Project Report On

Detection of human motion with the help of tri-axis Accelerometer and tri-axis Gyroscope and prediction of activity type using Machine Learning Model

"A dissertation submitted in partial fulfillment of the requirements of 8th Semester 2020 Project-II (CS-892) examination in Computer Science and Engineering of the Maulana Abul Kalam Azad University of Technology"

Submitted by

Krishna Bose (10200116050)

Anubhab Nandy (10200116060)

Indranil Chatterjee (10200116051)

Ankan Mukherjee (10200116061)

Under the guidance of

Dr. Koushik Mondal
Principal System Engineer
IIT (ISM) Dhanbad, India

Certificate of Approval

This is to certify that this report of B. Tech 8th Sem, 2020 project, entitled "**Detection of human motion with the help of tri axis Accelerometer and tri axis Gyroscope and prediction of activity type using Machine Learning Model"** is a record of bona-fide work, carried out by **Ankan Mukherjee, Anubhab Nandy, Indranil Chatterjee, Krishna Bose** under my supervision and guidance.

In my opinion, the report in its present form is in partial fulfillment of all the requirements, as specified by the ***Kalyani Government Engineering College*** and as per regulations of the ***Maulana Abul Kalam Azad University of Technology***. In fact, it has attained the standard, necessary for submission. To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report for the Project-II (CS-892) 8th Sem B. Tech programme in Computer Science and Engineering in the year 2020.

Guide / Supervisor
Dr. Koushik Mondal
IIT(ISM) Dhanbad

Examiner(s)

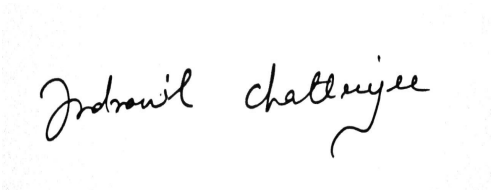
Head of the Department
Computer Science and Engineering
Kalyani Government Engineering College

Acknowledgement

We (**Ankan Mukherjee, Anubhab Nandy, Indranil Chatterjee, Krishna Bose**) would like to thank our mentor, **Dr. Koushik Mondal** for guiding us in this project and providing the oppurtunity to work on this project titled "**Detection of human motion with the help of tri-axis Accelerometer and tri-axis Gyroscope and prediction of activity type using Machine Learning Model**" where we researched a lot and learnt new technologies and built a solution to the project. We would also like to thank our friends and classmates who also helped us a lot in research.

Ankan Mukherjee

Anubhab Nandy



Indranil Chatterjee

Krishna Bose

Department of Computer Science and Engineering
Kalyani Government Engineering College

(Affiliated to Maulana Abul Kalam Azad University of Technology, West Bengal)
Kalyani - 741235, Nadia, WB

Krishna Bose[1], Anubhab Nandy[1], Indranil Chatterjee[1], Ankan Mukherjee[1], Koushik Mondal[2]
[1] Department of Computer Science and Engineering
Kalyani Government Engineering College, 741235, Kalyani, India
[2] Indian Institute of Technology (ISM), Dhanbad, India

Abstract

Smartphones and fitness bands have taken over the global market at a rapid pace and tracking physical activity of the users has become a new trend. With more sophisticated sensors and processing power available at our fingertips, it is now possible to harness its potential and detect a user's physical activities with high accuracy and low latency, offline in the user's device. Some of the established methodologies track a user's vital statistics and predict the general state of the user i.e, walking, standing, cycling or strenuous activity. These solutions either give a generalized idea about the user's state or utilize numerous sensors employed at different parts of the person's torso to deduce his state. Moreover, with data privacy concerns rising over the years we have reached a point where every piece of sensitive data collection needs to pass through a mammoth volume of governmental regulations. We propose to develop a model which is based purely on the positional and kinetic metrics of the person which can determine active and idle states, along with the type of activity, over a short span of time.

Table of contents

Introduction

Smartphones and fitness bands have taken over the global market at a rapid pace and tracking physical activity of the users has become a new trend. Some of the established methodologies track a user's vital statistics and predict the general state of the user like walking, standing, running etc activity. These solutions either give a generalised idea about the user's state or using the sensor data present in his handheld android phone. Standing in 2020, almost everyone has a smartphone with them all the time, and even the cheapest smartphones available now has a set of hardware sensors embedded in them which could be used for the purpose of activity recognition. The basic sensors are linear accelerometer, gyroscope and magnetometer. Since the newer smartphones have a good quality processor and sufficient memory on board, this is also feasible to run the deep learning model on their phones itself so that the detection can be performed offline without the use of a backend server or internet. In an attempt to understand how the above algorithms work, this report aims to build an deep learning based model which can detect and recognize several activities performed by the people bearing the smartphone equipped with the app. We have developed an android application which can collect the sensor data in realtime and store them locally until the session is complete. It then uploads the whole dataset to our server which forms a CSV file with the same and saves it on AWS S3 storage. It can then be fetched using script running on google colab which can be used to visualize or train the models as required.

