**A REPORT**

**ON**

**HUMAN ACTIVITY RECOGNITION APP**

**By**

Jaideep Reddy Gedi 1700215C203

## Prepared in the partial fulfillment of the

Practice School II Course

# AT

**Quintessential Innovations Pvt Ltd, Hyderabad.**

A Practice School II Station of



##### BML MUNJAL UNIVERSITY

**(July 2019)**

**Table of Contents**

**Contents Page No**

1. Introduction 6
2. Problem Statement 6
3. Company Profile 7
4. Methodology 8
   1. Data Analysis 8
   2. Feature Engineering 12
   3. Preparing Deep Learning Model 13
      1. Recurrent Neural Networks and LSTM 13
      2. LSTM Model(code) 18
      3. Training and Freezing Model 20
   4. Building App 21
      1. UI/UX 21
      2. Backend 22
5. Results and Discussion 23
6. Conclusions 24
7. References 25

##### C:\Users\jaideep reddy\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\5D5B3A44.tmp

##### Certificate of authenticity

**CERTIFICATE**

**This is to certify that Practice School Project of Jaideep Reddy Gedi**

**titled Human Activity Recognition App is an original work and that this work has not been submitted anywhere in any form. Indebtedness to other works/publications has been duly acknowledged at relevant places. The project work was carried during 20-5-2018 to 5-7-2019 in Quintessential Innovations Pvt Ltd**

|  |  |
| --- | --- |
|  |  |
| **Signature of PS-II faculty** | **Signature of industry mentor/Supervisor** |
| **Name: Dr Purnendu Pandey** | **Name: Daman Anil Patel** |
| **Designation: Assistant Professor,**  **SOET, BML Munjal University.** | **Designation: Head of Business Operations** |
| ***(Seal of the organization with Date)*** | ***(Seal of the organization with Date)*** |

**ACKNOWLEDGEMENT**

I hereby take this opportunity to thank all those people whose knowledge and experience helped me bring this report in its present form. All this journey and the work would’ve not been existed without them.

I am greatly indebted to **Mr. M.B.Srinivas**, **Dr. Purnendu Pandey**, **Mr. Daman Anil Patel** (Head of Business Operations), **Ameer Sohail** (Electronics Designer), **Atul Nandan**, **Anirudha Surabhi** (CEO), for coordinating this training and giving me an invaluable opportunity to work in a competitive yet amicable atmosphere and providing me with all the facilities required to carry out this project. I thank Ameer Sohail for being with me during this whole project and giving me valuable inputs and providing your out of box thinking thoughts. I thank Daman and Atul for making me get comfortable and familiarized inside and outside during the period of this internship . It was always a new and unique experience working with you.

I would like to express my gratitude towards my colleagues and seniors for their worthwhile suggestions, guidance and for all the knowledge they imparted to me during this period.

I would also like to express my deep appreciation to my family and friends who have been a constant source of inspiration. I am eternally grateful to them for always encouraging and being with me where ever and whenever I needed them.

**JAIDEEP REDDY GEDI**

**OBJECTIVES**

Human activity recognition is an important area of computer vision research and applications. The main objective of Human Activity recognition is to infer the actions of persons from a set of observations captured by sensors present inside smartphone of the user.

To secure accurate results one requires a well organised dataset, a perfect feature engineering technique as well as a suitable classification algorithm or a deep network model. Travel mode detection using sensor data is also a similar project, this project might be a prototype or a guide for building model for travel mode detection, this is the one of the objectives of doing this project explicitly.

**INTRODUCTION**

With the recent advancements in technology such as mobile phones, the smartphones are fed up with lots and lots of sensors, such as proximity, orientation, accelerometers, GPS, light sensors, image capturing sensors and so on with best accuracy rate and range. So, keeping these sensors in mind, android development has allowed developers to access the data from these sensors, this has opened windows for many recognition activities for fitness tracking such as step counters, heart beat rate finder and so on. This also has a role in safety monitoring too.

**PROBLEM STATEMENT**

To build a ML model to detect the human activities based on smartphone accelerometer data and deploy the model into an android app.

The dataset used in this project is provided by WISDM lab (wireless sensor data mining). The dataset consists of the accelerometer sensor data along all three-axis from smartphones of different subjects. The dataset consists of sensor data of six basic activities namely “sitting”, “standing”, “walking”, “jogging”,” upstairs”,” downstairs”. And an important point to note is that the data is collected by placing the smartphone in the subject’s front right pocket of their trouser.

Machine learning is used to detect the activities, and the deep network model used in this project is LSTM network which is best suitable for the time series data like sensor data or speech etc. The reason why LSTM models are best for time series data is explained in later sections and the model has returned a validation accuracy of 97%.

**COMPANY PROFILE**

**QUIN HELMETS**

**STORY OF QUIN**

Quintessential Design is the creator of Quin Design Helmets and Technology. Quin was established in 2017 by a diverse team whose backgrounds include award-winning industrial design, women's safety advocacy, patent law, and more.

**VISION AND MISSION**

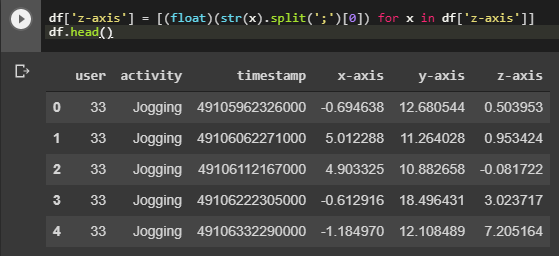
* Quin has a strong mission to change the way consumers think about normal safety standards in gravity and high adrenaline sports by creating technology that thinks about your safety while you think about your performance.
* Quintessential Design is an innovations lab with a strong mission to change the way consumers think about normal safety standards, and provide game changing safety solutions.
* Quintessential Design endeavors to create a better world through essential innovations in response to real problems.

**METHODOLOGY**

**DATA ANALYSIS**

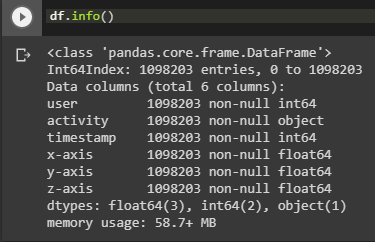
As said previously the dataset was provided by WISDM lab. So, let’s get familiarised with the dataset.

**The dataset looks like this:**

**Fig 1.1**

**Raw Dataset View**

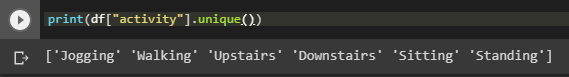
There are 6 columns: user number, activity, timestamp, x-axis data, y-axis data, z-axis data. As we need the model to be generic to all the users, we do not consider the user column in our model.

**Info on dataset:**

**Fig 1.2 Dataset info**

There are no null values and all the sensor data is of float type and activity is of string type(object). Total size of dataset is around 59 MB.

**Activities in dataset:**



**Fig 1.3 Activities in data set**

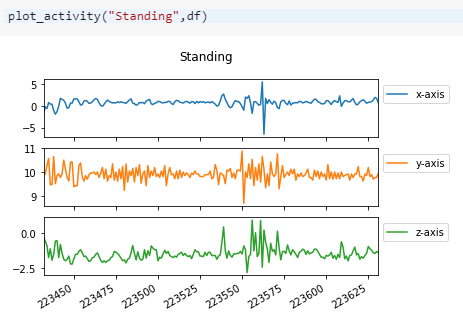
**Plots of samples of all 6 activities:**

**Sitting:**

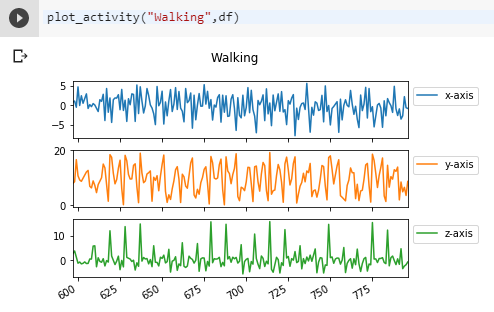
**Fig 1.4 Sitting**

The values of X and Z did not cross 3, and y is between 9 and 10 because of gravity. And these low values are reason for stationary position.

**Standing:**

** Fig 1.5 Standing**

**Walking:**

 **Fig 1.6 walking**

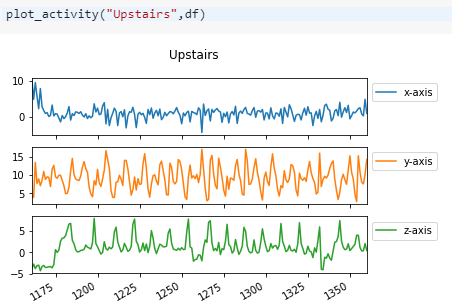
You can see the patterns along the axes and there are lots of fluctuations along all three axes and the y-axis is touching around 20.

