Implementation Project
On

"LSTM FOR HUMAN ACTIVITY RECOGNITION USING WEARABLE SENSORS"

Submitted by

Mr. Suraj Bhute

Roll No: - 21MM61R03

Program: - M.TECH (Medical Imaging and Informatics)
August 2021-Nov 2021

Under the guidance of

Prof. Debashish Sen

Subject: - Neural Networks and Application

Indian Institute of Technology Kharagpur

(Kharagpur, West Bengal 721302)

Table of Content

LIST OF FIGURES

LSTMs for Human Activity Recognition

(Implementation Project)

Abstract:

(This is an implementation project)

This project finds the opportunity for applying the LSTM architecture from recurrent neural network (RNN) to real world problem. The objective of the project is to predict six human activities like walking, sitting, upstairs, downstairs, standing, laying using the data obtain from Accelerometer (Acceleration measurement) and Gyroscope (Angular Velocity measurement) sensors used in smartphone or smart watches. Our dataset is public domain UCI data set[2]. The LSTM architecture which is used to solve the problem with only one layer of LSTM, which effectively avoiding the gradient vanishing problem.

Without using handcrafted engineered features and trying out various classical machine learning technique, this LSTM model gives accuracy more than 90%. It increases if the dataset is huge.

Finally the accuracy, loss and confusion matrix of UCI data set was analysed. From the confusion matrix it is showing the excellent classification of data points among six classes. Also from the loss plot of training and testing it is clear that our model is converging.

Keywords: Long Short-Term Memory (LSTM); Human Activity Recognition; Recurrent Neural Network; UCI dataset

1. Introduction:

1.1 Modern Smartphone and Smart watches Industry

From past few years, we have seen the increasing demand of smartphones and smart watches. We are observing continuous growth of these products. Most of the technological breakthroughs in the smart watches industry have revolutionized the production component through advanced sensors. The global smart watch market was estimated at $ 20.64 billion by 2019, with the market expected to be worth $ 96.31 billion by 2027.In modern society; health/fitness is a major concern. We know some powerful smart watches such as fit-bit, Apple watch. With powerful sensors such as accelerometer and gyroscope, these watches showing the heart beat rate, Calories burnt, sleep hours, walking steps, etc.

1.2 Human Activities and Health

Human Activity recognition is a time series problem which classifying the sequence of accelerometer and gyroscope recorded data on a smartphone or smart watches into well-defined movements. We all have going through very tough corona situation and we can understand the importance of health as well. By using these advanced smart watches, we can measure calories burnt, heart beat rate. So health point of view these products are very crucial to continuously monitor the health. Some research study proved that the smart watches like Apple can help in predicting pre-symptomatic detection of COVID-19

1.3 Problem Statement

For our human activity recognition activity, Accelerometer and gyroscope sensors data are used. These Accelerometer and gyroscope sensors are tri-axial sensors i.e. X, Y and Z direction it measures acceleration and angular velocity. So for our model these sensors 6 time series data is given as input and we are predicting the one of the six activities as an output. So it is a multiclass classification problem.

1.4 Classical Machine Learning Algorithm

Before application of Classical machine learning algorithm, it requires some pre-processing. It involves data cleaning (checking for duplicates, checking for null values, checking for data imbalance), Number of data points, Changing features names etc. Then moving towards the exploratory data analysis. Featuring Engineering from respective domain knowledge.

Without domain knowledge, exploratory data analysis is meaningless and without exploratory data analysis a problem is difficult to solve. Then trying out various algorithms like Logistics regression, decision tress, Support vector machine, Random forest and predicting the Accuracy and multiclass log loss.

1.5 Long Short-Term Memory (LSTM)

Long Short-Term Memory networks are a special kind of recurrent neural network. It works tremendously well on large varieties of problems and it is widely used. LSTM with only single layer approach shows best result compared to the other classical machine learning algorithm.

2. Dataset:

2.1 Human Activity Recognition

This project is to build a model that predicts human activities such as Walking, Walking Up, Walking Down, Sitting, Standing or Laying. This database is gathered for 30 people (referred to as Subject in this database), who perform various tasks with a smartphone at their waist. Data is recorded with the help of sensors (accelerometer and Gyroscope) on that smartphone. This test was recorded for personal data label.

2.2 How the data was recorded

Using sensors (Gyroscope and accelerometer) on the smartphone, they captured '3-axial linear acceleration'(tAcc-XYZ) from accelerometer and '3-axial angular velocity'(tGyro-XYZ) on Gyroscope in several variations. The prefix 't' in those metrics means time. The appendix 'XYZ' represents 3-axial signals on X, Y, and Z direction.

2.3 Feature names

1. These sensor signals are pre-processed using sound filters and sampled from fixed width windows (sliding windows) of 2.56 seconds each with a 50% interval i.e., each window has 128 readings.

2. In each window, a feature vector is obtained by calculating the variable from time to time and frequency domain. In our database, each data point represents a window with a different reading

3. The acceleration signal was subdivided into body and gravity acceleration signals (tBodyAcc-XYZ and tGravityAcc-XYZ) using a low pass filter with a corner frequency of 0.3Hz.

4. Subsequently, body line acceleration and angular velocity were obtained in time to detect jerk signals (tBodyAccJerk-XYZ and tBodyGyroJerk-XYZ).

5. The magnitude of these 3-dimensional signals is calculated using the Euclidian method. These magnitudes are represented as features names like tBodyAccMag, tGravityAccMag, tBodyAccJerkMag, tBodyGyroMag and tBodyGyroJerkMag.

6. Finally, we have received frequency domain signals from other available signals using FFT (Fast Fourier Transform). These received signals are labelled ‘f’ likewise original signals with prefix ‘t’. These signals are labelled as fBodyAcc-XYZ, fBodyGyroMag etc.

7. These are the signs that we have found so far.

- tBodyAcc-XYZ
- tGravityAcc-XYZ
- tBodyAccJerk-XYZ
- tBodyGyro-XYZ
- tBodyGyroJerk-XYZ
- tBodyAccMag
- tGravityAccMag
- tBodyAccJerkMag
- tBodyGyroMag
- tBodyGyroJerkMag
- fBodyAcc-XYZ
- fBodyAccJerk-XYZ
- fBodyGyro-XYZ
- fBodyAccMag
- fBodyAccJerkMag
- fBodyGyroMag
- fBodyGyroJerkMag

8. We can estimate a certain set of variables from the above signals. that is, We will rate the following features for each signal we have recorded so far.

- mean (): Meaning
- std (): Normal deviation
- mad (): Total Median deviation
- max (): Maximum value in system
- min (): The minimum value in the system
- sma (): Signal size area
- energy (): A measure of energy. The sum of the squares is divided by a number of numbers.
- iqr (): Interquartile range

- entropy (): Signal entropy
- arCoeff (): Automatic coefficients for Burg order equal to 4
- correlation (): coefficient of communication between two signals
- maxInds (): an indicator of a fraction of a fraction with the largest magnitude
- meanFreq (): Average number of fractions to find the average frequency
- skewness (): the skewness of the signal base domain
- kurtosis (): signature kurtosis of the frequency range
- Power belts (): Frequency intervals within 64 FFT barrels per window.
- angle (): The angle between the vectors.

9. We can detect other vectors by taking a signal measurement with a single window sample. These are applied to the angle() variable.

- gravity Mean
- tBodyAccMean
- tBodyAccJerkMean
- tBodyGyroMean
- tBodyGyroJerkMean

2.4 Y_Labels (Encoded)

• In the database, Y_ labels are shows as numbered from 1 to 6 as their identifiers.

- WALKING as 1
- WALKING_UPSTAIRS as 2
- WALKING_DOWNSTAIRS as 3
- SITTING as 4
- STANDING as 5
- LAYING as 6

2.5 Train and test data were separated

• The readings of 70% of volunteers was considered training data and the remaining 30% for test data.

2.6 Data

• All data is located in the 'UCI_HAR_dataset /' folder in the active directory.

o Feature names located in 'UCI_HAR_dataset / features.txt'

o Train Data

- 'UCI_HAR_dataset / train / X_train.txt'
- 'UCI_HAR_dataset / train / subject_train.txt'
- 'UCI_HAR_dataset / train / y_train.txt'

o Test Data

- 'UCI_HAR_dataset / test / X_test.txt'
- 'UCI_HAR_dataset / test / subject_test.txt'
- 'UCI_HAR_dataset / test / y_test.txt'

2.7 Glance of Dataset

• Accelerometer and Gyroscope studies are taken from 30 volunteers (called subjects) while performing the following 6 Tasks.

1. Walking

2. Walking Upstairs

3. Walking Downstairs

4. Standing

5. Sitting

6. Lying.

• The reading is divided into a 2.56 second window with 50% overlapped.

• Accelerometer readings are divided into gravity acceleration and body acceleration readings, consisting of parts x, y and z each.

• Gyroscope reading is a measures of angular velocity consisting of x, y and z segments.

• Jerk signals are calculated by Body Acceleration reading.

• Fourier Transforms are done in the time reading to get the reading of the frequencies.

• We find the vector element of 561 elements and these elements are given in the database.

• Each reading window is a data point for 561 features.

Problem Framework

• Data of 30 subjects (volunteers) randomly divided into 70% (21) test and 30% (7) train data.

• Each data point is linked to one of the 6 Tasks.

2.8 Problem Statement

• Given the new data point we have to predict the Activity

3. Methods:

3.1 Long Short-Term Memory (LSTM)

Long Short-Term Memory networks are a special kind of recurrent neural network. It takes care of long term dependencies as well as short term dependencies. It works tremendously well on large varieties of problems and it is widely used.

3.2 Need of LSTM

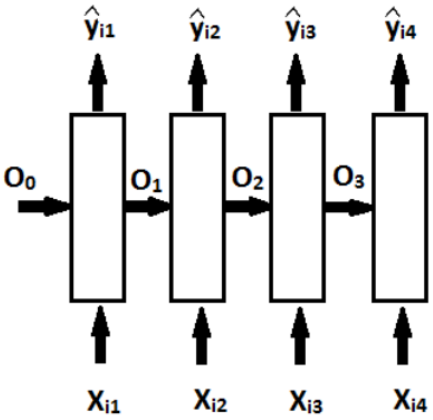

Fig 1. Many to many same lengths RNN

From above figure it is clear the Y_{i4} depends a lot on O_3 and X_{i4} and less depends on O_1 and X_{i1}. In real world problem if I requires reverse dependency i.e. Y_{i4} depends a lot on O_1 and X_{i1} and less depends on O_3 and X_{i4} then backpropogation will vanishes the gradients. Simple RNNs cannot take care of long term dependency. Thats'why we need different architecture which is LSTM.

3.3 LSTM Architecture

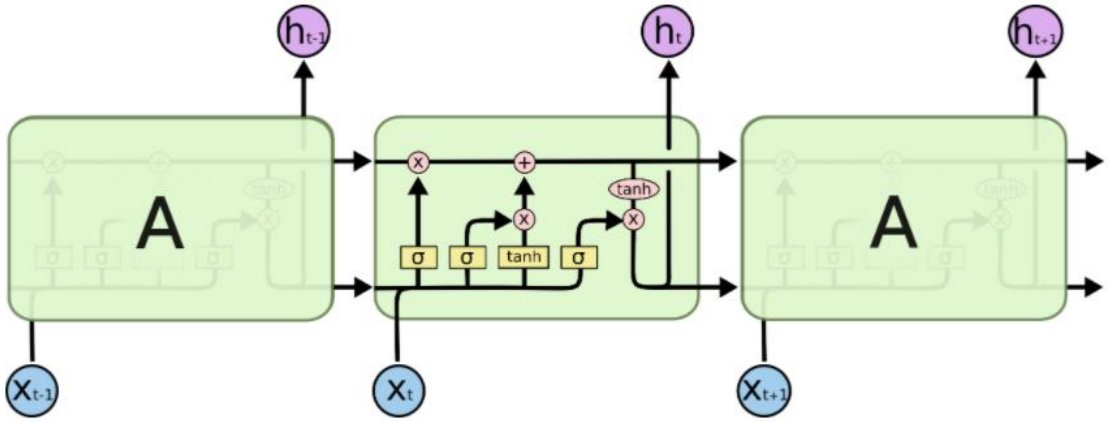

Fig 2. The repeating module in LSTM consists of four interactive layers.

All emerging neural networks include a series of repetitive network modules. LSTMs also have this repeating chain structure. But a repeating cell has a different structure compared to a simple RNN. Instead of a single neural layer there are four interacting pathways.

Each cell has following configuration:

- 3 inputs: previous cell state, input, output of previous cell
- 2 outputs: current output, current cell state
- Using identity we can pass same cell states
- Amount of previous cell state you want to pass can be manipulated

3.4 Step-by-Step LSTM Walk Through

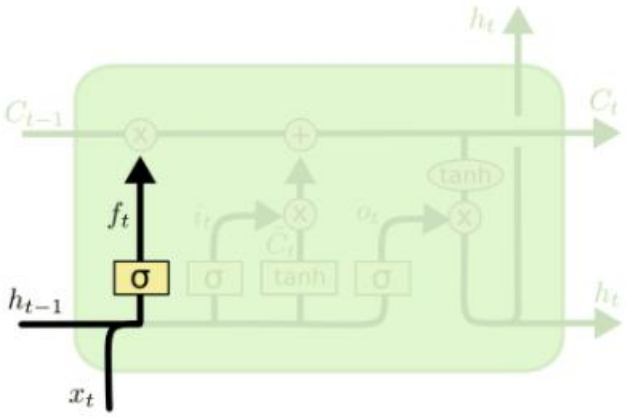

Fig 3. Forget gate

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

In first step, it takes two input h_{t-1} and x_t concatenate through weight W_f (forget weight) and adding bias and then passing through sigmoid function to get as f_t (Forget gate). This gate defines how much of your forget value from previous states before sending to the next state, hence it is called Input gate structure.

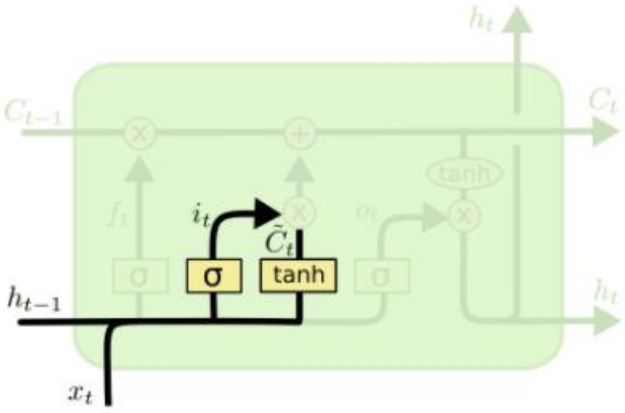

Fig 4. Input gate

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

In next step, it also take two input h_{t-1} and x_t concatenate through weight W_i(Input weight) and adding a bias and then passing through sigmoid function to get as it. Again two input

h_{t-1} and x_t concatenate through weight W_c and adding a bias and then passing through tanh function to get as $\tilde{C}_t$.

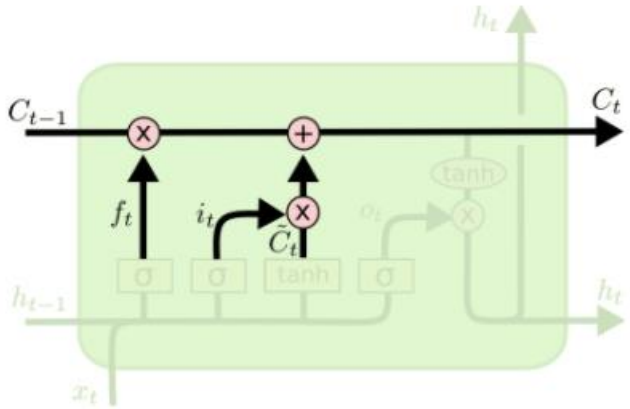

Fig 5. Input gate structure

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

In this step, the final Output C_t is a point wise multiplication of f_t and C_{t-1} and adding to the multiplication of it and $\tilde{C}_t$. Final out of this step defines how much should be add to the previous state before sending to the next state, hence it is called Input gate structure.

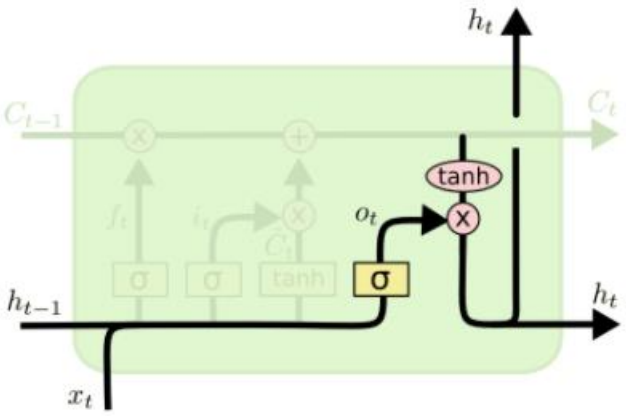

Fig 6. Output gate

$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

In next step, it also take two input h_{t-1} and x_t concatenate through weight W_o(output weight) and adding a bias and then passing through sigmoid function to get as O_t. Hence the Output depends on Input, previous output and current cell state which is illustrated in above equation.

3.5 Comparison of methods

Human Activity Recognition (HAR) using smartphone data and LSTM RNN. Divide the type of movement into six categories:

- WALKING,
- WALKING_UPSTAIRS,
- WALKING_DOWNSTAIRS,
- SITTING,
- STANDING,
- LAYING.

Compared to the classical machine learning algorithm method, using Recurrent Neural Networks (RNN) with long short-Term Memory (LSTMs) cells does not require or almost no feature engineering. The data can be entered directly into the neural network which acts as a black box, which models the problem correctly. Some research into data recognition functionality could use a large number of features engineering, which require digital signal processing expert for handcrafted feature extraction and it integrated with the old methods of data science. The method here is very simple depending on how well the data is processed.

By using Keras API, this demonstrates the use of LSTM, a type of Artificial Neural Network that can process sequential time series data.

4. Experiment:

We are training LSTM based deep learning model on raw time series data. We have six class classification tasks. Our time series data in Inertial signal, we have 9 time series data (3 accelerometer body, 3 accelerometer totals, 3 gyroscope body). For each time series data we broke everything into windows (overlapping window).After performing some signal processing each time series window in 128 dimensional vectors. At any time t we have 9 parallel recording of data. So, each window consists of 128 time steps and at each time steps we have 9 recording of vector data. By given a window data we are trying to predict what the class is.

Now we are building the model. We have six activities. Writing a code to print the confusion matrix. We define the signals and then we load both the data (train and test) from Inertial folder signal. Also loading all the class labels. We import the keras , also importing Sequential package, LSTM, Dense, and Dropout layer and then Initialising the parameter such as epochs, batch size and number of hidden layers. Also we are printing time steps, Input dimensions and total time series data. Now defining the architecture of LSTM model which consist of one LSTM layer, one dropout layer and dense layer. We have 7352 data points and having 5574 parameter means data points is large so it is very trivial to over fit the model, thats'why we added dropout layer.

We have use loss as categorical_crossentropy, optimizer as rmsprop and matrix as accuracy. Now we train the model for 40 epoch. At last we print confusion matrix, print accuracy, loss and plotting the graph for the same with respect to epoch.

5. Results and Discussion:

The accuracy and loss that we got after the 40 epoch are val_loss: 0.0910 val_accuracy: 0.9361. The Accuracy and loss for training and testing is give below.

```
460/460 [==============================] - 10s 22ms/step - loss: 0.1315 - accuracy: 0.9551 - val_loss: 0.0844 - val_accuracy: 0.9565
Epoch 25/40
460/460 [==============================] - 10s 22ms/step - loss: 0.1209 - accuracy: 0.9558 - val_loss: 0.1285 - val_accuracy: 0.9375
Epoch 26/40
460/460 [==============================] - 10s 22ms/step - loss: 0.1107 - accuracy: 0.9562 - val_loss: 0.0897 - val_accuracy: 0.9443
Epoch 27/40
460/460 [==============================] - 10s 22ms/step - loss: 0.1743 - accuracy: 0.9493 - val_loss: 0.1127 - val_accuracy: 0.9307
Epoch 28/40
460/460 [==============================] - 10s 22ms/step - loss: 0.1304 - accuracy: 0.9525 - val_loss: 0.1027 - val_accuracy: 0.9307
Epoch 29/40
460/460 [==============================] - 10s 22ms/step - loss: 0.1213 - accuracy: 0.9523 - val_loss: 0.1102 - val_accuracy: 0.9348
Epoch 30/40
460/460 [==============================] - 10s 22ms/step - loss: 0.1147 - accuracy: 0.9559 - val_loss: 0.2333 - val_accuracy: 0.9158
Epoch 31/40
460/460 [==============================] - 10s 22ms/step - loss: 0.1282 - accuracy: 0.9543 - val_loss: 0.2435 - val_accuracy: 0.9334
Epoch 32/40
460/460 [==============================] - 10s 22ms/step - loss: 0.1145 - accuracy: 0.9553 - val_loss: 0.1041 - val_accuracy: 0.9361
Epoch 33/40
460/460 [==============================] - 10s 22ms/step - loss: 0.1317 - accuracy: 0.9531 - val_loss: 0.1179 - val_accuracy: 0.9307
Epoch 34/40
460/460 [==============================] - 10s 22ms/step - loss: 0.1209 - accuracy: 0.9546 - val_loss: 0.1179 - val_accuracy: 0.9361
Epoch 35/40
460/460 [==============================] - 10s 22ms/step - loss: 0.1357 - accuracy: 0.9538 - val_loss: 0.1029 - val_accuracy: 0.9375
Epoch 36/40
460/460 [==============================] - 10s 22ms/step - loss: 0.1271 - accuracy: 0.9521 - val_loss: 0.0956 - val_accuracy: 0.9389
Epoch 37/40
460/460 [==============================] - 10s 22ms/step - loss: 0.1226 - accuracy: 0.9535 - val_loss: 0.1124 - val_accuracy: 0.9280
Epoch 38/40
460/460 [==============================] - 10s 23ms/step - loss: 0.1127 - accuracy: 0.9561 - val_loss: 0.0985 - val_accuracy: 0.9348
Epoch 39/40
460/460 [==============================] - 10s 22ms/step - loss: 0.1633 - accuracy: 0.9533 - val_loss: 0.1208 - val_accuracy: 0.9524
Epoch 40/40
460/460 [==============================] - 10s 22ms/step - loss: 0.1612 - accuracy: 0.9525 - val_loss: 0.0910 - val_accuracy: 0.9361
```

Fig 7. Loss and Accuracy of training and testing data

Then we have plotted Confusion matrix. In that matrix it is clear that Laying is classified very well but there is some confusion between “SITTING” and “STANDING”. The confusion matrix is given below.

```
[ ]  # Confusion Matrix Plotting
     print(confusion_matrix(Y_test, model.predict(X_test)))
```

Pred	LAYING	SITTING	...	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
True			...		
LAYING	494	0	...	0	43
SITTING	6	411	...	0	17
STANDING	0	92	...	0	5
WALKING	0	0	...	9	19
WALKING_DOWNSTAIRS	0	0	...	415	4
WALKING_UPSTAIRS	0	1	...	1	442

[6 rows x 6 columns]

Fig 8. Confusion Matrix

Also we have plotted Accuracy and loss with respect to epoch which is shown below. From that plot it is clear that our model is converging.

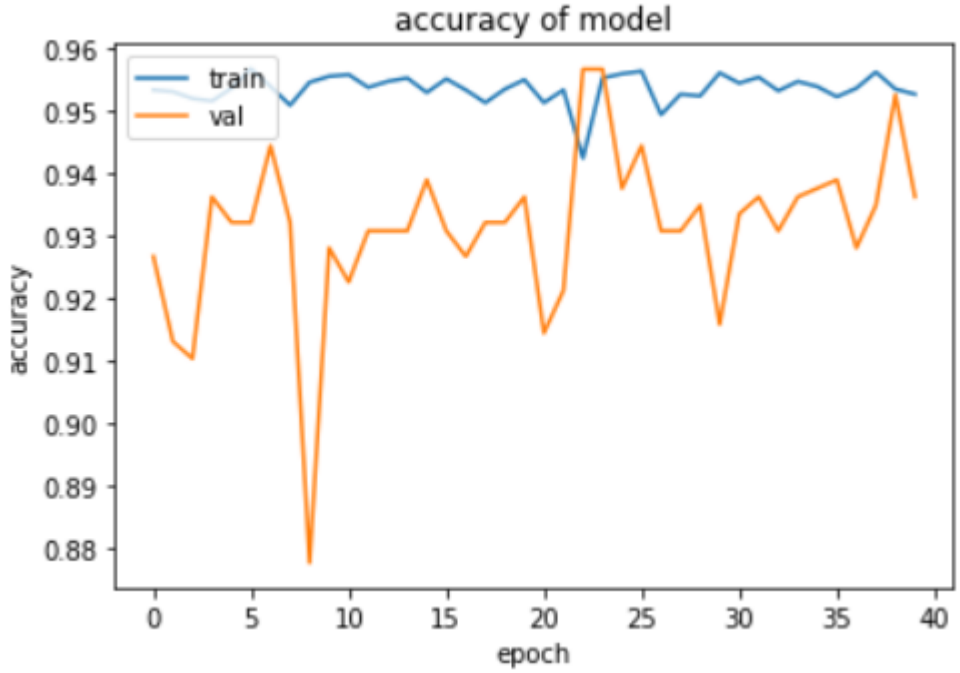

Fig 9. Accuracy plot of model

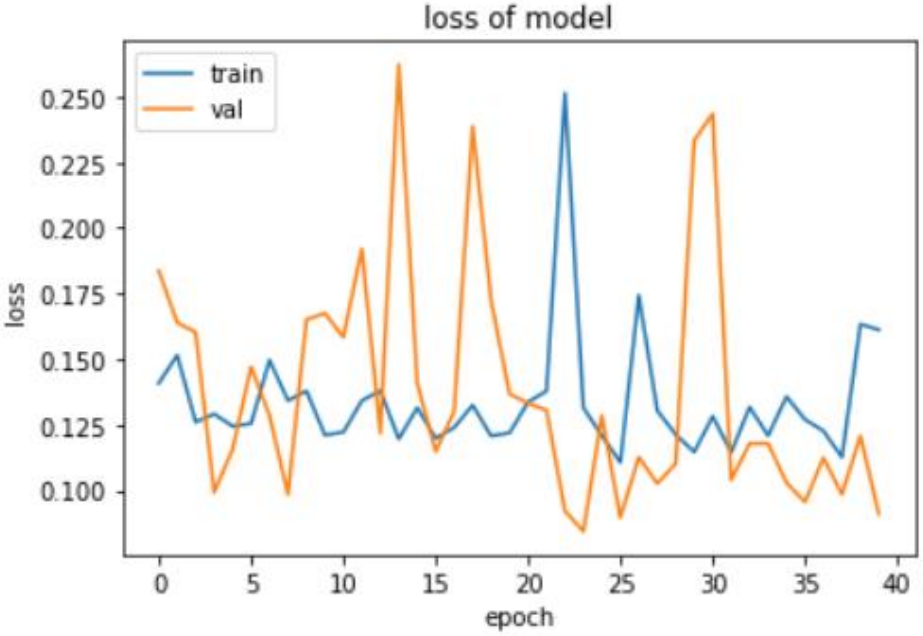

Fig 10. Loss plot of model

6. Conclusion:

In given problem, the importance of HAR research is analysed, and a summary of emerging trends in the LSTM model has tested. LSTM neural networks have been used for many innovations in natural language analysis, speech recognition, and weather forecasting, but in our problem we have tested it for time series data. This technology has been adapted to HAR functionality. We have developed a novel framework for the Deep-Res-LSTM network. This in-depth network can improve learning ability so it can learn quickly in early training.

Outstanding, the final accuracy is 88.32%. It can also reach above 90% if dataset is large or sometimes of luck during training, depending on how the neural network weights started at the beginning of the training, randomly.

We did not expect such good results from guessing between the labels "SITTING" and "STANDING". That seems to be almost identical by looking at a device placed at the waist depending on how the data was originally collected. When you think about it, it is still possible to see a small cluster in the confusion matrix between those classes, still getting a great result.

Future work should explore the most effective way of tuning parameters. Finally, using the Deep-Res-LSTM network in some fields may be noticeable. A good model should have an outstanding standard. Indeed, focusing on predictive timing problems is beneficial. Problems such as stock forecasting and trajectory prediction can be explored in future.

7. References:

[1] Yu Zhao, Rennong Yang, Guillaume Chevalier and Maoguo Gong, "**Deep Residual Bidir-LSTM for Human Activity Recognition Using Wearable Sensors**" in the journal of Mathematical Modelling in Engineering.
Volume 2018 |ArticleID 7316954 | https://doi.org/10.1155/2018/7316954

[2]https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones

[3] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[4] https://en.wikipedia.org/wiki/Long_short-term_memory#Applications

[5] https://github.com/guillaume-chevalier/LSTM-Human-Activity-Recognition