IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

HUMAN ACTIVITY RECOGNITION USING MACHINE LEARNING

K.N.S.S.V.PRASAD [1] Assistant Professor, MD.R.NADEEM [2] Assistant Professor
DEPARTMENT of COMPUTER SCIENCE & ENGINEERING
ADIKAVI NANNAYYA UNIVERSITY
KAKINADA, ANDHRA PRADESH, 533005

Abstract:
Human activity detection is one of the leading applications of the machine learning algorithm today. It is used in the field of biomedical engineering, game development, developing better statistics for sports training, etc. The data is collected from the embedded accelerometer, gyroscope and other sensors of the XiaomiRedmi Note 7 Pro smartphone. In this project we will use data available in the WSIDM Lab. It includes data generated by smartphone accelerometers, gyroscopes, and other sensors to train supervised predictive models using Long Short-Term Memory (LSTM) machine learning techniques to generate a model. The experimental results show that the proposed approach can help improve the detection rate up to 95.87% and the transient detection rate over 80%, which is better than most existing similar models used to predict the type of motion that can be performed by the person, which is divided into the eight categories of standing, sitting, walking, running, sitting down, getting up and climbing stairs.
Keywords: Machine learning, LSTM, Activity Detection, Smartphone

INTRODUCTION:

Human Activity Recognition (HAR) is the recognition, interpretation and recognition of human behaviors that Smart Health Care can leverage to actively support users according to their needs. Human activity detection has broad application perspectives, such as monitoring in smart homes, sports, game control, and health care, caring for the elderly, detection and identification of bad habits. It can make our daily life smarter, safer and more convenient. Currently, human behavioral data can be collected in two ways: one based on computer vision and the other on sensors. Behavioral recognition based on computer vision has been studied for a long time and has a mature theoretical basis. However, the vision-based approaches have many limitations in practice. For example, the use of a camera is limited by various factors such as light, position, angle, potential obstacles, and privacy invasion issues, making it difficult to limit it in practical use. Although the research time of sensor-based behavioral recognition is relatively short, with the development and maturity of microelectronics and sensor technology, there are different types of sensors, such as accelerometers, gyroscopes, magnetometers, and barometers. These sensors can be integrated into mobile phones and wearable devices such as watches, bracelets and clothing. In addition, advanced wearable sensors have solved the problem of antimagnetic field interference, which can accurately estimate the current acceleration and angular velocity of motion sensors in real time in the presence of magnetic field interference. Therefore, these mobile sensors are usually small in size, high in sensitivity, and strong in anti-interference ability, so the sensor-based identification method is more suitable for practical situations. In addition, sensor-based behavioral recognition is not limited by scene or time, which can better reflect the nature of human activities. Therefore, the research and application of human behavioral recognition based on sensors is becoming more and more valuable and meaningful. In addition, the HAR includes two types: basic actions and transition actions. Due to the low frequency and short duration of transitional movements, there are relatively few studies on transitional movement from standing to sitting, walking to standing, etc. in human activity recognition research. However, the study of transitional motion is a very important part of human behavioral recognition to improve the behavior detection rate. The recognition of transitional measures should not be neglected. The transitional action is the distinction of a multitude of basic actions in frequent alternation. The accurate division of the transition action can accurately segment the streaming data to a certain extent, ultimately improving the detection rate. In addition, the behavior detection methods based on traditional patterns have shortcomings, such as manual feature extraction. With the application and advancement of machine learning in various fields, the machine learning model also shows great advantages in the field of behavioral recognition

The main contributions of this work are summarized as follows:

- We presented a deep learning model consisting of recurring layers of long short-term memory that can automatically learn local features and model the time dependence between features.
- We discussed the influence of key parameters in the machine learning model on performance and finally identified the best parameters in the model.
- We analyzed and compared the experimental results with other models using the same common dataset. The results show that the proposed method is superior to other advanced methods.

In this work, we use both an accelerometer and a gyroscope sensor from smartphones to collect data and proposed a hybrid LSTM model to detect the transient motion. Long Short-Term Memory (LSTM) is a type of deep neural network used as a feature extractor. It excels in local dependency and hence has good performance in extracting local features. However, information about human activities belongs to a long instance, which consists of complex movements and changes over time. The Long Short-Term Memory (LSTM) neural network is a type of recursion network that includes memory to simulate a time-dependent sequence problem. Therefore, the mix of LSTM can accurately identify the fundamental and transitional characteristics of activities.

LITERATURE SURVEY

Human Activity Recognition by Smartphone using Machine Learning Algorithm for Remote Monitoring:

AUTHORS: Sajanraj Thandassery, Beena M V

Human Activity Recognition has many applications such as patient monitoring, rehabilitation and disability support. When mobile sensors are attached to the subject's body, they allow continuous monitoring of numerous signal patterns from the phone. This has appealing uses in healthcare applications. In order to improve the state of global healthcare, numerous healthcare devices have been introduced, enabling doctors to perform remote monitoring and increasing users' motivation and awareness. Nowadays smartphones are becoming a part of our daily life. The best way to implement the idea is via smartphones. The smartphones contain various inbuilt sensor units like accelerometer, gyroscope, GPS, compass sensor and barometer. A system is thus developed to record the states of a user. Here, the mobile sensor is used as an input device and estimates human movement activity using data mining and machine learning techniques. Here we use the ANN classification algorithm in the activity detection system, which supports training and classification using only accelerometer data. We can predict the performance of these classifiers based on a set of observations of human activities such as walking, running, mounting, and dismounting in an activity detection system.

Trends in human activity recognition using smartphones

AUTHORS: Anna Ferrari, Daniela Micucci

The detection of human activity and the monitoring of population behavior are fundamental needs of our society. Population security, crowd monitoring, healthcare and life support, and lifestyle and behavior tracking are some of the major applications that require human activity detection. For the past few decades, researchers have been investigating techniques that can automatically detect human activity. This line of research is commonly known as Human Activity Recognition (HAR). HAR encompasses many tasks: from signal acquisition to activity classification. The tasks involved are not easy and often require dedicated hardware, sophisticated technology, and computational and statistical techniques for data pre-processing and analysis. Over the years, different techniques have been tested and different solutions proposed to achieve a classification process that gives reliable results. This overview presents the latest solutions proposed for each task in the human activity classification process, i.e. H. Acquisition, pre-processing, data segmentation, feature extraction and classification. Solutions are analyzed by highlighting their strengths and weaknesses. For the sake of completeness, the survey also presents the metrics commonly used to assess the goodness of a classifier, as well as the datasets of inertial signals from smartphones, which are mainly used in the assessment phase.

Feature Engineering for Human Activity Recognition
AUTHORS: Basma A. Atalaa, Ibrahim Ziedan, Ahmed Alenany
Human Activity Recognition (HAR) techniques can significantly improve health and livelihood systems for older people. These techniques, which generally work with data collected by wearable sensors or sensors embedded in most smartphones, have therefore recently attracted increasing interest. In this paper, a random forest-based classifier for human activity detection is proposed. The classifier is trained using a set of time domain features extracted from raw sensor data after it has been segmented into windows of 5 second duration. A detailed examination of model parameter selection is presented using the statistical t-test. Several simulation experiments are performed with the WHARF(Wearable Human Activity Recognition Folder) accelerometer benchmark dataset to compare the performance of the proposed classifier with support vector machines (SVM) and artificial neural networks (ANN). The proposed model shows high detection rates for different activities in the WHARF dataset compared to other classifiers using the same set of features. In addition, it achieves an average overall precision of 86.1%, surpassing the 79.1% detection rate reported in the literature using Convolution Neural Networks (CNN) for the WHARF dataset. From a practical point of view, the proposed model is simple and efficient. Therefore, it is expected to be suitable for implementation in portable devices such as smartphones with their limited memory and computing resources.

OTHER TECHNOLOGIES:

Numerous other technologies are used in this project. The Python programming language is used for coding and Jupyter Notebook is used for writing the code and numerous other technologies that we will talk about.

Python:

Python is one of those languages that are simple and powerful at the same time. One may be surprised at how easy it is to focus on solving the problem and not on the syntax and structure of the language in which to program. Guido van Rossum, the creator of the Python language, named the language after the BBC show Monty Python's Flying Circus. He didn't particularly like snakes, which kill animals for food by wrapping their long bodies around them and ruthlessly squeezing them. Python is extremely easy to use. Python has exceptionally simple syntax. Another specialty of Python is that you can freely distribute copies of this software, then read its code, make changes to it, and use parts of it in new free programs. Whenever we write code, you don't have to worry about low-level details like managing the memory used by our program. Due to its open-source nature, Python has been ported to many platforms. All Python programs can work on any platform without requiring any modifications at all. Python is an interpreted language and not a compiled language. Python supports both procedural programming and object-oriented languages; the program is built around functions and procedures, which are nothing more than reusable parts of programs. In object-oriented languages, the program is built around objects that encapsulate data and functionality. Python has a very powerful but simplified way of doing OOP. It has an embeddable property; you can embed Python in your C/C++ programs to give the users of your programs scripting capabilities. The Python standard library is indeed huge. It can help in performing various actions that include regular expressions, documentation generation, web browser, databases, threading, unit testing, email, GUI, cryptography and other system dependent things.

Jupyter Notebook

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computing environment for creating Jupyter Notebook documents. The term notebook can colloquially refer to many different entities, mainly the Jupyter web application, the Jupyter Python web server, or the Jupyter document format, depending on the context. A Jupyter Notebook document is a JSON document that follows a versioned schema and contains an ordered list of input/output cells, which can contain code, text (using Markdown), math, charts, and rich media, and are typically associated with the Extension .ipynb ending. A Jupyter Notebook can be converted to a number of standard open output formats (HTML, presentation slides, LaTeX, PDF, restructured text, Markdown, Python) by using the web interface via the nbconvert library or the nbconvert - Command line downloaded from Jupyter interface in a shell. To simplify the visualization of Jupyter notebook documents on the web, the nbconvert library is provided as a service via NbViewer, which takes a URL to any publicly available notebook document, converts it to HTML on the fly, and displays it can the user.

TensorFlow

Tensor, one of the best Python libraries for data science for a job, is Google Brains second generation system. Written primarily in C++, it includes the Python bindings, performance is not a concern. One of the favourite features is the flexible architecture that allows deploying it on one or more CPUs or GPUs in a desktop, server or mobile device using the same API. Not many libraries, if any, can claim that. It was developed for the Google Brain project and is now used extensively. However, you need to spend some time to learn the API, but it is worth the time spent. After just the first few minutes of playing around with the core functionality, TensorFlow would allow you to spend more time implementing the network designs instead of navigating through the API.

TensorFlow includes many deep learning frame works.

Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on top of JetBrainsIntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux-based operating systems or as a subscription-based service in 2020. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for developing native Android applications. Android Studio was announced on May 16, 2013 at the Google I/O Conference. It was in early access preview from version 0.1 in May 2013 and then entered beta from version 0.8 released in June 2014. The first stable build was released in December 2014, starting with version 1.0 Android Studio supports all the same IntelliJ (and CLion) programming languages e.g. Java, C++ and more with extensions like Go and Android Studio 3.0 or higher supports Kotlin and all Java 7 language features, plus a subset of Java 8 language features that vary by platform version. External projects are back porting some Java 9 features. While IntelliJ states that Android Studio supports all released Java versions and Java 12, it is not clear at what level Android Studio supports Java versions up to Java 12 (documentation mentions partial support for Java 8). At least some new language features up to Java 12 can be used in Android.

Machine learning:

Machine learning is an area of artificial intelligence that uses statistical techniques to give computer systems the ability to learn from data (such as incrementally improving performance on a specific task) without being explicitly programmed. The name machine learning was coined by Arthur Samuel in 1959. Machine learning examines the study and construction of algorithms that can learn and make predictions from data, where such algorithms overcome adherence to strictly static program instructions by making data-driven predictions or decisions by building a model from sample inputs. Machine learning is used in a range of computational tasks where designing and programming explicit algorithms with good performance is difficult or impossible; Examples of applications include email filtering, network intrusion detection, and computer vision. Machine learning is closely related to (and often overlaps with) computer statistics, which also focuses on predictions through the use of computers. It has strong links to mathematical optimization, which provides methods, theory, and application domains for the field. Machine learning is sometimes confused with data mining, the latter subfield focusing more on exploratory data analysis and known as unsupervised learning. In the field of data analysis, machine learning is a method for developing complex models and algorithms suitable for prediction; in commercial use this is referred to as predictive analytics. These analytical models enable researchers, data scientists, engineers, and analysts to make reliable, repeatable decisions and results, and uncover hidden insights by learning from historical relationships and trends in the data

Existing System:

Several investigations looked at the use of widespread mobile devices. It collects data from just two users wearing a single accelerometer-based device, and then transmits that data to the phone worn by the user. Another person used accelerometer data from a small group of users along with audio and barometric sensor data to detect six daily activities. However, the data was generated using accelerometer-based devices worn by the user and then sent to the phone for storage. Some studies took advantage of the sensors built into the phones themselves. Yang developed an activity detection system using a smartphone to distinguish between different activities. However, stair climbing was not considered and their system was trained and tested with data from only four users.

Brezmes.et.Al. developed a real-time system for detecting six user activities. In their system, an activity detection model is trained for each user. There is no universal model that can be applied to new users who lack training data.

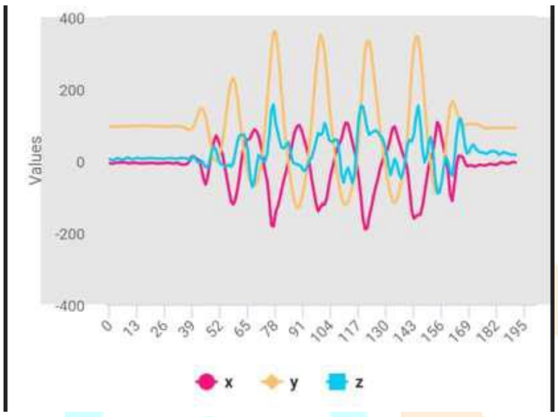
- System that uses sensors such as Tri-Axi accelerometers and gyroscopes from smartphones to estimate human activity.
- System detection related to WiFi and walls.
- System that detects human activity based on feature points

EXISTING SYSTEM ISSUES

- There is a need to attach activity detection sensors to humans.
- Limited accuracy and availability of sensors on HAR Sensors must be fitted.
- The cost of sensors is high

Proposed System

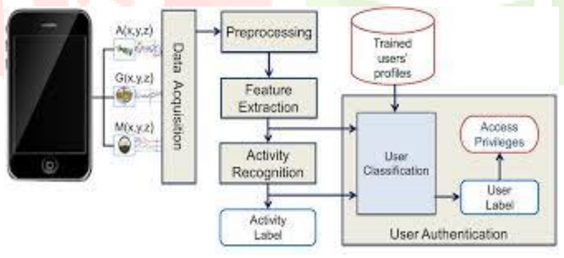
Human activity detection without the use of additional sensors and relying only on the use of smartphone sensors. Human activity data adopts a fixed time interval for activity determination and is fed to the model for detection and output, showing the activity performed. The data is created in a chart which shows how the sensor data works



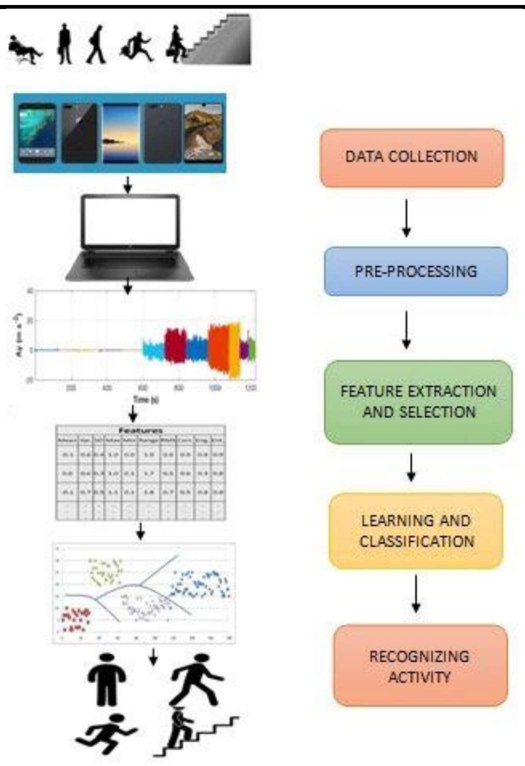
PROPOSED SYSTEM ADVANTAGES

No setup is required before implementation. Only one smartphone is used. The dependency on sensors is eliminated. Compared to others, fewer resources are required. Simple current model with no additional hardware requirements

Architecture Diagram:

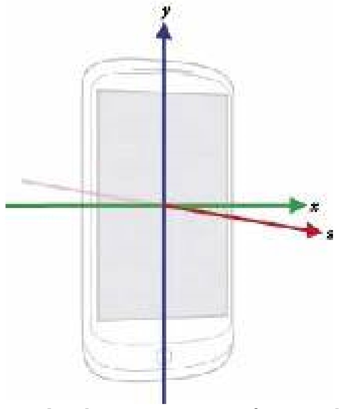


Data Flow Diagram:



IMPLEMENTAYION

The Human Activity Recognition using Smartphone dataset was created and made publicly available by Jennifier R. Kwapisz, Gary M, Weiss, and Samuel and can be downloaded for free from the WISDM lab. The raw data is now available. The smartphone sensor data was extracted from 1,098,207 samples. The set of physical activities focused by the authors are walking, sitting, standing, jogging, going upstairs & coming downstairs. The authors attached a XiaomiRedmi Note 7 prototype to each subject to collect sensor data. Each subject was instructed to perform each of the six activities twice. In the first trial, the smartphone was firmly attached to the left side of the subject's waist, but in the second trial, the subject was given the option to place the smartphone as they preferred. This assures that the data will vary depending on the phone's position for the same activity. Signals generated by the accelerometer and gyroscope embedded in the XiaomiRedmi Note 7 Pro are captured via a smartphone app. An accelerometer, as the name suggests, is used to measure the acceleration of the device. Values along the X, Y, and Z axes are used to capture motion such as swing, pitch, and so on. Below Figure shows the alignment of the axis of a three-axis accelerometer with respect to the device. Values provided via the three axes also include the acceleration due to gravity (g=9.81m/s2). When the mobile device is idle, it would only display gravity over one of the axes based on the orientation. A gyroscope, on the other hand, uses angular velocity to calculate rotation or twist in a smartphone device. Rotational speed is measured in rad/s along the three accelerometers that detect directional movement; a gyroscope detects the lateral orientation of the device. Both sensors are used to measure rate of change, but for different things.



Tasks associated with axis orientation of a smartphone device

Classification:

Number of examples: 1,098,207
Number of attributes: 6
Missing attribute values: None

Class distribution:

Walking -> 424,400 -> 38.6%,
Jogging -> 342,177 -> 31.2%,
Upstairs -> 122,869 -> 11.2%,
Downstairs -> 100,427 -> 9.1%
Sitting -> 59,939 -> 5.5%,
Standing -> 48,395 -> 4.4%

Raw.txt follows this format:
[User], [activity],[timestamp],[x-acceleration],[y-accel],[z-accel]

This line is a representative example:
33, Jogging, 49105962326000, -0.6946377, 12.680544, 0.50395286

Sampling rate:
20Hz (1 sample every 50ms)

Fields:
User: nominal, 1.36
Activity: nominal, {Walking, Jogging, Sitting, Standing, Upstairs, Downstairs}

Timestamp:
Numeric, generally the phone's uptime in nanoseconds (In future datasets this will be milliseconds since unix epoch.)

X-acceleration:

- Numeric floating point values between -20 and 20
- The acceleration in the x-direction as measured by the Android phone's accelerometer.
- A value of 10 = 1 g = 9.81 m/s^2 and 0 = no acceleration.
- The acceleration recorded includes gravitational acceleration towards the center of the earth, so when the phone is resting on a flat surface, the vertical axis reads +-10.

Y-acceleration:

- Numeric floating point values between -20 and 20
- The acceleration in the x-direction as measured by the Android phone's accelerometer.
- A value of 10 = 1 g = 9.81 m/s^2 and 0 = no acceleration.
- The acceleration recorded includes gravitational acceleration towards the center of the earth, so when the phone is resting on a flat surface, the vertical axis reads +-10.

Z-acceleration:

- Numeric, numeric, floating point values between -20 and 20
- The acceleration in the x-direction as measured by the Android phone's accelerometer.
- A value of 10 = 1g = 9.81 m/s^2 and 0 = no acceleration.
- The acceleration recorded includes the gravitational acceleration towards the center of the earth, so when the phone is resting on a flat surface, the vertical axis is + shows -10.

Working:

Human activity detection is one of the best features of human activity. It works on the Android application by using sensors e.g. Accelerometer, gyroscope. The sensors detect x-axis, y-axis, z-axis. In activity, man moves from one place to another. The input just took 200 entities and then removed previous values and took new values. It cannot be saved to the database. It only detects real-time activity and then visualizes the plotting diagram activity detection using sensors. The plot on the chart will display the legend (x-axis, y-axis, z-axis). The HAR application announces, in voice, the highest human activity value out of the different activities.

Feature Selection:

Feature selection is an important concept in machine learning applied as part of the pipeline. It is the concept of automatically or manually selecting a set of features that will help to improve the model and predictive output. This step is taken as it immensely affects the performance of the model in terms of build time as well as accuracy. Irrelevant features in the dataset can negatively affect the training, as this causes the model to train on data that does not contribute to the achievement of the prediction result. The impact is often seen on accuracy since the irrelevant data just acts as noise. Feature selection provides benefits by reducing over fitting, improving accuracy, and reducing training time. By applying feature selection, we reduce the dimensions of the dataset. This is often misunderstood as dimensionality reduction, which is not the case. Dimensionality reduction techniques often combine features together to reduce dimensions, while feature selection techniques eliminate attributes without affecting the rest. Two feature selection techniques are discussed in the following sections.

Long Short Term Memory (LSTM):
Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) **architecture [1]** used in the field of machine learning. Unlike standard feed-forward neural networks, LSTM has feedback connections. It can not only process individual data points (e.g. images), but also entire data sequences (e.g. voice or video). For example, LSTM is applicable to tasks such as un segmented, connected handwriting recognit**ion,[2]speech** recognition and anomaly detection in network traffic or IDSs (Intrusion Detection Systems). A typical LSTM unit consists of a cell, an input gate, an output gate and a forgetting gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information in and out of the cell. LSTM networks are well suited for classifying, processing, and making predictions based on time-series data because there can be delays of unknown duration between important events in a time-series. LSTMs were designed to overcome the vanishing gradient problem that can occur when training traditional RNNs. The relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models, and other sequence learning methods in numerous applications.

LSTM with a forget gate:
it will affect the performance of lstm greatly. adding a positive bias to the forget gate greatly improves the performance of the LSTM. Which means we should add a forget bias for lstm forget gate. The second most significant gate turns out to be the input gate.

Variables

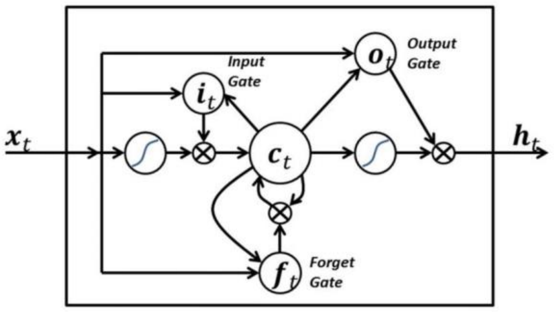
- input vector to the LSTM unit
- forget gate's activation vector
- input/update gate's activation vector
- output gate's activation vector
- hidden state vector also known as output vector of the LSTM unit
- cell input activation vector
- cell state vector
- weight matrices and bias vector parameters which need to be learned during training where the superscripts and refer to the number of input features and number of hidden units, respectively.

Activation functions:

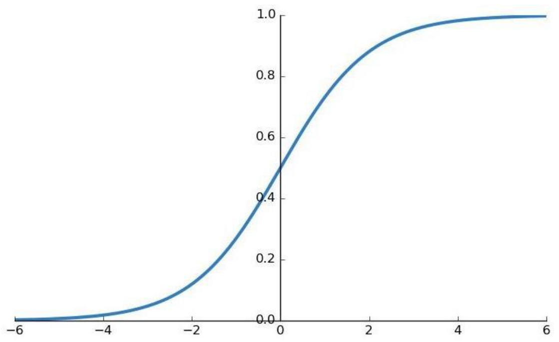
- Sigmoid function.
- Hyperbolic tangent function.
- Hyperbolic tangent function or as peephole LSTM paper

Peephole LSTM

A peephole LSTM unit with input output and forget gates. The figure is a graphical representation of an LSTM unit with peephole connections (i.e. a peephole LSTM). Peephole connections allow the gates to access the constant error carousel (CEC) whose activation is the cell state. Each of the gates can be thought as a "standard" neuron in a feed-forward (or multi-layer) neural network: that is, they compute an activation (using an activation function) of a weighted sum and represent the activations of respectively the input, output and forget gates, at time step .The 3 exit arrows from the memory cell to the 3 gates and represent the *peephole* connections. These peephole connections actually denote the contributions of the activation of the memory cell at time step, i.e. the contribution of (and not , as the picture may suggest). In other words, the gates and calculate their activations at time step (i.e., respectively, and) also considering the activation of the memory cell at time step i.e. The single left-to-right arrow exiting the memory cell is *not* a peephole connection and denotes The little circles containing a symbol represent an element-wise multiplication between its inputs. The big circles containing an *S*-like curve represent the application of a differentiable function (like the sigmoid function) to a weighted sum.

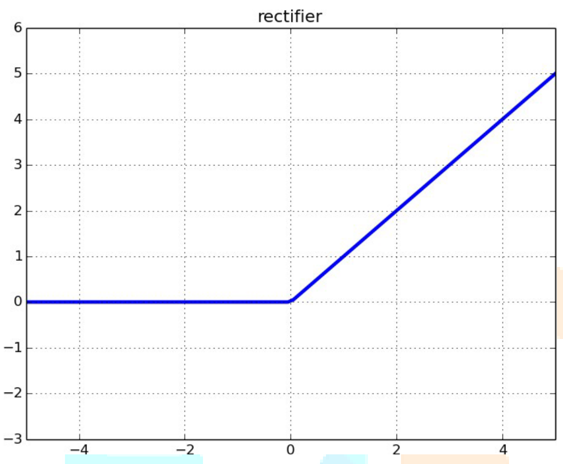


Sigmoid:
The sigmoid function has a curved characteristic and is commonly used in binary classification. The function generates a probability output between 0 and 1 for a probability output.