**Jogging:**

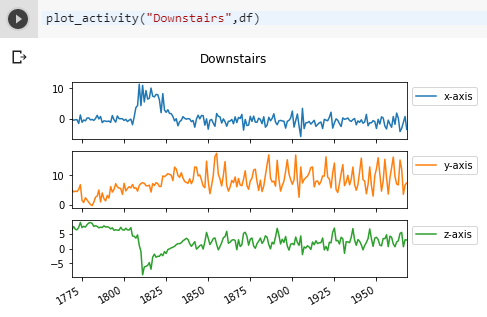
** Fig 1.7 Jogging**

Jogging and walking are looking a bit similar in patterns but the fluctuautions or peaks are more dense than in walking.

**Upstairs:**

** Fig 1.8 Upstairs**

**Downstairs:**

 **Fig 1.9 Downstairs**

In upstairs and downstairs figures, you can find the sudden and wide fluctuations or peaks mostly found along Z and Y axes.

**FEATURE ENGINEERING**

We have tried to make the data as raw as possible because, we have to recognize the activities in real time, so if the computation for feature extraction takes more time, the delay will become more. So, we have not extracted any features like mean, standard deviation, skewness, kurtosis etc. and no feature scaling was applied because we cannot estimate the maximum value of the sensor data, that’s why it is not applied.

We have applied a technique called dynamic window sliding technique, which takes a part of continuous data with a particular length and makes that part as one data point. And we need to set some overlap so that to make the windows continuous.

Example:

An external file that holds a picture, illustration, etc.
Object name is sensors-16-00716-g008.jpg

**Fig 2.1 sliding window technique**

Image source: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4883407/>

**PREPARING DEEP NETWORK MODEL**

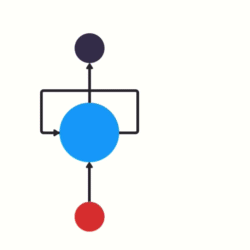
**RECURRENT NEURAL NETWORKS AND LSTM**

Recurrent neural networks, or RNNs for short, are a type of neural network that was designed to learn from sequence data, such as sequences of observations over time, or a sequence of words in a sentence.

RNN’s are good at predicting sequential data, how??

In sequential data, one cannot find or predict the output just by giving one timestep datapoint as input. So, these recurrent networks use the previous timestep information at the time of when an input of a particular timestep was given. We can compare this RNN’s with Markov chain rule. As it defines that the output of current state depends on the output of the previous state.

So, an RNN has a looping mechanism that acts as a highway to allow information to flow from one step to the next.

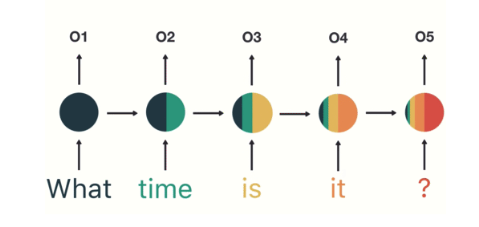


**Recurrent Neural Network**

So, the hidden state of past timesteps will be given as input along with the current timestep input data.

**Fig 3.1.1 Recurrent neural network**

**Illustration:**

Let’s think that the sequence is a sentence. **Fig 3.1.2 Illustration**

The model needs to know the intention of the statement. So, the output or intention can’t be decided just from a single word. So, recurrent networks used the hidden states of previous words in the sentence too.

**Problems with RNN’s**

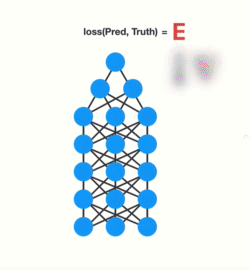
1. **Short term memory**
2. **Vanishing gradient**

**SHORT TERM MEMORY**

In the above picture of recurrent neural network, in the last hidden state, the information of the previous states and the current state are not equally distributed, it means that if the sequence is huge, then there might be the case of forgetting the hidden state information of some of the previous states will be lost. This problem is called short term memory.

