In [1]:

```
# Importing Libraries
```

In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

```
# Activities are the class labels
# It is a 6 class classification
ACTIVITIES = {
    0: 'WALKING',
    1: 'WALKING_UPSTAIRS',
    2: 'WALKING_DOWNSTAIRS',
    3: 'SITTING',
    4: 'STANDING',
    5: 'LAYING',
}

# Utility function to print the confusion matrix
def confusion_matrix(Y_true, Y_pred):
    Y_true = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_true, axis=1)])
    Y_pred = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_pred, axis=1)])

    return pd.crosstab(Y_true, Y_pred, rownames=['True'], colnames=['Pred'])
```

Data

In [3]:

```
# Data directory
DATADIR = 'UCI_HAR_Dataset'
```

In [4]:

```
# Raw data signals
# Signals are from Accelerometer and Gyroscope
# The signals are in x,y,z directions
# Sensor signals are filtered to have only body acceleration
# excluding the acceleration due to gravity
# Triaxial acceleration from the accelerometer is total acceleration
SIGNALS = [
    "body_acc_x",
    "body_acc_y",
    "body_acc_z",
    "body_gyro_x",
    "body_gyro_y",
    "body_gyro_z",
    "total_acc_x",
    "total_acc_y",
    "total_acc_z"
]
```

In [5]:

```
# Utility function to read the data from csv file
def _read_csv(filename):
    return pd.read_csv(filename, delim_whitespace=True, header=None)

# Utility function to load the load
def load_signals(subset):
    signals_data = []

    for signal in SIGNALS:
        filename = f'UCI_HAR_Dataset/{subset}/Inertial Signals/{signal}_{subset}.txt'
        signals_data.append(
            _read_csv(filename).as_matrix()
        )

    # Transpose is used to change the dimensionality of the output,
    # aggregating the signals by combination of sample/timestep.
    # Resultant shape is (7352 train/2947 test samples, 128 timesteps, 9 signals)
    return np.transpose(signals_data, (1, 2, 0))
```

In [6]:

```
def load_y(subset):
    """
    The objective that we are trying to predict is a integer, from 1 to 6,
    that represents a human activity. We return a binary representation of
    every sample objective as a 6 bits vector using One Hot Encoding
    (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html)
    """
    filename = f'UCI_HAR_Dataset/{subset}/y_{subset}.txt'
    y = _read_csv(filename)[0]

    return pd.get_dummies(y).as_matrix()
```

In [7]:

```
def load_data():
    """
    Obtain the dataset from multiple files.
    Returns: X_train, X_test, y_train, y_test
    """
    X_train, X_test = load_signals('train'), load_signals('test')
    y_train, y_test = load_y('train'), load_y('test')

    return X_train, X_test, y_train, y_test
```

In [8]:

```
# Importing tensorflow
np.random.seed(42)
import tensorflow as tf
tf.random.set_seed(42)
```

In [9]:

```
# Configuring a session
session_conf = tf.compat.v1.ConfigProto(
    intra_op_parallelism_threads=1,
    inter_op_parallelism_threads=1
)
```

In [10]:

```
# Import Keras
from keras import backend as K
sess = tf.compat.v1.Session(graph= tf.compat.v1.get_default_graph(), config=session_con
f)
tf.compat.v1.keras.backend.set_session(sess)
```

```
Using TensorFlow backend.
```

In [11]:

```
# Importing libraries
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers.core import Dense, Dropout
```

In [12]:

```
# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 32
```

In [13]:

```
# Utility function to count the number of classes
def _count_classes(y):
    return len(set([tuple(category) for category in y]))
```

In [14]:

```
# Loading the train and test data
X_train, X_test, Y_train, Y_test = load_data()
```

```
C:\Anaconda3\lib\site-packages\ipykernel_launcher.py:12: FutureWarning: Me
thod .as_matrix will be removed in a future version. Use .values instead.
  if sys.path[0] == '':
C:\Anaconda3\lib\site-packages\ipykernel_launcher.py:11: FutureWarning: Me
thod .as_matrix will be removed in a future version. Use .values instead.
  # This is added back by InteractiveShellApp.init_path()
```

In [15]:

```
timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = _count_classes(Y_train)

print(timesteps)
print(input_dim)
print(len(X_train))
```

```
128
9
7352
```

- Defining the Architecture of LSTM

In [24]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 32)                5376
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 198
=================================================================
Total params: 5,574
Trainable params: 5,574
Non-trainable params: 0
_________________________________________________________________

Exception ignored in: <bound method BaseSession.__del__ of <tensorflow.pyt
hon.client.session.Session object at 0x000001FA6354C630>>
Traceback (most recent call last):
  File "C:\Anaconda3\lib\site-packages\tensorflow_core\python\client\sessi
on.py", line 761, in __del__
    if self._session is not None:
AttributeError: 'Session' object has no attribute '_session'
```

In [25]:

```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [26]:

```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 64s 9ms/step - loss: 1.2773 -
accuracy: 0.4693 - val_loss: 1.0427 - val_accuracy: 0.5741
Epoch 2/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.9190 -
accuracy: 0.6035 - val_loss: 0.9349 - val_accuracy: 0.5677
Epoch 3/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.7878 -
accuracy: 0.6390 - val_loss: 0.7974 - val_accuracy: 0.6098
Epoch 4/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.7117 -
accuracy: 0.6517 - val_loss: 0.7804 - val_accuracy: 0.6132
Epoch 5/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.6653 -
accuracy: 0.6598 - val_loss: 0.7542 - val_accuracy: 0.6230
Epoch 6/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.6326 -
accuracy: 0.6632 - val_loss: 0.7230 - val_accuracy: 0.6183
Epoch 7/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.5961 -
accuracy: 0.6810 - val_loss: 0.9588 - val_accuracy: 0.5918
Epoch 8/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.5895 -
accuracy: 0.6968 - val_loss: 0.7776 - val_accuracy: 0.6322
Epoch 9/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.5400 -
accuracy: 0.7269 - val_loss: 0.6660 - val_accuracy: 0.7394
Epoch 10/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.5245 -
accuracy: 0.7576 - val_loss: 0.6615 - val_accuracy: 0.7384
Epoch 11/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.4953 -
accuracy: 0.7801 - val_loss: 0.5606 - val_accuracy: 0.7621
Epoch 12/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.5806 -
accuracy: 0.7232 - val_loss: 0.6552 - val_accuracy: 0.7201
Epoch 13/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.4487 -
accuracy: 0.7843 - val_loss: 0.5487 - val_accuracy: 0.7594
Epoch 14/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.4046 -
accuracy: 0.8067 - val_loss: 0.5499 - val_accuracy: 0.7774
Epoch 15/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.4314 -
accuracy: 0.7935 - val_loss: 0.6065 - val_accuracy: 0.7516
Epoch 16/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.4058 -
accuracy: 0.8036 - val_loss: 0.5485 - val_accuracy: 0.7638
Epoch 17/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.4001 -
accuracy: 0.8028 - val_loss: 0.5619 - val_accuracy: 0.7655
Epoch 18/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.3802 -
accuracy: 0.8165 - val_loss: 0.5104 - val_accuracy: 0.7828
Epoch 19/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.3702 -
accuracy: 0.8210 - val_loss: 0.5687 - val_accuracy: 0.7781
Epoch 20/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.3626 -
accuracy: 0.8327 - val_loss: 0.5845 - val_accuracy: 0.7995
```

```
Epoch 21/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.3460 -
accuracy: 0.8471 - val_loss: 0.4911 - val_accuracy: 0.8283
Epoch 22/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.3216 -
accuracy: 0.8908 - val_loss: 0.4370 - val_accuracy: 0.8724
Epoch 23/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.2585 -
accuracy: 0.9210 - val_loss: 0.5111 - val_accuracy: 0.8782
Epoch 24/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.2526 -
accuracy: 0.9297 - val_loss: 0.4791 - val_accuracy: 0.8914
Epoch 25/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.2659 -
accuracy: 0.9234 - val_loss: 0.4733 - val_accuracy: 0.8806
Epoch 26/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.2114 -
accuracy: 0.9397 - val_loss: 0.6636 - val_accuracy: 0.8812
Epoch 27/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.1921 -
accuracy: 0.9419 - val_loss: 0.5583 - val_accuracy: 0.8860
Epoch 28/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.2001 -
accuracy: 0.9416 - val_loss: 0.4912 - val_accuracy: 0.8979
Epoch 29/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.1836 -
accuracy: 0.9406 - val_loss: 0.5204 - val_accuracy: 0.9030
Epoch 30/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.1906 -
accuracy: 0.9416 - val_loss: 0.4287 - val_accuracy: 0.9063
```

Out[26]:

```
<keras.callbacks.callbacks.History at 0x1fa0dccefd0>
```

In [27]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS
\
True
LAYING                 510        0        10        0                   0
SITTING                  3      378       110        0                   0
STANDING                 0       69       462        1                   0
WALKING                  0        0         2      456                  25
WALKING_DOWNSTAIRS       0        0         0        0                 416
WALKING_UPSTAIRS         0        0         2        1                  19

Pred                WALKING_UPSTAIRS
True
LAYING                            17
SITTING                            0
STANDING                           0
WALKING                           13
WALKING_DOWNSTAIRS                 4
WALKING_UPSTAIRS                 449
```

In [28]:

```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 6s 2ms/step
```

In [29]:

```
score
```

Out[29]:

```
[0.4293035353590176, 0.9063454270362854]
```

- With a simple 2 layer architecture we got 90.09% accuracy and a loss of 0.30
- We can further imporve the performace with Hyperparameter tuning

ASSIGNMENT

In [43]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(32, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.25))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_9 (LSTM)                (None, 32)                5376
_________________________________________________________________
dropout_9 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_9 (Dense)              (None, 6)                 198
=================================================================
Total params: 5,574
Trainable params: 5,574
Non-trainable params: 0
_________________________________________________________________
```

In [44]:

```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

In [45]:

```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 63s 9ms/step - loss: 1.3283 -
accuracy: 0.4455 - val_loss: 1.2288 - val_accuracy: 0.4455
Epoch 2/30
7352/7352 [==============================] - 63s 9ms/step - loss: 1.1613 -
accuracy: 0.4951 - val_loss: 1.0449 - val_accuracy: 0.5718
Epoch 3/30
7352/7352 [==============================] - 63s 9ms/step - loss: 1.0198 -
accuracy: 0.5180 - val_loss: 1.1316 - val_accuracy: 0.5765
Epoch 4/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.8286 -
accuracy: 0.6193 - val_loss: 0.7495 - val_accuracy: 0.6278
Epoch 5/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.7025 -
accuracy: 0.6598 - val_loss: 0.7118 - val_accuracy: 0.6322
Epoch 6/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.6752 -
accuracy: 0.6609 - val_loss: 0.7598 - val_accuracy: 0.6328
Epoch 7/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.6403 -
accuracy: 0.6832 - val_loss: 0.7718 - val_accuracy: 0.6508
Epoch 8/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.6045 -
accuracy: 0.7225 - val_loss: 0.6928 - val_accuracy: 0.7078
Epoch 9/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.5704 -
accuracy: 0.7603 - val_loss: 0.8659 - val_accuracy: 0.6206
Epoch 10/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.7084 -
accuracy: 0.6989 - val_loss: 0.6693 - val_accuracy: 0.7370
Epoch 11/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.4659 -
accuracy: 0.8232 - val_loss: 0.6088 - val_accuracy: 0.7764
Epoch 12/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.3616 -
accuracy: 0.8857 - val_loss: 0.4857 - val_accuracy: 0.8470
Epoch 13/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.3145 -
accuracy: 0.8998 - val_loss: 0.4476 - val_accuracy: 0.8649
Epoch 14/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.2995 -
accuracy: 0.9094 - val_loss: 1.9471 - val_accuracy: 0.5277
Epoch 15/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.3885 -
accuracy: 0.8649 - val_loss: 0.5509 - val_accuracy: 0.7811
Epoch 16/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.2561 -
accuracy: 0.9135 - val_loss: 0.4959 - val_accuracy: 0.8636
Epoch 17/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.2663 -
accuracy: 0.9132 - val_loss: 0.4422 - val_accuracy: 0.8687
Epoch 18/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.2211 -
accuracy: 0.9242 - val_loss: 0.4575 - val_accuracy: 0.8653
Epoch 19/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.2396 -
accuracy: 0.9236 - val_loss: 0.4649 - val_accuracy: 0.8507
Epoch 20/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.2240 -
accuracy: 0.9298 - val_loss: 0.3755 - val_accuracy: 0.8734
```

```
Epoch 21/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.1750 -
accuracy: 0.9414 - val_loss: 0.4254 - val_accuracy: 0.8616
Epoch 22/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.1979 -
accuracy: 0.9363 - val_loss: 0.3879 - val_accuracy: 0.8785
Epoch 23/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.1677 -
accuracy: 0.9434 - val_loss: 0.4225 - val_accuracy: 0.8694
Epoch 24/30
7352/7352 [==============================] - 62s 8ms/step - loss: 0.1746 -
accuracy: 0.9412 - val_loss: 0.3846 - val_accuracy: 0.8829
Epoch 25/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.1604 -
accuracy: 0.9425 - val_loss: 0.3634 - val_accuracy: 0.8921
Epoch 26/30
7352/7352 [==============================] - 62s 9ms/step - loss: 0.1562 -
accuracy: 0.9456 - val_loss: 0.3725 - val_accuracy: 0.8761
Epoch 27/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.1555 -
accuracy: 0.9442 - val_loss: 0.3829 - val_accuracy: 0.8958
Epoch 28/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.1452 -
accuracy: 0.9465 - val_loss: 0.3644 - val_accuracy: 0.8731
Epoch 29/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.1891 -
accuracy: 0.9370 - val_loss: 0.3673 - val_accuracy: 0.8748
Epoch 30/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.1474 -
accuracy: 0.9452 - val_loss: 0.3594 - val_accuracy: 0.8948
```

Out[45]:

```
<keras.callbacks.callbacks.History at 0x1fbf4b4bd68>
```

In [46]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS
\
True
LAYING                 537        0         0        0                   0
SITTING                  0      379        88        0                   0
STANDING                 0       83       445        2                   0
WALKING                  0        0         0      454                  37
WALKING_DOWNSTAIRS       0        0         0        2                 409
WALKING_UPSTAIRS         0        1         1       31                  25

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                           24
STANDING                           2
WALKING                            5
WALKING_DOWNSTAIRS                 9
WALKING_UPSTAIRS                 413
```

In [47]:

```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 6s 2ms/step
```

In [48]:

```
score
```

Out[48]:

```
[0.3594353889522391, 0.894808292388916]
```

In [68]:

```
len(X_train[1])
```

Out[68]:

```
128
```

In [84]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(32,return_sequences = True,input_shape=(128,9)))
# Adding a dropout layer
model.add(Dropout(0.75))
model.add(LSTM(32,return_sequences = False,input_shape=(128,9)))
# Adding a dropout layer
model.add(Dropout(0.75))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_30"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_48 (LSTM)               (None, 128, 32)           5376
_________________________________________________________________
dropout_32 (Dropout)         (None, 128, 32)           0
_________________________________________________________________
lstm_49 (LSTM)               (None, 32)                8320
_________________________________________________________________
dropout_33 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_15 (Dense)             (None, 6)                 198
=================================================================
Total params: 13,894
Trainable params: 13,894
Non-trainable params: 0
_________________________________________________________________
```

In [85]:

```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

In [86]:

```
from keras.callbacks.callbacks import EarlyStopping
early_stop=EarlyStopping(monitor='accuracy',patience=2)
callbacks = [early_stop]
```

In [87]:

```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs,callbacks = callbacks)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 126s 17ms/step - loss: 1.4905
- accuracy: 0.3984 - val_loss: 1.3451 - val_accuracy: 0.3482
Epoch 2/30
7352/7352 [==============================] - 126s 17ms/step - loss: 1.1664
- accuracy: 0.4830 - val_loss: 0.9621 - val_accuracy: 0.5755
Epoch 3/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.9819
- accuracy: 0.5486 - val_loss: 0.8352 - val_accuracy: 0.6899
Epoch 4/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.8936
- accuracy: 0.5711 - val_loss: 0.7979 - val_accuracy: 0.6447
Epoch 5/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.9400
- accuracy: 0.5458 - val_loss: 0.8455 - val_accuracy: 0.5463
Epoch 6/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.8755
- accuracy: 0.5720 - val_loss: 0.7854 - val_accuracy: 0.5840
Epoch 7/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.8213
- accuracy: 0.5839 - val_loss: 0.7484 - val_accuracy: 0.5799
Epoch 8/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.7985
- accuracy: 0.5903 - val_loss: 0.7399 - val_accuracy: 0.5836
Epoch 9/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.7848
- accuracy: 0.6046 - val_loss: 0.7084 - val_accuracy: 0.6149
Epoch 10/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.7558
- accuracy: 0.6066 - val_loss: 0.6989 - val_accuracy: 0.6152
Epoch 11/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.7466
- accuracy: 0.6185 - val_loss: 0.6997 - val_accuracy: 0.6121
Epoch 12/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.7356
- accuracy: 0.6277 - val_loss: 0.7710 - val_accuracy: 0.6138
Epoch 13/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.7478
- accuracy: 0.6289 - val_loss: 1.0559 - val_accuracy: 0.4788
Epoch 14/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.8455
- accuracy: 0.5977 - val_loss: 0.7464 - val_accuracy: 0.6183
Epoch 15/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.7380
- accuracy: 0.6389 - val_loss: 0.7948 - val_accuracy: 0.6138
Epoch 16/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.7080
- accuracy: 0.6425 - val_loss: 0.8116 - val_accuracy: 0.6142
Epoch 17/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.7572
- accuracy: 0.6303 - val_loss: 0.8191 - val_accuracy: 0.6233
Epoch 18/30
7352/7352 [==============================] - 127s 17ms/step - loss: 0.7079
- accuracy: 0.6458 - val_loss: 0.7964 - val_accuracy: 0.6176
Epoch 19/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.6961
- accuracy: 0.6465 - val_loss: 0.7906 - val_accuracy: 0.6203
Epoch 20/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.7016
- accuracy: 0.6459 - val_loss: 0.8040 - val_accuracy: 0.6237
```

```
Epoch 21/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.6815
- accuracy: 0.6563 - val_loss: 0.7937 - val_accuracy: 0.6162
Epoch 22/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.6871
- accuracy: 0.6547 - val_loss: 0.7979 - val_accuracy: 0.6244
Epoch 23/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.6910
- accuracy: 0.6557 - val_loss: 0.7772 - val_accuracy: 0.6244
```

Out[87]:

```
<keras.callbacks.callbacks.History at 0x1fc08976ef0>
```

In [91]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(20,return_sequences = True,input_shape=(128,9)))
# Adding a dropout layer
model.add(Dropout(0.5))
model.add(LSTM(20,return_sequences = False,input_shape=(128,9)))
# Adding a dropout layer
model.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_32"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_52 (LSTM)               (None, 128, 20)           2400
_________________________________________________________________
dropout_36 (Dropout)         (None, 128, 20)           0
_________________________________________________________________
lstm_53 (LSTM)               (None, 20)                3280
_________________________________________________________________
dropout_37 (Dropout)         (None, 20)                0
_________________________________________________________________
dense_17 (Dense)             (None, 6)                 126
=================================================================
Total params: 5,806
Trainable params: 5,806
Non-trainable params: 0
_________________________________________________________________
```

In [92]:

```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [93]:

```
from keras.callbacks.callbacks import EarlyStopping
early_stop=EarlyStopping(monitor='accuracy',patience=2)
callbacks = [early_stop]
```

In [94]:

```
#Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs,callbacks = callbacks)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 122s 17ms/step - loss: 1.2968
- accuracy: 0.5180 - val_loss: 0.9934 - val_accuracy: 0.6295
Epoch 2/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.9169
- accuracy: 0.6438 - val_loss: 0.7864 - val_accuracy: 0.7282
Epoch 3/30
7352/7352 [==============================] - 121s 16ms/step - loss: 0.7902
- accuracy: 0.7095 - val_loss: 0.6773 - val_accuracy: 0.7357
Epoch 4/30
7352/7352 [==============================] - 121s 16ms/step - loss: 0.7136
- accuracy: 0.7330 - val_loss: 0.6054 - val_accuracy: 0.7849
Epoch 5/30
7352/7352 [==============================] - 123s 17ms/step - loss: 0.6720
- accuracy: 0.7591 - val_loss: 0.6572 - val_accuracy: 0.7645
Epoch 6/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.5740
- accuracy: 0.8332 - val_loss: 0.4878 - val_accuracy: 0.8449
Epoch 7/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.4653
- accuracy: 0.8735 - val_loss: 0.4785 - val_accuracy: 0.8558
Epoch 8/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.3891
- accuracy: 0.8996 - val_loss: 0.4222 - val_accuracy: 0.8819
Epoch 9/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.3449
- accuracy: 0.9100 - val_loss: 0.3793 - val_accuracy: 0.8928
Epoch 10/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.3270
- accuracy: 0.9135 - val_loss: 0.4120 - val_accuracy: 0.8816
Epoch 11/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.3265
- accuracy: 0.9176 - val_loss: 0.4085 - val_accuracy: 0.8989
Epoch 12/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.2833
- accuracy: 0.9227 - val_loss: 0.4937 - val_accuracy: 0.8812
Epoch 13/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.2583
- accuracy: 0.9289 - val_loss: 0.5589 - val_accuracy: 0.8734
Epoch 14/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.2394
- accuracy: 0.9325 - val_loss: 0.4196 - val_accuracy: 0.8826
Epoch 15/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.2395
- accuracy: 0.9317 - val_loss: 0.4278 - val_accuracy: 0.8870
Epoch 16/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.2187
- accuracy: 0.9339 - val_loss: 0.5180 - val_accuracy: 0.8819
Epoch 17/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.2193
- accuracy: 0.9324 - val_loss: 0.4594 - val_accuracy: 0.8873
Epoch 18/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.2146
- accuracy: 0.9397 - val_loss: 0.3987 - val_accuracy: 0.9002
Epoch 19/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.2306
- accuracy: 0.9324 - val_loss: 0.4380 - val_accuracy: 0.9030
Epoch 20/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.2062
- accuracy: 0.9366 - val_loss: 0.4683 - val_accuracy: 0.9040
```

Out[94]:

```
<keras.callbacks.callbacks.History at 0x1fc1a694f28>
```

In [95]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAI
RS  \
True
LAYING                 537        0         0        0
0
SITTING                  0      407        62       22
0
STANDING                 1       96       432        3
0
WALKING                  0        0         1      453
37
WALKING_DOWNSTAIRS       0        0         0       11                 4
08
WALKING_UPSTAIRS         0        0         0       16
28

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            0
STANDING                           0
WALKING                            5
WALKING_DOWNSTAIRS                 1
WALKING_UPSTAIRS                 427
```

In [96]:

```
score = model.evaluate(X_test, Y_test)
score
```

```
2947/2947 [==============================] - 11s 4ms/step
```

Out[96]:

```
[0.4682972089535494, 0.9039701223373413]
```

In [97]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(24,return_sequences = True,input_shape=(128,9)))
# Adding a dropout layer
model.add(Dropout(0.5))
model.add(LSTM(24,return_sequences = False,input_shape=(128,9)))
# Adding a dropout layer
model.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_33"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_54 (LSTM)               (None, 128, 24)           3264
_________________________________________________________________
dropout_38 (Dropout)         (None, 128, 24)           0
_________________________________________________________________
lstm_55 (LSTM)               (None, 24)                4704
_________________________________________________________________
dropout_39 (Dropout)         (None, 24)                0
_________________________________________________________________
dense_18 (Dense)             (None, 6)                 150
=================================================================
Total params: 8,118
Trainable params: 8,118
Non-trainable params: 0
_________________________________________________________________
```

In [98]:

```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [99]:

```
from keras.callbacks.callbacks import EarlyStopping
early_stop=EarlyStopping(monitor='val_accuracy',patience=2)
callbacks = [early_stop]
```

In [100]:

```
#Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs,callbacks = callbacks)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 126s 17ms/step - loss: 1.2037
- accuracy: 0.5173 - val_loss: 0.9069 - val_accuracy: 0.5718
Epoch 2/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.8513
- accuracy: 0.6268 - val_loss: 0.7921 - val_accuracy: 0.6003
Epoch 3/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.7388
- accuracy: 0.6911 - val_loss: 0.7528 - val_accuracy: 0.6851
Epoch 4/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.6451
- accuracy: 0.7440 - val_loss: 0.6266 - val_accuracy: 0.7333
Epoch 5/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.5591
- accuracy: 0.7775 - val_loss: 0.6781 - val_accuracy: 0.7214
Epoch 6/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.4845
- accuracy: 0.8033 - val_loss: 0.5954 - val_accuracy: 0.7631
Epoch 7/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.4270
- accuracy: 0.8478 - val_loss: 0.4335 - val_accuracy: 0.8612
Epoch 8/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.3299
- accuracy: 0.9104 - val_loss: 0.3942 - val_accuracy: 0.8768
Epoch 9/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.2954
- accuracy: 0.9192 - val_loss: 0.4490 - val_accuracy: 0.8734
Epoch 10/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.2617
- accuracy: 0.9289 - val_loss: 0.3657 - val_accuracy: 0.8951
Epoch 11/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.2330
- accuracy: 0.9344 - val_loss: 0.3991 - val_accuracy: 0.8951
Epoch 12/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.2185
- accuracy: 0.9374 - val_loss: 0.3687 - val_accuracy: 0.8989
Epoch 13/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.2235
- accuracy: 0.9343 - val_loss: 0.4335 - val_accuracy: 0.9006
Epoch 14/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.2105
- accuracy: 0.9351 - val_loss: 0.3721 - val_accuracy: 0.8938
Epoch 15/30
7352/7352 [==============================] - 182s 25ms/step - loss: 0.2006
- accuracy: 0.9377 - val_loss: 0.3333 - val_accuracy: 0.9070
Epoch 16/30
7352/7352 [==============================] - 213s 29ms/step - loss: 0.1993
- accuracy: 0.9354 - val_loss: 0.4473 - val_accuracy: 0.8935
Epoch 17/30
7352/7352 [==============================] - 214s 29ms/step - loss: 0.1782
- accuracy: 0.9414 - val_loss: 0.3812 - val_accuracy: 0.9016
```

Out[100]:

```
<keras.callbacks.callbacks.History at 0x1fc2f095eb8>
```

In [101]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS
\
True
LAYING                 536        0         0        0                   0
SITTING                  6      353       116        2                   0
STANDING                 0       74       451        4                   0
WALKING                  0        0         0      452                  32
WALKING_DOWNSTAIRS       0        0         0        2                 418
WALKING_UPSTAIRS         0        0         0       15                   9

Pred                WALKING_UPSTAIRS
True
LAYING                             1
SITTING                           14
STANDING                           3
WALKING                           12
WALKING_DOWNSTAIRS                 0
WALKING_UPSTAIRS                 447
```

In [102]:

```
score = model.evaluate(X_test, Y_test)
score
```

```
2947/2947 [==============================] - 19s 7ms/step
```

Out[102]:

```
[0.38118069241371727, 0.9015948176383972]
```

In [103]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(28,return_sequences = True,input_shape=(128,9)))
# Adding a dropout layer
model.add(Dropout(0.5))
model.add(LSTM(28,return_sequences = False,input_shape=(128,9)))
# Adding a dropout layer
model.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_34"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_56 (LSTM)               (None, 128, 28)           4256
_________________________________________________________________
dropout_40 (Dropout)         (None, 128, 28)           0
_________________________________________________________________
lstm_57 (LSTM)               (None, 28)                6384
_________________________________________________________________
dropout_41 (Dropout)         (None, 28)                0
_________________________________________________________________
dense_19 (Dense)             (None, 6)                 174
=================================================================
Total params: 10,814
Trainable params: 10,814
Non-trainable params: 0
_________________________________________________________________
```

In [104]:

```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [105]:

```
from keras.callbacks.callbacks import EarlyStopping
early_stop=EarlyStopping(monitor='val_accuracy',patience=2)
callbacks = [early_stop]
```

In [106]:

```
#Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs,callbacks = callbacks)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 214s 29ms/step - loss: 1.2012
- accuracy: 0.5029 - val_loss: 1.0125 - val_accuracy: 0.4425
Epoch 2/30
7352/7352 [==============================] - 212s 29ms/step - loss: 0.8261
- accuracy: 0.6285 - val_loss: 0.7214 - val_accuracy: 0.6216
Epoch 3/30
7352/7352 [==============================] - 209s 28ms/step - loss: 0.6916
- accuracy: 0.6912 - val_loss: 0.9501 - val_accuracy: 0.6250
Epoch 4/30
7352/7352 [==============================] - 214s 29ms/step - loss: 0.6101
- accuracy: 0.7364 - val_loss: 0.5421 - val_accuracy: 0.7655
Epoch 5/30
7352/7352 [==============================] - 229s 31ms/step - loss: 0.5116
- accuracy: 0.7809 - val_loss: 0.5140 - val_accuracy: 0.7635
Epoch 6/30
7352/7352 [==============================] - 258s 35ms/step - loss: 0.4753
- accuracy: 0.7968 - val_loss: 0.7294 - val_accuracy: 0.7234
```

Out[106]:

```
<keras.callbacks.callbacks.History at 0x1fc44b5bfd0>
```

In [107]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(28,return_sequences = True,input_shape=(128,9)))
# Adding a dropout layer
model.add(Dropout(0.75))
model.add(LSTM(28,return_sequences = False,input_shape=(128,9)))
# Adding a dropout layer
model.add(Dropout(0.75))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_35"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_58 (LSTM)               (None, 128, 28)           4256
_________________________________________________________________
dropout_42 (Dropout)         (None, 128, 28)           0
_________________________________________________________________
lstm_59 (LSTM)               (None, 28)                6384
_________________________________________________________________
dropout_43 (Dropout)         (None, 28)                0
_________________________________________________________________
dense_20 (Dense)             (None, 6)                 174
=================================================================
Total params: 10,814
Trainable params: 10,814
Non-trainable params: 0
_________________________________________________________________
```

In [108]:

```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [109]:

```
from keras.callbacks.callbacks import EarlyStopping
early_stop=EarlyStopping(monitor='val_accuracy',patience=2)
callbacks = [early_stop]
```

In [110]:

```
#Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs,callbacks = callbacks)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 122s 17ms/step - loss: 1.3003
- accuracy: 0.4597 - val_loss: 0.9513 - val_accuracy: 0.5375
Epoch 2/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.9403
- accuracy: 0.5325 - val_loss: 0.8734 - val_accuracy: 0.5182
Epoch 3/30
7352/7352 [==============================] - 121s 17ms/step - loss: 0.8687
- accuracy: 0.5355 - val_loss: 0.8599 - val_accuracy: 0.5205
```

Out[110]:

```
<keras.callbacks.callbacks.History at 0x1fc5d58bf28>
```

In [111]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(28,return_sequences = True,input_shape=(128,9)))
# Adding a dropout layer
model.add(Dropout(0.25))
model.add(LSTM(28,return_sequences = False,input_shape=(128,9)))
# Adding a dropout layer
model.add(Dropout(0.25))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_36"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_60 (LSTM)               (None, 128, 28)           4256
_________________________________________________________________
dropout_44 (Dropout)         (None, 128, 28)           0
_________________________________________________________________
lstm_61 (LSTM)               (None, 28)                6384
_________________________________________________________________
dropout_45 (Dropout)         (None, 28)                0
_________________________________________________________________
dense_21 (Dense)             (None, 6)                 174
=================================================================
Total params: 10,814
Trainable params: 10,814
Non-trainable params: 0
_________________________________________________________________
```

In [112]:

```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [113]:

```
from keras.callbacks.callbacks import EarlyStopping
early_stop=EarlyStopping(monitor='val_accuracy',patience=2)
callbacks = [early_stop]
```

In [114]:

```
#Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs,callbacks = callbacks)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 122s 17ms/step - loss: 1.1266
- accuracy: 0.5020 - val_loss: 0.9131 - val_accuracy: 0.5460
Epoch 2/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.7529
- accuracy: 0.6326 - val_loss: 0.7405 - val_accuracy: 0.6342
Epoch 3/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.6505
- accuracy: 0.6881 - val_loss: 0.7630 - val_accuracy: 0.6658
Epoch 4/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.5036
- accuracy: 0.7799 - val_loss: 0.5538 - val_accuracy: 0.7686
Epoch 5/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.4192
- accuracy: 0.8275 - val_loss: 0.4469 - val_accuracy: 0.8398
Epoch 6/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.3390
- accuracy: 0.8791 - val_loss: 0.3812 - val_accuracy: 0.8633
Epoch 7/30
7352/7352 [==============================] - 123s 17ms/step - loss: 0.2468
- accuracy: 0.9222 - val_loss: 0.4311 - val_accuracy: 0.8697
Epoch 8/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.2197
- accuracy: 0.9268 - val_loss: 0.3363 - val_accuracy: 0.8924
Epoch 9/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.1919
- accuracy: 0.9346 - val_loss: 0.2918 - val_accuracy: 0.9009
Epoch 10/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.1662
- accuracy: 0.9372 - val_loss: 0.3171 - val_accuracy: 0.8992
Epoch 11/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.1460
- accuracy: 0.9467 - val_loss: 0.3859 - val_accuracy: 0.9077
Epoch 12/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.1543
- accuracy: 0.9465 - val_loss: 0.3544 - val_accuracy: 0.8992
Epoch 13/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.1480
- accuracy: 0.9438 - val_loss: 0.4082 - val_accuracy: 0.9074
```

Out[114]:

```
<keras.callbacks.callbacks.History at 0x1fc77fd2fd0>
```

In [115]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS
\
True
LAYING                 537        0         0        0                   0
SITTING                  2      370        96       17                   2
STANDING                 0       61       468        3                   0
WALKING                  0        0         0      468                  26
WALKING_DOWNSTAIRS       0        0         0        3                 416
WALKING_UPSTAIRS         0        1         1        8                  46

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            4
STANDING                           0
WALKING                            2
WALKING_DOWNSTAIRS                 1
WALKING_UPSTAIRS                 415
```

In [116]:

```
score = model.evaluate(X_test, Y_test)
score
```

```
2947/2947 [==============================] - 12s 4ms/step
```

Out[116]:

```
[0.4081978703499752, 0.9073634147644043]
```

In [ ]:

In [16]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(64,return_sequences = True,input_shape=(128,9)))
# Adding a dropout layer
model.add(Dropout(0.25))
model.add(LSTM(32,return_sequences = False,input_shape=(128,9)))
# Adding a dropout layer
model.add(Dropout(0.25))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 128, 64)           18944
_________________________________________________________________
dropout_1 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
lstm_2 (LSTM)                (None, 32)                12416
_________________________________________________________________
dropout_2 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 198
=================================================================
Total params: 31,558
Trainable params: 31,558
Non-trainable params: 0
_________________________________________________________________
```

In [17]:

```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [18]:

```
from keras.callbacks.callbacks import EarlyStopping
early_stop=EarlyStopping(monitor='val_accuracy',patience=2)
callbacks = [early_stop]
```

In [19]:

```
#Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs,callbacks = callbacks)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 144s 20ms/step - loss: 1.1056
- accuracy: 0.5331 - val_loss: 0.8742 - val_accuracy: 0.6356
Epoch 2/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.6890
- accuracy: 0.7325 - val_loss: 0.6520 - val_accuracy: 0.7615
Epoch 3/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.4847
- accuracy: 0.8312 - val_loss: 0.4826 - val_accuracy: 0.8500
Epoch 4/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.2758
- accuracy: 0.9173 - val_loss: 0.4425 - val_accuracy: 0.8643
Epoch 5/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.2132
- accuracy: 0.9347 - val_loss: 0.3679 - val_accuracy: 0.8904
Epoch 6/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.1813
- accuracy: 0.9436 - val_loss: 0.4470 - val_accuracy: 0.8860
Epoch 7/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.1729
- accuracy: 0.9399 - val_loss: 0.4217 - val_accuracy: 0.8924
Epoch 8/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.1462
- accuracy: 0.9452 - val_loss: 0.4563 - val_accuracy: 0.9002
Epoch 9/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.1561
- accuracy: 0.9457 - val_loss: 0.4446 - val_accuracy: 0.8901
Epoch 10/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.1370
- accuracy: 0.9510 - val_loss: 0.3285 - val_accuracy: 0.9128
Epoch 11/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.1427
- accuracy: 0.9480 - val_loss: 0.5696 - val_accuracy: 0.8860
Epoch 12/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.1353
- accuracy: 0.9518 - val_loss: 0.4174 - val_accuracy: 0.9043
```

Out[19]:

```
<keras.callbacks.callbacks.History at 0x152db8f1f60>
```

In [20]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS
\
True
LAYING                 537        0         0        0                   0
SITTING                  2      375       112        0                   0
STANDING                 0       55       477        0                   0
WALKING                  0        0        15      424                  46
WALKING_DOWNSTAIRS       0        0         0        0                 420
WALKING_UPSTAIRS         0        0         3       13                  23

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            2
STANDING                           0
WALKING                           11
WALKING_DOWNSTAIRS                 0
WALKING_UPSTAIRS                 432
```

In [21]:

```
score = model.evaluate(X_test, Y_test)
score
```

```
2947/2947 [==============================] - 14s 5ms/step
```

Out[21]:

```
[0.41741838527891234, 0.9043094515800476]
```

In [22]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(64,return_sequences = True,input_shape=(128,9)))
# Adding a dropout layer
model.add(Dropout(0.5))
model.add(LSTM(32,return_sequences = False,input_shape=(128,9)))
# Adding a dropout layer
model.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 128, 64)           18944
_________________________________________________________________
dropout_3 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
lstm_4 (LSTM)                (None, 32)                12416
_________________________________________________________________
dropout_4 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 198
=================================================================
Total params: 31,558
Trainable params: 31,558
Non-trainable params: 0
_________________________________________________________________
```

In [25]:

```
from keras.callbacks.callbacks import EarlyStopping
from keras.callbacks.callbacks import ModelCheckpoint
checkpoint_1 = ModelCheckpoint("model_1.1", monitor="val_accuracy",mode="max",save_best
_only = True, verbose=1)


early_stop=EarlyStopping(monitor='val_accuracy',patience=2)
callbacks = [checkpoint_1,early_stop]
```

In [26]:

```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [27]:

```
#Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs,callbacks = callbacks)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 141s 19ms/step - loss: 1.1688
- accuracy: 0.5530 - val_loss: 0.8774 - val_accuracy: 0.6518

Epoch 00001: val_accuracy improved from -inf to 0.65185, saving model to m
odel_1.1
Epoch 2/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.7562
- accuracy: 0.6999 - val_loss: 0.6272 - val_accuracy: 0.7289

Epoch 00002: val_accuracy improved from 0.65185 to 0.72888, saving model t
o model_1.1
Epoch 3/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.6180
- accuracy: 0.7519 - val_loss: 0.6287 - val_accuracy: 0.7387

Epoch 00003: val_accuracy improved from 0.72888 to 0.73872, saving model t
o model_1.1
Epoch 4/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.5084
- accuracy: 0.8173 - val_loss: 0.4337 - val_accuracy: 0.8324

Epoch 00004: val_accuracy improved from 0.73872 to 0.83237, saving model t
o model_1.1
Epoch 5/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.3413
- accuracy: 0.8984 - val_loss: 0.3665 - val_accuracy: 0.8806

Epoch 00005: val_accuracy improved from 0.83237 to 0.88056, saving model t
o model_1.1
Epoch 6/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.2625
- accuracy: 0.9257 - val_loss: 0.3883 - val_accuracy: 0.8982

Epoch 00006: val_accuracy improved from 0.88056 to 0.89820, saving model t
o model_1.1
Epoch 7/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.2205
- accuracy: 0.9329 - val_loss: 0.4034 - val_accuracy: 0.8938

Epoch 00007: val_accuracy did not improve from 0.89820
Epoch 8/30
7352/7352 [==============================] - 142s 19ms/step - loss: 0.2163
- accuracy: 0.9354 - val_loss: 0.3237 - val_accuracy: 0.9152

Epoch 00008: val_accuracy improved from 0.89820 to 0.91517, saving model t
o model_1.1
Epoch 9/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.1889
- accuracy: 0.9418 - val_loss: 0.3270 - val_accuracy: 0.9114

Epoch 00009: val_accuracy did not improve from 0.91517
Epoch 10/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.1862
- accuracy: 0.9402 - val_loss: 0.5101 - val_accuracy: 0.8839

Epoch 00010: val_accuracy did not improve from 0.91517
```

Out[27]:

```
<keras.callbacks.callbacks.History at 0x152f0d08dd8>
```

In [31]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAI
RS  \
True
LAYING                 510        0         3        0
0
SITTING                  0      375       112        0
0
STANDING                 0       74       456        2
0
WALKING                  0        0         0      463
0
WALKING_DOWNSTAIRS       0        0         0       19                 3
50
WALKING_UPSTAIRS         0        0         0       20
0

Pred                WALKING_UPSTAIRS
True
LAYING                            24
SITTING                            4
STANDING                           0
WALKING                           33
WALKING_DOWNSTAIRS                51
WALKING_UPSTAIRS                 451
```

In [34]:

```
score = model.evaluate(X_test, Y_test)
score
```

```
2947/2947 [==============================] - 15s 5ms/step
```

Out[34]:

```
[0.5100547778261941, 0.8839497566223145]
```

In [35]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(64,return_sequences = True,input_shape=(128,9)))
# Adding a dropout layer
model.add(Dropout(0.75))
model.add(LSTM(32,return_sequences = False,input_shape=(128,9)))
# Adding a dropout layer
model.add(Dropout(0.75))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
WARNING:tensorflow:Large dropout rate: 0.75 (>0.5). In TensorFlow 2.x, dro
pout() uses dropout rate instead of keep_prob. Please ensure that this is
intended.
WARNING:tensorflow:Large dropout rate: 0.75 (>0.5). In TensorFlow 2.x, dro
pout() uses dropout rate instead of keep_prob. Please ensure that this is
intended.
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_5 (LSTM)                (None, 128, 64)           18944
_________________________________________________________________
dropout_5 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
lstm_6 (LSTM)                (None, 32)                12416
_________________________________________________________________
dropout_6 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 198
=================================================================
Total params: 31,558
Trainable params: 31,558
Non-trainable params: 0
_________________________________________________________________
```

In [36]:

```
from keras.callbacks.callbacks import EarlyStopping
from keras.callbacks.callbacks import ModelCheckpoint
checkpoint_1 = ModelCheckpoint("model_1.1", monitor="val_accuracy",mode="max",save_best
_only = True, verbose=1)


early_stop=EarlyStopping(monitor='val_accuracy',patience=2)
callbacks = [checkpoint_1,early_stop]
```

In [37]:

```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [38]:

```
#Training the model
LSTM=model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs,callbacks = callbacks)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 141s 19ms/step - loss: 1.3000
- accuracy: 0.4868 - val_loss: 0.9620 - val_accuracy: 0.5911

Epoch 00001: val_accuracy improved from -inf to 0.59111, saving model to m
odel_1.1
Epoch 2/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.9806
- accuracy: 0.5770 - val_loss: 0.8821 - val_accuracy: 0.5857

Epoch 00002: val_accuracy did not improve from 0.59111
Epoch 3/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.8653
- accuracy: 0.6137 - val_loss: 0.8124 - val_accuracy: 0.6077

Epoch 00003: val_accuracy improved from 0.59111 to 0.60774, saving model t
o model_1.1
Epoch 4/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.8039
- accuracy: 0.6249 - val_loss: 0.7777 - val_accuracy: 0.6196

Epoch 00004: val_accuracy improved from 0.60774 to 0.61961, saving model t
o model_1.1
Epoch 5/30
7352/7352 [==============================] - 142s 19ms/step - loss: 0.7677
- accuracy: 0.6489 - val_loss: 0.7959 - val_accuracy: 0.6227

Epoch 00005: val_accuracy improved from 0.61961 to 0.62267, saving model t
o model_1.1
Epoch 6/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.7455
- accuracy: 0.6540 - val_loss: 0.8052 - val_accuracy: 0.6240

Epoch 00006: val_accuracy improved from 0.62267 to 0.62402, saving model t
o model_1.1
Epoch 7/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.7277
- accuracy: 0.6726 - val_loss: 0.7568 - val_accuracy: 0.7268

Epoch 00007: val_accuracy improved from 0.62402 to 0.72684, saving model t
o model_1.1
Epoch 8/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.7001
- accuracy: 0.7150 - val_loss: 0.6642 - val_accuracy: 0.7397

Epoch 00008: val_accuracy improved from 0.72684 to 0.73974, saving model t
o model_1.1
Epoch 9/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.6514
- accuracy: 0.7252 - val_loss: 0.7880 - val_accuracy: 0.6861

Epoch 00009: val_accuracy did not improve from 0.73974
Epoch 10/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.6056
- accuracy: 0.7578 - val_loss: 0.5996 - val_accuracy: 0.8202

Epoch 00010: val_accuracy improved from 0.73974 to 0.82016, saving model t
o model_1.1
Epoch 11/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.5242
```

```
- accuracy: 0.8245 - val_loss: 0.5021 - val_accuracy: 0.8782

Epoch 00011: val_accuracy improved from 0.82016 to 0.87818, saving model t
o model_1.1
Epoch 12/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.4641
- accuracy: 0.8595 - val_loss: 1.1375 - val_accuracy: 0.8188

Epoch 00012: val_accuracy did not improve from 0.87818
Epoch 13/30
7352/7352 [==============================] - 138s 19ms/step - loss: 0.4272
- accuracy: 0.8841 - val_loss: 0.6221 - val_accuracy: 0.8751

Epoch 00013: val_accuracy did not improve from 0.87818
```

In [40]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAI
RS  \
True
LAYING                 510        0         0        0
0
SITTING                  0      418        69        1
0
STANDING                 0      114       406        1
0
WALKING                  0        0         0      449
2
WALKING_DOWNSTAIRS       0        0         0       25                 3
46
WALKING_UPSTAIRS         0        0         0       18
3

Pred                WALKING_UPSTAIRS
True
LAYING                            27
SITTING                            3
STANDING                          11
WALKING                           45
WALKING_DOWNSTAIRS                49
WALKING_UPSTAIRS                 450
```

In [41]:

```
score = model.evaluate(X_test, Y_test)
score
```

```
2947/2947 [==============================] - 16s 5ms/step
```

Out[41]:

```
[0.622152369769943, 0.8751272559165955]
```

In [20]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(64,return_sequences = True,input_shape=(128,9)))
# Adding a dropout layer
model.add(Dropout(0.6))
model.add(LSTM(64,return_sequences = False,input_shape=(128,9)))
# Adding a dropout layer
model.add(Dropout(0.6))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
WARNING:tensorflow:Large dropout rate: 0.6 (>0.5). In TensorFlow 2.x, drop
out() uses dropout rate instead of keep_prob. Please ensure that this is i
ntended.
WARNING:tensorflow:Large dropout rate: 0.6 (>0.5). In TensorFlow 2.x, drop
out() uses dropout rate instead of keep_prob. Please ensure that this is i
ntended.
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 128, 64)           18944
_________________________________________________________________
dropout_3 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
lstm_4 (LSTM)                (None, 64)                33024
_________________________________________________________________
dropout_4 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 390
=================================================================
Total params: 52,358
Trainable params: 52,358
Non-trainable params: 0
_________________________________________________________________
```

In [21]:

```
from keras.callbacks.callbacks import EarlyStopping
from keras.callbacks.callbacks import ModelCheckpoint
checkpoint_1 = ModelCheckpoint("model_1.1", monitor="val_accuracy",mode="max",save_best
_only = True, verbose=1)


early_stop=EarlyStopping(monitor='val_accuracy',patience=2)
callbacks = [checkpoint_1,early_stop]
```

In [22]:

```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [23]:

```
#Training the model
lstm=model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs,callbacks = callbacks)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 228s 31ms/step - loss: 1.0310
- accuracy: 0.5638 - val_loss: 0.7878 - val_accuracy: 0.6614

Epoch 00001: val_accuracy improved from -inf to 0.66135, saving model to m
odel_1.1
Epoch 2/30
7352/7352 [==============================] - 238s 32ms/step - loss: 0.6422
- accuracy: 0.7232 - val_loss: 0.7311 - val_accuracy: 0.7150

Epoch 00002: val_accuracy improved from 0.66135 to 0.71496, saving model t
o model_1.1
Epoch 3/30
7352/7352 [==============================] - 261s 35ms/step - loss: 0.4935
- accuracy: 0.7767 - val_loss: 0.5442 - val_accuracy: 0.7655

Epoch 00003: val_accuracy improved from 0.71496 to 0.76552, saving model t
o model_1.1
Epoch 4/30
7352/7352 [==============================] - 302s 41ms/step - loss: 0.3849
- accuracy: 0.8541 - val_loss: 0.4581 - val_accuracy: 0.8833

Epoch 00004: val_accuracy improved from 0.76552 to 0.88327, saving model t
o model_1.1
Epoch 5/30
7352/7352 [==============================] - 148s 20ms/step - loss: 0.2652
- accuracy: 0.9166 - val_loss: 0.3878 - val_accuracy: 0.8799

Epoch 00005: val_accuracy did not improve from 0.88327
Epoch 6/30
7352/7352 [==============================] - 130s 18ms/step - loss: 0.2290
- accuracy: 0.9282 - val_loss: 0.3590 - val_accuracy: 0.9026

Epoch 00006: val_accuracy improved from 0.88327 to 0.90261, saving model t
o model_1.1
Epoch 7/30
7352/7352 [==============================] - 128s 17ms/step - loss: 0.1847
- accuracy: 0.9361 - val_loss: 0.3647 - val_accuracy: 0.8996

Epoch 00007: val_accuracy did not improve from 0.90261
Epoch 8/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.1663
- accuracy: 0.9426 - val_loss: 0.4156 - val_accuracy: 0.8836

Epoch 00008: val_accuracy did not improve from 0.90261
```

In [24]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS
\
True
LAYING                 524        0         0        0                   0
SITTING                  2      462        23        2                   0
STANDING                 0      220       309        3                   0
WALKING                  0        0         0      441                  54
WALKING_DOWNSTAIRS       0        0         0        0                 420
WALKING_UPSTAIRS         0        0         0        8                  15

Pred                WALKING_UPSTAIRS
True
LAYING                            13
SITTING                            2
STANDING                           0
WALKING                            1
WALKING_DOWNSTAIRS                 0
WALKING_UPSTAIRS                 448
```

In [25]:

```
score = model.evaluate(X_test, Y_test)
score
```

```
2947/2947 [==============================] - 12s 4ms/step
```

Out[25]:

```
[0.41557583960709327, 0.8836104273796082]
```

In [16]:

```
# Initiliazing the sequential model
from keras.layers.normalization import BatchNormalization
model = Sequential()
# Configuring the parameters
model.add(LSTM(128, input_shape=(timesteps, input_dim)))
model.add(BatchNormalization())
# Adding a dropout layer
model.add(Dropout(0.25))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 128)               70656
_________________________________________________________________
batch_normalization_1 (Batch (None, 128)               512
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 774
=================================================================
Total params: 71,942
Trainable params: 71,686
Non-trainable params: 256
_________________________________________________________________
```

In [17]:

```
from keras.callbacks.callbacks import EarlyStopping
from keras.callbacks.callbacks import ModelCheckpoint
checkpoint_1 = ModelCheckpoint("best_model_1.1", monitor="val_accuracy",mode="max",save
_best_only = True, verbose=1)


early_stop=EarlyStopping(monitor='val_accuracy',patience=3)
callbacks = [checkpoint_1,early_stop]
```

In [18]:

```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [20]:

```
#Training the model
model.fit(X_train,
          Y_train,
          batch_size=32,
          validation_data=(X_test, Y_test),
          epochs=20,callbacks = callbacks)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/20
7352/7352 [==============================] - 74s 10ms/step - loss: 0.8231
- accuracy: 0.6401 - val_loss: 0.7539 - val_accuracy: 0.6705

Epoch 00001: val_accuracy improved from -inf to 0.67051, saving model to b
est_model_1.1
Epoch 2/20
7352/7352 [==============================] - 72s 10ms/step - loss: 0.6254
- accuracy: 0.7359 - val_loss: 0.5794 - val_accuracy: 0.7896

Epoch 00002: val_accuracy improved from 0.67051 to 0.78962, saving model t
o best_model_1.1
Epoch 3/20
7352/7352 [==============================] - 73s 10ms/step - loss: 0.3875
- accuracy: 0.8818 - val_loss: 0.5783 - val_accuracy: 0.8229

Epoch 00003: val_accuracy improved from 0.78962 to 0.82287, saving model t
o best_model_1.1
Epoch 4/20
7352/7352 [==============================] - 72s 10ms/step - loss: 0.2199
- accuracy: 0.9257 - val_loss: 0.3194 - val_accuracy: 0.9026

Epoch 00004: val_accuracy improved from 0.82287 to 0.90261, saving model t
o best_model_1.1
Epoch 5/20
7352/7352 [==============================] - 69s 9ms/step - loss: 0.1905 -
accuracy: 0.9305 - val_loss: 0.2341 - val_accuracy: 0.9077

Epoch 00005: val_accuracy improved from 0.90261 to 0.90770, saving model t
o best_model_1.1
Epoch 6/20
7352/7352 [==============================] - 70s 10ms/step - loss: 0.1669
- accuracy: 0.9370 - val_loss: 0.2713 - val_accuracy: 0.9118

Epoch 00006: val_accuracy improved from 0.90770 to 0.91177, saving model t
o best_model_1.1
Epoch 7/20
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1669
- accuracy: 0.9376 - val_loss: 0.2513 - val_accuracy: 0.9135

Epoch 00007: val_accuracy improved from 0.91177 to 0.91347, saving model t
o best_model_1.1
Epoch 8/20
7352/7352 [==============================] - 70s 9ms/step - loss: 0.1552 -
accuracy: 0.9408 - val_loss: 0.2227 - val_accuracy: 0.9264

Epoch 00008: val_accuracy improved from 0.91347 to 0.92637, saving model t
o best_model_1.1
Epoch 9/20
7352/7352 [==============================] - 73s 10ms/step - loss: 0.1447
- accuracy: 0.9425 - val_loss: 0.2288 - val_accuracy: 0.9277

Epoch 00009: val_accuracy improved from 0.92637 to 0.92772, saving model t
o best_model_1.1
Epoch 10/20
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1417
- accuracy: 0.9389 - val_loss: 1.2786 - val_accuracy: 0.7988

Epoch 00010: val_accuracy did not improve from 0.92772
Epoch 11/20
```

```
7352/7352 [==============================] - 71s 10ms/step - loss: 0.1501
- accuracy: 0.9399 - val_loss: 0.2467 - val_accuracy: 0.9345

Epoch 00011: val_accuracy improved from 0.92772 to 0.93451, saving model t
o best_model_1.1
Epoch 12/20
7352/7352 [==============================] - 71s 10ms/step - loss: 0.1430
- accuracy: 0.9381 - val_loss: 0.2908 - val_accuracy: 0.9131

Epoch 00012: val_accuracy did not improve from 0.93451
Epoch 13/20
7352/7352 [==============================] - 75s 10ms/step - loss: 0.1298
- accuracy: 0.9406 - val_loss: 0.3011 - val_accuracy: 0.9189

Epoch 00013: val_accuracy did not improve from 0.93451
Epoch 14/20
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1283
- accuracy: 0.9476 - val_loss: 0.3161 - val_accuracy: 0.9199

Epoch 00014: val_accuracy did not improve from 0.93451
```

Out[20]:

```
<keras.callbacks.callbacks.History at 0x2d9d8f00ef0>
```

In [21]:

```
from keras.models import load_model
saved_model = load_model('best_model_1.1')
```

In [22]:

```
# Confusion Matrix
print(confusion_matrix(Y_test, saved_model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAI
RS  \
True
LAYING                 537        0         0        0
0
SITTING                  2      405        82        0
0
STANDING                 0       69       463        0
0
WALKING                  0        1         2      464
24
WALKING_DOWNSTAIRS       0        0         0        0                 4
17
WALKING_UPSTAIRS         0        1         1        0
1

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            2
STANDING                           0
WALKING                            5
WALKING_DOWNSTAIRS                 3
WALKING_UPSTAIRS                 468
```

In [23]:

```
score = saved_model.evaluate(X_test, Y_test)
score
```

```
2947/2947 [==============================] - 13s 4ms/step
```

Out[23]:

```
[0.24674624278151186, 0.9345096945762634]
```

In [24]:

```
# Initiliazing the sequential model
model1 = Sequential()
# Configuring the parameters
model1.add(LSTM(128, return_sequences=True, input_shape=(128,9)))
# Adding a dropout layer
model1.add(Dropout(0.2))

model1.add(LSTM(64))
# Adding a dropout layer
model1.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model1.add(Dense(n_classes, activation='sigmoid'))
model1.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 128, 128)          70656
_________________________________________________________________
dropout_2 (Dropout)          (None, 128, 128)          0
_________________________________________________________________
lstm_3 (LSTM)                (None, 64)                49408
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 390
=================================================================
Total params: 120,454
Trainable params: 120,454
Non-trainable params: 0
_________________________________________________________________
```

In [25]:

```
from keras.callbacks.callbacks import EarlyStopping
from keras.callbacks.callbacks import ModelCheckpoint
checkpoint_1 = ModelCheckpoint("best_model_2.1", monitor="val_accuracy",mode="max",save
_best_only = True, verbose=1)


early_stop=EarlyStopping(monitor='val_accuracy',patience=3)
callbacks = [checkpoint_1,early_stop]
```

In [27]:

```
# Compiling the model
model1.compile(loss='categorical_crossentropy',
               optimizer='rmsprop',
               metrics=['accuracy'])
```

In [28]:

```
#Training the model
model1.fit(X_train,
          Y_train,
          batch_size=32,
          validation_data=(X_test, Y_test),
          epochs=epochs,callbacks = callbacks)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 106s 14ms/step - loss: 1.0653
- accuracy: 0.5418 - val_loss: 0.8596 - val_accuracy: 0.6308

Epoch 00001: val_accuracy improved from -inf to 0.63081, saving model to b
est_model_2.1
Epoch 2/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.7825
- accuracy: 0.6428 - val_loss: 0.9171 - val_accuracy: 0.5925

Epoch 00002: val_accuracy did not improve from 0.63081
Epoch 3/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.6863
- accuracy: 0.7084 - val_loss: 0.6890 - val_accuracy: 0.7014

Epoch 00003: val_accuracy improved from 0.63081 to 0.70139, saving model t
o best_model_2.1
Epoch 4/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.5301
- accuracy: 0.8040 - val_loss: 0.4430 - val_accuracy: 0.8680

Epoch 00004: val_accuracy improved from 0.70139 to 0.86800, saving model t
o best_model_2.1
Epoch 5/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.2994
- accuracy: 0.9071 - val_loss: 0.4317 - val_accuracy: 0.8738

Epoch 00005: val_accuracy improved from 0.86800 to 0.87377, saving model t
o best_model_2.1
Epoch 6/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.2194
- accuracy: 0.9305 - val_loss: 0.3004 - val_accuracy: 0.8996

Epoch 00006: val_accuracy improved from 0.87377 to 0.89956, saving model t
o best_model_2.1
Epoch 7/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1942
- accuracy: 0.9368 - val_loss: 0.3843 - val_accuracy: 0.9013

Epoch 00007: val_accuracy improved from 0.89956 to 0.90126, saving model t
o best_model_2.1
Epoch 8/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1742
- accuracy: 0.9419 - val_loss: 0.3104 - val_accuracy: 0.9002

Epoch 00008: val_accuracy did not improve from 0.90126
Epoch 9/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1577
- accuracy: 0.9436 - val_loss: 0.2489 - val_accuracy: 0.8985

Epoch 00009: val_accuracy did not improve from 0.90126
Epoch 10/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1640
- accuracy: 0.9411 - val_loss: 0.3142 - val_accuracy: 0.9131

Epoch 00010: val_accuracy improved from 0.90126 to 0.91313, saving model t
o best_model_2.1
Epoch 11/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1501
- accuracy: 0.9471 - val_loss: 0.3243 - val_accuracy: 0.9053
```

```
Epoch 00011: val_accuracy did not improve from 0.91313
Epoch 12/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1541
- accuracy: 0.9438 - val_loss: 0.3798 - val_accuracy: 0.9165

Epoch 00012: val_accuracy improved from 0.91313 to 0.91653, saving model t
o best_model_2.1
Epoch 13/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.1351
- accuracy: 0.9506 - val_loss: 0.3312 - val_accuracy: 0.9172

Epoch 00013: val_accuracy improved from 0.91653 to 0.91720, saving model t
o best_model_2.1
Epoch 14/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.1325
- accuracy: 0.9508 - val_loss: 0.3209 - val_accuracy: 0.9152

Epoch 00014: val_accuracy did not improve from 0.91720
Epoch 15/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.1524
- accuracy: 0.9475 - val_loss: 0.3072 - val_accuracy: 0.9138

Epoch 00015: val_accuracy did not improve from 0.91720
Epoch 16/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1317
- accuracy: 0.9502 - val_loss: 0.2809 - val_accuracy: 0.9192

Epoch 00016: val_accuracy improved from 0.91720 to 0.91924, saving model t
o best_model_2.1
Epoch 17/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1331
- accuracy: 0.9484 - val_loss: 0.2742 - val_accuracy: 0.9209

Epoch 00017: val_accuracy improved from 0.91924 to 0.92094, saving model t
o best_model_2.1
Epoch 18/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1178
- accuracy: 0.9535 - val_loss: 0.3097 - val_accuracy: 0.9199

Epoch 00018: val_accuracy did not improve from 0.92094
Epoch 19/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1320
- accuracy: 0.9523 - val_loss: 0.3973 - val_accuracy: 0.9158

Epoch 00019: val_accuracy did not improve from 0.92094
Epoch 20/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1196
- accuracy: 0.9536 - val_loss: 0.4392 - val_accuracy: 0.9145

Epoch 00020: val_accuracy did not improve from 0.92094
```

Out[28]:

```
<keras.callbacks.callbacks.History at 0x2d9f0f92eb8>
```

In [29]:

```
from keras.models import load_model
saved_model_1 = load_model('best_model_2.1')
```

In [30]:

```
# Confusion Matrix
print(confusion_matrix(Y_test, saved_model_1.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS
\
True
LAYING                 537        0         0        0                   0
SITTING                  5      378        99        0                   0
STANDING                 0       67       464        0                   0
WALKING                  0        0         0      467                  15
WALKING_DOWNSTAIRS       0        0         0        1                 412
WALKING_UPSTAIRS         0        0         0       13                   2

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            9
STANDING                           1
WALKING                           14
WALKING_DOWNSTAIRS                 7
WALKING_UPSTAIRS                 456
```

In [31]:

```
score = saved_model_1.evaluate(X_test, Y_test)
score
```

```
2947/2947 [==============================] - 14s 5ms/step
```

Out[31]:

```
[0.27421482619152027, 0.9209365248680115]
```

In [36]:

```
# Initiliazing the sequential model
model1 = Sequential()
# Configuring the parameters
model1.add(LSTM(128, return_sequences=True, input_shape=(128,9)))
# Adding a dropout layer
model1.add(Dropout(0.2))

model1.add(LSTM(64))
# Adding a dropout layer
model1.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model1.add(Dense(n_classes, activation='sigmoid'))
model1.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_6 (LSTM)                (None, 128, 128)          70656
_________________________________________________________________
dropout_6 (Dropout)          (None, 128, 128)          0
_________________________________________________________________
lstm_7 (LSTM)                (None, 64)                49408
_________________________________________________________________
dropout_7 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_4 (Dense)              (None, 6)                 390
=================================================================
Total params: 120,454
Trainable params: 120,454
Non-trainable params: 0
_________________________________________________________________
```

In [37]:

```
from keras.callbacks.callbacks import EarlyStopping
from keras.callbacks.callbacks import ModelCheckpoint
checkpoint_1 = ModelCheckpoint("best_model_3.1", monitor="val_accuracy",mode="max",save
_best_only = True, verbose=1)


early_stop=EarlyStopping(monitor='val_accuracy',patience=5)
callbacks = [checkpoint_1,early_stop]
```

In [38]:

```
# Compiling the model
model1.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [39]:

```
#Training the model
model1.fit(X_train,
          Y_train,
          batch_size=32,
          validation_data=(X_test, Y_test),
          epochs=epochs,callbacks = callbacks)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 71s 10ms/step - loss: 1.0782
- accuracy: 0.5335 - val_loss: 0.8807 - val_accuracy: 0.6040

Epoch 00001: val_accuracy improved from -inf to 0.60400, saving model to b
est_model_3.1
Epoch 2/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.7853
- accuracy: 0.6530 - val_loss: 0.7115 - val_accuracy: 0.7095

Epoch 00002: val_accuracy improved from 0.60400 to 0.70954, saving model t
o best_model_3.1
Epoch 3/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.6406
- accuracy: 0.7206 - val_loss: 0.5805 - val_accuracy: 0.7523

Epoch 00003: val_accuracy improved from 0.70954 to 0.75229, saving model t
o best_model_3.1
Epoch 4/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.4090
- accuracy: 0.8376 - val_loss: 0.4350 - val_accuracy: 0.8809

Epoch 00004: val_accuracy improved from 0.75229 to 0.88090, saving model t
o best_model_3.1
Epoch 5/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.2320
- accuracy: 0.9242 - val_loss: 0.4313 - val_accuracy: 0.8755

Epoch 00005: val_accuracy did not improve from 0.88090
Epoch 6/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.1961
- accuracy: 0.9297 - val_loss: 0.3692 - val_accuracy: 0.8785

Epoch 00006: val_accuracy did not improve from 0.88090
Epoch 7/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.1701
- accuracy: 0.9396 - val_loss: 0.6784 - val_accuracy: 0.8429

Epoch 00007: val_accuracy did not improve from 0.88090
Epoch 8/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.1557
- accuracy: 0.9436 - val_loss: 0.3458 - val_accuracy: 0.9070

Epoch 00008: val_accuracy improved from 0.88090 to 0.90702, saving model t
o best_model_3.1
Epoch 9/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.1580
- accuracy: 0.9440 - val_loss: 0.2560 - val_accuracy: 0.9070

Epoch 00009: val_accuracy did not improve from 0.90702
Epoch 10/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.1599
- accuracy: 0.9436 - val_loss: 0.3452 - val_accuracy: 0.9060

Epoch 00010: val_accuracy did not improve from 0.90702
Epoch 11/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1501
- accuracy: 0.9396 - val_loss: 0.2587 - val_accuracy: 0.9148

Epoch 00011: val_accuracy improved from 0.90702 to 0.91483, saving model t
```

```
o best_model_3.1
Epoch 12/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.1288
- accuracy: 0.9459 - val_loss: 0.4389 - val_accuracy: 0.9030

Epoch 00012: val_accuracy did not improve from 0.91483
Epoch 13/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.1542
- accuracy: 0.9419 - val_loss: 0.3629 - val_accuracy: 0.9063

Epoch 00013: val_accuracy did not improve from 0.91483
Epoch 14/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.1491
- accuracy: 0.9450 - val_loss: 0.3270 - val_accuracy: 0.9152

Epoch 00014: val_accuracy improved from 0.91483 to 0.91517, saving model t
o best_model_3.1
Epoch 15/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.1336
- accuracy: 0.9501 - val_loss: 0.3026 - val_accuracy: 0.9264

Epoch 00015: val_accuracy improved from 0.91517 to 0.92637, saving model t
o best_model_3.1
Epoch 16/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1317
- accuracy: 0.9491 - val_loss: 0.3852 - val_accuracy: 0.9247

Epoch 00016: val_accuracy did not improve from 0.92637
Epoch 17/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.1296
- accuracy: 0.9505 - val_loss: 0.2898 - val_accuracy: 0.9192

Epoch 00017: val_accuracy did not improve from 0.92637
Epoch 18/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.1260
- accuracy: 0.9528 - val_loss: 0.3559 - val_accuracy: 0.9131

Epoch 00018: val_accuracy did not improve from 0.92637
Epoch 19/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1273
- accuracy: 0.9540 - val_loss: 0.3287 - val_accuracy: 0.8985

Epoch 00019: val_accuracy did not improve from 0.92637
Epoch 20/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1333
- accuracy: 0.9514 - val_loss: 0.3077 - val_accuracy: 0.9097

Epoch 00020: val_accuracy did not improve from 0.92637
```

Out[39]:

```
<keras.callbacks.callbacks.History at 0x2da15718fd0>
```

In [40]:

```
from keras.models import load_model
saved_model_2 = load_model('best_model_3.1')
```

In [41]:

```
# Confusion Matrix
print(confusion_matrix(Y_test, saved_model_2.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS
\
True
LAYING                 537        0         0        0                   0
SITTING                 24      387        77        0                   0
STANDING                 0       61       470        0                   0
WALKING                  0        0         0      454                  40
WALKING_DOWNSTAIRS       0        0         0        0                 419
WALKING_UPSTAIRS         0        1         0        1                   6

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            3
STANDING                           1
WALKING                            2
WALKING_DOWNSTAIRS                 1
WALKING_UPSTAIRS                 463
```

In [42]:

```
score = saved_model_2.evaluate(X_test, Y_test)
score
```

```
2947/2947 [==============================] - 14s 5ms/step
```

Out[42]:

```
[0.3025784575979611, 0.9263657927513123]
```

RESULTS

In [117]:

```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ['S.NO',"No. of Hidden Layers", "No.of lstm units", "Dropout", "TEST ACCURACY",'TEST LOSS']
x.add_row(["1","1",'32','0.25','0.8948','0.3549'])
x.add_row(["2","2",'32','0.75','0.6244','0.7772'])
x.add_row(["3","2",'20','0.5','0.9040','0.4683'])
x.add_row(["4","2",'24','0.5','0.9016','0.3812'])
x.add_row(["5","2",'28','0.5','0.7234','0.7294'])
x.add_row(["6","2",'28','0.75','0.5205','0.8599'])
x.add_row(["7","2",'28','0.25','0.9074','0.4082'])

print(x)
```

```
+------+----------------------+-----------------+---------+-------------
-+-----------+
| S.NO | No. of Hidden Layers | No.of lstm units | Dropout | TEST ACCURACY
| TEST LOSS |
+------+----------------------+-----------------+---------+-------------
-+-----------+
|  1   |          1           |        32        |   0.25  |     0.8948
|   0.3549  |
|  2   |          2           |        32        |   0.75  |     0.6244
|   0.7772  |
|  3   |          2           |        20        |   0.5   |     0.9040
|   0.4683  |
|  4   |          2           |        24        |   0.5   |     0.9016
|   0.3812  |
|  5   |          2           |        28        |   0.5   |     0.7234
|   0.7294  |
|  6   |          2           |        28        |   0.75  |     0.5205
|   0.8599  |
|  7   |          2           |        28        |   0.25  |     0.9074
|   0.4082  |
+------+----------------------+-----------------+---------+-------------
-+-----------+
```

CONCLUSION

With 1 hidden layer with 128 lstm units and dropout rate of 0.25 we got test accuracy of 93.45% and test loss of 0.2467. With 2 hidden layers with 128 and 64 lstm units and dropout rates of 0.2 and 0.5 respectively we got test accuracy of 92.63% and test loss of 0.3025.

In [ ]: