In [1]:

```
# Importing Libraries
```

In [2]:

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

In [3]:

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

## Data

In [4]:

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

In [5]:

```
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()
        )

    return np.transpose(signals_data, (1, 2, 0))
```

In [7]:

```
def load_y(subset):

    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]:

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

In [10]:

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

In [11]:

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

```
Using TensorFlow backend.
```

In [12]:

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

In [13]:

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

In [14]:

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

In [15]:

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

In [16]:

```
timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
```

```
n_classes = _count_classes(Y_train)

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

```
128
9
7352
```

# (1) Single LSTM layer with 32-LSTM Units

In [17]:

```
model = Sequential()
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
model.add(Dropout(0.5))
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()

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

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

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_1 (LSTM)                (None, 32)                5376      
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0         
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 198       
=================================================================
Total params: 5,574
Trainable params: 5,574
Non-trainable params: 0
_________________________________________________________________
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 40s 5ms/step - loss: 1.3139 - acc: 0.4358 - val_loss:
1.1352 - val_acc: 0.4700
Epoch 2/30
7352/7352 [==============================] - 38s 5ms/step - loss: 0.9788 - acc: 0.5773 - val_loss:
0.9513 - val_acc: 0.5884
Epoch 3/30
7352/7352 [==============================] - 39s 5ms/step - loss: 0.7977 - acc: 0.6457 - val_loss:
0.8343 - val_acc: 0.6013
Epoch 4/30
7352/7352 [==============================] - 39s 5ms/step - loss: 0.6989 - acc: 0.6582 - val_loss:
0.7532 - val_acc: 0.6098
Epoch 5/30
7352/7352 [==============================] - 37s 5ms/step - loss: 0.6359 - acc: 0.6797 - val_loss:
0.7335 - val_acc: 0.6183
Epoch 6/30
7352/7352 [==============================] - 37s 5ms/step - loss: 0.5819 - acc: 0.6865 - val_loss:
0.8786 - val_acc: 0.6098
Epoch 7/30
7352/7352 [==============================] - 37s 5ms/step - loss: 0.5676 - acc: 0.7058 - val_loss:
0.8191 - val_acc: 0.6132
Epoch 8/30
7352/7352 [==============================] - 37s 5ms/step - loss: 0.5583 - acc: 0.7217 - val_loss:
0.6639 - val_acc: 0.7190
Epoch 9/30
7352/7352 [==============================] - 37s 5ms/step - loss: 0.5386 - acc: 0.7557 - val_loss:
0.6388 - val_acc: 0.7167
Epoch 10/30
7352/7352 [==============================] - 37s 5ms/step - loss: 0.4804 - acc: 0.7911 - val_loss:
0.5077 - val_acc: 0.7509
Epoch 11/30
7352/7352 [==============================] - 37s 5ms/step - loss: 0.4320 - acc: 0.8052 - val_loss:
```

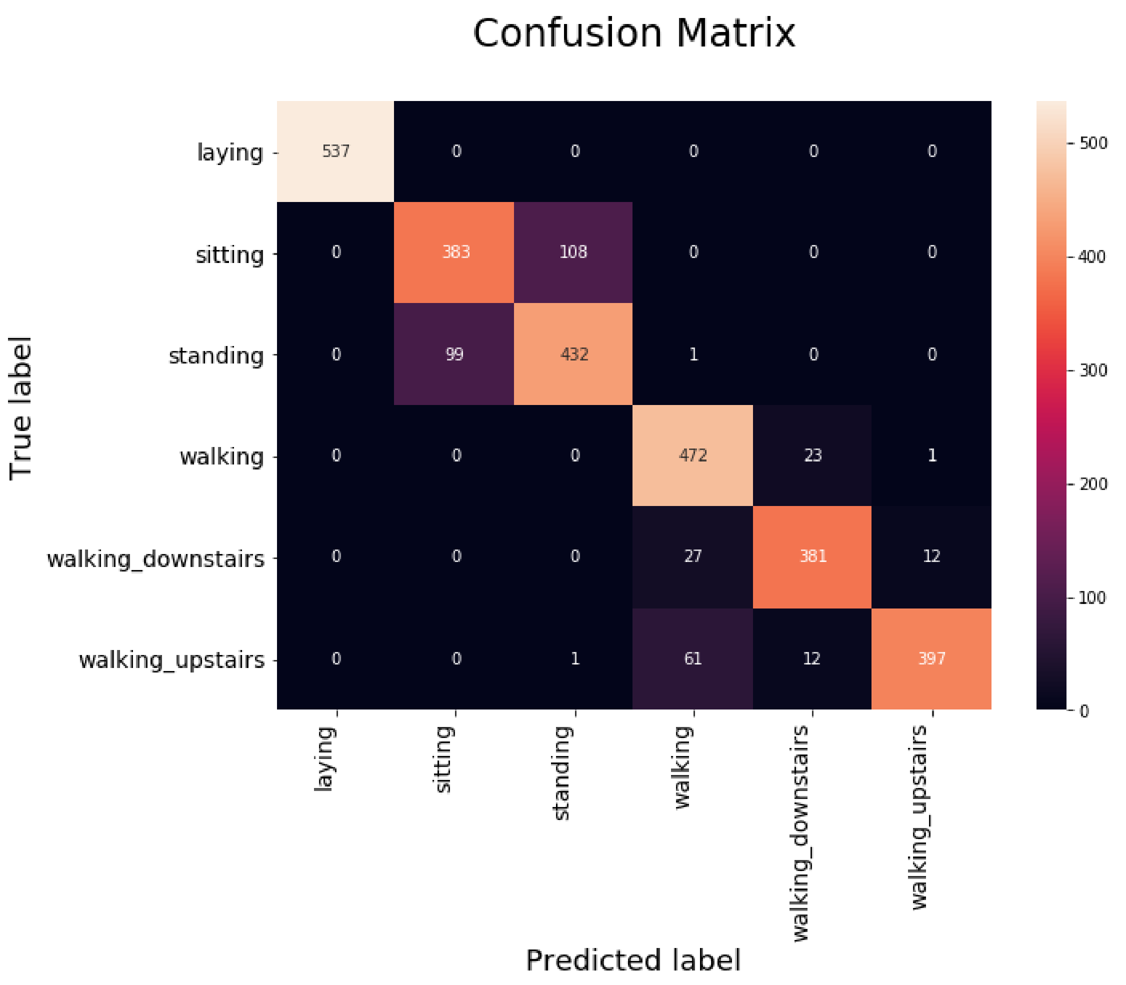
```
0.5143 - val_acc: 0.7418
Epoch 12/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.4279 - acc: 0.8062 - val_loss:
0.4951 - val_acc: 0.7472
Epoch 13/30
7352/7352 [==============================] - 38s 5ms/step - loss: 0.3911 - acc: 0.8130 - val_loss:
0.5606 - val_acc: 0.7516
Epoch 14/30
7352/7352 [==============================] - 37s 5ms/step - loss: 0.3898 - acc: 0.8313 - val_loss:
0.4518 - val_acc: 0.8137
Epoch 15/30
7352/7352 [==============================] - 37s 5ms/step - loss: 0.3308 - acc: 0.8942 - val_loss:
0.4732 - val_acc: 0.8633
Epoch 16/30
7352/7352 [==============================] - 37s 5ms/step - loss: 0.2891 - acc: 0.9176 - val_loss:
0.3794 - val_acc: 0.8765
Epoch 17/30
7352/7352 [==============================] - 38s 5ms/step - loss: 0.2660 - acc: 0.9246 - val_loss:
0.5082 - val_acc: 0.8660
Epoch 18/30
7352/7352 [==============================] - 38s 5ms/step - loss: 0.2538 - acc: 0.9251 - val_loss:
0.4772 - val_acc: 0.8806
Epoch 19/30
7352/7352 [==============================] - 38s 5ms/step - loss: 0.2502 - acc: 0.9312 - val_loss:
0.7013 - val_acc: 0.8307
Epoch 20/30
7352/7352 [==============================] - 46s 6ms/step - loss: 0.1980 - acc: 0.9382 - val_loss:
0.3988 - val_acc: 0.8890
Epoch 21/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.2018 - acc: 0.9372 - val_loss:
1.7682 - val_acc: 0.7075
Epoch 22/30
7352/7352 [==============================] - 39s 5ms/step - loss: 0.2455 - acc: 0.9310 - val_loss:
0.5812 - val_acc: 0.8687
Epoch 23/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.2194 - acc: 0.9329 - val_loss:
0.6468 - val_acc: 0.8744
Epoch 24/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.2282 - acc: 0.9304 - val_loss:
0.4721 - val_acc: 0.8741
Epoch 25/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.2166 - acc: 0.9359 - val_loss:
0.4131 - val_acc: 0.8938
Epoch 26/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.2173 - acc: 0.9350 - val_loss:
0.4841 - val_acc: 0.8887
Epoch 27/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.2224 - acc: 0.9353 - val_loss:
0.3590 - val_acc: 0.8935
Epoch 28/30
7352/7352 [==============================] - 39s 5ms/step - loss: 0.1961 - acc: 0.9385 - val_loss:
0.5297 - val_acc: 0.8802
Epoch 29/30
7352/7352 [==============================] - 39s 5ms/step - loss: 0.1876 - acc: 0.9416 - val_loss:
0.4324 - val_acc: 0.8924
Epoch 30/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1999 - acc: 0.9411 - val_loss:
0.4883 - val_acc: 0.8829
```

In [18]:

```
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=0)
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=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', 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: 0.488270
Test Accuracy: 88.293180%
```




- We have acheived 88% accuracy by just using two layer achitecture
- By just tuning the hyperparameter we can easily improve the performance

## (2) Single LSTM layer with 48-LSTM Units and optimizer as Adam optimizer

In [19]:

```
model_1 = Sequential()
model_1.add(LSTM(48, input_shape=(timesteps, input_dim)))
#dropout layer
model_1.add(Dropout(0.5))
#Activation function - output sigmoid
model_1.add(Dense(n_classes, activation='sigmoid'))
print(model_1.summary())

# Compiling the model
```

```
model_1.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])

model_1_accuracy = model_1.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test, Y_tes
t),epochs=epochs)
```

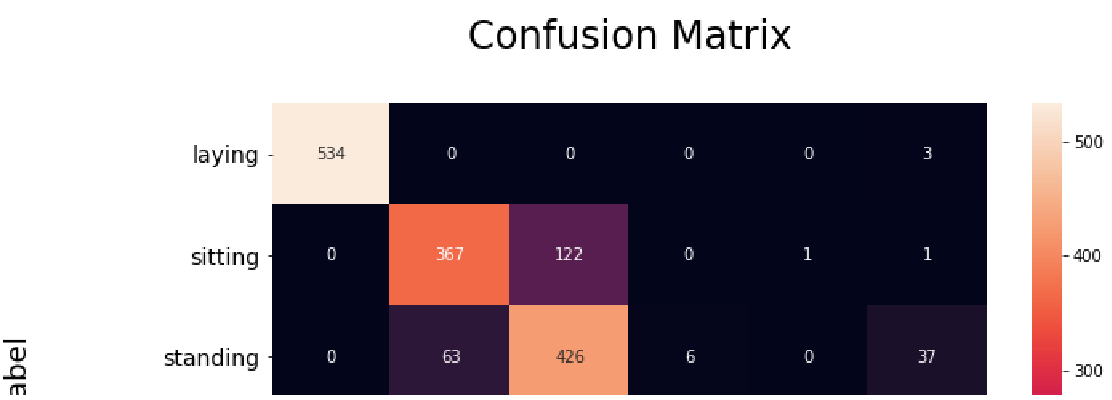
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_2 (LSTM)                (None, 48)                11136     
_________________________________________________________________
dropout_2 (Dropout)          (None, 48)                0         
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 294       
=================================================================
Total params: 11,430
Trainable params: 11,430
Non-trainable params: 0
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 45s 6ms/step - loss: 1.4210 - acc: 0.3677 - val_loss:
1.4543 - val_acc: 0.3424
Epoch 2/30
7352/7352 [==============================] - 43s 6ms/step - loss: 1.3615 - acc: 0.3659 - val_loss:
1.3897 - val_acc: 0.3502
Epoch 3/30
7352/7352 [==============================] - 43s 6ms/step - loss: 1.2965 - acc: 0.4147 - val_loss:
1.4920 - val_acc: 0.2389 acc
Epoch 4/30
7352/7352 [==============================] - 43s 6ms/step - loss: 1.2413 - acc: 0.4645 - val_loss:
1.2349 - val_acc: 0.4578
Epoch 5/30
7352/7352 [==============================] - 43s 6ms/step - loss: 1.1199 - acc: 0.5102 - val_loss:
0.9365 - val_acc: 0.6257
Epoch 6/30
7352/7352 [==============================] - 42s 6ms/step - loss: 1.0028 - acc: 0.5439 - val_loss:
1.0835 - val_acc: 0.5168
Epoch 7/30
7352/7352 [==============================] - 43s 6ms/step - loss: 1.0453 - acc: 0.5098 - val_loss:
1.0800 - val_acc: 0.4825
Epoch 8/30
7352/7352 [==============================] - 43s 6ms/step - loss: 1.1810 - acc: 0.4523 - val_loss:
1.2367 - val_acc: 0.4360
Epoch 9/30
7352/7352 [==============================] - 43s 6ms/step - loss: 1.2428 - acc: 0.4329 - val_loss:
1.0155 - val_acc: 0.5711
Epoch 10/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.9496 - acc: 0.5747 - val_loss:
1.2343 - val_acc: 0.4561
Epoch 11/30
7352/7352 [==============================] - 43s 6ms/step - loss: 1.0623 - acc: 0.5399 - val_loss:
0.9889 - val_acc: 0.5857
Epoch 12/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.8686 - acc: 0.6114 - val_loss:
0.9577 - val_acc: 0.5711
Epoch 13/30
7352/7352 [==============================] - 42s 6ms/step - loss: 1.0787 - acc: 0.4974 - val_loss:
1.1982 - val_acc: 0.5484
Epoch 14/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.9513 - acc: 0.5822 - val_loss:
0.9239 - val_acc: 0.6088
Epoch 15/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.8773 - acc: 0.5929 - val_loss:
0.8365 - val_acc: 0.5864
Epoch 16/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.7541 - acc: 0.6250 - val_loss:
0.7998 - val_acc: 0.6077
Epoch 17/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.7139 - acc: 0.6499 - val_loss:
0.7898 - val_acc: 0.6098
Epoch 18/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.7097 - acc: 0.6468 - val_loss:
0.7610 - val_acc: 0.6155
Epoch 19/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.6794 - acc: 0.6575 - val_loss:
0.7823 - val_acc: 0.6084
```

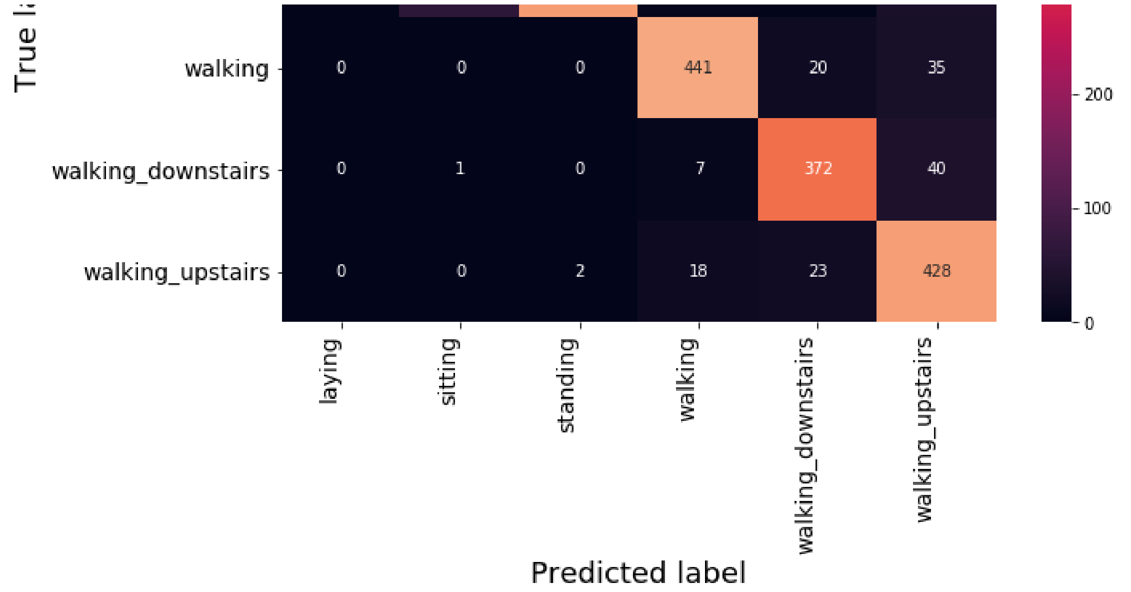
```
0.7822 - val_acc: 0.6084
Epoch 20/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.6810 - acc: 0.6553 - val_loss:
0.7495 - val_acc: 0.6179
Epoch 21/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.6905 - acc: 0.6468 - val_loss:
0.7460 - val_acc: 0.6247
Epoch 22/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.6648 - acc: 0.6712 - val_loss:
0.7577 - val_acc: 0.6563
Epoch 23/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.6891 - acc: 0.6727 - val_loss:
0.8226 - val_acc: 0.5843
Epoch 24/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.6327 - acc: 0.7331 - val_loss:
0.6253 - val_acc: 0.7706
Epoch 25/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.5341 - acc: 0.8074 - val_loss:
0.5621 - val_acc: 0.7771
Epoch 26/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.3767 - acc: 0.8696 - val_loss:
0.4193 - val_acc: 0.8548
Epoch 27/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.3015 - acc: 0.9015 - val_loss:
0.3831 - val_acc: 0.8795
Epoch 28/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.3263 - acc: 0.8989 - val_loss:
0.4398 - val_acc: 0.8368
Epoch 29/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.3337 - acc: 0.8976 - val_loss:
0.3343 - val_acc: 0.8809
Epoch 30/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.2294 - acc: 0.9272 - val_loss:
0.3442 - val_acc: 0.8714
```

In [20]:

```
scores1 = model1.evaluate(X_test, Y_test, verbose=0)
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)])

class_names = ['laying','sitting','standing','walking','walking_downstairs','walking_upstairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_names, columns=class
_names )
fig = plt.figure(figsize=(10,7))
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")

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: 0.344224
Test Accuracy: 87.139464%
```






# (3) Single LSTM layer with 48-LSTM Units with optimizer as RMSPROP

In [21]:

```
#sequential model
model_2 = Sequential()
#parameters
model_2.add(LSTM(48, input_shape=(timesteps, input_dim)))
#dropout layer
model_2.add(Dropout(0.5))
model_2.add(Dense(n_classes, activation='sigmoid'))
print(model_2.summary())

model_2.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])