**VANISHING GRADIENT**

Usually in a neural network model, after every feed forward propagation, the outputs are compared with real values and the loss function is calculated and the weights, biases of the network will be adjusted based on the gradients of the loss function. Higher the gradients, higher the adjustments.



**Fig 3.1.3 Back propagation**

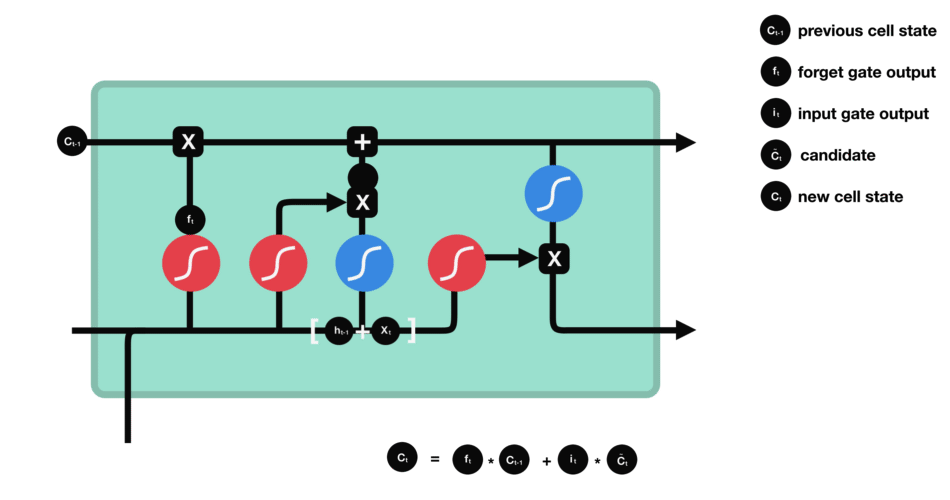
So, how does this cause problems in RNN’s. Usually, while back propagation, if the previous layers to the current layer have less gradients, then the adjustments to current layer will be small as the gradients along the all layers need to be multiplied. So, as the process goes on the gradient might get vanished. In, RNN’s we can assume that each time step as a layer, so as the timesteps increases the gradients gets smaller exponentially. So, when model tries to learn, as the gradient is small, the adjustments are small, so model can’t learn anything, this problem is called Vanishing gradient problem.

So, how to tackle this?

That’s why a new advanced RNN named **Long Short-Term Memory** network is introduced.

**LONG SHORT-TERM MEMORY NETWORKS:**

LSTM networks also have a similar control flow as a recurrent neural network. It processes data passing on information as it propagates forward. The differences are the operations within the LSTM’s cells.



**Fig 3.1.4 LSTM cell**

An LSTM cell consists of gates that control which information needs to be stored and which needs to be forgotten. The gates have the capability of recognizing the states which have high importance, and which do not. The main components of an LSTM cell are cell state and the gates. There are three types in an LSTM they are forget gate, input gate and output gate. These gates regulate the flow information across the time.

The **cell state** carries the important information through out the process of sequencing. The **forget gate** decides what information should be kept and what needs to be thrown away. It takes use of sigmoid function for doing its task. The input gate updates the cell state, it initially takes in the previous hidden state and current input and make them into a vector and applies sigmoid which defines what needs to be stayed and

what not and then it passes the both hidden and input to tanh function to get squish them to -1 and 1, and then multiply both get the data that needs to be updated into the cell state.

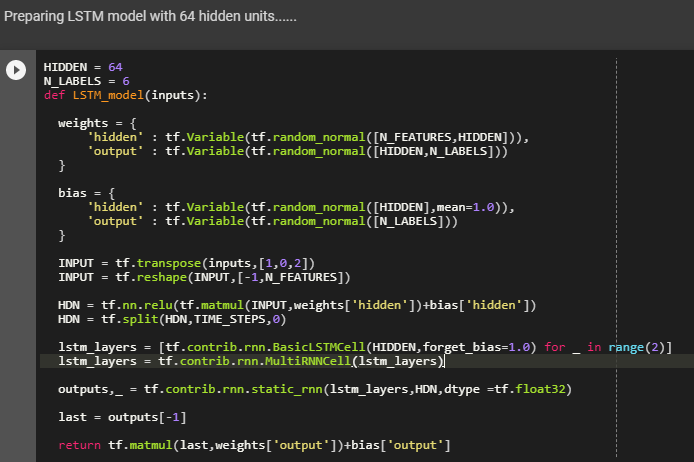
The output gate decides what the next hidden state should be. So, it sends the previous hidden state and current input through sigmoid and then it clubs this with the cell state to get the new hidden state for next time step.

In this way, LSTM eliminates some layers and keeps quintessential layers, there by reducing the problems of short-term memory and vanishing gradient.

The perfectness and clarity on this concept of RNN’s and LSTM’s is because of a blog on the internet, all the information above is written on my own, the pictures are taken from the blog, I want to credit the blog here:

# Illustrated Guide to Recurrent Neural Networks by Michael Nyugen.

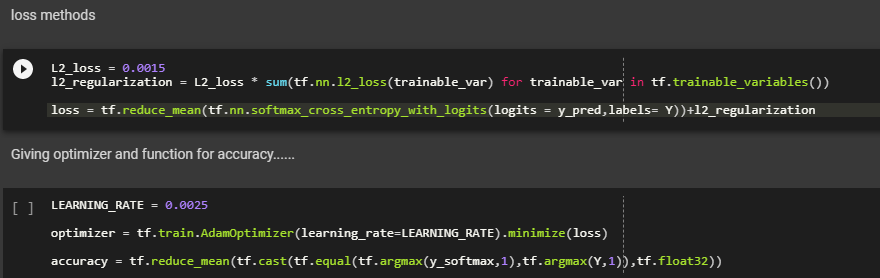
**LSTM MODEL CODE**



**Fig 3.2.1 LSTM model code**

Initially weights and biases for hidden and output were initialised randomly with some numbers. Hidden layer was constructed by multiplying weights and inputs and adding it to biases, in the same way out put layer is constructed by multiplying outputs of hidden layer stacked with LSTM cell to weights and adding them to biases declared initially. Now we prepare 2 LSTM cells with 64 hidden units and stack them and merge them with hidden layer. Then the values of the output are returned. Now we create a place-holders for inputs and labels promising that we will be giving input later on (i.e. variables) in real time. No of hidden units or nodes are 64 and the output layer consist of nodes equal to no of classes to classify of course.

Then we define loss function, optimizer, accuracy functions which needs applied on the outputs and make model better.

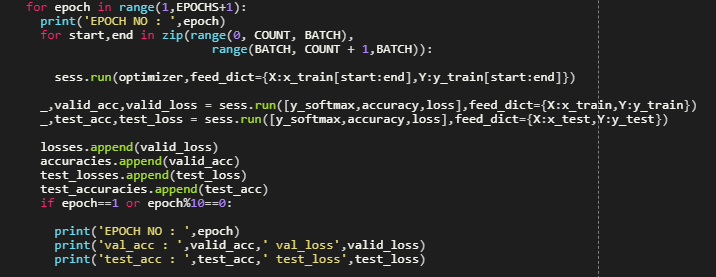


**Fig 3.2.2 Loss function, optimizer and accuracy functions**

The L2 regularisation is used to eradicate overfitting.

Learning needs to be a bit less, else the loss function becomes Nan sometimes because of high amount of data.

**TRAINING**

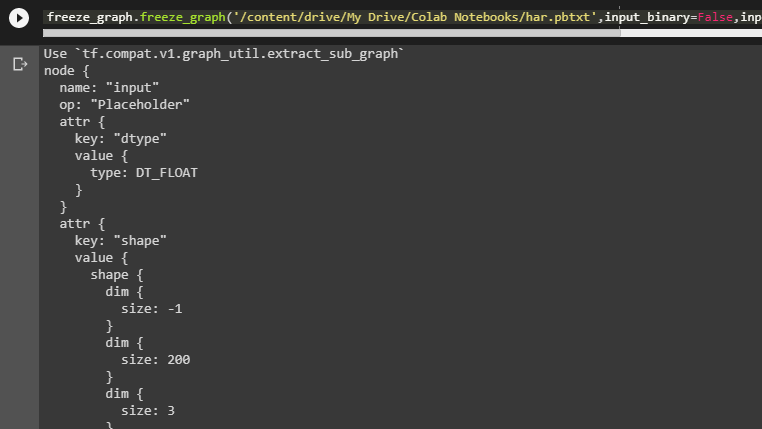


**Fig 3.2.3 Training code**

So, we have trained the data for 50 epochs, no of epochs no of times we train our model by changing the weights using backpropagation based on the validation accuracy and the loss we get.

**FREEZING MODEL**

What we usually do in freezing is we save the session which was trained on the data using train.saver into .pbtxt and .ckpt files and then we use these files to freeze the model using freeze library or package.

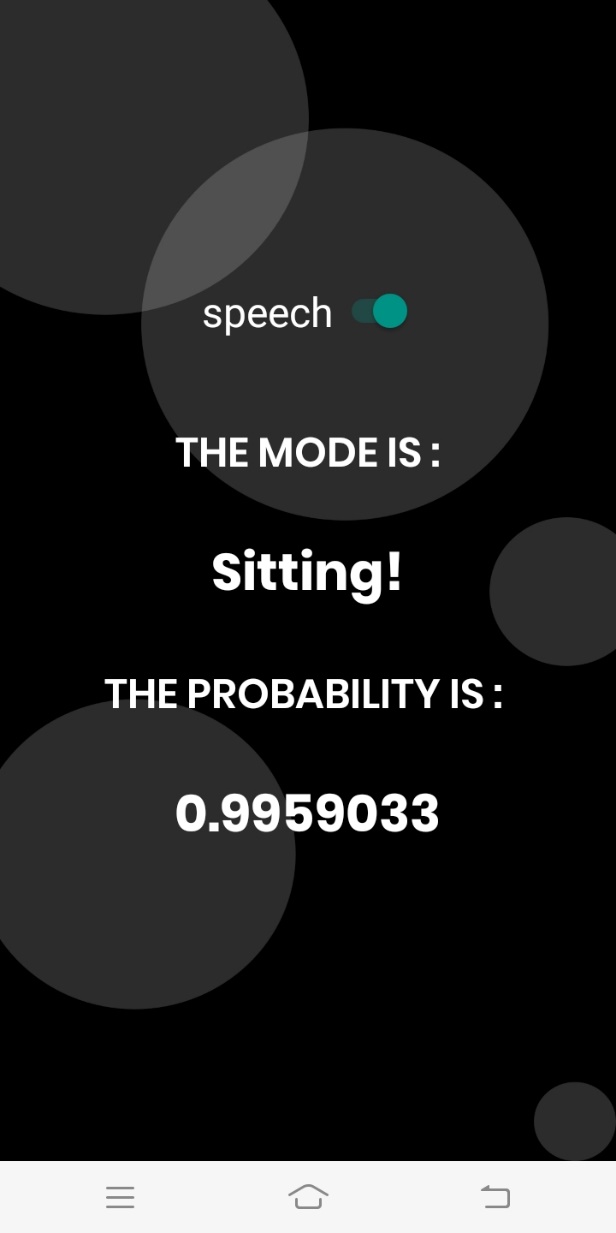


**Fig 3.3.1 Preview of info in .pb file**

The pretrained model or frozen graph is stored in .pb files i.e. protobuff format, which are similar to json files as shown in above figure. This .pb file will be used in our app to get the pretrained model of ours. And interface named tennsorflowInferenceInterface is used to retrieve the pretrained model and apply the model to our inputs and passes the outputs.

**BUILDING APP**

**UI/UX**



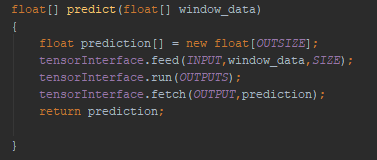
**Fig 4.1.1 App interface (preview)**

The app consists of a single layout with basic, text and switch views. It consists of a linear layout within it contains some text-views and a switch is also placed to turn on and off the speech mode.

**BACKEND**

We use TensorflowInferenceInterface class to use our pretrained model on the inputs that we provide in the real time. For this interface we need to add lib-tensorflow\_inference\_java library into the project of the app, this library contains required classes and packages of TensorFlow to perform our classification.

Initially we need to place our .pb file or frozen model file in assets folder of our app. And then we need to make a java class for our classifier and create a new inference class. To predict the probabilities of the activities the function goes like this:



**Fig 4.2.1 Classifier predict function**

We fed the inputs and run the classifier, and then fetch the outputs and then returning the predicted values.

In mainActivity.java file, we use SensorManager and Sensor classes to retrieve the values of the accelerometer sensor data in real time, we take a set of 200 datapoints which are continuous and then sent to the predict function of tensorClassifier. And then pick the label or class with highest probability and then set the text view to the name of that activity.

For text to speech we use textToSpeech class variable and initialize it and then when ever the classifier performs classification and text-view changes then the texttospeech variable is fed up with class name, so that it speaks that class name. we make use of switch.isChecked() function to check whether speech is turned on or not and then we call tts.speak() function to speak out the activity name.

**RESULTS AND DISCUSSION**

The model gave an accuracy of around 97% after 50 epochs









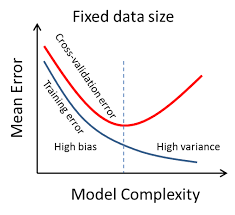




**Fig 5.1 Results for every 10 epochs**

We can see that after every 10 epochs we can see that there is recognizable decrease in loss and increase in accuracy of validation and train.

You might have asked that if you say that as epochs increases accuracy increase then why don’t you increase the number of epochs. It is because of over fitting, it means after a no of epochs the model try to remember the patterns instead to generalise the outputs, this might increase the train accuracy, but you might see a significant decrease in the validation accuracy because the model is trying to remember things instead to learn.



**Fig 5.2 Overfitting example**

Example for overfitting is shown in figure.

There are some methods or techniques to reduce overfitting they are:

Validation, Dropout, L2 regularization and also by having simple model and high amount of data.

We have used Validation as well as L2 regularisation and also kept our model simple to avoid overfitting in our model.

**CONCLUSIONS**

Though it showed a 97% accuracy, it still showed some error predictions when you try to perform classification in the app at the real time, I think the reason for this is due to difference in accelerometer specs and some reason might be faulty entries in the dataset. I feel that performance in app might be improved by changing the dataset.

**APPENDIX 1**

**LIST OF FIGURES**

Fig 1.1 Raw Dataset view 8

Fig 1.2 Dataset info 8

Fig 1.3 Activities in dataset 9

Fig 1.4 Sitting plot 9

Fig 1.5 Standing Plot 9

Fig 1.6 Walking plot 10

Fig 1.7 Jogging Plot 10

Fig 1.8 Upstairs plot 11

Fig 1.9 Downstairs Plot 11

Fig 2.1 Sliding window technique 12

Fig 3.1.1 Recurrent Neural Network 13

Fig 3.1.2 RNN illustration 14

Fig 3.1.3 Back propagation 15

Fig 3.1.4 LSTM cell structure 16

Fig 3.2.1 LSTM model code 18

Fig 3.2.2 Loss function and optimizer and accuracy functions 19

Fig 3.2.3 Training model 19

Fig 3.3.1 .pb file preview 20

Fig 4.1.1 App interface (preview) 21

Fig 4.2.1 Classifier Function 22

Fig 5.1 Results per every 10 epochs 23

Fig 5.2 Overfitting example 24

**APPENDIX 2**

**REFERENCES**

1. Dang-Nhac Lu, Duc-Nhan Nguyen, Thi-Hau Nguyen, Ha-Nam Nguyen .[Vehicle Mode and Driving Activity Detection Based on Analyzing Sensor Data of Smartphones](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5948751/).
2. Muhammad Awais Shafique, Eiji Hato.

[Travel Mode Detection with Varying Smartphone Data Collection Frequencies](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4883407/)

1. <https://www.edyoda.com/course/1429>

Jaley Dholakiya. Step by step guide to TensorFlow.

1. Illustrated Guide to Recurrent Neural Networks by Micheal Nyugen.
2. Illustrated Guide to LSTM Neural Networks by Micheal Nyugen.
3. Schalk Wilhelm Pienaar1, Reza Malekian1,2, Senior Member, IEEE. (Guide for coding)

Human Activity Recognition Using LSTM-RNN.

1. WISDM Lab: Dataset. (For dataset) Available: http://www.cis.fordham.edu/wisdm/dataset.php#actitracker.
2. Jennifer R. Kwapisz, Gary M. Weiss and Samuel A. Moore (2010). Activity Recognition using Cell Phone Accelerometers, *Proceedings of the Fourth International Workshop on Knowledge Discovery from Sensor Data (at KDD-10)*, Washington DC
3. <https://www.tensorflow.org/lite/guide>
4. Dan Jarvis. Using a Pre-Trained TensorFlow Model on Android. (Blog)
5. [www.developer.android.com](http://www.developer.android.com) (Android documentation)