ReLU:
Rectified linear units are most commonly used in the hidden layers of artificial neural networks. The function is such that if the input is less than zero the output is 0 and if the input is greater than zero it returns the input itself as the output.

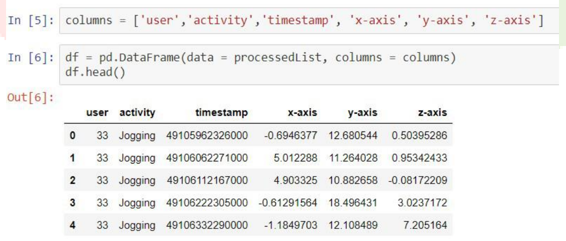
$$ReLU\ (x) = \max\ (0, x)$$



Softmax:
The Softmax activation function is used for multi-class classification. It calculates the probability distribution of each class over possible target classes. Based on the calculated probabilities, it determines the output for a given set of inputs.

Results:

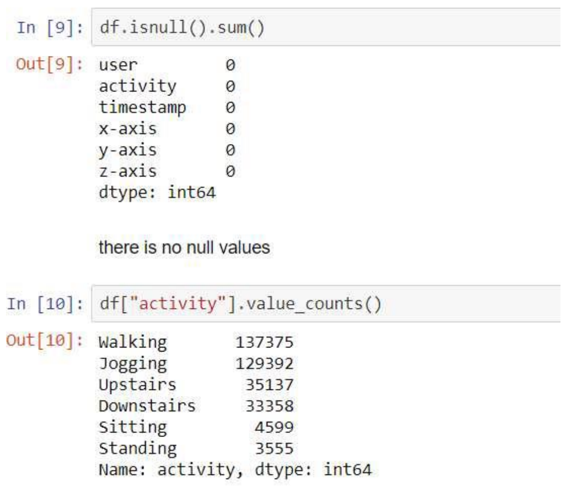
1. **Creating a Data Frame**



```
In [5]: columns = ['user','activity','timestamp', 'x-axis', 'y-axis', 'z-axis']

In [6]: df = pd.DataFrame(data = processedList, columns = columns)
        df.head()

Out[6]:
```

	user	activity	timestamp	x-axis	y-axis	z-axis
0	33	Jogging	49105962326000	-0.6946377	12.680544	0.50395286
1	33	Jogging	49106062271000	5.012288	11.264028	0.95342433
2	33	Jogging	49106112167000	4.903325	10.882658	-0.08172209
3	33	Jogging	49106222305000	-0.61291564	18.496431	3.0237172
4	33	Jogging	49106332290000	-1.1849703	12.108489	7.205164

2. **Check the null values and total Activity count**



```
In [9]: df.isnull().sum()

Out[9]: user         0
        activity     0
        timestamp    0
        x-axis       0
        y-axis       0
        z-axis       0
        dtype: int64
```

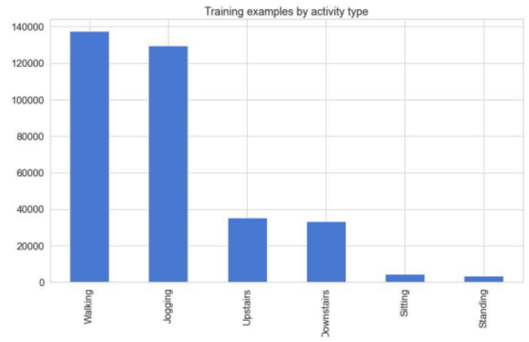
there is no null values

```
In [10]: df["activity"].value_counts()

Out[10]: Walking       137375
         Jogging       129392
         Upstairs       35137
         Downstairs     33358
         Sitting         4599
         Standing        3555
         Name: activity, dtype: int64
```

3. **Exploration: Training examples by activity type 4.4**

The columns we will be most interested in are activity, x-axis, y-axis and z-axis. Let's dive into the data:

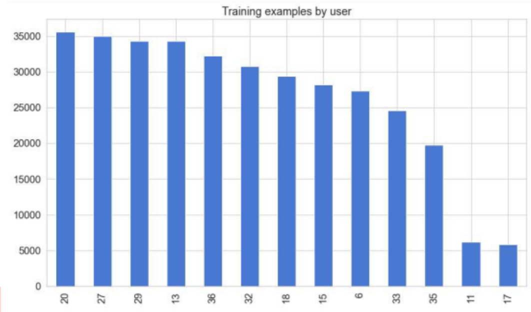
```
df['activity'].value_counts().plot(kind='bar', title='Training examples by activity type');
```



4. **Exploration: Training examples by user**

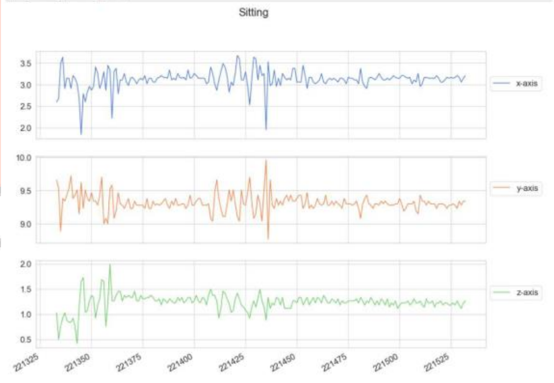
The columns we will be most interested in are activity, x-axis, y-axis and z-axis. Let's dive into the data:

```
df['user'].value_counts().plot(kind='bar', title='Training examples by user');
```



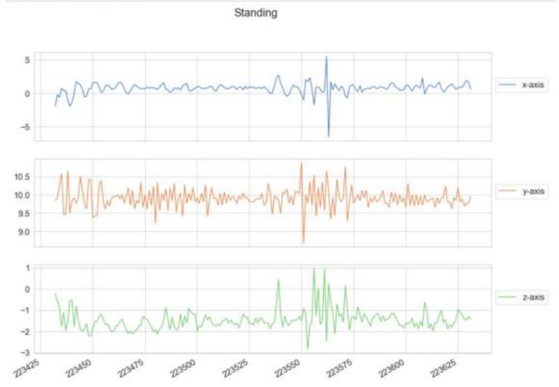
5. **Activity in Sitting:**

```
plot_activity("Sitting", df)
```

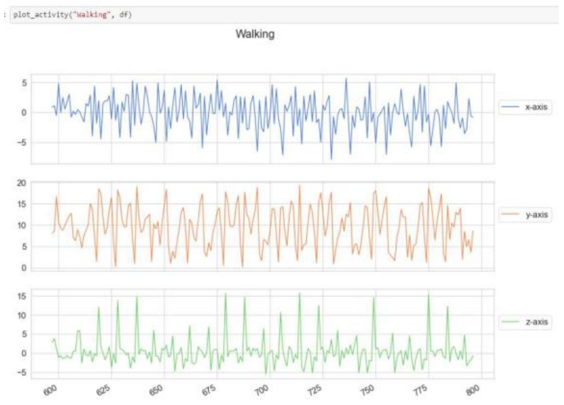


6. **Activity in Standing:**

```
plot_activity("Standing", df)
```

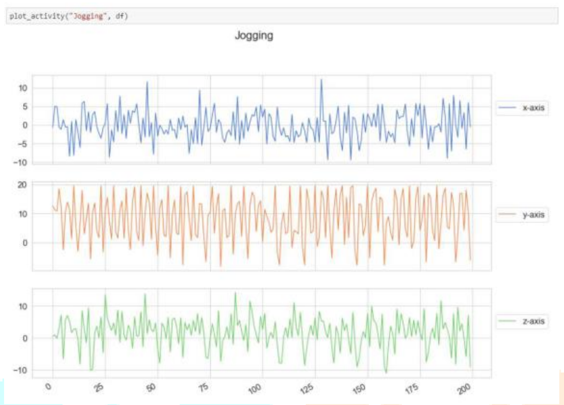


7. Activity in Walking:



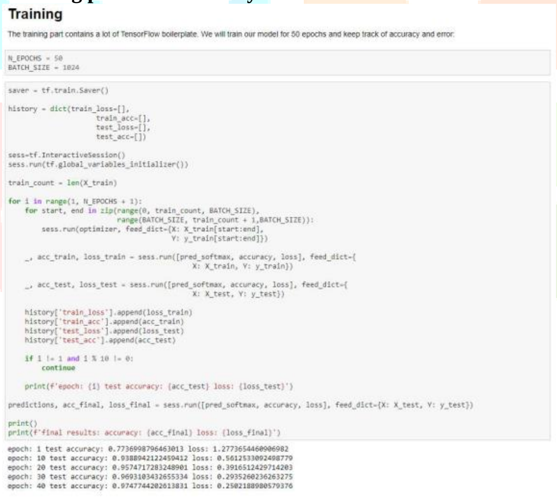
```
plot_activity("Walking", df)
```

8. Activity in Jogging:



```
plot_activity("Jogging", df)
```

9. Training period in accuracy value:



Training

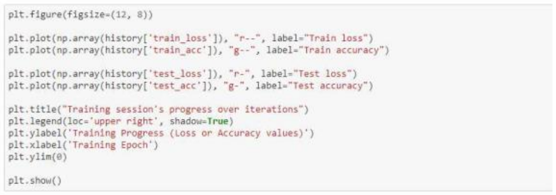
The training part contains a lot of TensorFlow boilerplate. We will train our model for 50 epochs and keep track of accuracy and error.

```
N_EPOCHS = 50
BATCH_SIZE = 1024
```

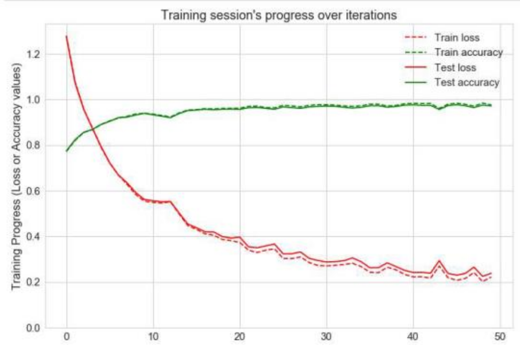
```
saver = tf.train.Saver()

history = dict(train_loss=[],
                     train_acc=[],
                     test_loss=[],
                     test_acc=[])

sess=tf.InteractiveSession()
sess.run(tf.global_variables_initializer())

train_count = len(X_train)

for i in range(1, N_EPOCHS + 1):
    for start, end in zip(range(0, train_count, BATCH_SIZE),
                          range(BATCH_SIZE, train_count + 1,BATCH_SIZE)):
        sess.run(optimizer, feed_dict={X: X_train[start:end],
                                       Y: y_train[start:end]})

    _, acc_train, loss_train = sess.run([pred_softmax, accuracy, loss], feed_dict={
                                            X: X_train, Y: y_train})

    _, acc_test, loss_test = sess.run([pred_softmax, accuracy, loss], feed_dict={
                                            X: X_test, Y: y_test})

    history['train_loss'].append(loss_train)
    history['train_acc'].append(acc_train)
    history['test_loss'].append(loss_test)
    history['test_acc'].append(acc_test)

    if i != 1 and i % 10 != 0:
        continue

    print(f'epoch: {i} test accuracy: {acc_test} loss: {loss_test}')

predictions, acc_final, loss_final = sess.run([pred_softmax, accuracy, loss], feed_dict={X: X_test, Y: y_test})

print()
print(f'final results: accuracy: {acc_final} loss: {loss_final}')
```

```
epoch: 1 test accuracy: 0.7736998796463013 loss: 1.2773654460906982
epoch: 10 test accuracy: 0.9388942122459412 loss: 0.5612533092498779
epoch: 20 test accuracy: 0.9574717283248901 loss: 0.3916512429714203
epoch: 30 test accuracy: 0.9693103432655334 loss: 0.2935260236263275
epoch: 40 test accuracy: 0.9747744202613831 loss: 0.2502188980579376
```

10. Evalution:

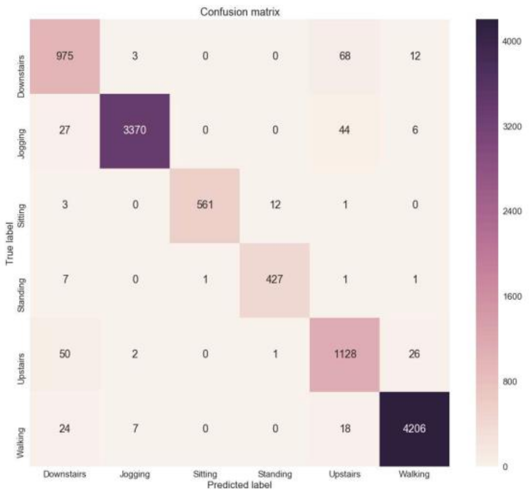


```
plt.figure(figsize=(12, 8))

plt.plot(np.array(history['train_loss']), "r--", label="Train loss")
plt.plot(np.array(history['train_acc']), "g--", label="Train accuracy")

plt.plot(np.array(history['test_loss']), "r-", label="Test loss")
plt.plot(np.array(history['test_acc']), "g-", label="Test accuracy")

plt.title("Training session's progress over iterations")
plt.legend(loc='upper right', shadow=True)
plt.ylabel('Training Progress (Loss or Accuracy values)')
plt.xlabel('Training Epoch')
plt.ylim(0)

plt.show()
```

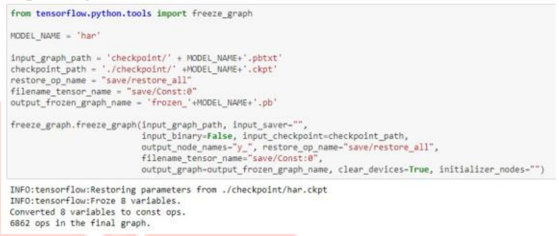


Our model seems to be learning well, with over 97% accuracy and about 0.2 loss. Let's take a look at the confusion matrix for the model's predictions

11. Confusion Matrix:



12. Exporting the Model:

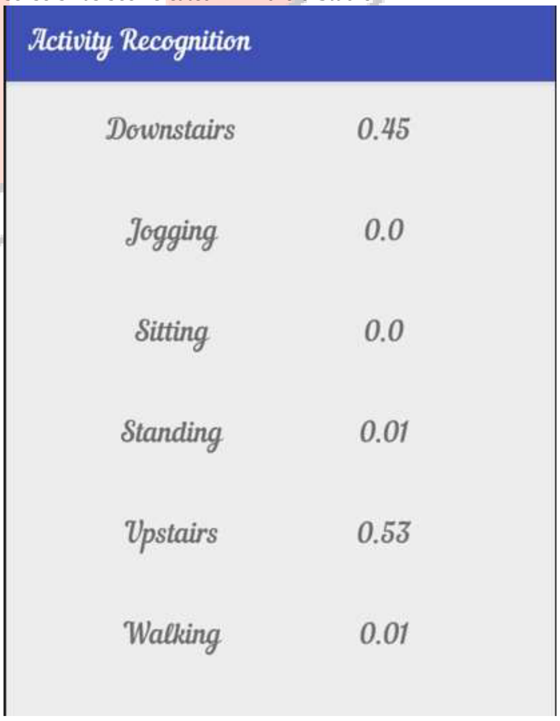


```
from tensorflow.python.tools import freeze_graph

MODEL_NAME = 'har'

input_graph_path = 'checkpoint/' + MODEL_NAME+'.pbtxt'
checkpoint_path = './checkpoint/' +MODEL_NAME+'.ckpt'
restore_op_name = "save/restore_all"
filename_tensor_name = "save/Const:0"
output_frozen_graph_name = 'frozen_'+MODEL_NAME+'.pb'

freeze_graph.freeze_graph(input_graph_path, input_saver="",
                          input_binary=False, input_checkpoint=checkpoint_path,
                          output_node_names="y_", restore_op_name="save/restore_all",
                          filename_tensor_name="save/Const:0",
                          output_graph=output_frozen_graph_name, clear_devices=True, initializer_nodes="")
```

```
INFO:tensorflow:Restoring parameters from ./checkpoint/har.ckpt
INFO:tensorflow:Froze 8 variables.
Converted 8 variables to const ops.
6862 ops in the final graph.
```

13. Human activity recognition by using sensor value

An Android application in human activity detection using sensors has been created in Android Studio

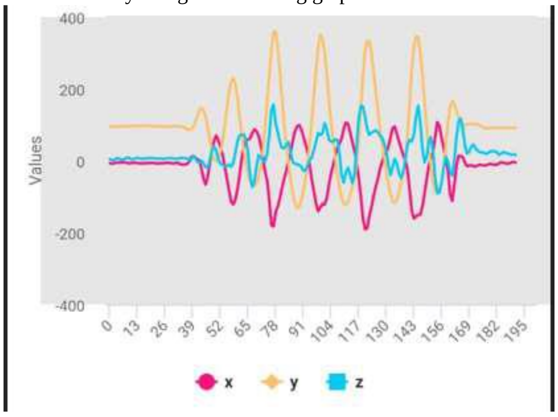


Activity Recognition

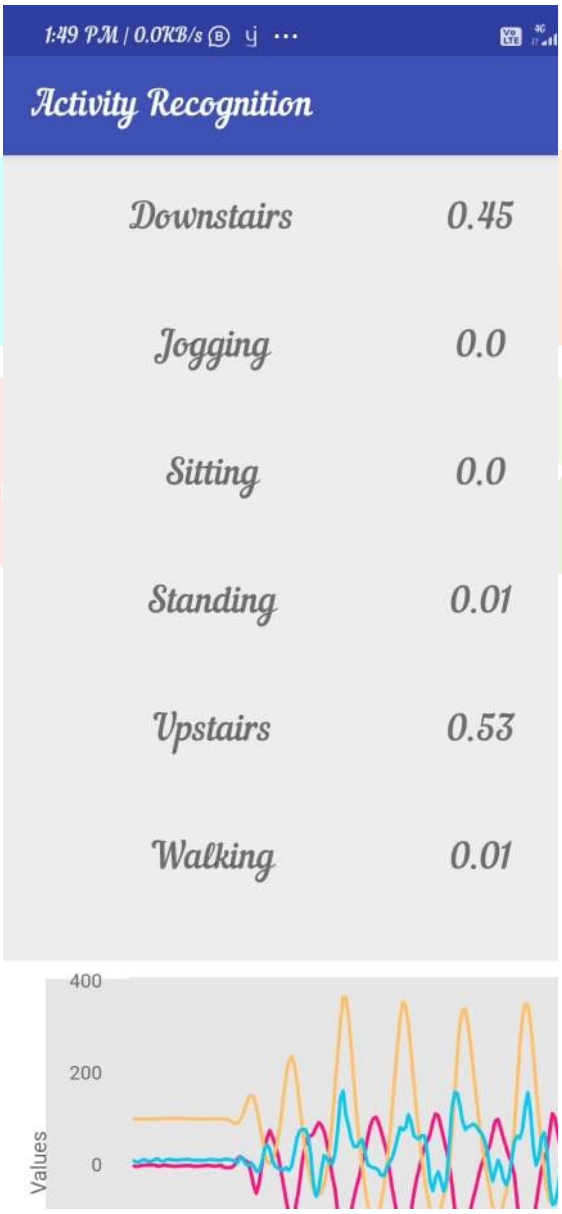
Downstairs	0.45
Jogging	0.0
Sitting	0.0
Standing	0.01
Upstairs	0.53
Walking	0.01

The above figure shows the prediction values of the Human Activities

14. Human activity recognition sensing graph



15. Human Activity recognition Final Output:



Activity Recognition

Downstairs	0.45
Jogging	0.0
Sitting	0.0
Standing	0.01
Upstairs	0.53
Walking	0.01

Conclusion:

The proposed work aims to develop a model to detect and classify human activities. For the human activity detection dataset, we used the Long Short-Term Memory (LSTM) model for human activity detection and classification. The power measurements are validated using the LSTM model designed with trained accuracy at 99.39% and after plotting between the accuracies we can conclude that our model is accurate. With the help of Loss as Binary Cross Entropy we put together our model. If we look at the above value after prediction, we can see that there is an array of value sensor data and if according to our approach we get any value in detecting human activity through the use of sensors. It gives the value in all activities and it gives the highest values in activity and human activity detection is body detection in sensors plotting graph visualization on x-axis and z-axis. However, the results prove that the designed network architecture has better progress and this application is widely used to achieve activity classification and detection.

References:

1. Adil Mehmood Khan, Young-Koo Lee, Sungyoung Y. Lee, and Tae-Seong Kim,A triaxial accelerometer-based physical -activity recognition via augmented-signal features and a hierarchical recognizer, Information Technology in Biomedicine, IEEE transactions on information technology in biomedicine, Volume 14, NO. 5, September2010
2. Charles Arthur,"Smartphone explosion in 2014 will see ownership in India pass US.", accessed on 30/3/2017,http://www.theguardian.com/technology/2014/jan/13/smartphone- explosion-2014-india-us-china-firefoxos-android
3. "Human activity recognition using smartphone dataset", accessedon 1/11/2016, https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+ Smartphones
4. Lester, Jonathan and Choudhury, Tanzeem and Borriello, Gaetano ,"A practical approach to recognizing physical activities", accessed on 4/10/2016, https://www.cs.cornell.edu/~tanzeem/pubs/JonathanLester_EDAS-1568973904.pdf
5. T.B. Moeslund, A.Hilton,V.Kruger, "A survey of advances in vision-based human motion capture and analysis, Computer Vision Image Understanding", Volume 104, Issues 2–3 ,December 2006.
6. L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," Pers Comput., Lecture Notes in computer Science, vol. 3001, pp. 1–17, 2004.
7. Kwapisz, Jennifer R and Weiss, Gary M and Moore, Samuel A, "Cell phone-based biometric identification", Biometrics: Theory Applications and Systems (BTAS), 2010 Fourth IEEE International Conference,10.1109/BTAS.2010.5634532, September2010
8. Kwapisz, Jennifer R and Weiss, Gary M and Moore, Samuel A, "Activity recognition using cell phone accelerometers", accessed on 21/11/2016, http://www.cis.fordham.edu/wisdm/includes/files/sensorKDD-2010.pdf
9. Ramiro J.Caro, "Central Limit Theorem Simulator" , accessed on15/11/2016, https://rpubs.com/RamiroJC/CLT_Slides
10. Central Limit Theorem ,accessed on 12/11/2016, http://www.investopedia.com/terms/c/central_limit_theorem.asp
11. " Principal component analysis" ,accessed on10/11/2017, https://en.wikipedia.org/wiki/Principal_component_analysis
12. " Practical Guide to Principal Component Analysis (PCA) in R & Python", accessed on5/11/2016,https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/
13. Leo Breiman , Adele Cutler, "Random forests", accessed on 10/11/2016, https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

14. Tavish Srivastava, "Introduction to Random forest – Simplified", accessed on 8/11/2016, https://www.analyticsvidhya.com/blog/2014/06/introduction-random-forest-simplified/
15. "k-nearest neighbors algorithm",accessed on 12/2/2017, https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
16. Ola Soder, "knn classifiers",accessed on 4/3/2017, http://www.fon.hum.uva.nl/praat/manual/kNN_classifiers_1What_is_a_kNN_classifier.html
17. David Meyer , Evgenia Dimitriadou , Kurt Hornik , Andreas Weingessel , Friedrich Leisch , Chih-Chung Chang ,Chih-Chen Lin ,"Department of Statistics ,Probability theory group,TU Wein",aceessed on2/3/2017,https://cran.r project.org/web/packages/e1071/index.html
18. Yu-Wei, David Chiu," Machine Learning with RCookbook",
19. Gary H.Y ," Notes for nnet, tree",accessed on2/2//2017, http://statisticsr.blogspot.in/2008/10/notes-for-nnet.html
20. Brian Ripley , William Venables,"Package nnet",accessed on 20/1/2017, https://cran.r-project.org/web/packages/nnet/nnet.pdf
21. Aarshay Jain," Fundamentals of Deep Learning – Starting with Artificial Neural Network", accessed on 15/1/2017, https://www.analyticsvidhya.com/blog/2016/03/introduction-deep-learning-fundamentals-neural-networks/
22. Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge L. Reyes- Ortiz, "A Public Domain Dataset for Human Activity Recognition Using Smartphones", European Symposium
23. Jorge L. Reyes-Ortiz, Alessandro Ghio, Davide Anguita , Xavier Parra, Joan Cabestany, Andreu Catal, "Human Activity and Motion Disorder Recognition: Towards Smarter Interactive Cognitive Environments", European Symposium on Artificial Neural Networks, 24-26 April2013.
24. TONI_K, "Accelerometer Basics", accessed on 20/9/2016, https://learn.sparkfun.com/tutorials/accelerometer-basics
25. Toni_K, "Gyroscope Basics" ,accessed on 20/9/2016,https://learn.sparkfun.com/tutorials/gyroscope
26. I.T. Jolliffe ,"Principal Component Analysis", Second Edition, springer, April2002" Principal component analysis" ,accessed on 10/11/2017, https://en.wikipedia.org/wiki/Principal_component_analysis
27. "Practical Guide to Principal Component Analysis (PCA) in R & Python", accessed on 5/11/2016,
28. https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/
29. Physical Human Activity Recognition Using Wearable Sensors Ferhat Attal 1 ,Samer Mohammed 1,*, Mariam Dedabrishvili 1 , Faicel Chamroukhi 2 , Latifa Oukhellou 3 and Yacine Amirat 1 Received: 11 September 2015; Accepted: 8 December 2015; Published: 11 December 2015 Academic Editor: Vittorio M.N.Passaro
30. Cleland, I.; Kikhia, B.; Nugent, C.; Boytsov, A.; Hallberg, J.; Synnes, K.;McClean, S.; Finlay, D. Optimal Placement of Accelerometers for the Detection of Everyday Activities. Sensors 2013, 13,9183–9200. Artificial Neural Networks, 24-26 April2013.