Implemant HUMAN ACTIVITY RECOGNITION via LSTM with different Architecture

In [1]:

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

In [3]:

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

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

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

Data

In [4]:

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

In [5]:

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

In [6]:

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

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

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

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

In [7]:

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

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

In [8]:

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

    return X_train, X_test, y_train, y_test
```

In [9]:

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

```
C:\Users\HIMANSHU NEGI\Anaconda3\lib\site-packages\h5py\__init__.py:36: Fu
tureWarning: Conversion of the second argument of issubdtype from `float`
to `np.floating` is deprecated. In future, it will be treated as `np.float
64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
```

In [12]:

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

In [13]:

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

In [14]:

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

In [49]:

```
# Initializing parameters
epochs = 20
batch_size = 16
n_hidden2 = 18
n_hidden1 = 36
```

In [50]:

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

In [51]:

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

```
C:\Users\HIMANSHU NEGI\Anaconda3\lib\site-packages\ipykernel_launcher.py:1
2: FutureWarning: Method .as_matrix will be removed in a future version. U
se .values instead.
  if sys.path[0] == '':
```

In [52]:

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

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

```
128
9
7352
```

- Defining the Architecture of LSTM

(1) Model having 1 LSTM layer with 32 LSTM Units

In [29]:

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

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_11 (LSTM)               (None, 32)                5376      
_________________________________________________________________
dropout_6 (Dropout)          (None, 32)                0         
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 198       
=================================================================
Total params: 5,574
Trainable params: 5,574
Non-trainable params: 0
_________________________________________________________________
```

In [30]:

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

In [31]:

```
# 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 1/20
7352/7352 [==============================] - 74s 10ms/step - loss: 1.3657
- acc: 0.4223 - val_loss: 1.2432 - val_acc: 0.4476
Epoch 2/20
7352/7352 [==============================] - 72s 10ms/step - loss: 1.1265
- acc: 0.4933 - val_loss: 1.0647 - val_acc: 0.4472
Epoch 3/20
7352/7352 [==============================] - 69s 9ms/step - loss: 0.9105 -
acc: 0.5741 - val_loss: 0.9171 - val_acc: 0.5843
Epoch 4/20
7352/7352 [==============================] - 69s 9ms/step - loss: 0.7974 -
acc: 0.6465 - val_loss: 0.8289 - val_acc: 0.6016
Epoch 5/20
7352/7352 [==============================] - 69s 9ms/step - loss: 0.6969 -
acc: 0.6568 - val_loss: 0.7589 - val_acc: 0.6098
Epoch 6/20
7352/7352 [==============================] - 70s 10ms/step - loss: 0.6716
- acc: 0.6560 - val_loss: 0.8594 - val_acc: 0.6047
Epoch 7/20
7352/7352 [==============================] - 69s 9ms/step - loss: 0.7123 -
acc: 0.6536 - val_loss: 0.7254 - val_acc: 0.6166
Epoch 8/20
7352/7352 [==============================] - 75s 10ms/step - loss: 0.6294
- acc: 0.6831 - val_loss: 0.7531 - val_acc: 0.6094
Epoch 9/20
7352/7352 [==============================] - 69s 9ms/step - loss: 0.5643 -
acc: 0.7058 - val_loss: 0.7404 - val_acc: 0.6630
Epoch 10/20
7352/7352 [==============================] - 76s 10ms/step - loss: 0.5790
- acc: 0.7360 - val_loss: 0.5795 - val_acc: 0.7435
Epoch 11/20
7352/7352 [==============================] - 50s 7ms/step - loss: 0.5650 -
acc: 0.7646 - val_loss: 0.6684 - val_acc: 0.7703
Epoch 12/20
7352/7352 [==============================] - 50s 7ms/step - loss: 0.5135 -
acc: 0.7930 - val_loss: 0.5949 - val_acc: 0.7482
Epoch 13/20
7352/7352 [==============================] - 49s 7ms/step - loss: 0.4577 -
acc: 0.8089 - val_loss: 0.5362 - val_acc: 0.7889
Epoch 14/20
7352/7352 [==============================] - 51s 7ms/step - loss: 0.5020 -
acc: 0.8127 - val_loss: 0.4887 - val_acc: 0.7967
Epoch 15/20
7352/7352 [==============================] - 62s 8ms/step - loss: 0.4043 -
acc: 0.8430 - val_loss: 0.6778 - val_acc: 0.8066
Epoch 16/20
7352/7352 [==============================] - 48s 7ms/step - loss: 0.3501 -
acc: 0.8777 - val_loss: 0.4626 - val_acc: 0.8527
Epoch 17/20
7352/7352 [==============================] - 48s 7ms/step - loss: 0.3009 -
acc: 0.9061 - val_loss: 0.4234 - val_acc: 0.8724
Epoch 18/20
7352/7352 [==============================] - 50s 7ms/step - loss: 0.2400 -
acc: 0.9266 - val_loss: 0.4140 - val_acc: 0.8707
Epoch 19/20
7352/7352 [==============================] - 49s 7ms/step - loss: 0.2388 -
acc: 0.9246 - val_loss: 0.3638 - val_acc: 0.8860
Epoch 20/20
7352/7352 [==============================] - 46s 6ms/step - loss: 0.2132 -
acc: 0.9327 - val_loss: 0.4219 - val_acc: 0.8792
```

Out[31]:

```
<keras.callbacks.History at 0x1611c2a5fd0>
```

In [32]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAI
RS  \
True
LAYING                 512        0         0        0
0
SITTING                  1      385       102        2
0
STANDING                 0      101       429        2
0
WALKING                  0        1         3      485
4
WALKING_DOWNSTAIRS       0        0         0       27                  3
89
WALKING_UPSTAIRS         1        0         1       56
22

Pred                WALKING_UPSTAIRS
True
LAYING                            25
SITTING                            1
STANDING                           0
WALKING                            3
WALKING_DOWNSTAIRS                 4
WALKING_UPSTAIRS                 391
```

In [33]:

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

```
2947/2947 [==============================] - 2s 621us/step
```

In [34]:

```
score
```

Out[34]:

```
[0.4219474009271998, 0.8791991856124872]
```

(2) Model having 1 LSTM layer with 64 LSTM Units and 'rmsprop' as an optimizer

In [35]:

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

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_12 (LSTM)               (None, 64)                18944
_________________________________________________________________
dropout_7 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 390
=================================================================
Total params: 19,334
Trainable params: 19,334
Non-trainable params: 0
_________________________________________________________________
```

In [36]:

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

In [37]:

```
# 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 1/20
7352/7352 [==============================] - 54s 7ms/step - loss: 1.2354 -
acc: 0.4479 - val_loss: 1.2075 - val_acc: 0.3868
Epoch 2/20
7352/7352 [==============================] - 53s 7ms/step - loss: 1.0049 -
acc: 0.5355 - val_loss: 0.9058 - val_acc: 0.5826
Epoch 3/20
7352/7352 [==============================] - 53s 7ms/step - loss: 0.7873 -
acc: 0.6240 - val_loss: 0.7922 - val_acc: 0.6047
Epoch 4/20
7352/7352 [==============================] - 54s 7ms/step - loss: 0.7871 -
acc: 0.6310 - val_loss: 0.7214 - val_acc: 0.6115
Epoch 5/20
7352/7352 [==============================] - 54s 7ms/step - loss: 0.6317 -
acc: 0.6597 - val_loss: 0.7588 - val_acc: 0.6206
Epoch 6/20
7352/7352 [==============================] - 54s 7ms/step - loss: 0.5755 -
acc: 0.6814 - val_loss: 0.7319 - val_acc: 0.6237
Epoch 7/20
7352/7352 [==============================] - 54s 7ms/step - loss: 0.5225 -
acc: 0.7531 - val_loss: 0.6146 - val_acc: 0.7988
Epoch 8/20
7352/7352 [==============================] - 54s 7ms/step - loss: 0.3943 -
acc: 0.8794 - val_loss: 0.4592 - val_acc: 0.8670
Epoch 9/20
7352/7352 [==============================] - 56s 8ms/step - loss: 0.2852 -
acc: 0.9138 - val_loss: 0.3710 - val_acc: 0.8721
Epoch 10/20
7352/7352 [==============================] - 56s 8ms/step - loss: 0.2336 -
acc: 0.9274 - val_loss: 0.8979 - val_acc: 0.7944
Epoch 11/20
7352/7352 [==============================] - 55s 8ms/step - loss: 0.2195 -
acc: 0.9286 - val_loss: 0.3862 - val_acc: 0.8870
Epoch 12/20
7352/7352 [==============================] - 54s 7ms/step - loss: 0.1841 -
acc: 0.9361 - val_loss: 0.3423 - val_acc: 0.8887
Epoch 13/20
7352/7352 [==============================] - 55s 7ms/step - loss: 0.1989 -
acc: 0.9314 - val_loss: 0.4002 - val_acc: 0.8958
Epoch 14/20
7352/7352 [==============================] - 55s 7ms/step - loss: 0.1692 -
acc: 0.9403 - val_loss: 0.2981 - val_acc: 0.9118
Epoch 15/20
7352/7352 [==============================] - 55s 8ms/step - loss: 0.1765 -
acc: 0.9402 - val_loss: 0.3853 - val_acc: 0.8846
Epoch 16/20
7352/7352 [==============================] - 55s 8ms/step - loss: 0.1503 -
acc: 0.9455 - val_loss: 0.3878 - val_acc: 0.9111
Epoch 17/20
7352/7352 [==============================] - 55s 7ms/step - loss: 0.1496 -
acc: 0.9452 - val_loss: 0.2945 - val_acc: 0.9063
Epoch 18/20
7352/7352 [==============================] - 55s 7ms/step - loss: 0.1426 -
acc: 0.9441 - val_loss: 0.4237 - val_acc: 0.8887
Epoch 19/20
7352/7352 [==============================] - 55s 7ms/step - loss: 0.1393 -
acc: 0.9478 - val_loss: 0.3919 - val_acc: 0.8965
Epoch 20/20
7352/7352 [==============================] - 55s 7ms/step - loss: 0.1491 -
acc: 0.9463 - val_loss: 0.5296 - val_acc: 0.8887
```

Out[37]:

<keras.callbacks.History at 0x1611cebd160>

In [38]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAI
RS  \
True
LAYING                 510        0        27        0
0
SITTING                  0      383       105        0
1
STANDING                 0      108       424        0
0
WALKING                  0        0         0      440
54
WALKING_DOWNSTAIRS       0        0         0        3                 4
17
WALKING_UPSTAIRS         0        0         0        4
22

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            2
STANDING                           0
WALKING                            2
WALKING_DOWNSTAIRS                 0
WALKING_UPSTAIRS                 445
```

In [39]:

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

```
2947/2947 [==============================] - 3s 903us/step
```

In [40]:

```
score
```

Out[40]:

```
[0.5296491873798939, 0.8887003732609433]
```

(3) Model having 2 LSTM layer with 32 LSTM Units and 'rmsprop' as an optimizer

In [59]:

```
# Initiliazing the sequential model
model4 = Sequential()
# Configuring the parameters
model4.add(LSTM(32,return_sequences=True, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model4.add(Dropout(0.5))

# Configuring the parameters
model4.add(LSTM(32))
# Adding a dropout layer
model4.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model4.add(Dense(n_classes, activation='sigmoid'))
print(model4.summary())

# Compiling the model
model4.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'
])

# Training the model
history4 = model4.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test, Y_
test),epochs=epochs)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_21 (LSTM)               (None, 128, 32)           5376      
_________________________________________________________________
dropout_15 (Dropout)         (None, 128, 32)           0         
_________________________________________________________________
lstm_22 (LSTM)               (None, 32)                8320      
_________________________________________________________________
dropout_16 (Dropout)         (None, 32)                0         
_________________________________________________________________
dense_5 (Dense)              (None, 6)                 198       
=================================================================
Total params: 13,894
Trainable params: 13,894
Non-trainable params: 0
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/20
7352/7352 [==============================] - 116s 16ms/step - loss: 1.2293
- acc: 0.4739 - val_loss: 1.0718 - val_acc: 0.5511
Epoch 2/20
7352/7352 [==============================] - 94s 13ms/step - loss: 0.8660
- acc: 0.6144 - val_loss: 0.9926 - val_acc: 0.5443
Epoch 3/20
7352/7352 [==============================] - 96s 13ms/step - loss: 0.7223
- acc: 0.6919 - val_loss: 0.9335 - val_acc: 0.5775
Epoch 4/20
7352/7352 [==============================] - 94s 13ms/step - loss: 0.6039
- acc: 0.7514 - val_loss: 0.6213 - val_acc: 0.7377
Epoch 5/20
7352/7352 [==============================] - 95s 13ms/step - loss: 0.5112
- acc: 0.7767 - val_loss: 0.5726 - val_acc: 0.7526
Epoch 6/20
7352/7352 [==============================] - 97s 13ms/step - loss: 0.4599
- acc: 0.8084 - val_loss: 0.5772 - val_acc: 0.8259
Epoch 7/20
7352/7352 [==============================] - 95s 13ms/step - loss: 0.3722
- acc: 0.8681 - val_loss: 0.5780 - val_acc: 0.8449
Epoch 8/20
7352/7352 [==============================] - 96s 13ms/step - loss: 0.3117
- acc: 0.9064 - val_loss: 0.4944 - val_acc: 0.8751
Epoch 9/20
7352/7352 [==============================] - 96s 13ms/step - loss: 0.2528
- acc: 0.9237 - val_loss: 0.5478 - val_acc: 0.8571
Epoch 10/20
7352/7352 [==============================] - 96s 13ms/step - loss: 0.2194
- acc: 0.9378 - val_loss: 0.5815 - val_acc: 0.8636
Epoch 11/20
7352/7352 [==============================] - 96s 13ms/step - loss: 0.1888
- acc: 0.9365 - val_loss: 0.5230 - val_acc: 0.8911
Epoch 12/20
7352/7352 [==============================] - 97s 13ms/step - loss: 0.2101
- acc: 0.9334 - val_loss: 0.5186 - val_acc: 0.8843
Epoch 13/20
7352/7352 [==============================] - 96s 13ms/step - loss: 0.1820
- acc: 0.9374 - val_loss: 0.5300 - val_acc: 0.8795
Epoch 14/20
7352/7352 [==============================] - 96s 13ms/step - loss: 0.1796
- acc: 0.9426 - val_loss: 0.4395 - val_acc: 0.8941
```

```
Epoch 15/20
7352/7352 [==============================] - 96s 13ms/step - loss: 0.1554
- acc: 0.9449 - val_loss: 0.6407 - val_acc: 0.8890
Epoch 16/20
7352/7352 [==============================] - 96s 13ms/step - loss: 0.1763
- acc: 0.9422 - val_loss: 0.6363 - val_acc: 0.8738
Epoch 17/20
7352/7352 [==============================] - 97s 13ms/step - loss: 0.1534
- acc: 0.9486 - val_loss: 0.9969 - val_acc: 0.8575
Epoch 18/20
7352/7352 [==============================] - 97s 13ms/step - loss: 0.1770
- acc: 0.9436 - val_loss: 0.4199 - val_acc: 0.9019
Epoch 19/20
7352/7352 [==============================] - 97s 13ms/step - loss: 0.1640
- acc: 0.9452 - val_loss: 0.4450 - val_acc: 0.9016
Epoch 20/20
7352/7352 [==============================] - 97s 13ms/step - loss: 0.1538
- acc: 0.9497 - val_loss: 0.4741 - val_acc: 0.9080
```

In [61]:

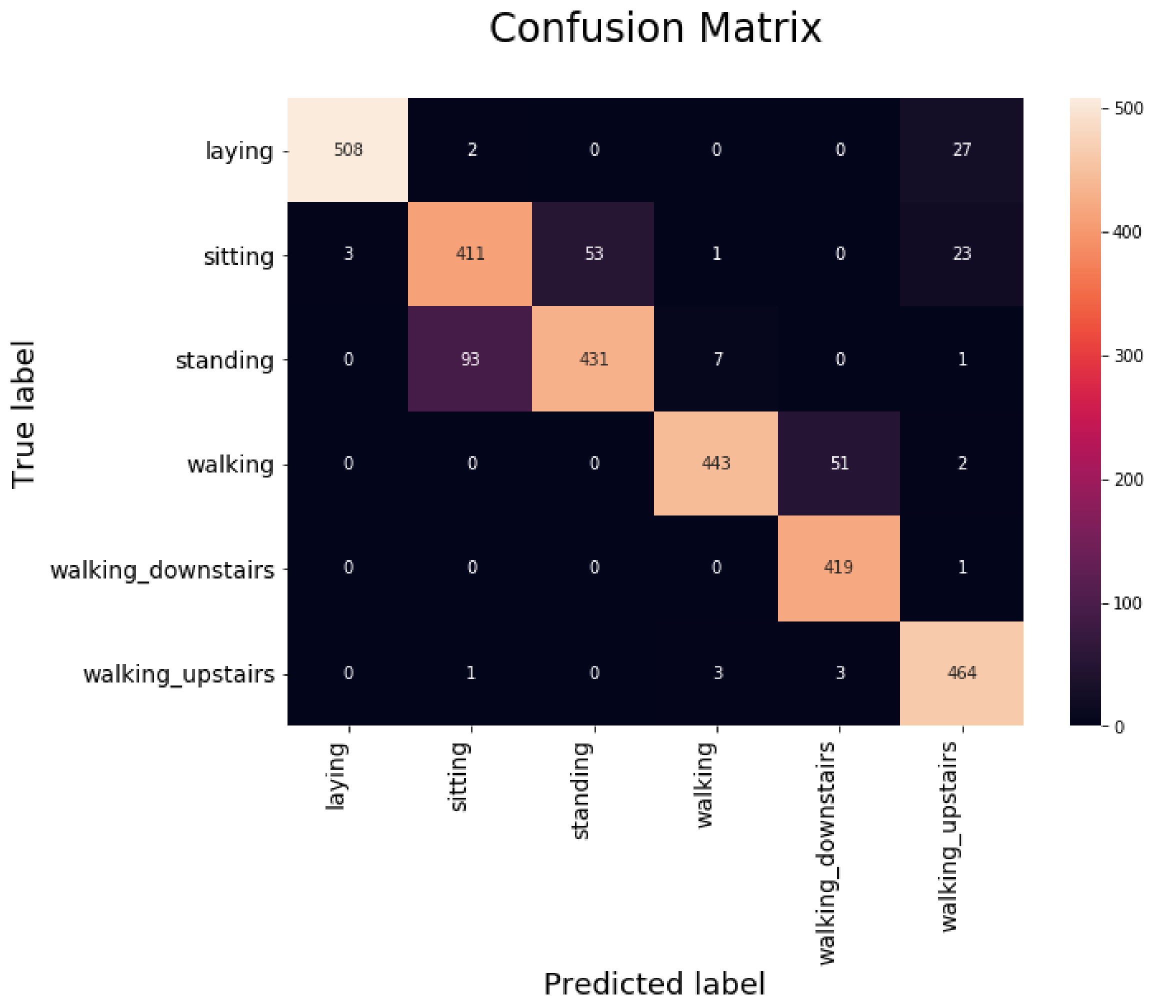
```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.metrics import confusion_matrix
scores4 = model4.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores4[0]))
print("Test Accuracy: %f%%" % (scores4[1]*100))

# Confusion Matrix
Y_true = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_test, axis=1)])
Y_predictions = pd.Series([ACTIVITIES[y] for y in np.argmax(model4.predict(X_test), axi
s=1)])

# Code for drawing seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downstairs','walking_up
stairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_names, c
olumns=class_names )
fig = plt.figure(figsize=(10,7))
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")

# Setting tick labels for heatmap
heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotation=0, ha='right', fo
ntsize=14)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=90, ha='right', f
ontsize=14)
plt.ylabel('True label',size=18)
plt.xlabel('Predicted label',size=18)
plt.title("Confusion Matrix\n",size=24)
plt.show()
```

```
Test Score: 0.474115
Test Accuracy: 90.804208%
```



(4) Model having 2 LSTM layer with 32 LSTM Units and 'rmsprop' as an optimizer

In [72]:

```
# Initiliazing the sequential model
model7 = Sequential()
# Configuring the parameters
model7.add(LSTM(32,return_sequences=True, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model7.add(Dropout(0.4))

# Configuring the parameters
model7.add(LSTM(32))
# Adding a dropout layer
model7.add(Dropout(0.4))
# Adding a dense output layer with sigmoid activation
model7.add(Dense(n_classes, activation='sigmoid'))
print(model7.summary())

# Compiling the model
model7.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'
])

# Training the model
history7 = model7.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test, Y_
test),epochs=30)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_26 (LSTM)               (None, 128, 32)           5376      
_________________________________________________________________
dropout_20 (Dropout)         (None, 128, 32)           0         
_________________________________________________________________
lstm_27 (LSTM)               (None, 32)                8320      
_________________________________________________________________
dropout_21 (Dropout)         (None, 32)                0         
_________________________________________________________________
dense_9 (Dense)              (None, 6)                 198       
=================================================================
Total params: 13,894
Trainable params: 13,894
Non-trainable params: 0
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 112s 15ms/step - loss: 1.1430
- acc: 0.5148 - val_loss: 0.9322 - val_acc: 0.5490
Epoch 2/30
7352/7352 [==============================] - 116s 16ms/step - loss: 0.7485
- acc: 0.6473 - val_loss: 0.7438 - val_acc: 0.6105
Epoch 3/30
7352/7352 [==============================] - 151s 21ms/step - loss: 0.6597
- acc: 0.7055 - val_loss: 0.7685 - val_acc: 0.6800
Epoch 4/30
7352/7352 [==============================] - 114s 15ms/step - loss: 0.5819
- acc: 0.7447 - val_loss: 0.5570 - val_acc: 0.7693
Epoch 5/30
7352/7352 [==============================] - 103s 14ms/step - loss: 0.5028
- acc: 0.7806 - val_loss: 0.6238 - val_acc: 0.7513
Epoch 6/30
7352/7352 [==============================] - 104s 14ms/step - loss: 0.4340
- acc: 0.8084 - val_loss: 0.5107 - val_acc: 0.8314
Epoch 7/30
7352/7352 [==============================] - 121s 16ms/step - loss: 0.3709
- acc: 0.8663 - val_loss: 0.5936 - val_acc: 0.8202
Epoch 8/30
7352/7352 [==============================] - 168s 23ms/step - loss: 0.2850
- acc: 0.9106 - val_loss: 0.4576 - val_acc: 0.8802
Epoch 9/30
7352/7352 [==============================] - 166s 23ms/step - loss: 0.2369
- acc: 0.9236 - val_loss: 0.4088 - val_acc: 0.8839
Epoch 10/30
7352/7352 [==============================] - 168s 23ms/step - loss: 0.1951
- acc: 0.9346 - val_loss: 0.3617 - val_acc: 0.8979
Epoch 11/30
7352/7352 [==============================] - 164s 22ms/step - loss: 0.1827
- acc: 0.9369 - val_loss: 0.6861 - val_acc: 0.8381
Epoch 12/30
7352/7352 [==============================] - 170s 23ms/step - loss: 0.1798
- acc: 0.9392 - val_loss: 0.3199 - val_acc: 0.9063
Epoch 13/30
7352/7352 [==============================] - 175s 24ms/step - loss: 0.1682
- acc: 0.9436 - val_loss: 0.3391 - val_acc: 0.9050
Epoch 14/30
7352/7352 [==============================] - 159s 22ms/step - loss: 0.1495
- acc: 0.9495 - val_loss: 0.3261 - val_acc: 0.9030
```

```
Epoch 15/30
7352/7352 [==============================] - 151s 20ms/step - loss: 0.1568
- acc: 0.9440 - val_loss: 0.2831 - val_acc: 0.9257
Epoch 16/30
7352/7352 [==============================] - 151s 21ms/step - loss: 0.1412
- acc: 0.9490 - val_loss: 0.4067 - val_acc: 0.9108
Epoch 17/30
7352/7352 [==============================] - 153s 21ms/step - loss: 0.1427
- acc: 0.9493 - val_loss: 0.3324 - val_acc: 0.9084
Epoch 18/30
7352/7352 [==============================] - 165s 22ms/step - loss: 0.1548
- acc: 0.9468 - val_loss: 0.3240 - val_acc: 0.9165
Epoch 19/30
7352/7352 [==============================] - 160s 22ms/step - loss: 0.1355
- acc: 0.9484 - val_loss: 0.3541 - val_acc: 0.9131
Epoch 20/30
7352/7352 [==============================] - 149s 20ms/step - loss: 0.1464
- acc: 0.9489 - val_loss: 0.4399 - val_acc: 0.8975
Epoch 21/30
7352/7352 [==============================] - 94s 13ms/step - loss: 0.1597
- acc: 0.9448 - val_loss: 0.3024 - val_acc: 0.9192
Epoch 22/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.1336
- acc: 0.9508 - val_loss: 0.3142 - val_acc: 0.9226
Epoch 23/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.1433
- acc: 0.9484 - val_loss: 0.3331 - val_acc: 0.9257
Epoch 24/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.1431
- acc: 0.9470 - val_loss: 0.2522 - val_acc: 0.9277
Epoch 25/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.1340
- acc: 0.9498 - val_loss: 0.3704 - val_acc: 0.9162
Epoch 26/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.1334
- acc: 0.9524 - val_loss: 0.2666 - val_acc: 0.9192
Epoch 27/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.1325
- acc: 0.9486 - val_loss: 0.2636 - val_acc: 0.9125
Epoch 28/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.1261
- acc: 0.9518 - val_loss: 0.3061 - val_acc: 0.9091
Epoch 29/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.1382
- acc: 0.9501 - val_loss: 0.3471 - val_acc: 0.9175
Epoch 30/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.1260
- acc: 0.9543 - val_loss: 0.3902 - val_acc: 0.9053
```

In [74]:

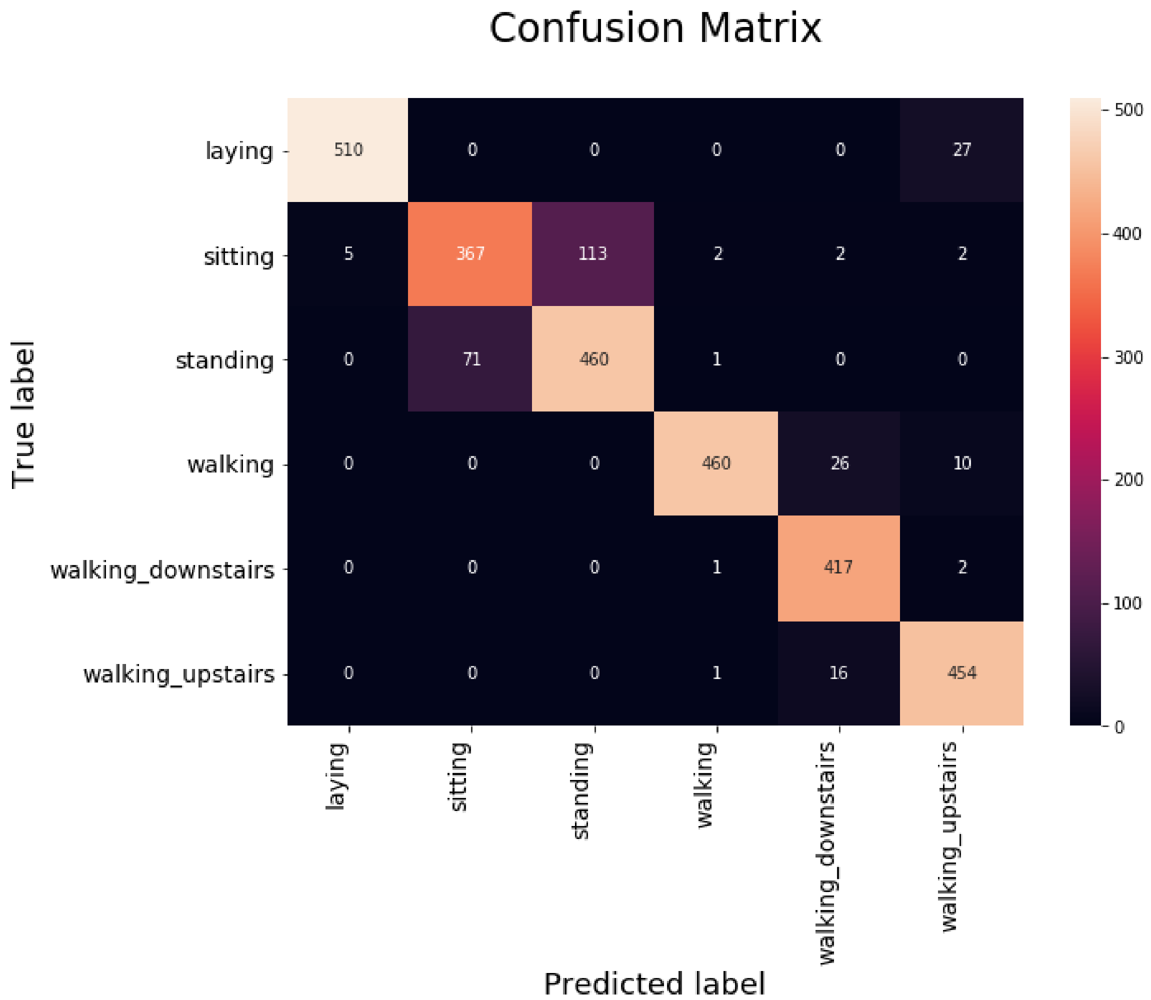
```
scores7 = model7.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores7[0]))
print("Test Accuracy: %f%%" % (scores7[1]*100))

# Confusion Matrix
Y_true = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_test, axis=1)])
Y_predictions = pd.Series([ACTIVITIES[y] for y in np.argmax(model7.predict(X_test), axi
s=1)])

# Code for drawing seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downstairs','walking_up
stairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_names, c
olumns=class_names )
fig = plt.figure(figsize=(10,7))
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")

# Setting tick labels for heatmap
heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotation=0, ha='right', fo
ntsize=14)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=90, ha='right', f
ontsize=14)
plt.ylabel('True label',size=18)
plt.xlabel('Predicted label',size=18)
plt.title("Confusion Matrix\n",size=24)
plt.show()
```

```
Test Score: 0.390201
Test Accuracy: 90.532745%
```



(5) Model having 2 LSTM layer with 64 LSTM Units

In [75]:

```
# Initiliazing the sequential model
model8 = Sequential()
# Configuring the parameters
model8.add(LSTM(64,return_sequences=True, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model8.add(Dropout(0.3))

# Configuring the parameters
model8.add(LSTM(64))
# Adding a dropout layer
model8.add(Dropout(0.3))
# Adding a dense output layer with sigmoid activation
model8.add(Dense(n_classes, activation='sigmoid'))
print(model8.summary())

# Compiling the model
model8.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'
])

# Training the model
history8 = model8.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test, Y_
test),epochs=30)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_30 (LSTM)               (None, 128, 64)           18944     
_________________________________________________________________
dropout_24 (Dropout)         (None, 128, 64)           0         
_________________________________________________________________
lstm_31 (LSTM)               (None, 64)                33024     
_________________________________________________________________
dropout_25 (Dropout)         (None, 64)                0         
_________________________________________________________________
dense_11 (Dense)             (None, 6)                 390       
=================================================================
Total params: 52,358
Trainable params: 52,358
Non-trainable params: 0
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 126s 17ms/step - loss: 1.0656
- acc: 0.5140 - val_loss: 0.8124 - val_acc: 0.5928
Epoch 2/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.7328
- acc: 0.6503 - val_loss: 0.7082 - val_acc: 0.7292
Epoch 3/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.6599
- acc: 0.7157 - val_loss: 0.6380 - val_acc: 0.7126
Epoch 4/30
7352/7352 [==============================] - 123s 17ms/step - loss: 0.5255
- acc: 0.7816 - val_loss: 0.5473 - val_acc: 0.7947
Epoch 5/30
7352/7352 [==============================] - 123s 17ms/step - loss: 0.3961
- acc: 0.8505 - val_loss: 0.4307 - val_acc: 0.8351
Epoch 6/30
7352/7352 [==============================] - 123s 17ms/step - loss: 0.2608
- acc: 0.9104 - val_loss: 0.3643 - val_acc: 0.8785
Epoch 7/30
7352/7352 [==============================] - 124s 17ms/step - loss: 0.2043
- acc: 0.9276 - val_loss: 0.3512 - val_acc: 0.8890
Epoch 8/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.1735
- acc: 0.9374 - val_loss: 0.3139 - val_acc: 0.9040
Epoch 9/30
7352/7352 [==============================] - 135s 18ms/step - loss: 0.1540
- acc: 0.9406 - val_loss: 0.3518 - val_acc: 0.9080
Epoch 10/30
7352/7352 [==============================] - 120s 16ms/step - loss: 0.1568
- acc: 0.9392 - val_loss: 0.3272 - val_acc: 0.9091
Epoch 11/30
7352/7352 [==============================] - 120s 16ms/step - loss: 0.1413
- acc: 0.9457 - val_loss: 0.2985 - val_acc: 0.9091
Epoch 12/30
7352/7352 [==============================] - 120s 16ms/step - loss: 0.1362
- acc: 0.9465 - val_loss: 0.3764 - val_acc: 0.8921
Epoch 13/30
7352/7352 [==============================] - 120s 16ms/step - loss: 0.1336
- acc: 0.9497 - val_loss: 0.4482 - val_acc: 0.8843
Epoch 14/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.1318
- acc: 0.9527 - val_loss: 0.3553 - val_acc: 0.8921
```

```
Epoch 15/30
7352/7352 [==============================] - 124s 17ms/step - loss: 0.1350
- acc: 0.9518 - val_loss: 0.4011 - val_acc: 0.9002
Epoch 16/30
7352/7352 [==============================] - 124s 17ms/step - loss: 0.1283
- acc: 0.9483 - val_loss: 0.4361 - val_acc: 0.9036
Epoch 17/30
7352/7352 [==============================] - 123s 17ms/step - loss: 0.1323
- acc: 0.9497 - val_loss: 0.3424 - val_acc: 0.9101
Epoch 18/30
7352/7352 [==============================] - 124s 17ms/step - loss: 0.1254
- acc: 0.9527 - val_loss: 0.4371 - val_acc: 0.9040
Epoch 19/30
7352/7352 [==============================] - 124s 17ms/step - loss: 0.1225
- acc: 0.9498 - val_loss: 0.3420 - val_acc: 0.9063
Epoch 20/30
7352/7352 [==============================] - 124s 17ms/step - loss: 0.1308
- acc: 0.9493 - val_loss: 0.3504 - val_acc: 0.9040
Epoch 21/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.1227
- acc: 0.9508 - val_loss: 0.3189 - val_acc: 0.9148
Epoch 22/30
7352/7352 [==============================] - 124s 17ms/step - loss: 0.1258
- acc: 0.9518 - val_loss: 0.3695 - val_acc: 0.9125
Epoch 23/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.1190
- acc: 0.9548 - val_loss: 0.4747 - val_acc: 0.8962
Epoch 24/30
7352/7352 [==============================] - 124s 17ms/step - loss: 0.1221
- acc: 0.9533 - val_loss: 0.3035 - val_acc: 0.9186
Epoch 25/30
7352/7352 [==============================] - 143s 20ms/step - loss: 0.1213
- acc: 0.9538 - val_loss: 0.3753 - val_acc: 0.9223
Epoch 26/30
7352/7352 [==============================] - 139s 19ms/step - loss: 0.1304
- acc: 0.9518 - val_loss: 0.3458 - val_acc: 0.8999
Epoch 27/30
7352/7352 [==============================] - 198s 27ms/step - loss: 0.1144
- acc: 0.9543 - val_loss: 0.3485 - val_acc: 0.9155
Epoch 28/30
7352/7352 [==============================] - 175s 24ms/step - loss: 0.1286
- acc: 0.9489 - val_loss: 0.4093 - val_acc: 0.9006
Epoch 29/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.1334
- acc: 0.9495 - val_loss: 0.3099 - val_acc: 0.9036
Epoch 30/30
7352/7352 [==============================] - 134s 18ms/step - loss: 0.1364
- acc: 0.9523 - val_loss: 0.4480 - val_acc: 0.9094
```

In [76]:

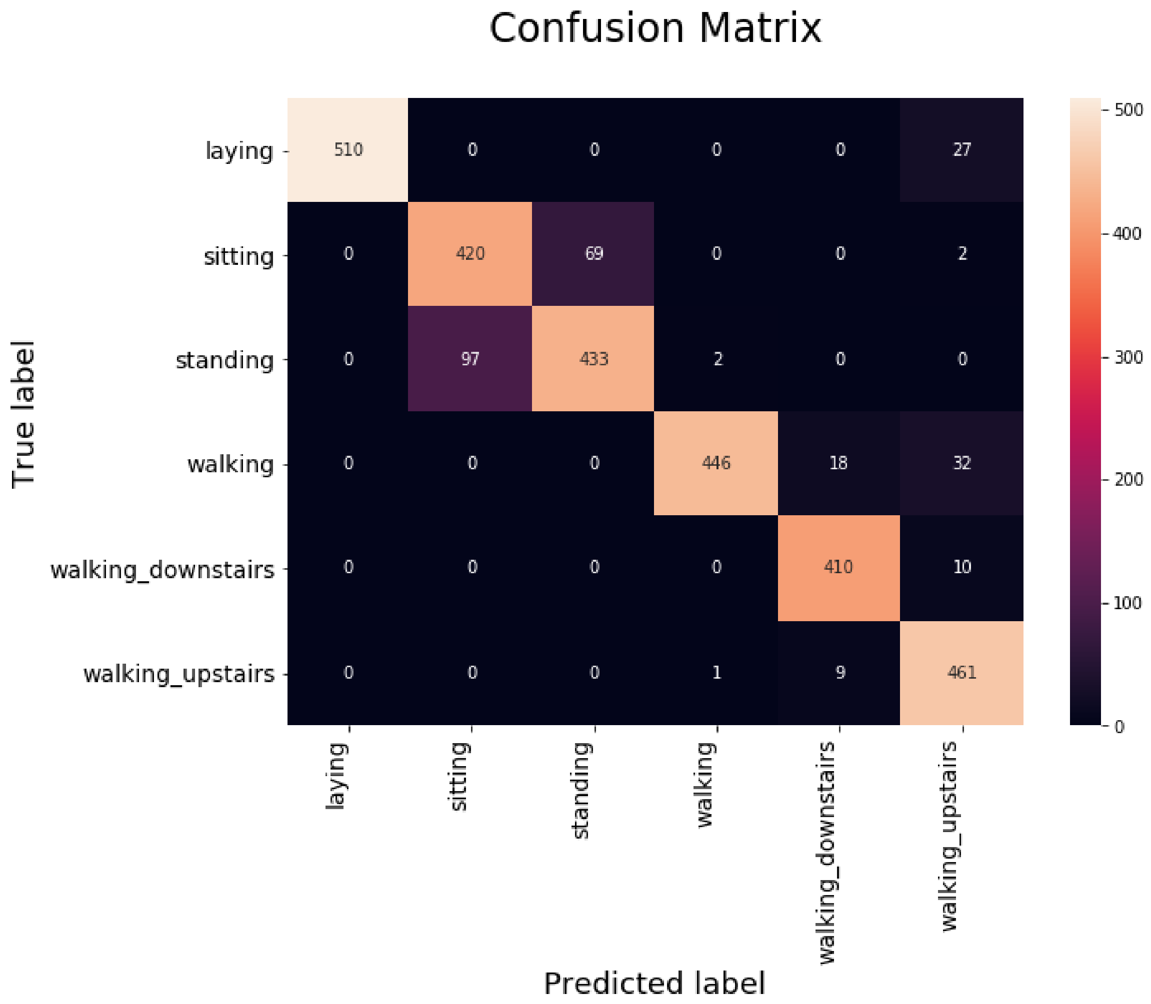
```
scores8 = model8.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores8[0]))
print("Test Accuracy: %f%%" % (scores8[1]*100))

# Confusion Matrix
Y_true = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_test, axis=1)])
Y_predictions = pd.Series([ACTIVITIES[y] for y in np.argmax(model8.predict(X_test), axi
s=1)])

# Code for drawing seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downstairs','walking_up
stairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_names, c
olumns=class_names )
fig = plt.figure(figsize=(10,7))
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")

# Setting tick labels for heatmap
heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotation=0, ha='right', fo
ntsize=14)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=90, ha='right', f
ontsize=14)
plt.ylabel('True label',size=18)
plt.xlabel('Predicted label',size=18)
plt.title("Confusion Matrix\n",size=24)
plt.show()
```

```
Test Score: 0.448009
Test Accuracy: 90.939939%
```



(6) Model having 2 LSTM layer with 64 LSTM Units

In [77]:

```
# Initiliazing the sequential model
model9 = Sequential()
# Configuring the parameters
model9.add(LSTM(64,return_sequences=True, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model9.add(Dropout(0.5))

# Configuring the parameters
model9.add(LSTM(64))
# Adding a dropout layer
model9.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model9.add(Dense(n_classes, activation='sigmoid'))
print(model9.summary())

# Compiling the model
model9.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'
])

# Training the model
history9 = model9.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test, Y_
test),epochs=30)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_32 (LSTM)               (None, 128, 64)           18944     
_________________________________________________________________
dropout_26 (Dropout)         (None, 128, 64)           0         
_________________________________________________________________
lstm_33 (LSTM)               (None, 64)                33024     
_________________________________________________________________
dropout_27 (Dropout)         (None, 64)                0         
_________________________________________________________________
dense_12 (Dense)             (None, 6)                 390       
=================================================================
Total params: 52,358
Trainable params: 52,358
Non-trainable params: 0
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 142s 19ms/step - loss: 1.0596
- acc: 0.5354 - val_loss: 0.8040 - val_acc: 0.6057
Epoch 2/30
7352/7352 [==============================] - 124s 17ms/step - loss: 0.7153
- acc: 0.6545 - val_loss: 0.7176 - val_acc: 0.6518
Epoch 3/30
7352/7352 [==============================] - 144s 20ms/step - loss: 0.7113
- acc: 0.6729 - val_loss: 0.6954 - val_acc: 0.7061
Epoch 4/30
7352/7352 [==============================] - 182s 25ms/step - loss: 0.5630
- acc: 0.7380 - val_loss: 0.6496 - val_acc: 0.7374
Epoch 5/30
7352/7352 [==============================] - 179s 24ms/step - loss: 0.4935
- acc: 0.7738 - val_loss: 0.7803 - val_acc: 0.7302
Epoch 6/30
7352/7352 [==============================] - 131s 18ms/step - loss: 0.4744
- acc: 0.7822 - val_loss: 0.6037 - val_acc: 0.7438
Epoch 7/30
7352/7352 [==============================] - 164s 22ms/step - loss: 0.4461
- acc: 0.7829 - val_loss: 0.7946 - val_acc: 0.7384
Epoch 8/30
7352/7352 [==============================] - 128s 17ms/step - loss: 0.4657
- acc: 0.7769 - val_loss: 0.5941 - val_acc: 0.7557
Epoch 9/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.4341
- acc: 0.7950 - val_loss: 0.6260 - val_acc: 0.7465
Epoch 10/30
7352/7352 [==============================] - 128s 17ms/step - loss: 0.3719
- acc: 0.8362 - val_loss: 0.5230 - val_acc: 0.8429
Epoch 11/30
7352/7352 [==============================] - 153s 21ms/step - loss: 0.3067
- acc: 0.8979 - val_loss: 0.4232 - val_acc: 0.8687
Epoch 12/30
7352/7352 [==============================] - 226s 31ms/step - loss: 0.2034
- acc: 0.9343 - val_loss: 0.3980 - val_acc: 0.8728
Epoch 13/30
7352/7352 [==============================] - 137s 19ms/step - loss: 0.1850
- acc: 0.9414 - val_loss: 0.4111 - val_acc: 0.8887
Epoch 14/30
7352/7352 [==============================] - 124s 17ms/step - loss: 0.1811
- acc: 0.9370 - val_loss: 0.3506 - val_acc: 0.8941
```

```
Epoch 15/30
7352/7352 [==============================] - 124s 17ms/step - loss: 0.1631
- acc: 0.9418 - val_loss: 0.3627 - val_acc: 0.9074
Epoch 16/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.1604
- acc: 0.9438 - val_loss: 0.6482 - val_acc: 0.8588
Epoch 17/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.1462
- acc: 0.9404 - val_loss: 0.3852 - val_acc: 0.9013
Epoch 18/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.1399
- acc: 0.9461 - val_loss: 0.3426 - val_acc: 0.9087
Epoch 19/30
7352/7352 [==============================] - 123s 17ms/step - loss: 0.1365
- acc: 0.9461 - val_loss: 0.3597 - val_acc: 0.9141
Epoch 20/30
7352/7352 [==============================] - 137s 19ms/step - loss: 0.1318
- acc: 0.9498 - val_loss: 0.4201 - val_acc: 0.9097
Epoch 21/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.1375
- acc: 0.9487 - val_loss: 0.4146 - val_acc: 0.9070
Epoch 22/30
7352/7352 [==============================] - 130s 18ms/step - loss: 0.1296
- acc: 0.9493 - val_loss: 0.3749 - val_acc: 0.9203
Epoch 23/30
7352/7352 [==============================] - 123s 17ms/step - loss: 0.1329
- acc: 0.9525 - val_loss: 0.4103 - val_acc: 0.8941
Epoch 24/30
7352/7352 [==============================] - 123s 17ms/step - loss: 0.1401
- acc: 0.9487 - val_loss: 0.4844 - val_acc: 0.8860
Epoch 25/30
7352/7352 [==============================] - 123s 17ms/step - loss: 0.1471
- acc: 0.9482 - val_loss: 0.4122 - val_acc: 0.9053
Epoch 26/30
7352/7352 [==============================] - 124s 17ms/step - loss: 0.1242
- acc: 0.9505 - val_loss: 0.4268 - val_acc: 0.9057
Epoch 27/30
7352/7352 [==============================] - 123s 17ms/step - loss: 0.1408
- acc: 0.9479 - val_loss: 0.5572 - val_acc: 0.8945
Epoch 28/30
7352/7352 [==============================] - 123s 17ms/step - loss: 0.1318
- acc: 0.9523 - val_loss: 0.5135 - val_acc: 0.9077
Epoch 29/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.1437
- acc: 0.9474 - val_loss: 0.6846 - val_acc: 0.8907
Epoch 30/30
7352/7352 [==============================] - 130s 18ms/step - loss: 0.1325
- acc: 0.9516 - val_loss: 0.5240 - val_acc: 0.9050
```

In [78]:

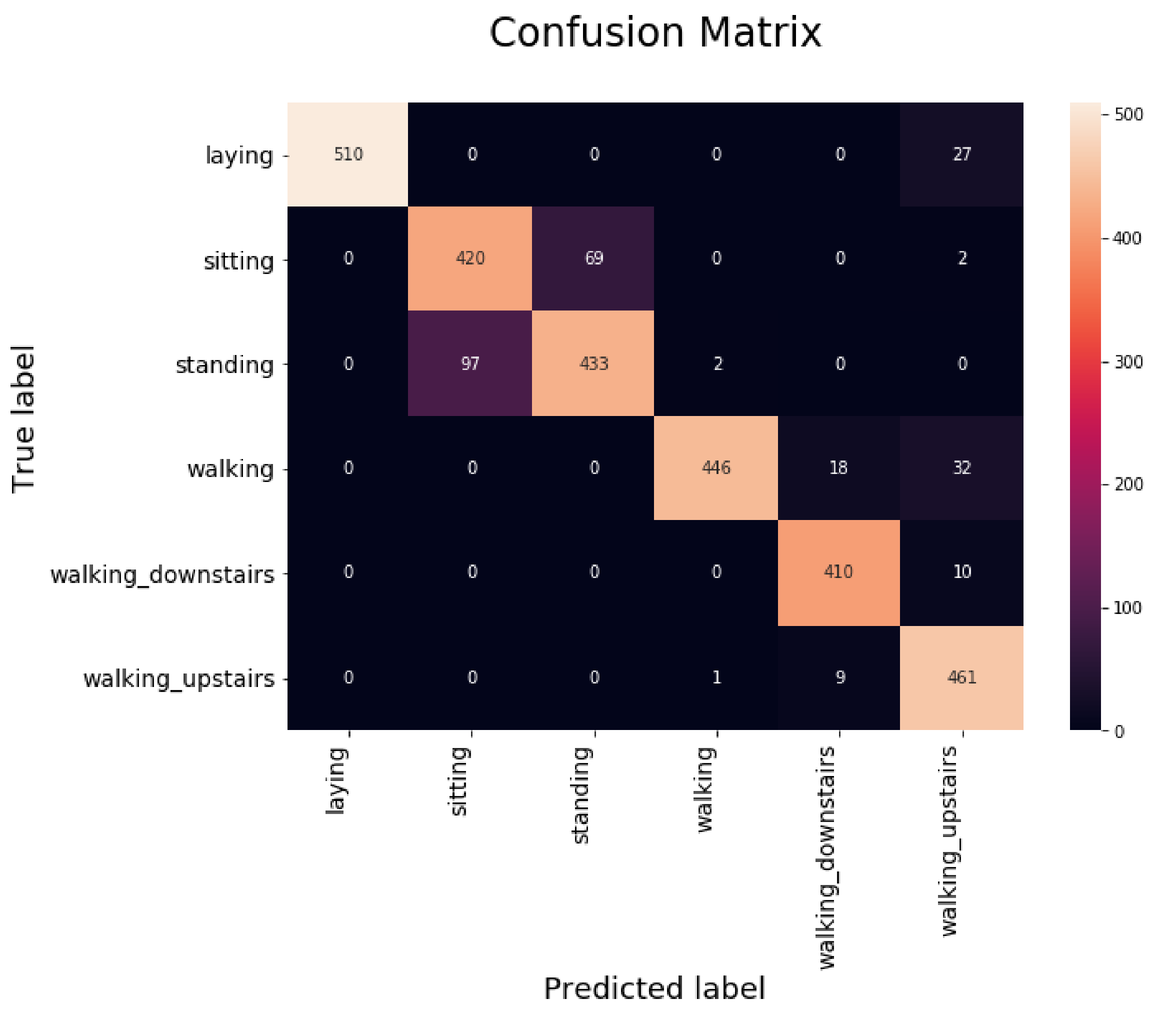
```
scores9 = model9.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores9[0]))
print("Test Accuracy: %f%%" % (scores9[1]*100))

# Confusion Matrix
Y_true = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_test, axis=1)])
Y_predictions = pd.Series([ACTIVITIES[y] for y in np.argmax(model8.predict(X_test), axi
s=1)])

# Code for drawing seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downstairs','walking_up
stairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_names, c
olumns=class_names )
fig = plt.figure(figsize=(10,7))
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")

# Setting tick labels for heatmap
heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotation=0, ha='right', fo
ntsize=14)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=90, ha='right', f
ontsize=14)
plt.ylabel('True label',size=18)
plt.xlabel('Predicted label',size=18)
plt.title("Confusion Matrix\n",size=24)
plt.show()
```

```
Test Score: 0.523963
Test Accuracy: 90.498812%
```



CONCLUSION

(a). Procedure Followed :

STEP 1 :- Load the data and split into training_data and test_data

STEP 2:-Try out different LSTM architectures

STEP 3:- Find test score and accuracy for each model

STEP 4:- Draw confusion matrix using seaborn heatmap for each model

(b). Table (Model performances) :

In [83]:

```
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Model","Epocs","DROPOUT","Training_Accuracy% ", "Test_Accuracy% "]

x.add_row(["1 LSTM layer with 32 LSTM Units(Optimizer-->rmsprop)",20,.5,93.27,87.91])

x.add_row(["1 LSTM layer with 64 LSTM Units(Optimizer-->rmsprop)",20,.5,94.63,88.87])

x.add_row(["2 LSTM layer with 32 LSTM Units(Optimizer-->rmsprop)",20,.5,94.97,90.80])

x.add_row(["2 LSTM layer with 32 LSTM Units(Optimizer-->rmsprop)",30,.4,95.43,90.53])

x.add_row(["2 LSTM layer with 64 LSTM Units(Optimizer-->rmsprop)",30,.3,95.23,90.93])

x.add_row(["2 LSTM layer with 64 LSTM Units(Optimizer-->rmsprop)",30,.5,95.16,90.49])

print(x)
```

```
+-------------------------------------------------------+-------+---------+
---------------------+-----------------+
|                         Model                         | Epocs | DROPOUT |
Training_Accuracy%  | Test_Accuracy%  |
+-------------------------------------------------------+-------+---------+
---------------------+-----------------+
| 1 LSTM layer with 32 LSTM Units(Optimizer-->rmsprop) |   20  |   0.5   |
93.27        |      87.91      |
| 1 LSTM layer with 64 LSTM Units(Optimizer-->rmsprop) |   20  |   0.5   |
94.63        |      88.87      |
| 2 LSTM layer with 32 LSTM Units(Optimizer-->rmsprop) |   20  |   0.5   |
94.97        |       90.8      |
| 2 LSTM layer with 32 LSTM Units(Optimizer-->rmsprop) |   30  |   0.4   |
95.43        |      90.53      |
| 2 LSTM layer with 64 LSTM Units(Optimizer-->rmsprop) |   30  |   0.3   |
95.23        |      90.93      |
| 2 LSTM layer with 64 LSTM Units(Optimizer-->rmsprop) |   30  |   0.5   |
95.16        |      90.49      |
+-------------------------------------------------------+-------+---------+
---------------------+-----------------+
```