HUMAN ACTIVITY RECOGNITION USING LSTM MODEL

BY
TANISHKA NATH PASUMARTHI
PRANAVI NADIPALLI
ALISHA TAK
DENI SHAJI

OBJECTIVE

- To be able to detect **Human Activity using the Long-Short-Term Memory (LSTM)**, an artificial recurrent neural network architecture for the human activity recognition task.
- This recurrent neural network will learn complex features automatically from the raw accelerometer signal that will be able to differentiate between common human activities.
- The data is collected from 36 different users performing their day-to-day human activities. These activities include walking, sitting, standing, jogging, ascending and descending stairs for a specific period of time.

HUMAN ACTIVITY RECOGNITION USING LSTM MODEL

- Due to the rapid increase in the usage of smartphones equipped with sophisticated sensors such as accelerometers and gyroscopes. Accelerometers embedded in smartphones are used to detect the orientation of the phone. The gyroscope tracks the rotation or twist which adds an additional dimension to the information provided by the accelerometer.
- While the accelerometer measures the linear acceleration, the gyro on the other hand measures the rotational angular velocity. The information provided by these sensors supplements each other in the process of motion sensing.
- We employed Long short-term memory (LSTM), an artificial recurrent neural network architecture for the human activity recognition task, which will learn complex features automatically from the raw accelerometer signal to be able to differentiate between common human activities.

Data Description

- Smartphones contain tri-axial accelerometers that measure acceleration in all three spatial dimensions
- We used raw accelerometer signal data sourced from **WISDM Lab**, **Department of Computer & Information Science**, **Fordham University, NY**
- The data is collected from 36 different users performing their day-to-day human activities. These activities include walking, sitting, standing, jogging, ascending and descending stairs for a specific period of time.
- In all cases, data is collected at a **frequency of 20 samples per second**, that is one record every 50 milliseconds.
- The dataset has 6 columns – 'user', 'activity', 'timestamp', 'x-axis', 'y-axis', and 'z-axis'.
- Our target variable(class-label) is 'activity' which we intend to predict.

Data Cleaning & Preprocessing

1. Drop Null values.
2. Change datatype of the ‘z-axis’ column to float.
3. Drop the rows where the timestamp = 0.
4. Sort data in ascending order of ‘user’ and ‘timestamp’ columns.

	user	activity	timestamp	x-axis	y-axis	z-axis
0	1	Walking	4991922345000	0.69	10.80	-2.030000
1	1	Walking	4991972333000	6.85	7.44	-0.500000
2	1	Walking	4992022351000	0.93	5.63	-0.500000
3	1	Walking	4992072339000	-2.11	5.01	-0.690000
4	1	Walking	4992122358000	-4.59	4.29	-1.950000
...	...	...	...	...	...	...
1085355	36	Standing	15049012250000	-0.91	9.43	2.533385
1085356	36	Standing	15049062268000	-1.18	9.51	2.492524
1085357	36	Standing	15049112287000	-1.50	9.53	2.533385
1085358	36	Standing	15049162275000	-2.07	8.77	2.179256
1085359	36	Standing	15049212262000	-2.14	9.89	3.255263

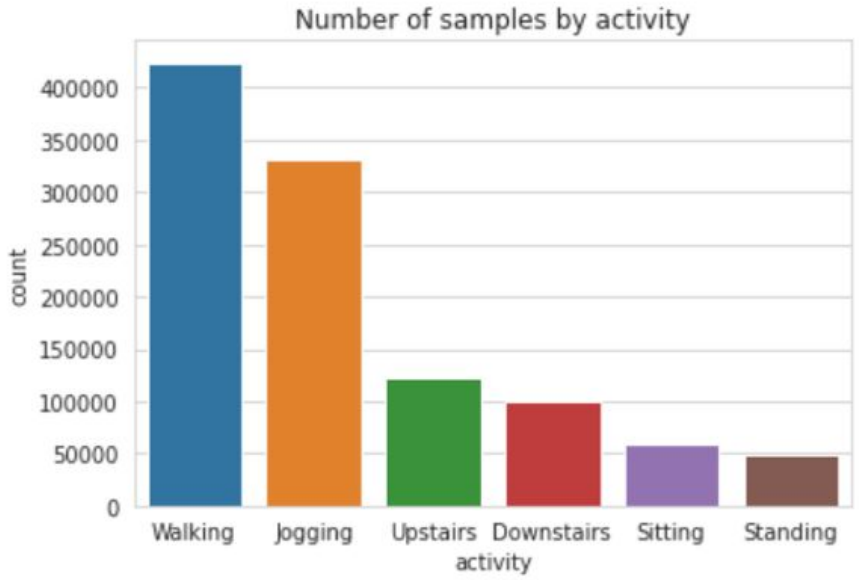
1085360 rows × 6 columns

➢ Now we are left with 1085360 rows.

EXPLORATORY DATA ANALYSIS

Analyzing class label distribution -

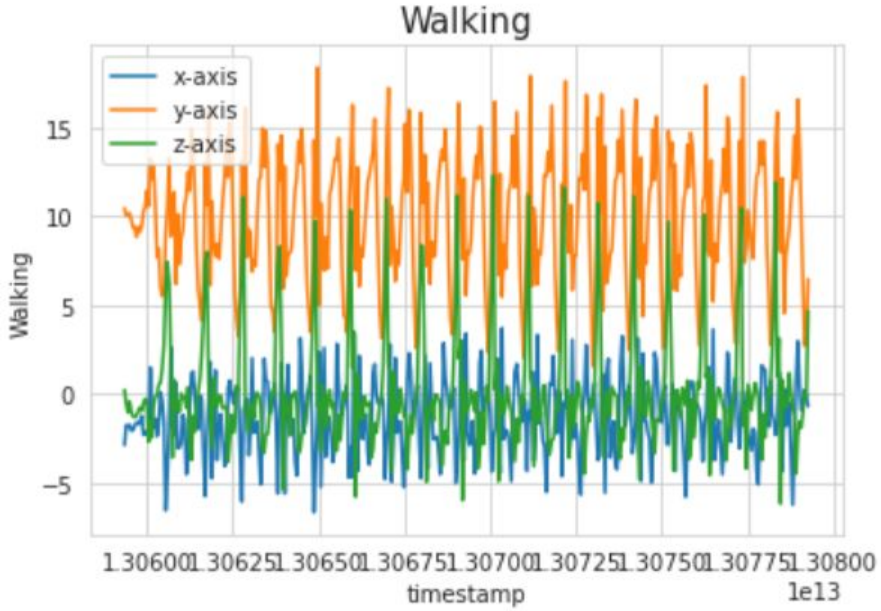
```
sns.set_style('whitegrid')
sns.countplot(x = "activity", data = df)
plt.title("Number of samples by activity")
plt.show()
```

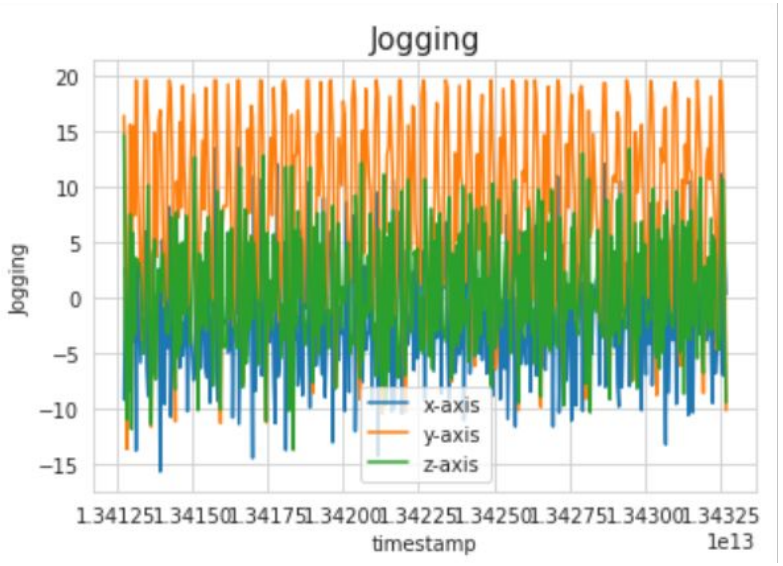


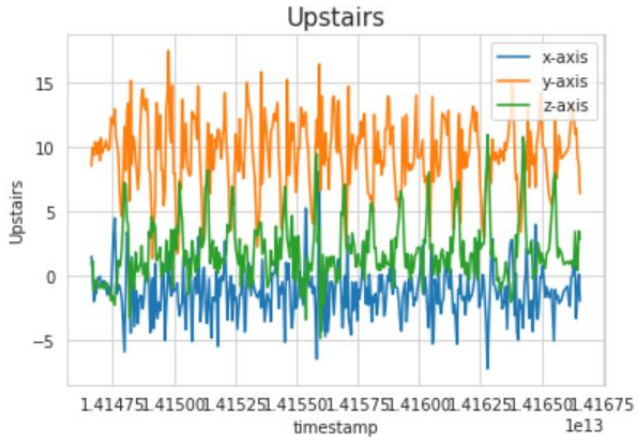
- It can be seen, there is a significant class imbalance here with the majority of the samples having class-label 'Walking' and 'Jogging'. 'Sitting' and 'Standing' activities have the least representation in the dataset.

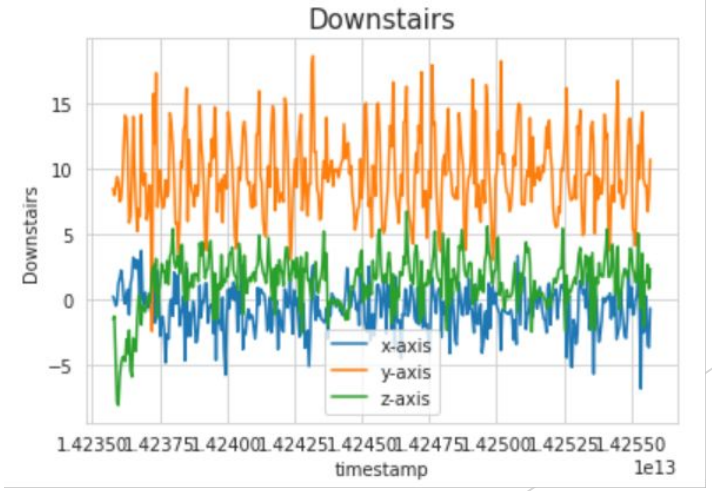
EXPLORATORY DATA ANALYSIS

- Visualizing how the signal values of each activity in the *x, y,* and *z* dimensions vary with time.

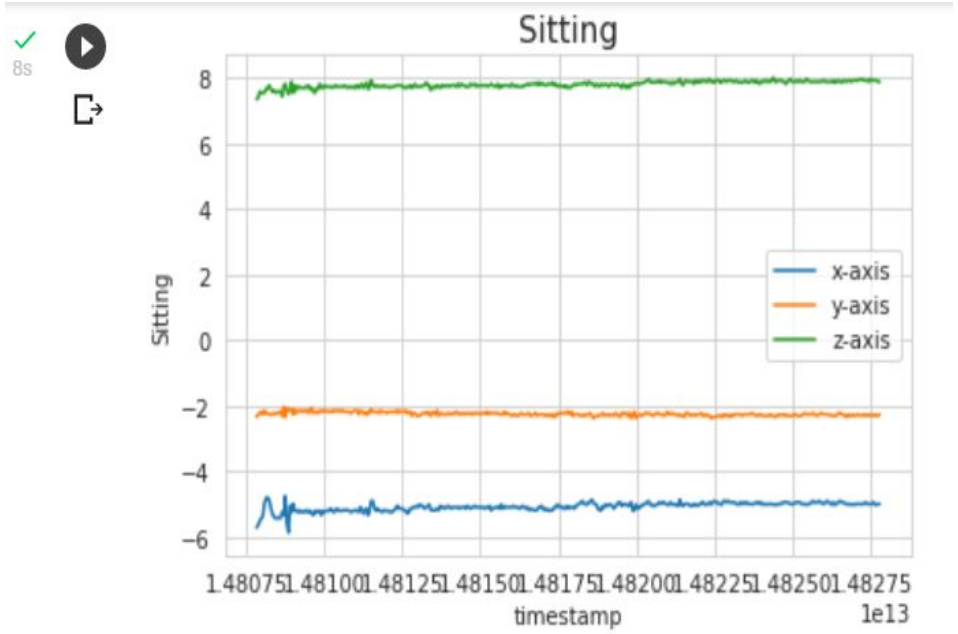


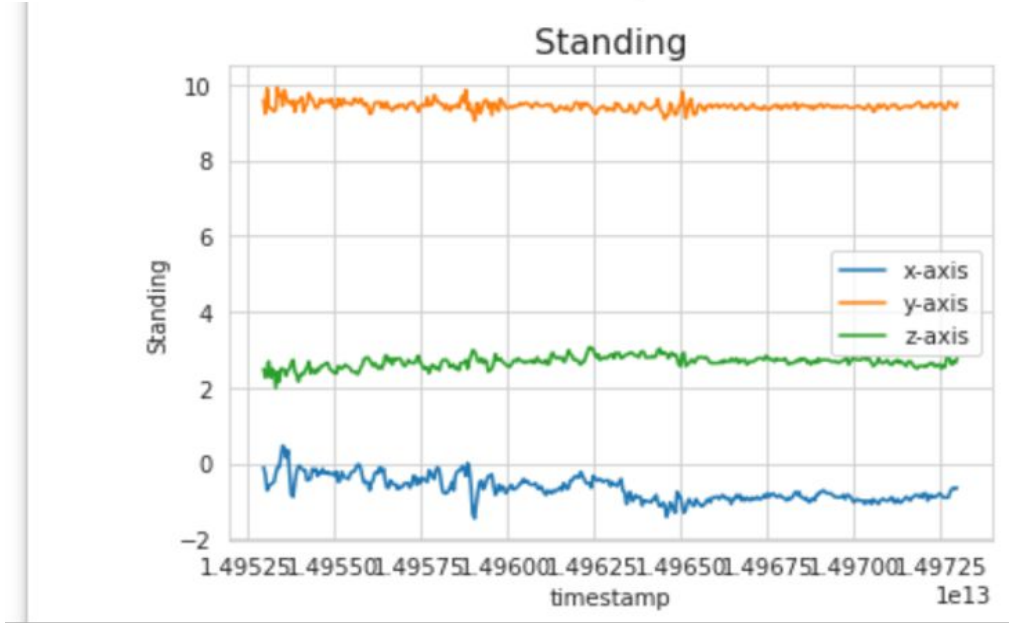






EXPLORATORY DATA ANALYSIS





- For visualization, we have considered a subset of 400 samples. That is equivalent to 20 secs of activity.

Here, we see the signal shows periodic behavior for activities like Walking, Jogging, Upstairs, and Downstairs while it has the very little movement for stationary activities like Sitting and Standing. These signals can be modeled as time-series data.

PREPARING DATA

```
[14] random_seed = 42
     n_time_steps = 50
     n_features = 3
     step = 10
     n_classes = 6
     n_epochs = 50
     batch_size = 1024
     learning_rate = 0.0025
     l2_loss = 0.0015
```

```
segments = []
labels = []

for i in range(0,  df.shape[0]- n_time_steps, step):

    xs = df["x-axis"].values[i: i + 50]

    ys = df["y-axis"].values[i: i + 50]

    zs = df["z-axis"].values[i: i + 50]

    label = stats.mode(df["activity"][i: i + 50])[0][0]

    segments.append([xs, ys, zs])

    labels.append(label)

#reshape the segments which is (list of arrays) to a list
reshaped_segments = np.asarray(segments, dtype= np.float32).reshape(-1, n_time_steps, n_features)

labels = np.asarray(pd.get_dummies(labels), dtype = np.float32)
```

- The LSTM model expects fixed-length sequences as training data.
- The code shows a method for generating, it will contain 50 records corresponding to 2.5 seconds of activity
- Here we have considered overlapping windows (with 80% overlap) of data. Because our activity is continuous, this overlap ensures that each subsequent window carries some information from the previous window.

PREPARING DATA

- If we check the shape of our transformed data

```
[16] reshaped_segments.shape
0s
     (108531, 50, 3)
```

Output: (108531, 50, 3)

- It gives 108531 sequences of 200 rows, each containing *x*, *y,* and *z* data. Our original raw dataset has drastically reduced size after the transformation. The class label assigned to a sequence (window) is the activity that occurs most frequently in that window.

- Splitting the dataset into training and testing sets:

```
[17] from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(reshaped_segments, labels, test_size = 0.2, random_state = random_seed)
```

Building Model Architecture

```
from keras.models import Sequential
from keras.layers import LSTM, Dense, Flatten, Dropout

model = Sequential()
# RNN layer
model.add(LSTM(units = 128, input_shape = (X_train.shape[1], X_train.shape[2])))
# Dropout layer
model.add(Dropout(0.5))
# Dense layer with ReLu
model.add(Dense(units = 64, activation='relu'))
# Softmax layer
model.add(Dense(y_train.shape[1], activation = 'softmax'))
# Compile model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

- This code defines a sequential model using Keras for a recurrent neural network (RNN) with LSTM units. It includes a dropout layer for regularization, a dense layer with ReLU activation, and a softmax layer for classification. The model is compiled with categorical cross-entropy loss and the Adam optimizer.

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 128)	67,584
dropout (Dropout)	(None, 128)	0
dense (Dense)	(None, 64)	8,256
dense_1 (Dense)	(None, 6)	390

Total params: 76,230 (297.77 KB)

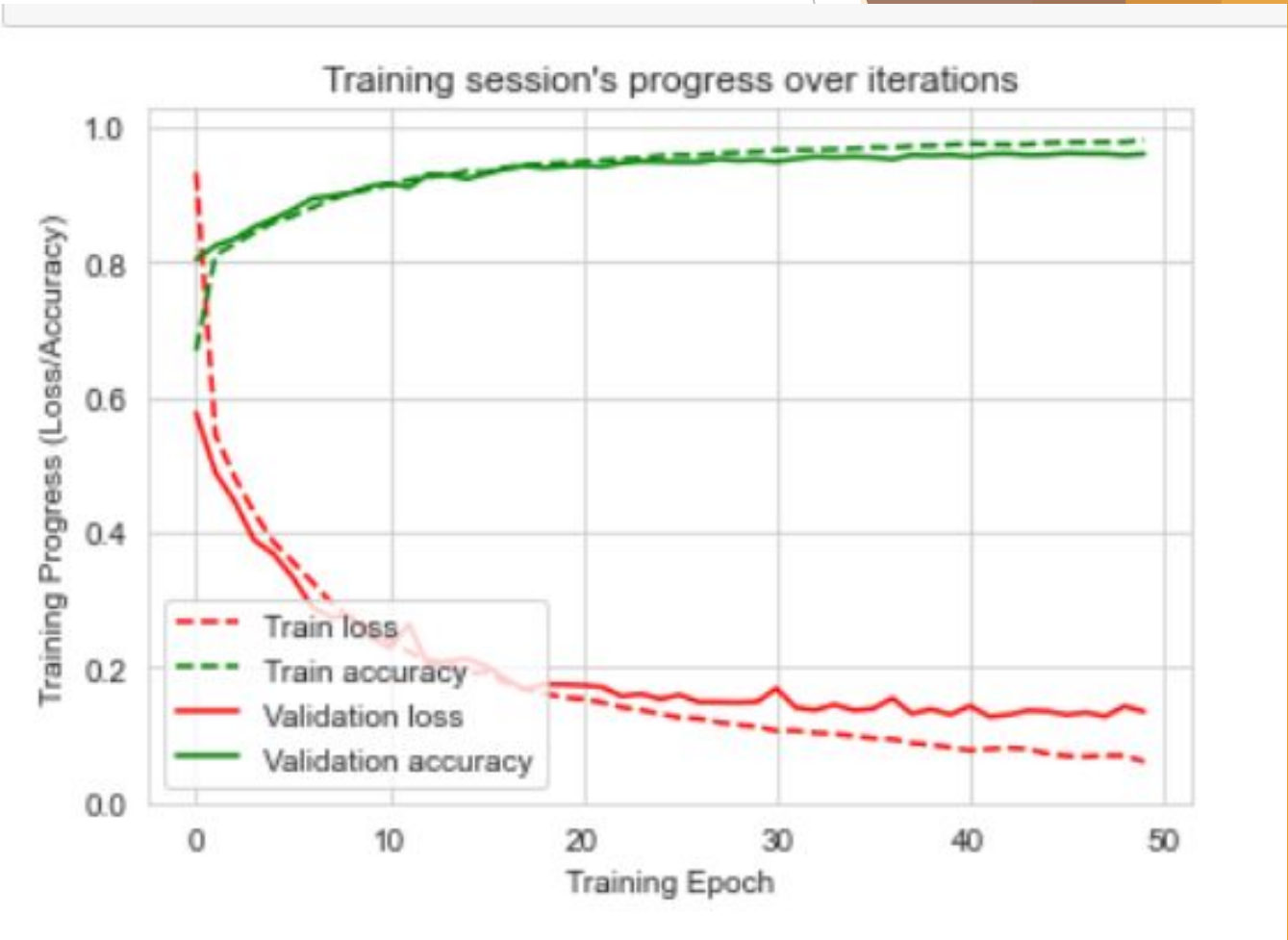
Trainable params: 76,230 (297.77 KB)

Non-trainable params: 0 (0.00 B)

- This is our model summary we have got more than 76k trainable parameters. Because of so many trainable parameters, the model tends to overfit easily. That is why a dropout layer is recommended to avoid overfitting.

Model Training & Evaluation

- We will train our model for 50 epochs and keep track of accuracy and error on the validation set.
- Plotting training and validation accuracy and cross-entropy loss against a number of epochs –



- Our LSTM model seems to learn well with accuracy reaching above 96% and cross-entropy loss is well below 0.2 for both training and validation data.
- Now let's assess the performance of the trained model on the test dataset –

```
22/22 ━━━━━━━━━━━━━━━━━━━━ 2s 84ms/step - accuracy: 0.9605 - loss: 0.1365
Test Accuracy : 0.9619938135147095
Test Loss : 0.13152720034122467
```

- Thus, we have got over 96% accuracy on the test data with cross-entropy loss equal to 0.13.
- Now let's observe the confusion matrix to see how our model performs on each class label.

Confusion Matrix

- As it can be noticed from the confusion matrix, the two most common activities in our dataset i.e. Walking and Jogging are correctly classified with very high accuracy. Although Sitting and Standing are minority classes, yet our model is accurately able to differentiate them.

- The accuracy is not as high as the other classes for Upstairs and Downstairs activities. This is expected as these two are very similar activities so the underlying data may not be sufficient to accurately different them.

CONCLUSION

- Human Activity Recognition can provide information about the identity of a person, their personality, and psychological state.
- Most everyday human tasks can be simplified or automated if they can be recognized through this activity recognizing system.
- Can be useful to those who are visually impaired or to those that require utmost security in specific places.
- LSTM model learns complex features automatically from the sequential data to be able to predict the class label with high accuracy.
- Human activity recognition will be detecting Human Activity without human supervision which will be significantly helpful.