In [1]:

```
# Importing Libraries
```

In [2]:

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

In [3]:

```
# 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 [4]:

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

In [5]:

```
# 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 [6]:

```
# 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 [7]:

```
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 [8]:

```
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 [9]:

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

```
C:\Users\Santosh\Anaconda3\lib\site-packages\h5py\__init__.py:72: UserWarning: h5py is running
against HDF5 1.10.2 when it was built against 1.10.3, this may cause problems
  '{0}.{1}.{2}'.format(*version.hdf5_built_version_tuple)
```

In [10]:

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

In [11]:

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

```
Using TensorFlow backend.
```

In [12]:

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

In [13]:

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

In [14]:

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

In [15]:

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

```
C:\Users\Santosh\Anaconda3\lib\site-packages\ipykernel_launcher.py:12: FutureWarning: Method
.as_matrix will be removed in a future version. Use .values instead.
  if sys.path[0] == '':
```

In [16]:

```
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 [17]:

```
# 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()
```

```
WARNING:tensorflow:From C:\Users\Santosh\Anaconda3\lib\site-
packages\tensorflow\python\ops\resource_variable_ops.py:435: colocate_with (from
tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
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
```

In [18]:

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

In [19]:

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

```
WARNING:tensorflow:From C:\Users\Santosh\Anaconda3\lib\site-
packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 22s 3ms/step - loss: 1.2941 - accuracy: 0.4523 - val_
loss: 1.1074 - val_accuracy: 0.4924
Epoch 2/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.9406 - accuracy: 0.5964 - val_
loss: 0.8839 - val_accuracy: 0.5938
Epoch 3/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.7833 - accuracy: 0.6432 - val_
loss: 0.7483 - val_accuracy: 0.6125
Epoch 4/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.6809 - accuracy: 0.6619 - val_
loss: 0.7228 - val_accuracy: 0.6169
Epoch 5/30
7352/7352 [==============================] - 21s 3ms/step - loss: 0.6409 - accuracy: 0.6759 - val_
loss: 0.7215 - val_accuracy: 0.6328
Epoch 6/30
7352/7352 [==============================] - 21s 3ms/step - loss: 0.5755 - accuracy: 0.7025 - val_
loss: 0.7347 - val_accuracy: 0.6586
Epoch 7/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.5586 - accuracy: 0.7500 - val_
loss: 0.6474 - val_accuracy: 0.7441
Epoch 8/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.4938 - accuracy: 0.7784 - val_
loss: 0.7891 - val_accuracy: 0.7112
Epoch 9/30
7352/7352 [==============================] - 27s 4ms/step - loss: 0.4602 - accuracy: 0.7907 - val_
loss: 0.7612 - val_accuracy: 0.7177
Epoch 10/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.4374 - accuracy: 0.7922 - val_
loss: 0.6681 - val_accuracy: 0.7245
Epoch 11/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.4076 - accuracy: 0.8082 - val_
loss: 0.5656 - val_accuracy: 0.7540
Epoch 12/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.3870 - accuracy: 0.8371 - val_
loss: 0.5157 - val_accuracy: 0.8392
Epoch 13/30
7352/7352 [==============================] - 21s 3ms/step - loss: 0.3303 - accuracy: 0.8954 - val_
loss: 0.5163 - val_accuracy: 0.8782
Epoch 14/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.2876 - accuracy: 0.9210 - val_
loss: 0.5793 - val_accuracy: 0.8446
Epoch 15/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2383 - accuracy: 0.9291 - val_
loss: 0.3697 - val_accuracy: 0.8975
Epoch 16/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.2225 - accuracy: 0.9309 - val_
loss: 0.3675 - val_accuracy: 0.8924
Epoch 17/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2062 - accuracy: 0.9363 - val_
loss: 0.4045 - val_accuracy: 0.8999
```

```
loss: 0.4045 - val_accuracy: 0.8999
Epoch 18/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1936 - accuracy: 0.9411 - val_
loss: 0.6056 - val_accuracy: 0.8707
Epoch 19/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2009 - accuracy: 0.9408 - val_
loss: 0.7823 - val_accuracy: 0.8524
Epoch 20/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.2246 - accuracy: 0.9381 - val_
loss: 0.3956 - val_accuracy: 0.8996
Epoch 21/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1845 - accuracy: 0.9412 - val_
loss: 0.4611 - val_accuracy: 0.8918
Epoch 22/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1814 - accuracy: 0.9457 - val_
loss: 0.4755 - val_accuracy: 0.9013
Epoch 23/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.2044 - accuracy: 0.9389 - val_
loss: 0.4691 - val_accuracy: 0.8948
Epoch 24/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2306 - accuracy: 0.9384 - val_
loss: 0.5468 - val_accuracy: 0.8948
Epoch 25/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.2119 - accuracy: 0.9397 - val_
loss: 0.4385 - val_accuracy: 0.8928
Epoch 26/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1913 - accuracy: 0.9423 - val_
loss: 0.5487 - val_accuracy: 0.8843
Epoch 27/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1650 - accuracy: 0.9470 - val_
loss: 0.5596 - val_accuracy: 0.8965
Epoch 28/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1642 - accuracy: 0.9475 - val_
loss: 0.5737 - val_accuracy: 0.8968
Epoch 29/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1988 - accuracy: 0.9430 - val_
loss: 0.5129 - val_accuracy: 0.8935
Epoch 30/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1878 - accuracy: 0.9467 - val_
loss: 0.5882 - val_accuracy: 0.8951
```

Out[19]:

```
<keras.callbacks.callbacks.History at 0x1cae777c240>
```

In [20]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 510        0         0        0                   0
SITTING                  0      380       101        0                   0
STANDING                 0       75       457        0                   0
WALKING                  0        0         1      441                  35
WALKING_DOWNSTAIRS       0        0         0        0                 415
WALKING_UPSTAIRS         0        1         0        2                  33

Pred                WALKING_UPSTAIRS
True
LAYING                            27
SITTING                           10
STANDING                           0
WALKING                           19
WALKING_DOWNSTAIRS                 5
WALKING_UPSTAIRS                 435
```

In [21]:

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

```
2947/2947 [==============================] - 1s 472us/step
```

In [22]:

```
score
```

Out[22]:

```
[0.5882155524368978, 0.8951476216316223]
```

- 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

Model 1: 1 layer LSTM(38)+Dropout(0.5)

In [23]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(38, 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_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 38)                7296
_________________________________________________________________
dropout_2 (Dropout)          (None, 38)                0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 234
=================================================================
Total params: 7,530
Trainable params: 7,530
Non-trainable params: 0
_________________________________________________________________
```

In [24]:

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

In [25]:

```
# 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 [==============================] - 24s 3ms/step - loss: 1.2608 - accuracy: 0.4841 - val_
loss: 1.0873 - val_accuracy: 0.5331
Epoch 2/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.8848 - accuracy: 0.6276 - val_
loss: 0.8493 - val_accuracy: 0.6437
Epoch 3/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.7686 - accuracy: 0.6704 - val_
```

```
7352/7352 [==============================] - 23s 3ms/step - loss: 0.7686 - accuracy: 0.6704 - val_
loss: 0.8431 - val_accuracy: 0.6549
Epoch 4/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.6314 - accuracy: 0.7480 - val_
loss: 0.6157 - val_accuracy: 0.7251
Epoch 5/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.5177 - accuracy: 0.7865 - val_
loss: 0.5887 - val_accuracy: 0.7716
Epoch 6/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.4803 - accuracy: 0.8035 - val_
loss: 0.6082 - val_accuracy: 0.7682
Epoch 7/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.4648 - accuracy: 0.8055 - val_
loss: 0.5773 - val_accuracy: 0.7998
Epoch 8/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.4090 - accuracy: 0.8470 - val_
loss: 0.5187 - val_accuracy: 0.8541
Epoch 9/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.3426 - accuracy: 0.8889 - val_
loss: 0.3952 - val_accuracy: 0.8775
Epoch 10/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2641 - accuracy: 0.9223 - val_
loss: 0.8859 - val_accuracy: 0.7743
Epoch 11/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.2546 - accuracy: 0.9217 - val_
loss: 0.3682 - val_accuracy: 0.8802
Epoch 12/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.2263 - accuracy: 0.9295 - val_
loss: 0.3907 - val_accuracy: 0.8999
Epoch 13/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1955 - accuracy: 0.9369 - val_
loss: 0.4317 - val_accuracy: 0.9043
Epoch 14/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1912 - accuracy: 0.9387 - val_
loss: 0.4244 - val_accuracy: 0.9067
Epoch 15/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1758 - accuracy: 0.9403 - val_
loss: 0.3278 - val_accuracy: 0.9006
Epoch 16/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1889 - accuracy: 0.9377 - val_
loss: 0.2627 - val_accuracy: 0.9111
Epoch 17/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1805 - accuracy: 0.9385 - val_
loss: 0.3675 - val_accuracy: 0.9040
Epoch 18/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1674 - accuracy: 0.9419 - val_
loss: 0.4200 - val_accuracy: 0.9074
Epoch 19/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1662 - accuracy: 0.9449 - val_
loss: 0.4317 - val_accuracy: 0.9057
Epoch 20/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1699 - accuracy: 0.9416 - val_
loss: 0.2859 - val_accuracy: 0.9131
Epoch 21/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1601 - accuracy: 0.9465 - val_
loss: 0.4015 - val_accuracy: 0.9131
Epoch 22/30
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1657 - accuracy: 0.9427 - val_
loss: 0.4350 - val_accuracy: 0.8877
Epoch 23/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.1522 - accuracy: 0.9463 - val_
loss: 0.3205 - val_accuracy: 0.9040
Epoch 24/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1767 - accuracy: 0.9387 - val_
loss: 0.2447 - val_accuracy: 0.9141
Epoch 25/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1647 - accuracy: 0.9421 - val_
loss: 0.3080 - val_accuracy: 0.9030
Epoch 26/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1663 - accuracy: 0.9457 - val_
loss: 0.2910 - val_accuracy: 0.9192
Epoch 27/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1452 - accuracy: 0.9483 - val_
loss: 0.3577 - val_accuracy: 0.9128
Epoch 28/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1557 - accuracy: 0.9480 - val_
loss: 0.2914 - val_accuracy: 0.9162
Epoch 29/30
```

```
Epoch 29/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1429 - accuracy: 0.9470 - val_
loss: 0.2325 - val_accuracy: 0.9250
Epoch 30/30
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1947 - accuracy: 0.9427 - val_
loss: 0.2568 - val_accuracy: 0.9247
```

Out[25]:

```
<keras.callbacks.callbacks.History at 0x1caefc2de10>
```

In [26]:

```
# 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      443        36        1                   0
STANDING                 0      140       388        4                   0
WALKING                  0        0         0      478                   4
WALKING_DOWNSTAIRS       0        0         0        1                 419
WALKING_UPSTAIRS         0        4         0        4                   3

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                           11
STANDING                           0
WALKING                           14
WALKING_DOWNSTAIRS                 0
WALKING_UPSTAIRS                 460
```

In [27]:

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

```
2947/2947 [==============================] - 1s 420us/step
```

In [28]:

```
score
```

Out[28]:

```
[0.25677269414946985, 0.9246691465377808]
```

Module -2: 2- LSTM(55)+BN+Dropout(0.8)

In [29]:

```
from keras.regularizers import L1L2
from keras.layers import LSTM , BatchNormalization
reg = L1L2(0.01, 0.01)

model = Sequential()
model.add(LSTM(55, input_shape=(timesteps, input_dim), kernel_initializer='glorot_normal' ,
return_sequences=True, bias_regularizer=reg))
model.add(BatchNormalization())
model.add(Dropout(0.8))
model.add(LSTM(30))
model.add(Dropout(0.8))
model.add(Dense(n_classes, activation='sigmoid'))
print("Model Summary: ")
model.summary()
```

```
Model Summary:
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 128, 55)           14300
_________________________________________________________________
batch_normalization_1 (Batch (None, 128, 55)           220
_________________________________________________________________
dropout_3 (Dropout)          (None, 128, 55)           0
_________________________________________________________________
lstm_4 (LSTM)                (None, 30)                10320
_________________________________________________________________
dropout_4 (Dropout)          (None, 30)                0
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 186
=================================================================
Total params: 25,026
Trainable params: 24,916
Non-trainable params: 110
_________________________________________________________________
```

In [30]:

```
# Compiling the model

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

In [32]:

```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=25)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/25
7352/7352 [==============================] - 61s 8ms/step - loss: 2.1787 - accuracy: 0.4395 - val_
loss: 1.6111 - val_accuracy: 0.4897
Epoch 2/25
7352/7352 [==============================] - 58s 8ms/step - loss: 1.3174 - accuracy: 0.5528 - val_
loss: 1.5537 - val_accuracy: 0.4503
Epoch 3/25
7352/7352 [==============================] - 57s 8ms/step - loss: 0.9550 - accuracy: 0.5639 - val_
loss: 1.0835 - val_accuracy: 0.5979
Epoch 4/25
7352/7352 [==============================] - 56s 8ms/step - loss: 0.8928 - accuracy: 0.5876 - val_
loss: 1.0230 - val_accuracy: 0.6088
Epoch 5/25
7352/7352 [==============================] - 55s 7ms/step - loss: 0.8599 - accuracy: 0.6132 - val_
loss: 0.8395 - val_accuracy: 0.6145
Epoch 6/25
7352/7352 [==============================] - 54s 7ms/step - loss: 0.8306 - accuracy: 0.6194 - val_
loss: 0.7006 - val_accuracy: 0.6593
Epoch 7/25
7352/7352 [==============================] - 54s 7ms/step - loss: 0.8255 - accuracy: 0.6421 - val_
loss: 0.7505 - val_accuracy: 0.6563
Epoch 8/25
7352/7352 [==============================] - 54s 7ms/step - loss: 0.7836 - accuracy: 0.6533 - val_
loss: 0.7317 - val_accuracy: 0.6736
Epoch 9/25
7352/7352 [==============================] - 54s 7ms/step - loss: 0.7626 - accuracy: 0.6854 - val_
loss: 0.6883 - val_accuracy: 0.7547
Epoch 10/25
7352/7352 [==============================] - 54s 7ms/step - loss: 0.7200 - accuracy: 0.6945 - val_
loss: 0.5191 - val_accuracy: 0.8507
Epoch 11/25
7352/7352 [==============================] - 54s 7ms/step - loss: 0.6900 - accuracy: 0.7088 - val_
loss: 0.5241 - val_accuracy: 0.8361
Epoch 12/25
7352/7352 [==============================] - 53s 7ms/step - loss: 0.6641 - accuracy: 0.7371 - val_
```

```
loss: 0.5242 - val_accuracy: 0.8239
Epoch 13/25
7352/7352 [==============================] - 54s 7ms/step - loss: 0.6381 - accuracy: 0.7416 - val_
loss: 0.5214 - val_accuracy: 0.8432
Epoch 14/25
7352/7352 [==============================] - 56s 8ms/step - loss: 0.6336 - accuracy: 0.7693 - val_
loss: 0.4916 - val_accuracy: 0.8527
Epoch 15/25
7352/7352 [==============================] - 54s 7ms/step - loss: 0.6046 - accuracy: 0.7809 - val_
loss: 0.7898 - val_accuracy: 0.7971
Epoch 16/25
7352/7352 [==============================] - 57s 8ms/step - loss: 0.6003 - accuracy: 0.7979 - val_
loss: 0.5996 - val_accuracy: 0.8378
Epoch 17/25
7352/7352 [==============================] - 54s 7ms/step - loss: 0.5511 - accuracy: 0.8149 - val_
loss: 0.3288 - val_accuracy: 0.9040
Epoch 18/25
7352/7352 [==============================] - 55s 7ms/step - loss: 0.5314 - accuracy: 0.8236 - val_
loss: 0.4131 - val_accuracy: 0.8782
Epoch 19/25
7352/7352 [==============================] - 55s 7ms/step - loss: 0.5183 - accuracy: 0.8311 - val_
loss: 0.3343 - val_accuracy: 0.9060
Epoch 20/25
7352/7352 [==============================] - 55s 7ms/step - loss: 0.4856 - accuracy: 0.8409 - val_
loss: 0.2938 - val_accuracy: 0.9050
Epoch 21/25
7352/7352 [==============================] - 55s 8ms/step - loss: 0.4835 - accuracy: 0.8462 - val_
loss: 0.3486 - val_accuracy: 0.9036
Epoch 22/25
7352/7352 [==============================] - 53s 7ms/step - loss: 0.4665 - accuracy: 0.8492 - val_
loss: 0.3833 - val_accuracy: 0.9016
Epoch 23/25
7352/7352 [==============================] - 53s 7ms/step - loss: 0.4548 - accuracy: 0.8568 - val_
loss: 0.4510 - val_accuracy: 0.8928
Epoch 24/25
7352/7352 [==============================] - 53s 7ms/step - loss: 0.4508 - accuracy: 0.8577 - val_
loss: 0.4406 - val_accuracy: 0.8931
Epoch 25/25
7352/7352 [==============================] - 54s 7ms/step - loss: 0.4178 - accuracy: 0.8704 - val_
loss: 0.3928 - val_accuracy: 0.8982
```

Out[32]:

```
<keras.callbacks.callbacks.History at 0x1caf1e86978>
```

In [33]:

```
# Confusion Matrix

print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 510        0        20        0                   6
SITTING                  0      424        65        0                   0
STANDING                 0      134       397        0                   0
WALKING                  0        3         0      466                  26
WALKING_DOWNSTAIRS       0        0         0        0                 420
WALKING_UPSTAIRS         0        0         4        2                  35

Pred                WALKING_UPSTAIRS
True
LAYING                             1
SITTING                            2
STANDING                           1
WALKING                            1
WALKING_DOWNSTAIRS                 0
WALKING_UPSTAIRS                 430
```

In [34]:

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

```
2947/2947 [==============================] - 3s 1ms/step
```

2947/2947 [==============================] - 38 1ms/step

In [35]:

```
score
```

Out[35]:

```
[0.3928260555393516, 0.898201584815979]
```

Model 3: 2 LSTM(70,35)+Dropout(0.6)

In [40]:

```
model = Sequential()
model.add(LSTM(70, input_shape=(timesteps, input_dim), kernel_initializer='glorot_normal' ,
return_sequences=True, bias_regularizer=reg))
model.add(Dropout(0.6))
model.add(LSTM(35))
model.add(Dropout(0.6))
model.add(Dense(n_classes, activation='sigmoid'))
print("Model Summary: ")
model.summary()
```

```
Model Summary:
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_17 (LSTM)               (None, 128, 70)           22400
_________________________________________________________________
dropout_13 (Dropout)         (None, 128, 70)           0
_________________________________________________________________
lstm_18 (LSTM)               (None, 35)                14840
_________________________________________________________________
dropout_14 (Dropout)         (None, 35)                0
_________________________________________________________________
dense_4 (Dense)              (None, 6)                 216
=================================================================
Total params: 37,456
Trainable params: 37,456
Non-trainable params: 0
_________________________________________________________________
```

In [41]:

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

In [42]:

```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=25)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/25
7352/7352 [==============================] - 66s 9ms/step - loss: 2.2385 - accuracy: 0.4835 - val_
loss: 1.6759 - val_accuracy: 0.5358
Epoch 2/25
7352/7352 [==============================] - 65s 9ms/step - loss: 1.3105 - accuracy: 0.5910 - val_
loss: 1.0784 - val_accuracy: 0.6010
Epoch 3/25
7352/7352 [==============================] - 65s 9ms/step - loss: 0.8694 - accuracy: 0.6291 - val_
loss: 0.9420 - val_accuracy: 0.6060
Epoch 4/25
```

```
Epoch 4/25
7352/7352 [==============================] - 62s 8ms/step - loss: 0.7610 - accuracy: 0.6400 - val_
loss: 0.8358 - val_accuracy: 0.6172
Epoch 5/25
7352/7352 [==============================] - 62s 8ms/step - loss: 0.7239 - accuracy: 0.6495 - val_
loss: 0.8580 - val_accuracy: 0.6220
Epoch 6/25
7352/7352 [==============================] - 60s 8ms/step - loss: 0.7325 - accuracy: 0.6507 - val_
loss: 0.8955 - val_accuracy: 0.6128
Epoch 7/25
7352/7352 [==============================] - 70s 10ms/step - loss: 0.7106 - accuracy: 0.6496 - val
_loss: 0.8542 - val_accuracy: 0.6508
Epoch 8/25
7352/7352 [==============================] - 61s 8ms/step - loss: 0.6770 - accuracy: 0.6785 - val_
loss: 0.8461 - val_accuracy: 0.6576
Epoch 9/25
7352/7352 [==============================] - 62s 8ms/step - loss: 0.6676 - accuracy: 0.6844 - val_
loss: 0.8827 - val_accuracy: 0.6356
Epoch 10/25
7352/7352 [==============================] - 61s 8ms/step - loss: 0.6842 - accuracy: 0.6907 - val_
loss: 1.1169 - val_accuracy: 0.6478
Epoch 11/25
7352/7352 [==============================] - 62s 8ms/step - loss: 0.5922 - accuracy: 0.7390 - val_
loss: 0.9562 - val_accuracy: 0.6698
Epoch 12/25
7352/7352 [==============================] - 61s 8ms/step - loss: 0.5262 - accuracy: 0.7647 - val_
loss: 0.7163 - val_accuracy: 0.7414
Epoch 13/25
7352/7352 [==============================] - 62s 8ms/step - loss: 0.4625 - accuracy: 0.8062 - val_
loss: 0.7867 - val_accuracy: 0.7954
Epoch 14/25
7352/7352 [==============================] - 62s 8ms/step - loss: 0.4308 - accuracy: 0.8364 - val_
loss: 0.7427 - val_accuracy: 0.8001
Epoch 15/25
7352/7352 [==============================] - 65s 9ms/step - loss: 0.3738 - accuracy: 0.8647 - val_
loss: 0.5351 - val_accuracy: 0.8656
Epoch 16/25
7352/7352 [==============================] - 61s 8ms/step - loss: 0.3081 - accuracy: 0.9087 - val_
loss: 0.5187 - val_accuracy: 0.8714
Epoch 17/25
7352/7352 [==============================] - 61s 8ms/step - loss: 0.2606 - accuracy: 0.9211 - val_
loss: 0.4039 - val_accuracy: 0.8839
Epoch 18/25
7352/7352 [==============================] - 61s 8ms/step - loss: 0.2349 - accuracy: 0.9241 - val_
loss: 0.4407 - val_accuracy: 0.8918
Epoch 19/25
7352/7352 [==============================] - 61s 8ms/step - loss: 0.2160 - accuracy: 0.9351 - val_
loss: 0.5404 - val_accuracy: 0.8853
Epoch 20/25
7352/7352 [==============================] - 61s 8ms/step - loss: 0.1879 - accuracy: 0.9395 - val_
loss: 0.5459 - val_accuracy: 0.8867
Epoch 21/25
7352/7352 [==============================] - 66s 9ms/step - loss: 0.1827 - accuracy: 0.9440 - val_
loss: 0.5079 - val_accuracy: 0.8863
Epoch 22/25
7352/7352 [==============================] - 68s 9ms/step - loss: 0.1806 - accuracy: 0.9426 - val_
loss: 0.5243 - val_accuracy: 0.9053
Epoch 23/25
7352/7352 [==============================] - 62s 8ms/step - loss: 0.1626 - accuracy: 0.9480 - val_
loss: 0.4818 - val_accuracy: 0.8958
Epoch 24/25
7352/7352 [==============================] - 65s 9ms/step - loss: 0.1746 - accuracy: 0.9479 - val_
loss: 0.5969 - val_accuracy: 0.8877
Epoch 25/25
7352/7352 [==============================] - 64s 9ms/step - loss: 0.1550 - accuracy: 0.9478 - val_
loss: 0.6773 - val_accuracy: 0.8884
```

Out[42]:

```
<keras.callbacks.callbacks.History at 0x1caf7ae8128>
```

In [43]:

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

```
2947/2947 [                                  ]  - 4s 1ms/step
```

```
2947/2947 [==============================] - 4s 1ms/step
```

In [44]:

```
score
```

Out[44]:

```
[0.6772659795652995, 0.8883610367774963]
```

model 4: 2 LSTM(120,60)+BN

In [46]:

```
model = Sequential()
model.add(LSTM(120, input_shape=(timesteps, input_dim), kernel_initializer='glorot_normal' , return
_sequences=True, bias_regularizer=reg))
model.add(BatchNormalization())
model.add(LSTM(60))
model.add(BatchNormalization())
model.add(Dense(n_classes, activation='sigmoid'))
print("Model Summary: ")
model.summary()
```

```
Model Summary:
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_19 (LSTM)               (None, 128, 120)          62400
_________________________________________________________________
batch_normalization_2 (Batch (None, 128, 120)          480
_________________________________________________________________
lstm_20 (LSTM)               (None, 60)                43440
_________________________________________________________________
batch_normalization_3 (Batch (None, 60)                240
_________________________________________________________________
dense_5 (Dense)              (None, 6)                 366
=================================================================
Total params: 106,926
Trainable params: 106,566
Non-trainable params: 360
_________________________________________________________________
```

In [47]:

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

In [48]:

```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=15)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/15
7352/7352 [==============================] - 96s 13ms/step - loss: 1.8490 - accuracy: 0.9329 - val
_loss: 1.1441 - val_accuracy: 0.9339
Epoch 2/15
7352/7352 [==============================] - 95s 13ms/step - loss: 0.5745 - accuracy: 0.9769 - val
_loss: 0.1965 - val_accuracy: 0.9650
Epoch 3/15
7352/7352 [==============================] - 94s 13ms/step - loss: 0.0696 - accuracy: 0.9783 - val
_loss: 0.1029 - val_accuracy: 0.9524
```

```
Epoch 4/15
7352/7352 [==============================] - 96s 13ms/step - loss: 0.0568 - accuracy: 0.9793 - val
_loss: 0.1121 - val_accuracy: 0.9656
Epoch 5/15
7352/7352 [==============================] - 101s 14ms/step - loss: 0.0537 - accuracy: 0.9797 - va
l_loss: 0.1520 - val_accuracy: 0.9395
Epoch 6/15
7352/7352 [==============================] - 100s 14ms/step - loss: 0.0514 - accuracy: 0.9805 - va
l_loss: 0.1512 - val_accuracy: 0.9604
Epoch 7/15
7352/7352 [==============================] - 104s 14ms/step - loss: 0.0516 - accuracy: 0.9812 - va
l_loss: 0.0818 - val_accuracy: 0.9733
Epoch 8/15
7352/7352 [==============================] - 95s 13ms/step - loss: 0.0506 - accuracy: 0.9817 - val
_loss: 0.0669 - val_accuracy: 0.9751
Epoch 9/15
7352/7352 [==============================] - 96s 13ms/step - loss: 0.0486 - accuracy: 0.9814 - val
_loss: 0.0923 - val_accuracy: 0.9690
Epoch 10/15
7352/7352 [==============================] - 93s 13ms/step - loss: 0.0485 - accuracy: 0.9819 - val
_loss: 0.0933 - val_accuracy: 0.9714
Epoch 11/15
7352/7352 [==============================] - 104s 14ms/step - loss: 0.0491 - accuracy: 0.9816 - va
l_loss: 0.0816 - val_accuracy: 0.9718
Epoch 12/15
7352/7352 [==============================] - 95s 13ms/step - loss: 0.0471 - accuracy: 0.9824 - val
_loss: 0.0835 - val_accuracy: 0.9719
Epoch 13/15
7352/7352 [==============================] - 96s 13ms/step - loss: 0.0493 - accuracy: 0.9820 - val
_loss: 0.1110 - val_accuracy: 0.9693
Epoch 14/15
7352/7352 [==============================] - 97s 13ms/step - loss: 0.0466 - accuracy: 0.9825 - val
_loss: 0.0778 - val_accuracy: 0.9751
Epoch 15/15
7352/7352 [==============================] - 95s 13ms/step - loss: 0.0463 - accuracy: 0.9825 - val
_loss: 0.0790 - val_accuracy: 0.9744
```

Out[48]:

```
<keras.callbacks.callbacks.History at 0x1caf865c048>
```

In [49]:

```
# 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                  6      383        99        0                   0
STANDING                 0      108       424        0                   0
WALKING                  0        0         0      494                   1
WALKING_DOWNSTAIRS       0        0         0        0                 418
WALKING_UPSTAIRS         0        5         0        1                  19

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            3
STANDING                           0
WALKING                            1
WALKING_DOWNSTAIRS                 2
WALKING_UPSTAIRS                 446
```

In [50]:

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

```
2947/2947 [==============================] - 7s 2ms/step
```

In [51]:

```
score
```

Out[51]:

```
[0.07902003169861822, 0.9744372367858887]
```

model-5: 2 LSTM(90,45)+BN+Dropout(0.6)

In [83]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(90, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.7))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_18"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_32 (LSTM)               (None, 90)                36000
_________________________________________________________________
dropout_20 (Dropout)         (None, 90)                0
_________________________________________________________________
dense_14 (Dense)             (None, 6)                 546
=================================================================
Total params: 36,546
Trainable params: 36,546
Non-trainable params: 0
_________________________________________________________________
```

In [84]:

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

In [85]:

```
# 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 [==============================] - 166s 23ms/step - loss: 1.2887 - accuracy: 0.4429 - va
l_loss: 1.3044 - val_accuracy: 0.4754
Epoch 2/30
7352/7352 [==============================] - 154s 21ms/step - loss: 0.9033 - accuracy: 0.5966 - va
l_loss: 0.8463 - val_accuracy: 0.5857
Epoch 3/30
7352/7352 [==============================] - 159s 22ms/step - loss: 0.7381 - accuracy: 0.6515 - va
l_loss: 0.7199 - val_accuracy: 0.6105
Epoch 4/30
7352/7352 [==============================] - 204s 28ms/step - loss: 0.6682 - accuracy: 0.7005 - va
l_loss: 0.6860 - val_accuracy: 0.7065
Epoch 5/30
7352/7352 [==============================] - 291s 40ms/step - loss: 0.5849 - accuracy: 0.7782 - va
l_loss: 0.5954 - val_accuracy: 0.7910
Epoch 6/30
7352/7352 [==============================] - 276s 38ms/step - loss: 0.5090 - accuracy: 0.8277 - va
l_loss: 0.6676 - val_accuracy: 0.6651
Epoch 7/30
7352/7352 [==============================] - 274s 37ms/step - loss: 0.3933 - accuracy: 0.8855 - va
```

```
7352/7352 [==============================] - 274s 37ms/step - loss: 0.3933 - accuracy: 0.8655 - va
l_loss: 0.3782 - val_accuracy: 0.8578
Epoch 8/30
7352/7352 [==============================] - 306s 42ms/step - loss: 0.2963 - accuracy: 0.9117 - va
l_loss: 0.4029 - val_accuracy: 0.9030
Epoch 9/30
7352/7352 [==============================] - 282s 38ms/step - loss: 0.2917 - accuracy: 0.9113 - va
l_loss: 0.3223 - val_accuracy: 0.8744
Epoch 10/30
7352/7352 [==============================] - 171s 23ms/step - loss: 0.2523 - accuracy: 0.9266 - va
l_loss: 0.5067 - val_accuracy: 0.8772
Epoch 11/30
7352/7352 [==============================] - 207s 28ms/step - loss: 0.2421 - accuracy: 0.9280 - va
l_loss: 0.2874 - val_accuracy: 0.9053
Epoch 12/30
7352/7352 [==============================] - 238s 32ms/step - loss: 0.2170 - accuracy: 0.9342 - va
l_loss: 0.2411 - val_accuracy: 0.9162
Epoch 13/30
7352/7352 [==============================] - 182s 25ms/step - loss: 0.2025 - accuracy: 0.9393 - va
l_loss: 0.3195 - val_accuracy: 0.9169
Epoch 14/30
7352/7352 [==============================] - 188s 26ms/step - loss: 0.2021 - accuracy: 0.9365 - va
l_loss: 0.2639 - val_accuracy: 0.9230
Epoch 15/30
7352/7352 [==============================] - 183s 25ms/step - loss: 0.2358 - accuracy: 0.9306 - va
l_loss: 0.3108 - val_accuracy: 0.9023
Epoch 16/30
7352/7352 [==============================] - 183s 25ms/step - loss: 0.1814 - accuracy: 0.9421 - va
l_loss: 0.3025 - val_accuracy: 0.9175
Epoch 17/30
7352/7352 [==============================] - 186s 25ms/step - loss: 0.1823 - accuracy: 0.9393 - va
l_loss: 0.3004 - val_accuracy: 0.9036
Epoch 18/30
7352/7352 [==============================] - 185s 25ms/step - loss: 0.1694 - accuracy: 0.9445 - va
l_loss: 0.3888 - val_accuracy: 0.9070
Epoch 19/30
7352/7352 [==============================] - 202s 27ms/step - loss: 0.1803 - accuracy: 0.9408 - va
l_loss: 0.2800 - val_accuracy: 0.9118
Epoch 20/30
7352/7352 [==============================] - 228s 31ms/step - loss: 0.1631 - accuracy: 0.9448 - va
l_loss: 0.3290 - val_accuracy: 0.9084
Epoch 21/30
7352/7352 [==============================] - 241s 33ms/step - loss: 0.1841 - accuracy: 0.9406 - va
l_loss: 0.3720 - val_accuracy: 0.9097
Epoch 22/30
7352/7352 [==============================] - 198s 27ms/step - loss: 0.1584 - accuracy: 0.9470 - va
l_loss: 0.3928 - val_accuracy: 0.9135
Epoch 23/30
7352/7352 [==============================] - 196s 27ms/step - loss: 0.1716 - accuracy: 0.9414 - va
l_loss: 0.3335 - val_accuracy: 0.9094
Epoch 24/30
7352/7352 [==============================] - 199s 27ms/step - loss: 0.1612 - accuracy: 0.9445 - va
l_loss: 0.3326 - val_accuracy: 0.9182
Epoch 25/30
7352/7352 [==============================] - 176s 24ms/step - loss: 0.1533 - accuracy: 0.9472 - va
l_loss: 0.3555 - val_accuracy: 0.9074
Epoch 26/30
7352/7352 [==============================] - 180s 24ms/step - loss: 0.1462 - accuracy: 0.9498 - va
l_loss: 0.4300 - val_accuracy: 0.9094
Epoch 27/30
7352/7352 [==============================] - 180s 24ms/step - loss: 0.2310 - accuracy: 0.9406 - va
l_loss: 0.4702 - val_accuracy: 0.9162
Epoch 28/30
7352/7352 [==============================] - 179s 24ms/step - loss: 0.1475 - accuracy: 0.9467 - va
l_loss: 0.2552 - val_accuracy: 0.9155
Epoch 29/30
7352/7352 [==============================] - 164s 22ms/step - loss: 0.1756 - accuracy: 0.9486 - va
l_loss: 0.5594 - val_accuracy: 0.8968
Epoch 30/30
7352/7352 [==============================] - 162s 22ms/step - loss: 0.1704 - accuracy: 0.9484 - va
l_loss: 0.4978 - val_accuracy: 0.9108
```

Out[85]:

```
<keras.callbacks.callbacks.History at 0x1ca8e4c4c50>
```

In [86]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 510        0        27        0                   0
SITTING                  0      392        96        2                   0
STANDING                 0       67       456        9                   0
WALKING                  0        0         0      473                   8
WALKING_DOWNSTAIRS       0        0         0       11                 406
WALKING_UPSTAIRS         0        1         0       10                  13

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            1
STANDING                           0
WALKING                           15
WALKING_DOWNSTAIRS                 3
WALKING_UPSTAIRS                 447
```

In [87]:

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

```
2947/2947 [==============================] - 10s 3ms/step
```

In [88]:

```
score
```

Out[88]:

```
[0.4977966246420733, 0.9107567071914673]
```

model-6:LSTM(90)+BN+Dropout(0.2)+LSTM(45)+BN+Dropout(0.2)

In [59]:

```
model = Sequential()
model.add(LSTM(90, input_shape=(timesteps, input_dim), kernel_initializer='glorot_normal' ,
return_sequences=True, bias_regularizer=reg))
model.add(BatchNormalization())
model.add(Dropout(0.2))
model.add(LSTM(45))
model.add(BatchNormalization())
model.add(Dropout(0.2))
model.add(Dense(n_classes, activation='sigmoid'))
print("Model Summary: ")
model.summary()
```

```
Model Summary:
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_23 (LSTM)               (None, 128, 90)           36000
_________________________________________________________________
batch_normalization_5 (Batch (None, 128, 90)           360
_________________________________________________________________
dropout_16 (Dropout)         (None, 128, 90)           0
_________________________________________________________________
lstm_24 (LSTM)               (None, 45)                24480
_________________________________________________________________
batch_normalization_6 (Batch (None, 45)                180
_________________________________________________________________
dropout_17 (Dropout)         (None, 45)                0
_________________________________________________________________
```

```
dense_7 (Dense)              (None, 6)                 276
=================================================================
Total params: 61,296
Trainable params: 61,026
Non-trainable params: 270
_________________________________________________________________
```

In [60]:

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

In [61]:

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

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/20
7352/7352 [==============================] - 94s 13ms/step - loss: 1.5275 - accuracy: 0.8838 - val
_loss: 1.0169 - val_accuracy: 0.8931
Epoch 2/20
7352/7352 [==============================] - 92s 13ms/step - loss: 0.4774 - accuracy: 0.9718 - val
_loss: 0.1813 - val_accuracy: 0.9706
Epoch 3/20
7352/7352 [==============================] - 92s 12ms/step - loss: 0.0760 - accuracy: 0.9772 - val
_loss: 0.0883 - val_accuracy: 0.9716
Epoch 4/20
7352/7352 [==============================] - 92s 12ms/step - loss: 0.0614 - accuracy: 0.9785 - val
_loss: 0.0758 - val_accuracy: 0.9730
Epoch 5/20
7352/7352 [==============================] - 92s 12ms/step - loss: 0.0595 - accuracy: 0.9784 - val
_loss: 0.1108 - val_accuracy: 0.9709
Epoch 6/20
7352/7352 [==============================] - 92s 13ms/step - loss: 0.0581 - accuracy: 0.9784 - val
_loss: 0.0693 - val_accuracy: 0.9793
Epoch 7/20
7352/7352 [==============================] - 91s 12ms/step - loss: 0.0578 - accuracy: 0.9788 - val
_loss: 0.0984 - val_accuracy: 0.9691
Epoch 8/20
7352/7352 [==============================] - 92s 12ms/step - loss: 0.0545 - accuracy: 0.9812 - val
_loss: 0.0861 - val_accuracy: 0.9738
Epoch 9/20
7352/7352 [==============================] - 92s 13ms/step - loss: 0.0508 - accuracy: 0.9811 - val
_loss: 0.1134 - val_accuracy: 0.9697
Epoch 10/20
7352/7352 [==============================] - 93s 13ms/step - loss: 0.0538 - accuracy: 0.9806 - val
_loss: 0.1183 - val_accuracy: 0.9710
Epoch 11/20
7352/7352 [==============================] - 92s 12ms/step - loss: 0.0541 - accuracy: 0.9803 - val
_loss: 0.1059 - val_accuracy: 0.9699
Epoch 12/20
7352/7352 [==============================] - 91s 12ms/step - loss: 0.0502 - accuracy: 0.9806 - val
_loss: 0.1324 - val_accuracy: 0.9699
Epoch 13/20
7352/7352 [==============================] - 92s 12ms/step - loss: 0.0507 - accuracy: 0.9818 - val
_loss: 0.1259 - val_accuracy: 0.9666
Epoch 14/20
7352/7352 [==============================] - 91s 12ms/step - loss: 0.0518 - accuracy: 0.9813 - val
_loss: 0.1268 - val_accuracy: 0.9718
Epoch 15/20
7352/7352 [==============================] - 298s 41ms/step - loss: 0.0507 - accuracy: 0.9817 - va
l_loss: 0.1339 - val_accuracy: 0.9732
Epoch 16/20
7352/7352 [==============================] - 301s 41ms/step - loss: 0.0492 - accuracy: 0.9827 - va
l_loss: 0.0941 - val_accuracy: 0.9750
Epoch 17/20
7352/7352 [==============================] - 300s 41ms/step - loss: 0.0507 - accuracy: 0.9816 - va
l_loss: 0.1061 - val_accuracy: 0.9742
```

```
Epoch 18/20
7352/7352 [==============================] - 4238s 576ms/step - loss: 0.0489 - accuracy: 0.9823 -
val_loss: 0.1266 - val_accuracy: 0.9703
Epoch 19/20
7352/7352 [==============================] - 289s 39ms/step - loss: 0.0513 - accuracy: 0.9823 - va
l_loss: 0.0920 - val_accuracy: 0.9768
Epoch 20/20
7352/7352 [==============================] - 289s 39ms/step - loss: 0.0508 - accuracy: 0.9815 - va
l_loss: 0.1068 - val_accuracy: 0.9760
```

Out[61]:

```
<keras.callbacks.callbacks.History at 0x1ca86434ba8>
```

In [62]:

```
# 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      408        80        0                   0
STANDING                 0       81       451        0                   0
WALKING                  0        0         0      491                   0
WALKING_DOWNSTAIRS       0        0         0       12                 396
WALKING_UPSTAIRS         0        0         1        7                  12

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            3
STANDING                           0
WALKING                            5
WALKING_DOWNSTAIRS                12
WALKING_UPSTAIRS                 451
```

In [63]:

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

```
2947/2947 [==============================] - 20s 7ms/step
```

In [64]:

```
score
```

Out[64]:

```
[0.10677718073892851, 0.9760208129882812]
```

Conclusion

In [90]:

```
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Model","LSTM Layers","BN","Dropout", "Test loss", "Test Accuracy"]

x.add_row(["1","1 Layer of LSTM(38)","No","Yes(0.5)","0.25677269414946985", "0.9246691465377808"])

x.add_row(["2","2 layers of LSTM(55,30)", "Yes", "Yes(0.8)","0.3928260555393516",
"0.898201584815979"])

x.add_row(["3","2 Layers of LSTM(70,35)","No","Yes(0.6)","0.6772659795652995",
"0.8883610367774963"])
```

```
x.add_row(["4","2 Layer of LSTM(120,60)","Yes","No","0.07902003169861822", "0.9744372367858887"])

x.add_row(["5","1 layer of LSTM(90)","No","Yes(0.7)", "0.4977966246420733","0.9107567071914673"])

x.add_row(["6","2 layers of LSTM(90,45)", "Yes", "Yes(0.2)","0.10677718073892851",
"0.9760208129882812"])

print(x)
```

```
+-------+-------------------------+-----+----------+---------------------+--------------------+
| Model |       LSTM Layers       |  BN | Dropout  |      Test loss      |   Test Accuracy    |
+-------+-------------------------+-----+----------+---------------------+--------------------+
|   1   |   1 Layer of LSTM(38)   |  No | Yes(0.5) | 0.25677269414946985 | 0.9246691465377808 |
|   2   | 2 layers of LSTM(55,30) | Yes | Yes(0.8) |  0.3928260555393516 | 0.898201584815979  |
|   3   | 2 Layers of LSTM(70,35) |  No | Yes(0.6) |  0.6772659795652995 | 0.8883610367774963 |
|   4   | 2 Layer of LSTM(120,60) | Yes |    No    | 0.07902003169861822 | 0.9744372367858887 |
|   5   |   1 layer of LSTM(90)   |  No | Yes(0.7) |  0.4977966246420733 | 0.9107567071914673 |
|   6   | 2 layers of LSTM(90,45) | Yes | Yes(0.2) | 0.10677718073892851 | 0.9760208129882812 |
+-------+-------------------------+-----+----------+---------------------+--------------------+
```

In [ ]:

```
```