model_2_accuracy = model_2.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test, Y_tes
t),epochs=epochs)
```

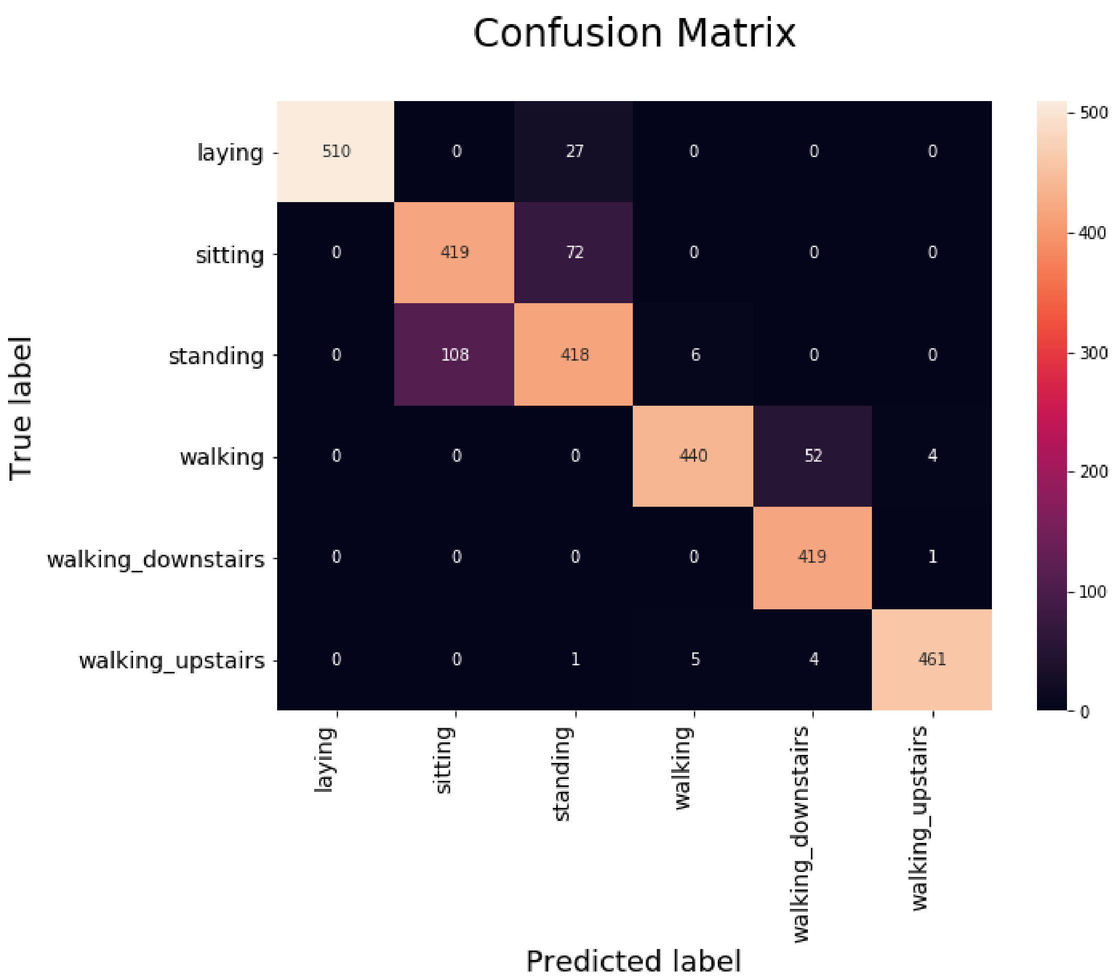
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_3 (LSTM)                (None, 48)                11136     
_________________________________________________________________
dropout_3 (Dropout)          (None, 48)                0         
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 294       
=================================================================
Total params: 11,430
Trainable params: 11,430
Non-trainable params: 0
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 43s 6ms/step - loss: 1.2313 - acc: 0.4780 - val_loss:
1.0087 - val_acc: 0.5674
Epoch 2/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.8782 - acc: 0.6073 - val_loss:
0.8074 - val_acc: 0.6498
Epoch 3/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.7840 - acc: 0.6542 - val_loss:
0.9888 - val_acc: 0.5938
Epoch 4/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.6928 - acc: 0.6900 - val_loss:
0.7771 - val_acc: 0.6790
Epoch 5/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.6225 - acc: 0.7348 - val_loss:
0.7603 - val_acc: 0.7553
Epoch 6/30
```

```
7352/7352 [==============================] - 43s 6ms/step - loss: 0.5056 - acc: 0.8290 - val_loss:
0.6061 - val_acc: 0.8185
Epoch 7/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.3532 - acc: 0.8900 - val_loss:
0.4807 - val_acc: 0.8595
Epoch 8/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.2994 - acc: 0.9113 - val_loss:
0.6068 - val_acc: 0.8602
Epoch 9/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.2638 - acc: 0.9212 - val_loss:
0.4591 - val_acc: 0.8761
Epoch 10/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.2297 - acc: 0.9276 - val_loss:
0.5101 - val_acc: 0.8856
Epoch 11/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.2190 - acc: 0.9336 - val_loss:
0.4846 - val_acc: 0.8795
Epoch 12/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.2153 - acc: 0.9329 - val_loss:
0.5808 - val_acc: 0.8839
Epoch 13/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.2055 - acc: 0.9376 - val_loss:
0.3690 - val_acc: 0.8826
Epoch 14/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1898 - acc: 0.9366 - val_loss:
0.4428 - val_acc: 0.8935
Epoch 15/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.2032 - acc: 0.9319 - val_loss:
0.4052 - val_acc: 0.8975
Epoch 16/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1801 - acc: 0.9416 - val_loss:
0.5809 - val_acc: 0.8829
Epoch 17/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1810 - acc: 0.9423 - val_loss:
0.4727 - val_acc: 0.9030
Epoch 18/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1714 - acc: 0.9452 - val_loss:
0.3016 - val_acc: 0.9077
Epoch 19/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1654 - acc: 0.9411 - val_loss:
0.3503 - val_acc: 0.9040
Epoch 20/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1795 - acc: 0.9455 - val_loss:
0.3498 - val_acc: 0.9192
Epoch 21/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1676 - acc: 0.9404 - val_loss:
0.3858 - val_acc: 0.9067
Epoch 22/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1811 - acc: 0.9423 - val_loss:
0.3532 - val_acc: 0.9125
Epoch 23/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1563 - acc: 0.9449 - val_loss:
0.4389 - val_acc: 0.8975
Epoch 24/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1495 - acc: 0.9449 - val_loss:
0.4716 - val_acc: 0.9043
Epoch 25/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1740 - acc: 0.9436 - val_loss:
0.4915 - val_acc: 0.9053
Epoch 26/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1564 - acc: 0.9446 - val_loss:
0.4718 - val_acc: 0.8941
Epoch 27/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1648 - acc: 0.9475 - val_loss:
0.4253 - val_acc: 0.8975
Epoch 28/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1504 - acc: 0.9438 - val_loss:
0.4370 - val_acc: 0.9013
Epoch 29/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1501 - acc: 0.9468 - val_loss:
0.5412 - val_acc: 0.8867
Epoch 30/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1647 - acc: 0.9471 - val_loss:
0.4105 - val_acc: 0.9050
```

In [22]:

```
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)])

class_names = ['laying','sitting','standing','walking','walking_downstairs','walking_upstairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_names, columns=class
_names )
fig = plt.figure(figsize=(10,7))
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")

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: 0.410484
Test Accuracy: 90.498812%
```




# (4) Single LSTM layer with 64-LSTM Units and the same optimiser RMSPROP

In [23]:

```
# sequential model
model_3 = Sequential()
# parameters
model_3.add(LSTM(64, input_shape=(timesteps, input_dim)))
# ropout layer
model_3.add(Dropout(0.5))
```

```
model_3.add(Dense(n_classes, activation='sigmoid'))
print(model_3.summary())

model_3.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])

model_3_accuracy = model_3.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test, Y_tes
t),epochs=epochs)
```

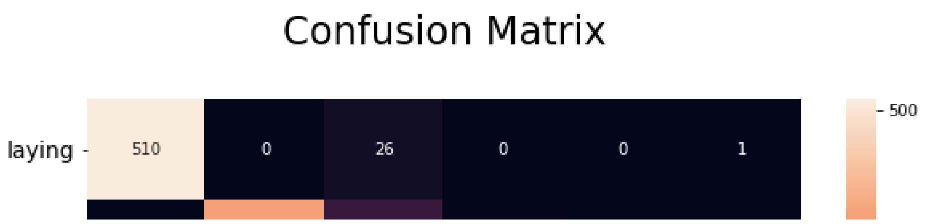
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_4 (LSTM)                (None, 64)                18944     
_________________________________________________________________
dropout_4 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_4 (Dense)              (None, 6)                 390       
=================================================================
Total params: 19,334
Trainable params: 19,334
Non-trainable params: 0
_________________________________________________________________
None
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 47s 6ms/step - loss: 1.2746 - acc: 0.4457 - val_loss:
1.1393 - val_acc: 0.5246
Epoch 2/30
7352/7352 [==============================] - 46s 6ms/step - loss: 0.9587 - acc: 0.6020 - val_loss:
0.8720 - val_acc: 0.6349
Epoch 3/30
7352/7352 [==============================] - 46s 6ms/step - loss: 1.0225 - acc: 0.5890 - val_loss:
0.9470 - val_acc: 0.6176
Epoch 4/30
7352/7352 [==============================] - 46s 6ms/step - loss: 0.7561 - acc: 0.6812 - val_loss:
0.7023 - val_acc: 0.7021
Epoch 5/30
7352/7352 [==============================] - 46s 6ms/step - loss: 0.6203 - acc: 0.7402 - val_loss:
0.6757 - val_acc: 0.7218
Epoch 6/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.4874 - acc: 0.8249 - val_loss:
0.5334 - val_acc: 0.8358
Epoch 7/30
7352/7352 [==============================] - 48s 7ms/step - loss: 0.3588 - acc: 0.8905 - val_loss:
0.4300 - val_acc: 0.8660
Epoch 8/30
7352/7352 [==============================] - 52s 7ms/step - loss: 0.2826 - acc: 0.9042 - val_loss:
1.0640 - val_acc: 0.7852
Epoch 9/30
7352/7352 [==============================] - 49s 7ms/step - loss: 0.2855 - acc: 0.9033 - val_loss:
0.4491 - val_acc: 0.8490
Epoch 10/30
7352/7352 [==============================] - 48s 7ms/step - loss: 0.2367 - acc: 0.9197 - val_loss:
0.4427 - val_acc: 0.8826
Epoch 11/30
7352/7352 [==============================] - 48s 6ms/step - loss: 0.2891 - acc: 0.9064 - val_loss:
0.3384 - val_acc: 0.8968
Epoch 12/30
7352/7352 [==============================] - 48s 7ms/step - loss: 0.2101 - acc: 0.9327 - val_loss:
0.2863 - val_acc: 0.9067
Epoch 13/30
7352/7352 [==============================] - 49s 7ms/step - loss: 0.1883 - acc: 0.9309 - val_loss:
0.3804 - val_acc: 0.8806
Epoch 14/30
7352/7352 [==============================] - 48s 6ms/step - loss: 0.1781 - acc: 0.9354 - val_loss:
0.4222 - val_acc: 0.8778
Epoch 15/30
7352/7352 [==============================] - 47s 6ms/step - loss: 0.1812 - acc: 0.9344 - val_loss:
0.3767 - val_acc: 0.8887
Epoch 16/30
7352/7352 [==============================] - 48s 6ms/step - loss: 0.1701 - acc: 0.9414 - val_loss:
0.2908 - val_acc: 0.9053 2s - loss:
Epoch 17/30
7352/7352 [==============================] - 48s 6ms/step - loss: 0.1603 - acc: 0.9446 - val_loss:
0.3604 - val_acc: 0.8968
Epoch 18/30
7352/7352 [==============================] - 48s 7ms/step - loss: 0.1494 - acc: 0.9460 - val_loss:
```

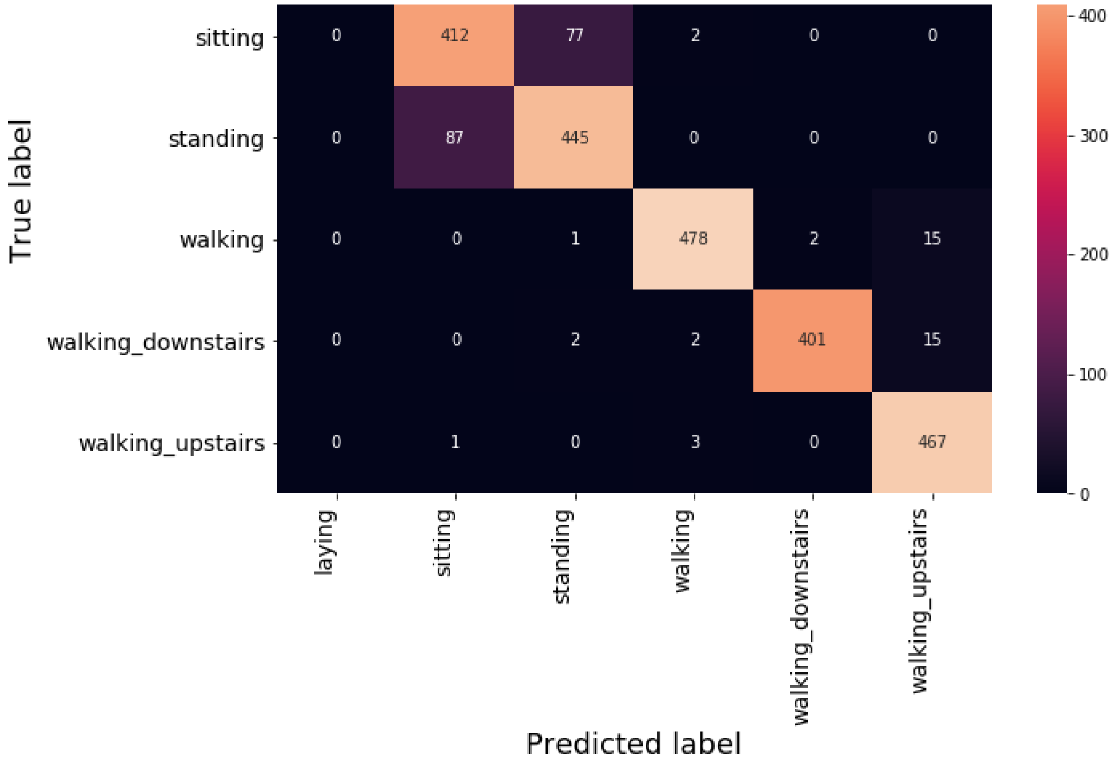
```
0.3924 - val_acc: 0.9030
Epoch 19/30
7352/7352 [==============================] - 47s 6ms/step - loss: 0.1555 - acc: 0.9445 - val_loss:
0.2972 - val_acc: 0.9125
Epoch 20/30
7352/7352 [==============================] - 47s 6ms/step - loss: 0.1413 - acc: 0.9498 - val_loss:
0.3077 - val_acc: 0.9216
Epoch 21/30
7352/7352 [==============================] - 47s 6ms/step - loss: 0.1674 - acc: 0.9444 - val_loss:
0.2407 - val_acc: 0.9141
Epoch 22/30
7352/7352 [==============================] - 47s 6ms/step - loss: 0.1550 - acc: 0.9430 - val_loss:
0.3160 - val_acc: 0.9104
Epoch 23/30
7352/7352 [==============================] - 47s 6ms/step - loss: 0.1551 - acc: 0.9450 - val_loss:
0.2295 - val_acc: 0.9287
Epoch 24/30
7352/7352 [==============================] - 46s 6ms/step - loss: 0.1679 - acc: 0.9440 - val_loss:
0.7719 - val_acc: 0.8755
Epoch 25/30
7352/7352 [==============================] - 47s 6ms/step - loss: 0.1543 - acc: 0.9472 - val_loss:
0.2647 - val_acc: 0.9192
Epoch 26/30
7352/7352 [==============================] - 47s 6ms/step - loss: 0.1457 - acc: 0.9459 - val_loss:
0.2418 - val_acc: 0.9108
Epoch 27/30
7352/7352 [==============================] - 47s 6ms/step - loss: 0.1383 - acc: 0.9476 - val_loss:
0.3972 - val_acc: 0.9036
Epoch 28/30
7352/7352 [==============================] - 46s 6ms/step - loss: 0.1412 - acc: 0.9508 - val_loss:
0.3194 - val_acc: 0.9199
Epoch 29/30
7352/7352 [==============================] - 47s 6ms/step - loss: 0.1496 - acc: 0.9464 - val_loss:
0.3358 - val_acc: 0.9145
Epoch 30/30
7352/7352 [==============================] - 47s 6ms/step - loss: 0.1439 - acc: 0.9490 - val_loss:
0.2993 - val_acc: 0.9206
```

In [24]:

```python
# 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=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', 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: 0.299268
Test Accuracy: 92.059722%
```






## (5) dual LSTM layer with 32-LSTM Units and RSPROP optimizer

In [25]:

```
model_4 = Sequential()
model_4.add(LSTM(32,return_sequences=True, input_shape=(timesteps, input_dim)))
model_4.add(Dropout(0.5))

model_4.add(LSTM(32))
model_4.add(Dropout(0.5))
model_4.add(Dense(n_classes, activation='sigmoid'))
print(model_4.summary())

model_4.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])

model_4_accuracy = model_4.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test, Y_tes
t),epochs=epochs)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_5 (LSTM)                (None, 128, 32)           5376
_________________________________________________________________
dropout_5 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
lstm_6 (LSTM)                (None, 32)                8320
_________________________________________________________________
dropout_6 (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 1/30
7352/7352 [==============================] - 86s 12ms/step - loss: 1.2107 - acc: 0.5061 - val_loss
: 0.8905 - val_acc: 0.6390
Epoch 2/30
7352/7352 [==============================] - 84s 11ms/step - loss: 0.7991 - acc: 0.6766 - val_loss
: 0.6888 - val_acc: 0.7167
```

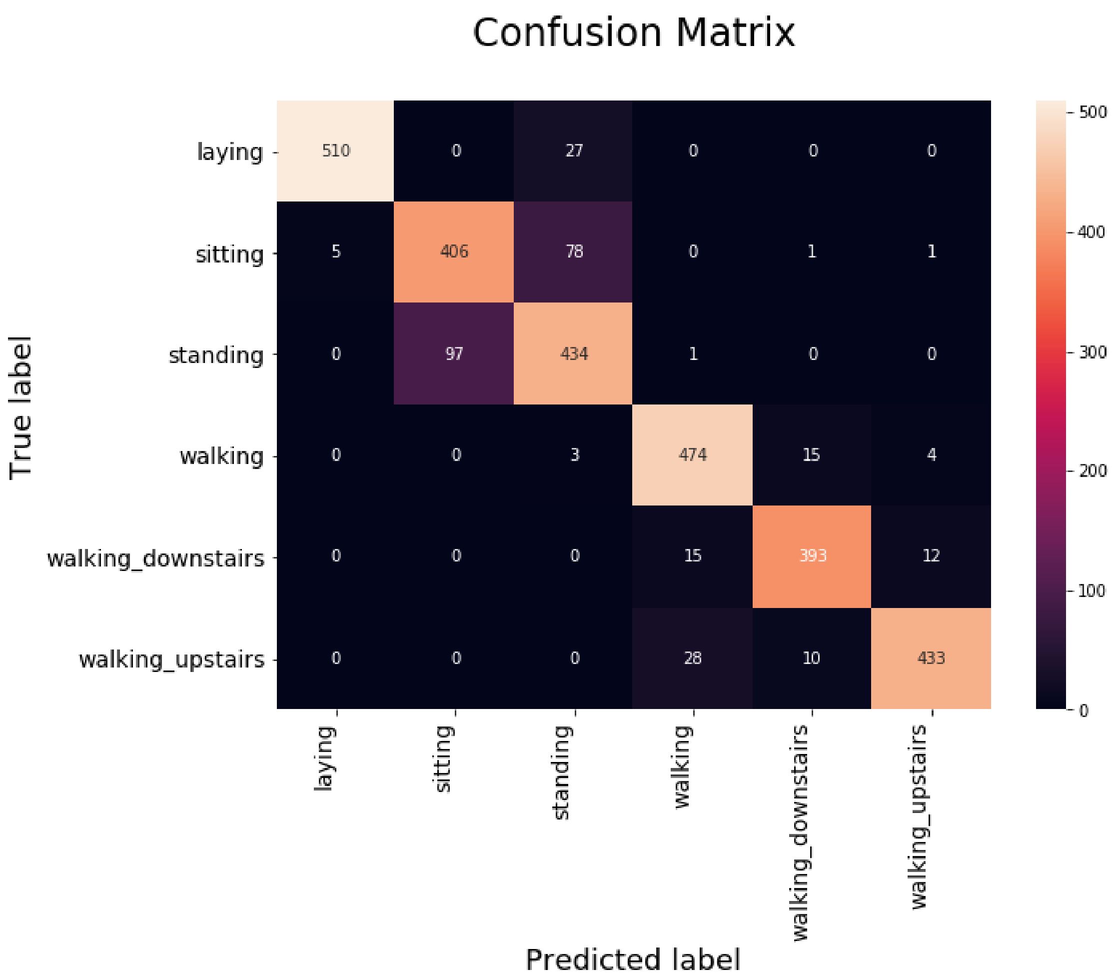
```
Epoch 3/30
7352/7352 [==============================] - 83s 11ms/step - loss: 0.6209 - acc: 0.7542 - val_loss
: 0.6014 - val_acc: 0.7275
Epoch 4/30
7352/7352 [==============================] - 83s 11ms/step - loss: 0.4968 - acc: 0.7802 - val_loss
: 0.7280 - val_acc: 0.7119
Epoch 5/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.4323 - acc: 0.8009 - val_loss
: 1.0594 - val_acc: 0.6841
Epoch 6/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.4538 - acc: 0.8247 - val_loss
: 0.5700 - val_acc: 0.8280
Epoch 7/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.3524 - acc: 0.8785 - val_loss
: 0.5126 - val_acc: 0.8626
Epoch 8/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.3199 - acc: 0.9115 - val_loss
: 0.5495 - val_acc: 0.8717
Epoch 9/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.2609 - acc: 0.9285 - val_loss
: 0.4831 - val_acc: 0.8772
Epoch 10/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.2290 - acc: 0.9339 - val_loss
: 0.4644 - val_acc: 0.8768
Epoch 11/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.2160 - acc: 0.9353 - val_loss
: 0.4657 - val_acc: 0.8931
Epoch 12/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.2236 - acc: 0.9321 - val_loss
: 0.6295 - val_acc: 0.8724
Epoch 13/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1718 - acc: 0.9455 - val_loss
: 0.6721 - val_acc: 0.8602
Epoch 14/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1740 - acc: 0.9359 - val_loss
: 0.4371 - val_acc: 0.9002
Epoch 15/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1693 - acc: 0.9423 - val_loss
: 0.4365 - val_acc: 0.8938
Epoch 16/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1813 - acc: 0.9459 - val_loss
: 0.4093 - val_acc: 0.9006
Epoch 17/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1687 - acc: 0.9472 - val_loss
: 0.5450 - val_acc: 0.8931
Epoch 18/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1557 - acc: 0.9474 - val_loss
: 0.4184 - val_acc: 0.8951
Epoch 19/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1479 - acc: 0.9471 - val_loss
: 0.6616 - val_acc: 0.8826
Epoch 20/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1479 - acc: 0.9493 - val_loss
: 0.3981 - val_acc: 0.9019
Epoch 21/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1465 - acc: 0.9509 - val_loss
: 0.4721 - val_acc: 0.9033
Epoch 22/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1508 - acc: 0.9508 - val_loss
: 0.5981 - val_acc: 0.8816
Epoch 23/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1512 - acc: 0.9489 - val_loss
: 0.5368 - val_acc: 0.8955
Epoch 24/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1434 - acc: 0.9513 - val_loss
: 0.5763 - val_acc: 0.8897
Epoch 25/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1805 - acc: 0.9414 - val_loss
: 0.5735 - val_acc: 0.8914
Epoch 26/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1453 - acc: 0.9528 - val_loss
: 0.4694 - val_acc: 0.9016
Epoch 27/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1385 - acc: 0.9520 - val_loss
: 0.5944 - val_acc: 0.8999
Epoch 28/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1420 - acc: 0.9533 - val_loss
```

```
: 0.8538 - val_acc: 0.8738
Epoch 29/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1288 - acc: 0.9547 - val_loss
: 0.6149 - val_acc: 0.8860
Epoch 30/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1291 - acc: 0.9532 - val_loss
: 0.5455 - val_acc: 0.8992
```

In [26]:

```
scores4 = model4.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores4[0]))
print("Test Accuracy: %f%%" % (scores4[1]*100))

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)])

class_names = ['laying','sitting','standing','walking','walking_downstairs','walking_upstairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_names, columns=class
_names )
fig = plt.figure(figsize=(10,7))
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")

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: 0.545492
Test Accuracy: 89.921955%
```




## (6) dual LSTM layer with 64-LSTM Units and RSPROP optimizer

optimizer

In [27]:

```
model_5 = Sequential()
model_5.add(LSTM(64,return_sequences=True, input_shape=(timesteps, input_dim)))
model_5.add(Dropout(0.7))

model_5.add(LSTM(64))
model_5.add(Dropout(0.7))
model_5.add(Dense(n_classes, activation='sigmoid'))
print(model_5.summary())

# Compiling the model
model_5.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])

model_5_accuracy = model_5.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test, Y_tes
t),epochs=epochs)
```

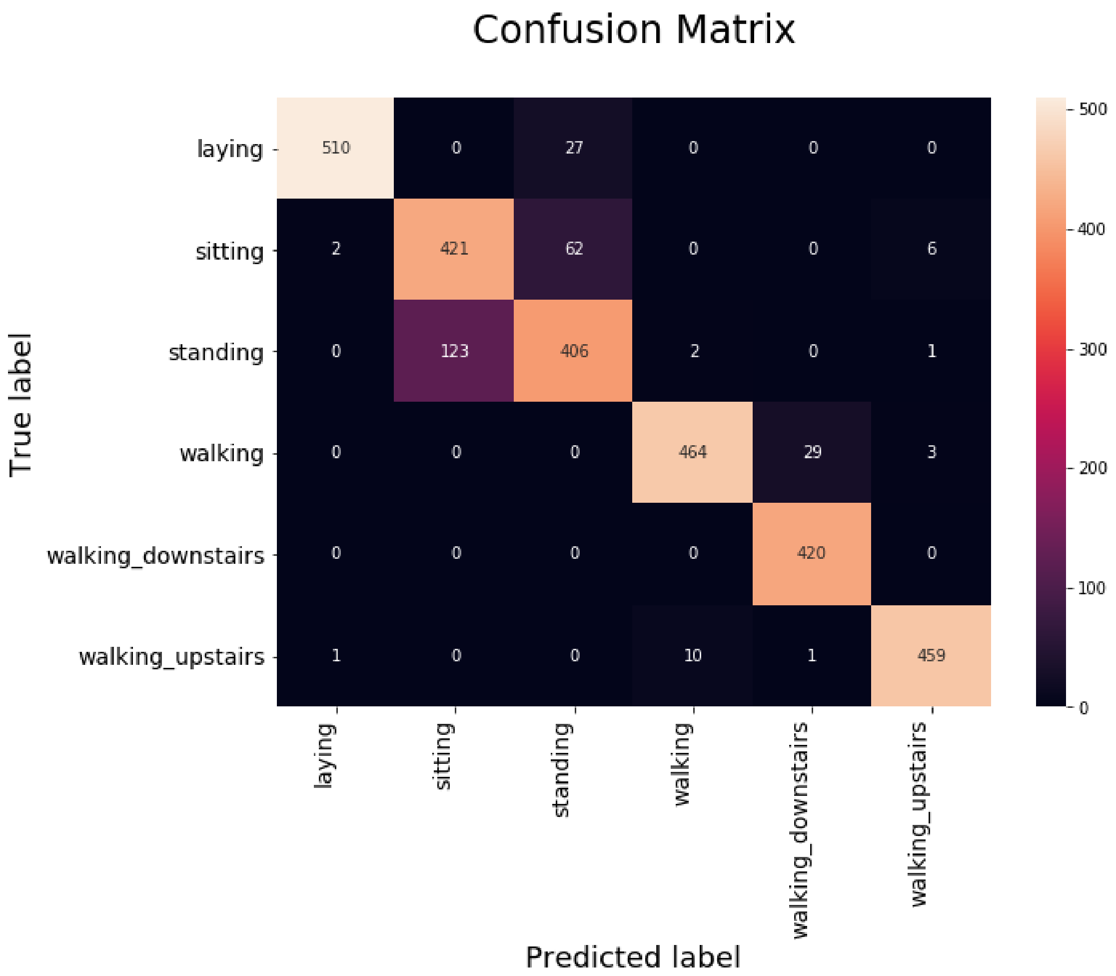
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_7 (LSTM)                (None, 128, 64)           18944
_________________________________________________________________
dropout_7 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
lstm_8 (LSTM)                (None, 64)                33024
_________________________________________________________________
dropout_8 (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 [==============================] - 110s 15ms/step - loss: 1.1611 - acc: 0.4890 - val_los
s: 0.8595 - val_acc: 0.6461
Epoch 2/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.8021 - acc: 0.6549 - val_los
s: 0.7768 - val_acc: 0.6383
Epoch 3/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.7392 - acc: 0.6670 - val_los
s: 0.7168 - val_acc: 0.7048
Epoch 4/30
7352/7352 [==============================] - 109s 15ms/step - loss: 0.6489 - acc: 0.7338 - val_los
s: 0.6764 - val_acc: 0.7421
Epoch 5/30
7352/7352 [==============================] - 107s 15ms/step - loss: 0.5545 - acc: 0.7625 - val_los
s: 0.6935 - val_acc: 0.7258
Epoch 6/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.4380 - acc: 0.8164 - val_los
s: 0.5711 - val_acc: 0.8436
Epoch 7/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.3323 - acc: 0.8966 - val_los
s: 0.4062 - val_acc: 0.8823
Epoch 8/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.2380 - acc: 0.9301 - val_los
s: 0.3921 - val_acc: 0.8907
Epoch 9/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.2048 - acc: 0.9370 - val_los
s: 0.3001 - val_acc: 0.9067
Epoch 10/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1991 - acc: 0.9389 - val_los
s: 0.3917 - val_acc: 0.8982
Epoch 11/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1775 - acc: 0.9429 - val_los
s: 0.4253 - val_acc: 0.9074
Epoch 12/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1724 - acc: 0.9436 - val_los
s: 0.4833 - val_acc: 0.9101
Epoch 13/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1628 - acc: 0.9453 - val_los
s: 0.4220 - val_acc: 0.9030
```

```
Epoch 14/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1754 - acc: 0.9440 - val_los
s: 0.5106 - val_acc: 0.8907
Epoch 15/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1556 - acc: 0.9465 - val_los
s: 0.7246 - val_acc: 0.8870
Epoch 16/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1733 - acc: 0.9452 - val_los
s: 0.3389 - val_acc: 0.9267
Epoch 17/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1737 - acc: 0.9448 - val_los
s: 0.3460 - val_acc: 0.9141
Epoch 18/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1698 - acc: 0.9442 - val_los
s: 0.5056 - val_acc: 0.9063
Epoch 19/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1469 - acc: 0.9512 - val_los
s: 0.4818 - val_acc: 0.8924
Epoch 20/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1492 - acc: 0.9461 - val_los
s: 0.4129 - val_acc: 0.9131
Epoch 21/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1532 - acc: 0.9490 - val_los
s: 0.4454 - val_acc: 0.9138
Epoch 22/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1758 - acc: 0.9436 - val_los
s: 0.4868 - val_acc: 0.9111
Epoch 23/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1528 - acc: 0.9489 - val_los
s: 0.4740 - val_acc: 0.9114
Epoch 24/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1589 - acc: 0.9438 - val_los
s: 0.4851 - val_acc: 0.9165
Epoch 25/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1522 - acc: 0.9437 - val_los
s: 0.5046 - val_acc: 0.9050
Epoch 26/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1363 - acc: 0.9495 - val_los
s: 0.5402 - val_acc: 0.9046
Epoch 27/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1480 - acc: 0.9459 - val_los
s: 0.4211 - val_acc: 0.9060
Epoch 28/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1347 - acc: 0.9495 - val_los
s: 0.5436 - val_acc: 0.8962
Epoch 29/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1489 - acc: 0.9474 - val_los
s: 0.4395 - val_acc: 0.9043
Epoch 30/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1491 - acc: 0.9486 - val_los
s: 0.4127 - val_acc: 0.9094
```

In [28]:

```
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)])

class_names = ['laying','sitting','standing','walking','walking_downstairs','walking_upstairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_names, columns=class
_names )
fig = plt.figure(figsize=(10,7))
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")

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: 0.412691
Test Accuracy: 90.939939%
```




# CONCLUSION

## (a). Procedure Followed :

STEP 1 :- Here training and testing dataset is created

STEP 2:- Various Architectures of LSTM

STEP 3:- For every model accuracy and test score

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

## (b). Table :

In [29]:

```
# PrettyTable library
from prettytable import PrettyTable

# models
names =['1 LSTM layer with 32 LSTM Units(Optimizer-->rmsprop)','1 LSTM layer with 48 LSTM Units(Optimizer-->adam)',\
        '1 LSTM layer with 48 LSTM Units(Optimizer-->rmsprop)','1 LSTM layer with 64 LSTM Units(Optimizer-->rmsprop)',\
        '2 LSTM layer with 32 LSTM Units(Optimizer-->rmsprop)','2 LSTM layer with 64 LSTM Units(Optimizer-->rmsprop)']
```

```
# accuracies of training
train_acc = [model_accuracy.model_accuracy['acc'][29],model_1_accuracy.model_accuracy['acc'][29],mo
del_2_accuracy.model_accuracy['acc'][29],\
             model_3_accuracy.model_accuracy['acc'][29],model_4_accuracy.model_accuracy['acc'][29],
model_5_accuracy.model_accuracy['acc'][29]]

# accuracies of test
test_acc =[scores[1],scores1[1],scores2[1],scores3[1],scores4[1],scores5[1]]

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

ptable = PrettyTable()

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)

print(ptable)
```

```
+--------+-------------------------------------------------------+---------------------+--------------
--+
| S.NO. |                        MODEL                          | Training Accuracy   |   Test
Accuracy    |
+--------+-------------------------------------------------------+---------------------+--------------
--+
|   1    | 1 LSTM layer with 32 LSTM Units(Optimizer-->rmsprop) | 0.9411044613710555 |
0.8829317950458093 |
|   2    |  1 LSTM layer with 48 LSTM Units(Optimizer-->adam)   | 0.9272306855277476 | 0.87139463861
55412 |
|   3    | 1 LSTM layer with 48 LSTM Units(Optimizer-->rmsprop) | 0.9470892274211099 |
0.9049881235154394 |
|   4    | 1 LSTM layer with 64 LSTM Units(Optimizer-->rmsprop) | 0.948993471164309  |
0.9205972175093315 |
|   5    | 2 LSTM layer with 32 LSTM Units(Optimizer-->rmsprop) | 0.9532100108813928 |
0.8992195453003053 |
|   6    | 2 LSTM layer with 64 LSTM Units(Optimizer-->rmsprop) | 0.9485854189336235 |
0.9093993892093655 |
+--------+-------------------------------------------------------+---------------------+--------------
--+
```

In [ ]: