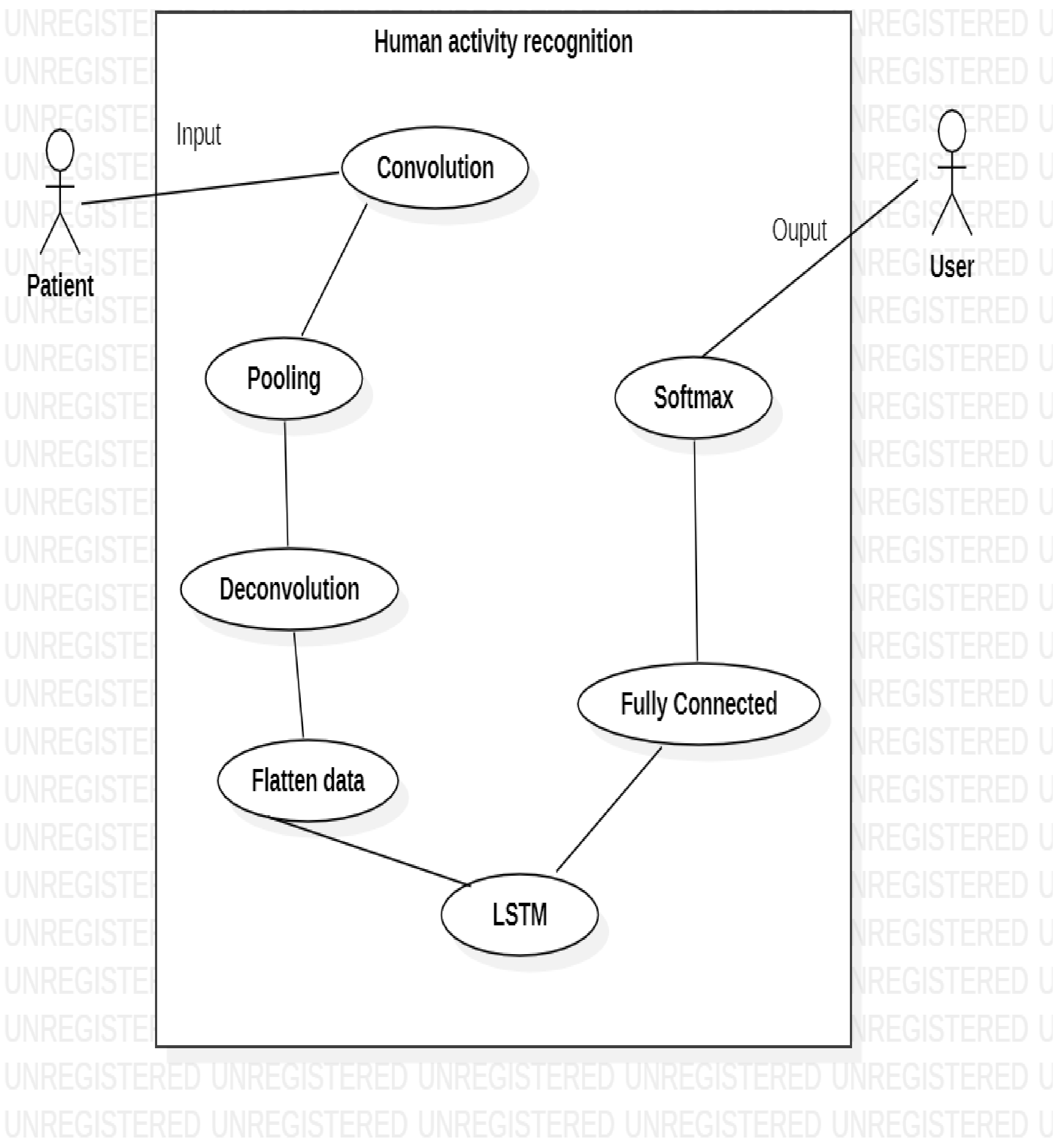
- Use cases
- Actors
- Dependency, generalization and association relationships.

Use cases

A use case is a software and system engineering term that describes how a user uses system to accomplish a particular goal.

Actors

An actor is a person, organisation or external system that plays a role in one or more interactions with the system.



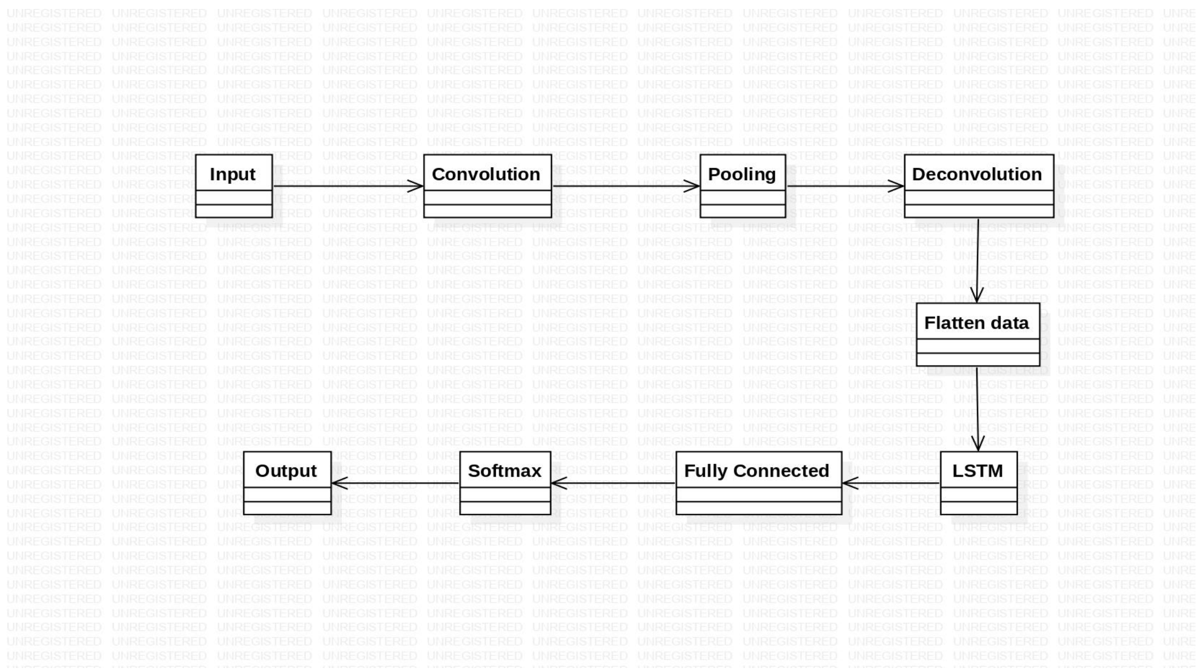
4.3.1 UseCase Diagram

4.3.2 Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages. It is also known as a structural diagram.

Class diagram contains

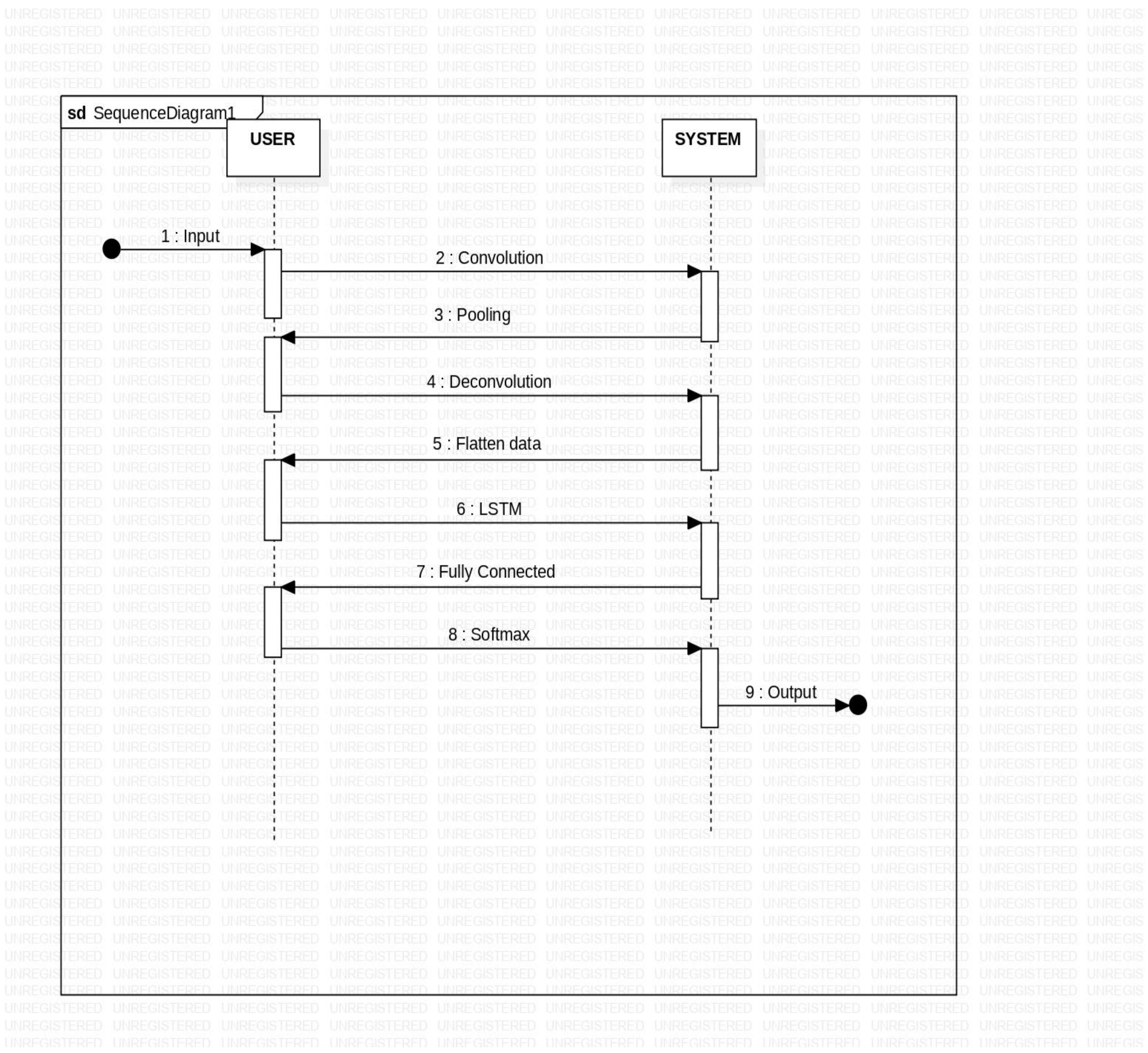
- Classes
- Interfaces
- Dependency, generalization and association.



4.3.2 Class Diagram

4.3.3 Sequence Diagram

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. Sequence diagrams are used to formalize the behavior of the system and to visualize the communication among objects. These are useful for identifying additional objects that participate in the use cases. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.



4.3.3 Sequence Diagram

4.3.4 Activity Diagram

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all types of flow control by using different elements such as fork, join, etc. The basic purpose of the activity diagram is similar to the other four diagrams.

It captures the dynamic behaviour of the system. Other four diagrams are used to show the message flow from one object to another Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

- Describe the sequence from one activity to another.
- Draw the activity flow of a system.
- Describe the parallel, branched and concurrent flow of the system.

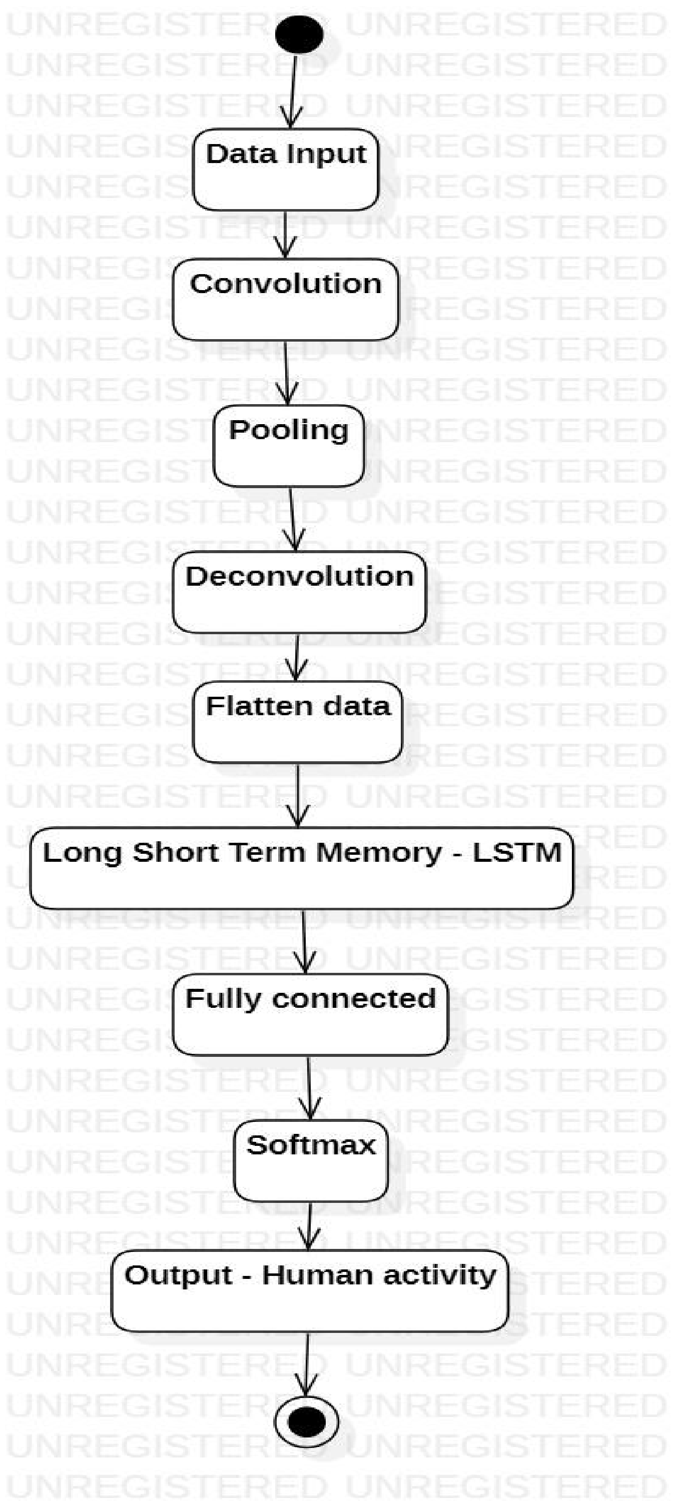


Fig 4.3.4 Activity Diagram

4.3.5 State Chart Diagram

State chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of a State chart diagram is to model the lifetime of an object from creation to termination. State chart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.

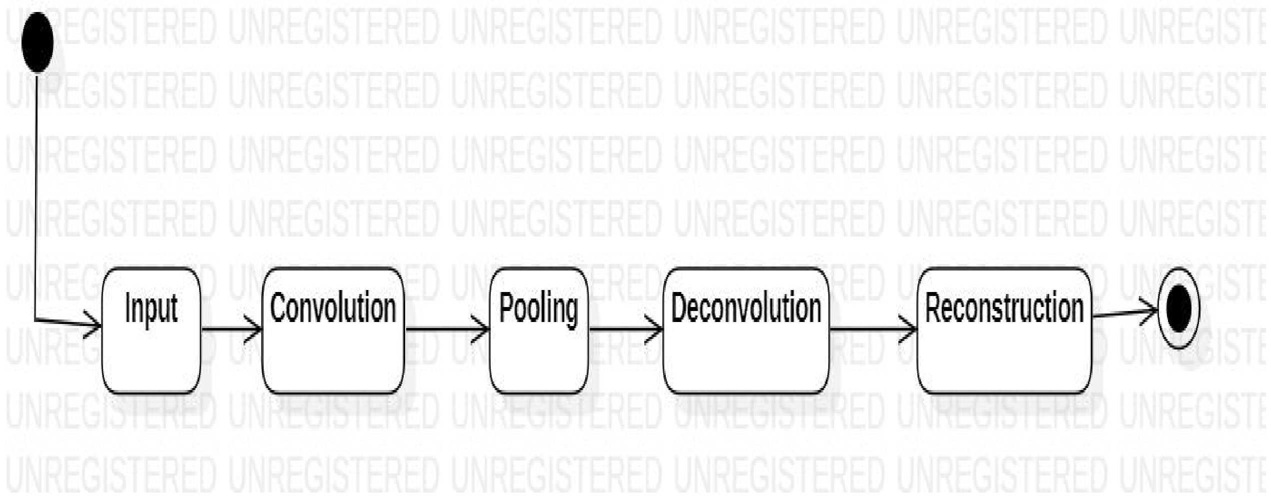


Fig 4.3.2 State Chart Diagram

SYSTEM IMPLEMENTATION

5. System Implementation

5.1 Introduction

The purpose of system implementation can be summarized as follows: making the new system available to the prepared set of users (the deployment), and positioning ongoing support and maintenance of the system within the performing organization (the transaction). At a finer level necessary to educate the consumer on the use of the system, placing the newly developed system into production, confirming that business functions that interact with the system and functioning properly. Transitioning the system support responsibilities involve changing from a system development to the system and maintenance mode of operation, with ownership of the new system moving from the project team to the performing organization. A key difference between system implementation and all other phases of the lifecycle is that all project activities up to this point have been performed in safe, protected and checked environments. It is through the careful planning, execution and management of system implementation activities that the project team can minimize the likelihood of these occurrences and determine appropriate contingency plans in the event of the problem.

5.2 Implementation

5.2.1. Dataset

1. We are creating firstly more signals from these existing ‘9’ by using some expert/domain knowledge. And using statistical techniques like mean, median, standard deviation etc to create a large number of features from each of the signals.

By using the sensors(Gyroscope and accelerometer) in a smartphone, they have captured '3-axial linear acceleration'(tAcc-XYZ) from accelerometer and '3-axial angular velocity' (tGyro-XYZ) from Gyroscope with several variations in collecting dataset.

- prefix 't' in those metrics denotes time.
- suffix 'XYZ' represents 3-axial signals in X , Y, and Z directions.

So, In the original dataset we have only ‘9’ (Nine) signals which are,

Body acceleration in x,y,z directions

Body gyro in x,y,z directions

Total acceleration in x,y,z directions

Now, we transformed these signals.

These sensor signals are preprocessed by applying noise filters and then sampled in fixed-width windows(sliding windows) of 2.56 seconds each with 50% overlap. ie., each window has 128 readings.

2. From Each window, a feature vector was obtianed by calculating variables from the time and frequency domain.In our dataset, each datapoint represents a window with different readings
3. The accelertion signal was saperated into Body and Gravity acceleration signals(tBodyAcc-XYZ and tGravityAcc-XYZ) using some low pass filter with corner frequecy of 0.3Hz.
4. After that, the body linear acceleration and angular velocity were derived in time to obtian jerk signals (tBodyAccJerk-XYZ and tBodyGyroJerk-XYZ).
5. The magnitude of these 3-dimensional signals were calculated using the Euclidian norm. This magnitudes are represented as features with names like tBodyAccMag, tGravityAccMag, tBodyAccJerkMag, tBodyGyroMag and tBodyGyroJerkMag.
6. Finally, We've got frequency domain signals from some of the available signals by applying a FFT (Fast Fourier Transform). These signals obtained were labeled with prefix 'f' just like original signals with prefix 't'. These signals are labeled as fBodyAcc-XYZ, fBodyGyroMag etc.,.
7. These are the signals that we got so far.
    - tBodyAcc-XYZ
    - tGravityAcc-XYZ
    - tBodyAccJerk-XYZ
    - tBodyGyro-XYZ
    - tBodyGyroJerk-XYZ
    - tBodyAccMag
    - tGravityAccMag
    - tBodyAccJerkMag
    - tBodyGyroMag
    - tBodyGyroJerkMag
    - fBodyAcc-XYZ
    - fBodyAccJerk-XYZ

- fBodyGyro-XYZ
- fBodyAccMag
- fBodyAccJerkMag
- fBodyGyroMag
- fBodyGyroJerkMag

8. We can esitmate some set of variables from the above signals. ie., We will estimate the following properties on each and every signal that we recoreded so far.

- **mean():** Mean value
- **std():** Standard deviation
- **med():** Median absolute deviation
- **max():** Largest value in array
- **min():** Smallest value in array
- **sma():** Signal magnitude area
- **energy():** Energy measure. Sum of the squares divided by the number of values.
- **iqr():** Interquartile range
- **entropy():** Signal entropy
- **arCoeff():** Autoregression coefficients with Burg order equal to 4
- **correlation():** correlation coefficient between two signals
- **maxInds():** index of the frequency component with largest magnitude
- **meanFreq():** Weighted average of the frequency components to obtain a mean frequency
- **skewness():** skewness of the frequency domain signal

◦ **kurtosis():** kurtosis of the frequency domain signal

◦ **bandsEnergy():** Energy of a frequency interval within the 64 bins of the FFT of each window.

◦ **angle():** Angle between two vectors.

9. We can obtain some other vectors by taking the average of signals in a single window sample. These are used on the angle() variable' `

◦ gravityMean

◦ tBodyAccMean

◦ tBodyAccJerkMean

◦ tBodyGyroMean

◦ tBodyGyroJerkMean

5.2.2. Exploratory Data Analysis

Data

- All the data is present in the 'UCI_HAR_dataset/' folder in the present working directory.
    - Feature names are present in 'UCI_HAR_dataset/features.txt'
    - *Train Data*
        - 'UCI_HAR_dataset/train/X_train.txt'
        - 'UCI_HAR_dataset/train/subject_train.txt'
        - 'UCI_HAR_dataset/train/y_train.txt'
    - *Test Data*
        - 'UCI_HAR_dataset/test/X_test.txt'
        - 'UCI_HAR_dataset/test/subject_test.txt'
        - 'UCI_HAR_dataset/test/y_test.txt'

Obtain the train data

```
# get the data from txt files to pandas dataframe
X_train = pd.read_csv('UCI_HAR_Dataset/train/X_train.txt', delim_whitespace=True, header=None)
X_train.columns = features


# add subject column to the dataframe
X_train['subject'] = pd.read_csv('UCI_HAR_Dataset/train/subject_train.txt', header=None, squeeze=True)



y_train = pd.read_csv('UCI_HAR_Dataset/train/y_train.txt', names=['Activity'], squeeze=True)
y_train_labels = y_train.map({1: 'WALKING', 2:'WALKING_UPSTAIRS',3:'WALKING_DOWNSTAIRS',\
                       4:'SITTING', 5:'STANDING',6:'LAYING'})


# put all columns in a single dataframe
```

```
train = X_train

train['Activity'] = y_train

train['ActivityName'] = y_train_labels

train.sample()
```

Obtain the test data

```
# get the data from txt files to pandas dataffame

X_test = pd.read_csv('UCI_HAR_Dataset/test/X_test.txt',
delim_whitespace=True, header=None)

X_test.columns = features


# add subject column to the dataframe

X_test['subject'] =
pd.read_csv('UCI_HAR_Dataset/test/subject_test.txt', header=None,
squeeze=True)


# get y labels from the txt file

y_test = pd.read_csv('UCI_HAR_Dataset/test/y_test.txt',
names=['Activity'], squeeze=True)

y_test_labels = y_test.map({1: 'WALKING',
2:'WALKING_UPSTAIRS',3:'WALKING_DOWNSTAIRS',\

                       4:'SITTING', 5:'STANDING',6:'LAYING'})



# put all columns in a single dataframe

test = X_test

test['Activity'] = y_test

test['ActivityName'] = y_test_labels

test.sample()
```

Data Cleaning

- **Check for Duplicates**

```
print('No of duplicates in train: {}'.format(sum(train.duplicated())))

print('No of duplicates in test : {}'.format(sum(test.duplicated())))
```

- **Checking for NaN/null values**

```
print('We have {} NaN/Null values in train'.format(train.isnull().values.sum()))

print('We have {} NaN/Null values in test'.format(test.isnull().values.sum()))
```

- **Check for data imbalance**

```
sns.set_style('whitegrid')

plt.rcParams['font.family'] = 'Dejavu Sans'
```

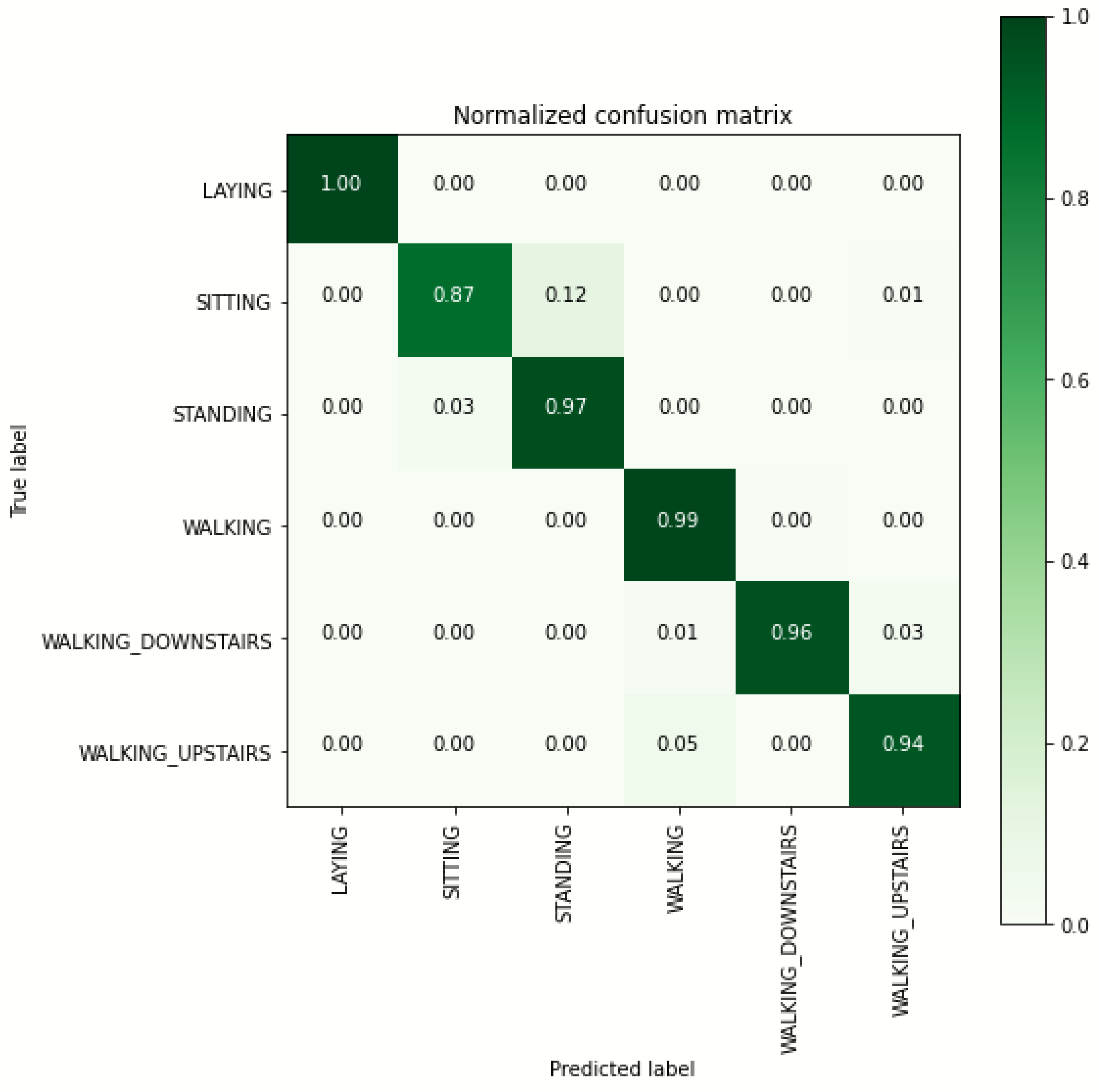
- **Changing feature names**

```
columns = train.columns


# Removing '()' from column names

columns = columns.str.replace('[()]','')

columns = columns.str.replace('[-]', '')

columns = columns.str.replace('[,]','')


train.columns = columns

test.columns = columns


test.columns
```

5.2.3. Prediction Models

- **Logistic Regression with Grid Search**

```
parameters = {'C':[0.01, 0.1, 1, 10, 20, 30], 'penalty':['l2','l1']}

log_reg = linear_model.LogisticRegression()

log_reg_grid = GridSearchCV(log_reg, param_grid=parameters, cv=3,
verbose=1, n_jobs=-1)

log_reg_grid_results = perform_model(log_reg_grid, X_train, y_train,
X_test, y_test, class_labels=labels)
```



	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
LAYING	1.00	0.00	0.00	0.00	0.00	0.00
SITTING	0.00	0.87	0.12	0.00	0.00	0.01
STANDING	0.00	0.03	0.97	0.00	0.00	0.00
WALKING	0.00	0.00	0.00	0.99	0.00	0.00
WALKING_DOWNSTAIRS	0.00	0.00	0.00	0.01	0.96	0.03
WALKING_UPSTAIRS	0.00	0.00	0.00	0.05	0.00	0.94

Fig 5.2.1.1 Normalized Confusion Matrix of Logistic Regression wit Grid Search

- **Linear SVC with GridSearch**

```
parameters = {'C':[0.125, 0.5, 1, 2, 8, 16]}
```

```
lr_svc = LinearSVC(tol=0.00005)

lr_svc_grid  =  GridSearchCV(lr_svc,  param_grid=parameters,  n_jobs=-1,
verbose=1)

lr_svc_grid_results   =   perform_model(lr_svc_grid,   X_train,   y_train,
X_test, y_test, class_labels=labels)
```

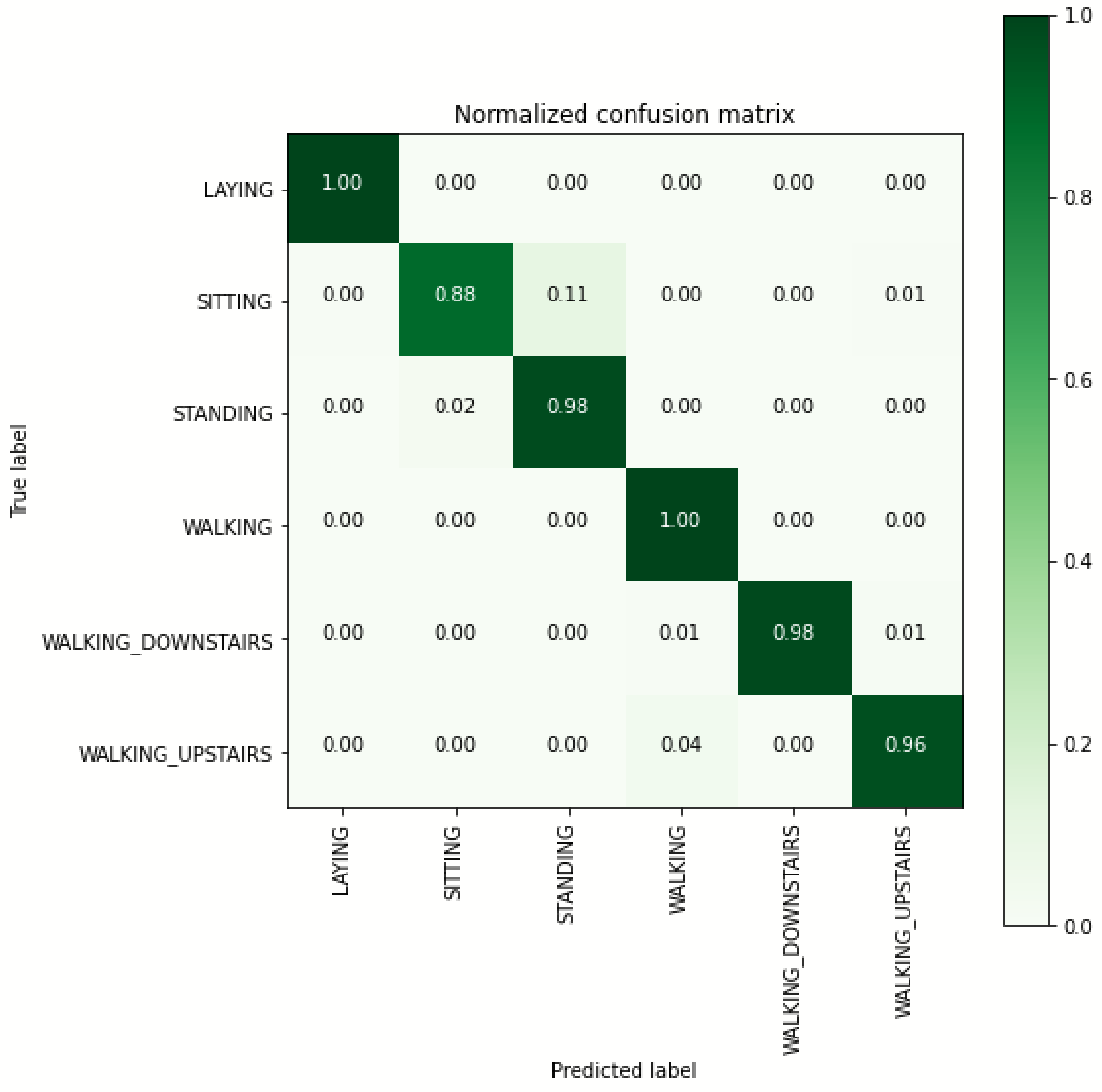


Fig 5.2.1.2 Normalized Confusion Matrix of Linear SVC with Grid Search

- **Kernel SVM with GridSearch**

```
from sklearn.svm import SVC

parameters = {'C':[2,8,16],\

              'gamma': [ 0.0078125, 0.125, 2]}

rbf_svm = SVC(kernel='rbf')
```

```
rbf_svm_grid = GridSearchCV(rbf_svm,param_grid=parameters, n_jobs=-1)

rbf_svm_grid_results  =  perform_model(rbf_svm_grid,  X_train,  y_train,
X_test, y_test, class_labels=labels)
```

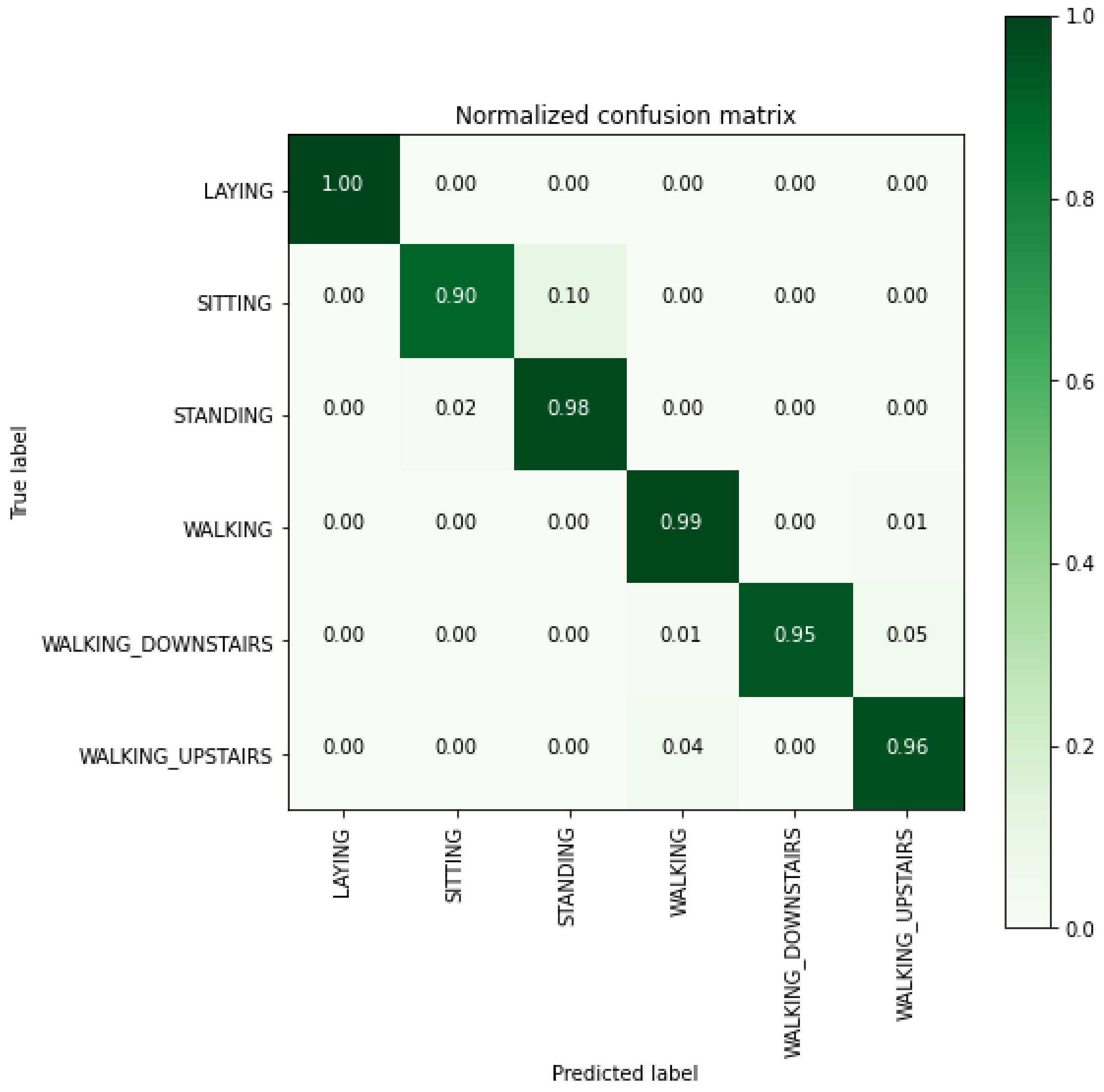


Fig 5.2.1.3 Normalized Confusion Matrix of Kernel SVM with Grid Seaarch

- **Decision Trees with GridSearchCV**

```
from sklearn.tree import DecisionTreeClassifier

parameters = {'max_depth':np.arange(3,10,2)}

dt = DecisionTreeClassifier()

dt_grid = GridSearchCV(dt,param_grid=parameters, n_jobs=-1)

dt_grid_results  =  perform_model(dt_grid,  X_train,  y_train,  X_test,
y_test, class_labels=labels)
```

```
print_grid_search_attributes(dt_grid_results['model'])
```

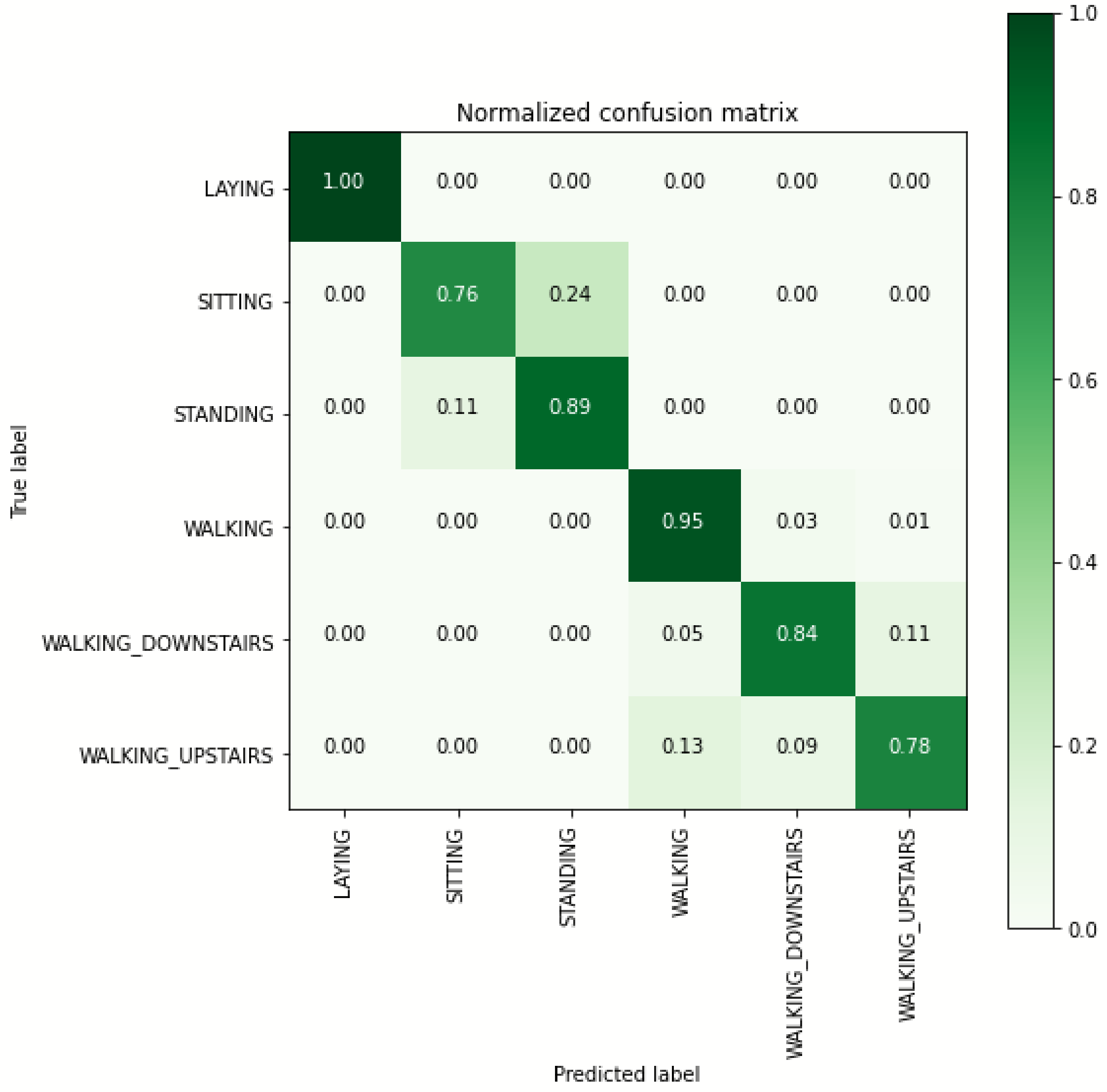
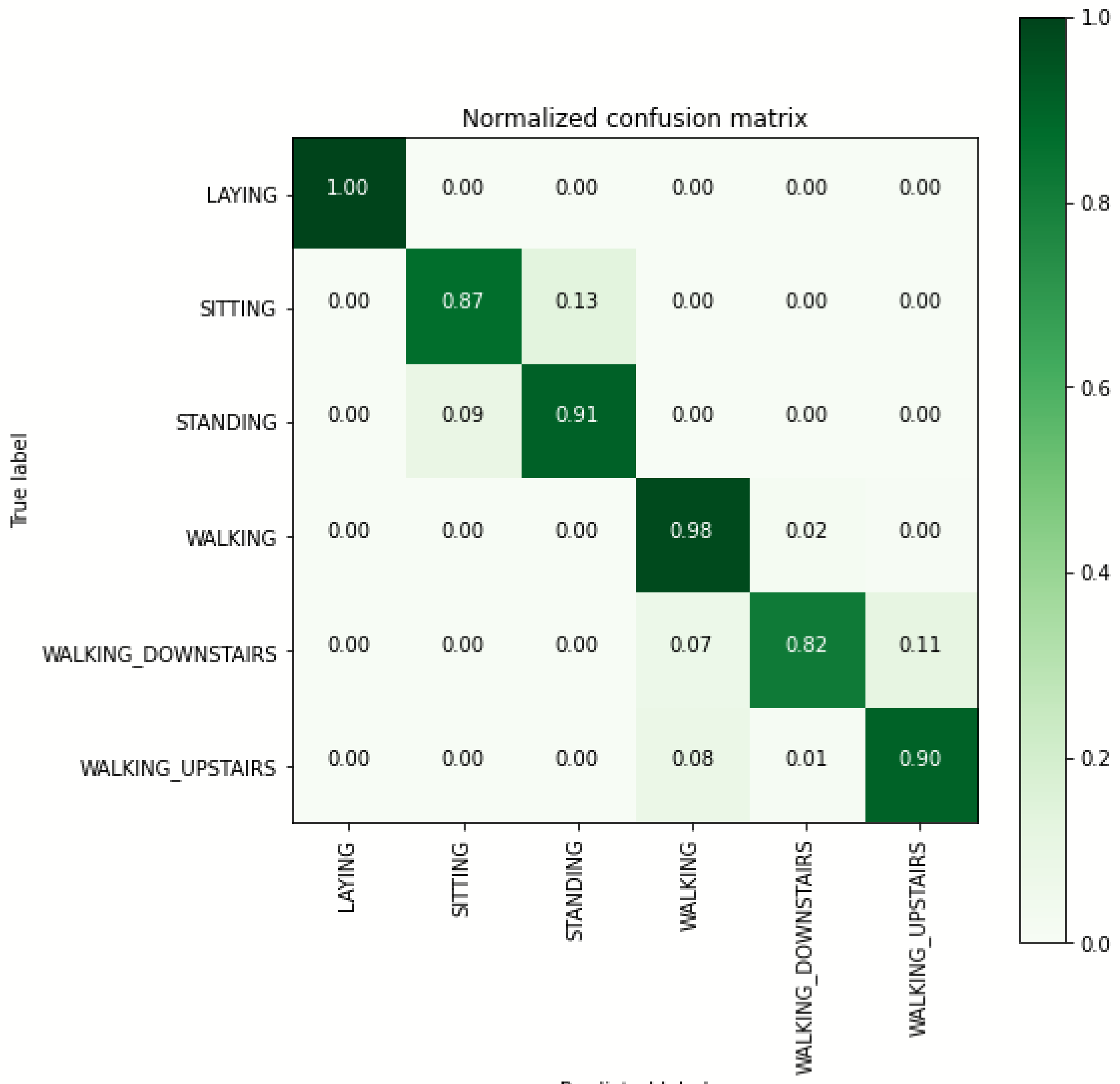


Fig 5.2.1.4 Normalized Confusion Matrix of Decision Trees with GridSearchCV

- **Random Forest Classifier with GridSearch**

```
from sklearn.ensemble import RandomForestClassifier

params         =         {'n_estimators':        np.arange(10,201,20),
'max_depth':np.arange(3,15,2)}

rfc = RandomForestClassifier()

rfc_grid = GridSearchCV(rfc, param_grid=params, n_jobs=-1)

rfc_grid_results  =  perform_model(rfc_grid,  X_train,  y_train,  X_test,
y_test, class_labels=labels)

print_grid_search_attributes(rfc_grid_results['model'])
```



True label \ Predicted label	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
LAYING	1.00	0.00	0.00	0.00	0.00	0.00
SITTING	0.00	0.87	0.13	0.00	0.00	0.00
STANDING	0.00	0.09	0.91	0.00	0.00	0.00
WALKING	0.00	0.00	0.00	0.98	0.02	0.00
WALKING_DOWNSTAIRS	0.00	0.00	0.00	0.07	0.82	0.11
WALKING_UPSTAIRS	0.00	0.00	0.00	0.08	0.01	0.90

Fig 5.2.1.5 Normalized Confusion Matrix of Random Forest Classifier with Grid Search

Model

```
model = Sequential()
model.add{
    tf.keras.layers.TimeDistributed{
        tf.keras.layers.Conv1D{
            filters=64, kernel_size=3,strides=2,
            activation="relu",
            input_shape=(n_length, n_features)
        }
    }
}
model.add{
    tf.keras.layers.TimeDistributed{
      tf.keras.layers.MaxPool1D(pool_size=2, strides=2)

}
}
model.add{
    tf.keras.layers.TimeDistributed{
        tf.keras.layers.Conv1DTranspose{
            filters=64, kernel_size=3,strides=2,
            activation="relu"
        }
    }
}
model.add{
    tf.keras.layers.TimeDistributed{
        tf.keras.layers.Flatten()
    }
}
```

```
model.add(Dropout(0.5))
model.add(LSTM(64))
model.add(Dense(64, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(n_classes, activation='softmax')
```

Compiling the model

```
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
from keras.utils.vis_utils import plot_model
plot_model(model, to_file='model_plot.png', show_shapes=True,
show_layer_names=True)
```

Training the model

```
history = model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=200)


import matplotlib.pyplot as plt
# list all data in history
print(history.history.keys())
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper left')

plt.show()

# summarize history for loss

plt.plot(history.history['loss'])

plt.plot(history.history['val_loss'])

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper left')

plt.show()
```

Confusion Matrix

```
print(confusion_matrix(Y_test, model.predict(X_test)))

cf_matrix = confusion_matrix(Y_test, model.predict(X_test))


import seaborn as sns

import matplotlib.pyplot as plt

ax = sns.heatmap(cf_matrix, annot=True, cmap='rocket',fmt='g')

ax.set_title('Confusion Matrix with labels\n\n');

ax.set_xlabel('\nPredicted Values')

ax.set_ylabel('Actual Values ');


# Ticket labels - List must be in alphabetical order

ax.xaxis.set_ticklabels(['LAYING','SITTING','STANDING',        'WALKING'
,'WALKING_DOWNSTAIRS', 'WALKING_UPSTAIRS'])

ax.yaxis.set_ticklabels(['LAYING','SITTING','STANDING',        'WALKING'
,'WALKING_DOWNSTAIRS', 'WALKING_UPSTAIRS'])


# Display the visualization of the Confusion Matrix.
```

```
plt.show()


def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

K Fold Validation

```
kf = KFold(5, shuffle=True, random_state=42)


oos_y = []

oos_pred = []

fold = 0

x = X_train

y = Y_train


# Must specify y StratifiedKFold for

for train, test in kf.split(x):

    fold+=1

    print(f"Fold #{fold}")


    x_train = x[train]

    y_train = y[train]

    x_test = x[test]

    y_test = y[test]

    model = Sequential()


    model.add(

        tf.keras.layers.TimeDistributed(

            tf.keras.layers.Conv1D(

                filters=128, kernel_size=6,strides=2,

                activation="relu",

                input_shape=(n_length, n_features)

            )

        )

    )
```

```
model.add(
    tf.keras.layers.TimeDistributed(
      tf.keras.layers.MaxPool1D(pool_size=2, strides=2)
    )
)




model.add(
    tf.keras.layers.TimeDistributed(
        tf.keras.layers.Conv1DTranspose(
            filters=128, kernel_size=6,strides=2,
            activation="relu"
        )
    )
)



model.add(
    tf.keras.layers.TimeDistributed(
        tf.keras.layers.Flatten()
    )
)

model.add(Dropout(0.5))

model.add(LSTM(64))
```

```
    # Adding a dense layer with relu activation
    model.add(Dense(64, activation='relu'))

    # Adding a dense layer with relu activation
    model.add(Dense(64, activation='relu'))

    model.add(Dense(n_classes, activation='softmax'))

    model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
                     model.fit(x_train,y_train,      batch_size=128,
validation_data=(x_test,y_test),verbose=1,
              epochs=EPOCHS)

    pred = model.predict(x_test)

    oos_y.append(y_test)
    oos_pred.append(pred)

    # Measure this fold's accuracy
    score = model.evaluate(x_test,y_test)
    print(f"Fold score (Accuracy): {score}")

# Build the oos prediction list and calculate the error.
oos_y = np.concatenate(oos_y)
oos_pred = np.concatenate(oos_pred)
score=metrics.accuracy_score(np.argmax(oos_pred,axis=1),np.argmax(oos_y
,axis=1))
print(f"Final, out of sample score (Accuracy): {score}")
```

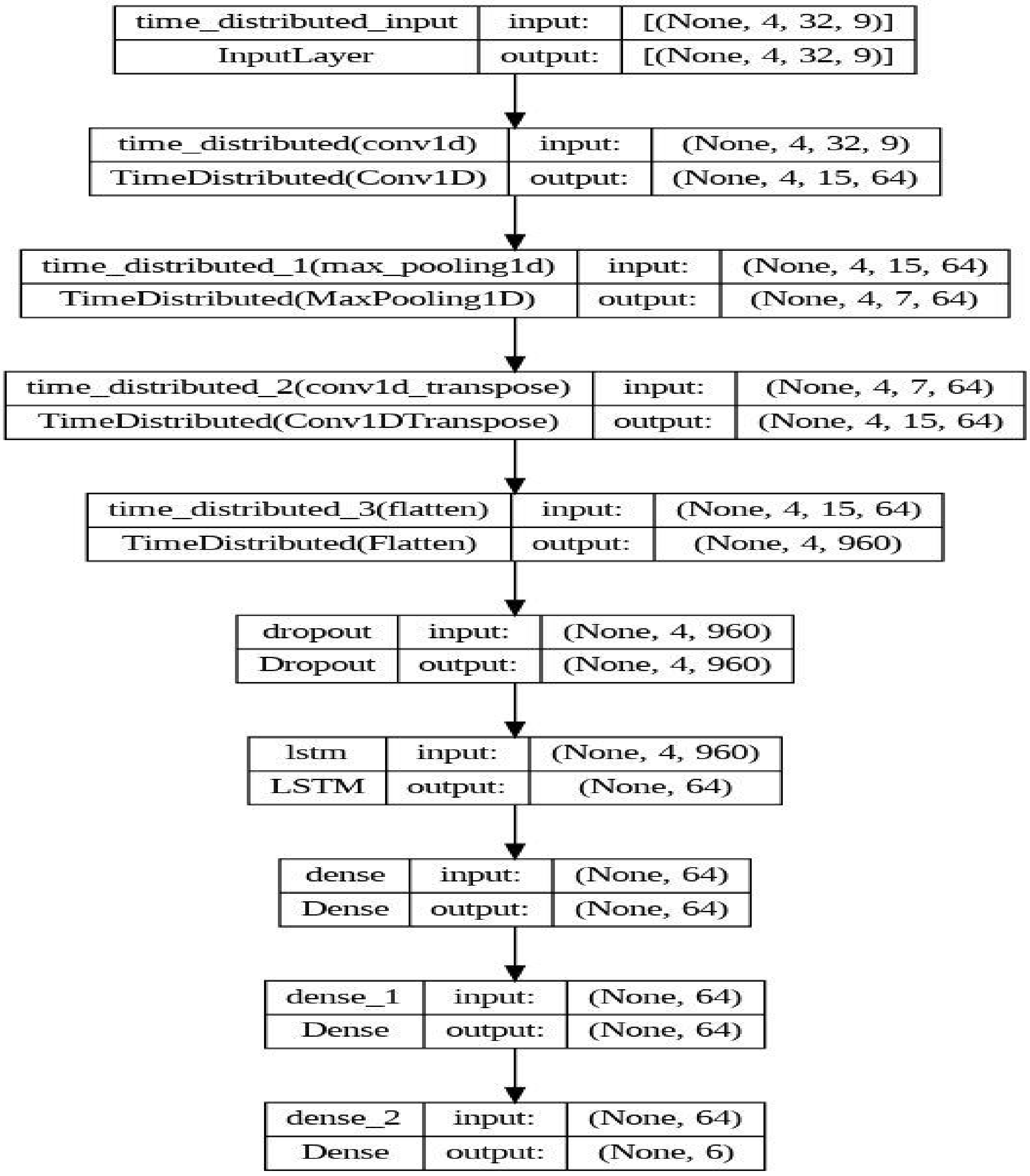

time_distributed_input | input: | [(None, 4, 32, 9)]
InputLayer | output: | [(None, 4, 32, 9)]
time_distributed(conv1d) | input: | (None, 4, 32, 9)
TimeDistributed(Conv1D) | output: | (None, 4, 15, 64)
time_distributed_1(max_pooling1d) | input: | (None, 4, 15, 64)
TimeDistributed(MaxPooling1D) | output: | (None, 4, 7, 64)
time_distributed_2(conv1d_transpose) | input: | (None, 4, 7, 64)
TimeDistributed(Conv1DTranspose) | output: | (None, 4, 15, 64)
time_distributed_3(flatten) | input: | (None, 4, 15, 64)
TimeDistributed(Flatten) | output: | (None, 4, 960)
dropout | input: | (None, 4, 960)
Dropout | output: | (None, 4, 960)
lstm | input: | (None, 4, 960)
LSTM | output: | (None, 64)
dense | input: | (None, 64)
Dense | output: | (None, 64)
dense_1 | input: | (None, 64)
Dense | output: | (None, 64)
dense_2 | input: | (None, 64)
Dense | output: | (None, 6)

Fig 5.2.1 Attributes under flow

5.3 Results with Discussion

The network is trained using the training set where the error is being calculated by taking the actual output and the predicted output. The errors are then back propagated using the Adam optimizer in sequence of the layers to update the hyperparameters of the network.

To perform the experiment, the first two datasets are divided into two different groups. 70% of volunteers are selected for training and 30% are used to test the proposed HAR solution. Therefore, data from the same subject is not included in both the training and test datasets. In our experiment, we use a simple 5-way cross-validation test to generate multiple training and validation splits from a training set. This is because cross-validation is less computationally complex than other methods such as leave-one-out cross-validation.

We have used the data from one subject for the test and the data from the other subject for training. This cross-subject test is more rigorous because the test data is hidden from the model, making it a more realistic framework for verifying the generalization capabilities of the model. All datasets are used to perform 1D convolution on the input layer of the CNN. In our experiments, ReLU is used as the activation function for a convolutional layer with a kernel size of 3, a stride of 2, and a filter size of 64. The learning rate (alpha) is set to 0.001. The optimizer reduces the loss function by updating and calculating network parameters that affect the training and output process of the model to approach or reach optimal values.

Using the UCI dataset to perform comprehensive experiments on a variety of ML and DL approaches, including, All approaches are used in our experiments to demonstrate the effectiveness of the proposed method in the same experimental settings. The computational time of the proposed model is very competitive with the computational time of the DL approach above in the same computational environment. In addition, the calculation time of Conv-AE with LSTM is very competitive with the calculation time of the state-of-the-art approach proposed in one of the previous works. With CNN, the training and test calculation times are 3.4374 seconds and 371.6 ms, respectively. Using the UCI dataset, the test accuracy of Conv-AE with LSTM is 93.42%, which is much higher than other common DL approaches.

Figure shows the detailed classification results of the proposed model. This proposed model uses both convolutional AEs and LSTMs in combination, so the F1 values for activities such as walking, stairs down, and stairs up are 99%, 100%, and 94%, respectively. Therefore, it can be concluded that the proposed method can identify similar activity patterns very efficiently. This cannot be achieved with the CNN method described in the HAR literature alone. Similarly, if you include static features such as "sitting," "lying," and "standing," the F1 values will be 97%, 98%, and 98%, respectively, and a CNN-based HAR solution. From the experimental results, it can be easily concluded that the proposed method can not only efficiently distinguish between static and dynamic activities, but also efficiently identify similar activity patterns. Figure 4 and Figure 5 shows the confusion matrix for the various activities in the test set. It also analyzes the activity detection accuracy of the proposed ConvAE with LSTM method.

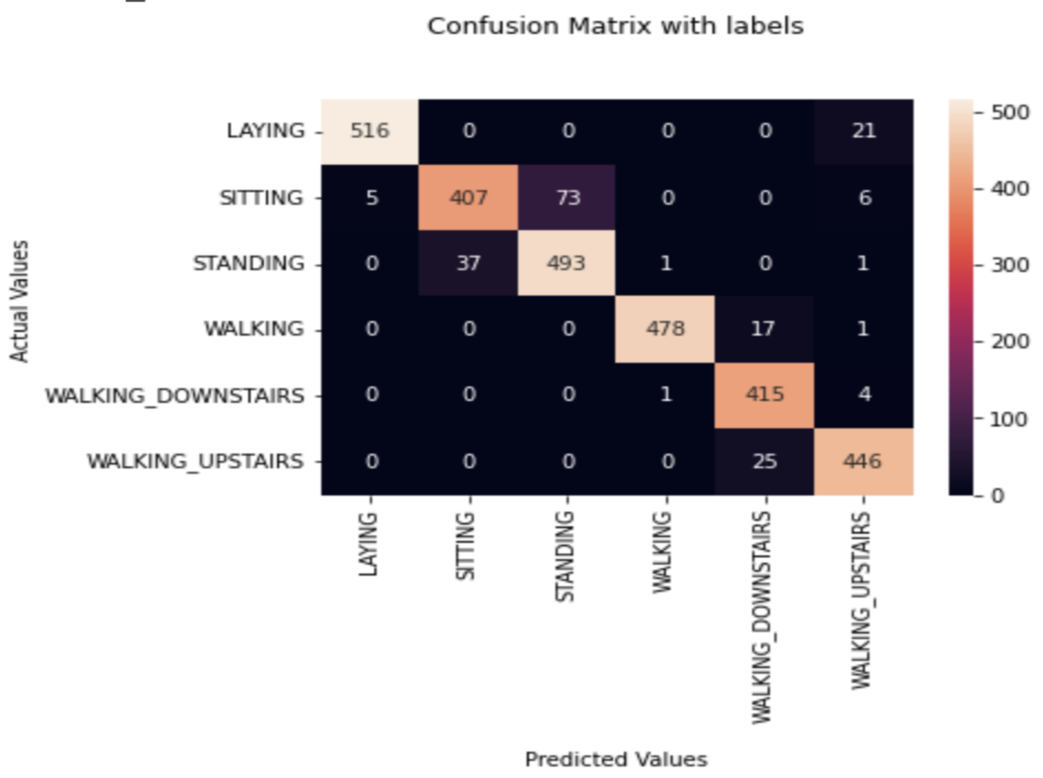
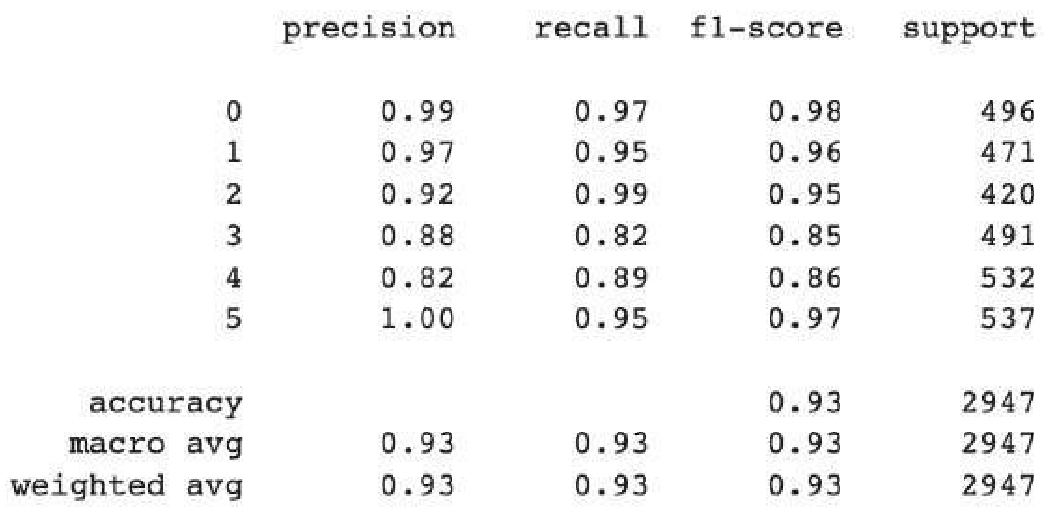


Fig 5.3.1 Confusion Matrix



```
              precision    recall  f1-score   support

           0       0.99      0.97      0.98       496
           1       0.97      0.95      0.96       471
           2       0.92      0.99      0.95       420
           3       0.88      0.82      0.85       491
           4       0.82      0.89      0.86       532
           5       1.00      0.95      0.97       537

    accuracy                           0.93      2947
   macro avg       0.93      0.93      0.93      2947
weighted avg       0.93      0.93      0.93      2947
```

Fig 5.3.2 Model Performance

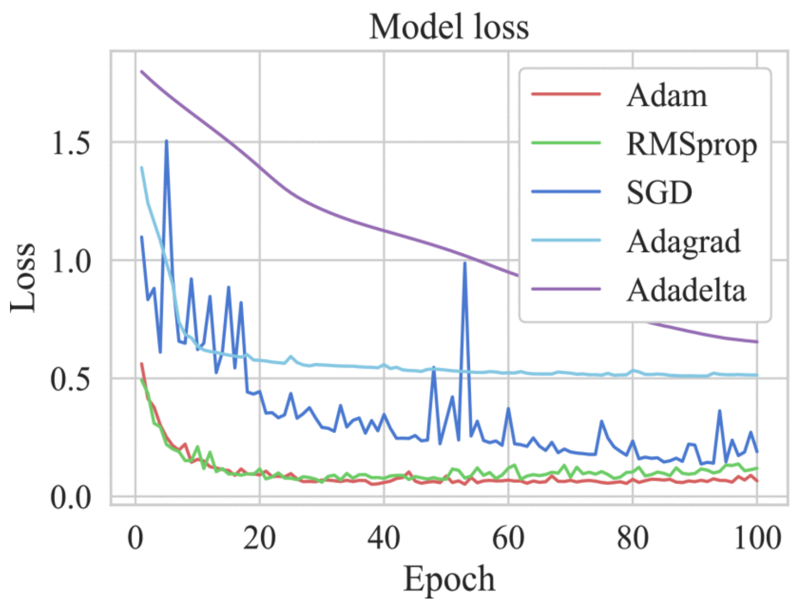


Fig 5.3.3 Model Loss

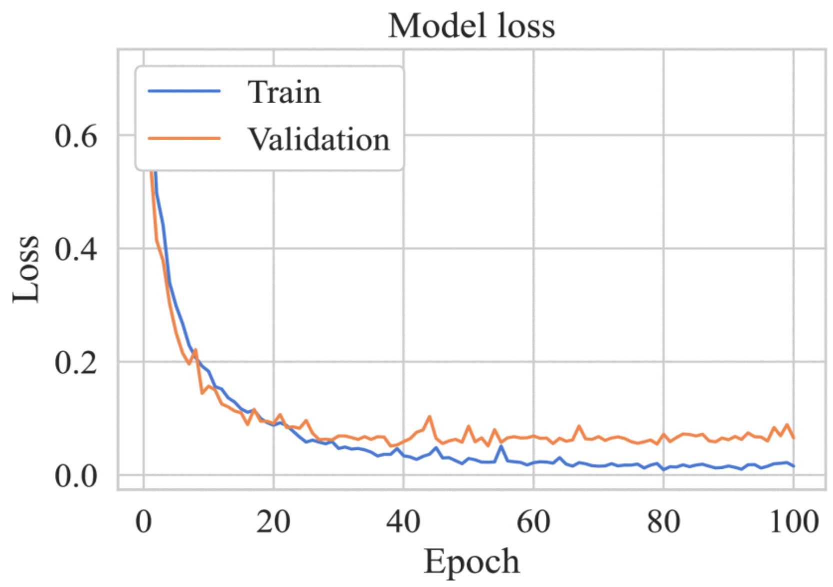


Fig 5.3.4 Model Loss

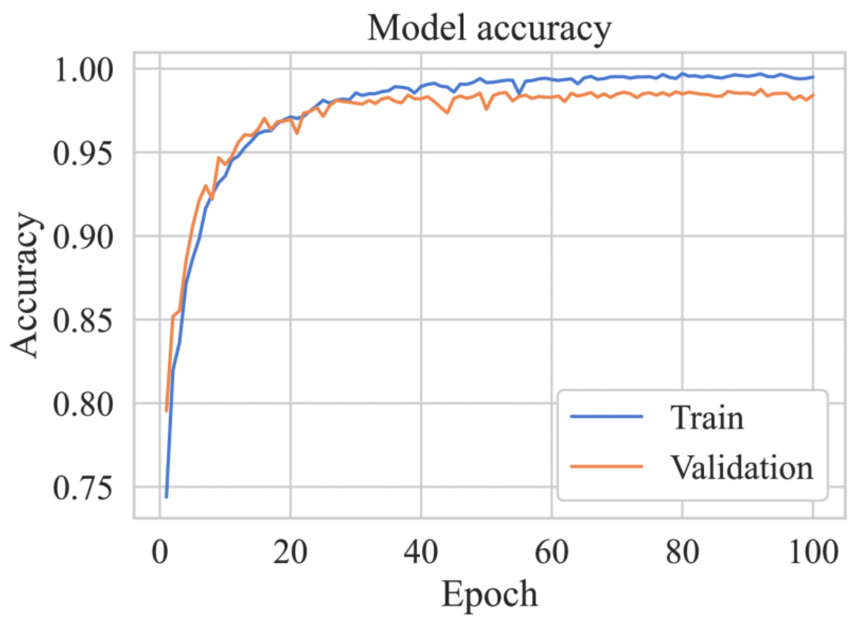
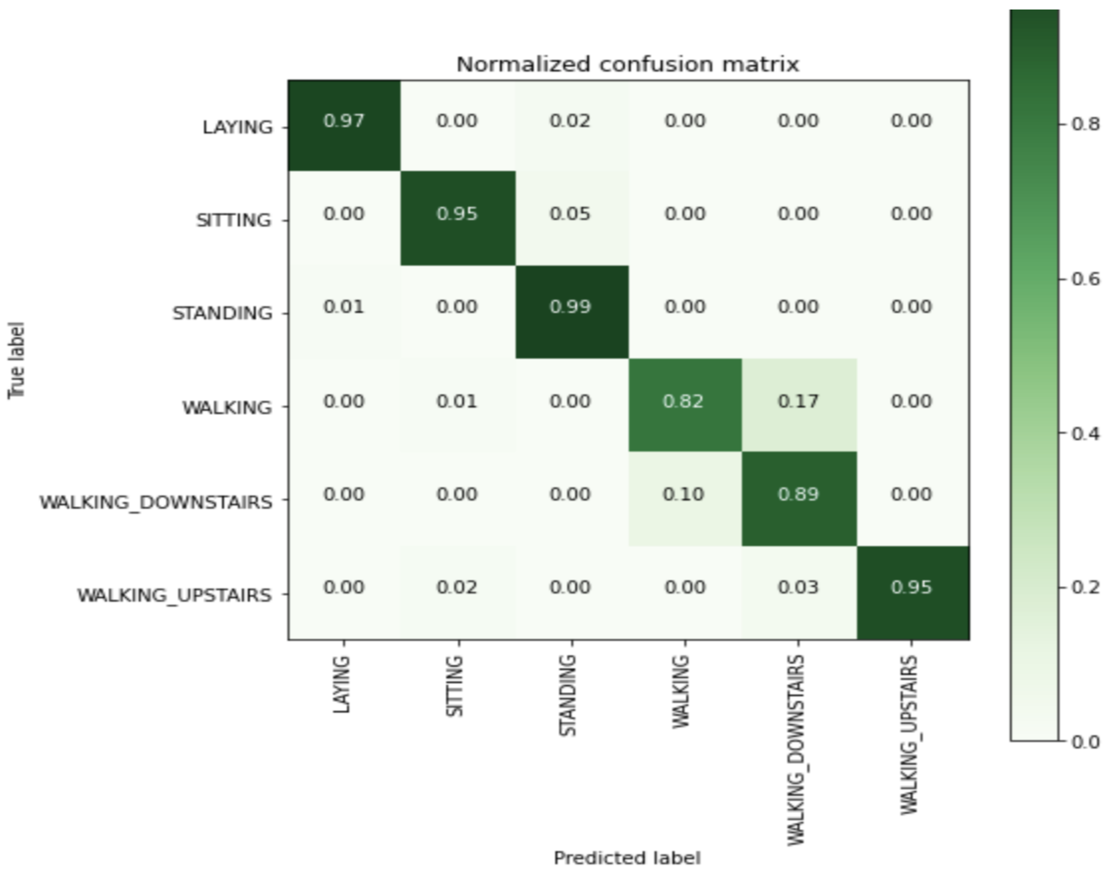


Fig 5.3.5 Model Accuracy



	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
LAYING	0.97	0.00	0.02	0.00	0.00	0.00
SITTING	0.00	0.95	0.05	0.00	0.00	0.00
STANDING	0.01	0.00	0.99	0.00	0.00	0.00
WALKING	0.00	0.01	0.00	0.82	0.17	0.00
WALKING_DOWNSTAIRS	0.00	0.00	0.00	0.10	0.89	0.00
WALKING_UPSTAIRS	0.00	0.02	0.00	0.00	0.03	0.95

Fig 5.3.6 Normalized Confusion Matrix

SYSTEM TESTING

6.System Testing

6.1 Introduction

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, subassemblies, 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.

6.2 TESTING METHODOLOGIES

The following are the Testing Methodologies:

- Unit Testing.
- Integration Testing.
- User Acceptance Testing.
- Output Testing.
- Validation Testing.

6.2.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module’s control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing. During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All important processing path are tested for the expected results. All error handling paths are also tested.

6.2.2 Integration Testing

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

The following are the types of Integration Testing:

1.Top Down Integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner. In this method, the software is tested from the main module and individual stubs are replaced when the test proceeds downwards.

2. Bottom-up Integration

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

♠The low-level modules are combined into clusters that perform a specific Software sub-function.

♠ A driver (i.e.) the control program for testing is written to coordinate test case input and output.

♠ The cluster is tested.

♠ Drivers are removed and clusters are combined moving upward in the program structure

The bottom up approaches test each module individually and then each module is integrated with a main module and tested for functionality.

6.2.3 User Acceptance Testing

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

6.2.4 Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format

6.2.5 Validation Checking

Validation checks are performed on the following fields.

Text Field

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entries always flash and error messages.

Numeric Field

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error message. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to a test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested. A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

Preparation of Test Data

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

Using Live Test Data

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves. It is difficult to obtain live data in

sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations of formats that can enter the system. This bias toward typical values then does not provide

a true systems test and in fact ignores the cases most likely to cause system failure.

Using Artificial Test Data

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program. The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications. The package "Virtual Private Network" has satisfied all the requirements specified as per software requirement specification and was accepted.

USER TRAINING

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

MAINTENANCE

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user's requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing is simple and easy to understand which will make maintenance easier.

6.3 TESTING STRATEGY

A strategy for system testing integrates system test cases and design techniques into a well planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation .A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements. Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

SYSTEM TESTING

Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

UNIT TESTING

In unit testing different modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goals to test the internal logic of the modules. Using the detailed design description as a guide, important Conrail paths are tested to uncover errors within the boundary of the modules. This testing is carried out during the programming stage itself. In this type of testing step, each module was found to be working satisfactorily as regards to the expected output from the module. In Due Course, latest technology advancements will be taken into consideration. As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this. The future holds a lot to offer to the development and refinement of this project.

CONCLUSION

7.Conclusion

In conclusion, we proposed a unified architecture.This architecture combines the abilities of CNN, AE and LSTM. CNN with its automatic feature extraction and AE with its ability of efficient dimensionality reduction and LSTM with the ability to preserve sequence information. Currently we have primarily got the performance results of the proposed architecture on UCI-HAR dataset. Further, we want to study the performance of the model in other datasets for this research work to be presented.

REFERENCES

8. References

1. D.Anguita,A.Ghio,L.Oneto,X.Parra,andJ.L.Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones," in *Proc. Eur. Symp. Artif. Neural Netw., Comput. Intell. Mach. Learn.*, 2013, pp. 1–3.
2. O.S.EyobuandD.Han, "Feature Representation Data Augmentation for human activity classification based on wearable IMU sensor data using a deep LSTM neural network," *Sensors*, vol. 18, no. 9, p. 2892, Aug. 2018.
3. D.ThakurandS.Biswas, "Smartphone Based Human Activity Monitoring and recognition using ML and DL: A comprehensive survey," *J. Ambient Intell. Humanized Comput.*, vol. 11, no. 11, pp. 5433–5444, Mar. 2020.
4. Z.Chen,C.Jiang,andL.Xie, "AnovelensembleELMforhumanactiv- ity recognition using smartphone sensors," *IEEE Trans. Ind. Informat.*, vol. 15, no. 5, pp. 2691–2699, May 2019.
5. E.Bulbul,A.Cetin,andI.A.Dogru, "Human Activity Recognition Using smartphones," in *Proc. 2nd Int. Symp. Multidisciplinary Stud. Innov. Tech. (ISMSIT)*, Oct. 2018, pp. 1–6.
6. R.-A.Voicu,C.Dobre,L.Bajenaru,andR.-I.Ciobanu, "Humanphysical activity recognition using smartphone sensors," *Sensors*, vol. 19, no. 3, p. 458, 2019.
7. D.Anguita,A.Ghio,L.Oneto,X.Parra,andJ.L.Reyes-Ortiz, "Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine," in *Proc. Int. Workshop Ambient Assist. Living*, 2012, pp. 216–223.
8. L. F. Mejia-Ricart, P. Helling, and A. Olmsted, "Evaluate action primitives for human activity recognition using unsupervised learning approach," in *Proc. 12th Int. Conf. Internet Technol. Secured Trans. (ICITST)*, Dec. 2017, pp. 186–188.
9. O.S.EyobuandD.Han, "Feature Representation Data Augmentation for human activity classification based on wearable IMU sensor data using a deep LSTM neural network," *Sensors*, vol. 18, no. 9, p. 2892, Aug. 2018.
10. Y. Tian and W. Chen, "MEMS-based human activity recognition using smartphone," in *Proc. 35th Chin. Control Conf. (CCC)*, Jul. 2016, pp. 3984–3989.
11. H. Mazaar, E. Emary, and H. Onsi, "Evaluation of feature selection on human activity recognition," in *Proc. IEEE 7th Int. Conf. Intell. Comput. Inf. Syst. (ICICIS)*, Dec. 2015, pp. 591–599.
12. J. Suto, S. Oniga, and P. P. Sitar, "Comparison of wrapper and filter feature selection algorithms on human activity recognition," in *Proc. 6th Int. Conf. Comput. Commun. Control (ICCCC)*, May 2016, pp. 124–129.
13. C. A. Ronao and S.-B. Cho, "Human activity recognition with smart- phone sensors using deep learning neural networks," *Expert Syst. Appl.*, vol. 59, pp. 235–244, Oct. 2016.

14. Zeng, M., Nguyen, L.T., Yu, B., Mengshoel, O.J., Zhu, J., Wu, P., & Zhang, J.Y. (2014). Convolutional Neural Networks for human activity recognition using mobile sensors. *6th International Conference on Mobile Computing, Applications and Services*, 197-205.
15. P.Zappi,C.Lombriser,T.Stiefmeier,E.Farella,D.Roggen,L.Benini, and G. Tröster, "Activity recognition from on-body sensors: Accuracy- power trade-off by dynamic sensor selection," in *Wireless Sen- sor Networks*, R. Verdone, Ed. Berlin, Germany: Springer, 2008, pp. 17–33.
16. R. Chavarriaga, H. Sagha, A. Calatroni, S. T. Digumarti, G. Tröster, J. R. Del Millán, and D. Roggen, "The opportunity challenge: A bench- mark database for on-body sensorbased activity recognition," *Pattern Recognit. Lett.*, vol. 34, no. 15, pp. 2033–2042, Jan.
    2009.
17. J.W.Lockhart,G.M.Weiss,J.C.Xue,S.T.Gallagher,A.B.Grosner,and T. T. Pulickal, "Design considerations for the WISDM smart phone-based sensor mining architecture," in *Proc. 5th Int. Workshop Knowl. Discovery Sensor Data*, New York, NY, USA, 2011, pp. 25–33.
18. C. Ronao and S.-B. Cho, "Deep convolutional neural networks for human activity recognition with smartphone sensors," *Lect. Notes Comput. Sci.*, vol. 9492, pp. 46–53, Nov. 2015.
19. D.Anguita,A.Ghio,L.Oneto,X.Parra,andJ.L.Reyes-Ortiz, "Public domain dataset for human activity recognition using smartphones," in *Proc. Eur. Symp. Artif. Neural Netw., Comput. Intell.*
    Mach. Learn., 2013, pp. 1–3.
20. D. Ravì, C. Wong, B. Lo, and G.-Z. Yang, "A deep learning approach to on-node sensor data analytics for mobile or wearable devices," *IEEE J. Biomed. Health Inform.*, vol. 21, no. 1, pp. 56–64, Jan. 2017.
21. J.R.Kwapisz,G.M.Weiss,andS.A.Moore,"Activity Recognition Using cell phone accelerometers," *ACM SIGKDD Explor. Newslett.*, vol. 12, no. 2, pp. 74–82, May 2011.
22. M.Bachlin,M.Plotnik,D.Roggen,I.Maidan,J.M.Hausdorff,N.Giladi, and G. Troster, "Wearable assistant for Parkinson's disease patients with the freezing of gait symptom," *IEEE Trans. Inf. Technol. Biomed.*, vol. 14, no. 2, pp. 436–446, Nov. 2010.
23. X. Jiang, Y. Lu, Z. Lu, and H. Zhou, "Smartphone-based human activity recognition using CNN in frequency domain," in *Proc. APWeb/WAIM Workshops*, 2018, pp. 101–110.
24. I. Andrey, "Real-time human activity recognition from accelerometer data using convolutional neural networks," *Appl. Soft Comput.*, vol. 62, pp. 915–922, Jan. 2017.
25. M.Gholamrezaii and S.M.Taghi Almodarresi, "Humanactivityrecog- nition using 2D convolutional neural networks," in *Proc. 27th Iranian Conf. Electr. Eng. (ICEE)*, Apr. 2019, pp. 1682–1686.

26. S. Wan, L. Qi, X. Xu, C. Tong, and Z. Gu, "Deep learning models for real-time human activity recognition with smartphones," *Mobile Netw. Appl.*, vol. 25, no. 2, pp. 743–755, Apr. 2020.
27. A. Reiss and D. Stricker, "Introducing a new benchmarked dataset for activity monitoring," in *Proc. 16th Int. Symp. Wearable Comput.*, Jun. 2012, pp. 108–109.
28. Y. Li, D. Shi, B. Ding, and D. Liu, "Unsupervised feature learning for human activity recognition using smartphone sensors," in *Mining Intelligence and Knowledge Exploration* (Lecture Notes In Computer Science), vol. 8891. Cham, Switzerland: Springer, 2014, pp. 99–107.
29. M. Hasan and A. K. Roy-Chowdhury, "Continuous learning of human activity models using deep nets," in *Computer Vision*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham, Switzerland: Springer, 2014, pp. 705–720.
30. B. Almaslukh, J. AlMuhtadi, and A. Artoli, "An effective deep autoen- coder approach for online smartphone based human activity recogni- tion," *Int. J. Comput. Sci. Netw. Secur.*, vol. 17, no. 4, pp. 160–165, 2017.
31. D. Balabka, "Semi-supervised learning for human activity recognition using adversarial autoencoders," in *Proc. UbiComp/ISWC*, New York, NY, USA, 2019, pp. 685–688.
32. X. Gao, H. Luo, Q. Wang, F. Zhao, L. Ye, and Y. Zhang, "A human activity recognition algorithm based on stacking denoising autoencoder and lightGBM," *Sensors*, vol. 19, no. 4, p. 947, 2019.
33. T.OzcanandA.Basturk, "Humanactionrecognitionwithdeeplearning and structural optimization using a hybrid heuristic algorithm," *Cluster Comput.*, vol. 23, no. 4, pp. 2847–2860, Dec. 2020.
34. C. Geng and J. Song, "Human action recognition based on convolutional neural networks with a convolutional auto-encoder," in *Proc. 5th Int. Conf. Comput. Sci. Autom. Eng.*, 2016, pp. 933–938.
35. T.Zebin,M.Sperrin,N.Peek,andA.J.Casson, "Humanactivityrecogni- tion from inertial sensor timeseries using batch normalized deep LSTM recurrent networks," in *Proc. 40th Annu. Int. Conf. Eng. Med. Biol. Soc. (EMBC)*, 2018, pp. 1–4.
36. S. Yu and L. Qin, "Human activity recognition with smartphone inertial sensors using bidir-LSTM networks," in *Proc. 3rd Int. Conf. Mech., Control Comput. Eng. (ICMCCE)*, Sep. 2018, pp. 219– 224.
37. Y. Zhao, R. Yang, G. Chevalier, X. Xu, and Z. Zhang, "Deep residual bidir-lstm for human activity recognition using wearable sensors," *Math. Problems Eng.*, vol. 2018, Dec. 2018, Art. no. 7316954.
38. H. Wang, J. Zhao, J. Li, L. Tian, P. Tu, T. Cao, Y. An, K. Wang, and S. Li, "Wearable sensor-based human activity recognition using hybrid deep learning techniques," *Secur. Commun. Netw.*, vol.

2020, p. 12, Jan. 2020.

39. R.MutegekiandD.S.Han, "ACNN-LSTMapproachtohumanactivity recognition," in *Proc. Int. Conf. Artif. Intell. Inf. Commun. (ICAIIC)*, Feb. 2020, pp. 362–366.
40. H. Zou, Y. Zhou, J. Yang, H. Jiang, L. Xie, and C. J. Spanos, "DeepSense: Device-free human activity recognition via autoencoder long-term recur- rent convolutional network," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.
41. C. Schu ̈ldt, I. Laptev, and B. Caputo. Recognizing human actions: a local SVM approach. In *Proc.*

    ICPR, 2004.

APPENDIX

9. Appendix

9.1 Introduction to python

Python is an open source , high-level programming language developed by Guido van Rossum in the late 1980s and presently administered by Python Software Foundation. It came from the ABC language that he helped create early on in his career. Python is a powerful language that you can use to create games, write GUIs and develop web applications.

It is a high-level language. Reading and writing codes in Python is much like reading and writing regular English statements. Because they are not written in machine readable language, Python programs need to be processed before machines can run them. Python is an interpreted language. This means that every time a program is run, its interpreter runs through the code and translates it into machine readable byte code.

Python is an object-oriented language that allows users to manage and control data structures or objects to create and run programs. Everything in Python is latest version of Python, in fact, first class. All objects, data types, functions, methods, and classes take equal position in Python. Programming languages are created to satisfy the needs of programmers and users for an effective tool to develop applications that impact lives, lifestyles, economy, and society. They help make lives better by increasing productivity, enhancing communication, and improving efficiency. Languages die and become obsolete when they fail to live up to expectations and are replaced and superseded by languages that are more powerful.

Python is a programming language that has stood the test of time and has remained relevant across industries and businesses and among programmers, and individual users. It is a living, thriving, and highly useful language that is highly recommended as a first programming language for those who want to dive into and experience programming. Advantages of Using Python Here are reasons why you would prefer to learn and use Python over other high-level languages

Readability

Python programs use clear, simple, and concise instructions that are easy to read even by those who have no substantial programming background. Programs written in Python are, therefore, easier to maintain, debug, or enhance.

Higher productivity

Codes used in Python are considerably shorter, simpler, and less verbose than other high level programming languages such as Java and C++. In addition, it has well-designed built-in features and standard library as well as access to third party modules and source libraries. These features make programming in Python more efficient.

Less learning time

Python is relatively easy to learn. Many find Python a good first language for learning programming because it uses simple syntax and shorter codes. Python works on Windows, Linux/UNIX, Mac OS X, other operating systems and small form devices. It also runs on microcontrollers used in appliances, toys, remote controls, embedded devices, and other similar devices.

9.2 Introduction to Machine Learning

Machine Learning is a system of computer algorithms that can learn from examples through self-improvement without being explicitly coded by a programmer. Machine learning is a part of artificial intelligence which combines data with statistical tools to predict an output that can be used to make actionable insights.

The breakthrough comes with the idea that a machine can singularly learn from the data (i.e., an example) to produce accurate results. Machine learning is closely related to data mining and Bayesian predictive modeling. The machine receives data as input and uses an algorithm to formulate answers.

Machine learning can be grouped into two broad learning tasks:

1. Supervised ML
2. Unsupervised ML

There are many other algorithms.

1. **Supervised learning:**

   An algorithm uses training data and feedback from humans to learn the relationship between given inputs to a given output. For instance, a practitioner can use marketing expenses and weather forecasts as input data to predict the sales of cans. You can use supervised learning when the output data is known. The algorithm will predict new data.

   There are two categories of supervised learning:

   1. Classification task
   2. Regression task

Classification

Imagine you want to predict the gender of a customer for a commercial. You will start gathering data on the height, weight, job, salary, purchasing basket, etc. from your customer database. You know the gender of each of your customer, it can only be male or female. The objective of the classifier will be to assign a probability of being a male or a female (i.e., the label) based on the information (i.e., features you have collected). When the model learned how to recognize male or female, you can use new data to make a prediction. For instance, you just got new information from an unknown customer, and you want to know if it is a male or female. If the classifier predicts male = 70%, it means the algorithm is sure at 70% that this customer is a male, and 30% it is a female.The label can be for two or more classes. The above Machine learning example has only two classes, but if a classifier needs to predict an object, it has dozens of classes (e.g., glass, table, shoes, etc. each object represents a class)

Regression

When the output is a continuous value, the task is a regression. For instance, a financial analyst may need to forecast the value of a stock based on a range of feature like equity, previous stock performances, macroeconomics index. The system will be trained to estimate the price of the stocks with the lowest possible error.

2.Unsupervised learning :

In unsupervised learning, an algorithm explores input data without being given an explicit output variable (e.g., explores customer demographic data to identify patterns). You can use it when you do not know how to classify the data, and you want the algorithm to find patterns and classify the data for you.

9.2.1 Convolution

Convolution is a mathematical operation that allows the merging of two sets of information. In the case of CNN, convolution is applied to the input data to filter the information and produce a feature map. This filter is also called a kernel, or feature detector.

Convolution is an orderly procedure where two sources of information are intertwined; it's an operation that changes a function into something else. Convolutions have been used for a long time typically in image processing to blur and sharpen images, but also to perform other operations. (e.g. enhance edges and emboss) CNNs enforce a local connectivity pattern between neurons of adjacent layers.

9.2.2 Pooling

Pooling is nothing other than down sampling of an image. The most common pooling layer filter is of size 2x2, which discards three forth of the activations. Role of pooling layer is to reduce the resolution of the feature map but retaining features of the map required for classification through translational and rotational invariants. In addition to spatial invariance robustness, pooling will reduce the computation cost by a great deal.

Backpropagation is used for training of pooling operation. It again helps the processor to process things faster.

9.2.3 Deconvolution

Deconvolution is a quantitative approach that uses the picture as an estimate of the real specimen intensity and conducts the mathematical inverse of the imaging process to generate an improved estimate of the image intensity using a formula for the point spread function.

The deconvolution operation is an upsampling procedure that both upsamples feature maps and keeps the connectivity pattern. The deconvolutional layers essentially increase and densify the input by employing convolution-like procedures with numerous filters. Deconvolution, unlike previous scaling algorithms, has trainable parameters. During network training, the weights of deconvolutional layers are constantly updated and refined. It is accomplished by inserting zeros between the consecutive neurons in the receptive field on the input side, and then one convolution kernel with a unit stride is used on top.

9.2.4 Flatten

Flattening involves taking the pooled feature map that is generated in the pooling step and transforming it into a one-dimensional vector. The reason why we transform the pooled feature map into a one-dimensional vector is because this vector will now be fed into an artificial neural networks.

Once the pooled featured map is obtained, the next step is to flatten it. Flattening involves transforming the entire pooled feature map matrix into a single column which is then fed to the neural network for processing.

9.2.5 Long / Short Term Memory

Schmidhuber and Hochreiter in 1997 built a neural network which is called long short term memory networks (LSTMs). Its main goal is to remember things for a long time in a memory cell that is explicitly defined. Previous values are stored in the memory cell unless told to forget the values by “forget gate”. New stuff is added through the “input gate” to the memory cell, and it is passed to the next hidden state from the cell along the vectors which is decided by the “output gate”. Composition of primitive music, writing like Shakespeare, or learning complex sequences are some of the applications of LSTMs.

Temporal features play an important role in modeling human movements. In recent years, LSTMs have performed impressively in HAR and various other domains. The temporal features are extracted from time sensory signals by the LSTM architecture because of its long-term dependencies and temporal characteristics.

In our proposed architecture as explained earlier, convolutional AE is followed by a LSTM model. The results of the convolutional AE are passed as inputs to the LSTM to deduce the latent temporal interactions across the timeframes.

9.2.6 Fully connected Layer

A fully connected layer refers to a neural network in which each neuron applies a linear transformation to the input vector through a weights matrix. As a result, all possible connections layer-to-layer are present, meaning every input of the input vector influences every output of the output vector.

9.2.7 Softmax

Softmax is implemented through a neural network layer just before the output layer. The Softmax layer must have the same number of nodes as the output layer. Full Softmax is the Softmax we've been discussing; that is, Softmax calculates a probability for every possible class. Candidate sampling means that Softmax calculates a probability for all the positive labels but only for a random sample of negative labels. For example, if we are interested in determining whether an input image is a beagle or a bloodhound, we don't have to provide probabilities for every non-doggy example.

Full Softmax is fairly cheap when the number of classes is small but becomes prohibitively expensive when the number of classes climbs. Candidate sampling can improve efficiency in problems having a large number of classes.

9.3 Colab Notebook:

Colab notebook allows you to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more. When you create your own Colab notebooks, they are stored in your Google Drive account. You can easily share your Colab notebooks with co-workers or friends, allowing them to comment on your notebooks or even edit them.