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

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

In [10]:
```
X_test.shape
```

Out[10]: (2947, 128, 9)

In [11]:
```
def data():
    '''
    Data providing function:
    This function is separated from model() so that hyperopt
    won't reload data for each evaluation run.
    '''
    (X_train, y_train), (X_test, y_test) = load_data()
    X_train = X_train.reshape(7352, 128, 9)
    X_test = X_test.reshape(2947, 128, 9)
    X_train = X_train.astype('float32')
    X_test = X_test.astype('float32')
    X_train /= 255
    X_test /= 255
    nb_classes = 6
    Y_train = np_utils.to_categorical(y_train, nb_classes)
    Y_test = np_utils.to_categorical(y_test, nb_classes)
    return X_train, Y_train, X_test, Y_test
```

In [12]:
```
# Importing tensorflow
np.random.seed(42)
import tensorflow as tf
tf.set_random_seed(42)
```

In [13]:
```
# Configuring a session
session_conf = tf.ConfigProto(
    intra_op_parallelism_threads=1,
    inter_op_parallelism_threads=1
)
```

In [14]:
```
# Import Keras
from keras import backend as K
sess = tf.Session(graph=tf.get_default_graph(), config=session_conf)
K.set_session(sess)
```

```
Using TensorFlow backend.
```

In [34]:
```
# Importing libraries
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers.core import Dense, Dropout
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
<ipython-input-34-6e3432c44f90> in <module>
      1 # Importing libraries
----> 2 from keras.models import Sequential,BatchNormalization
      3 from keras.layers import LSTM
      4 from keras.layers.core import Dense, Dropout

ImportError: cannot import name 'BatchNormalization'
```

In [16]:
```
# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 32
```

In [17]:
```
# Utility function to count the number of classes
def _count_classes(y):
    return len(set([tuple(category) for category in y]))
```

In [18]:
```
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
```

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

In [62]:
```
# 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()

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

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

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_16 (LSTM)               (None, 32)                5376
_________________________________________________________________
dropout_15 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_12 (Dense)             (None, 6)                 198
=================================================================
Total params: 5,574
Trainable params: 5,574
Non-trainable params: 0
_________________________________________________________________
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 51s 7ms/step - loss: 1.3230 - acc: 0.4313 - v
al_loss: 1.1829 - val_acc: 0.4581
Epoch 2/30
7352/7352 [==============================] - 47s 6ms/step - loss: 1.1046 - acc: 0.5026 - v
al_loss: 1.1248 - val_acc: 0.4971
Epoch 3/30
7352/7352 [==============================] - 48s 6ms/step - loss: 0.9713 - acc: 0.5584 - v
al_loss: 0.9203 - val_acc: 0.5619
Epoch 4/30
7352/7352 [==============================] - 48s 7ms/step - loss: 0.8708 - acc: 0.6017 - v
al_loss: 0.8737 - val_acc: 0.5596
Epoch 5/30
7352/7352 [==============================] - 49s 7ms/step - loss: 0.7975 - acc: 0.6283 - v
al_loss: 0.7808 - val_acc: 0.6142
Epoch 6/30
7352/7352 [==============================] - 50s 7ms/step - loss: 0.7230 - acc: 0.6487 - v
al_loss: 0.7735 - val_acc: 0.6244
Epoch 7/30
7352/7352 [==============================] - 49s 7ms/step - loss: 0.6699 - acc: 0.6817 - v
al_loss: 0.7623 - val_acc: 0.6366
Epoch 8/30
7352/7352 [==============================] - 50s 7ms/step - loss: 0.6912 - acc: 0.6674 - v
al_loss: 0.7512 - val_acc: 0.6498
Epoch 9/30
7352/7352 [==============================] - 50s 7ms/step - loss: 0.6376 - acc: 0.6914 - v
al_loss: 0.7344 - val_acc: 0.6902
```

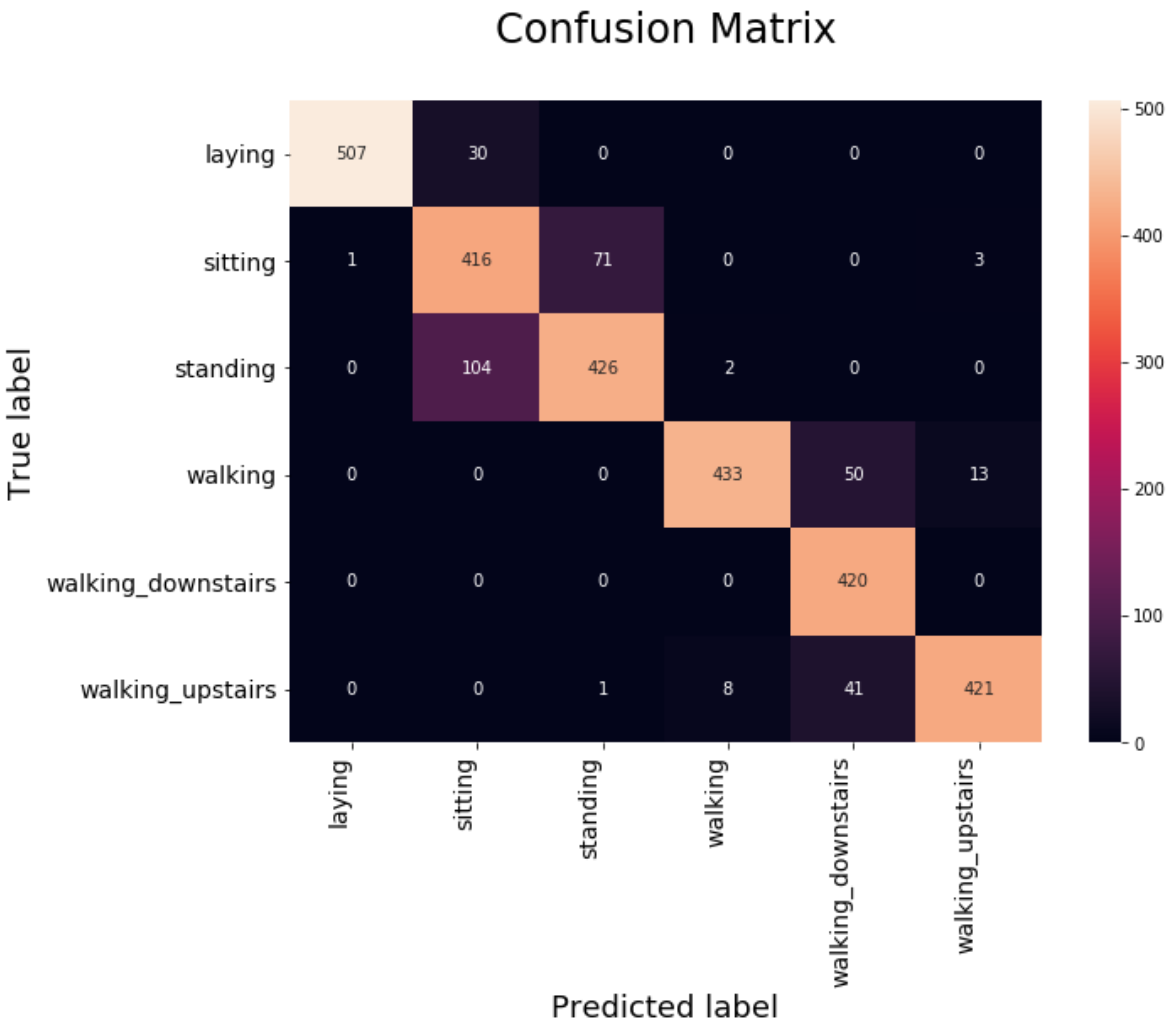
```
Epoch 10/30
7352/7352 [==============================] - 50s 7ms/step - loss: 0.6139 - acc: 0.7163 - v
al_loss: 0.6932 - val_acc: 0.6837
Epoch 11/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.5442 - acc: 0.7730 - v
al_loss: 0.6092 - val_acc: 0.7570
Epoch 12/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.4676 - acc: 0.8200 - v
al_loss: 0.5451 - val_acc: 0.7852
Epoch 13/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.5096 - acc: 0.8410 - v
al_loss: 0.3855 - val_acc: 0.8629
Epoch 14/30
7352/7352 [==============================] - 92s 13ms/step - loss: 0.3351 - acc: 0.9003 -
val_loss: 0.5337 - val_acc: 0.8337
Epoch 15/30
7352/7352 [==============================] - 79s 11ms/step - loss: 0.2603 - acc: 0.9192 -
val_loss: 0.4121 - val_acc: 0.8907
Epoch 16/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.2444 - acc: 0.9286 - v
al_loss: 0.5127 - val_acc: 0.8558
Epoch 17/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.2457 - acc: 0.9242 -
val_loss: 0.3653 - val_acc: 0.9036
Epoch 18/30
7352/7352 [==============================] - 97s 13ms/step - loss: 0.1964 - acc: 0.9329 -
val_loss: 0.3084 - val_acc: 0.8931
Epoch 19/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.1899 - acc: 0.9389 -
val_loss: 0.3581 - val_acc: 0.9019s - l
Epoch 20/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.2160 - acc: 0.9350 -
val_loss: 0.2692 - val_acc: 0.9063
Epoch 21/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1694 - acc: 0.9408 - v
al_loss: 0.3861 - val_acc: 0.9084
Epoch 22/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1776 - acc: 0.9416 - v
al_loss: 0.4859 - val_acc: 0.8863
Epoch 23/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1789 - acc: 0.9414 - v
al_loss: 0.3394 - val_acc: 0.9087
Epoch 24/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1806 - acc: 0.9404 - v
al_loss: 0.3293 - val_acc: 0.9053
Epoch 25/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.1764 - acc: 0.9429 - v
al_loss: 0.3081 - val_acc: 0.9046
Epoch 26/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1773 - acc: 0.9410 - v
al_loss: 0.4084 - val_acc: 0.9013
Epoch 27/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.1683 - acc: 0.9440 - v
al_loss: 0.3903 - val_acc: 0.9053
Epoch 28/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.1742 - acc: 0.9457 - v
al_loss: 0.4623 - val_acc: 0.9006
```

```
Epoch 29/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.1798 - acc: 0.9440 - v
al_loss: 0.4666 - val_acc: 0.8928
Epoch 30/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.1695 - acc: 0.9460 - v
al_loss: 0.5087 - val_acc: 0.8901
```

In [63]:

```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Final evaluation of the model
scores = model.evaluate(X_test, Y_test, verbose=1)
print("Test Score: %f" % (scores[0]))
print("Test Accuracy: %f%%" % (scores[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(model.predict(X_test), axis=1)])

# Code for drawing seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downstairs','walking_upstairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_names, columns=
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', fontsize=14)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=90, ha='right', fontsize=14)
plt.ylabel('True label',size=18)
plt.xlabel('Predicted label',size=18)
plt.title("Confusion Matrix\n",size=24)
plt.show()
```

```
2947/2947 [==============================] - 2s 671us/step
Test Score: 0.508714
Test Accuracy: 89.005769%
```



(2) Model having 1 LSTM layer with 64 LSTM Units¶

In [64]:
```
# Initiliazing the sequential model
model1 = Sequential()
# Configuring the parameters
model1.add(LSTM(64, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model1.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model1.add(Dense(n_classes, activation='sigmoid'))
model1.summary()

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

# Training the model
history1 = model1.fit(X_train, Y_train, batch_size=batch_size,validation_data=(X_test, Y_test),epochs
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_17 (LSTM)               (None, 64)                18944
_________________________________________________________________
dropout_16 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_13 (Dense)             (None, 6)                 390
=================================================================
Total params: 19,334
Trainable params: 19,334
Non-trainable params: 0
_________________________________________________________________
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 64s 9ms/step - loss: 1.2460 - acc: 0.4499 - v
al_loss: 1.1414 - val_acc: 0.5097
Epoch 2/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.9701 - acc: 0.5705 - v
al_loss: 0.9003 - val_acc: 0.5908
Epoch 3/30
7352/7352 [==============================] - 59s 8ms/step - loss: 0.8082 - acc: 0.6376 - v
al_loss: 0.8418 - val_acc: 0.6257
Epoch 4/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.6972 - acc: 0.6884 - v
al_loss: 0.9682 - val_acc: 0.6892
Epoch 5/30
7352/7352 [==============================] - 60s 8ms/step - loss: 0.7036 - acc: 0.7341 - v
al_loss: 0.7319 - val_acc: 0.7577
Epoch 6/30
7352/7352 [==============================] - 60s 8ms/step - loss: 0.5129 - acc: 0.8259 - v
al_loss: 0.7188 - val_acc: 0.7431
Epoch 7/30
7352/7352 [==============================] - 60s 8ms/step - loss: 0.3887 - acc: 0.8723 - v
al_loss: 0.6141 - val_acc: 0.8178
Epoch 8/30
7352/7352 [==============================] - 60s 8ms/step - loss: 0.3144 - acc: 0.8988 - v
al_loss: 0.5157 - val_acc: 0.8592
Epoch 9/30
7352/7352 [==============================] - 60s 8ms/step - loss: 0.2587 - acc: 0.9150 - v
al_loss: 0.4853 - val_acc: 0.8351
```

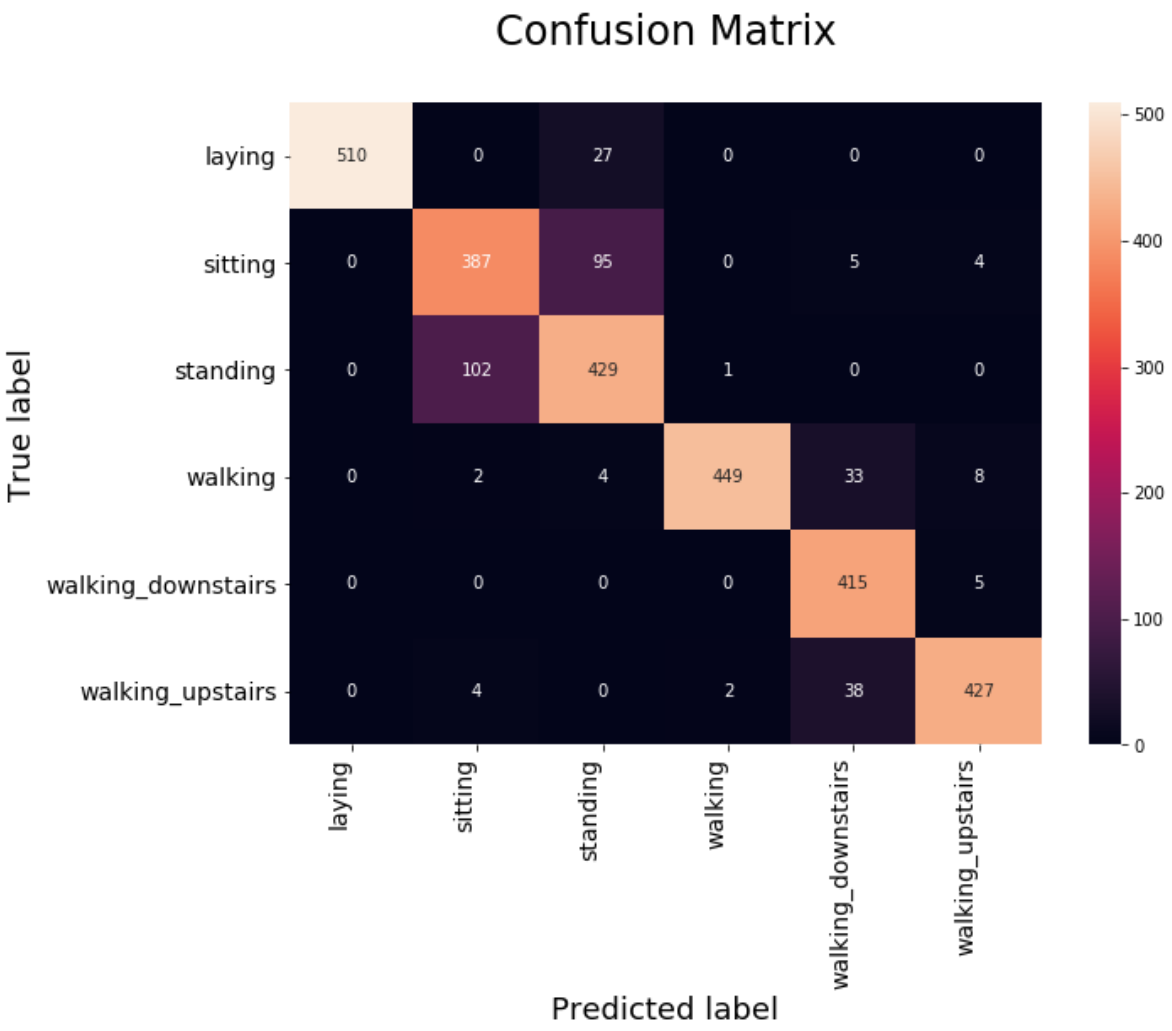
```
Epoch 10/30
7352/7352 [==============================] - 60s 8ms/step - loss: 0.2273 - acc: 0.9249 - v
al_loss: 0.4464 - val_acc: 0.8694
Epoch 11/30
7352/7352 [==============================] - 75s 10ms/step - loss: 0.2335 - acc: 0.9245 -
val_loss: 0.5008 - val_acc: 0.8510
Epoch 12/30
7352/7352 [==============================] - 116s 16ms/step - loss: 0.1920 - acc: 0.9344
- val_loss: 0.4005 - val_acc: 0.8744
Epoch 13/30
7352/7352 [==============================] - 107s 15ms/step - loss: 0.1844 - acc: 0.9353
- val_loss: 0.2801 - val_acc: 0.8877
Epoch 14/30
7352/7352 [==============================] - 109s 15ms/step - loss: 0.1950 - acc: 0.9357
- val_loss: 0.3505 - val_acc: 0.8846
Epoch 15/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.2384 - acc: 0.9301 -
val_loss: 0.3624 - val_acc: 0.8799
Epoch 16/30
7352/7352 [==============================] - 61s 8ms/step - loss: 0.1716 - acc: 0.9411 - v
al_loss: 0.3616 - val_acc: 0.8921
Epoch 17/30
7352/7352 [==============================] - 62s 8ms/step - loss: 0.1527 - acc: 0.9476 - v
al_loss: 0.3772 - val_acc: 0.9016
Epoch 18/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.1812 - acc: 0.9412 -
val_loss: 0.6447 - val_acc: 0.8772
Epoch 19/30
7352/7352 [==============================] - 62s 8ms/step - loss: 0.1604 - acc: 0.9450 - v
al_loss: 0.3433 - val_acc: 0.8982
Epoch 20/30
7352/7352 [==============================] - 61s 8ms/step - loss: 0.1773 - acc: 0.9440 - v
al_loss: 0.3273 - val_acc: 0.9063
Epoch 21/30
7352/7352 [==============================] - 62s 8ms/step - loss: 0.1522 - acc: 0.9448 - v
al_loss: 0.3966 - val_acc: 0.8982
Epoch 22/30
7352/7352 [==============================] - 61s 8ms/step - loss: 0.1552 - acc: 0.9468 - v
al_loss: 0.3475 - val_acc: 0.8829
Epoch 23/30
7352/7352 [==============================] - 62s 8ms/step - loss: 0.1545 - acc: 0.9463 - v
al_loss: 0.3805 - val_acc: 0.8826
Epoch 24/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.1584 - acc: 0.9448 - v
al_loss: 0.5053 - val_acc: 0.8928
Epoch 25/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.1319 - acc: 0.9497 -
val_loss: 0.4133 - val_acc: 0.9121
Epoch 26/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1348 - acc: 0.9483 -
val_loss: 0.4508 - val_acc: 0.8968
Epoch 27/30
7352/7352 [==============================] - 68s 9ms/step - loss: 0.1386 - acc: 0.9479 - v
al_loss: 0.5065 - val_acc: 0.8914
Epoch 28/30
7352/7352 [==============================] - 77s 10ms/step - loss: 0.1351 - acc: 0.9501 -
val_loss: 0.5275 - val_acc: 0.9009
```

```
Epoch 29/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.1361 - acc: 0.9524 - v
al_loss: 0.5963 - val_acc: 0.8860
Epoch 30/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.1478 - acc: 0.9509 - v
al_loss: 0.5619 - val_acc: 0.8880
```

In [65]:

```
# Final evaluation of the model
scores1 = model1.evaluate(X_test, Y_test, verbose=1)
print("Test Score: %f" % (scores1[0]))
print("Test Accuracy: %f%%" % (scores1[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(model1.predict(X_test), axis=1)])

# Code for drawing seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downstairs','walking_upstairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_names, columns=
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', fontsize=14)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=90, ha='right', fontsize=14)
plt.ylabel('True label',size=18)
plt.xlabel('Predicted label',size=18)
plt.title("Confusion Matrix\n",size=24)
plt.show()
```

```
2947/2947 [==============================] - 3s 908us/step
Test Score: 0.561873
Test Accuracy: 88.802172%
```



(3) Model having 1 LSTM layer with 128 LSTM Units¶

In [23]:
```
# Initiliazing the sequential model
model2 = Sequential()
# Configuring the parameters
model2.add(LSTM(128, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model2.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model2.add(Dense(n_classes, activation='sigmoid'))
print(model2.summary())

# Compiling the model
model2.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])

# Training the model
history2 = model2.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test, Y_test),epochs=
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 128)               70656
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 774
=================================================================
Total params: 71,430
Trainable params: 71,430
Non-trainable params: 0
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 98s 13ms/step - loss: 1.2616 - acc: 0.4445 - val_loss: 1.1359 - val_acc: 0.5382
Epoch 2/30
7352/7352 [==============================] - 94s 13ms/step - loss: 0.9373 - acc: 0.5985 - val_loss: 0.9608 - val_acc: 0.5955
Epoch 3/30
7352/7352 [==============================] - 92s 13ms/step - loss: 0.7015 - acc: 0.7248 - val_loss: 0.6800 - val_acc: 0.7150
Epoch 4/30
7352/7352 [==============================] - 95s 13ms/step - loss: 0.4596 - acc: 0.8369 - val_loss: 0.4640 - val_acc: 0.8476
Epoch 5/30
7352/7352 [==============================] - 94s 13ms/step - loss: 0.2913 - acc: 0.9017 - val_loss: 0.4764 - val_acc: 0.8619
Epoch 6/30
7352/7352 [==============================] - 96s 13ms/step - loss: 0.2329 - acc: 0.9191 - val_loss: 0.3484 - val_acc: 0.8992
Epoch 7/30
7352/7352 [==============================] - 98s 13ms/step - loss: 0.1916 - acc: 0.9312 - val_loss: 0.3203 - val_acc: 0.8921
Epoch 8/30
7352/7352 [==============================] - 97s 13ms/step - loss: 0.1663 - acc: 0.9403 - val_loss: 0.4614 - val_acc: 0.8918
Epoch 9/30
7352/7352 [==============================] - 96s 13ms/step - loss: 0.1650 - acc: 0.9396 -
```

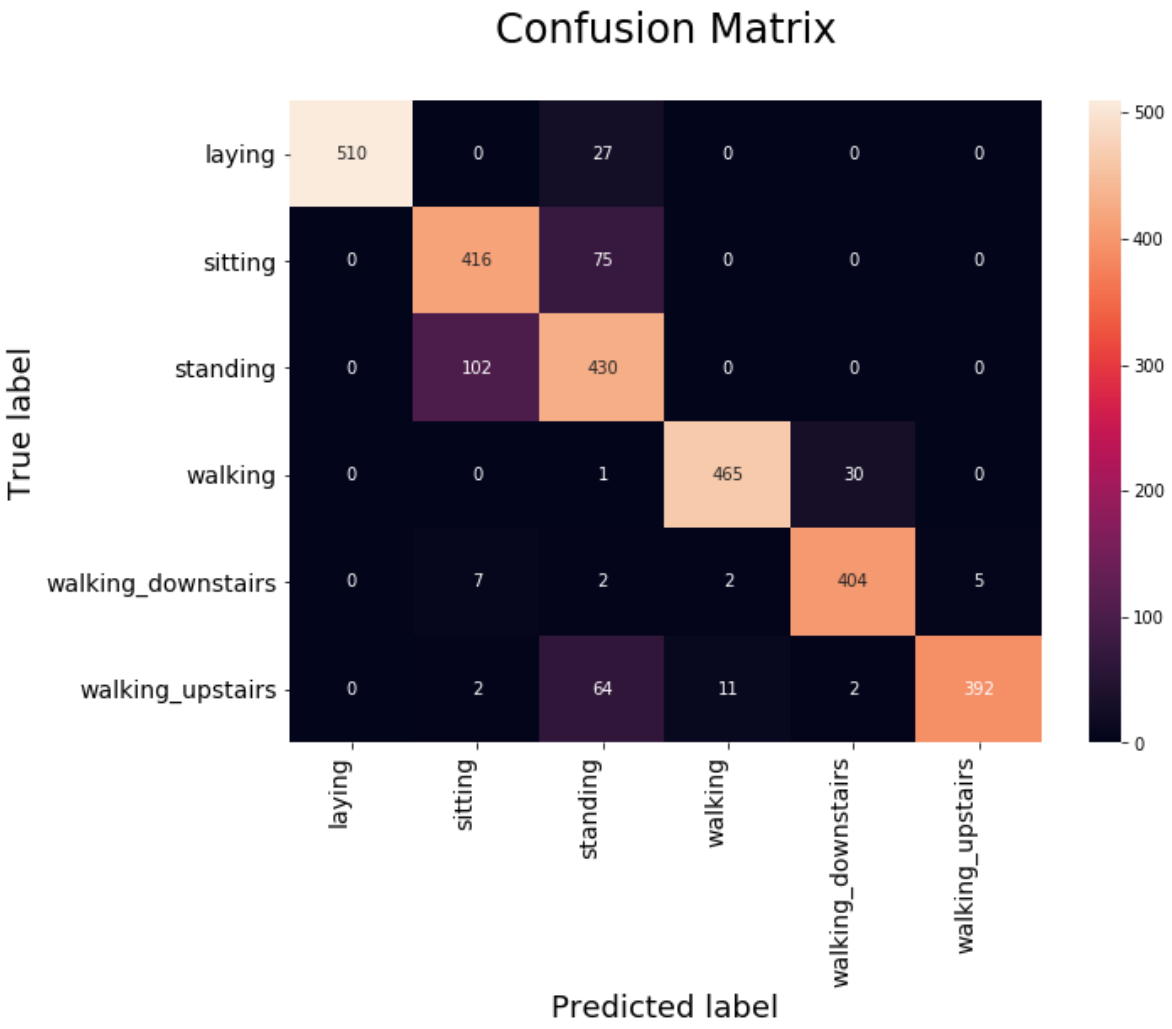
```
val_loss: 0.2804 - val_acc: 0.9104
Epoch 10/30
7352/7352 [==============================] - 99s 13ms/step - loss: 0.1619 - acc: 0.9436 -
val_loss: 0.3111 - val_acc: 0.9046
Epoch 11/30
7352/7352 [==============================] - 95s 13ms/step - loss: 0.1472 - acc: 0.9437 -
val_loss: 0.2696 - val_acc: 0.9213
Epoch 12/30
7352/7352 [==============================] - 97s 13ms/step - loss: 0.1721 - acc: 0.9388 -
val_loss: 0.4223 - val_acc: 0.8945
Epoch 13/30
7352/7352 [==============================] - 96s 13ms/step - loss: 0.1412 - acc: 0.9464 -
val_loss: 0.4510 - val_acc: 0.9111
Epoch 14/30
7352/7352 [==============================] - 95s 13ms/step - loss: 0.1460 - acc: 0.9484 -
val_loss: 0.4221 - val_acc: 0.9040
Epoch 15/30
7352/7352 [==============================] - 95s 13ms/step - loss: 0.1397 - acc: 0.9479 -
val_loss: 0.2759 - val_acc: 0.9196
Epoch 16/30
7352/7352 [==============================] - 98s 13ms/step - loss: 0.1485 - acc: 0.9478 -
val_loss: 0.3829 - val_acc: 0.8880
Epoch 17/30
7352/7352 [==============================] - 94s 13ms/step - loss: 0.1508 - acc: 0.9465 -
val_loss: 0.2784 - val_acc: 0.9148
Epoch 18/30
7352/7352 [==============================] - 94s 13ms/step - loss: 0.1285 - acc: 0.9474 -
val_loss: 0.3298 - val_acc: 0.9270
Epoch 19/30
7352/7352 [==============================] - 96s 13ms/step - loss: 0.1364 - acc: 0.9505 -
val_loss: 0.3332 - val_acc: 0.9087
Epoch 20/30
7352/7352 [==============================] - 96s 13ms/step - loss: 0.1262 - acc: 0.9533 -
val_loss: 0.4421 - val_acc: 0.9023
Epoch 21/30
7352/7352 [==============================] - 96s 13ms/step - loss: 0.1176 - acc: 0.9518 -
val_loss: 0.3397 - val_acc: 0.9264
Epoch 22/30
7352/7352 [==============================] - 98s 13ms/step - loss: 0.1410 - acc: 0.9478 -
val_loss: 0.3969 - val_acc: 0.9057
Epoch 23/30
7352/7352 [==============================] - 97s 13ms/step - loss: 0.1455 - acc: 0.9461 -
val_loss: 0.3380 - val_acc: 0.9097
Epoch 24/30
7352/7352 [==============================] - 98s 13ms/step - loss: 0.1307 - acc: 0.9513 -
val_loss: 0.3138 - val_acc: 0.9192
Epoch 25/30
7352/7352 [==============================] - 98s 13ms/step - loss: 0.1367 - acc: 0.9495 -
val_loss: 0.4656 - val_acc: 0.9179
Epoch 26/30
7352/7352 [==============================] - 97s 13ms/step - loss: 0.1267 - acc: 0.9484 -
val_loss: 0.3286 - val_acc: 0.9376
Epoch 27/30
7352/7352 [==============================] - 98s 13ms/step - loss: 0.1232 - acc: 0.9517 -
val_loss: 0.3801 - val_acc: 0.8982
Epoch 28/30
7352/7352 [==============================] - 98s 13ms/step - loss: 0.1223 - acc: 0.9533 -
```

```
val_loss: 0.3739 - val_acc: 0.9169
Epoch 29/30
7352/7352 [==============================] - 98s 13ms/step - loss: 0.1180 - acc: 0.9535 -
val_loss: 0.6495 - val_acc: 0.8904
Epoch 30/30
7352/7352 [==============================] - 97s 13ms/step - loss: 0.1412 - acc: 0.9527 -
val_loss: 0.5031 - val_acc: 0.8880
```

In [24]:

```
# Final evaluation of the model
scores2 = model2.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores2[0]))
print("Test Accuracy: %f%%" % (scores2[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(model2.predict(X_test), axis=1)])

# Code for drawing seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downstairs','walking_upstairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_names, columns=
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', fontsize=14)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=90, ha='right', fontsize=14)
plt.ylabel('True label',size=18)
plt.xlabel('Predicted label',size=18)
plt.title("Confusion Matrix\n",size=24)
plt.show()
```

```
Test Score: 0.503083
Test Accuracy: 88.802172%
```



(4) Model having 2 LSTM layer with 32 LSTM Units¶

In [25]:
```
# Initiliazing the sequential model
model3 = Sequential()
# Configuring the parameters
model3.add(LSTM(32,return_sequences=True, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model3.add(Dropout(0.5))

# Configuring the parameters
model3.add(LSTM(32))
# Adding a dropout layer
model3.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model3.add(Dense(n_classes, activation='sigmoid'))
print(model3.summary())

# Compiling the model
model3.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])

# Training the model
history3 = model3.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test, Y_test),epochs=
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_4 (LSTM)                (None, 128, 32)           5376
_________________________________________________________________
dropout_4 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
lstm_5 (LSTM)                (None, 32)                8320
_________________________________________________________________
dropout_5 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_4 (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 [==============================] - 107s 15ms/step - loss: 1.2150 - acc: 0.4944
- val_loss: 0.9705 - val_acc: 0.5847
Epoch 2/30
7352/7352 [==============================] - 101s 14ms/step - loss: 0.8392 - acc: 0.6152
- val_loss: 0.7750 - val_acc: 0.6634
Epoch 3/30
7352/7352 [==============================] - 113s 15ms/step - loss: 0.7124 - acc: 0.6680
- val_loss: 0.7619 - val_acc: 0.6688
Epoch 4/30
7352/7352 [==============================] - 133s 18ms/step - loss: 0.6315 - acc: 0.7221
- val_loss: 0.7247 - val_acc: 0.6990
Epoch 5/30
7352/7352 [==============================] - 109s 15ms/step - loss: 0.5048 - acc: 0.7753
- val_loss: 0.7067 - val_acc: 0.7177
Epoch 6/30
7352/7352 [==============================] - 104s 14ms/step - loss: 0.4320 - acc: 0.7996
```

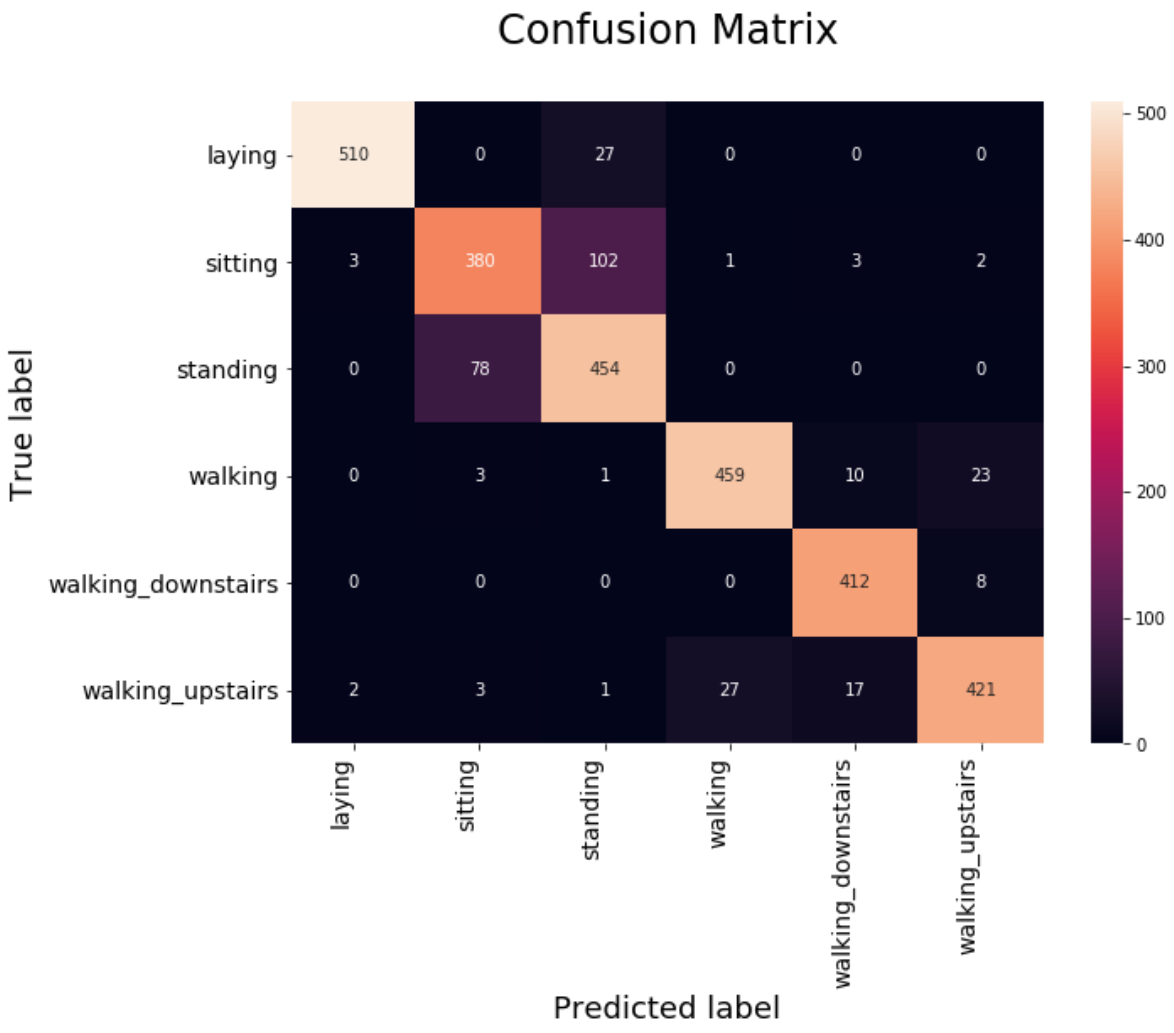
```
- val_loss: 0.7591 - val_acc: 0.7275
Epoch 7/30
7352/7352 [==============================] - 104s 14ms/step - loss: 0.4175 - acc: 0.8292
- val_loss: 0.9364 - val_acc: 0.7492
Epoch 8/30
7352/7352 [==============================] - 105s 14ms/step - loss: 0.3415 - acc: 0.8876
- val_loss: 0.5570 - val_acc: 0.8537
Epoch 9/30
7352/7352 [==============================] - 105s 14ms/step - loss: 0.2640 - acc: 0.9184
- val_loss: 0.5190 - val_acc: 0.8792
Epoch 10/30
7352/7352 [==============================] - 104s 14ms/step - loss: 0.2352 - acc: 0.9310
- val_loss: 0.4971 - val_acc: 0.8605
Epoch 11/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.2060 - acc: 0.9365
- val_loss: 0.4087 - val_acc: 0.8931
Epoch 12/30
7352/7352 [==============================] - 184s 25ms/step - loss: 0.1909 - acc: 0.9381
- val_loss: 0.3961 - val_acc: 0.8968
Epoch 13/30
7352/7352 [==============================] - 113s 15ms/step - loss: 0.1909 - acc: 0.9380
- val_loss: 0.4197 - val_acc: 0.8887
Epoch 14/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1984 - acc: 0.9373
- val_loss: 0.4044 - val_acc: 0.8846
Epoch 15/30
7352/7352 [==============================] - 106s 14ms/step - loss: 0.1771 - acc: 0.9389
- val_loss: 0.3914 - val_acc: 0.8931
Epoch 16/30
7352/7352 [==============================] - 107s 15ms/step - loss: 0.1879 - acc: 0.9411
- val_loss: 0.3429 - val_acc: 0.9046
Epoch 17/30
7352/7352 [==============================] - 107s 15ms/step - loss: 0.1648 - acc: 0.9474
- val_loss: 0.4437 - val_acc: 0.8979
Epoch 18/30
7352/7352 [==============================] - 106s 14ms/step - loss: 0.1613 - acc: 0.9444
- val_loss: 0.4181 - val_acc: 0.8996
Epoch 19/30
7352/7352 [==============================] - 206s 28ms/step - loss: 0.1868 - acc: 0.9414
- val_loss: 0.3927 - val_acc: 0.8599
Epoch 20/30
7352/7352 [==============================] - 136s 18ms/step - loss: 0.1649 - acc: 0.9445
- val_loss: 0.4569 - val_acc: 0.9040
Epoch 21/30
7352/7352 [==============================] - 127s 17ms/step - loss: 0.1526 - acc: 0.9468
- val_loss: 0.4284 - val_acc: 0.8982
Epoch 22/30
7352/7352 [==============================] - 109s 15ms/step - loss: 0.2004 - acc: 0.9404
- val_loss: 0.4617 - val_acc: 0.9074
Epoch 23/30
7352/7352 [==============================] - 106s 14ms/step - loss: 0.1710 - acc: 0.9455
- val_loss: 0.5515 - val_acc: 0.8833
Epoch 24/30
7352/7352 [==============================] - 105s 14ms/step - loss: 0.1587 - acc: 0.9465
- val_loss: 0.5813 - val_acc: 0.8904
Epoch 25/30
7352/7352 [==============================] - 104s 14ms/step - loss: 0.1556 - acc: 0.9476
```

```
- val_loss: 0.5234 - val_acc: 0.8894
Epoch 26/30
7352/7352 [==============================] - 103s 14ms/step - loss: 0.1928 - acc: 0.9327
- val_loss: 0.5000 - val_acc: 0.8880
Epoch 27/30
7352/7352 [==============================] - 137s 19ms/step - loss: 0.1472 - acc: 0.9465
- val_loss: 0.4803 - val_acc: 0.8890
Epoch 28/30
7352/7352 [==============================] - 131s 18ms/step - loss: 0.1565 - acc: 0.9489
- val_loss: 0.4395 - val_acc: 0.9043
Epoch 29/30
7352/7352 [==============================] - 105s 14ms/step - loss: 0.1459 - acc: 0.9440
- val_loss: 0.3842 - val_acc: 0.9118
Epoch 30/30
7352/7352 [==============================] - 107s 15ms/step - loss: 0.1439 - acc: 0.9521
- val_loss: 0.4543 - val_acc: 0.8945
```

In [26]:

```
# Final evaluation of the model
scores3 = model3.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores3[0]))
print("Test Accuracy: %f%%" % (scores3[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(model3.predict(X_test), axis=1)])

# Code for drawing seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downstairs','walking_upstairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_names, columns=
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', fontsize=14)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=90, ha='right', fontsize=14)
plt.ylabel('True label',size=18)
plt.xlabel('Predicted label',size=18)
plt.title("Confusion Matrix\n",size=24)
plt.show()
```

Test Score: 0.454262
Test Accuracy: 89.446895%



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

In [27]:

```
model4= Sequential()
# Configuring the parameters
model4.add(LSTM(64,return_sequences=True, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model4.add(Dropout(0.7))

# Configuring the parameters
model4.add(LSTM(64))
# Adding a dropout layer
model4.add(Dropout(0.7))
# 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=
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_6 (LSTM)                (None, 128, 64)           18944
_________________________________________________________________
dropout_6 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
lstm_7 (LSTM)                (None, 64)                33024
_________________________________________________________________
dropout_7 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_5 (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: 1.1618 - acc: 0.5181
- val_loss: 1.1417 - val_acc: 0.5494
Epoch 2/30
7352/7352 [==============================] - 136s 18ms/step - loss: 0.8034 - acc: 0.6532
- val_loss: 1.0669 - val_acc: 0.5762
Epoch 3/30
7352/7352 [==============================] - 136s 18ms/step - loss: 0.7067 - acc: 0.6850
- val_loss: 0.7242 - val_acc: 0.7041
Epoch 4/30
7352/7352 [==============================] - 136s 18ms/step - loss: 0.6044 - acc: 0.7493
- val_loss: 0.8400 - val_acc: 0.7112
Epoch 5/30
7352/7352 [==============================] - 134s 18ms/step - loss: 0.5143 - acc: 0.7851
- val_loss: 0.6612 - val_acc: 0.7445
Epoch 6/30
7352/7352 [==============================] - 238s 32ms/step - loss: 0.4299 - acc: 0.8039
- val_loss: 0.6006 - val_acc: 0.7587
```

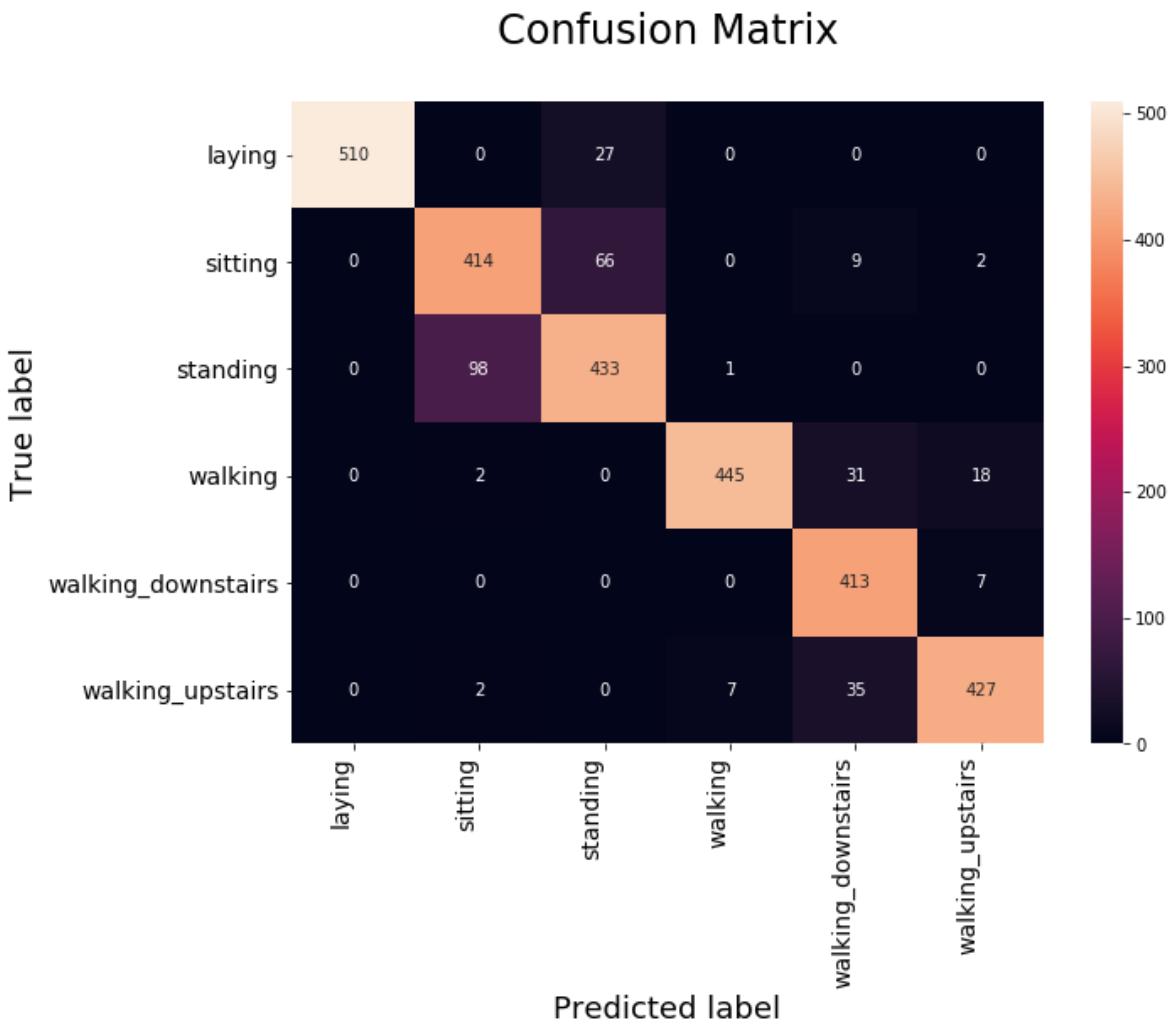
```
Epoch 7/30
7352/7352 [==============================] - 209s 28ms/step - loss: 0.4120 - acc: 0.8625
- val_loss: 0.4780 - val_acc: 0.8595
Epoch 8/30
7352/7352 [==============================] - 169s 23ms/step - loss: 0.2728 - acc: 0.9168
- val_loss: 0.4273 - val_acc: 0.8836
Epoch 9/30
7352/7352 [==============================] - 182s 25ms/step - loss: 0.2537 - acc: 0.9271
- val_loss: 0.6533 - val_acc: 0.8541
Epoch 10/30
7352/7352 [==============================] - 206s 28ms/step - loss: 0.2466 - acc: 0.9295
- val_loss: 0.5366 - val_acc: 0.8765
Epoch 11/30
7352/7352 [==============================] - 172s 23ms/step - loss: 0.2245 - acc: 0.9323
- val_loss: 0.4275 - val_acc: 0.8965
Epoch 12/30
7352/7352 [==============================] - 177s 24ms/step - loss: 0.1916 - acc: 0.9411
- val_loss: 0.4206 - val_acc: 0.8904
Epoch 13/30
7352/7352 [==============================] - 168s 23ms/step - loss: 0.2025 - acc: 0.9344
- val_loss: 0.4532 - val_acc: 0.8945
Epoch 14/30
7352/7352 [==============================] - 170s 23ms/step - loss: 0.1905 - acc: 0.9343
- val_loss: 0.6007 - val_acc: 0.8931
Epoch 15/30
7352/7352 [==============================] - 175s 24ms/step - loss: 0.2042 - acc: 0.9363
- val_loss: 0.5489 - val_acc: 0.8904
Epoch 16/30
7352/7352 [==============================] - 197s 27ms/step - loss: 0.1855 - acc: 0.9412
- val_loss: 0.5627 - val_acc: 0.9002
Epoch 17/30
7352/7352 [==============================] - 241s 33ms/step - loss: 0.2171 - acc: 0.9382
- val_loss: 0.5950 - val_acc: 0.8935
Epoch 18/30
7352/7352 [==============================] - 186s 25ms/step - loss: 0.1718 - acc: 0.9449
- val_loss: 0.7397 - val_acc: 0.8833
Epoch 19/30
7352/7352 [==============================] - 191s 26ms/step - loss: 0.2656 - acc: 0.9359
- val_loss: 0.5967 - val_acc: 0.8819
Epoch 20/30
7352/7352 [==============================] - 174s 24ms/step - loss: 0.1631 - acc: 0.9437
- val_loss: 0.7020 - val_acc: 0.8867
Epoch 21/30
7352/7352 [==============================] - 165s 22ms/step - loss: 0.1682 - acc: 0.9459
- val_loss: 0.8516 - val_acc: 0.8541
Epoch 22/30
7352/7352 [==============================] - 173s 24ms/step - loss: 0.1906 - acc: 0.9414
- val_loss: 0.6821 - val_acc: 0.8833
Epoch 23/30
7352/7352 [==============================] - 188s 26ms/step - loss: 0.1681 - acc: 0.9448
- val_loss: 0.5941 - val_acc: 0.8921
Epoch 24/30
7352/7352 [==============================] - 162s 22ms/step - loss: 0.1902 - acc: 0.9436
- val_loss: 0.5133 - val_acc: 0.8962
Epoch 25/30
7352/7352 [==============================] - 142s 19ms/step - loss: 0.2180 - acc: 0.9332
- val_loss: 0.5660 - val_acc: 0.8955
```

```
Epoch 26/30
7352/7352 [==============================] - 159s 22ms/step - loss: 0.1675 - acc: 0.9448
- val_loss: 0.5508 - val_acc: 0.8928
Epoch 27/30
7352/7352 [==============================] - 187s 25ms/step - loss: 0.1694 - acc: 0.9478
- val_loss: 0.7239 - val_acc: 0.8799
Epoch 28/30
7352/7352 [==============================] - 167s 23ms/step - loss: 0.1584 - acc: 0.9479
- val_loss: 0.8545 - val_acc: 0.8856
Epoch 29/30
7352/7352 [==============================] - 173s 24ms/step - loss: 0.1459 - acc: 0.9521
- val_loss: 0.6020 - val_acc: 0.9013
Epoch 30/30
7352/7352 [==============================] - 154s 21ms/step - loss: 0.1451 - acc: 0.9525
- val_loss: 0.6563 - val_acc: 0.8965
```

In [28]:

```
# Final evaluation of the model
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), axis=1)])

# Code for drawing seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downstairs','walking_upstairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_names, columns=
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', fontsize=14)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=90, ha='right', fontsize=14)
plt.ylabel('True label',size=18)
plt.xlabel('Predicted label',size=18)
plt.title("Confusion Matrix\n",size=24)
plt.show()
```

Test Score: 0.656322
Test Accuracy: 89.650492%



(6) Model having 2 LSTM layer with 128 LSTM Units¶

In [29]:

```
model5= Sequential()
# Configuring the parameters
model5.add(LSTM(64,return_sequences=True, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model4.add(Dropout(0.7))

# Configuring the parameters
model5.add(LSTM(64))
# Adding a dropout layer
model5.add(Dropout(0.7))
# Adding a dense output layer with sigmoid activation
model5.add(Dense(n_classes, activation='sigmoid'))
print(model5.summary())

# Compiling the model
model5.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])

# Training the model
history5= model5.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test, Y_test),epochs=
```

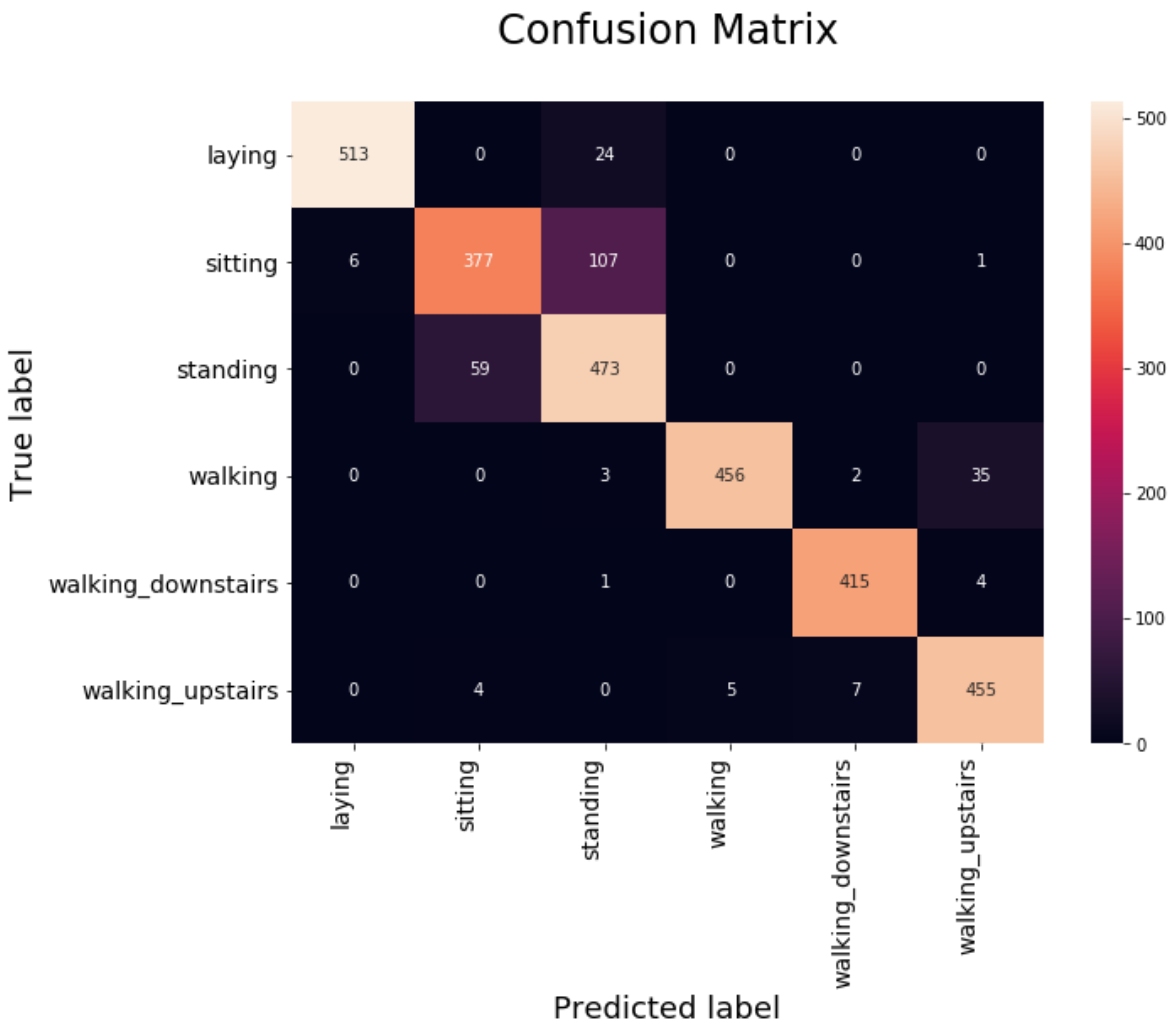
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_8 (LSTM)                (None, 128, 64)           18944
_________________________________________________________________
lstm_9 (LSTM)                (None, 64)                33024
_________________________________________________________________
dropout_9 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_6 (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 [==============================] - 148s 20ms/step - loss: 1.1115 - acc: 0.5272
- val_loss: 0.9108 - val_acc: 0.6400
Epoch 2/30
7352/7352 [==============================] - 157s 21ms/step - loss: 0.8765 - acc: 0.6412
- val_loss: 0.7469 - val_acc: 0.7156
Epoch 3/30
7352/7352 [==============================] - 146s 20ms/step - loss: 0.7262 - acc: 0.7121
- val_loss: 0.7786 - val_acc: 0.7262
Epoch 4/30
7352/7352 [==============================] - 154s 21ms/step - loss: 0.6651 - acc: 0.7462
- val_loss: 0.6932 - val_acc: 0.7363
Epoch 5/30
7352/7352 [==============================] - 172s 23ms/step - loss: 0.6076 - acc: 0.7535
- val_loss: 0.6103 - val_acc: 0.7811
Epoch 6/30
7352/7352 [==============================] - 218s 30ms/step - loss: 0.5689 - acc: 0.8335
- val_loss: 0.5340 - val_acc: 0.8385
Epoch 7/30
7352/7352 [==============================] - 169s 23ms/step - loss: 0.3667 - acc: 0.8998
```

```
- val_loss: 0.6557 - val_acc: 0.8154
Epoch 8/30
7352/7352 [==============================] - 155s 21ms/step - loss: 0.2571 - acc: 0.9215
- val_loss: 0.3751 - val_acc: 0.8873
Epoch 9/30
7352/7352 [==============================] - 161s 22ms/step - loss: 0.2130 - acc: 0.9342
- val_loss: 0.4391 - val_acc: 0.8894
Epoch 10/30
7352/7352 [==============================] - 186s 25ms/step - loss: 0.2192 - acc: 0.9366
- val_loss: 0.4561 - val_acc: 0.8982
Epoch 11/30
7352/7352 [==============================] - 170s 23ms/step - loss: 0.1805 - acc: 0.9387
- val_loss: 0.4012 - val_acc: 0.8968
Epoch 12/30
7352/7352 [==============================] - 162s 22ms/step - loss: 0.1977 - acc: 0.9338
- val_loss: 0.5979 - val_acc: 0.8717
Epoch 13/30
7352/7352 [==============================] - 172s 23ms/step - loss: 0.1637 - acc: 0.9448
- val_loss: 0.4463 - val_acc: 0.9026
Epoch 14/30
7352/7352 [==============================] - 179s 24ms/step - loss: 0.1713 - acc: 0.9448
- val_loss: 0.5005 - val_acc: 0.9053
Epoch 15/30
7352/7352 [==============================] - 161s 22ms/step - loss: 0.1641 - acc: 0.9499
- val_loss: 0.4298 - val_acc: 0.9033
Epoch 16/30
7352/7352 [==============================] - 142s 19ms/step - loss: 0.1671 - acc: 0.9438
- val_loss: 0.4132 - val_acc: 0.9006
Epoch 17/30
7352/7352 [==============================] - 147s 20ms/step - loss: 0.1670 - acc: 0.9445
- val_loss: 0.4367 - val_acc: 0.9043
Epoch 18/30
7352/7352 [==============================] - 147s 20ms/step - loss: 0.1594 - acc: 0.9474
- val_loss: 0.4806 - val_acc: 0.8955
Epoch 19/30
7352/7352 [==============================] - 166s 23ms/step - loss: 0.1501 - acc: 0.9502
- val_loss: 0.2782 - val_acc: 0.9301
Epoch 20/30
7352/7352 [==============================] - 169s 23ms/step - loss: 0.1421 - acc: 0.9499
- val_loss: 0.3967 - val_acc: 0.9155
Epoch 21/30
7352/7352 [==============================] - 171s 23ms/step - loss: 0.1525 - acc: 0.9489
- val_loss: 0.4528 - val_acc: 0.8958
Epoch 22/30
7352/7352 [==============================] - 163s 22ms/step - loss: 0.1496 - acc: 0.9506
- val_loss: 0.4937 - val_acc: 0.9040
Epoch 23/30
7352/7352 [==============================] - 143s 19ms/step - loss: 0.1707 - acc: 0.9487
- val_loss: 0.4406 - val_acc: 0.8887
Epoch 24/30
7352/7352 [==============================] - 130s 18ms/step - loss: 0.1568 - acc: 0.9472
- val_loss: 0.3643 - val_acc: 0.9111
Epoch 25/30
7352/7352 [==============================] - 132s 18ms/step - loss: 0.1423 - acc: 0.9520
- val_loss: 0.3914 - val_acc: 0.9043
Epoch 26/30
7352/7352 [==============================] - 134s 18ms/step - loss: 0.1380 - acc: 0.9459
```

```
- val_loss: 0.4908 - val_acc: 0.9158
Epoch 27/30
7352/7352 [==============================] - 134s 18ms/step - loss: 0.1361 - acc: 0.9491
- val_loss: 0.3442 - val_acc: 0.9206
Epoch 28/30
7352/7352 [==============================] - 136s 18ms/step - loss: 0.1531 - acc: 0.9482
- val_loss: 0.3601 - val_acc: 0.9237
Epoch 29/30
7352/7352 [==============================] - 173s 24ms/step - loss: 0.1450 - acc: 0.9489
- val_loss: 0.4572 - val_acc: 0.9135
Epoch 30/30
7352/7352 [==============================] - 147s 20ms/step - loss: 0.1495 - acc: 0.9504
- val_loss: 0.3938 - val_acc: 0.9125
```

In [30]:
```
# Final evaluation of the model
scores5 = model5.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores5[0]))
print("Test Accuracy: %f%%" % (scores5[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(model5.predict(X_test), axis=1)])

# Code for drawing seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downstairs','walking_upstairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_names, columns=
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', fontsize=14)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=90, ha='right', fontsize=14)
plt.ylabel('True label',size=18)
plt.xlabel('Predicted label',size=18)
plt.title("Confusion Matrix\n",size=24)
plt.show()
```

Test Score: 0.393826
Test Accuracy: 91.245334%



(7) Model with 1 LSTM Layer having 128 LSTM units + Batch Normalization

In [40]:
```
from keras.layers.normalization import BatchNormalization
from keras.optimizers import RMSprop
model6 = Sequential() # Initiliazing the sequential model3
model6.add(LSTM(128, input_shape=(timesteps, input_dim))) # Configuring the parameters
model6.add(BatchNormalization())
model6.add(Dropout(0.40)) # Adding a dropout layer
model6.add(Dense(n_classes, activation='sigmoid')) # Adding a dense output layer with sigmoid acti
model6.summary()

optim=RMSprop(epsilon=0.00001, decay=1e-6, clipnorm =1)
model6.compile(loss='categorical_crossentropy', optimizer=optim, metrics=['accuracy']) # Compilin

# Training the model
history6= model6.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test, Y_test),epochs=
```

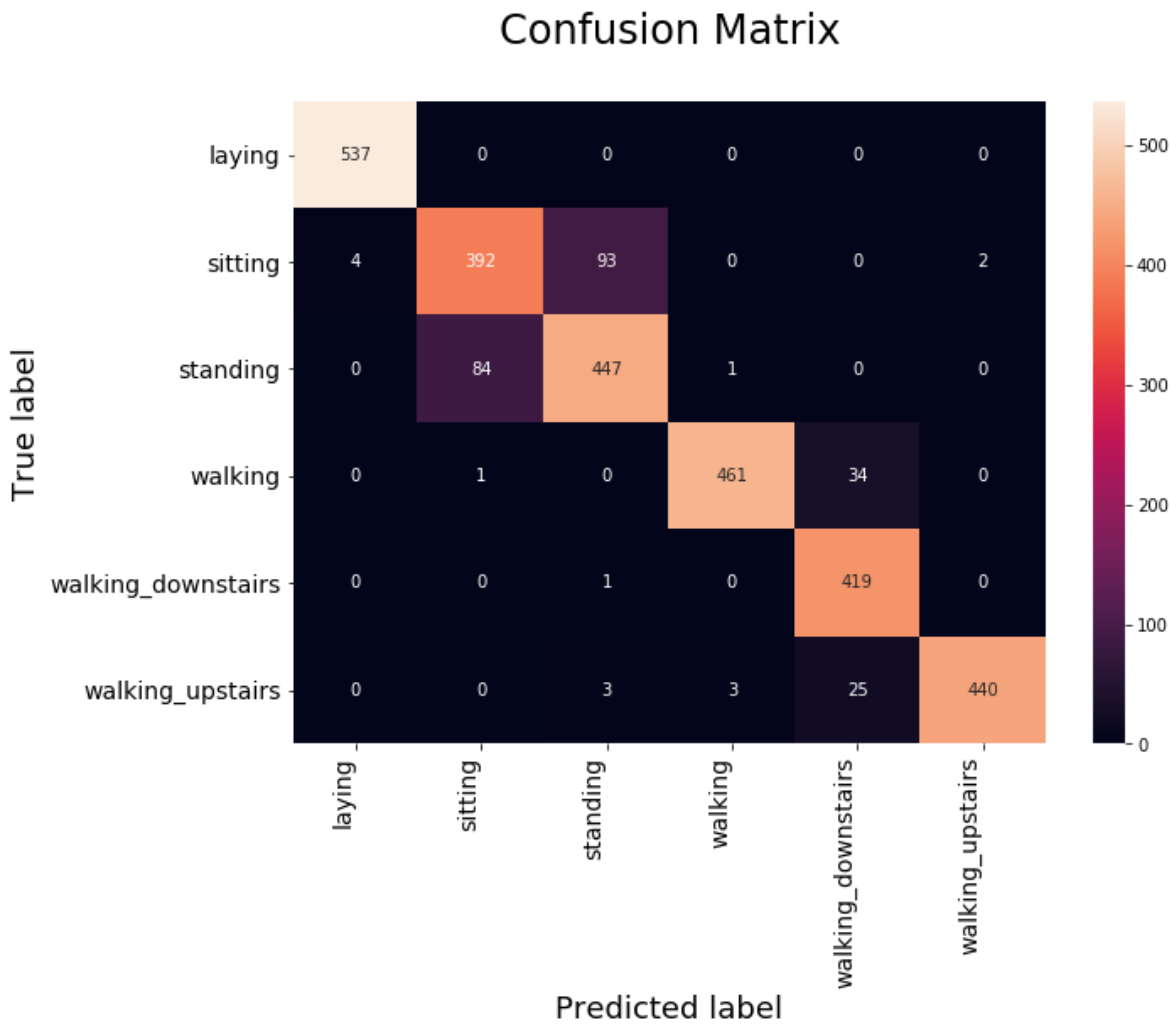
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_15 (LSTM)               (None, 128)               70656
_________________________________________________________________
batch_normalization_5 (Batch (None, 128)               512
_________________________________________________________________
dropout_14 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_11 (Dense)             (None, 6)                 774
=================================================================
Total params: 71,942
Trainable params: 71,686
Non-trainable params: 256
_________________________________________________________________
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.9934 - acc: 0.5604 -
val_loss: 0.8194 - val_acc: 0.5894
Epoch 2/30
7352/7352 [==============================] - 83s 11ms/step - loss: 0.7753 - acc: 0.6349 -
val_loss: 0.8324 - val_acc: 0.6155
Epoch 3/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.6902 - acc: 0.6683 -
val_loss: 0.7181 - val_acc: 0.6695
Epoch 4/30
7352/7352 [==============================] - 83s 11ms/step - loss: 0.5726 - acc: 0.7743 -
val_loss: 0.7140 - val_acc: 0.8181
Epoch 5/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.4352 - acc: 0.8677 -
val_loss: 0.3463 - val_acc: 0.8833
Epoch 6/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.2683 - acc: 0.9211 -
val_loss: 0.5163 - val_acc: 0.8768
Epoch 7/30
7352/7352 [==============================] - 83s 11ms/step - loss: 0.3089 - acc: 0.9055 -
val_loss: 0.3130 - val_acc: 0.8972
Epoch 8/30
7352/7352 [==============================] - 84s 11ms/step - loss: 0.2179 - acc: 0.9331 -
val_loss: 0.4856 - val_acc: 0.8850
Epoch 9/30
```

```
7352/7352 [==============================] - 94s 13ms/step - loss: 0.1798 - acc: 0.9418 - val_loss: 0.2862 - val_acc: 0.9267
Epoch 10/30
7352/7352 [==============================] - 100s 14ms/step - loss: 0.1713 - acc: 0.9468 - val_loss: 0.3581 - val_acc: 0.9196
Epoch 11/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.1941 - acc: 0.9408 - val_loss: 0.2452 - val_acc: 0.9359
Epoch 12/30
7352/7352 [==============================] - 97s 13ms/step - loss: 0.1787 - acc: 0.9452 - val_loss: 0.3084 - val_acc: 0.9114
Epoch 13/30
7352/7352 [==============================] - 119s 16ms/step - loss: 0.1588 - acc: 0.9461 - val_loss: 0.4043 - val_acc: 0.8985
Epoch 14/30
7352/7352 [==============================] - 114s 15ms/step - loss: 0.1514 - acc: 0.9491 - val_loss: 0.4234 - val_acc: 0.9165
Epoch 15/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.1626 - acc: 0.9490 - val_loss: 0.3365 - val_acc: 0.9087
Epoch 16/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.1370 - acc: 0.9516 - val_loss: 0.5282 - val_acc: 0.9019
Epoch 17/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.1548 - acc: 0.9499 - val_loss: 0.2353 - val_acc: 0.9342
Epoch 18/30
7352/7352 [==============================] - 92s 13ms/step - loss: 0.1400 - acc: 0.9520 - val_loss: 0.4388 - val_acc: 0.9094
Epoch 19/30
7352/7352 [==============================] - 94s 13ms/step - loss: 0.1841 - acc: 0.9471 - val_loss: 0.4176 - val_acc: 0.9104
Epoch 20/30
7352/7352 [==============================] - 96s 13ms/step - loss: 0.1846 - acc: 0.9478 - val_loss: 0.4138 - val_acc: 0.9118
Epoch 21/30
7352/7352 [==============================] - 93s 13ms/step - loss: 0.1480 - acc: 0.9504 - val_loss: 0.4467 - val_acc: 0.9196
Epoch 22/30
7352/7352 [==============================] - 107s 15ms/step - loss: 0.1887 - acc: 0.9463 - val_loss: 0.4077 - val_acc: 0.9165
Epoch 23/30
7352/7352 [==============================] - 135s 18ms/step - loss: 0.1708 - acc: 0.9471 - val_loss: 0.3943 - val_acc: 0.9152
Epoch 24/30
7352/7352 [==============================] - 107s 15ms/step - loss: 0.1505 - acc: 0.9518 - val_loss: 0.3944 - val_acc: 0.9233
Epoch 25/30
7352/7352 [==============================] - 138s 19ms/step - loss: 0.1422 - acc: 0.9528 - val_loss: 0.3664 - val_acc: 0.9175
Epoch 26/30
7352/7352 [==============================] - 97s 13ms/step - loss: 0.1378 - acc: 0.9540 - val_loss: 0.4202 - val_acc: 0.9203
Epoch 27/30
7352/7352 [==============================] - 95s 13ms/step - loss: 0.1605 - acc: 0.9497 - val_loss: 0.4970 - val_acc: 0.9175
Epoch 28/30
```

```
7352/7352 [==============================] - 96s 13ms/step - loss: 0.1504 - acc: 0.9524 - val_loss: 0.4115 - val_acc: 0.9203
Epoch 29/30
7352/7352 [==============================] - 97s 13ms/step - loss: 0.1391 - acc: 0.9517 - val_loss: 0.6128 - val_acc: 0.8924
Epoch 30/30
7352/7352 [==============================] - 94s 13ms/step - loss: 0.1760 - acc: 0.9484 - val_loss: 0.4541 - val_acc: 0.9148
```

In [60]:
```
# Final evaluation of the model
scores6 = model6.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores6[0]))
print("Test Accuracy: %f%%" % (scores6[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(model6.predict(X_test), axis=1)])

# Code for drawing seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downstairs','walking_upstairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_names, columns=
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', fontsize=14)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=90, ha='right', fontsize=14)
plt.ylabel('True label',size=18)
plt.xlabel('Predicted label',size=18)
plt.title("Confusion Matrix\n",size=24)
plt.show()
```

Test Score: 0.454131
Test Accuracy: 91.482864%



CONCLUSION

In [66]:

```
# Creating table using PrettyTable library
from prettytable import PrettyTable

# Names of models
names =['1 LSTM layer with 32 LSTM Units(Optimizer-->rmsprop)','1 LSTM layer with 64 LSTM Uni
      '1 LSTM layer with 128 LSTM Units(Optimizer-->rmsprop)','2 LSTM layer with 32 LSTM Units(O
      '2 LSTM layer with 64 LSTM Units(Optimizer-->rmsprop)','2 LSTM layer with 128 LSTM Units(O
     ' 1 LSTM layer with 128 LSTM Units(Optimizer-->rmsprop)+BatchNorm']

# Training accuracies
train_acc = [history.history['acc'][10],history.history['acc'][29],history2.history['acc'][29],\
         history3.history['acc'][29],history4.history['acc'][29],history5.history['acc'][29],history6.hist

# Test accuracies
test_acc =[scores[1],scores1[1],scores2[1],scores3[1],scores4[1],scores5[1],scores6[1]]

numbering = [1,2,3,4,5,6,7]

# Initializing prettytable
ptable = PrettyTable()

# Adding columns
ptable.add_column("S.NO.",numbering)
ptable.add_column("MODEL",names)
ptable.add_column("Training Accuracy",train_acc)
ptable.add_column("Test Accuracy",test_acc)

# Printing the Table
print(ptable)
```

```
+-------+-----------------------------------------------------------------+-------------------+--------------------+
| S.NO. |                      MODEL                      | Training Accuracy |   Test Accuracy    |
+-------+-----------------------------------------------------------------+-------------------+--------------------+
|   1   |     1 LSTM layer with 32 LSTM Units(Optimizer-->rmsprop)      | 0.7729869423286181 | 0.8
900576857821514 |
|   2   |     1 LSTM layer with 64 LSTM Units(Optimizer-->rmsprop)      | 0.9460010881392819 | 0.8
880217170003394 |
|   3   |     1 LSTM layer with 128 LSTM Units(Optimizer-->rmsprop)      | 0.9526659412404788 | 0.
8880217170205649 |
|   4   |     2 LSTM layer with 32 LSTM Units(Optimizer-->rmsprop)      | 0.9521218715995647 | 0.8
944689514760774 |
|   5   |     2 LSTM layer with 64 LSTM Units(Optimizer-->rmsprop)      | 0.9525299238302503 | 0.8
965049202578894 |
|   6   |     2 LSTM layer with 128 LSTM Units(Optimizer-->rmsprop)      | 0.9503536452665942 | 0.
9124533423820834 |
|   7   | 1 LSTM layer with 128 LSTM Units(Optimizer-->rmsprop)+BatchNorm | 0.9484494015233
95  | 0.9148286392941974 |
+-------+-----------------------------------------------------------------+-------------------+--------------------+
```

Sl. no 6 and 7 are giving a accuracy of almost 91.5%

In [ ]: