International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/CSEIT228442

Human Activity Classification Using Deep Learning

Irfan Ayoub, Bhawna Sharma, Sheetal Gandotra, Manisha Manhas, Mehak Sharma, Umar Farooq
Department of Computer Engineering, Government College of Engineering and Technology, Jammu, India

Article Info
Publication Issue :
Volume 8, Issue 5
September-October-2022
Page Number : 64-71
Article History
Accepted: 10 Sep 2022
Published: 24 Sep 2022

ABSTRACT

This paper describes a method to classify human activities using accelerometer data by training a deep learning model and using it in an android app which gathers real time accelerometer data while the device is with user and classifies his activity by assigning a probabilistic value with highest probability being the activity predicated. The dataset used in this paper is freely available which is provided by WISDM Lab and Google cloud-based instances running tensor flow library for python to code and train the model.

Keywords : Dataset, Deep Learning, Long Short-Term Memory (LSTM) Model, Tensor Flow.

I. INTRODUCTION

The activity classification is an important part of many medical diagnostic approaches. It also has a wide application in consumer wearable market in activity trackers, health monitors and AR/VR applications. Smartphones nowadays ship with inertial measurement units such as accelerometers and gyroscopes of great accuracy and precision. This paper explains an approach to train a machine learning model to gather and predict the human activity by using data collected from accelerometer sensor of a smartphone and using it in an android application for prediction of basic human activities such as walking, running, sitting, standing, climbing stairs and so on. The dataset used is provided by Wireless Sensor Data Mining (WISDM) Lab which has 1,098,207 rows and 6 columns. The columns contain the serial no., user, activity, x-axis, y-axis and z-axis data during said activity. A Long Short-Term Memory (LSTM) Neural Network is trained in tensor flow and further used in android application. All machine learning code is written in Python and uses Google's open source machine learning library Tensor Flow for training the model. The android application is built using java in android studio and incorporating the exported trained model from last stage.

II. SYSTEM OVERVIEW

The Basic components of the system are Data Import, Data Preprocessing, Model Training, Model Testing, Model Exporting, Android Build and Activity prediction.
The overall system consists of 3 main stages starting with data preprocessing, training and android app integration. The python library pandas is used for data preprocessing and for training the ML model python tensor flow library is used, and finally after generating the model we have used it by exporting the Shared Object file (.SO file) inside the android application written in Java.

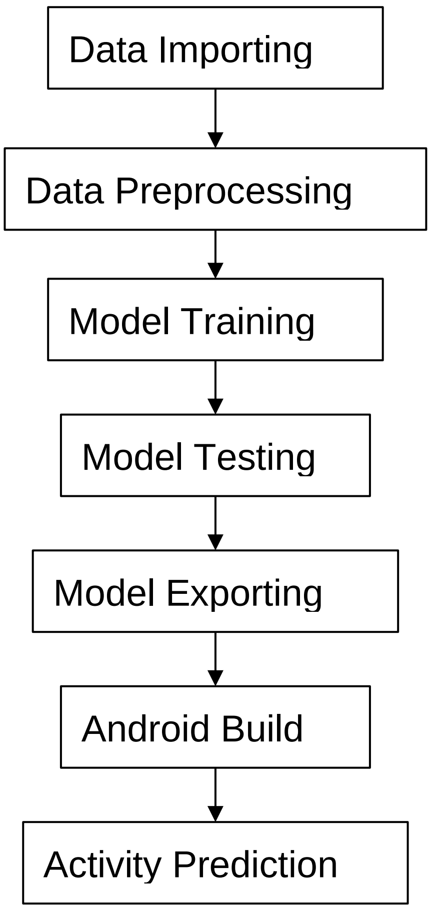


Fig.1. Flow Chart of Model

The software used includes anaconda for running the python code interactively using Jupyter notebooks so as to make it easier to track progress and save the work. Jupyter notebooks provide an interactive way to run python files line by line making execution intuitive and debugging easier.

The android application developed in this runs on android devices from version 6.0 above and is fully compatible with all kinds of different architectures.

III. SYSTEM SPECIFICATIONS

A Google cloud instance was used, running 32 CPU cores and 64 GB RAM with an attached 265 GB SSD running windows server 2016 datacenter. We used Anaconda distribution of python to run the python code; and Android studio to build the android application. For testing the application we used Samsung Galaxy S8 running Android version 9.

IV. DATA IMPORTING AND EXPLORATION

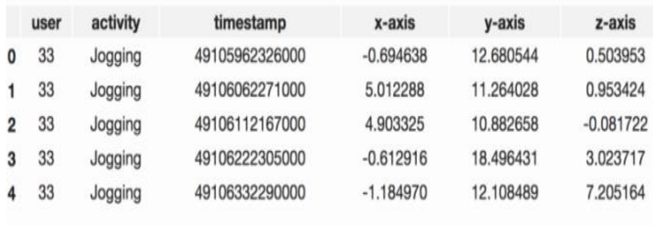
A free available data from site hosted by WISDM Lab, Fordham University, Bronx, NY was downloaded. The data was in CSV(Comma-Separated Values) format and we stored it locally in the folder that was being used for the project. This data was collected in a controlled setting by the lab. The data contains 6 columns namely the user, activity, timestamp of the activity, X, Y, Z axis values of the accelerometer while the activity is being performed.

The data was imported into a variable by using Pandas function.

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

   df = pd.read_csv (
'data/WISDM_ar_v1.1_raw.txt', header = None,
names = columns )
```

Taking a look, it is organized as following in tabular form:

df.head( )



	user	activity	timestamp	x-axis	y-axis	z-axis
0	33	Jogging	49105962326000	-0.694638	12.680544	0.503953
1	33	Jogging	49106062271000	5.012288	11.264028	0.953424
2	33	Jogging	49106112167000	4.903325	10.882658	-0.081722
3	33	Jogging	49106222305000	-0.612916	18.496431	3.023717
4	33	Jogging	49106332290000	-1.184970	12.108489	7.205164

Fig.2. Dataset Snapshot

For pictorial representation of data and to get an estimate of data points for each of the activities, the data was plotted as a bar chart using a simple value count function for both the activities and users.

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

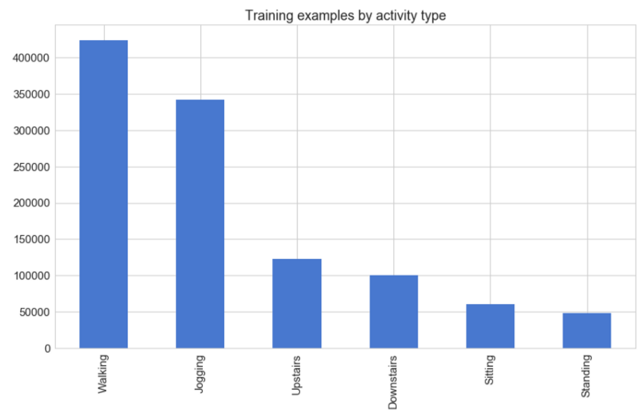


Fig. 3. Activity wise data plot

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

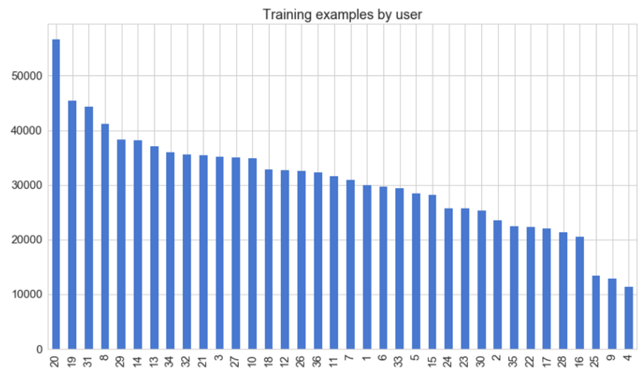


Fig. 4. User wise data plot

Next, to visualize the accelerometer data collected for each of the activities, Graphs were plotted as:

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

 plot_activity("Sitting", df)
```

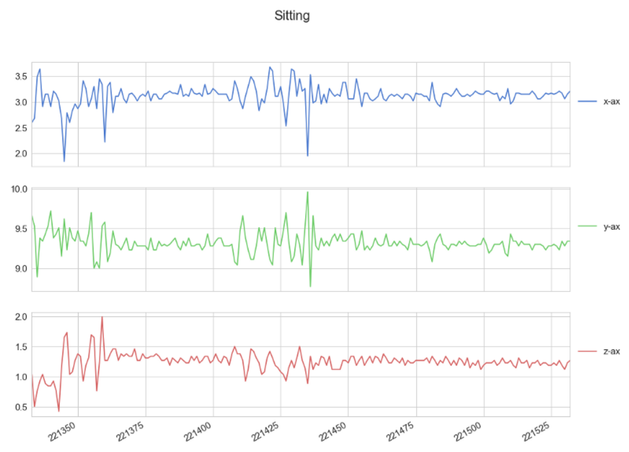


Fig. 5. Sitting Accelerometer data plot

plot_activity("Standing", df)

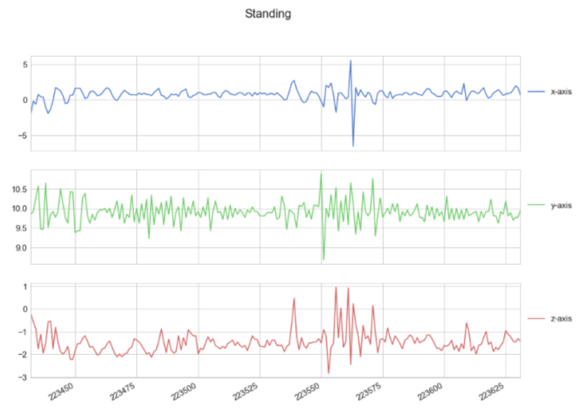


Fig. 6. Standing Accelerometer data plot

plot_activity("Jogging", df)

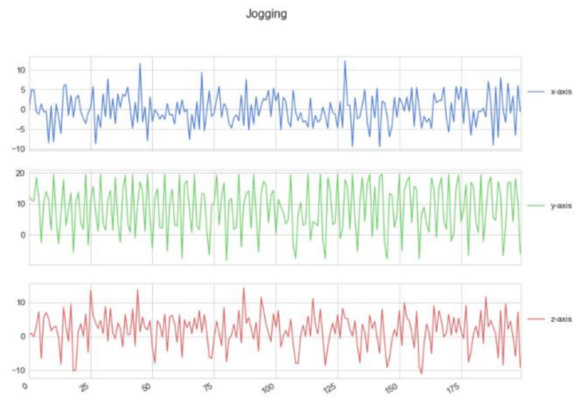


Fig 7. Jogging Accelerometer data

plot_activity("Walking", df)

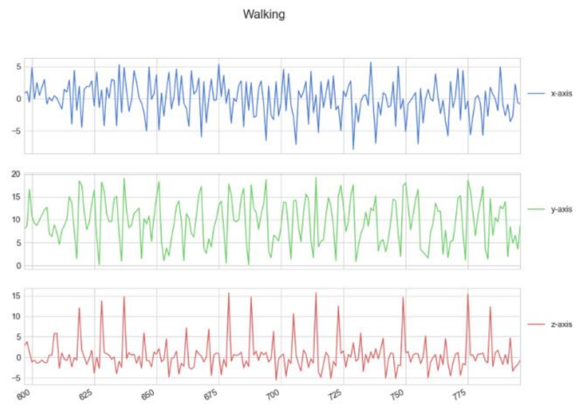


Fig. 8. Walking Accelerometer data plot

V. DATA PROCESSING

For LSTM training fixed length data segments of length 200 were created.

N_TIME_STEPS = 200

N_FEATURES = 3

```
step = 20
segments = []
labels = []
for i in range(0, len(df) - N_TIME_STEPS,
step):
    xs=df['x-axis'].values[i:i+N_TIME_STEPS]
    ys=df['y-axis'].values[i:i+N_TIME_STEPS]
    zs=df['z-axis'].values[i:i+N_TIME_STEPS]
 label    =    stats.mode    (df['activity']
[i:i+N_TIME_STEPS])[0][0]
    segments.append([xs, ys, zs])
    labels.append(label)
```

The result of transformation is:

```
np.array(segments).shape

>

(54901, 3, 200)
```

That is 54901 samples of 3 rows and 200 columns. Now transforming this into 200 rows of 3 columns each.

```
reshaped_segments=np.asarray(segments,dtype=
np.float32).reshape(-
1,N_TIME_STEPS,N_FEATURES)
labels=np.asarray(pd.get_dummies(labels),dty
pe=np.float32)
```

Thus giving us segmesnts of shapoe as:

reshaped_segments.shape

>

(54901, 200, 3)

Final step of transformation is to split the data into 2 giving 80% to training the model and 20% for validating the model.

```
X_train,X_test,y_train,y_test=train_test_spl
it(reshaped_segments,labels,test_size=0.2,
random_state=RANDOM_SEED)
```

VI. MODELBUILDING AND TRAINING

The model was trained using the available data. The model consisted of stacked two fully-connected and two LSTM layers with 64 units for each of these. Further a placeholder was created for the model and L2 regulations were used and it was accounted in the loss. And finally accuracy and optimizer ops were defined.

```
N_CLASSES = 6
N_HIDDEN_UNITS = 64
def create_LSTM_model(inputs):
    W = {
        'hidden':
tf.Variable(tf.random_normal([N_FEATURES,
N_HIDDEN_UNITS])),
        'output':
tf.Variable(tf.random_normal([N_HIDDEN_UNITS
, N_CLASSES]))
    }
    biases = {
        'hidden':
tf.Variable(tf.random_normal([N_HIDDEN_UNITS
], mean=1.0)),
        'output':
tf.Variable(tf.random_normal([N_CLASSES]))
    }


    X = tf.transpose(inputs, [1, 0, 2])
    X = tf.reshape(X, [-1, N_FEATURES])
    hidden = tf.nn.relu(tf.matmul(X,
W['hidden']) + biases['hidden'])
    hidden = tf.split(hidden, N_TIME_STEPS,
0)
    # Stack 2 LSTM layers
    lstm_layers =
[tf.contrib.rnn.BasicLSTMCell(N_HIDDEN_UNITS
, forget_bias=1.0) for _ in range(2)]
    lstm_layers =
tf.contrib.rnn.MultiRNNCell(lstm_layers)
    outputs, _ =
tf.contrib.rnn.static_rnn(lstm_layers,
hidden, dtype=tf.float32)
    # Get output for the last time step
    lstm_last_output = outputs[-1]
    return tf.matmul(lstm_last_output,
W['output']) + biases['output']
tf.reset_default_graph()
X = tf.placeholder(tf.float32, [None,
N_TIME_STEPS, N_FEATURES], name="input")
Y = tf.placeholder( tf.float32, [None,
N_CLASSES])
pred_Y = create_LSTM_model(X)
pred_softmax = tf.nn.softmax( pred_Y,
name="y_")
L2_LOSS = 0.0015
l2 = L2_LOSS * \
    sum(tf.nn.l2_loss(tf_var) for tf_var in

tf.trainable_variables())
loss = tf.reduce_mean(
tf.nn.softmax_cross_entropy_with_logits(logi
ts = pred_Y, labels = Y))+l2
LEARNING_RATE = 0.0025
optimizer = tf.train.AdamOptimizer(
learning_rate=LEARNING_RATE).minimize(loss)
correct_pred = tf.equal(
tf.argmax(pred_softmax, 1), tf.argmax(Y, 1))
accuracy = tf.reduce_mean(
tf.cast(correct_pred, dtype=tf.float32))
LEARNING_RATE = 0.0025
optimizer = tf.train.AdamOptimizer(
learning_rate=LEARNING_RATE).minimize(loss)

correct_pred = tf.equal( tf.argmax(
pred_softmax, 1), tf.argmax(Y, 1))
accuracy = tf.reduce_mean( tf.cast(
correct_pred, dtype=tf.float32))
```

Since TensorFlow part was setup, the model was started to be trained for 50 epochs and a track of accuracy and error was kept.

```
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
```

Now that training was complete, the model was stored on the disk.

```
pickle.dump(predictions, open(
"predictions.p", "wb"))
pickle.dump(history, open("history.p",
"wb"))
tf.train.write_graph( sess.graph_def, '.',
'./checkpoint/har.pbtxt')
saver.save( sess, save_path =
"./checkpoint/har.ckpt")
sess.close()
```

VII. MODEL TESTING AND EXPORTING

Now the model was trained and stored onto disk. On Loading it back and plotting the test curve for each epoch, it was observed that the model achieved 97% of accuracy and a loss of around 0.2.

```
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()
```

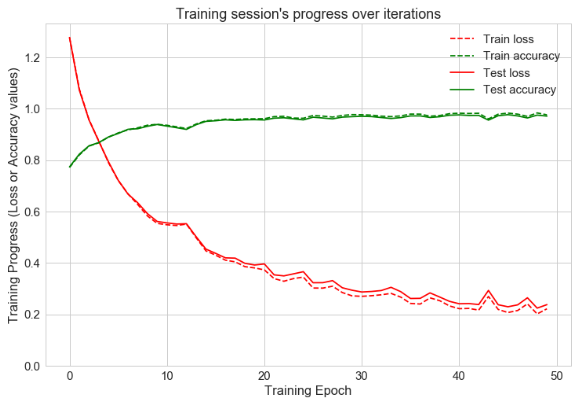


Fig. 9. Training Accuracy plot

The confusion matrix for each activity is given as:

```
LABELS = ['Downstairs', 'Jogging',
'Sitting', 'Standing', 'Upstairs',
'Walking']
max_test = np.argmax(y_test, axis=1)
max_predictions = np.argmax(predictions,
axis=1)
confusion_matrix = metrics.confusion_matrix(
max_test, max_predictions)
plt.figure(figsize=(16, 14))
sns.heatmap(confusion_matrix,
xticklabels=LABELS, yticklabels=LABELS,
annot=True, fmt="d");
plt.title("Confusion matrix")
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show();
```

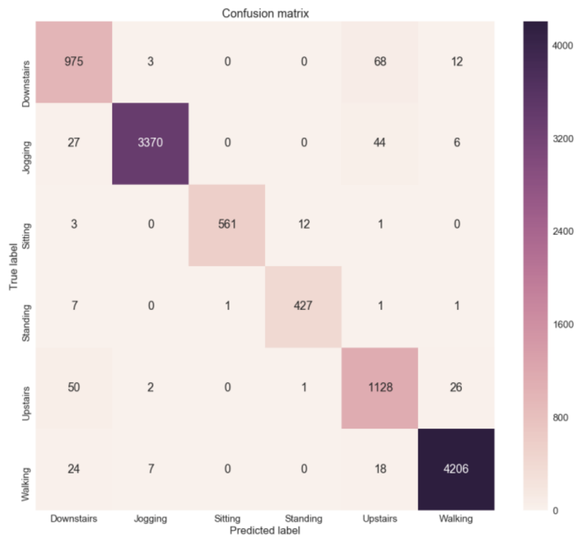


Fig 10. Confusion Matrix

This shows that the model works great for all activities with some confusion between upstairs and downstairs and some exceptions in case of jogging. Now as the model was ready and tested, it was finally exported for android so that it could be used to interface within the android application.

```
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="" )
```

VIII. ACTIVITY PREDICTION

Finally, in the android application, input and output dimensions and names were defined.

```
String INPUT_NODE = "inputs";
String[] OUTPUT_NODES = {"y_"};
String OUTPUT_NODE = "y_";
long[] INPUT_SIZE = {1, 200, 3};
int OUTPUT_SIZE = 6;
```

Then a tensorflow interface was created for interaction with the saved model.

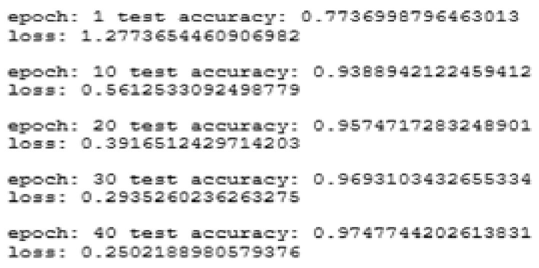
```
inferenceInterface = new
TensorFlowInferenceInterface (
context.getAssets(), MODEL_FILE);
```

And finally, code for interacting with the model interface for each real time value of input gathered from accelerometer was written.

```
public float[] predictProbabilities(float[]
data) {
    float[] result = new float[OUTPUT_SIZE];
    inferenceInterface.feed(INPUT_NODE,
data, INPUT_SIZE);
    inferenceInterface.run(OUTPUT_NODES);
    inferenceInterface.fetch(OUTPUT_NODE,
result);
    return result;
}
```

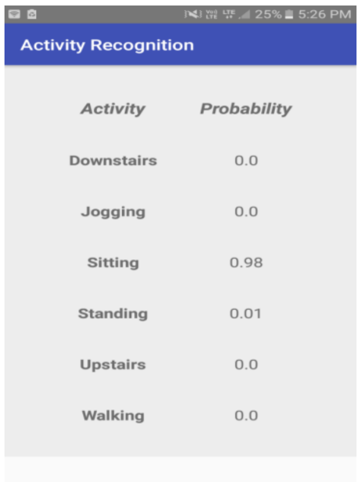
IX. RESULTS

The product developed is a model that predicts human activity using LSTM with 200-time steps, achieving an accuracy of 97% and the model built was exported to be used in android application by interfacing with live accelerometer data.



```
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
```

Fig. 11. Accuracy achieved after training the model



5:26 PM 25%

Activity Recognition

Activity	Probability
Downstairs	0.0
Jogging	0.0
Sitting	0.98
Standing	0.01
Upstairs	0.0
Walking	0.0

Fig. 12. Activity recognized by the android application when user is in sitting position.

X. ACKNOWLEDGMENT

We would like to thank WISDM Lab, Fordham University, Bronx, NY for the data and a sincere thanks to Hiteshwar Sharrna for his guidance and assistance during the duration of project.

[1] NSC, "Injury Facts Online. Itasca, Ill.: National Safety Council'. Available at: www.nsc.org. Accessed on 17 December 2001.

XI. REFERENCES

[1]. Mart´ın Abadi et al, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems". Available at: https://www.tensorflow.org 2015.

[2]. C. E. Matthews et al, "Amount of time spent in sedentary behaviors and cause-specific mortality in US adults", American Journal of Clinical Nutrition 2012. Available at: www.ncbi.nlm.nih.gov/pubmed/22218159.

[3]. Xiaoyong Chai and Qiang Yang, "Multiple-Goal Recognition from Low-Level Signals". In: AAAI (2005). URL: http://www.aaai.org /Papers/AAAI/2005/AAAI05-001.pdf.

[4]. Trevor Hasting, "The Elements of Statistical Learning." Available at : https://web.standford.edu/~hastie/Papers/ESLII.pdf.

[5]. Panu Korpipa Juha Parkka Miikka Ermes. "Activity Classification Using Realistic Data From Wearable Sensors". In: IEEE Transactions On Information Technology In Biomedicine, IEEE, 2006. Available at: https://s3.amazonaws.com/academia.edu.documents/12195796/parkka06.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL&Expires=544770601&Signature=QBZmGI0nx4\%2FLEqsaBbGl6Sm1AXQ\%3D&response-content-disposition=inline\%3B\%20filename\%3DActivityClassificationUsingRealistic.pdf

[6]. P. T. Katzmarzyk and I-M. Lee. "Sedentary behavior and life expectancy in the USA: A cause-deleted life table analysis". In: BMJ Open (2012). URL: https://bmjopen.bmj.com/content/2/4/e000828.

[7]. Youngwook Kim and Hao Ling. "Human Activity Classification Based on Micro-Doppler Signatures Using a Support Vector Machine". In: IEEE Transactions On Geoscience And Remote Sensing. IEEE, 2009. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4801689&tag=1.

[8]. Nie Lanshun Li Jiazhen Ding Renjie Zhan Dechen Chu Dianhui Li Xue Si Xiandong. "Understanding and Improving Deep Neural Network for Activity Recognition". In: School of Computer Science and Technology, Harbin Institute of Technology. Available at:: https://arxiv.org/ftp/arxiv/papers/1805/1805.07020.pdf.

[9]. B; Aarts R.M Long X.; Yin. "Single-accelerometerbased daily physical activity classification". In: Proceedings of the EMBC. IEEE, 2009. Available at: https://pure.tue.nl/ws/files/2828549/Metis234421.pdf.

[10]. Francis Bach Mark Schmidt Nicolas Le Roux, "Minimizing Finite Sums with the Stochastic Average Gradient", 2nd ed.Springer.

[11]. Andrew Ng. CS229 Course Notes Decision Trees. URL: http://cs229.stanford.edu/notes/rf-notes.pdf.

[12]. Thomas Plotz Nils Y. Hammerla1 Shane Halloran2. "Deep, Convolutional, and Recurrent Models for Human Activity Recognition using Wearables". In: Open Lab, School of Computing Science, Newcastle University, UK. Available at: https://arxiv.org/pdf/1604.08880.pdf.

[13]. Rajan Patel. STATS 202 Lecture 19. Available at:

https://web.stanford.edu/class/stats202/content/lec19-cond.pdf.

[14]. F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: Journal Of Machine Learning Research 12, pp.2825–2830,2011

[15]. Attila Reiss. “PAMAP2 Dataset: Physical Activity Monitoring”. Available at: : https://archive.ics.uci.edu/ml/machinelearning - databases/ 00231/ readme . pdf. UCI Machine Learning dataset.

[16]. Attila Reiss and Dider Stricker. “Introducing a New Benchmarked Dataset for Activity Monitoring”. In: IEEE Computer Society Washington (2012). URL: https://dl.acm.org/citation.cfm?id=2358027.

Cite this article as :

Irfan Ayoub, Bhawna Sharma, Sheetal Gandotra, Manisha Manhas, Mehak Sharma, Umar Farooq, "Human Activity Classification Using Deep Learning", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 8 Issue 5, pp. 64-71, September-October 2022. Available at doi : https://doi.org/10.32628/CSEIT228442
Journal URL : https://ijsrcseit.com/CSEIT228442