Literature Survey

1. **'Deep learning for sensor-based activity recognition: A survey'**

Jindong Wanga, b , Yiqiang Chena, b,* , Shuji Haoc , Xiaohui Penga, b , Lisha Hua, b

Recent advances in wearable electronic devices have seen an amazing increase in applications targeted towards the detection and identification of activity that's being performed by the bearer of the same. The more traditional approaches towards tackling this task includes dependence on tailored heuristic feature extraction algorithms which consequently hinders their generalizability. Furthermore, the prevailing methods are ill-fitted for unsupervised and incremental learning tasks. The state-of-the-art deep learning procedures make it feasible for it to extract high level features from the information, hereby producing astonishing outputs in numerous areas. Presently many sensor-based activity determination techniques harness the potential of deep learning based methods. The cited paper aims to demonstrate the recent advances in the field of activity recognition, categorized from three aspects, namely sensor modality, application, and deep learning model implemented along with posing the future challenges and insights into the latest achievements

2. **'Detection of Type, Duration, and Intensity of Physical Activity Using an Accelerometer'**

BONOMI, A. G., A. H. GORIS, B. YIN, and K. R. WESTERTERP.

This study is aimed to produce a model which was capable of detecting specific physical activities along with their type, tenure and intensity. The data was accumulated by closely monitoring twenty subjects of age 29 ± 6 yr, and BMI of 23.6 ± 3.2 kg/m2. Selected subjects performed a set of predefined activities such as walking, running and cycling and were equipped with one tri-axial accelerometer, which was mounted on their lower back. The algorithm utilizes a decision tree to detect and accurately identify the activity. The acceleration data which is continuously monitored, is used to evaluate features of the acceleration signal. The feature set is tokenized into smaller segments of defined length, whose size determines the time resolution of the tree and assesses the duration of an activity. Multiple instances of the tree with a time resolution of 0.4, 0.8, 1.6, 3.2, 6.4 and 12.8s were developed and their comparative performance has been tabulated.

3. **Country Skiing Techniques Using Wearable Gyroscope Sensors**

Department of Industrial and Systems Engineering, Korea Advanced Institute of Science and Technology, Korea
Department of Mechanical and Industrial Engineering, IIT Roorkee, India
Division of Liberal Arts and Science, Korea National Sport University, Korea
Department of Physical Education, Korea National Sport University, Korea

The key idea behind this study is to develop an autonomous classification algorithm which can identify the different cross-country (XC) skiing techniques. The technique uses data

collected from wearable sensors to determine crucial statistics and generate insights for optimizing the performance which would be helpful to professional skiers. This paper proposes a deep learning model which could detect and recognize eight distinct techniques prevalent in classical and skating style XC skiing alongside optimizing the same for the quantity of gyroscope sensors deployed by conducting an comparative study of five different configurations produced by a varied number of sensor attachments. The data has been gathered from four professional skiers on flat and naturally occurring courses. The model is initially trained with the flat course data collected from two skiers and tested on the flat and natural course data received from a third skier, this was performed in a leave-one-out fashion, resulting in an average accuracy of ~80% over the course of three independent selections. Secondly, the model is trained with flat course data obtained from 3 skiers and tested on the flat and natural course data taken from one new skier, which resulted in an average accuracy of around 87.2% and 95.1% respectively, which used the previously identified sensor configuration of five gyroscope sensors (both hands, both feet, and the pelvis) which was identified to be most efficient.

Proposed Work

The total work planned can be summarised with the following bullets.

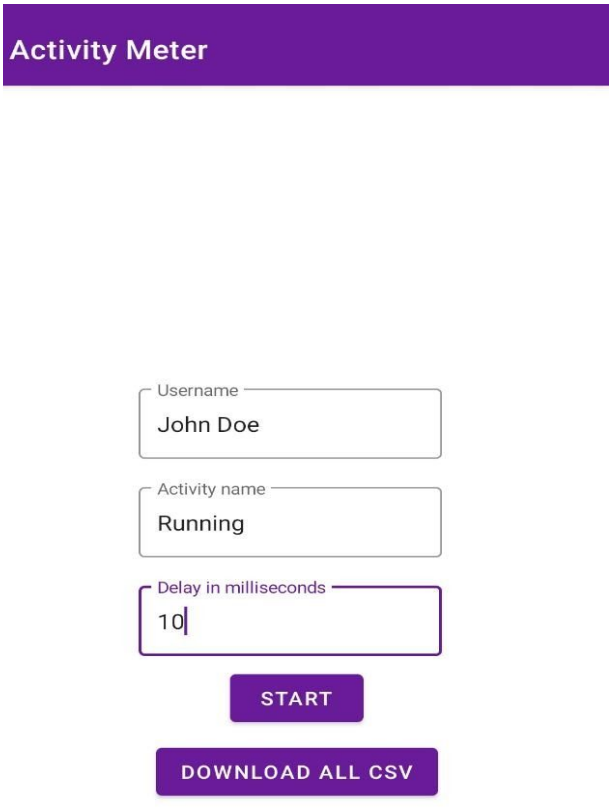
- Recognition and determination of a physical activity requires the movement data to be recorded and analysed, which could then be inferred into classes like running, walking or any specific sport.
- The abstract 'movement data' is quantified as linear acceleration in three mutually perpendicular axes, rotational velocity in three mutually perpendicular axes and orientation of the sensor with respect to earth's magnetic field.
- Required data is collected using a three axis gyroscope and a three axis linear accelerometer coupled with a magnetometer. Though the algorithm is hardware agnostic, for this experiment we have collected all the sensor data from the hardwares that is embedded inside available smartphones.
- The collected data is grouped in our mobile app which is formatted and serialized for further processing. This includes mapping the collected data with a specific timestamp each which could be later used to serialize the data in order and also the interval between two data points can be measured if required.
- The dataset hence obtained is sent to a cloud computer for inference. The backend server takes the data using a REST API, and generates a CSV file for the session. Along with hardware data, it also stores the type of activity and the user who performed it, for later usage. This is then stored in a cloud storage (Amazon S3) and the metadata is stored in a mongoDB. Later on a total list of available data can be fetched using this backend service which refers to this mongoDB for quick indexing of the stored activity data.
- We shall obtain various data such as orientation, acceleration, angular velocity and instantaneous velocity. The raw input from hardwares gives us the linear acceleration, gyroscope and magnetic field intensity in 3 perpendicular axes.
- Accelerometer data will be useful to predict the direction of motion and velocity with which the person is moving.
- Gyroscope data will be useful to predict the posture of hands of the person which will be crucial to distinguish between different types of activities.
- Magnetometer will be used to detect the earth's axis and align it with the user's axis. This can be then used to normalize the activity so that a person walking southwards and a person walking eastwards are not treated as different activities.
- Using the accelerometer the acceleration and velocity of the person along the three axes is measured and using the gyroscope the angular velocities along three axes.
- Before applying any algorithm the data needs to be filtered to modify or delete wrong values. We plan to use different filters for the same purpose and determine which gives the best results.
- A neural network will be used.
- Firstly one LSTM (Long Short Term Memory) layer with 256 nodes will be used to separate out the features. Since we are taking the data points with an interval of 10ms,

we get 100 data points in a second. To make the deep learning model understand the sequential nature of the data, we have divided the data into small segments of 256 data points each, and it corresponds to 2.5s of physical activity, which is enough to provide the nuances of movement and make them distinctly evident on the sequential segments.

- Also, to overcome the effects of broken ends due to data segmentation, we are planning to introduce data segment overlap and see if it helps to improve the accuracy of the model. We plan to use 0%, 25% and 50% overlap and see which performs the best.
- In the second stage a Dense layer of 64 nodes will be used. Even though the LSTM's output can be directly attached as input to the final layer containing 2 nodes, we shall be using an intermediate buffer layer to act as a bridge between them.
- In the third stage a deep neural network layer of 2 nodes will be used to classify the data into 2 categories respectively walking and running. This can be extended to match the number of activities we plan to detect.
- Log loss error, F1 score, accuracy will be calculated for different models and among them the best model will be selected.
- Accuracy will be taken into consideration for training the dataset.
- Adam optimizer will be used as an optimization algorithm.
- Relu and softmax activation functions will be used in intermediate and final layers.

Result and Discussion :

- An infrastructure for the proposed solution was planned and implemented.
- The infrastructure consists of some primary elements as mentioned below
    - An android application software.
    - A backend server hosted in the cloud.
    - Google collab engine is used for training machine learning models.
    - The deep learning model is saved after training the data.
    - After getting the data of the user an API call is made to the server where the data is tested on the pre trained model.
- Each of the above elements were connected using a dynamic pipeline for collecting, preprocessing, analysing and obtaining the results.
- The android app has been developed with the following features in mind:
    - The app has a homepage which take input with the following parameters Username,Activity_Name and the Delay in milliseconds:



- After Entering the following details the data is recorded in a sequential manner.
- The data from the three accelerometers, magnetometer and gyroscope of the x,y and z axis is taken.
- Data is logged using Android's **SensorManager** Class.
- SensorManager has many type of data, out of which only 3 are logged
    - TYPE_LINEAR_ACCELERATION
    - TYPE_GYROSCOPE
    - TYPE_MAGNETIC_FIELD
- Data is logged as per the DELAY_TIME provided by the user.
-
- Real time visual representation of the data collected for the user is displayed in the UI of the app.

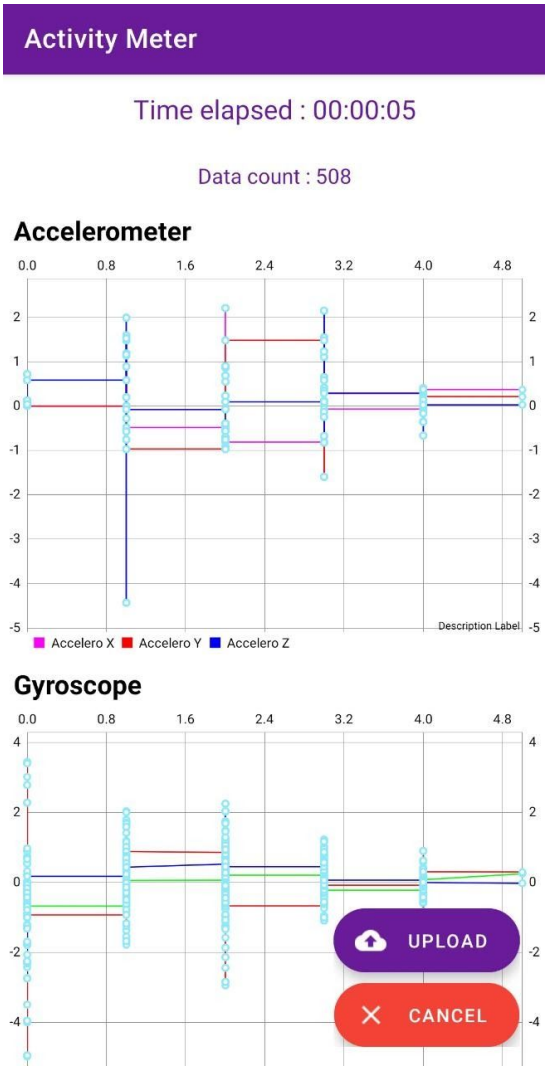


Fig:- The page displayed when sensor data logging is in progress

- Listeners are registered when the activity is created and the OnResume() method is called.

```
@Override
protected void onResume() {
   registerListeners();
   super.onResume();
}
```

```
private void registerListeners() {

sensorManager.registerListener(this,

sensorManager.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION),
           SensorManager.SENSOR_DELAY_NORMAL);
boolean is = sensorManager.registerListener(this,
           sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE),
           SensorManager.SENSOR_DELAY_NORMAL);
sensorManager.registerListener(this,

sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD),
           SensorManager.SENSOR_DELAY_NORMAL);
}
```

- onSensorChanged is called after there is a change in sensor data

```
@Override
public void onSensorChanged(SensorEvent sensorEvent) {
                  if       (sensorEvent.sensor.getType()       ==
Sensor.TYPE_LINEAR_ACCELERATION  ||  sensorEvent.sensor.getType()  ==
Sensor.TYPE_GYROSCOPE     ||     sensorEvent.sensor.getType()     ==
Sensor.TYPE_MAGNETIC_FIELD) {
      getData(sensorEvent);
   }
}
```

- The following function gets triggered after the change in sensor data from onSensorChanged() function

```
private void getData(SensorEvent event) {

   float[] values = event.values;
   float x = values[0];
   float y = values[1];
   float z = values[2];


   if(event.sensor.getType() == Sensor.TYPE_GYROSCOPE) {
       gyrox = x;
       gyroy = y;
```

```
        gyroz = z;

    }
    else if(event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD)
    {
        magnetox = x;
        magnetoy = y;
        magnetoz = z;

    }
                    else       if(event.sensor.getType()       ==
Sensor.TYPE_LINEAR_ACCELERATION)
    {
        timestampTemp = event.timestamp;
        accelerox = x;
        acceleroy = y;
        acceleroz = z;
    }

}
```

- addData() gets called after a DELAY_TIME and all the 6 variables get added to their respective list.

```
private  void addData()
{
    gyroxList.add(gyrox);
    gyroyList.add(gyroy);
    gyrozList.add(gyroz);

    acceleroxList.add(accelerox);
    acceleroyList.add(acceleroy);
    accelerozList.add(acceleroz);

    magnetoxList.add(magnetox);
    magnetoyList.add(magnetoy);
    magnetozList.add(magnetoz);
}
```

- On the click of the upload button the data collected is converted into a CSV format.

- The CSV file is then uploaded to the cloud storage along with its specific details against the particular user.

- On the click of the cancel button the user is taken back to the home screen where he can restart the process with a new set of parameters.

- On the server side the data is tested and a result is given by the pre - trained ML model which is shown in the UI of the mobile app.

- The server code is hosted on Amazon Web Services Cloud Platform.
- The entire stack of the backend code consists of the following item.
  - Flask Framework (Python)
  - MongoDb NoSQL Database
  - Github as a version control repository
  - Amazon S3 Bucket to store the CSV files
  - Amazon Lambda as the serverless architecture for our backend codes.

- The main file consists of the routes as described below with each of their functionality.
  - Method: GET
  - Route: '/'

This route is used to download all the csv files stored in the S3 bucket.

A snapshot of the code is present below.

```
@app.route('/', methods=['GET'])
def hello():
    return {
        'hello': 'noobs',
        'apis': [{
            'postfix':'/listFiles',
            'method': 'GET',
            'description': 'to list all the records present in the database'
        },
        {
            'postfix': '/postActivity',
            'method': 'POST',
            'description': 'create a new activity and send the details to store in the server',
            'fields': [
                {
                    'user': 'string'
                },
                {
                    'activity': 'string'
                },
                {
                    'startTime': 'string'
                },
                {
                    'endTime': 'string'
                },
```

- The second route is a POST method to take in the activities and convert it into a csv file and upload it to s3 bucket.The snapshot of the code displays the same.

```
@app.route('/postActivity', methods=['POST'])
def postActivity():
    try:
        body = request.get_json()
        reqParams = ['user', 'activity', 'startTime', 'endTime', 'timestamp', 'rotmax']
        hw = ['magnet', 'gyro', 'accelero']
        coord = ['x', 'y', 'z']

        for h in hw:
            for c in coord:
                reqParams.append(h+c)

        for x in reqParams:
            if x not in body:
                return invalidUsage('Missing Field: ' + x, 400)

        length = len(body['gyrox'])
        for h in hw:
            for c in coord:
                if len(body[h+c]) != length:
                    return invalidUsage('Array ' + h + c + 'has different length from gyrox', 400)

        # use the above 9 fields and create a file f

        fields = ['magnetx', 'magnety', 'magnetz', 'gyrox','gyroy','gyroz','accelerox','acceleroy','ac

        for i in range(0,16):
            fields.append('rotmax' + str(i))
        rows=[]
        for i in range (len(body["magnetx"])):
```

Here we can see the input parameters are taken as input and processed into a CSV file then uploaded into S3.

- The problem of activity recognition from recorded data was approached in 2 distinct way:-
    - Using a model named activity watch in which visualisation of data was done and the original path of the activity was attempted to be plotted in a 3D space.
    - Using a model in which conventional LSTM and dense layers were used over recorded data to figure out the activity type.
- In the first approach of activity watch:-
    - Direct synchronisation of data from cloud server to the google colab window is present.
    - Due to this link with the cloud any data recorded previously can be accessed using the indices of the data which is provided in a list of files.
    - To trace out the graph from the CSV file, plotly have been used.
    - The goal of Activity Watch is to visualize the data.

- Visualising the actual track of data means the path that has been followed by a particular individual while recording the data is tried to be retraced in this model.
- For this along with graphical tools provided in python like matplotlib we have used the magnetometer data and double integration over acceleration data recorded by the accelerometer.
- Double integration over acceleration data recorded by an accelerometer provides the distance traveled by the person who has recorded the data.
- Magnetometer provides the rotation matrix.
- The android creates its own virtual plain for recording data,which differs from phone to phone.A general convention is the perpendicular of the phone's plain is Z axis and the two horizontal plain on phone surface perpendicular to each other are X and Y axis.This leads to the conclusion that android's virtual plain is not aligned or same as original coordinate axes i.e Earth's coordinate axes.For example the East-West direction can be considered as X axis and similarly North-South as Y axis and up-down as Z axis.
- As the original Earth's coordinate is not aligned with the phone's virtual coordinate axes so the data recorded by phone cannot be directly considered to retrace the path of the person who recorded the data.
- Magnetometer data and rotation matrix was used for this purpose.Multiplying the data recorded in virtual plain with rotation matrix leads rotation of the virtual axis data into Earth's coordinate axes.Magnetometer provides the deviation from the original coordinate axes from which this rotation matrix is calculated at each instance.
- Axes rotation over origin is done in two ways :-
  - On acceleration data
  - On distance data which is achieved by doubly integrating the acceleration data.
  - The direct conversion of virtual axes acceleration data to real axes was not effective while calculating the distance from this acceleration.
  - The rotation over already converted distance data results better.

- One issue was faced in this stage.The path retraced from rotating distance data using rotation matrix was deviating from the original path followed by the user and the cause of this deviation was some errors.

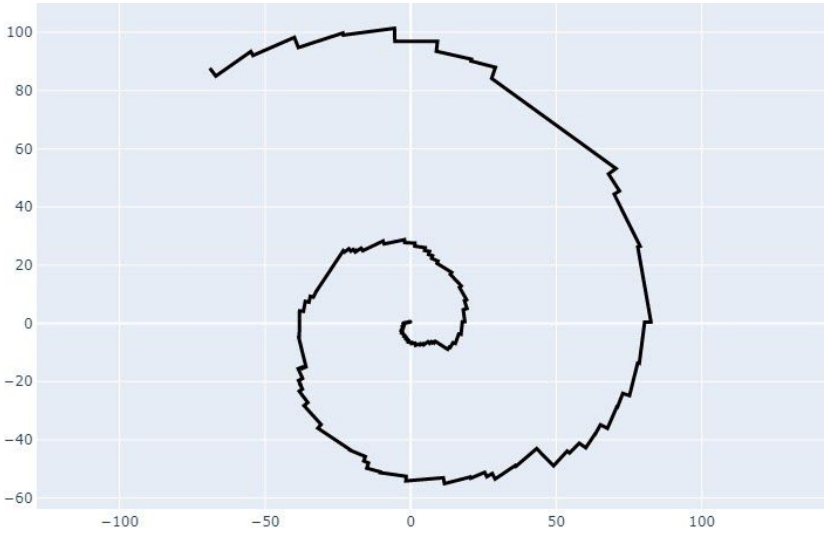


Fig:- This is a XY representation of a subject running in a proper circle. The outwardly biased noise causes the actual trajectory to end up looking like a spiral. (same behaviour was noticed for 3D representations as well)

- By retracing the whole strategy it was seen the reason behind this error was an inherent error produced by the accelerometer of the android.
- While integrating this accelerometer into distance travelled in virtual axes the distance was exponentially increasing due to an exponential error present in the original acceleration data which resulted in the deviation in the retraced path of the user.
- The error is not zero-mean error and it cannot be filtered using any filter.
- So the activity watch model is used to observe the acceleration data, and different smoothing filters like the Savitzky-Golay filter were used.
- The effect of these filters were used by this visualisation of techniques present in this model.
- Two type of observation is done in this model:-
  - The first observation is the 3 axes data observation in a single 3D graph which helps us to see the distance travelled altogether along virtual axes
  - The second observation is the individual observation of 3 axes data like acceleration or distance.
  - The second observation helps to determine the smoothing factor and window size of the smoothing filter applied over acceleration data.

- In the second model :-
  - The data recorded using Android have 9 columns.
    - 3 axes of Accelerometer
    - 3 axes of Gyroscope

    - 3 axes of Magnetometer
- Among them only accelerometer data and gyroscope data are used in a neural network.
    - Sample data of accelerometer and gyroscope of activity waking looks like:-

gyrox	gyroy	gyroz	accelerox	acceleroy	acceleroz
-0.08997	-0.06627	0.498657	-0.43599	-0.53423	-0.17535
0.041489	-0.00662	0.384277	0.204149	0.181479	0.210819
0.077515	-0.07996	0.510849	-0.38482	-0.74332	0.431431
-0.03058	0.047287	0.226486	-0.36214	-0.33383	-0.56797

- The libraries used for this model formation are :-

```
import requests
import numpy as np
import pandas as pd
import keras
from keras.models import Sequential
from keras.optimizers import Adam
from keras.layers import Dense,Flatten,LSTM,Dropout
from keras.metrics import categorical_crossentropy
from sklearn import metrics
from datetime import datetime
```

- LSTM and Dense neural network layers are used in combination to create a model which will predict the activity.
- The input data are raw gyroscope and accelerometer data separated into timestamps.
- LSTM takes 3D input data which is represented in the form of [batch_size,timestamps,features]
- In this 3D input data the first dimension represents the batch size data fed to the network.
- Each batch of data is a 2D matrix which contains timestamp and features.

- Timestamp represents a chunk of data of a particular time interval, LSTM takes this data as a single input and from that it derives the continuity relation of data points.
- Each timestamp of data contains several features. In this dataset there are 6 features namely gyrox, gyroy, gyroz, accelerox, acceleroy, acceleroz.
- The appropriate timestamp size problem is dependent, plug and play needs to be done to determine the appropriate timestamp size of this problem.
- The input shape of the LSTM of this model is [batch_size,timestamp,6].
- After one or two LSTM layer Dense layers are used. The final layer is always a dense layer with 2 nodes which will distinguish between 2 output classes.
- How many LSTM and Dense layers need to be used is a matter of testing and determining. Similarly how many nodes need to be there in the model that also needs to be tested.
- Batch size determines the number of data fit in one pass which also needs to be determined by testing.
- Learning rate of the optimizer is how small a step it will take towards the optimized value which needs to be determined by testing.
- The optimizer used in the neural network is adam's optimization algorithm.
- The Relu algorithm is used to calculate the weight of the nodes in the intermediate and first levels and softmax is used for the final level as softmax gives a probability value.
- Different variable parameters are used in plug and play mode to create the neural network .Those are:-
    - Number of layers : 3 to 6
    - Number of LSTM layers : 1 to 3
    - Number of Dense layers : 1 to 3
    - Batch Size : 1 to 20
    - Learning rate : 0.00001 to 0.001
    - Number of timestamps : 50 to 250
    - The optimizer used in the model : Adam
    - Activation function used : relu, softmax

        - The final model contains the parameter values as:-
            - Number of layers : 3
            - Number of LSTM layers: 1
            - Nodes in LSTM layers : 256
            - Number of Dense layers : 2
            - Number of nodes in first Dense layer : 64
            - Number of nodes in second Dense layer :- (output layers):-2
            - Batch Size : 1
            - Learning Rate : 0.00003

- Number of timestamps : 100
- The optimizer used : Adam
- Activation function in first layer: Relu
- Activation function in second layer : Relu
- Activation function in final layer: softmax

- The code for model creation is :-
  -

```
def model_create(number):
    model=Sequential()

model.add(LSTM(256,input_shape=(number,6),activation
="relu",batch_size=1))
    model.add(Dense(64,activation="relu"))
    model.add(Dense(2,activation="softmax"))

model.compile(Adam(lr=0.00003),loss="categorical_cro
ssentropy",metrics=["accuracy"])

    print(model.summary())
    return model
```

- The model Summary is :-

```
Model: "sequential_1"
_________________________________________________
_____________
Layer  (type)                                Output  Shape
Param #
=================================================
=============
lstm_1  (LSTM)                                (1,  256)
266240
_________________________________________________
_____________
dense_1  (Dense)                               (1,  64)
16448
_________________________________________________
_____________
dense_2  (Dense)                               (1,  2)
130
=================================================
=============
Total params: 282,818
Trainable params: 282,818
```

```
Non-trainable params: 0
```

- The timestamp used in this model is 100, so the input shape becomes [batch_size, 100, 6]
- So each set of data is trimmed into a particular size suitable for creating the train data.
- Equally from starting and ending data is deleted for creating suitable size.
- The trim function used is :-
  -

```
def trim(data,val):
 l1=int((data.shape[0]-val)/2)
 l=data.shape[0]-val-l1
 d1=list(np.arange(val+l,data.shape[0]))
 print(len(d1))
 d2=list(np.arange(0,l))
 print(len(d2))
 data.drop(d1,inplace=True)
 data.drop(d2,inplace=True)
 print(data.shape[0])
 return data
```

- The data is subdivided into [batchsize, 100, 6] using the function :-
  -

```
def shaping(data,number):
 ll=[]
 print(data.shape)
 print(data.columns.values)
 print(number)
 for n in range(int(data.shape[0]/number)):
   xx=np.array(data["accelerox"].iloc[number*n:number*(n+1)])
   yy=np.array(data["acceleroy"].iloc[number*n:number*(n+1)])
   zz=np.array(data["acceleroz"].iloc[number*n:number*(n+1)])
   aa=np.array(data["gyrox"].iloc[number*n:number*(n+1)])
   bb=np.array(data["gyroy"].iloc[number*n:number*(n+1)])
   cc=np.array(data["gyroz"].iloc[number*n:number*(n+1)])
   l=[]
   for xyz in range(len(xx)):
      temp=np.array([xx[xyz],yy[xyz],zz[xyz],aa[xyz],bb[xyz],cc[xyz]])
      l.append(temp)
```

```
    l=np.array(l)
    ll.append(l)
```

- Labels of train and valid dataset are created :-

```
def label_create(trainsize,testsize,validsize,cho):

    ytrain=list(np.zeros((1,int(trainsize/2))).ravel().astype("int"))
    ytrain+=list(np.ones((1,int(trainsize/2))).ravel().astype("int"))
    ytrain=keras.utils.to_categorical(np.array(ytrain))
    yvalid=list(np.zeros((1,int(validsize/2))).ravel().astype("int"))
    yvalid+=list(np.ones((1,int(validsize/2))).ravel().astype("int"))
    yvalid=keras.utils.to_categorical(np.array(yvalid))

    if cho==0:
        ytest=[1]
        ytest+=list(np.zeros((1,testsize)).ravel().astype("int"))
        ytest=keras.utils.to_categorical(np.array(ytest))
        ytest=np.delete(ytest,[0],axis=0)
    if cho==1:
        ytest=[0]
        ytest+=list(np.ones((1,testsize)).ravel().astype("int"))
        ytest=keras.utils.to_categorical(np.array(ytest))
        ytest=np.delete(ytest,[0],axis=0)


    return (ytrain,ytest,yvalid)
```

- Model is studied using the code :-

```
def model_study(model,xtrain,xtest,xvalid,ytrain,ytest,yvalid):

model.fit(xtrain,ytrain,epochs=10,verbose=2,batch_size=1,validation_data=(xvalid,y
valid))
    predictions = model.predict(xtest,batch_size=1,verbose=0)

   return metrics.accuracy_score(ytest.argmax(axis=1), predictions.argmax(axis=1))
```

- The output distinguishes between running and walking using this model.Any test input data is subdivided into 100 timestamp blocks as prescribed before and fed to the neural networks for prediction.
- The accuracy of both the activities with which the model classifies are:-

Running	Walking
79.5%(approx)	85.4%(approx)

Conclusion:

The aim of the project was to detect, identify and infer the type of activity a person is undergoing while carrying the input device along with them. It infers what kind of activity the person is performing like running or walking . For future work, the data collected can be used to determine workout stats such as the calories burnt in a certain period of time. An application software will be built to monitor the activity and send periodic notifications regarding his vital stats. As a future aspect of the project this can also be later modified to predict how fitness of someone can be maintained. By tracking the workflow of a certain period for a certain time period, these can be used for personal digital health assistants.

Reference:

1. Deep learning for sensor-based activity recognition: A survey
Jindong Wanga,b, Yiqiang Chena,b,∗, Shuji Haoc, Xiaohui Penga,b, Lisha Hua,b

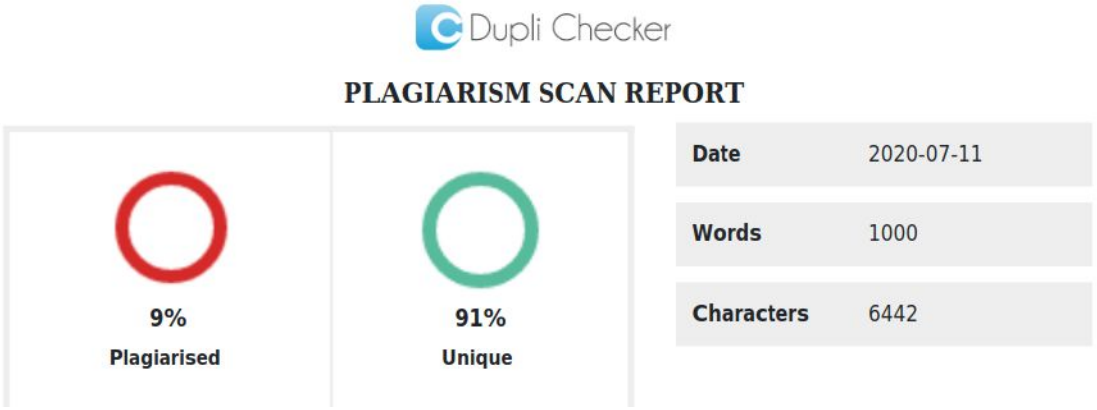
2. Detection of Type, Duration, and Intensity of Physical Activity Using an Accelerometer
ALBERTO G. BONOMI1,2, ANNELIES H.C. GORIS3 , BIN YIN4 , and KLAAS R. WESTERTERP1
BONOMI, A. G., A. H. GORIS, B. YIN, and K. R. WESTERTERP

3. Country Skiing Techniques Using Wearable Gyroscope Sensors
Department of Industrial and Systems Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 34141, Korea
Department of Mechanical and Industrial Engineering, Indian Institute of Technology Roorkee, Uttarakhand 247667, India
Division of Liberal Arts and Science, Korea National Sport University, Seoul 05541, Korea
Department of Physical Education, Korea National Sport University, Seoul 05541, Korea

Plagiarism report:

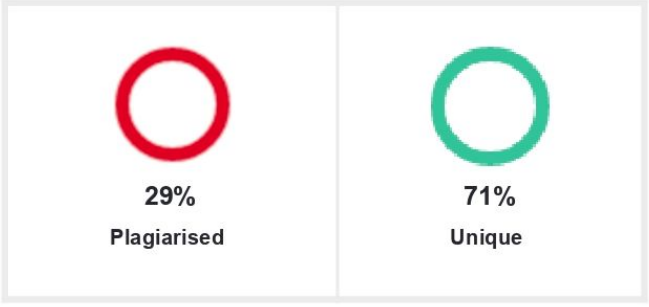
The abstract of this report is 100% unique and the literature survey is 94% unique. Both the proposed work and conclusion are 86% unique, along with the reference being 71% unique.
The average plagiarism of the whole report is 88% unique.
The key duplicacy in the literature survey and reference section of this report arises from the fact that the paper name and authors names of the surveyed papers are quoted verbatim from the source whence these are obtained. This is flagged by the automated detectors as an act of plagiarism and we abide by its statistics notwithstanding the fact that none of the original content in this report was inspired by any standard sources available in public/restricted domains.

Some sample of our reports are given below with the following images:



Dupli Checker

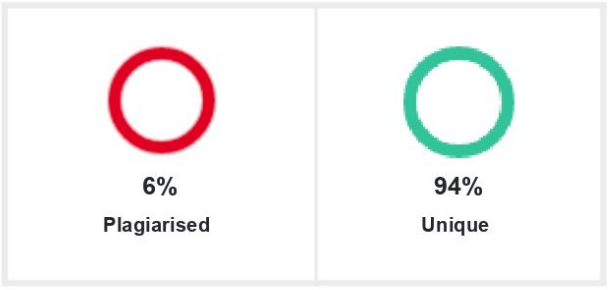
PLAGIARISM SCAN REPORT



Date	2020-07-11
Words	250
Characters	1604

Dupli Checker

PLAGIARISM SCAN REPORT



Date	2020-07-11
Words	990
Characters	6481

Dupli Checker

PLAGIARISM SCAN REPORT

Date	2020-07-11
Words	949
Characters	5950