Assignment-21: Human activity detection [M]

Objective:

- Various Models with different no. of hidden layer and optimizer with different dropout in LSTM

In [2]:
```
# Importing Libraries
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.metrics import confusion_matrix
# Importing libraries
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers.core import Dense, Dropout
from keras import backend as K
```

```
Using TensorFlow backend.
```

In [3]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import time
# https://gist.github.com/greydanus/f6eee59eaf1d90fcb3b534a25362cea4
# https://stackoverflow.com/a/14434334
# this function is used to update the plots for each epoch and error
def plt_dynamic(x, vy, ty):
```

```
    fig = plt.figure( facecolor='c', edgecolor='k')
    plt.plot(x, vy, 'b', label="Validation Loss")
    plt.plot(x, ty, 'r', label="Train Loss")
    plt.xlabel('Epochs')
    plt.ylabel('Categorical Crossentropy Loss')
    plt.legend()
    plt.grid()
    plt.show()
```

In [4]:

```
# 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',
}
```

Data

In [5]:

```
# Data directory
DATADIR = '/floyd/input/uci_har_dataset'
```

In [6]:

```
# 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 [7]:
```
# 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'/floyd/input/uci_har_dataset/{subset}/Inertial Sig
nals/{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 [8]:
```
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'/floyd/input/uci_har_dataset/{subset}/y_{subset}.txt'
```

```python
    y = _read_csv(filename)[0]

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

In [9]:
```python
def load_data():
    """
    Obtain the dataset from multiple files.
    Returns: X_train, X_test, y_train, y_test
    """
    X_train, X_test = load_signals('train'), load_signals('test')
    y_train, y_test = load_y('train'), load_y('test')

    return X_train, X_test, y_train, y_test
```

In [10]:
```python
# Importing tensorflow
np.random.seed(42)
import tensorflow as tf
tf.set_random_seed(42)
```

In [11]:
```python
# Configuring a session
session_conf = tf.ConfigProto(
    intra_op_parallelism_threads=1,
    inter_op_parallelism_threads=1
)
```

In [12]:
```python
# Import Keras

sess = tf.Session(graph=tf.get_default_graph(), config=session_conf)
K.set_session(sess)
```

In [13]:
```python
# Initializing parameters
epochs = 30
batch_size = 16
```

In [14]:
```python
# 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
```

- Defining the Architecture of LSTM

1) 32 LSTM + 1 layer LSTM + rmsprop optimizer

In [17]:
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(32, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 32)                5376
_________________________________________________________________
```

```
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 198
=================================================================
Total params: 5,574
Trainable params: 5,574
Non-trainable params: 0
_________________________________________________________________
```

In [18]:

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

In [19]:

```python
# Training the model
hist1=model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 24s 3ms/step - loss: 1.324
7 - acc: 0.4308 - val_loss: 1.1474 - val_acc: 0.4808
Epoch 2/30
7352/7352 [==============================] - 24s 3ms/step - loss: 1.099
5 - acc: 0.5196 - val_loss: 1.0965 - val_acc: 0.5236
Epoch 3/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.896
8 - acc: 0.6092 - val_loss: 0.8670 - val_acc: 0.6312
Epoch 4/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.736
6 - acc: 0.6532 - val_loss: 0.7453 - val_acc: 0.6125
Epoch 5/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.663
6 - acc: 0.6768 - val_loss: 0.9799 - val_acc: 0.5989
Epoch 6/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.619
```

```
7 - acc: 0.6989 - val_loss: 0.7784 - val_acc: 0.6498
Epoch 7/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.575
2 - acc: 0.7444 - val_loss: 0.8042 - val_acc: 0.6820
Epoch 8/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.547
9 - acc: 0.7636 - val_loss: 0.5897 - val_acc: 0.7438
Epoch 9/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.474
2 - acc: 0.7856 - val_loss: 0.6495 - val_acc: 0.7258
Epoch 10/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.441
9 - acc: 0.8069 - val_loss: 0.6326 - val_acc: 0.7788
Epoch 11/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.433
6 - acc: 0.8388 - val_loss: 0.5086 - val_acc: 0.8554
Epoch 12/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.346
1 - acc: 0.8980 - val_loss: 0.4601 - val_acc: 0.8605
Epoch 13/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.301
2 - acc: 0.9075 - val_loss: 0.4970 - val_acc: 0.8504
Epoch 14/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.262
0 - acc: 0.9210 - val_loss: 0.3690 - val_acc: 0.8884
Epoch 15/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.268
8 - acc: 0.9223 - val_loss: 0.4479 - val_acc: 0.8744
Epoch 16/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.239
1 - acc: 0.9238 - val_loss: 0.3821 - val_acc: 0.9002
Epoch 17/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.229
9 - acc: 0.9280 - val_loss: 0.4713 - val_acc: 0.8856
Epoch 18/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.218
1 - acc: 0.9372 - val_loss: 0.4589 - val_acc: 0.8989
Epoch 19/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.192
```

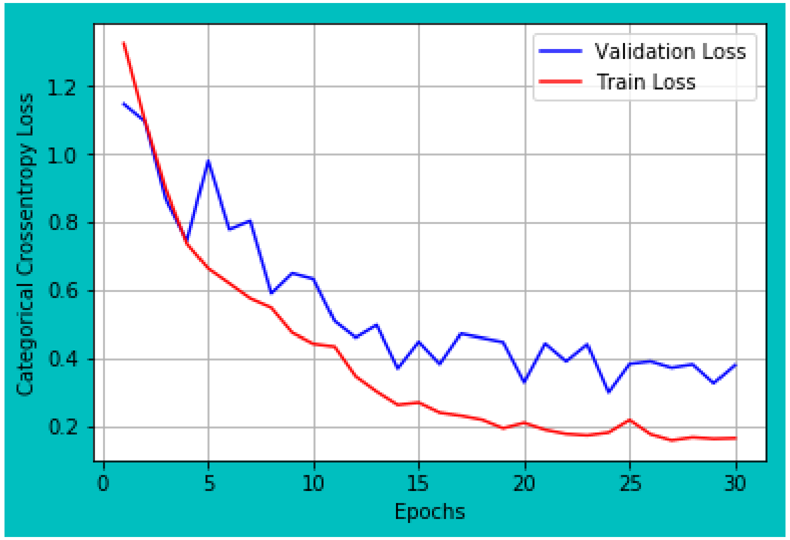
```
9 - acc: 0.9377 - val_loss: 0.4467 - val_acc: 0.8992
Epoch 20/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.209
2 - acc: 0.9363 - val_loss: 0.3284 - val_acc: 0.8914
Epoch 21/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.188
4 - acc: 0.9400 - val_loss: 0.4429 - val_acc: 0.8921
Epoch 22/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.176
3 - acc: 0.9412 - val_loss: 0.3904 - val_acc: 0.9043
Epoch 23/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.172
1 - acc: 0.9403 - val_loss: 0.4405 - val_acc: 0.9050
Epoch 24/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.180
5 - acc: 0.9397 - val_loss: 0.2988 - val_acc: 0.9074
Epoch 25/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.217
5 - acc: 0.9402 - val_loss: 0.3831 - val_acc: 0.8996
Epoch 26/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.175
3 - acc: 0.9408 - val_loss: 0.3904 - val_acc: 0.8968
Epoch 27/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.157
3 - acc: 0.9412 - val_loss: 0.3716 - val_acc: 0.9033
Epoch 28/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.166
3 - acc: 0.9421 - val_loss: 0.3816 - val_acc: 0.9094
Epoch 29/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.161
9 - acc: 0.9434 - val_loss: 0.3260 - val_acc: 0.9135
Epoch 30/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.163
6 - acc: 0.9474 - val_loss: 0.3792 - val_acc: 0.9077
```

In [20]:

```
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores[0]))
test_acc1= scores[1]*100
train_acc1=(max(hist1.history['acc']))* 100
```

```
print("Train Accuracy: %f%%"% (train_acc1))

print("Test Accuracy: %f%%" % (test_acc1))

# error plot
x=list(range(1,epochs+1))
vy=hist1.history['val_loss'] #validation loss
ty=hist1.history['loss'] # train loss
plt_dynamic(x, vy, ty)
```

```
Test Score: 0.379200
Train Accuracy: 94.736126%
Test Accuracy: 90.770275%
```



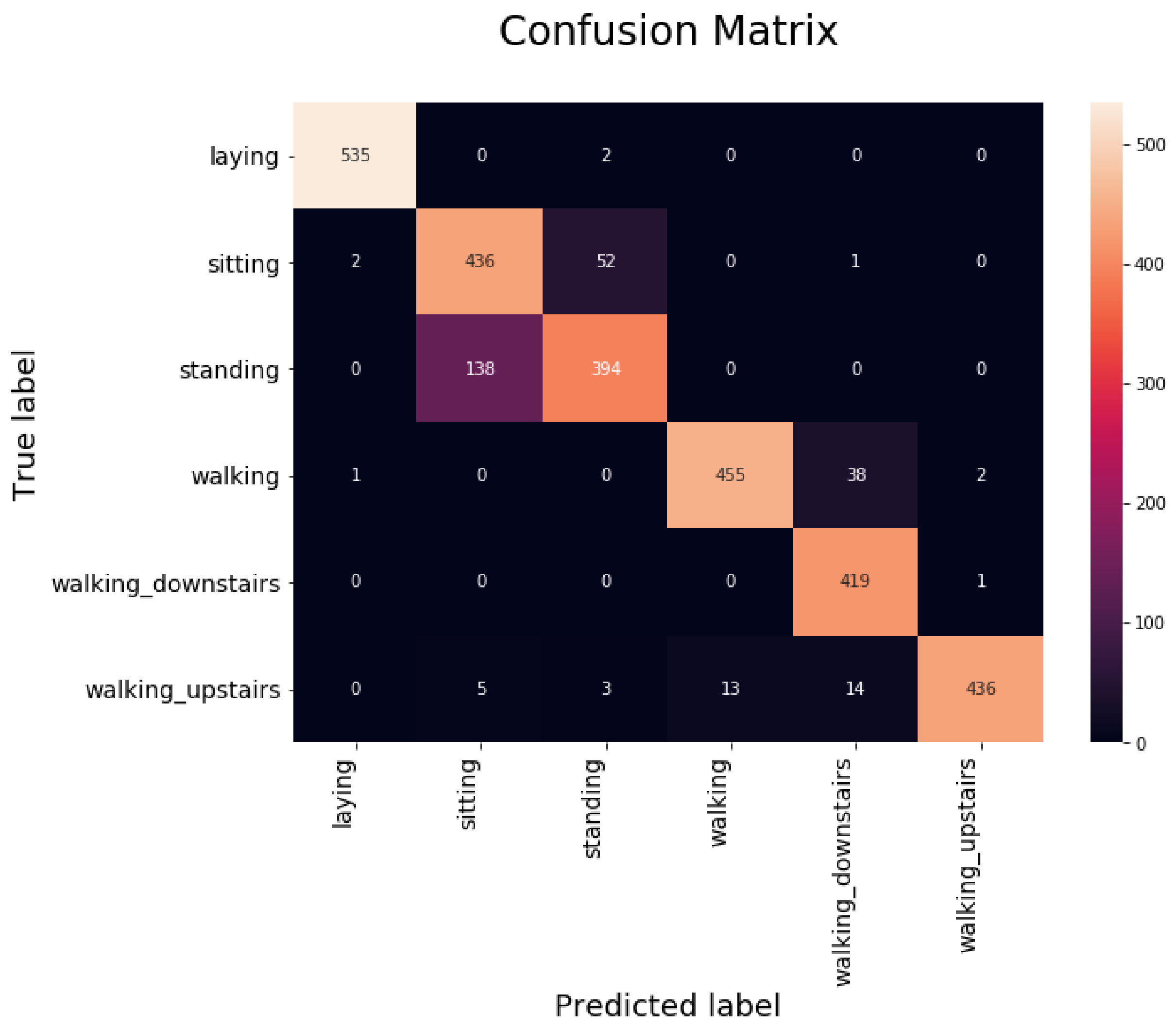
Observation:

- From above plot,it can be diagnosied that model is performing overfitting.
- The training error graph is reducing continuously and Validation graph is descreasing upto inflection point and later it's increasing.

In [21]:
```
# Confusion_Matrix
```

```
Y_true = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_test, axis=1)])
print('1st')

Y_predictions = pd.Series([ACTIVITIES[y] for y in np.argmax(model.predi
ct(X_test),
                                                          axis=1)])
print('2nd')

# seaborn heatmaps
class_names = ['laying','sitting',
               'standing','walking',
               'walking_downstairs',
               'walking_upstairs']
con_mat=confusion_matrix(Y_true,Y_predictions)
print('3rd')
df_heatmap = pd.DataFrame(con_mat,
                          index=class_names,
                          columns=class_names )
print('4th')
fig = plt.figure(figsize=(10,7))
heatmap = sns.heatmap(df_heatmap,
                      annot=True, fmt="d")

#  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()
```

```
1st
2nd
3rd
4th
```



2) 32 LSTM + 1 layer LSTM + Adam optimizer

In [22]:
```
# 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='adam',
              metrics=['accuracy'])
# Training the model
hist2=model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_2 (LSTM)                (None, 32)                5376      
_________________________________________________________________
dropout_2 (Dropout)          (None, 32)                0         
_________________________________________________________________
dense_2 (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 [==============================] - 24s 3ms/step - loss: 1.370
0 - acc: 0.4207 - val_loss: 1.3605 - val_acc: 0.3858
Epoch 2/30
7352/7352 [==============================] - 24s 3ms/step - loss: 1.239
9 - acc: 0.4460 - val_loss: 1.3068 - val_acc: 0.4150
Epoch 3/30
7352/7352 [==============================] - 24s 3ms/step - loss: 1.182
```

```
8 - acc: 0.4551 - val_loss: 1.1227 - val_acc: 0.4645
Epoch 4/30
7352/7352 [==============================] - 24s 3ms/step - loss: 1.102
2 - acc: 0.5275 - val_loss: 1.0100 - val_acc: 0.6016
Epoch 5/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.857
0 - acc: 0.6140 - val_loss: 0.9916 - val_acc: 0.5836
Epoch 6/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.887
3 - acc: 0.6019 - val_loss: 0.8917 - val_acc: 0.6281
Epoch 7/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.903
4 - acc: 0.6064 - val_loss: 0.8474 - val_acc: 0.6135
Epoch 8/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.769
3 - acc: 0.6442 - val_loss: 0.9360 - val_acc: 0.5786
Epoch 9/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.839
6 - acc: 0.6164 - val_loss: 0.8555 - val_acc: 0.6250
Epoch 10/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.741
5 - acc: 0.6522 - val_loss: 0.8564 - val_acc: 0.6298
Epoch 11/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.707
6 - acc: 0.6581 - val_loss: 0.8041 - val_acc: 0.6454
Epoch 12/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.686
7 - acc: 0.6712 - val_loss: 1.0092 - val_acc: 0.6118
Epoch 13/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.664
4 - acc: 0.6999 - val_loss: 0.8403 - val_acc: 0.6960
Epoch 14/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.583
2 - acc: 0.7692 - val_loss: 0.7345 - val_acc: 0.7723
Epoch 15/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.533
9 - acc: 0.8195 - val_loss: 0.6299 - val_acc: 0.7825
Epoch 16/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.409
```

```
6 - acc: 0.8643 - val_loss: 0.5387 - val_acc: 0.8409
Epoch 17/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.396
3 - acc: 0.8713 - val_loss: 0.4963 - val_acc: 0.8514
Epoch 18/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.341
9 - acc: 0.9038 - val_loss: 0.4934 - val_acc: 0.8690
Epoch 19/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.305
5 - acc: 0.9011 - val_loss: 0.3953 - val_acc: 0.8816
Epoch 20/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.256
3 - acc: 0.9172 - val_loss: 0.4246 - val_acc: 0.8918
Epoch 21/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.385
2 - acc: 0.8768 - val_loss: 0.4623 - val_acc: 0.8595
Epoch 22/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.249
9 - acc: 0.9227 - val_loss: 0.4276 - val_acc: 0.8860
Epoch 23/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.203
9 - acc: 0.9310 - val_loss: 0.3738 - val_acc: 0.8863
Epoch 24/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.402
3 - acc: 0.8913 - val_loss: 0.3902 - val_acc: 0.8904
Epoch 25/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.267
3 - acc: 0.9215 - val_loss: 0.3429 - val_acc: 0.8877
Epoch 26/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.218
3 - acc: 0.9305 - val_loss: 0.3252 - val_acc: 0.8968
Epoch 27/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.378
1 - acc: 0.8656 - val_loss: 0.5003 - val_acc: 0.8300
Epoch 28/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.345
0 - acc: 0.8724 - val_loss: 0.4568 - val_acc: 0.8836
Epoch 29/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.247
```

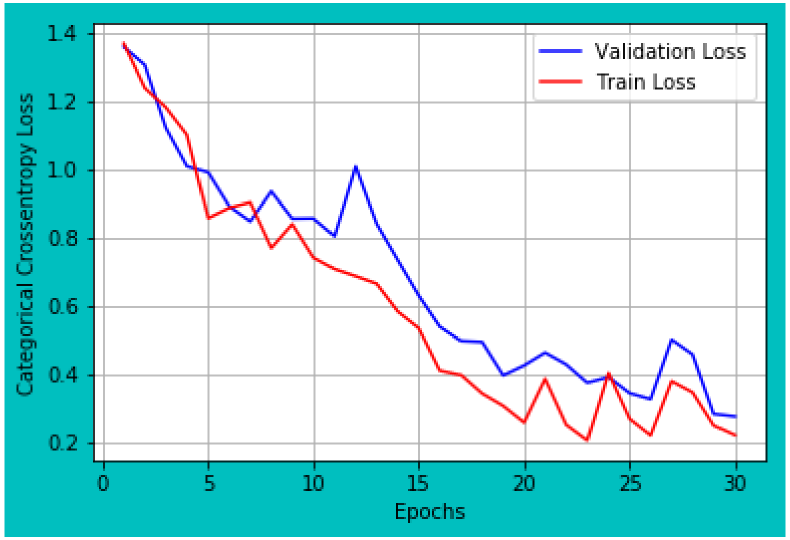
```
7 - acc: 0.9135 - val_loss: 0.2817 - val_acc: 0.8856
Epoch 30/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.219
2 - acc: 0.9291 - val_loss: 0.2744 - val_acc: 0.8999
```

In [ ]:

In [23]:

```
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores[0]))
test_acc2= scores[1]*100
train_acc2=(max(hist2.history['acc']))* 100
print("Train Accuracy: %f%%"% (train_acc2))

print("Test Accuracy: %f%%" % (test_acc2))
# error plot
x=list(range(1,epochs+1))
vy=hist2.history['val_loss'] #validation loss
ty=hist2.history['loss'] # train loss
plt_dynamic(x, vy, ty)
```

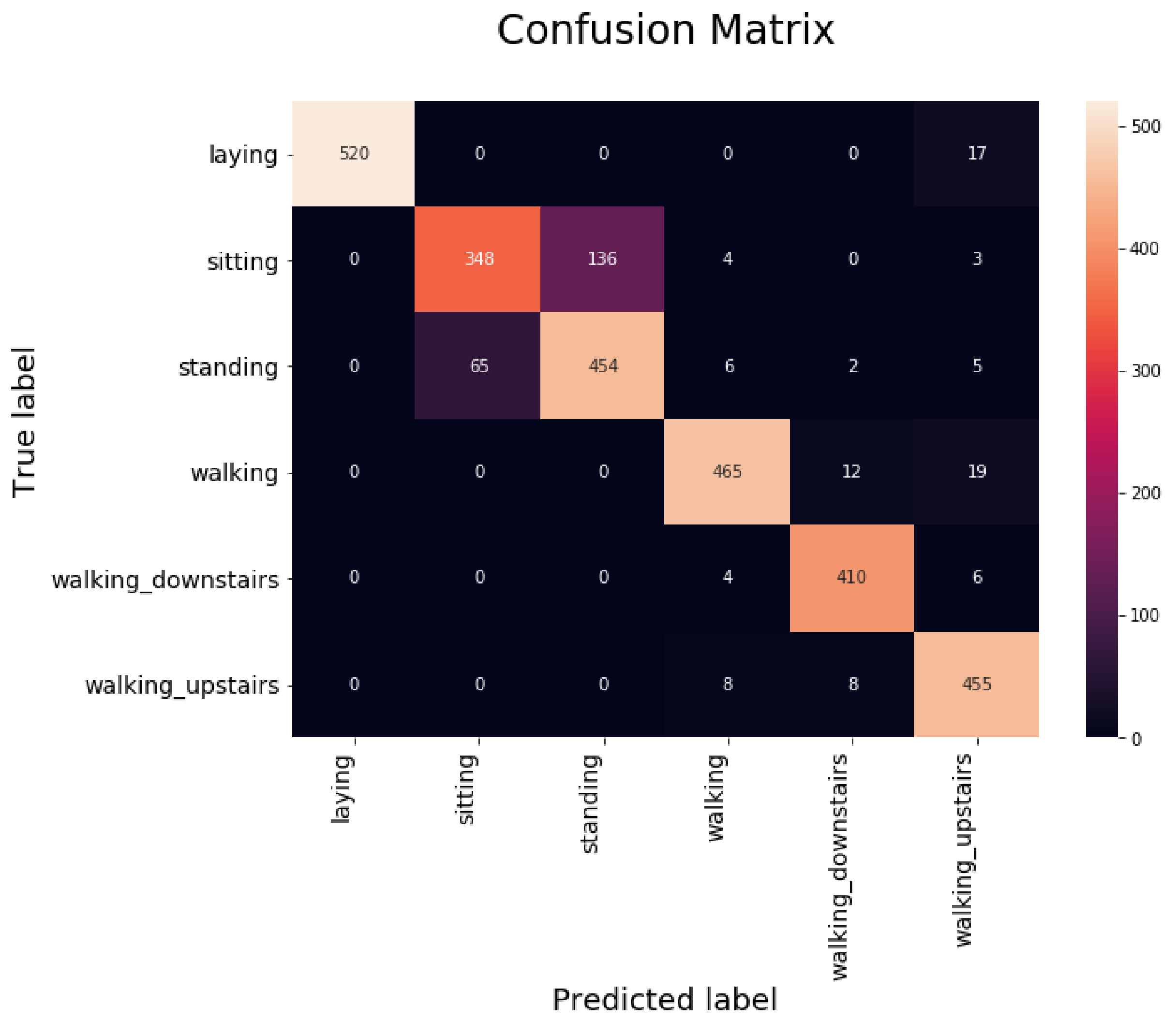
```
Test Score: 0.274399
Train Accuracy: 93.103917%
Test Accuracy: 89.989820%
```



- Above model performs overfitting.

In [24]:

```
# 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.predi
ct(X_test), axis=1)])

# seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downsta
irs','walking_upstairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), inde
x=class_names, columns=class_names )
fig = plt.figure(figsize=(10,7))
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")

#  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()
```



3) 64 LSTM + 1 layer LSTM + rmsprop optimizer

In [25]:
```
# Initiliazing the sequential model
model = Sequential()
```

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

# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
# Training the model
hist3=model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 64)                18944
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_3 (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 [==============================] - 31s 4ms/step - loss: 1.282
7 - acc: 0.4340 - val_loss: 1.1831 - val_acc: 0.4574
Epoch 2/30
7352/7352 [==============================] - 30s 4ms/step - loss: 1.040
5 - acc: 0.5345 - val_loss: 0.9734 - val_acc: 0.5667
Epoch 3/30
```

```
7352/7352 [==============================] - 30s 4ms/step - loss: 0.744
8 - acc: 0.6507 - val_loss: 0.8002 - val_acc: 0.6759
Epoch 4/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.796
4 - acc: 0.6745 - val_loss: 1.0634 - val_acc: 0.5660
Epoch 5/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.850
7 - acc: 0.6462 - val_loss: 0.8188 - val_acc: 0.6240
Epoch 6/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.551
6 - acc: 0.7889 - val_loss: 0.6243 - val_acc: 0.7628
Epoch 7/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.459
2 - acc: 0.8497 - val_loss: 1.2035 - val_acc: 0.6240
Epoch 8/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.505
5 - acc: 0.8305 - val_loss: 0.4375 - val_acc: 0.8548
Epoch 9/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.306
7 - acc: 0.9008 - val_loss: 0.6739 - val_acc: 0.8202
Epoch 10/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.262
9 - acc: 0.9162 - val_loss: 0.5049 - val_acc: 0.8694
Epoch 11/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.230
3 - acc: 0.9278 - val_loss: 0.5054 - val_acc: 0.8707
Epoch 12/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.206
9 - acc: 0.9302 - val_loss: 0.4603 - val_acc: 0.8768
Epoch 13/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.181
9 - acc: 0.9393 - val_loss: 0.5414 - val_acc: 0.8904
Epoch 14/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.202
0 - acc: 0.9344 - val_loss: 0.4737 - val_acc: 0.8795
Epoch 15/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.170
7 - acc: 0.9422 - val_loss: 0.3429 - val_acc: 0.9040
Epoch 16/30
```

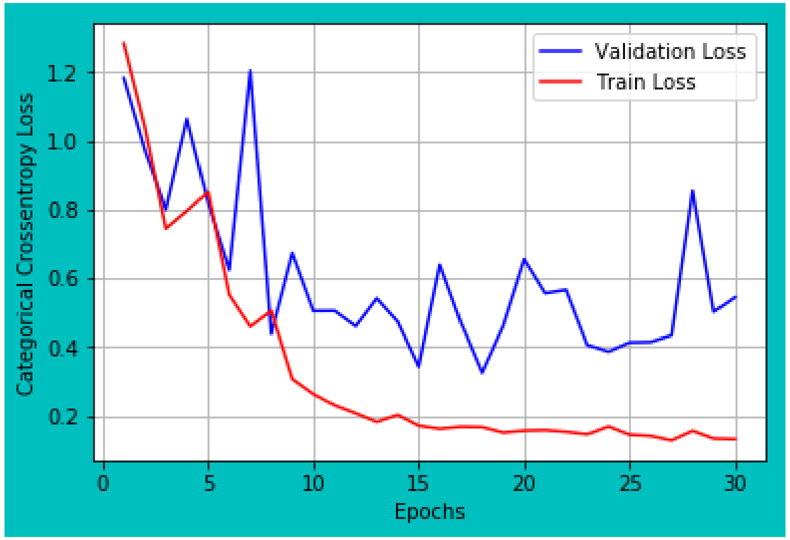
```
7352/7352 [==============================] - 30s 4ms/step - loss: 0.161
7 - acc: 0.9433 - val_loss: 0.6396 - val_acc: 0.8622
Epoch 17/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.167
6 - acc: 0.9452 - val_loss: 0.4702 - val_acc: 0.8823
Epoch 18/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.166
4 - acc: 0.9437 - val_loss: 0.3252 - val_acc: 0.9060
Epoch 19/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.150
6 - acc: 0.9448 - val_loss: 0.4614 - val_acc: 0.8945
Epoch 20/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.156
1 - acc: 0.9460 - val_loss: 0.6557 - val_acc: 0.8870
Epoch 21/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.157
4 - acc: 0.9446 - val_loss: 0.5562 - val_acc: 0.8907
Epoch 22/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.152
7 - acc: 0.9478 - val_loss: 0.5662 - val_acc: 0.8928
Epoch 23/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.145
7 - acc: 0.9459 - val_loss: 0.4055 - val_acc: 0.9013
Epoch 24/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.168
2 - acc: 0.9437 - val_loss: 0.3864 - val_acc: 0.9043
Epoch 25/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.144
3 - acc: 0.9499 - val_loss: 0.4129 - val_acc: 0.9091
Epoch 26/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.140
8 - acc: 0.9513 - val_loss: 0.4141 - val_acc: 0.9030
Epoch 27/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.127
8 - acc: 0.9491 - val_loss: 0.4337 - val_acc: 0.8996
Epoch 28/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.155
5 - acc: 0.9494 - val_loss: 0.8541 - val_acc: 0.8711
Epoch 29/30
```

```
7352/7352 [==============================] - 29s 4ms/step - loss: 0.133
2 - acc: 0.9520 - val_loss: 0.5022 - val_acc: 0.9023
Epoch 30/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.131
5 - acc: 0.9493 - val_loss: 0.5443 - val_acc: 0.9019
```

In [26]:

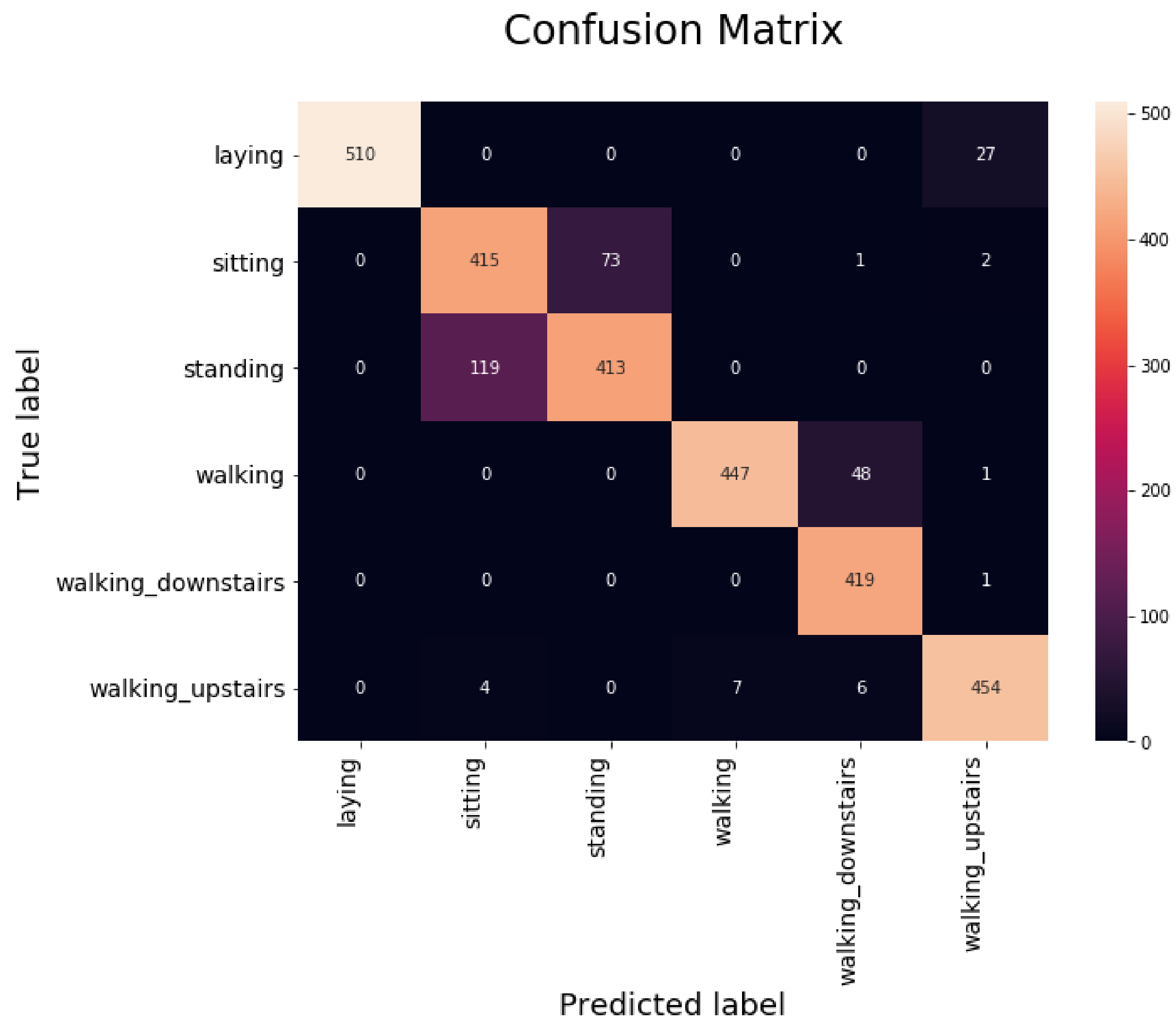
```
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores[0]))
test_acc3= scores[1]*100
train_acc3=(max(hist3.history['acc']))* 100
print("Train Accuracy: %f%%"% (train_acc3))

print("Test Accuracy: %f%%" % (test_acc3))
# error plot
x=list(range(1,epochs+1))
vy=hist3.history['val_loss'] #validation loss
ty=hist3.history['loss'] # train loss
plt_dynamic(x, vy, ty)
```

```
Test Score: 0.544287
Train Accuracy: 95.198585%
Test Accuracy: 90.193417%
```



In [27]:

```
# 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.predi
ct(X_test), axis=1)])

# seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downsta
irs','walking_upstairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), inde
x=class_names, columns=class_names )
fig = plt.figure(figsize=(10,7))
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")

#  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()
```



4) 64 LSTM + 1 layer LSTM + adam optimizer

In [28]:
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
```

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

# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
# Training the model
hist4=model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          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
_________________________________________________________________
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 30s 4ms/step - loss: 1.386
1 - acc: 0.3980 - val_loss: 1.3311 - val_acc: 0.3987
Epoch 2/30
7352/7352 [==============================] - 29s 4ms/step - loss: 1.264
6 - acc: 0.4353 - val_loss: 1.3222 - val_acc: 0.4506
Epoch 3/30
7352/7352 [==============================] - 29s 4ms/step - loss: 1.392
```

```
8 - acc: 0.3628 - val_loss: 1.3819 - val_acc: 0.3519
Epoch 4/30
7352/7352 [==============================] - 29s 4ms/step - loss: 1.338
3 - acc: 0.3727 - val_loss: 1.3513 - val_acc: 0.3482
Epoch 5/30
7352/7352 [==============================] - 29s 4ms/step - loss: 1.330
6 - acc: 0.3740 - val_loss: 1.3417 - val_acc: 0.3482
Epoch 6/30
7352/7352 [==============================] - 29s 4ms/step - loss: 1.297
1 - acc: 0.3943 - val_loss: 1.2594 - val_acc: 0.4299
Epoch 7/30
7352/7352 [==============================] - 29s 4ms/step - loss: 1.332
7 - acc: 0.3868 - val_loss: 1.3132 - val_acc: 0.4038
Epoch 8/30
7352/7352 [==============================] - 29s 4ms/step - loss: 1.266
7 - acc: 0.4391 - val_loss: 1.2172 - val_acc: 0.4995
Epoch 9/30
7352/7352 [==============================] - 29s 4ms/step - loss: 1.207
3 - acc: 0.4894 - val_loss: 1.3158 - val_acc: 0.3858
Epoch 10/30
7352/7352 [==============================] - 29s 4ms/step - loss: 1.171
6 - acc: 0.4988 - val_loss: 0.9594 - val_acc: 0.5453
Epoch 11/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.840
4 - acc: 0.6019 - val_loss: 0.8761 - val_acc: 0.5938
Epoch 12/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.740
7 - acc: 0.6356 - val_loss: 0.7237 - val_acc: 0.6342
Epoch 13/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.691
4 - acc: 0.6619 - val_loss: 0.7510 - val_acc: 0.6098
Epoch 14/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.719
2 - acc: 0.6585 - val_loss: 0.7900 - val_acc: 0.6149
Epoch 15/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.700
8 - acc: 0.6564 - val_loss: 0.7663 - val_acc: 0.6518
Epoch 16/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.665
```

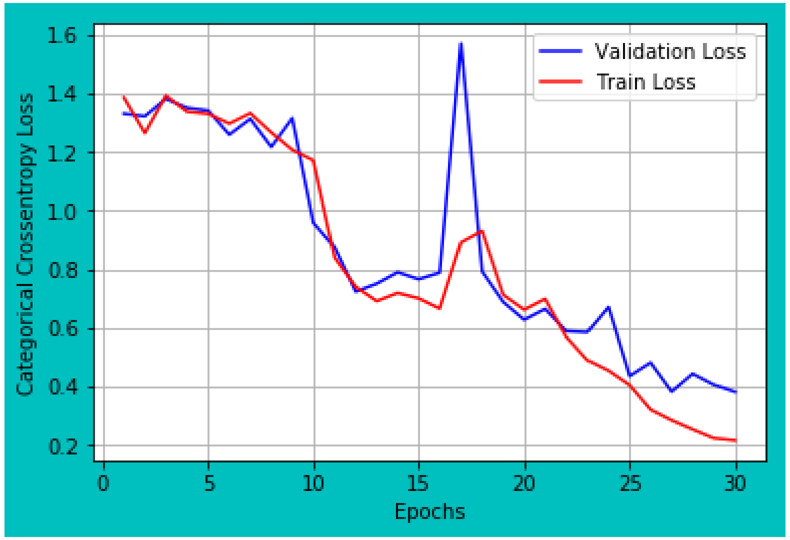
```
9 - acc: 0.6955 - val_loss: 0.7885 - val_acc: 0.7048
Epoch 17/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.892
5 - acc: 0.5839 - val_loss: 1.5695 - val_acc: 0.2945
Epoch 18/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.931
8 - acc: 0.5718 - val_loss: 0.7928 - val_acc: 0.6325
Epoch 19/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.713
2 - acc: 0.6574 - val_loss: 0.6886 - val_acc: 0.6814
Epoch 20/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.661
9 - acc: 0.6903 - val_loss: 0.6277 - val_acc: 0.7139
Epoch 21/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.699
2 - acc: 0.7240 - val_loss: 0.6648 - val_acc: 0.7126
Epoch 22/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.569
9 - acc: 0.7561 - val_loss: 0.5894 - val_acc: 0.7570
Epoch 23/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.490
2 - acc: 0.7930 - val_loss: 0.5857 - val_acc: 0.7706
Epoch 24/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.454
7 - acc: 0.8192 - val_loss: 0.6714 - val_acc: 0.7258
Epoch 25/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.405
5 - acc: 0.8542 - val_loss: 0.4363 - val_acc: 0.8364
Epoch 26/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.321
6 - acc: 0.8862 - val_loss: 0.4823 - val_acc: 0.8490
Epoch 27/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.285
4 - acc: 0.9032 - val_loss: 0.3832 - val_acc: 0.8782
Epoch 28/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.254
2 - acc: 0.9121 - val_loss: 0.4442 - val_acc: 0.8524
Epoch 29/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.224
```

```
3 - acc: 0.9232 - val_loss: 0.4070 - val_acc: 0.8660
Epoch 30/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.216
5 - acc: 0.9208 - val_loss: 0.3825 - val_acc: 0.8928
```

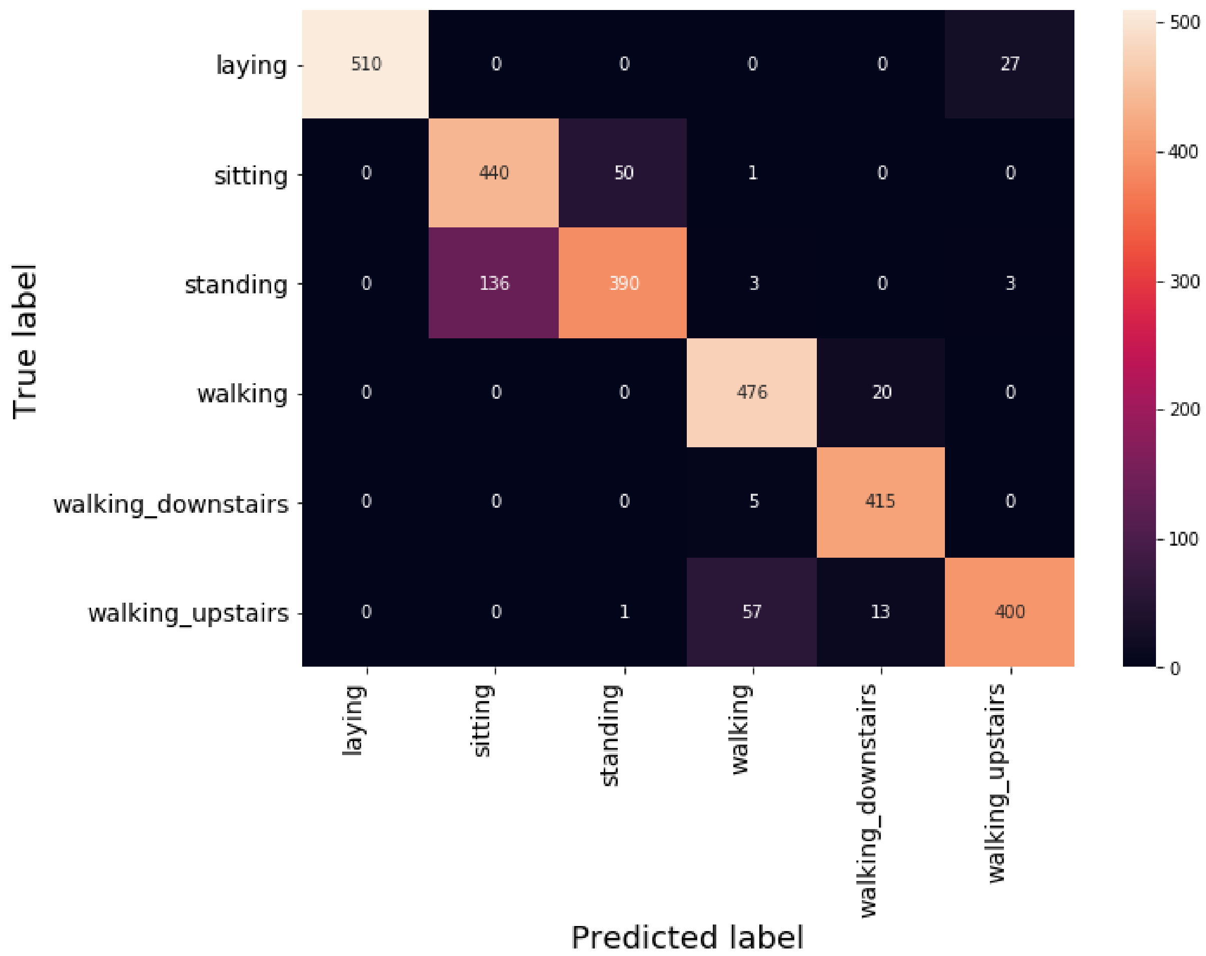
In [29]:

```
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores[0]))
test_acc4= scores[1]*100
train_acc4=(max(hist4.history['acc']))* 100
print("Train Accuracy: %f%%"% (train_acc4))

print("Test Accuracy: %f%%" % (test_acc4))
# error plot
vy=hist4.history['val_loss'] #validation loss
ty=hist4.history['loss'] # train loss
plt_dynamic(x, vy, ty)

# 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.predi
ct(X_test), axis=1)])

# seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downsta
irs','walking_upstairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), inde
x=class_names, columns=class_names )
fig = plt.figure(figsize=(10,7))
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")

#  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: 0.382528
Train Accuracy: 92.315016%
Test Accuracy: 89.277231%
```



Confusion Matrix



5) 32 LSTM + 2 layer LSTM + rmsprop optimizer

In [30]:
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(32,return_sequences=True,
```

```
                input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.65))
# second LSTM layer
model.add(LSTM(32))
model.add(Dropout(0.65))
# 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
hist5=model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          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
_________________________________________________________________
Train on 7352 samples, validate on 2947 samples

Epoch 1/30
```

```
Epoch 1/30
7352/7352 [==============================] - 54s 7ms/step - loss: 1.314
9 - acc: 0.4533 - val_loss: 1.0325 - val_acc: 0.4913
Epoch 2/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.920
4 - acc: 0.5929 - val_loss: 0.8322 - val_acc: 0.6108
Epoch 3/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.808
5 - acc: 0.6246 - val_loss: 0.7718 - val_acc: 0.6030
Epoch 4/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.731
8 - acc: 0.6541 - val_loss: 0.7303 - val_acc: 0.6420
Epoch 5/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.669
6 - acc: 0.6748 - val_loss: 0.6972 - val_acc: 0.6641
Epoch 6/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.605
2 - acc: 0.7114 - val_loss: 0.6408 - val_acc: 0.7160
Epoch 7/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.558
5 - acc: 0.7816 - val_loss: 0.5922 - val_acc: 0.7435
Epoch 8/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.472
3 - acc: 0.8357 - val_loss: 0.5611 - val_acc: 0.8483
Epoch 9/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.395
1 - acc: 0.8814 - val_loss: 0.7218 - val_acc: 0.8229
Epoch 10/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.314
8 - acc: 0.9072 - val_loss: 0.5134 - val_acc: 0.8758
Epoch 11/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.266
9 - acc: 0.9223 - val_loss: 0.4106 - val_acc: 0.9009
Epoch 12/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.248
9 - acc: 0.9286 - val_loss: 0.4730 - val_acc: 0.8901
Epoch 13/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.234
7 - acc: 0.9359 - val_loss: 0.5271 - val_acc: 0.8863

Epoch 14/30
```

```
Epoch 14/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.236
7 - acc: 0.9342 - val_loss: 0.4882 - val_acc: 0.9023
Epoch 15/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.233
2 - acc: 0.9343 - val_loss: 0.4688 - val_acc: 0.9036
Epoch 16/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.213
5 - acc: 0.9372 - val_loss: 0.6286 - val_acc: 0.8911
Epoch 17/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.190
1 - acc: 0.9399 - val_loss: 0.4936 - val_acc: 0.9074
Epoch 18/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.197
8 - acc: 0.9388 - val_loss: 0.6069 - val_acc: 0.8901
Epoch 19/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.223
6 - acc: 0.9344 - val_loss: 0.4921 - val_acc: 0.9141
Epoch 20/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.181
3 - acc: 0.9382 - val_loss: 0.6226 - val_acc: 0.8880
Epoch 21/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.209
6 - acc: 0.9358 - val_loss: 0.5694 - val_acc: 0.9074
Epoch 22/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.204
9 - acc: 0.9363 - val_loss: 0.4863 - val_acc: 0.9067
Epoch 23/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.181
0 - acc: 0.9415 - val_loss: 0.5510 - val_acc: 0.9023
Epoch 24/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.183
3 - acc: 0.9392 - val_loss: 0.4904 - val_acc: 0.9104
Epoch 25/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.191
7 - acc: 0.9419 - val_loss: 0.5181 - val_acc: 0.9009
Epoch 26/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.176
1 - acc: 0.9421 - val_loss: 0.5803 - val_acc: 0.9213

Epoch 27/30
```

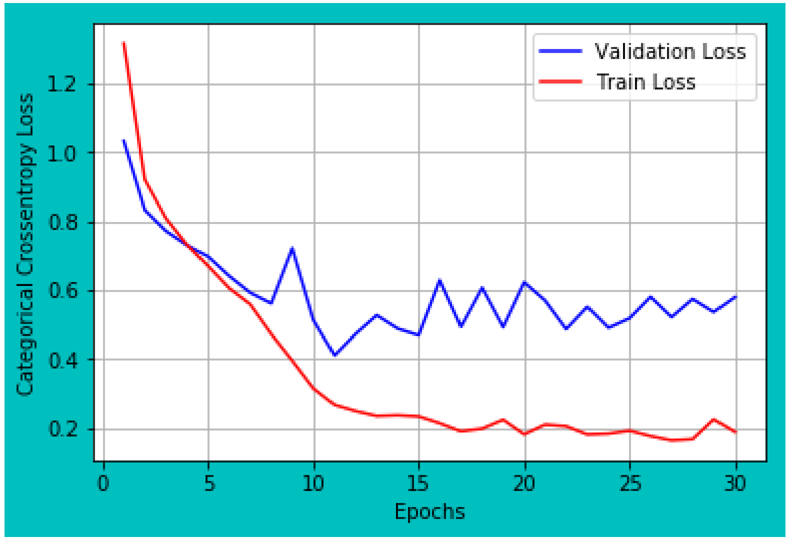
```
Epoch 27/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.163
7 - acc: 0.9455 - val_loss: 0.5213 - val_acc: 0.9050
Epoch 28/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.167
1 - acc: 0.9418 - val_loss: 0.5739 - val_acc: 0.9128
Epoch 29/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.224
6 - acc: 0.9381 - val_loss: 0.5360 - val_acc: 0.9101
Epoch 30/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.188
3 - acc: 0.9402 - val_loss: 0.5793 - val_acc: 0.9009
```

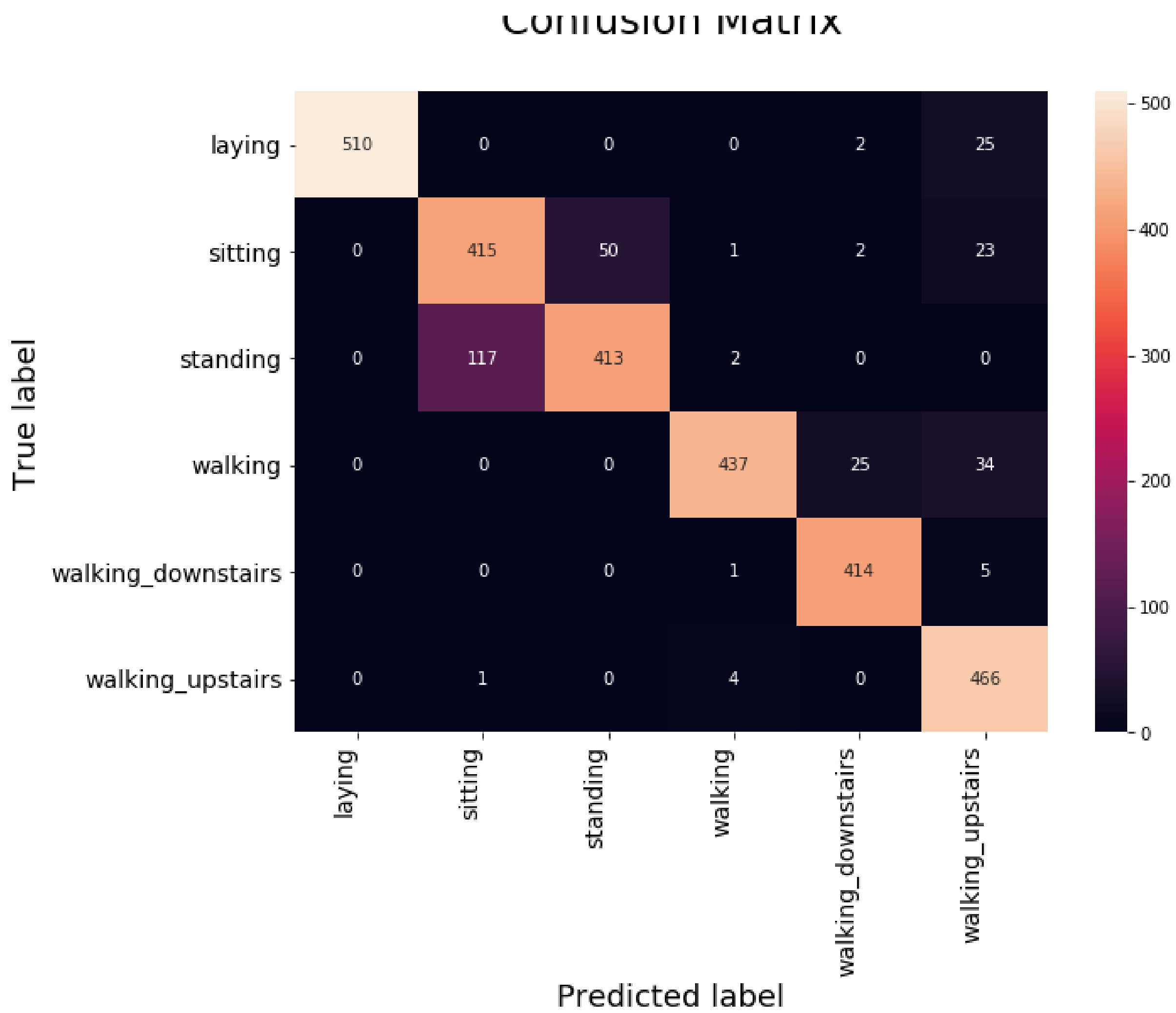
In [31]:

```
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores[0]))
test_acc5= scores[1]*100
train_acc5=(max(hist5.history['acc']))* 100
print("Train Accuracy: %f%%"% (train_acc5))

print("Test Accuracy: %f%%" % (test_acc5))
# error plot
vy=hist5.history['val_loss'] #validation loss
ty=hist5.history['loss'] # train loss
plt_dynamic(x, vy, ty)

# 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.predi
ct(X_test), axis=1)])

# seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downsta
irs','walking_upstairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), inde
x=class_names, columns=class_names )
fig = plt.figure(figsize=(10,7))
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")

#  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: 0.579289
Train Accuracy: 94.545702%
Test Accuracy: 90.091619%
```





6) 32 LSTM + 2 layer LSTM + adam optimizer

In [32]:
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(32,return_sequences=True,
```

```
                    input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.65))
# second LSTM layer
model.add(LSTM(32))
model.add(Dropout(0.65))
# 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='adam',
              metrics=['accuracy'])
# Training the model
hist6=model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_7 (LSTM)                (None, 128, 32)           5376      
_________________________________________________________________
dropout_7 (Dropout)          (None, 128, 32)           0         
_________________________________________________________________
lstm_8 (LSTM)                (None, 32)                8320      
_________________________________________________________________
dropout_8 (Dropout)          (None, 32)                0         
_________________________________________________________________
dense_6 (Dense)              (None, 6)                 198       
=================================================================
Total params: 13,894
Trainable params: 13,894
Non-trainable params: 0
_________________________________________________________________
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
```

```
7352/7352 [==============================] - 56s 8ms/step - loss: 1.381
7 - acc: 0.4410 - val_loss: 1.3614 - val_acc: 0.3651
Epoch 2/30
7352/7352 [==============================] - 55s 7ms/step - loss: 1.065
6 - acc: 0.5241 - val_loss: 0.9279 - val_acc: 0.5803
Epoch 3/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.853
4 - acc: 0.5861 - val_loss: 0.8546 - val_acc: 0.5405
Epoch 4/30
7352/7352 [==============================] - 55s 7ms/step - loss: 0.873
9 - acc: 0.5762 - val_loss: 0.7923 - val_acc: 0.6067
Epoch 5/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.807
4 - acc: 0.5754 - val_loss: 0.8162 - val_acc: 0.5792
Epoch 6/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.779
7 - acc: 0.5896 - val_loss: 0.8430 - val_acc: 0.6135
Epoch 7/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.744
3 - acc: 0.6247 - val_loss: 0.7904 - val_acc: 0.6115
Epoch 8/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.707
1 - acc: 0.6469 - val_loss: 0.7394 - val_acc: 0.6250
Epoch 9/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.690
6 - acc: 0.6620 - val_loss: 0.7115 - val_acc: 0.6233
Epoch 10/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.649
3 - acc: 0.6789 - val_loss: 0.7445 - val_acc: 0.6227
Epoch 11/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.629
1 - acc: 0.6971 - val_loss: 0.7597 - val_acc: 0.6403
Epoch 12/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.721
1 - acc: 0.6330 - val_loss: 0.7211 - val_acc: 0.6203
Epoch 13/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.719
0 - acc: 0.6421 - val_loss: 0.7027 - val_acc: 0.6291
Epoch 14/30
```

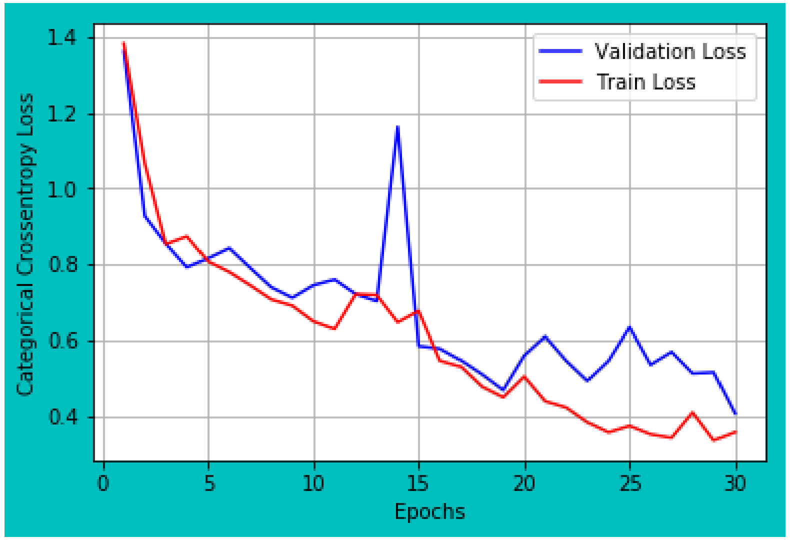
```
7352/7352 [==============================] - 53s 7ms/step - loss: 0.646
9 - acc: 0.6729 - val_loss: 1.1626 - val_acc: 0.4822
Epoch 15/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.677
1 - acc: 0.6659 - val_loss: 0.5845 - val_acc: 0.6332
Epoch 16/30
7352/7352 [==============================] - 55s 7ms/step - loss: 0.546
1 - acc: 0.7035 - val_loss: 0.5777 - val_acc: 0.6315
Epoch 17/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.530
1 - acc: 0.7149 - val_loss: 0.5467 - val_acc: 0.7499
Epoch 18/30
7352/7352 [==============================] - 55s 7ms/step - loss: 0.477
4 - acc: 0.7743 - val_loss: 0.5099 - val_acc: 0.7581
Epoch 19/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.449
4 - acc: 0.7881 - val_loss: 0.4685 - val_acc: 0.7391
Epoch 20/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.504
2 - acc: 0.7654 - val_loss: 0.5593 - val_acc: 0.7394
Epoch 21/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.438
9 - acc: 0.7833 - val_loss: 0.6096 - val_acc: 0.7520
Epoch 22/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.422
3 - acc: 0.7911 - val_loss: 0.5456 - val_acc: 0.7513
Epoch 23/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.383
2 - acc: 0.8074 - val_loss: 0.4929 - val_acc: 0.7845
Epoch 24/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.356
7 - acc: 0.8104 - val_loss: 0.5456 - val_acc: 0.7608
Epoch 25/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.373
8 - acc: 0.8138 - val_loss: 0.6342 - val_acc: 0.7679
Epoch 26/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.351
6 - acc: 0.8160 - val_loss: 0.5351 - val_acc: 0.7621
Epoch 27/30
```

```
7352/7352 [==============================] - 54s 7ms/step - loss: 0.342
3 - acc: 0.8305 - val_loss: 0.5691 - val_acc: 0.7642
Epoch 28/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.409
2 - acc: 0.8251 - val_loss: 0.5124 - val_acc: 0.8385
Epoch 29/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.335
5 - acc: 0.8434 - val_loss: 0.5154 - val_acc: 0.8690
Epoch 30/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.357
3 - acc: 0.8507 - val_loss: 0.4068 - val_acc: 0.8663
```

In [33]:

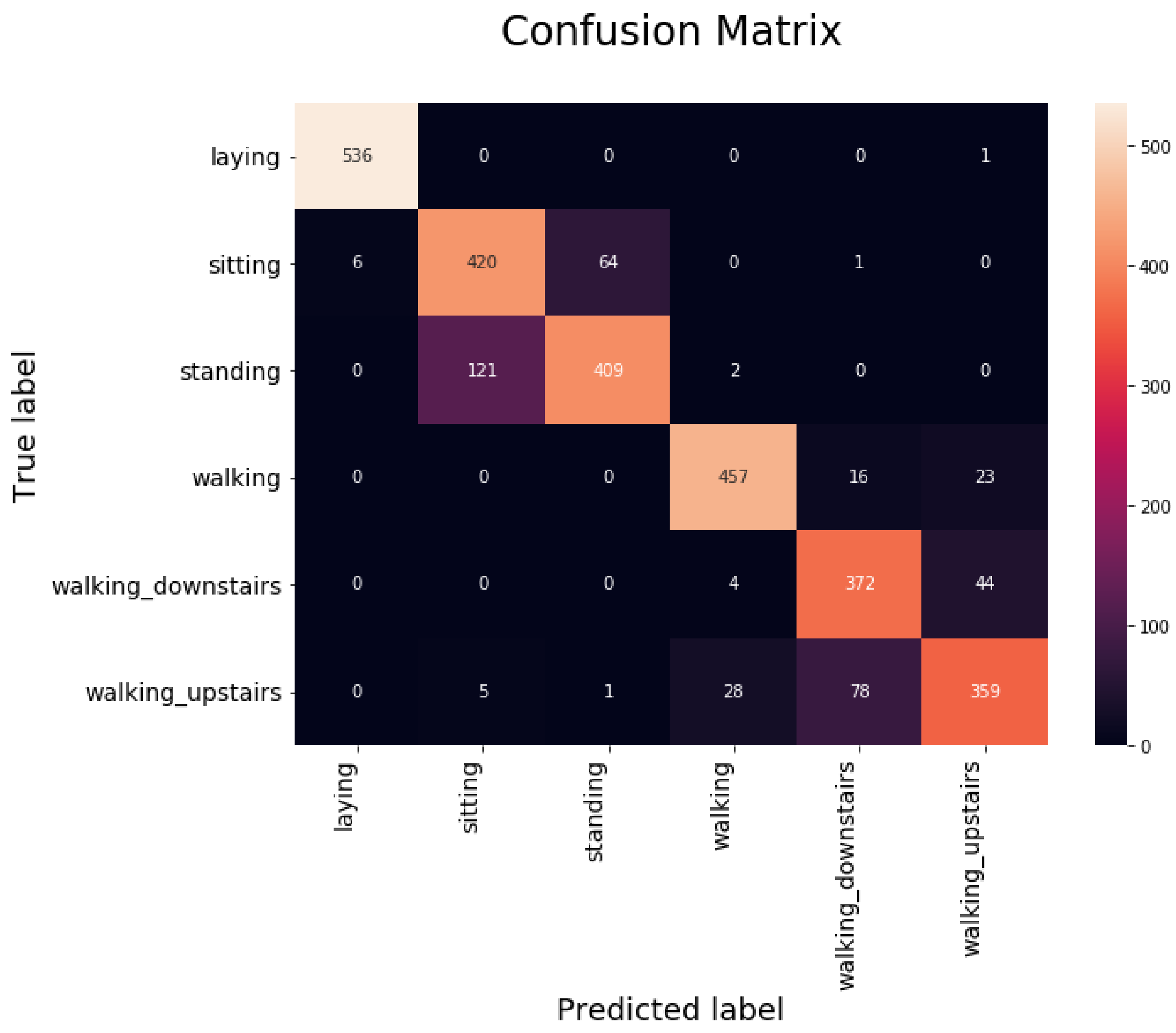
```
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores[0]))
test_acc6= scores[1]*100
train_acc6=(max(hist6.history['acc']))* 100
print("Train Accuracy: %f%%"% (train_acc6))

print("Test Accuracy: %f%%" % (test_acc6))
# error plot
vy=hist6.history['val_loss'] #validation loss
ty=hist6.history['loss'] # train loss
plt_dynamic(x, vy, ty)
```

```
Test Score: 0.406773
Train Accuracy: 85.065288%
Test Accuracy: 86.630472%
```



In [34]:

```
# 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.predi
ct(X_test), axis=1)])

# seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downsta
irs','walking_upstairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), inde
x=class_names, columns=class_names )
fig = plt.figure(figsize=(10,7))
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")

#  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()
```



7) 64 LSTM + 2 layer LSTM + adam optimizer+ 0.65 drop_out

In [35]:
```
# Initiliazing the sequential model
model = Sequential()
```

```python
# Configuring the parameters
model.add(LSTM(64,return_sequences=True,
               input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.65))
# second LSTM layer
model.add(LSTM(64))
model.add(Dropout(0.65))
# 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='adam',
              metrics=['accuracy'])
# Training the model
hist7=model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_9 (LSTM)                (None, 128, 64)           18944
_________________________________________________________________
dropout_9 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
lstm_10 (LSTM)               (None, 64)                33024
_________________________________________________________________
dropout_10 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_7 (Dense)              (None, 6)                 390
=================================================================
Total params: 52,358
Trainable params: 52,358
Non-trainable params: 0
_________________________________________________________________
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 75s 10ms/step - loss: 1.24
60 - acc: 0.4650 - val_loss: 1.1507 - val_acc: 0.4608
Epoch 2/30
7352/7352 [==============================] - 72s 10ms/step - loss: 1.09
60 - acc: 0.5037 - val_loss: 1.3114 - val_acc: 0.5053
Epoch 3/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.92
48 - acc: 0.5747 - val_loss: 0.8509 - val_acc: 0.5935
Epoch 4/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.82
30 - acc: 0.5872 - val_loss: 0.8356 - val_acc: 0.5921
Epoch 5/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.73
64 - acc: 0.6193 - val_loss: 0.7757 - val_acc: 0.6359
Epoch 6/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.73
91 - acc: 0.6291 - val_loss: 0.8603 - val_acc: 0.5042
Epoch 7/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.74
22 - acc: 0.6066 - val_loss: 0.7962 - val_acc: 0.6037
Epoch 8/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.73
99 - acc: 0.6348 - val_loss: 0.7643 - val_acc: 0.6108
Epoch 9/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.66
58 - acc: 0.6737 - val_loss: 0.6736 - val_acc: 0.6335
Epoch 10/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.53
10 - acc: 0.7695 - val_loss: 0.5680 - val_acc: 0.7923
Epoch 11/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.44
09 - acc: 0.8456 - val_loss: 0.4612 - val_acc: 0.8595
Epoch 12/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.42
54 - acc: 0.8546 - val_loss: 0.4187 - val_acc: 0.8728
Epoch 13/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.34
```

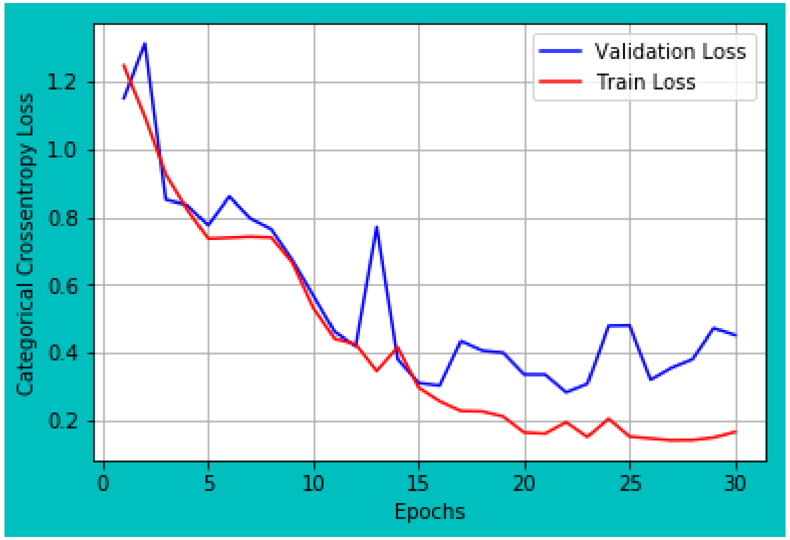
```
59 - acc: 0.8916 - val_loss: 0.7711 - val_acc: 0.6875
Epoch 14/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.41
64 - acc: 0.8493 - val_loss: 0.3807 - val_acc: 0.8890
Epoch 15/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.29
67 - acc: 0.9066 - val_loss: 0.3106 - val_acc: 0.8911
Epoch 16/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.25
65 - acc: 0.9098 - val_loss: 0.3027 - val_acc: 0.8975
Epoch 17/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.22
76 - acc: 0.9264 - val_loss: 0.4344 - val_acc: 0.8843
Epoch 18/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.22
62 - acc: 0.9328 - val_loss: 0.4064 - val_acc: 0.8884
Epoch 19/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.21
13 - acc: 0.9295 - val_loss: 0.4000 - val_acc: 0.9067
Epoch 20/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.16
37 - acc: 0.9463 - val_loss: 0.3357 - val_acc: 0.9067
Epoch 21/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.16
05 - acc: 0.9418 - val_loss: 0.3353 - val_acc: 0.9053
Epoch 22/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.19
43 - acc: 0.9369 - val_loss: 0.2827 - val_acc: 0.9155
Epoch 23/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.15
10 - acc: 0.9484 - val_loss: 0.3076 - val_acc: 0.9169
Epoch 24/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.20
44 - acc: 0.9301 - val_loss: 0.4783 - val_acc: 0.8856
Epoch 25/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.15
16 - acc: 0.9440 - val_loss: 0.4793 - val_acc: 0.9033
Epoch 26/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.14
```

```
60 - acc: 0.9482 - val_loss: 0.3199 - val_acc: 0.9169
Epoch 27/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.14
05 - acc: 0.9531 - val_loss: 0.3554 - val_acc: 0.9223
Epoch 28/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.14
14 - acc: 0.9464 - val_loss: 0.3807 - val_acc: 0.9141
Epoch 29/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.14
87 - acc: 0.9489 - val_loss: 0.4713 - val_acc: 0.8982
Epoch 30/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.16
54 - acc: 0.9372 - val_loss: 0.4509 - val_acc: 0.8955
```

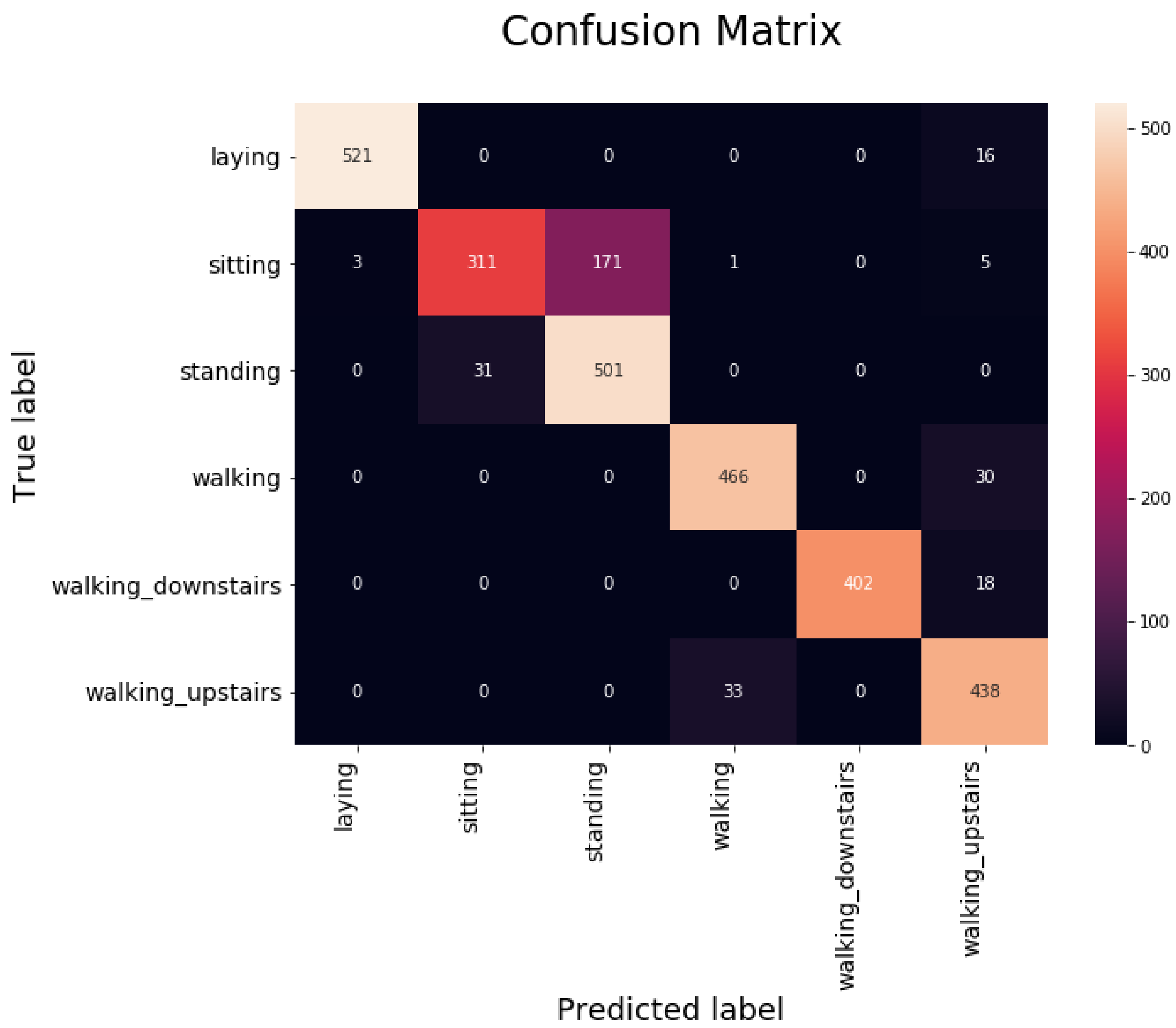
In [36]:

```
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores[0]))
test_acc7= scores[1]*100
train_acc7=(max(hist7.history['acc']))* 100
print("Train Accuracy: %f%%"% (train_acc7))

print("Test Accuracy: %f%%" % (test_acc7))
# error plot
vy=hist7.history['val_loss'] #validation loss
ty=hist7.history['loss'] # train loss
plt_dynamic(x, vy, ty)
```

```
Test Score: 0.450901
Train Accuracy: 95.307399%
Test Accuracy: 89.548694%
```



In [37]:
```
# 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.predi
ct(X_test), axis=1)])

# seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downsta
irs','walking_upstairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), inde
x=class_names, columns=class_names )
fig = plt.figure(figsize=(10,7))
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")

#  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()
```



8) 64 LSTM + 2 layer LSTM + rmsprop optimizer+ 0.65 drop_out

In [38]:

```
# Initiliazing the sequential model
model = Sequential()
```

```python
# Configuring the parameters
model.add(LSTM(64,return_sequences=True,
               input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.65))
# second LSTM layer
model.add(LSTM(64))
model.add(Dropout(0.65))
# 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='adam',
              metrics=['accuracy'])
# Training the model
hist8=model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_11 (LSTM)               (None, 128, 64)           18944     
_________________________________________________________________
dropout_11 (Dropout)         (None, 128, 64)           0         
_________________________________________________________________
lstm_12 (LSTM)               (None, 64)                33024     
_________________________________________________________________
dropout_12 (Dropout)         (None, 64)                0         
_________________________________________________________________
dense_8 (Dense)              (None, 6)                 390       
=================================================================
Total params: 52,358
Trainable params: 52,358
Non-trainable params: 0
_________________________________________________________________
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 76s 10ms/step - loss: 1.28
49 - acc: 0.4374 - val_loss: 0.8938 - val_acc: 0.5351
Epoch 2/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.84
02 - acc: 0.5975 - val_loss: 0.8225 - val_acc: 0.6016
Epoch 3/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.82
33 - acc: 0.6019 - val_loss: 1.1463 - val_acc: 0.4937
Epoch 4/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.91
99 - acc: 0.5400 - val_loss: 0.9346 - val_acc: 0.5402
Epoch 5/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.78
38 - acc: 0.5828 - val_loss: 0.8286 - val_acc: 0.5592
Epoch 6/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.80
73 - acc: 0.5964 - val_loss: 1.0598 - val_acc: 0.5083
Epoch 7/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.78
81 - acc: 0.6172 - val_loss: 0.7484 - val_acc: 0.6013
Epoch 8/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.67
85 - acc: 0.6458 - val_loss: 1.0201 - val_acc: 0.5606
Epoch 9/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.65
42 - acc: 0.6480 - val_loss: 0.7750 - val_acc: 0.6233
Epoch 10/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.65
70 - acc: 0.6518 - val_loss: 0.7340 - val_acc: 0.6328
Epoch 11/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.71
31 - acc: 0.6432 - val_loss: 0.7556 - val_acc: 0.6166
Epoch 12/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.66
11 - acc: 0.6495 - val_loss: 0.7423 - val_acc: 0.6172
Epoch 13/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.64
```

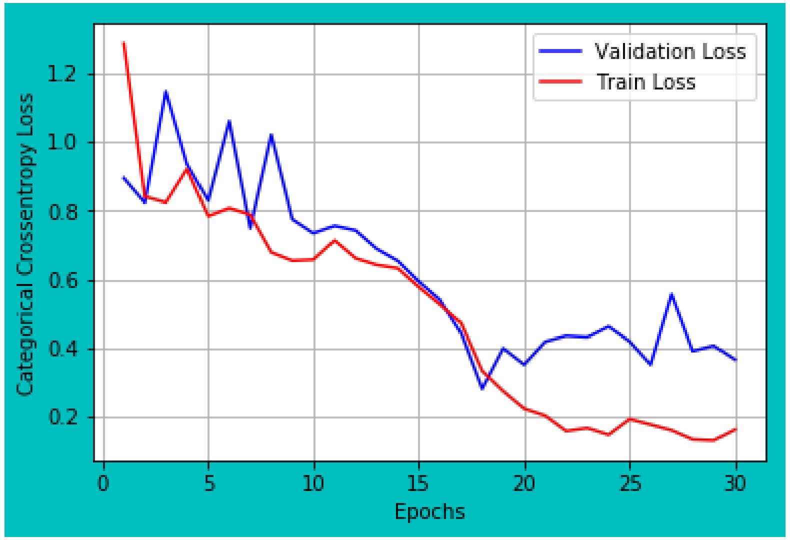
```
19 - acc: 0.6619 - val_loss: 0.6886 - val_acc: 0.6278
Epoch 14/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.63
21 - acc: 0.6634 - val_loss: 0.6538 - val_acc: 0.6261
Epoch 15/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.57
74 - acc: 0.6809 - val_loss: 0.5937 - val_acc: 0.6301
Epoch 16/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.52
68 - acc: 0.7296 - val_loss: 0.5404 - val_acc: 0.7774
Epoch 17/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.47
30 - acc: 0.8115 - val_loss: 0.4436 - val_acc: 0.8619
Epoch 18/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.33
36 - acc: 0.8942 - val_loss: 0.2814 - val_acc: 0.8972
Epoch 19/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.27
48 - acc: 0.9079 - val_loss: 0.3992 - val_acc: 0.8768
Epoch 20/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.22
37 - acc: 0.9316 - val_loss: 0.3516 - val_acc: 0.8931
Epoch 21/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.20
33 - acc: 0.9350 - val_loss: 0.4181 - val_acc: 0.8683
Epoch 22/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.15
83 - acc: 0.9478 - val_loss: 0.4353 - val_acc: 0.8911
Epoch 23/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.16
70 - acc: 0.9452 - val_loss: 0.4313 - val_acc: 0.9030
Epoch 24/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.14
74 - acc: 0.9489 - val_loss: 0.4626 - val_acc: 0.9013
Epoch 25/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.19
30 - acc: 0.9374 - val_loss: 0.4189 - val_acc: 0.8958
Epoch 26/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.17
```

```
71 - acc: 0.9445 - val_loss: 0.3515 - val_acc: 0.9019
Epoch 27/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.16
06 - acc: 0.9414 - val_loss: 0.5563 - val_acc: 0.8884
Epoch 28/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.13
41 - acc: 0.9499 - val_loss: 0.3909 - val_acc: 0.8999
Epoch 29/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.13
11 - acc: 0.9514 - val_loss: 0.4066 - val_acc: 0.8999
Epoch 30/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.16
24 - acc: 0.9429 - val_loss: 0.3663 - val_acc: 0.8928
```

In [39]:

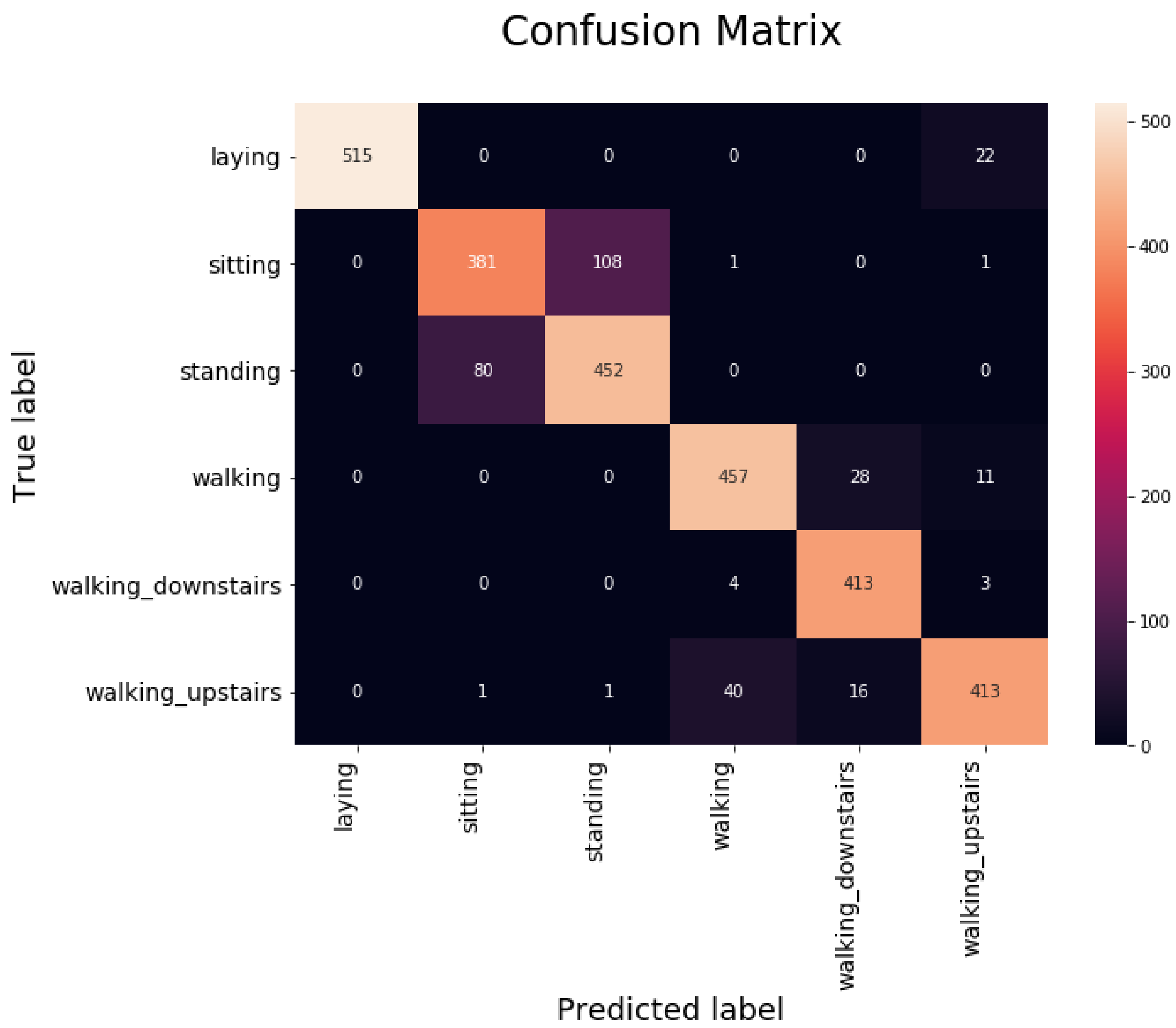
```
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores[0]))
test_acc8= scores[1]*100
train_acc8=(max(hist8.history['acc']))* 100
print("Train Accuracy: %f%%"% (train_acc8))

print("Test Accuracy: %f%%" % (test_acc8))
# error plot
vy=hist8.history['val_loss'] #validation loss
ty=hist8.history['loss'] # train loss
plt_dynamic(x, vy, ty)
```

```
Test Score: 0.366313
Train Accuracy: 95.144178%
Test Accuracy: 89.277231%
```



In [40]:

```
# 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.predi
ct(X_test), axis=1)])

# seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downsta
irs','walking_upstairs']
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), inde
x=class_names, columns=class_names )
fig = plt.figure(figsize=(10,7))
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")

#  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()
```



Observation

In [41]:
```
from prettytable import PrettyTable
models=['32LSTM+1layerLSTM +rmsprop_optimizer',
        '32LSTM+1layerLSTM +adam_optimizer',
```

```
        '64LSTM+1layerLSTM +rmsprop_optimizer',
        '64LSTM+1layerLSTM +adam_optimizer',
        '32LSTM+2layerLSTM +rmsprop_optimizer+0.65drop_out',
        '32LSTM+2layerLSTM +adam_optimizer+0.65drop_out',
    '64LSTM+2layerLSTM+adam_optimizer+0.65drop_out',
    '64LSTM+2layerLSTM+rmsprop_optimizer+0.65drop_out']
training_accuracy=[train_acc1,train_acc2,train_acc3,
                  train_acc4,train_acc5,train_acc6,train_acc7,
                  train_acc8]
test_accuracy=[test_acc1,test_acc2,test_acc3,test_acc4,
              test_acc5,test_acc6,test_acc7,test_acc8]
INDEX = [1,2,3,4,5,6,7,8]
# Initializing prettytable
Model_Performance = PrettyTable()
# Adding columns
Model_Performance.add_column("INDEX.",INDEX)
Model_Performance.add_column("MODEL_NAME",models)
Model_Performance.add_column("TRAINING ACCURACY",training_accuracy)
Model_Performance.add_column("TESTING ACCURACY",test_accuracy)
#Model_Performance.add_column("TEST SCORE",test_score)

# Printing the Model_Performance
print(Model_Performance)
```

```
+--------+--------------------------------------------------+---------
----------+------------------+
| INDEX. |                    MODEL_NAME                    | TRAINING
ACCURACY |  TESTING ACCURACY |
+--------+--------------------------------------------------+---------
----------+------------------+
|   1    |        32LSTM+1layerLSTM +rmsprop_optimizer       | 94.73612
622415669 | 90.77027485578554 |
|   2    |         32LSTM+1layerLSTM +adam_optimizer         | 93.10391
730141458 | 89.98982015609094 |
|   3    |        64LSTM+1layerLSTM +rmsprop_optimizer       | 95.19858
541893362 | 90.19341703427214 |
|   4    |         64LSTM+1layerLSTM +adam_optimizer         | 92.31501
632208922 | 89.27723108245674 |
|   5    | 32LSTM+2layerLSTM +rmsprop_optimizer+0.65drop_out | 94.54570
184983679 | 90.09161859518154 |
```

```
|   6    |   32LSTM+2layerLSTM +adam_optimizer+0.65drop_out  | 85.06528
835690969 | 86.63047166610112 |
|   7    |   64LSTM+2layerLSTM+adam_optimizer+0.65drop_out   | 95.30739
934711643 | 89.54869358669833 |
|   8    |  64LSTM+2layerLSTM+rmsprop_optimizer+0.65drop_out | 95.14417
845484222 | 89.27723108245674 |
+--------+---------------------------------------------------+---------
----------+------------------+
```

- adam optimizer's accuracy is less comparatively with rmsprop optimizer.
- When number of hidden layer incresed from 32 to 64 with 1layer of LSTM , Model's test accuracy is descreased .
- when Number of LSTM layers incresed , model is overfitting.