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)
```

```
/Users/bhawesh/anaconda3/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:526: Fu
tureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version
of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/Users/bhawesh/anaconda3/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:527: Fu
tureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version
of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/Users/bhawesh/anaconda3/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:528: Fu
tureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version
of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/Users/bhawesh/anaconda3/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:529: Fu
tureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version
of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/Users/bhawesh/anaconda3/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:530: Fu
tureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version
of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/Users/bhawesh/anaconda3/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:535: Fu
tureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version
of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

In [10]:

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

```
    inter_op_parallelism_threads=1
)
```

In [11]:

```
# Import Keras
import os
os.environ['KMP_DUPLICATE_LIB_OK']='True'
from tensorflow.keras import backend as K
sess = tf.Session(graph=tf.get_default_graph(), config=session_conf)
K.set_session(sess)
```

In [12]:

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

```
Using TensorFlow backend.
```

In [13]:

```
# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 64
```

In [14]:

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

In [16]:

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

```
/Users/bhawesh/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:12: FutureWarning:
Method .as_matrix will be removed in a future version. Use .values instead.
  if sys.path[0] == '':
/Users/bhawesh/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:11: FutureWarning:
Method .as_matrix will be removed in a future version. Use .values instead.
  # This is added back by InteractiveShellApp.init_path()
```

In [17]:

```
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
```

In [18]:

```
X_train.shape
```

Out[18]:

```
(7352, 128, 9)
```

- Defining the Architecture of LSTM

In [19]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(32,return_sequences=True,input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.5))

model.add(LSTM(28,input_shape=(timesteps, input_dim)))
# 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:From /Users/bhawesh/anaconda3/lib/python3.7/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, 128, 32)           5376      
_________________________________________________________________
dropout_1 (Dropout)          (None, 128, 32)           0         
_________________________________________________________________
lstm_2 (LSTM)                (None, 28)                6832      
_________________________________________________________________
dropout_2 (Dropout)          (None, 28)                0         
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 174       
=================================================================
Total params: 12,382
Trainable params: 12,382
Non-trainable params: 0
_________________________________________________________________
```

In [73]:

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

In [74]:

```
# 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 [==============================] - 258s 35ms/step - loss: 1.3024 - accuracy: 0.4498 - va
l_loss: 0.9632 - val_accuracy: 0.6220
Epoch 2/30
7352/7352 [==============================] - 236s 32ms/step - loss: 0.8855 - accuracy: 0.5652 - va
l_loss: 0.8105 - val_accuracy: 0.6502
Epoch 3/30
7352/7352 [==============================] - 235s 32ms/step - loss: 0.7803 - accuracy: 0.6270 - va
l_loss: 0.7567 - val_accuracy: 0.6423
Epoch 4/30
7352/7352 [==============================] - 235s 32ms/step - loss: 0.7245 - accuracy: 0.6662 - va
l_loss: 0.6691 - val_accuracy: 0.7129
```

```
Epoch 5/30
7352/7352 [==============================] - 1078s 147ms/step - loss: 0.6872 - accuracy: 0.6834 -
val_loss: 0.6223 - val_accuracy: 0.7258
Epoch 6/30
7352/7352 [==============================] - 265s 36ms/step - loss: 0.6280 - accuracy: 0.7338 - va
l_loss: 0.6051 - val_accuracy: 0.7489
Epoch 7/30
7352/7352 [==============================] - 261s 36ms/step - loss: 0.5281 - accuracy: 0.7729 - va
l_loss: 0.7489 - val_accuracy: 0.7248
Epoch 8/30
7352/7352 [==============================] - 247s 34ms/step - loss: 0.4950 - accuracy: 0.7836 - va
l_loss: 0.4933 - val_accuracy: 0.7669
Epoch 9/30
7352/7352 [==============================] - 240s 33ms/step - loss: 0.4596 - accuracy: 0.7954 - va
l_loss: 0.4845 - val_accuracy: 0.7930
Epoch 10/30
7352/7352 [==============================] - 246s 33ms/step - loss: 0.4105 - accuracy: 0.8443 - va
l_loss: 0.5197 - val_accuracy: 0.8588
Epoch 11/30
7352/7352 [==============================] - 265s 36ms/step - loss: 0.3510 - accuracy: 0.8872 - va
l_loss: 0.3937 - val_accuracy: 0.8748
Epoch 12/30
7352/7352 [==============================] - 262s 36ms/step - loss: 0.3146 - accuracy: 0.9026 - va
l_loss: 0.3701 - val_accuracy: 0.8989
Epoch 13/30
7352/7352 [==============================] - 249s 34ms/step - loss: 0.2819 - accuracy: 0.9169 - va
l_loss: 0.3248 - val_accuracy: 0.8972
Epoch 14/30
7352/7352 [==============================] - 253s 34ms/step - loss: 0.2587 - accuracy: 0.9274 - va
l_loss: 0.3103 - val_accuracy: 0.9053
Epoch 15/30
7352/7352 [==============================] - 267s 36ms/step - loss: 0.2205 - accuracy: 0.9317 - va
l_loss: 0.3870 - val_accuracy: 0.9030
Epoch 16/30
7352/7352 [==============================] - 245s 33ms/step - loss: 0.2079 - accuracy: 0.9362 - va
l_loss: 0.3751 - val_accuracy: 0.9019
Epoch 17/30
7352/7352 [==============================] - 230s 31ms/step - loss: 0.2133 - accuracy: 0.9369 - va
l_loss: 0.3343 - val_accuracy: 0.9050
Epoch 18/30
7352/7352 [==============================] - 230s 31ms/step - loss: 0.1981 - accuracy: 0.9427 - va
l_loss: 0.3600 - val_accuracy: 0.9006
Epoch 19/30
7352/7352 [==============================] - 230s 31ms/step - loss: 0.1929 - accuracy: 0.9403 - va
l_loss: 0.3001 - val_accuracy: 0.9230
Epoch 20/30
7352/7352 [==============================] - 2031s 276ms/step - loss: 0.1762 - accuracy: 0.9399 -
val_loss: 0.3171 - val_accuracy: 0.9131
Epoch 21/30
7352/7352 [==============================] - 248s 34ms/step - loss: 0.1749 - accuracy: 0.9446 - va
l_loss: 0.3270 - val_accuracy: 0.9131
Epoch 22/30
7352/7352 [==============================] - 277s 38ms/step - loss: 0.1693 - accuracy: 0.9418 - va
l_loss: 0.3348 - val_accuracy: 0.9199
Epoch 23/30
7352/7352 [==============================] - 303s 41ms/step - loss: 0.1847 - accuracy: 0.9440 - va
l_loss: 0.3564 - val_accuracy: 0.9148
Epoch 24/30
7352/7352 [==============================] - 272s 37ms/step - loss: 0.1776 - accuracy: 0.9418 - va
l_loss: 0.5241 - val_accuracy: 0.8975
Epoch 25/30
7352/7352 [==============================] - 276s 38ms/step - loss: 0.1682 - accuracy: 0.9441 - va
l_loss: 0.3894 - val_accuracy: 0.9220
Epoch 26/30
7352/7352 [==============================] - 266s 36ms/step - loss: 0.1628 - accuracy: 0.9423 - va
l_loss: 0.3690 - val_accuracy: 0.9230
Epoch 27/30
7352/7352 [==============================] - 265s 36ms/step - loss: 0.1621 - accuracy: 0.9441 - va
l_loss: 0.3947 - val_accuracy: 0.9121
Epoch 28/30
7352/7352 [==============================] - 263s 36ms/step - loss: 0.1665 - accuracy: 0.9433 - va
l_loss: 0.3694 - val_accuracy: 0.9114
Epoch 29/30
7352/7352 [==============================] - 269s 37ms/step - loss: 0.1622 - accuracy: 0.9448 - va
l_loss: 0.3727 - val_accuracy: 0.9182
Epoch 30/30
7352/7352 [==============================] - 248s 34ms/step - loss: 0.1752 - accuracy: 0.9425 - va
```

```
l_loss: 0.4158 - val_accuracy: 0.9101
```

Out[74]:

```
<keras.callbacks.callbacks.History at 0x639c9d5c0>
```

In [75]:

```
# 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                  4      420        46        0                   0
STANDING                 0      132       400        0                   0
WALKING                  0        0         0      469                  21
WALKING_DOWNSTAIRS       0        0         0       12                 401
WALKING_UPSTAIRS         0        2         1       13                   0

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                           21
STANDING                           0
WALKING                            6
WALKING_DOWNSTAIRS                 7
WALKING_UPSTAIRS                 455
```

In [76]:

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

```
2947/2947 [==============================] - 18s 6ms/step
```

In [77]:

```
score
```

Out[77]:

```
[0.4158135156924696, 0.9100780487060547]
```

- 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

Changing LSTM

In [116]:

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

In [118]:

```
# 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(0.4))
# 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 [119]:

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

In [120]:

```
# 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 [==============================] - 168s 23ms/step - loss: 1.2421 - accuracy: 0.4939 - va
l_loss: 1.0321 - val_accuracy: 0.6074
Epoch 2/30
7352/7352 [==============================] - 196s 27ms/step - loss: 0.8385 - accuracy: 0.6436 - va
l_loss: 0.8230 - val_accuracy: 0.6308
Epoch 3/30
7352/7352 [==============================] - 175s 24ms/step - loss: 0.6884 - accuracy: 0.6912 - va
l_loss: 0.6510 - val_accuracy: 0.7435
Epoch 4/30
7352/7352 [==============================] - 178s 24ms/step - loss: 0.5859 - accuracy: 0.7564 - va
l_loss: 0.5855 - val_accuracy: 0.7611
Epoch 5/30
7352/7352 [==============================] - 149s 20ms/step - loss: 0.4952 - accuracy: 0.7969 - va
l_loss: 0.5028 - val_accuracy: 0.7967
Epoch 6/30
7352/7352 [==============================] - 145s 20ms/step - loss: 0.4476 - accuracy: 0.8411 - va
l_loss: 0.5677 - val_accuracy: 0.7869
Epoch 7/30
7352/7352 [==============================] - 144s 20ms/step - loss: 0.4050 - accuracy: 0.8637 - va
l_loss: 0.7944 - val_accuracy: 0.7760
Epoch 8/30
7352/7352 [==============================] - 145s 20ms/step - loss: 0.3277 - accuracy: 0.8976 - va
l_loss: 0.5426 - val_accuracy: 0.8208
Epoch 9/30
7352/7352 [==============================] - 144s 20ms/step - loss: 0.3029 - accuracy: 0.9070 - va
l_loss: 0.4130 - val_accuracy: 0.8578
Epoch 10/30
7352/7352 [==============================] - 1738s 236ms/step - loss: 0.2610 - accuracy: 0.9208 -
val_loss: 0.4450 - val_accuracy: 0.8571
Epoch 11/30
7352/7352 [==============================] - 145s 20ms/step - loss: 0.2344 - accuracy: 0.9244 - va
l_loss: 0.4185 - val_accuracy: 0.8792
Epoch 12/30
7352/7352 [==============================] - 146s 20ms/step - loss: 0.2109 - accuracy: 0.9305 - va
l_loss: 0.3942 - val_accuracy: 0.8812
Epoch 13/30
7352/7352 [==============================] - 147s 20ms/step - loss: 0.2157 - accuracy: 0.9332 - va
l_loss: 0.5447 - val_accuracy: 0.8609
```

```
Epoch 14/30
7352/7352 [==============================] - 168s 23ms/step - loss: 0.2064 - accuracy: 0.9340 - va
l_loss: 0.5232 - val_accuracy: 0.8748
Epoch 15/30
7352/7352 [==============================] - 170s 23ms/step - loss: 0.1824 - accuracy: 0.9389 - va
l_loss: 0.3321 - val_accuracy: 0.8799
Epoch 16/30
7352/7352 [==============================] - 184s 25ms/step - loss: 0.1679 - accuracy: 0.9407 - va
l_loss: 0.4579 - val_accuracy: 0.8663
Epoch 17/30
7352/7352 [==============================] - 150s 20ms/step - loss: 0.1710 - accuracy: 0.9412 - va
l_loss: 0.3797 - val_accuracy: 0.9016
Epoch 18/30
7352/7352 [==============================] - 150s 20ms/step - loss: 0.1726 - accuracy: 0.9437 - va
l_loss: 0.4645 - val_accuracy: 0.8924
Epoch 19/30
7352/7352 [==============================] - 161s 22ms/step - loss: 0.1719 - accuracy: 0.9389 - va
l_loss: 0.4814 - val_accuracy: 0.8860
Epoch 20/30
7352/7352 [==============================] - 160s 22ms/step - loss: 0.1550 - accuracy: 0.9444 - va
l_loss: 0.3362 - val_accuracy: 0.8992
Epoch 21/30
7352/7352 [==============================] - 158s 22ms/step - loss: 0.1674 - accuracy: 0.9430 - va
l_loss: 0.3746 - val_accuracy: 0.8945
Epoch 22/30
7352/7352 [==============================] - 156s 21ms/step - loss: 0.1546 - accuracy: 0.9468 - va
l_loss: 0.4101 - val_accuracy: 0.9016
Epoch 23/30
7352/7352 [==============================] - 167s 23ms/step - loss: 0.1529 - accuracy: 0.9452 - va
l_loss: 0.5036 - val_accuracy: 0.8799
Epoch 24/30
7352/7352 [==============================] - 166s 23ms/step - loss: 0.1416 - accuracy: 0.9475 - va
l_loss: 0.3752 - val_accuracy: 0.9077
Epoch 25/30
7352/7352 [==============================] - 182s 25ms/step - loss: 0.1360 - accuracy: 0.9482 - va
l_loss: 0.3901 - val_accuracy: 0.9091
Epoch 26/30
7352/7352 [==============================] - 192s 26ms/step - loss: 0.1892 - accuracy: 0.9406 - va
l_loss: 0.6442 - val_accuracy: 0.8697
Epoch 27/30
7352/7352 [==============================] - 189s 26ms/step - loss: 0.1508 - accuracy: 0.9493 - va
l_loss: 0.3408 - val_accuracy: 0.9043
Epoch 28/30
7352/7352 [==============================] - 170s 23ms/step - loss: 0.1324 - accuracy: 0.9491 - va
l_loss: 0.4604 - val_accuracy: 0.8904
Epoch 29/30
7352/7352 [==============================] - 157s 21ms/step - loss: 0.1316 - accuracy: 0.9498 - va
l_loss: 0.5240 - val_accuracy: 0.8880
Epoch 30/30
7352/7352 [==============================] - 144s 20ms/step - loss: 0.1471 - accuracy: 0.9474 - va
l_loss: 0.7513 - val_accuracy: 0.8755
```

Out[120]:

```
<keras.callbacks.callbacks.History at 0x650f76f98>
```

In [121]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 513        0        24        0                   0
SITTING                  0      425        64        0                   1
STANDING                 0      117       405        2                   0
WALKING                  0        0         1      386                  49
WALKING_DOWNSTAIRS       0        0         0        0                 420
WALKING_UPSTAIRS         0        4         1        4                  31

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            1
STANDING                           8
```

```
WALKING                            60
WALKING_DOWNSTAIRS                  0
WALKING_UPSTAIRS                  431
```

In [122]:

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

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

In [123]:

```
score
```

Out[123]:

```
[0.751543751711401, 0.8754665851593018]
```

In [124]:

```
# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 64
```

In [125]:

```
# 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(0.55))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_10 (LSTM)               (None, 64)                18944
_________________________________________________________________
dropout_10 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_10 (Dense)             (None, 6)                 390
=================================================================
Total params: 19,334
Trainable params: 19,334
Non-trainable params: 0
_________________________________________________________________
```

In [126]:

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

In [127]:

```
# 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
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 335s 46ms/step - loss: 1.2323 - accuracy: 0.4727 - va
l_loss: 1.0838 - val_accuracy: 0.5517
Epoch 2/30
7352/7352 [==============================] - 253s 34ms/step - loss: 0.8582 - accuracy: 0.6133 - va
l_loss: 0.7665 - val_accuracy: 0.6634
Epoch 3/30
7352/7352 [==============================] - 210s 29ms/step - loss: 0.7849 - accuracy: 0.6678 - va
l_loss: 0.7073 - val_accuracy: 0.7150
Epoch 4/30
7352/7352 [==============================] - 237s 32ms/step - loss: 0.6339 - accuracy: 0.7333 - va
l_loss: 0.6145 - val_accuracy: 0.7737
Epoch 5/30
7352/7352 [==============================] - 270s 37ms/step - loss: 0.5641 - accuracy: 0.7807 - va
l_loss: 0.4794 - val_accuracy: 0.8276
Epoch 6/30
7352/7352 [==============================] - 275s 37ms/step - loss: 0.4266 - accuracy: 0.8621 - va
l_loss: 0.3470 - val_accuracy: 0.8782
Epoch 7/30
7352/7352 [==============================] - 211s 29ms/step - loss: 0.2898 - accuracy: 0.9059 - va
l_loss: 0.4922 - val_accuracy: 0.8653
Epoch 8/30
7352/7352 [==============================] - 212s 29ms/step - loss: 0.3078 - accuracy: 0.9007 - va
l_loss: 0.3282 - val_accuracy: 0.8853
Epoch 9/30
7352/7352 [==============================] - 176s 24ms/step - loss: 0.2432 - accuracy: 0.9226 - va
l_loss: 0.4303 - val_accuracy: 0.8785
Epoch 10/30
7352/7352 [==============================] - 177s 24ms/step - loss: 0.2080 - accuracy: 0.9301 - va
l_loss: 0.3728 - val_accuracy: 0.8985
Epoch 11/30
7352/7352 [==============================] - 179s 24ms/step - loss: 0.1814 - accuracy: 0.9361 - va
l_loss: 0.4181 - val_accuracy: 0.9053
Epoch 12/30
7352/7352 [==============================] - 185s 25ms/step - loss: 0.1669 - accuracy: 0.9389 - va
l_loss: 0.4217 - val_accuracy: 0.8972
Epoch 13/30
7352/7352 [==============================] - 168s 23ms/step - loss: 0.1848 - accuracy: 0.9363 - va
l_loss: 0.4813 - val_accuracy: 0.9074
Epoch 14/30
7352/7352 [==============================] - 178s 24ms/step - loss: 0.1907 - accuracy: 0.9391 - va
l_loss: 0.6607 - val_accuracy: 0.8585
Epoch 15/30
7352/7352 [==============================] - 170s 23ms/step - loss: 0.1611 - accuracy: 0.9449 - va
l_loss: 0.2482 - val_accuracy: 0.9175
Epoch 16/30
7352/7352 [==============================] - 176s 24ms/step - loss: 0.1775 - accuracy: 0.9434 - va
l_loss: 0.3645 - val_accuracy: 0.9169
Epoch 17/30
7352/7352 [==============================] - 185s 25ms/step - loss: 0.1508 - accuracy: 0.9464 - va
l_loss: 0.4787 - val_accuracy: 0.9091
Epoch 18/30
7352/7352 [==============================] - 183s 25ms/step - loss: 0.1494 - accuracy: 0.9476 - va
l_loss: 0.3080 - val_accuracy: 0.9172
Epoch 19/30
7352/7352 [==============================] - 214s 29ms/step - loss: 0.1494 - accuracy: 0.9470 - va
l_loss: 0.3242 - val_accuracy: 0.9240
Epoch 20/30
7352/7352 [==============================] - 226s 31ms/step - loss: 0.1448 - accuracy: 0.9448 - va
l_loss: 0.3845 - val_accuracy: 0.9179
Epoch 21/30
7352/7352 [==============================] - 184s 25ms/step - loss: 0.1441 - accuracy: 0.9491 - va
l_loss: 0.3254 - val_accuracy: 0.9138
Epoch 22/30
7352/7352 [==============================] - 154s 21ms/step - loss: 0.1370 - accuracy: 0.9493 - va
l_loss: 0.4371 - val_accuracy: 0.9138
Epoch 23/30
7352/7352 [==============================] - 156s 21ms/step - loss: 0.1372 - accuracy: 0.9494 - va
l_loss: 0.3453 - val_accuracy: 0.9074
Epoch 24/30
7352/7352 [==============================] - 195s 26ms/step - loss: 0.1329 - accuracy: 0.9516 - va
l_loss: 0.3688 - val_accuracy: 0.9284
Epoch 25/30
7352/7352 [==============================] - 215s 29ms/step - loss: 0.1404 - accuracy: 0.9533 - va
l_loss: 0.4563 - val_accuracy: 0.9026
Epoch 26/30
7352/7352 [==============================] - 183s 25ms/step - loss: 0.1448 - accuracy: 0.9487 - va
```

```
l_loss: 0.4691 - val_accuracy: 0.9063
Epoch 27/30
7352/7352 [==============================] - 213s 29ms/step - loss: 0.1358 - accuracy: 0.9524 - va
l_loss: 0.4233 - val_accuracy: 0.9036
Epoch 28/30
7352/7352 [==============================] - 275s 37ms/step - loss: 0.1342 - accuracy: 0.9502 - va
l_loss: 0.2551 - val_accuracy: 0.9284
Epoch 29/30
7352/7352 [==============================] - 183s 25ms/step - loss: 0.1285 - accuracy: 0.9531 - va
l_loss: 0.3803 - val_accuracy: 0.9162
Epoch 30/30
7352/7352 [==============================] - 176s 24ms/step - loss: 0.1666 - accuracy: 0.9467 - va
l_loss: 0.2756 - val_accuracy: 0.9284
```

Out[127]:

```
<keras.callbacks.callbacks.History at 0x64f3f29b0>
```

In [131]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 535        0         0        0                   0
SITTING                  0      416        67        0                   0
STANDING                 0       96       434        1                   0
WALKING                  0        0         0      491                   3
WALKING_DOWNSTAIRS       0        0         0        6                 413
WALKING_UPSTAIRS         0        0         1       15                   8

Pred                WALKING_UPSTAIRS
True
LAYING                             2
SITTING                            8
STANDING                           1
WALKING                            2
WALKING_DOWNSTAIRS                 1
WALKING_UPSTAIRS                 447
```

In [132]:

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

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

In [133]:

```
score
```

Out[133]:

```
[0.27563241636243435, 0.92840176820755]
```

Changing Dropout

In [41]:

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

In [42]:

```
# 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(0.7))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_5 (LSTM)                (None, 32)                5376      
_________________________________________________________________
dropout_5 (Dropout)          (None, 32)                0         
_________________________________________________________________
dense_5 (Dense)              (None, 6)                 198       
=================================================================
Total params: 5,574
Trainable params: 5,574
Non-trainable params: 0
_________________________________________________________________
```

In [43]:

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

In [44]:

```
# 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 [==============================] - 131s 18ms/step - loss: 1.3737 - accuracy: 0.4276 - va
l_loss: 1.1522 - val_accuracy: 0.4951
Epoch 2/30
7352/7352 [==============================] - 118s 16ms/step - loss: 1.0582 - accuracy: 0.5358 - va
l_loss: 0.9802 - val_accuracy: 0.5643
Epoch 3/30
7352/7352 [==============================] - 117s 16ms/step - loss: 0.9794 - accuracy: 0.5681 - va
l_loss: 0.9824 - val_accuracy: 0.5168
Epoch 4/30
7352/7352 [==============================] - 118s 16ms/step - loss: 1.0273 - accuracy: 0.5491 - va
l_loss: 0.9189 - val_accuracy: 0.5775
Epoch 5/30
7352/7352 [==============================] - 119s 16ms/step - loss: 0.8047 - accuracy: 0.6411 - va
l_loss: 0.8748 - val_accuracy: 0.5969
Epoch 6/30
7352/7352 [==============================] - 119s 16ms/step - loss: 0.8253 - accuracy: 0.6299 - va
l_loss: 0.8473 - val_accuracy: 0.6057
Epoch 7/30
7352/7352 [==============================] - 134s 18ms/step - loss: 0.7336 - accuracy: 0.6629 - va
l_loss: 0.7836 - val_accuracy: 0.6081
Epoch 8/30
7352/7352 [==============================] - 120s 16ms/step - loss: 0.7220 - accuracy: 0.6835 - va
l_loss: 0.8439 - val_accuracy: 0.6529
Epoch 9/30
7352/7352 [==============================] - 116s 16ms/step - loss: 0.7002 - accuracy: 0.7101 - va
l_loss: 0.8434 - val_accuracy: 0.7143
Epoch 10/30
7352/7352 [==============================] - 116s 16ms/step - loss: 0.6881 - accuracy: 0.7369 - va
l_loss: 0.6961 - val_accuracy: 0.7479
Epoch 11/30
7352/7352 [==============================] - 131s 18ms/step - loss: 0.5817 - accuracy: 0.7787 - va
l_loss: 0.8060 - val_accuracy: 0.7462
Epoch 12/30
```

```
7352/7352 [==============================] - 136s 18ms/step - loss: 0.5308 - accuracy: 0.7969 - va
l_loss: 0.5615 - val_accuracy: 0.7818
Epoch 13/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.5646 - accuracy: 0.7976 - va
l_loss: 0.6740 - val_accuracy: 0.7665
Epoch 14/30
7352/7352 [==============================] - 119s 16ms/step - loss: 0.4805 - accuracy: 0.8300 - va
l_loss: 0.6425 - val_accuracy: 0.7920
Epoch 15/30
7352/7352 [==============================] - 115s 16ms/step - loss: 0.5441 - accuracy: 0.8162 - va
l_loss: 0.8308 - val_accuracy: 0.7825
Epoch 16/30
7352/7352 [==============================] - 115s 16ms/step - loss: 0.4790 - accuracy: 0.8426 - va
l_loss: 0.5679 - val_accuracy: 0.8076
Epoch 17/30
7352/7352 [==============================] - 118s 16ms/step - loss: 0.4450 - accuracy: 0.8588 - va
l_loss: 0.8075 - val_accuracy: 0.8022
Epoch 18/30
7352/7352 [==============================] - 130s 18ms/step - loss: 0.4052 - accuracy: 0.8829 - va
l_loss: 0.5603 - val_accuracy: 0.8442
Epoch 19/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.3791 - accuracy: 0.8915 - va
l_loss: 0.4587 - val_accuracy: 0.8473
Epoch 20/30
7352/7352 [==============================] - 130s 18ms/step - loss: 0.3629 - accuracy: 0.8936 - va
l_loss: 0.4256 - val_accuracy: 0.8738
Epoch 21/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.3635 - accuracy: 0.9013 - va
l_loss: 0.6416 - val_accuracy: 0.8582
Epoch 22/30
7352/7352 [==============================] - 135s 18ms/step - loss: 0.3661 - accuracy: 0.8950 - va
l_loss: 0.5373 - val_accuracy: 0.8371
Epoch 23/30
7352/7352 [==============================] - 142s 19ms/step - loss: 0.3318 - accuracy: 0.9055 - va
l_loss: 0.4484 - val_accuracy: 0.8744
Epoch 24/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.3218 - accuracy: 0.9142 - va
l_loss: 0.7172 - val_accuracy: 0.8527
Epoch 25/30
7352/7352 [==============================] - 127s 17ms/step - loss: 0.2925 - accuracy: 0.9163 - va
l_loss: 0.5847 - val_accuracy: 0.8843
Epoch 26/30
7352/7352 [==============================] - 130s 18ms/step - loss: 0.2644 - accuracy: 0.9212 - va
l_loss: 0.6235 - val_accuracy: 0.8707
Epoch 27/30
7352/7352 [==============================] - 132s 18ms/step - loss: 0.2724 - accuracy: 0.9223 - va
l_loss: 0.6380 - val_accuracy: 0.8826
Epoch 28/30
7352/7352 [==============================] - 131s 18ms/step - loss: 0.2841 - accuracy: 0.9214 - va
l_loss: 0.5960 - val_accuracy: 0.8829
Epoch 29/30
7352/7352 [==============================] - 130s 18ms/step - loss: 0.3320 - accuracy: 0.9125 - va
l_loss: 0.8264 - val_accuracy: 0.8514
Epoch 30/30
7352/7352 [==============================] - 127s 17ms/step - loss: 0.3255 - accuracy: 0.9191 - va
l_loss: 0.8671 - val_accuracy: 0.8476
```

Out[44]:

```
<keras.callbacks.callbacks.History at 0x645d1ada0>
```

In [45]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 484       25         6        1                   0
SITTING                  0      370        92        2                   1
STANDING                 0       65       398        8                   0
WALKING                  0        0         0      431                   9
WALKING_DOWNSTAIRS       0        0         0       25                 388
WALKING_UPSTAIRS         0        2         0       30                  12
```

```
Pred                   WALKING_UPSTAIRS
True
LAYING                               21
SITTING                              26
STANDING                             61
WALKING                              56
WALKING_DOWNSTAIRS                    7
WALKING_UPSTAIRS                    427
```

In [46]:

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

```
2947/2947 [==============================] - 8s 3ms/step
```

In [47]:

```
score
```

Out[47]:

```
[0.8671318305163158, 0.84764164686203]
```

Adding More layers

In [48]:

```
# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 64
```

In [52]:

```
# 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(0.4))
# 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_12 (LSTM)               (None, 64)                18944
_________________________________________________________________
dropout_9 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_6 (Dense)              (None, 6)                 390
=================================================================
Total params: 19,334
Trainable params: 19,334
Non-trainable params: 0
_________________________________________________________________
```

In [53]:

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

In [54]:

```python
# 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 [==============================] - 181s 25ms/step - loss: 1.2111 - accuracy: 0.4767 - va
l_loss: 1.1095 - val_accuracy: 0.5497
Epoch 2/30
7352/7352 [==============================] - 191s 26ms/step - loss: 0.8816 - accuracy: 0.6336 - va
l_loss: 0.8614 - val_accuracy: 0.6400
Epoch 3/30
7352/7352 [==============================] - 184s 25ms/step - loss: 0.6760 - accuracy: 0.7191 - va
l_loss: 1.2741 - val_accuracy: 0.5185
Epoch 4/30
7352/7352 [==============================] - 180s 24ms/step - loss: 0.5214 - accuracy: 0.8161 - va
l_loss: 0.5649 - val_accuracy: 0.8273
Epoch 5/30
7352/7352 [==============================] - 181s 25ms/step - loss: 0.3538 - accuracy: 0.8876 - va
l_loss: 0.3780 - val_accuracy: 0.8799
Epoch 6/30
7352/7352 [==============================] - 176s 24ms/step - loss: 0.2669 - accuracy: 0.9116 - va
l_loss: 0.5281 - val_accuracy: 0.8585
Epoch 7/30
7352/7352 [==============================] - 172s 23ms/step - loss: 0.2330 - accuracy: 0.9207 - va
l_loss: 0.4112 - val_accuracy: 0.8721
Epoch 8/30
7352/7352 [==============================] - 188s 26ms/step - loss: 0.2066 - accuracy: 0.9302 - va
l_loss: 0.3681 - val_accuracy: 0.8880
Epoch 9/30
7352/7352 [==============================] - 189s 26ms/step - loss: 0.1992 - accuracy: 0.9309 - va
l_loss: 0.3448 - val_accuracy: 0.8935
Epoch 10/30
7352/7352 [==============================] - 181s 25ms/step - loss: 0.1873 - accuracy: 0.9355 - va
l_loss: 0.5202 - val_accuracy: 0.8680
Epoch 11/30
7352/7352 [==============================] - 182s 25ms/step - loss: 0.1750 - accuracy: 0.9419 - va
l_loss: 0.3328 - val_accuracy: 0.9013
Epoch 12/30
7352/7352 [==============================] - 176s 24ms/step - loss: 0.1649 - accuracy: 0.9389 - va
l_loss: 0.3874 - val_accuracy: 0.8846
Epoch 13/30
7352/7352 [==============================] - 177s 24ms/step - loss: 0.1559 - accuracy: 0.9450 - va
l_loss: 0.3335 - val_accuracy: 0.9111
Epoch 14/30
7352/7352 [==============================] - 181s 25ms/step - loss: 0.1632 - accuracy: 0.9431 - va
l_loss: 0.4399 - val_accuracy: 0.9043
Epoch 15/30
7352/7352 [==============================] - 174s 24ms/step - loss: 0.1525 - accuracy: 0.9436 - va
l_loss: 0.3367 - val_accuracy: 0.8958
Epoch 16/30
7352/7352 [==============================] - 184s 25ms/step - loss: 0.1397 - accuracy: 0.9472 - va
l_loss: 0.4192 - val_accuracy: 0.8958
Epoch 17/30
7352/7352 [==============================] - 170s 23ms/step - loss: 0.1426 - accuracy: 0.9501 - va
l_loss: 0.3113 - val_accuracy: 0.9087
Epoch 18/30
7352/7352 [==============================] - 173s 24ms/step - loss: 0.1371 - accuracy: 0.9497 - va
l_loss: 0.3204 - val_accuracy: 0.9057
Epoch 19/30
7352/7352 [==============================] - 170s 23ms/step - loss: 0.1412 - accuracy: 0.9497 - va
l_loss: 0.2370 - val_accuracy: 0.9138
Epoch 20/30
7352/7352 [==============================] - 165s 22ms/step - loss: 0.1490 - accuracy: 0.9475 - va
l_loss: 0.2868 - val_accuracy: 0.9128
Epoch 21/30
7352/7352 [==============================] - 175s 24ms/step - loss: 0.1303 - accuracy: 0.9472 - va
l_loss: 0.7616 - val_accuracy: 0.8748
Epoch 22/30
7352/7352 [==============================] - 175s 24ms/step - loss: 0.1344 - accuracy: 0.9474 - va
l_loss: 0.3695 - val_accuracy: 0.9019
Epoch 23/30
7352/7352 [==============================] - 186s 25ms/step - loss: 0.1287 - accuracy: 0.9490 - va
l_loss: 0.2724 - val_accuracy: 0.9101
```

```
l_loss: 0.2724 - val_accuracy: 0.9101
Epoch 24/30
7352/7352 [==============================] - 174s 24ms/step - loss: 0.1210 - accuracy: 0.9518 - va
l_loss: 0.2277 - val_accuracy: 0.9152
Epoch 25/30
7352/7352 [==============================] - 188s 26ms/step - loss: 0.1201 - accuracy: 0.9520 - va
l_loss: 0.3673 - val_accuracy: 0.9158
Epoch 26/30
7352/7352 [==============================] - 171s 23ms/step - loss: 0.1258 - accuracy: 0.9527 - va
l_loss: 0.3518 - val_accuracy: 0.9182
Epoch 27/30
7352/7352 [==============================] - 180s 24ms/step - loss: 0.1389 - accuracy: 0.9461 - va
l_loss: 0.3393 - val_accuracy: 0.9196
Epoch 28/30
7352/7352 [==============================] - 172s 23ms/step - loss: 0.1268 - accuracy: 0.9494 - va
l_loss: 0.3260 - val_accuracy: 0.8941
Epoch 29/30
7352/7352 [==============================] - 170s 23ms/step - loss: 0.1226 - accuracy: 0.9529 - va
l_loss: 0.2653 - val_accuracy: 0.9223
Epoch 30/30
7352/7352 [==============================] - 166s 23ms/step - loss: 0.1291 - accuracy: 0.9510 - va
l_loss: 0.3104 - val_accuracy: 0.9131
```

Out[54]:

```
<keras.callbacks.callbacks.History at 0x6549c6320>
```

In [55]:

```
# 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      394        96        1                   0
STANDING                 0       63       468        1                   0
WALKING                  0        0         0      489                   0
WALKING_DOWNSTAIRS       0        0         0        9                 410
WALKING_UPSTAIRS         0        0         0       47                   4

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            0
STANDING                           0
WALKING                            7
WALKING_DOWNSTAIRS                 1
WALKING_UPSTAIRS                 420
```

In [56]:

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

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

In [57]:

```
score
```

Out[57]:

```
[0.31043342269294943, 0.9131320118904114]
```

In [24]:

```
# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 64
```

In [125]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden,return_sequences=True,input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.55))
# Configuring the parameters
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
# 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()
```

```
Model: "sequential_29"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_45 (LSTM)               (None, 128, 64)           18944     
_________________________________________________________________
dropout_39 (Dropout)         (None, 128, 64)           0         
_________________________________________________________________
lstm_46 (LSTM)               (None, 64)                33024     
_________________________________________________________________
dropout_40 (Dropout)         (None, 64)                0         
_________________________________________________________________
dense_51 (Dense)             (None, 6)                 390       
=================================================================
Total params: 52,358
Trainable params: 52,358
Non-trainable params: 0
_________________________________________________________________
```

In [126]:

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

In [68]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True                                                                         
LAYING                 512        0         0        0                   0   
SITTING                  3      417        67        1                   0   
STANDING                 0      111       420        0                   0   
WALKING                  0        0         1      467                  17   
WALKING_DOWNSTAIRS       0        0         0        2                 415   
WALKING_UPSTAIRS         0        1         2        4                  12   

Pred                WALKING_UPSTAIRS  
True                                  
LAYING                            25  
SITTING                            3  
STANDING                           1  
WALKING                           11  
WALKING_DOWNSTAIRS                 3  
WALKING_UPSTAIRS                 452  
```

In [19]:

```
Y_train_dynamic=[]
X_train_dynamic=[]
Y_train_static=[]
X_train_static=[]
for i in range(0,len(Y_train)):
```

```
    if(Y_train[i][0]==1 or Y_train[i][1]==1 or Y_train[i][2]==1):
        Y_train_dynamic.append(Y_train[i])
        X_train_dynamic.append(X_train[i])
    else:
        Y_train_static.append(Y_train[i])
        X_train_static.append(X_train[i])
```

In [20]:

```
X_train_dynamic=np.array(X_train_dynamic)
Y_train_dynamic=np.array(Y_train_dynamic)
X_train_static=np.array(X_train_static)
Y_train_static=np.array(Y_train_static)
```

In [21]:

```
Y_train_dynamic=Y_train_dynamic[0:len(Y_train_dynamic),0:3]
```

In [22]:

```
Y_train_dynamic.shape
```

Out[22]:

```
(3285, 3)
```

In [27]:

```
Y_test_dynamic=Y_test_dynamic[0:len(Y_test_dynamic),0:3]
```

In [24]:

```
Y_test_dynamic=[]
X_test_dynamic=[]
Y_test_static=[]
X_test_static=[]
for i in range(0,len(Y_test)):
    if(Y_test[i][0]==1 or Y_test[i][1]==1 or Y_test[i][2]==1):
        Y_test_dynamic.append(Y_test[i])
        X_test_dynamic.append(X_test[i])
    else:
        Y_test_static.append(Y_test[i])
        X_test_static.append(X_test[i])
```

In [26]:

```
X_test_dynamic=np.array(X_test_dynamic)
Y_test_dynamic=np.array(Y_test_dynamic)
X_test_static=np.array(X_test_static)
Y_test_static=np.array(Y_test_static)
```

In [28]:

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

```
2947/2947 [==============================] - 24s 8ms/step
```

In [29]:

```
score
```

Out[29]:

```
[0.3660856851979343, 0.9205971956253052]
```

In [257]:

```
# Initializing parameters
epochs = 30
batch_size = 32
n_hidden = 64
```

In [258]:

```
from keras.regularizers import l2
```

In [259]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(128,return_sequences=True,input_shape=(timesteps,
input_dim),bias_regularizer=l2(0.001)))
# Adding a dropout layer
model.add(Dropout(0.2))

model.add(LSTM(64,input_shape=(timesteps, input_dim),bias_regularizer=l2(0.001)))
# Adding a dropout layer
model.add(Dropout(0.5))
model.add(Dense(50, activation='sigmoid'))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_78"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_53 (LSTM)               (None, 128, 128)          70656
_________________________________________________________________
dropout_81 (Dropout)         (None, 128, 128)          0
_________________________________________________________________
lstm_54 (LSTM)               (None, 64)                49408
_________________________________________________________________
dropout_82 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_121 (Dense)            (None, 50)                3250
_________________________________________________________________
dense_122 (Dense)            (None, 6)                 306
=================================================================
Total params: 123,620
Trainable params: 123,620
Non-trainable params: 0
_________________________________________________________________
```

In [260]:

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

In [261]:

```
print(X_train.shape)
print(Y_train.shape)
print(X_test.shape)
print(Y_test.shape)
```

```
(7352, 128, 9)
(7352, 6)
(2947, 128, 9)
(2947, 6)
```

In [262]:

```
# Training the model
history=model.fit(X_train,
```

```
        Y_train,
        batch_size=25,
        validation_data=(X_test, Y_test),
        epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 373s 51ms/step - loss: 1.4446 - accuracy: 0.4539 - va
l_loss: 1.1004 - val_accuracy: 0.5338
Epoch 2/30
7352/7352 [==============================] - 383s 52ms/step - loss: 0.9190 - accuracy: 0.6235 - va
l_loss: 0.8570 - val_accuracy: 0.6932
Epoch 3/30
7352/7352 [==============================] - 346s 47ms/step - loss: 0.7800 - accuracy: 0.6782 - va
l_loss: 0.6684 - val_accuracy: 0.7472
Epoch 4/30
7352/7352 [==============================] - 337s 46ms/step - loss: 0.6132 - accuracy: 0.7606 - va
l_loss: 0.5975 - val_accuracy: 0.7645
Epoch 5/30
7352/7352 [==============================] - 362s 49ms/step - loss: 0.5592 - accuracy: 0.7724 - va
l_loss: 0.7208 - val_accuracy: 0.6715
Epoch 6/30
7352/7352 [==============================] - 351s 48ms/step - loss: 0.4514 - accuracy: 0.8458 - va
l_loss: 0.4499 - val_accuracy: 0.8751
Epoch 7/30
7352/7352 [==============================] - 391s 53ms/step - loss: 0.3300 - accuracy: 0.9106 - va
l_loss: 0.3942 - val_accuracy: 0.9016
Epoch 8/30
7352/7352 [==============================] - 380s 52ms/step - loss: 0.2662 - accuracy: 0.9274 - va
l_loss: 0.3214 - val_accuracy: 0.8951
Epoch 9/30
7352/7352 [==============================] - 391s 53ms/step - loss: 0.2099 - accuracy: 0.9387 - va
l_loss: 0.3543 - val_accuracy: 0.8918
Epoch 10/30
7352/7352 [==============================] - 384s 52ms/step - loss: 0.1986 - accuracy: 0.9396 - va
l_loss: 0.3606 - val_accuracy: 0.8938
Epoch 11/30
7352/7352 [==============================] - 379s 52ms/step - loss: 0.1981 - accuracy: 0.9416 - va
l_loss: 0.9166 - val_accuracy: 0.8215
Epoch 12/30
7352/7352 [==============================] - 364s 49ms/step - loss: 0.1856 - accuracy: 0.9415 - va
l_loss: 0.2741 - val_accuracy: 0.9040
Epoch 13/30
7352/7352 [==============================] - 382s 52ms/step - loss: 0.1749 - accuracy: 0.9453 - va
l_loss: 0.2946 - val_accuracy: 0.9070
Epoch 14/30
7352/7352 [==============================] - 378s 51ms/step - loss: 0.1781 - accuracy: 0.9426 - va
l_loss: 0.3398 - val_accuracy: 0.9023
Epoch 15/30
7352/7352 [==============================] - 386s 53ms/step - loss: 0.1457 - accuracy: 0.9489 - va
l_loss: 0.3091 - val_accuracy: 0.9169
Epoch 16/30
7352/7352 [==============================] - 367s 50ms/step - loss: 0.1538 - accuracy: 0.9479 - va
l_loss: 0.4602 - val_accuracy: 0.8870
Epoch 17/30
7352/7352 [==============================] - 394s 54ms/step - loss: 0.1433 - accuracy: 0.9501 - va
l_loss: 0.2852 - val_accuracy: 0.9179
Epoch 18/30
7352/7352 [==============================] - 379s 52ms/step - loss: 0.1330 - accuracy: 0.9501 - va
l_loss: 0.3222 - val_accuracy: 0.9145
Epoch 19/30
7352/7352 [==============================] - 387s 53ms/step - loss: 0.1630 - accuracy: 0.9463 - va
l_loss: 0.5332 - val_accuracy: 0.9040
Epoch 20/30
7352/7352 [==============================] - 405s 55ms/step - loss: 0.1536 - accuracy: 0.9467 - va
l_loss: 0.4004 - val_accuracy: 0.9148
Epoch 21/30
7352/7352 [==============================] - 401s 54ms/step - loss: 0.1383 - accuracy: 0.9506 - va
l_loss: 0.3985 - val_accuracy: 0.8965
Epoch 22/30
7352/7352 [==============================] - 407s 55ms/step - loss: 0.1261 - accuracy: 0.9508 - va
l_loss: 0.3109 - val_accuracy: 0.9169
Epoch 23/30
7352/7352 [==============================] - 393s 53ms/step - loss: 0.1416 - accuracy: 0.9497 - va
l_loss: 0.4637 - val_accuracy: 0.9070
Epoch 24/30
```

```
7352/7352 [==============================] - 388s 53ms/step - loss: 0.1351 - accuracy: 0.9529 - va
l_loss: 0.4951 - val_accuracy: 0.9070
Epoch 25/30
7352/7352 [==============================] - 376s 51ms/step - loss: 0.1660 - accuracy: 0.9505 - va
l_loss: 0.7259 - val_accuracy: 0.8707
Epoch 26/30
7352/7352 [==============================] - 380s 52ms/step - loss: 0.1229 - accuracy: 0.9523 - va
l_loss: 0.3463 - val_accuracy: 0.9128
Epoch 27/30
7352/7352 [==============================] - 375s 51ms/step - loss: 0.1289 - accuracy: 0.9540 - va
l_loss: 0.3967 - val_accuracy: 0.9131
Epoch 28/30
7352/7352 [==============================] - 394s 54ms/step - loss: 0.1193 - accuracy: 0.9525 - va
l_loss: 0.4694 - val_accuracy: 0.8972
Epoch 29/30
7352/7352 [==============================] - 391s 53ms/step - loss: 0.1182 - accuracy: 0.9543 - va
l_loss: 0.3567 - val_accuracy: 0.9067
Epoch 30/30
7352/7352 [==============================] - 386s 52ms/step - loss: 0.1226 - accuracy: 0.9521 - va
l_loss: 0.3985 - val_accuracy: 0.9118
```

In [21]:

```
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D
from keras.layers import Dense, Activation, Flatten
from keras.layers import LeakyReLU
model = Sequential()
model.add(Conv1D(filters=18,kernel_size=1, activation=LeakyReLU(alpha=0.3)
                 , input_shape=(timesteps,input_dim),kernel_initializer='glorot_uniform'))
model.add(MaxPooling1D(pool_size=2))
model.add(Conv1D(filters=36, kernel_size=3,kernel_initializer='glorot_uniform' ,activation=LeakyReL
U(alpha=0.3)))
model.add(MaxPooling1D(pool_size=2,strides=2))
model.add(Conv1D(filters=36, kernel_size=3, kernel_initializer='glorot_uniform',activation=LeakyReL
U(alpha=0.3)))
model.add(Dropout(0.5))
model.add(Conv1D(filters=144, kernel_size=3,kernel_initializer='glorot_uniform' ,activation=LeakyRe
LU(alpha=0.3),strides=2))
model.add(MaxPooling1D(pool_size=2,strides=2))
model.add(Dropout(0.5))

model.add(Flatten())
model.add(Dense(32, activation='sigmoid'))
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
WARNING:tensorflow:From /Users/bhawesh/anaconda3/lib/python3.7/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.
```

```
/Users/bhawesh/anaconda3/lib/python3.7/site-packages/keras/activations.py:235: UserWarning: Do not
pass a layer instance (such as LeakyReLU) as the activation argument of another layer. Instead, ad
vanced activation layers should be used just like any other layer in a model.
  identifier=identifier.__class__.__name__))
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 128, 18)           180
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 64, 18)            0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 62, 36)            1980
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 31, 36)            0
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 29, 36)            3924
_________________________________________________________________
dropout_1 (Dropout)          (None, 29, 36)            0
_________________________________________________________________
```

```
conv1d_4 (Conv1D)            (None, 14, 144)           15696
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 7, 144)            0
_________________________________________________________________
dropout_2 (Dropout)          (None, 7, 144)            0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1008)              0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                32288
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 198
=================================================================
Total params: 54,266
Trainable params: 54,266
Non-trainable params: 0
_________________________________________________________________
```

In [30]:

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

In [31]:

```
history=model.fit(X_train,
          Y_train,
          batch_size=50,
          validation_data=(X_test, Y_test),
          epochs=15)
```

```
WARNING:tensorflow:From /Users/bhawesh/anaconda3/lib/python3.7/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 1/15
7352/7352 [==============================] - 22s 3ms/step - loss: 1.2981 - accuracy: 0.5016 - val_loss: 0.9130 - val_accuracy: 0.6508
Epoch 2/15
7352/7352 [==============================] - 19s 3ms/step - loss: 0.6765 - accuracy: 0.7889 - val_loss: 0.5744 - val_accuracy: 0.8235
Epoch 3/15
7352/7352 [==============================] - 18s 2ms/step - loss: 0.4748 - accuracy: 0.8674 - val_loss: 0.5086 - val_accuracy: 0.8181
Epoch 4/15
7352/7352 [==============================] - 18s 2ms/step - loss: 0.3690 - accuracy: 0.9006 - val_loss: 0.4396 - val_accuracy: 0.8476
Epoch 5/15
7352/7352 [==============================] - 18s 2ms/step - loss: 0.2896 - accuracy: 0.9192 - val_loss: 0.3617 - val_accuracy: 0.8819
Epoch 6/15
7352/7352 [==============================] - 18s 2ms/step - loss: 0.2357 - accuracy: 0.9336 - val_loss: 0.3510 - val_accuracy: 0.8758
Epoch 7/15
7352/7352 [==============================] - 17s 2ms/step - loss: 0.2078 - accuracy: 0.9406 - val_loss: 0.3206 - val_accuracy: 0.8856
Epoch 8/15
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1901 - accuracy: 0.9421 - val_loss: 0.3095 - val_accuracy: 0.8863
Epoch 9/15
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1592 - accuracy: 0.9490 - val_loss: 0.3055 - val_accuracy: 0.8924
Epoch 10/15
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1489 - accuracy: 0.9514 - val_loss: 0.2877 - val_accuracy: 0.9013
Epoch 11/15
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1508 - accuracy: 0.9501 - val_loss: 0.2989 - val_accuracy: 0.8951
Epoch 12/15
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1367 - accuracy: 0.9531 - val_loss: 0.2901 - val_accuracy: 0.9006
Epoch 13/15
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1305 - accuracy: 0.9542 - val_loss: 0.2776 - val_accuracy: 0.9060
```

```
Epoch 14/15
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1270 - accuracy: 0.9546 - val_
loss: 0.3109 - val_accuracy: 0.9067
Epoch 15/15
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1229 - accuracy: 0.9565 - val_
loss: 0.2777 - val_accuracy: 0.9033
```

In [57]:

```
history=model.fit(X_train,
          Y_train,
          batch_size=16,
          validation_data=(X_test, Y_test),
          epochs=15)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/15
7352/7352 [==============================] - 26s 4ms/step - loss: 0.1001 - accuracy: 0.9596 - val_
loss: 0.3262 - val_accuracy: 0.9077
Epoch 2/15
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1092 - accuracy: 0.9551 - val_
loss: 0.3122 - val_accuracy: 0.9182
Epoch 3/15
7352/7352 [==============================] - 27s 4ms/step - loss: 0.1011 - accuracy: 0.9554 - val_
loss: 0.2787 - val_accuracy: 0.9189
Epoch 4/15
7352/7352 [==============================] - 26s 4ms/step - loss: 0.1040 - accuracy: 0.9553 - val_
loss: 0.3150 - val_accuracy: 0.9148
Epoch 5/15
7352/7352 [==============================] - 25s 3ms/step - loss: 0.0999 - accuracy: 0.9567 - val_
loss: 0.2987 - val_accuracy: 0.9108
Epoch 6/15
7352/7352 [==============================] - 25s 3ms/step - loss: 0.0891 - accuracy: 0.9608 - val_
loss: 0.3206 - val_accuracy: 0.9121
Epoch 7/15
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1057 - accuracy: 0.9547 - val_
loss: 0.2989 - val_accuracy: 0.9162
Epoch 8/15
7352/7352 [==============================] - 28s 4ms/step - loss: 0.1012 - accuracy: 0.9581 - val_
loss: 0.2825 - val_accuracy: 0.9152
Epoch 9/15
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1013 - accuracy: 0.9561 - val_
loss: 0.3459 - val_accuracy: 0.9172
Epoch 10/15
7352/7352 [==============================] - 28s 4ms/step - loss: 0.0924 - accuracy: 0.9593 - val_
loss: 0.3192 - val_accuracy: 0.9080
Epoch 11/15
7352/7352 [==============================] - 29s 4ms/step - loss: 0.0894 - accuracy: 0.9580 - val_
loss: 0.3449 - val_accuracy: 0.9186
Epoch 12/15
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0971 - accuracy: 0.9567 - val_
loss: 0.3526 - val_accuracy: 0.9192
Epoch 13/15
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0890 - accuracy: 0.9606 - val_
loss: 0.3095 - val_accuracy: 0.9206
Epoch 14/15
7352/7352 [==============================] - 31s 4ms/step - loss: 0.0875 - accuracy: 0.9616 - val_
loss: 0.3480 - val_accuracy: 0.9023
Epoch 15/15
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0897 - accuracy: 0.9606 - val_
loss: 0.3954 - val_accuracy: 0.9046
```

In [249]:

```
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      424        60        0                   0
STANDING                 0      117       415        0                   0
WALKING                  0        0         0      485                   7
WALKING_DOWNSTAIRS       0        0         0        0                 420
WALKING_UPSTAIRS         0        0         0        1                  26
```

```
Pred                    WALKING_UPSTAIRS
True
LAYING                                 0
SITTING                                1
STANDING                               0
WALKING                                4
WALKING_DOWNSTAIRS                     0
WALKING_UPSTAIRS                     444
```

In [250]:

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

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

In [251]:

```
score
```

Out[251]:

```
[0.3760031298141544, 0.9246691465377808]
```

Divide and Conquor

In [ ]:

```
#https://scholarcommons.usf.edu/cgi/viewcontent.cgi?article=8755&context=etd
```

Model for prediction of Static or dynamic

In [22]:

```
model_d = Sequential()
model_d.add(Conv1D(filters=64, kernel_size=5, activation='relu',kernel_initializer='lecun_uniform',
input_shape=(128,9)))
model_d.add(Conv1D(filters=32, kernel_size=3, activation='relu',kernel_initializer='lecun_uniform')
)
model_d.add(Dropout(0.45))
model_d.add(MaxPooling1D(pool_size=3))
model_d.add(Flatten())
model_d.add(Dense(32, activation='relu'))
model_d.add(Dense(2, activation='softmax'))
model_d.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv1d_5 (Conv1D)            (None, 124, 64)           2944      
_________________________________________________________________
conv1d_6 (Conv1D)            (None, 122, 32)           6176      
_________________________________________________________________
dropout_3 (Dropout)          (None, 122, 32)           0         
_________________________________________________________________
max_pooling1d_4 (MaxPooling1 (None, 40, 32)            0         
_________________________________________________________________
flatten_2 (Flatten)          (None, 1280)              0         
_________________________________________________________________
dense_3 (Dense)              (None, 32)                40992     
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 66        
=================================================================
Total params: 50,178
Trainable params: 50,178
Non-trainable params: 0
_________________________________________________________________
```

In [23]:

```
## Classifying data as 2 class dynamic vs static
##data preparation
def data_scaled_2class():
    # Data directory
    DATADIR = 'UCI_HAR_Dataset'

    # 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"
        ]

    # 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))

    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]
        for i in range(0,len(y)):
            if(y[i]<=3):
                y[i] = 0
            else:
                y[i]=1
        return pd.get_dummies(y).as_matrix()

    X_train_2c, X_val_2c = load_signals('train'), load_signals('test')
    Y_train_2c, Y_val_2c = load_y('train'), load_y('test')

    return X_train_2c, Y_train_2c, X_val_2c,  Y_val_2c
```

In [24]:

```
X_train_2c, Y_train_2c, X_val_2c,  Y_val_2c = data_scaled_2class()
```

```
/Users/bhawesh/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:35: FutureWarning:
Method .as_matrix will be removed in a future version. Use .values instead.
/Users/bhawesh/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:56: FutureWarning:
Method .as_matrix will be removed in a future version. Use .values instead.
```

In [25]:

```
print(X_train_2c.shape)
print(Y_val_2c.shape)
```

```
(7352, 128, 9)
(2947, 2)
```

In [202]:

```
from keras.callbacks import *
filepath="epochs:{epoch:03d}-val_acc:{val_accuracy:.3f}.hdf7"
checkpoint_2 = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1, mode='max')
```

In [203]:

```
callbacks_list=[checkpoint_2]
```

In [204]:

```
model_d.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [205]:

```
model_d.fit(X_train_2c,Y_train_2c, epochs=20, batch_size=16,validation_data=(X_val_2c, Y_val_2c),
            verbose=1,callbacks=callbacks_list)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/20
7352/7352 [==============================] - 23s 3ms/step - loss: 0.0246 - accuracy: 0.9895 - val_
loss: 0.0512 - val_accuracy: 0.9885

Epoch 00001: saving model to epochs:001-val_acc:0.988.hdf7
Epoch 2/20
7352/7352 [==============================] - 23s 3ms/step - loss: 9.4761e-04 - accuracy: 0.9996 -
val_loss: 0.0492 - val_accuracy: 0.9878

Epoch 00002: saving model to epochs:002-val_acc:0.988.hdf7
Epoch 3/20
7352/7352 [==============================] - 22s 3ms/step - loss: 0.0064 - accuracy: 0.9985 - val_
loss: 0.0075 - val_accuracy: 0.9976

Epoch 00003: saving model to epochs:003-val_acc:0.998.hdf7
Epoch 4/20
7352/7352 [==============================] - 22s 3ms/step - loss: 8.6226e-04 - accuracy: 0.9997 -
val_loss: 0.0025 - val_accuracy: 0.9990

Epoch 00004: saving model to epochs:004-val_acc:0.999.hdf7
Epoch 5/20
7352/7352 [==============================] - 24s 3ms/step - loss: 8.3795e-04 - accuracy: 0.9997 -
val_loss: 0.0024 - val_accuracy: 0.9997

Epoch 00005: saving model to epochs:005-val_acc:1.000.hdf7
Epoch 6/20
7352/7352 [==============================] - 21s 3ms/step - loss: 1.5161e-04 - accuracy: 1.0000 -
val_loss: 0.0037 - val_accuracy: 0.9990

Epoch 00006: saving model to epochs:006-val_acc:0.999.hdf7
Epoch 7/20
7352/7352 [==============================] - 21s 3ms/step - loss: 1.6798e-05 - accuracy: 1.0000 -
val_loss: 0.0063 - val_accuracy: 0.9983

Epoch 00007: saving model to epochs:007-val_acc:0.998.hdf7
Epoch 8/20
7352/7352 [==============================] - 21s 3ms/step - loss: 7.9455e-06 - accuracy: 1.0000 -
val_loss: 0.0046 - val_accuracy: 0.9986

Epoch 00008: saving model to epochs:008-val_acc:0.999.hdf7
Epoch 9/20
7352/7352 [==============================] - 21s 3ms/step - loss: 3.7265e-06 - accuracy: 1.0000 -
val_loss: 0.0043 - val_accuracy: 0.9986
```

```
Epoch 00009: saving model to epochs:009-val_acc:0.999.hdf7
Epoch 10/20
7352/7352 [==============================] - 23s 3ms/step - loss: 3.1222e-06 - accuracy: 1.0000 - val_loss: 0.0039 - val_accuracy: 0.9990

Epoch 00010: saving model to epochs:010-val_acc:0.999.hdf7
Epoch 11/20
7352/7352 [==============================] - 21s 3ms/step - loss: 4.5465e-06 - accuracy: 1.0000 - val_loss: 0.0042 - val_accuracy: 0.9986

Epoch 00011: saving model to epochs:011-val_acc:0.999.hdf7
Epoch 12/20
7352/7352 [==============================] - 21s 3ms/step - loss: 2.7394e-06 - accuracy: 1.0000 - val_loss: 0.0041 - val_accuracy: 0.9986

Epoch 00012: saving model to epochs:012-val_acc:0.999.hdf7
Epoch 13/20
7352/7352 [==============================] - 24s 3ms/step - loss: 1.2381e-06 - accuracy: 1.0000 - val_loss: 0.0044 - val_accuracy: 0.9986

Epoch 00013: saving model to epochs:013-val_acc:0.999.hdf7
Epoch 14/20
7352/7352 [==============================] - 22s 3ms/step - loss: 1.0768e-06 - accuracy: 1.0000 - val_loss: 0.0042 - val_accuracy: 0.9986

Epoch 00014: saving model to epochs:014-val_acc:0.999.hdf7
Epoch 15/20
7352/7352 [==============================] - 22s 3ms/step - loss: 1.1284e-06 - accuracy: 1.0000 - val_loss: 0.0055 - val_accuracy: 0.9986

Epoch 00015: saving model to epochs:015-val_acc:0.999.hdf7
Epoch 16/20
7352/7352 [==============================] - 22s 3ms/step - loss: 1.3785e-06 - accuracy: 1.0000 - val_loss: 0.0051 - val_accuracy: 0.9986

Epoch 00016: saving model to epochs:016-val_acc:0.999.hdf7
Epoch 17/20
7352/7352 [==============================] - 22s 3ms/step - loss: 6.9404e-07 - accuracy: 1.0000 - val_loss: 0.0047 - val_accuracy: 0.9986

Epoch 00017: saving model to epochs:017-val_acc:0.999.hdf7
Epoch 18/20
7352/7352 [==============================] - 22s 3ms/step - loss: 5.4426e-07 - accuracy: 1.0000 - val_loss: 0.0044 - val_accuracy: 0.9986

Epoch 00018: saving model to epochs:018-val_acc:0.999.hdf7
Epoch 19/20
7352/7352 [==============================] - 23s 3ms/step - loss: 4.5994e-07 - accuracy: 1.0000 - val_loss: 0.0043 - val_accuracy: 0.9986

Epoch 00019: saving model to epochs:019-val_acc:0.999.hdf7
Epoch 20/20
7352/7352 [==============================] - 23s 3ms/step - loss: 3.5206e-07 - accuracy: 1.0000 - val_loss: 0.0040 - val_accuracy: 0.9990

Epoch 00020: saving model to epochs:020-val_acc:0.999.hdf7
```

Out[205]:

```
<keras.callbacks.callbacks.History at 0x1a6989cb00>
```

In [22]:

```
best_model_division = load_model('epochs:009-val_acc:0.999.hdf7')
```

```
WARNING:tensorflow:From /Users/bhawesh/anaconda3/lib/python3.7/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.
```

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 [10]:

```
# Importing tensorflow
```

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

```
/Users/bhawesh/anaconda3/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:526: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/Users/bhawesh/anaconda3/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:527: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/Users/bhawesh/anaconda3/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:528: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/Users/bhawesh/anaconda3/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:529: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/Users/bhawesh/anaconda3/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:530: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/Users/bhawesh/anaconda3/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:535: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

In [11]:

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

In [12]:

```
# Import Keras
import os
os.environ['KMP_DUPLICATE_LIB_OK']='True'
from tensorflow.keras import backend as K
sess = tf.Session(graph=tf.get_default_graph(), config=session_conf)
K.set_session(sess)
```

In [21]:

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

In [14]:

```
# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 64
```

In [15]:

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

In [16]:

```
# Loading the train and test data
```

```
X_train, X_test, Y_train, Y_test = load_data()
```

```
/Users/bhawesh/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:12: FutureWarning:
Method .as_matrix will be removed in a future version. Use .values instead.
  if sys.path[0] == '':
/Users/bhawesh/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:11: FutureWarning:
Method .as_matrix will be removed in a future version. Use .values instead.
  # This is added back by InteractiveShellApp.init_path()
```

In [17]:

```
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
```

In [18]:

```
X_train[0].shape
```

Out[18]:

```
(128, 9)
```

In [34]:

```
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D
from keras.layers import Dense, Activation, Flatten
from keras.layers import LeakyReLU
model1 = Sequential()
model1.add(Conv1D(filters=18,kernel_size=1, activation=LeakyReLU(alpha=0.3)
                 , input_shape=(timesteps,input_dim),kernel_initializer='he_uniform'))
model1.add(MaxPooling1D(pool_size=2))
model1.add(Conv1D(filters=36, kernel_size=3,kernel_initializer='he_uniform' ,activation=LeakyReLU(a
lpha=0.3)))
model1.add(Dropout(0.5))
model1.add(Conv1D(filters=144, kernel_size=3,kernel_initializer='he_uniform' ,activation=LeakyReLU(
alpha=0.3),strides=2))
model1.add(MaxPooling1D(pool_size=2,strides=2))
model1.add(Dropout(0.5))

model1.add(Flatten())
model1.add(Dense(32, activation='relu'))
model1.add(Dense(3, activation='sigmoid'))
model1.summary()
```

```
/Users/bhawesh/anaconda3/lib/python3.7/site-packages/keras/activations.py:235: UserWarning: Do not
pass a layer instance (such as LeakyReLU) as the activation argument of another layer. Instead, ad
vanced activation layers should be used just like any other layer in a model.
  identifier=identifier.__class__.__name__))
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 128, 18)           180
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 64, 18)            0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 62, 36)            1980
_________________________________________________________________
dropout_3 (Dropout)          (None, 62, 36)            0
_________________________________________________________________
```

```
conv1d_3 (Conv1D)            (None, 30, 144)           15696
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 15, 144)           0
_________________________________________________________________
dropout_4 (Dropout)          (None, 15, 144)           0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2160)              0
_________________________________________________________________
dense_2 (Dense)              (None, 32)                69152
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 99
=================================================================
Total params: 87,107
Trainable params: 87,107
Non-trainable params: 0
_________________________________________________________________
```

In [44]:

```
model1 = Sequential()
model1.add(Conv1D(filters=64, kernel_size=5, activation='relu',kernel_initializer='lecun_uniform',i
nput_shape=(128,9)))
model1.add(Conv1D(filters=32, kernel_size=3, activation='relu',kernel_initializer='lecun_uniform'))
model1.add(Dropout(0.45))
model1.add(MaxPooling1D(pool_size=3))
model1.add(Flatten())
model1.add(Dense(32, activation='relu'))
model1.add(Dense(3, activation='softmax'))
model1.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_6 (Conv1D)            (None, 124, 64)           2944
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 122, 32)           6176
_________________________________________________________________
dropout_6 (Dropout)          (None, 122, 32)           0
_________________________________________________________________
max_pooling1d_4 (MaxPooling1 (None, 40, 32)            0
_________________________________________________________________
flatten_3 (Flatten)          (None, 1280)              0
_________________________________________________________________
dense_6 (Dense)              (None, 32)                40992
_________________________________________________________________
dense_7 (Dense)              (None, 3)                 99
=================================================================
Total params: 50,211
Trainable params: 50,211
Non-trainable params: 0
_________________________________________________________________
```

In [23]:

```
Y_pred_train=best_model_division.predict(X_train)
Y_pred_test=best_model_division.predict(X_test)
```

In [25]:

```
Y_pred_train =  np.argmax(Y_pred_train, axis=1)
Y_pred_test =  np.argmax(Y_pred_test, axis=1)
```

In [26]:

```
Y_static_tr=Y_train[Y_pred_train==1]
Y_dynamic_tr=Y_train[Y_pred_train==0]
Y_static_test=Y_test[Y_pred_test==1]
Y_dynamic_test=Y_test[Y_pred_test==0]
```

In [39]:

```
Y_dynamic_tr=Y_dynamic_tr[0:len(Y_dynamic_tr),0:3]
Y_dynamic_test=Y_dynamic_test[0:len(Y_dynamic_test),0:3]
Y_static_tr=Y_static_tr[0:len(Y_static_tr),3:6]
Y_static_test=Y_static_test[0:len(Y_static_test),3:6]
```

In [28]:

```
X_static_tr=X_train[Y_pred_train==1]
X_dynamic_tr=X_train[Y_pred_train==0]
X_static_test=X_test[Y_pred_test==1]
X_dynamic_test=X_test[Y_pred_test==0]
```

In [ ]:

```

```

In [29]:

```
from keras.callbacks import *
filepath="epochs:{epoch:03d}-val_acc:{val_accuracy:.3f}.hdf5"
checkpoint_2 = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1, mode='max')
```

In [30]:

```
callbacks_list = [checkpoint_2]
```

In [45]:

```
model1.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [46]:

```
history=model1.fit(X_dynamic_tr,
          Y_dynamic_tr,
          batch_size=20,
          validation_data=(X_dynamic_test, Y_dynamic_test),
          epochs=20,callbacks=callbacks_list)
```

```
Train on 3285 samples, validate on 1391 samples
Epoch 1/20
3285/3285 [==============================] - 12s 4ms/step - loss: 0.4729 - accuracy: 0.7988 - val_
loss: 0.3071 - val_accuracy: 0.9037

Epoch 00001: saving model to epochs:001-val_acc:0.904.hdf5
Epoch 2/20
3285/3285 [==============================] - 10s 3ms/step - loss: 0.0362 - accuracy: 0.9903 - val_
loss: 0.2010 - val_accuracy: 0.9303

Epoch 00002: saving model to epochs:002-val_acc:0.930.hdf5
Epoch 3/20
3285/3285 [==============================] - 10s 3ms/step - loss: 0.0157 - accuracy: 0.9954 - val_
loss: 0.3736 - val_accuracy: 0.8778

Epoch 00003: saving model to epochs:003-val_acc:0.878.hdf5
Epoch 4/20
3285/3285 [==============================] - 10s 3ms/step - loss: 0.0099 - accuracy: 0.9976 - val_
loss: 0.1543 - val_accuracy: 0.9490

Epoch 00004: saving model to epochs:004-val_acc:0.949.hdf5
Epoch 5/20
3285/3285 [==============================] - 10s 3ms/step - loss: 9.9896e-04 - accuracy: 1.0000 -
val_loss: 0.1406 - val_accuracy: 0.9561

Epoch 00005: saving model to epochs:005-val_acc:0.956.hdf5
Epoch 6/20
3285/3285 [==============================] - 10s 3ms/step - loss: 5.9479e-04 - accuracy: 1.0000 -
val_loss: 0.1455 - val_accuracy: 0.9547

Epoch 00006: saving model to epochs:006-val_acc:0.955.hdf5
Epoch 7/20
```

```
3285/3285 [==============================] - 10s 3ms/step - loss: 4.6674e-04 - accuracy: 1.0000 -
val_loss: 0.1500 - val_accuracy: 0.9547

Epoch 00007: saving model to epochs:007-val_acc:0.955.hdf5
Epoch 8/20
3285/3285 [==============================] - 10s 3ms/step - loss: 2.3807e-04 - accuracy: 1.0000 -
val_loss: 0.1495 - val_accuracy: 0.9540

Epoch 00008: saving model to epochs:008-val_acc:0.954.hdf5
Epoch 9/20
3285/3285 [==============================] - 10s 3ms/step - loss: 0.0012 - accuracy: 1.0000 - val_
loss: 0.1095 - val_accuracy: 0.9648

Epoch 00009: saving model to epochs:009-val_acc:0.965.hdf5
Epoch 10/20
3285/3285 [==============================] - 11s 3ms/step - loss: 2.1861e-04 - accuracy: 1.0000 -
val_loss: 0.1404 - val_accuracy: 0.9590

Epoch 00010: saving model to epochs:010-val_acc:0.959.hdf5
Epoch 11/20
3285/3285 [==============================] - 11s 3ms/step - loss: 2.1276e-04 - accuracy: 1.0000 -
val_loss: 0.1587 - val_accuracy: 0.9569

Epoch 00011: saving model to epochs:011-val_acc:0.957.hdf5
Epoch 12/20
3285/3285 [==============================] - 11s 3ms/step - loss: 1.3620e-04 - accuracy: 1.0000 -
val_loss: 0.1735 - val_accuracy: 0.9554

Epoch 00012: saving model to epochs:012-val_acc:0.955.hdf5
Epoch 13/20
3285/3285 [==============================] - 11s 3ms/step - loss: 9.3560e-05 - accuracy: 1.0000 -
val_loss: 0.1607 - val_accuracy: 0.9605

Epoch 00013: saving model to epochs:013-val_acc:0.960.hdf5
Epoch 14/20
3285/3285 [==============================] - 11s 3ms/step - loss: 0.0064 - accuracy: 0.9985 - val_
loss: 0.4949 - val_accuracy: 0.8914

Epoch 00014: saving model to epochs:014-val_acc:0.891.hdf5
Epoch 15/20
3285/3285 [==============================] - 10s 3ms/step - loss: 0.0280 - accuracy: 0.9924 - val_
loss: 0.3351 - val_accuracy: 0.9281

Epoch 00015: saving model to epochs:015-val_acc:0.928.hdf5
Epoch 16/20
3285/3285 [==============================] - 11s 3ms/step - loss: 0.0040 - accuracy: 0.9997 - val_
loss: 0.2164 - val_accuracy: 0.9475

Epoch 00016: saving model to epochs:016-val_acc:0.948.hdf5
Epoch 17/20
3285/3285 [==============================] - 11s 3ms/step - loss: 8.2682e-04 - accuracy: 1.0000 -
val_loss: 0.1670 - val_accuracy: 0.9590

Epoch 00017: saving model to epochs:017-val_acc:0.959.hdf5
Epoch 18/20
3285/3285 [==============================] - 11s 3ms/step - loss: 1.4110e-04 - accuracy: 1.0000 -
val_loss: 0.1244 - val_accuracy: 0.9641

Epoch 00018: saving model to epochs:018-val_acc:0.964.hdf5
Epoch 19/20
3285/3285 [==============================] - 10s 3ms/step - loss: 1.0327e-04 - accuracy: 1.0000 -
val_loss: 0.1471 - val_accuracy: 0.9633

Epoch 00019: saving model to epochs:019-val_acc:0.963.hdf5
Epoch 20/20
3285/3285 [==============================] - 11s 3ms/step - loss: 4.7334e-05 - accuracy: 1.0000 -
val_loss: 0.1413 - val_accuracy: 0.9648

Epoch 00020: saving model to epochs:020-val_acc:0.965.hdf5
```

In [47]:

```
best_model_d = load_model('epochs:020-val_acc:0.965.hdf5')
```

In [48]:

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

# 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'])
```

In [51]:

```
print(confusion_matrix(Y_dynamic_test, best_model_d.predict(X_dynamic_test)))
```

```
Pred                WALKING  WALKING_DOWNSTAIRS  WALKING_UPSTAIRS
True
WALKING                 478                  18                 4
WALKING_DOWNSTAIRS        2                 418                 0
WALKING_UPSTAIRS          1                  24               446
```

In [55]:

```
score = best_model_d.evaluate(X_dynamic_test, Y_dynamic_test)
```

```
1391/1391 [==============================] - 2s 1ms/step
```

In [56]:

```
score
```

Out[56]:

```
[0.1413168757945732, 0.9647735357284546]
```

Static

In [90]:

```
Y_train_static=Y_train_static[0:len(Y_train_static),3:6]
Y_test_static=Y_test_static[0:len(Y_test_static),3:6]
```

In [19]:

```
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D
from keras.layers import Dense, Activation, Flatten
from keras.layers import LeakyReLU
model = Sequential()
model.add(Conv1D(filters=18,kernel_size=1, activation=LeakyReLU(alpha=0.3)
                 , input_shape=(timesteps,input_dim),kernel_initializer='glorot_uniform'))
model.add(MaxPooling1D(pool_size=2))
model.add(Conv1D(filters=36, kernel_size=3,kernel_initializer='glorot_uniform' ,activation=LeakyReL
U(alpha=0.3)))
model.add(Dropout(0.5))
model.add(Conv1D(filters=144, kernel_size=3,kernel_initializer='glorot_uniform' ,activation=LeakyRe
LU(alpha=0.3),strides=2))
model.add(MaxPooling1D(pool_size=2,strides=2))
model.add(Dropout(0.5))

model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(Dense(3, activation='sigmoid'))
model.summary()
```

```
WARNING:tensorflow:From /Users/bhawesh/anaconda3/lib/python3.7/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.
```

```
/Users/bhawesh/anaconda3/lib/python3.7/site-packages/keras/activations.py:235: UserWarning: Do not pass a layer instance (such as LeakyReLU) as the activation argument of another layer. Instead, advanced activation layers should be used just like any other layer in a model.
  identifier=identifier.__class__.__name__))
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv1d_1 (Conv1D)            (None, 128, 18)           180       
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 64, 18)            0         
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 62, 36)            1980      
_________________________________________________________________
dropout_1 (Dropout)          (None, 62, 36)            0         
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 30, 144)           15696     
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 15, 144)           0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 15, 144)           0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 2160)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 32)                69152     
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 99        
=================================================================
Total params: 87,107
Trainable params: 87,107
Non-trainable params: 0
_________________________________________________________________
```

In [167]:

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

In [67]:

```
history=model.fit(X_static_tr,
          Y_train_static,
          batch_size=300,
          validation_data=(X_static_test, Y_static_test),
          epochs=15)
```

```
Train on 4067 samples, validate on 1560 samples
Epoch 1/15
4067/4067 [==============================] - 16s 4ms/step - loss: 0.3267 - accuracy: 0.8439 - val_loss: 0.3313 - val_accuracy: 0.8577
Epoch 2/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.2088 - accuracy: 0.9085 - val_loss: 0.4369 - val_accuracy: 0.8327
Epoch 3/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.2031 - accuracy: 0.9112 - val_loss: 0.3389 - val_accuracy: 0.8885
Epoch 4/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.1989 - accuracy: 0.9090 - val_loss: 0.3526 - val_accuracy: 0.8679
Epoch 5/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.1902 - accuracy: 0.9157 - val_loss: 0.3836 - val_accuracy: 0.8744
Epoch 6/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.1905 - accuracy: 0.9216 - val_loss: 0.3504 - val_accuracy: 0.8891
```

```
Epoch 7/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.1869 - accuracy: 0.9181 - val_
loss: 0.3327 - val_accuracy: 0.8782
Epoch 8/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.1853 - accuracy: 0.9240 - val_
loss: 0.3239 - val_accuracy: 0.8853
Epoch 9/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.1922 - accuracy: 0.9191 - val_
loss: 0.3431 - val_accuracy: 0.8769
Epoch 10/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.1870 - accuracy: 0.9216 - val_
loss: 0.3342 - val_accuracy: 0.8814
Epoch 11/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.1805 - accuracy: 0.9225 - val_
loss: 0.3906 - val_accuracy: 0.8885
Epoch 12/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.1771 - accuracy: 0.9245 - val_
loss: 0.3355 - val_accuracy: 0.8872
Epoch 13/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.1991 - accuracy: 0.9191 - val_
loss: 0.3002 - val_accuracy: 0.8885
Epoch 14/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.1796 - accuracy: 0.9240 - val_
loss: 0.3527 - val_accuracy: 0.8936
Epoch 15/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.1747 - accuracy: 0.9250 - val_
loss: 0.3506 - val_accuracy: 0.8878
```

In [57]:

```
model = Sequential()
model.add(Conv1D(filters=64, kernel_size=5, activation='relu',kernel_initializer='glorot_uniform',i
nput_shape=(128,9)))
model.add(Conv1D(filters=32, kernel_size=3, activation='relu',kernel_initializer='glorot_uniform'))
model.add(Dropout(0.45))
model.add(MaxPooling1D(pool_size=3))
model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_8 (Conv1D)            (None, 124, 64)           2944
_________________________________________________________________
conv1d_9 (Conv1D)            (None, 122, 32)           6176
_________________________________________________________________
dropout_7 (Dropout)          (None, 122, 32)           0
_________________________________________________________________
max_pooling1d_5 (MaxPooling1 (None, 40, 32)            0
_________________________________________________________________
flatten_4 (Flatten)          (None, 1280)              0
_________________________________________________________________
dense_8 (Dense)              (None, 32)                40992
_________________________________________________________________
dense_9 (Dense)              (None, 3)                 99
=================================================================
Total params: 50,211
Trainable params: 50,211
Non-trainable params: 0
_________________________________________________________________
```

In [60]:

```
from keras.callbacks import *
filepath="epochs:{epoch:03d}-val_acc:{val_accuracy:.3f}.hdf6"
checkpoint_2 = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1, mode='max')
```

In [61]:

```
callbacks_list = [checkpoint_2]
```

In [62]:

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

In [64]:

```
history=model.fit(X_static_tr,
          Y_static_tr,
          batch_size=50,
          validation_data=(X_static_test, Y_static_test),
          epochs=15,callbacks=callbacks_list)
```

```
Train on 4067 samples, validate on 1556 samples
Epoch 1/15
4067/4067 [==============================] - 12s 3ms/step - loss: 0.2928 - accuracy: 0.8866 - val_
loss: 0.3406 - val_accuracy: 0.8599

Epoch 00001: saving model to epochs:001-val_acc:0.860.hdf6
Epoch 2/15
4067/4067 [==============================] - 10s 3ms/step - loss: 0.2238 - accuracy: 0.9036 - val_
loss: 0.3242 - val_accuracy: 0.8683

Epoch 00002: saving model to epochs:002-val_acc:0.868.hdf6
Epoch 3/15
4067/4067 [==============================] - 10s 2ms/step - loss: 0.1978 - accuracy: 0.9157 - val_
loss: 0.3095 - val_accuracy: 0.8817

Epoch 00003: saving model to epochs:003-val_acc:0.882.hdf6
Epoch 4/15
4067/4067 [==============================] - 10s 2ms/step - loss: 0.1867 - accuracy: 0.9186 - val_
loss: 0.3133 - val_accuracy: 0.8760

Epoch 00004: saving model to epochs:004-val_acc:0.876.hdf6
Epoch 5/15
4067/4067 [==============================] - 11s 3ms/step - loss: 0.1792 - accuracy: 0.9213 - val_
loss: 0.3005 - val_accuracy: 0.8721

Epoch 00005: saving model to epochs:005-val_acc:0.872.hdf6
Epoch 6/15
4067/4067 [==============================] - 10s 3ms/step - loss: 0.1724 - accuracy: 0.9223 - val_
loss: 0.3236 - val_accuracy: 0.8702

Epoch 00006: saving model to epochs:006-val_acc:0.870.hdf6
Epoch 7/15
4067/4067 [==============================] - 11s 3ms/step - loss: 0.1789 - accuracy: 0.9203 - val_
loss: 0.3563 - val_accuracy: 0.8650

Epoch 00007: saving model to epochs:007-val_acc:0.865.hdf6
Epoch 8/15
4067/4067 [==============================] - 10s 3ms/step - loss: 0.1664 - accuracy: 0.9255 - val_
loss: 0.3095 - val_accuracy: 0.8856

Epoch 00008: saving model to epochs:008-val_acc:0.886.hdf6
Epoch 9/15
4067/4067 [==============================] - 11s 3ms/step - loss: 0.1650 - accuracy: 0.9250 - val_
loss: 0.3105 - val_accuracy: 0.8798

Epoch 00009: saving model to epochs:009-val_acc:0.880.hdf6
Epoch 10/15
4067/4067 [==============================] - 11s 3ms/step - loss: 0.1556 - accuracy: 0.9294 - val_
loss: 0.3306 - val_accuracy: 0.8933

Epoch 00010: saving model to epochs:010-val_acc:0.893.hdf6
Epoch 11/15
4067/4067 [==============================] - 10s 2ms/step - loss: 0.1358 - accuracy: 0.9368 - val_
loss: 0.3567 - val_accuracy: 0.8843

Epoch 00011: saving model to epochs:011-val_acc:0.884.hdf6
Epoch 12/15
4067/4067 [==============================] - 10s 2ms/step - loss: 0.1270 - accuracy: 0.9400 - val_
loss: 0.3717 - val_accuracy: 0.8753

Epoch 00012: saving model to epochs:012-val_acc:0.875.hdf6
Epoch 13/15
```

```
Epoch 13/15
4067/4067 [==============================] - 10s 2ms/step - loss: 0.1259 - accuracy: 0.9412 - val_
loss: 0.3280 - val_accuracy: 0.9055

Epoch 00013: saving model to epochs:013-val_acc:0.906.hdf6
Epoch 14/15
4067/4067 [==============================] - 10s 2ms/step - loss: 0.1185 - accuracy: 0.9454 - val_
loss: 0.3891 - val_accuracy: 0.8946

Epoch 00014: saving model to epochs:014-val_acc:0.895.hdf6
Epoch 15/15
4067/4067 [==============================] - 10s 2ms/step - loss: 0.1066 - accuracy: 0.9498 - val_
loss: 0.3448 - val_accuracy: 0.8901

Epoch 00015: saving model to epochs:015-val_acc:0.890.hdf6
```

In [65]:

```
history=model.fit(X_static_tr,
          Y_static_tr,
          batch_size=50,
          validation_data=(X_static_test, Y_static_test),
          epochs=15,callbacks=callbacks_list)
```

```
Train on 4067 samples, validate on 1556 samples
Epoch 1/15
4067/4067 [==============================] - 10s 2ms/step - loss: 0.1106 - accuracy: 0.9503 - val_
loss: 0.3441 - val_accuracy: 0.8959

Epoch 00001: saving model to epochs:001-val_acc:0.896.hdf6
Epoch 2/15
4067/4067 [==============================] - 9s 2ms/step - loss: 0.1069 - accuracy: 0.9489 - val_l
oss: 0.3390 - val_accuracy: 0.9042

Epoch 00002: saving model to epochs:002-val_acc:0.904.hdf6
Epoch 3/15
4067/4067 [==============================] - 10s 3ms/step - loss: 0.0985 - accuracy: 0.9550 - val_
loss: 0.4258 - val_accuracy: 0.8888

Epoch 00003: saving model to epochs:003-val_acc:0.889.hdf6
Epoch 4/15
4067/4067 [==============================] - 10s 3ms/step - loss: 0.0901 - accuracy: 0.9607 - val_
loss: 0.4983 - val_accuracy: 0.8830

Epoch 00004: saving model to epochs:004-val_acc:0.883.hdf6
Epoch 5/15
4067/4067 [==============================] - 10s 2ms/step - loss: 0.0913 - accuracy: 0.9594 - val_
loss: 0.5533 - val_accuracy: 0.8772

Epoch 00005: saving model to epochs:005-val_acc:0.877.hdf6
Epoch 6/15
4067/4067 [==============================] - 10s 2ms/step - loss: 0.0892 - accuracy: 0.9597 - val_
loss: 0.3399 - val_accuracy: 0.8914

Epoch 00006: saving model to epochs:006-val_acc:0.891.hdf6
Epoch 7/15
4067/4067 [==============================] - 9s 2ms/step - loss: 0.1008 - accuracy: 0.9567 - val_l
oss: 0.3362 - val_accuracy: 0.8869

Epoch 00007: saving model to epochs:007-val_acc:0.887.hdf6
Epoch 8/15
4067/4067 [==============================] - 9s 2ms/step - loss: 0.1028 - accuracy: 0.9584 - val_l
oss: 0.2660 - val_accuracy: 0.9165

Epoch 00008: saving model to epochs:008-val_acc:0.916.hdf6
Epoch 9/15
4067/4067 [==============================] - 9s 2ms/step - loss: 0.0864 - accuracy: 0.9639 - val_l
oss: 0.3291 - val_accuracy: 0.8997

Epoch 00009: saving model to epochs:009-val_acc:0.900.hdf6
Epoch 10/15
4067/4067 [==============================] - 9s 2ms/step - loss: 0.0781 - accuracy: 0.9626 - val_l
oss: 0.3717 - val_accuracy: 0.8817

Epoch 00010: saving model to epochs:010-val_acc:0.882.hdf6
Epoch 11/15
```

```
4067/4067 [==============================] - 9s 2ms/step - loss: 0.0808 - accuracy: 0.9639 - val_l
oss: 0.3663 - val_accuracy: 0.9049

Epoch 00011: saving model to epochs:011-val_acc:0.905.hdf6
Epoch 12/15
4067/4067 [==============================] - 9s 2ms/step - loss: 0.0657 - accuracy: 0.9705 - val_l
oss: 0.3521 - val_accuracy: 0.9023

Epoch 00012: saving model to epochs:012-val_acc:0.902.hdf6
Epoch 13/15
4067/4067 [==============================] - 10s 3ms/step - loss: 0.0725 - accuracy: 0.9693 - val_
loss: 0.3425 - val_accuracy: 0.8850

Epoch 00013: saving model to epochs:013-val_acc:0.885.hdf6
Epoch 14/15
4067/4067 [==============================] - 10s 3ms/step - loss: 0.1182 - accuracy: 0.9498 - val_
loss: 0.3595 - val_accuracy: 0.9120

Epoch 00014: saving model to epochs:014-val_acc:0.912.hdf6
Epoch 15/15
4067/4067 [==============================] - 9s 2ms/step - loss: 0.0676 - accuracy: 0.9678 - val_l
oss: 0.3424 - val_accuracy: 0.9049

Epoch 00015: saving model to epochs:015-val_acc:0.905.hdf6
```

In [66]:

```
history=model.fit(X_static_tr,
          Y_static_tr,
          batch_size=20,
          validation_data=(X_static_test, Y_static_test),
          epochs=15,callbacks=callbacks_list)
```

```
Train on 4067 samples, validate on 1556 samples
Epoch 1/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.1032 - accuracy: 0.9584 - val_
loss: 0.3711 - val_accuracy: 0.8843

Epoch 00001: saving model to epochs:001-val_acc:0.884.hdf6
Epoch 2/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.1028 - accuracy: 0.9584 - val_
loss: 0.3037 - val_accuracy: 0.9132

Epoch 00002: saving model to epochs:002-val_acc:0.913.hdf6
Epoch 3/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.1246 - accuracy: 0.9530 - val_
loss: 0.2716 - val_accuracy: 0.9017

Epoch 00003: saving model to epochs:003-val_acc:0.902.hdf6
Epoch 4/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.0846 - accuracy: 0.9614 - val_
loss: 0.2659 - val_accuracy: 0.9068

Epoch 00004: saving model to epochs:004-val_acc:0.907.hdf6
Epoch 5/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.0705 - accuracy: 0.9678 - val_
loss: 0.3229 - val_accuracy: 0.9120

Epoch 00005: saving model to epochs:005-val_acc:0.912.hdf6
Epoch 6/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.0748 - accuracy: 0.9641 - val_
loss: 0.2726 - val_accuracy: 0.9177

Epoch 00006: saving model to epochs:006-val_acc:0.918.hdf6
Epoch 7/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.0721 - accuracy: 0.9658 - val_
loss: 0.2923 - val_accuracy: 0.9216

Epoch 00007: saving model to epochs:007-val_acc:0.922.hdf6
Epoch 8/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.0724 - accuracy: 0.9666 - val_
loss: 0.3132 - val_accuracy: 0.9203

Epoch 00008: saving model to epochs:008-val_acc:0.920.hdf6
Epoch 9/15
```

```
4067/4067 [==============================] - 14s 3ms/step - loss: 0.0699 - accuracy: 0.9690 - val_
loss: 0.3696 - val_accuracy: 0.9036

Epoch 00009: saving model to epochs:009-val_acc:0.904.hdf6
Epoch 10/15
4067/4067 [==============================] - 11s 3ms/step - loss: 0.0708 - accuracy: 0.9700 - val_
loss: 0.3251 - val_accuracy: 0.9222

Epoch 00010: saving model to epochs:010-val_acc:0.922.hdf6
Epoch 11/15
4067/4067 [==============================] - 12s 3ms/step - loss: 0.0657 - accuracy: 0.9734 - val_
loss: 0.3887 - val_accuracy: 0.9036

Epoch 00011: saving model to epochs:011-val_acc:0.904.hdf6
Epoch 12/15
4067/4067 [==============================] - 14s 3ms/step - loss: 0.0735 - accuracy: 0.9715 - val_
loss: 0.3849 - val_accuracy: 0.9100

Epoch 00012: saving model to epochs:012-val_acc:0.910.hdf6
Epoch 13/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.0754 - accuracy: 0.9661 - val_
loss: 0.3262 - val_accuracy: 0.9229

Epoch 00013: saving model to epochs:013-val_acc:0.923.hdf6
Epoch 14/15
4067/4067 [==============================] - 14s 3ms/step - loss: 0.0610 - accuracy: 0.9739 - val_
loss: 0.3206 - val_accuracy: 0.9242

Epoch 00014: saving model to epochs:014-val_acc:0.924.hdf6
Epoch 15/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.0586 - accuracy: 0.9747 - val_
loss: 0.3893 - val_accuracy: 0.9023

Epoch 00015: saving model to epochs:015-val_acc:0.902.hdf6
```

In [68]:

```
history=model.fit(X_static_tr,
          Y_static_tr,
          batch_size=20,
          validation_data=(X_static_test, Y_static_test),
          epochs=15,callbacks=callbacks_list)
```

```
Train on 4067 samples, validate on 1556 samples
Epoch 1/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.0540 - accuracy: 0.9766 - val_
loss: 0.3624 - val_accuracy: 0.9235

Epoch 00001: saving model to epochs:001-val_acc:0.924.hdf6
Epoch 2/15
4067/4067 [==============================] - 10s 3ms/step - loss: 0.0595 - accuracy: 0.9739 - val_
loss: 0.3420 - val_accuracy: 0.9171

Epoch 00002: saving model to epochs:002-val_acc:0.917.hdf6
Epoch 3/15
4067/4067 [==============================] - 14s 3ms/step - loss: 0.0528 - accuracy: 0.9779 - val_
loss: 0.4323 - val_accuracy: 0.9242

Epoch 00003: saving model to epochs:003-val_acc:0.924.hdf6
Epoch 4/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.0537 - accuracy: 0.9779 - val_
loss: 0.4035 - val_accuracy: 0.9120

Epoch 00004: saving model to epochs:004-val_acc:0.912.hdf6
Epoch 5/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.0551 - accuracy: 0.9754 - val_
loss: 0.4302 - val_accuracy: 0.9100

Epoch 00005: saving model to epochs:005-val_acc:0.910.hdf6
Epoch 6/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.0513 - accuracy: 0.9779 - val_
loss: 0.4064 - val_accuracy: 0.9190

Epoch 00006: saving model to epochs:006-val_acc:0.919.hdf6
Epoch 7/15
4067/4067 [==============================] - 11s 3ms/step - loss: 0.0413 - accuracy: 0.9823 - val_
```

```
4067/4067 [==============================] - 11s 3ms/step - loss: 0.0415 - accuracy: 0.9825 - val_
loss: 0.4155 - val_accuracy: 0.9254

Epoch 00007: saving model to epochs:007-val_acc:0.925.hdf6
Epoch 8/15
4067/4067 [==============================] - 11s 3ms/step - loss: 0.0417 - accuracy: 0.9808 - val_
loss: 0.3827 - val_accuracy: 0.9210

Epoch 00008: saving model to epochs:008-val_acc:0.921.hdf6
Epoch 9/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.0435 - accuracy: 0.9830 - val_
loss: 0.3935 - val_accuracy: 0.9229

Epoch 00009: saving model to epochs:009-val_acc:0.923.hdf6
Epoch 10/15
4067/4067 [==============================] - 14s 3ms/step - loss: 0.0418 - accuracy: 0.9825 - val_
loss: 0.3929 - val_accuracy: 0.9171

Epoch 00010: saving model to epochs:010-val_acc:0.917.hdf6
Epoch 11/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.0433 - accuracy: 0.9816 - val_
loss: 0.3933 - val_accuracy: 0.9299

Epoch 00011: saving model to epochs:011-val_acc:0.930.hdf6
Epoch 12/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.0447 - accuracy: 0.9823 - val_
loss: 0.4470 - val_accuracy: 0.9094

Epoch 00012: saving model to epochs:012-val_acc:0.909.hdf6
Epoch 13/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.0473 - accuracy: 0.9821 - val_
loss: 0.6168 - val_accuracy: 0.9132

Epoch 00013: saving model to epochs:013-val_acc:0.913.hdf6
Epoch 14/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.0486 - accuracy: 0.9818 - val_
loss: 0.7143 - val_accuracy: 0.8965

Epoch 00014: saving model to epochs:014-val_acc:0.897.hdf6
Epoch 15/15
4067/4067 [==============================] - 14s 3ms/step - loss: 0.0355 - accuracy: 0.9850 - val_
loss: 0.6869 - val_accuracy: 0.9087

Epoch 00015: saving model to epochs:015-val_acc:0.909.hdf6
```

In [70]:

```
history=model.fit(X_static_tr,
          Y_static_tr,
          batch_size=16,
          validation_data=(X_static_test, Y_static_test),
          epochs=15,callbacks=callbacks_list)
```

```
Train on 4067 samples, validate on 1556 samples
Epoch 1/15
4067/4067 [==============================] - 14s 3ms/step - loss: 0.0449 - accuracy: 0.9838 - val_
loss: 0.6696 - val_accuracy: 0.9055

Epoch 00001: saving model to epochs:001-val_acc:0.906.hdf6
Epoch 2/15
4067/4067 [==============================] - 14s 3ms/step - loss: 0.0650 - accuracy: 0.9828 - val_
loss: 0.6728 - val_accuracy: 0.9030

Epoch 00002: saving model to epochs:002-val_acc:0.903.hdf6
Epoch 3/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.1393 - accuracy: 0.9717 - val_
loss: 0.7011 - val_accuracy: 0.8972

Epoch 00003: saving model to epochs:003-val_acc:0.897.hdf6
Epoch 4/15
4067/4067 [==============================] - 12s 3ms/step - loss: 0.1116 - accuracy: 0.9796 - val_
loss: 0.7002 - val_accuracy: 0.9100

Epoch 00004: saving model to epochs:004-val_acc:0.910.hdf6
Epoch 5/15
4067/4067 [==============================] - 12s 3ms/step - loss: 0.1032 - accuracy: 0.9830 - val
```

```
loss: 0.6342 - val_accuracy: 0.9139

Epoch 00005: saving model to epochs:005-val_acc:0.914.hdf6
Epoch 6/15
4067/4067 [==============================] - 14s 4ms/step - loss: 0.0980 - accuracy: 0.9840 - val_
loss: 0.7298 - val_accuracy: 0.9049

Epoch 00006: saving model to epochs:006-val_acc:0.905.hdf6
Epoch 7/15
4067/4067 [==============================] - 14s 4ms/step - loss: 0.0976 - accuracy: 0.9840 - val_
loss: 0.6854 - val_accuracy: 0.9055

Epoch 00007: saving model to epochs:007-val_acc:0.906.hdf6
Epoch 8/15
4067/4067 [==============================] - 14s 4ms/step - loss: 0.1008 - accuracy: 0.9816 - val_
loss: 0.6970 - val_accuracy: 0.8991

Epoch 00008: saving model to epochs:008-val_acc:0.899.hdf6
Epoch 9/15
4067/4067 [==============================] - 14s 3ms/step - loss: 0.0918 - accuracy: 0.9867 - val_
loss: 0.7512 - val_accuracy: 0.9017

Epoch 00009: saving model to epochs:009-val_acc:0.902.hdf6
Epoch 10/15
4067/4067 [==============================] - 15s 4ms/step - loss: 0.1010 - accuracy: 0.9828 - val_
loss: 0.6946 - val_accuracy: 0.9030

Epoch 00010: saving model to epochs:010-val_acc:0.903.hdf6
Epoch 11/15
4067/4067 [==============================] - 14s 4ms/step - loss: 0.1076 - accuracy: 0.9835 - val_
loss: 0.7945 - val_accuracy: 0.8940

Epoch 00011: saving model to epochs:011-val_acc:0.894.hdf6
Epoch 12/15
4067/4067 [==============================] - 14s 3ms/step - loss: 0.0966 - accuracy: 0.9838 - val_
loss: 0.7454 - val_accuracy: 0.9132

Epoch 00012: saving model to epochs:012-val_acc:0.913.hdf6
Epoch 13/15
4067/4067 [==============================] - 14s 3ms/step - loss: 0.0928 - accuracy: 0.9862 - val_
loss: 0.7262 - val_accuracy: 0.8997

Epoch 00013: saving model to epochs:013-val_acc:0.900.hdf6
Epoch 14/15
4067/4067 [==============================] - 14s 3ms/step - loss: 0.1073 - accuracy: 0.9816 - val_
loss: 0.7007 - val_accuracy: 0.9094

Epoch 00014: saving model to epochs:014-val_acc:0.909.hdf6
Epoch 15/15
4067/4067 [==============================] - 14s 3ms/step - loss: 0.0928 - accuracy: 0.9860 - val_
loss: 0.6904 - val_accuracy: 0.9100

Epoch 00015: saving model to epochs:015-val_acc:0.910.hdf6
```

In [73]:

```
history=model.fit(X_static_tr,
          Y_static_tr,
          batch_size=20,
          validation_data=(X_static_test, Y_static_test),
          epochs=15,callbacks=callbacks_list)
```

```
Train on 4067 samples, validate on 1556 samples
Epoch 1/15
4067/4067 [==============================] - 12s 3ms/step - loss: 0.0773 - accuracy: 0.9921 - val_
loss: 0.7688 - val_accuracy: 0.9177

Epoch 00001: saving model to epochs:001-val_acc:0.918.hdf6
Epoch 2/15
4067/4067 [==============================] - 11s 3ms/step - loss: 0.0753 - accuracy: 0.9934 - val_
loss: 0.7692 - val_accuracy: 0.9139

Epoch 00002: saving model to epochs:002-val_acc:0.914.hdf6
Epoch 3/15
4067/4067 [==============================] - 11s 3ms/step - loss: 0.0744 - accuracy: 0.9931 - val_
```

```
loss: 0.7468 - val_accuracy: 0.9107

Epoch 00003: saving model to epochs:003-val_acc:0.911.hdf6
Epoch 4/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.0796 - accuracy: 0.9904 - val_
loss: 0.7810 - val_accuracy: 0.9126

Epoch 00004: saving model to epochs:004-val_acc:0.913.hdf6
Epoch 5/15
4067/4067 [==============================] - 14s 3ms/step - loss: 0.0826 - accuracy: 0.9914 - val_
loss: 0.7787 - val_accuracy: 0.9152

Epoch 00005: saving model to epochs:005-val_acc:0.915.hdf6
Epoch 6/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.0841 - accuracy: 0.9882 - val_
loss: 0.7761 - val_accuracy: 0.9132

Epoch 00006: saving model to epochs:006-val_acc:0.913.hdf6
Epoch 7/15
4067/4067 [==============================] - 14s 3ms/step - loss: 0.0704 - accuracy: 0.9956 - val_
loss: 0.8016 - val_accuracy: 0.9120

Epoch 00007: saving model to epochs:007-val_acc:0.912.hdf6
Epoch 8/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.0706 - accuracy: 0.9951 - val_
loss: 0.7982 - val_accuracy: 0.9145

Epoch 00008: saving model to epochs:008-val_acc:0.915.hdf6
Epoch 9/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.0889 - accuracy: 0.9907 - val_
loss: 0.7980 - val_accuracy: 0.9087

Epoch 00009: saving model to epochs:009-val_acc:0.909.hdf6
Epoch 10/15
4067/4067 [==============================] - 14s 3ms/step - loss: 0.0861 - accuracy: 0.9909 - val_
loss: 0.8433 - val_accuracy: 0.9126

Epoch 00010: saving model to epochs:010-val_acc:0.913.hdf6
Epoch 11/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.0757 - accuracy: 0.9916 - val_
loss: 0.7659 - val_accuracy: 0.9094

Epoch 00011: saving model to epochs:011-val_acc:0.909.hdf6
Epoch 12/15
4067/4067 [==============================] - 14s 3ms/step - loss: 0.0669 - accuracy: 0.9953 - val_
loss: 0.8215 - val_accuracy: 0.9126

Epoch 00012: saving model to epochs:012-val_acc:0.913.hdf6
Epoch 13/15
4067/4067 [==============================] - 14s 3ms/step - loss: 0.1012 - accuracy: 0.9872 - val_
loss: 0.7749 - val_accuracy: 0.9062

Epoch 00013: saving model to epochs:013-val_acc:0.906.hdf6
Epoch 14/15
4067/4067 [==============================] - 13s 3ms/step - loss: 0.1057 - accuracy: 0.9798 - val_
loss: 0.7737 - val_accuracy: 0.9152

Epoch 00014: saving model to epochs:014-val_acc:0.915.hdf6
Epoch 15/15
4067/4067 [==============================] - 14s 3ms/step - loss: 0.0675 - accuracy: 0.9943 - val_
loss: 0.8425 - val_accuracy: 0.9010

Epoch 00015: saving model to epochs:015-val_acc:0.901.hdf6
```

In [74]:

```
best_model_s = load_model('epochs:011-val_acc:0.930.hdf6')
```

In [75]:

```
# Activities are the class labels
# It is a 6 class classification
ACTIVITIES = {
    0: 'SITTING',
    1: 'STANDING',
```

```
    2: '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'])
```

In [76]:

```
print(confusion_matrix(Y_static_test, model.predict(X_static_test)))
```

```
Pred      LAYING  SITTING  STANDING
True
LAYING       512        0        25
SITTING        0      389       102
STANDING       0       27       501
```

In [78]:

```
score = best_model_s.evaluate(X_static_test, Y_static_test)
```

```
1556/1556 [==============================] - 2s 1ms/step
```

In [79]:

```
score
```

Out[79]:

```
[0.39326874985088, 0.9299485683441162]
```

Final Pipelining

In [80]:

```
# 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"
]
# 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))

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 y
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 [81]:

```
X_train, X_test, Y_train, Y_test = load_data()
```

```
/Users/bhawesh/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:29: FutureWarning:
Method .as_matrix will be removed in a future version. Use .values instead.
```

In [82]:

```
len(Y_train)
```

Out[82]:

```
7352
```

In [83]:

```
Y_pred_train=best_model_division.predict(X_train)
Y_pred_test=best_model_division.predict(X_test)
```

In [84]:

```
Y_pred_train =  np.argmax(Y_pred_train, axis=1)
Y_pred_test =  np.argmax(Y_pred_test, axis=1)
```

In [85]:

```
X_static_tr=X_train[Y_pred_train==1]
X_dynamic_tr=X_train[Y_pred_train==0]
X_static_test=X_test[Y_pred_test==1]
X_dynamic_test=X_test[Y_pred_test==0]
```

In [86]:

```
Y_pred_static_tr=best_model_s.predict(X_static_tr)
Y_pred_static_test=best_model_s.predict(X_static_test)
```

In [88]:

```
Y_pred_dynamic_tr=best_model_d.predict(X_dynamic_tr)
Y_pred_dynamic_test=best_model_d.predict(X_dynamic_test)
```

In [89]:

```
Y_pred_dynamic_tr=np.argmax(Y_pred_dynamic_tr, axis=1)
Y_pred_dynamic_test=np.argmax(Y_pred_dynamic_test, axis=1)
Y_pred_static_tr=np.argmax(Y_pred_static_tr, axis=1)
Y_pred_static_test=np.argmax(Y_pred_static_test, axis=1)
```

In [90]:

```
Y_pred_static_tr=Y_pred_static_tr+4
Y_pred_static_test=Y_pred_static_test+4
```

In [91]:

```
Y_pred_dynamic_tr=Y_pred_dynamic_tr+1
Y_pred_dynamic_test=Y_pred_dynamic_test+1
```

In [92]:

```
k,j = 0,0
final_pred_tr = []
for i in Y_pred_train:
    if i == 1:
        final_pred_tr.append(Y_pred_static_tr[k])
        k = k + 1
    else:
        final_pred_tr.append(Y_pred_dynamic_tr[j])
        j = j + 1
```

In [93]:

```
k,j = 0,0
final_pred_test = []
for i in Y_pred_test:
    if i == 1:
        final_pred_test.append(Y_pred_static_test[k])
        k = k + 1
    else:
        final_pred_test.append(Y_pred_dynamic_test[j])
        j = j + 1
```

In [94]:

```
##accuracy of train and test
from sklearn.metrics import accuracy_score
print('Accuracy of train data',accuracy_score(Y_train,final_pred_tr))
print('Accuracy of validation data',accuracy_score(Y_test,final_pred_test))
```

```
Accuracy of train data 0.9944232861806311
Accuracy of validation data 0.9463861554122837
```

In [ ]: