Assignment-23: 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 Signals/{signal}_{subset}.txt'
        signals_data.append(
            _read_csv(filename).as_matrix()
        )

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

In [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'
    y = _read_csv(filename)[0]

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

In [9]:

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

    return X_train, X_test, y_train, y_test
```

In [10]:

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

In [11]:

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

In [12]:

```
# Import Keras

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

In [13]:

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

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
```

- 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]:

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

In [19]:

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

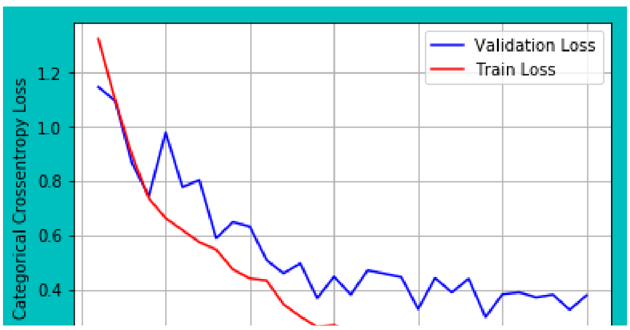
```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 24s 3ms/step - loss: 1.3247 - acc: 0.4308 - val_loss:
1.1474 - val_acc: 0.4808
Epoch 2/30
7352/7352 [==============================] - 24s 3ms/step - loss: 1.0995 - acc: 0.5196 - val_loss:
1.0965 - val_acc: 0.5236
Epoch 3/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.8968 - acc: 0.6092 - val_loss:
0.8670 - val_acc: 0.6312
Epoch 4/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.7366 - acc: 0.6532 - val_loss:
0.7453 - val_acc: 0.6125
Epoch 5/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.6636 - acc: 0.6768 - val_loss:
0.9799 - val_acc: 0.5989
Epoch 6/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.6197 - acc: 0.6989 - val_loss:
0.7784 - val_acc: 0.6498
Epoch 7/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.5752 - acc: 0.7444 - val_loss:
0.8042 - val_acc: 0.6820
Epoch 8/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.5479 - acc: 0.7636 - val_loss:
0.5897 - val_acc: 0.7438
Epoch 9/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.4742 - acc: 0.7856 - val_loss:
0.6495 - val_acc: 0.7258
Epoch 10/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.4419 - acc: 0.8069 - val_loss:
0.6326 - val_acc: 0.7788
Epoch 11/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.4336 - acc: 0.8388 - val_loss:
0.5086 - val_acc: 0.8554
Epoch 12/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.3461 - acc: 0.8980 - val_loss:
0.4601 - val_acc: 0.8605
Epoch 13/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.3012 - acc: 0.9075 - val_loss:
0.4970 - val_acc: 0.8504
Epoch 14/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2620 - acc: 0.9210 - val_loss:
0.3690 - val_acc: 0.8884
Epoch 15/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2688 - acc: 0.9223 - val_loss:
0.4479 - val_acc: 0.8744
Epoch 16/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2391 - acc: 0.9238 - val_loss:
0.3821 - val_acc: 0.9002
```

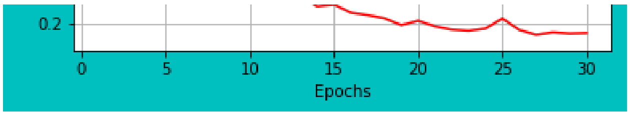
```
Epoch 17/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2299 - acc: 0.9280 - val_loss:
0.4713 - val_acc: 0.8856
Epoch 18/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2181 - acc: 0.9372 - val_loss:
0.4589 - val_acc: 0.8989
Epoch 19/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1929 - acc: 0.9377 - val_loss:
0.4467 - val_acc: 0.8992
Epoch 20/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2092 - acc: 0.9363 - val_loss:
0.3284 - val_acc: 0.8914
Epoch 21/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1884 - acc: 0.9400 - val_loss:
0.4429 - val_acc: 0.8921
Epoch 22/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1763 - acc: 0.9412 - val_loss:
0.3904 - val_acc: 0.9043
Epoch 23/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1721 - acc: 0.9403 - val_loss:
0.4405 - val_acc: 0.9050
Epoch 24/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1805 - acc: 0.9397 - val_loss:
0.2988 - val_acc: 0.9074
Epoch 25/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2175 - acc: 0.9402 - val_loss:
0.3831 - val_acc: 0.8996
Epoch 26/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1753 - acc: 0.9408 - val_loss:
0.3904 - val_acc: 0.8968
Epoch 27/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1573 - acc: 0.9412 - val_loss:
0.3716 - val_acc: 0.9033
Epoch 28/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1663 - acc: 0.9421 - val_loss:
0.3816 - val_acc: 0.9094
Epoch 29/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1619 - acc: 0.9434 - val_loss:
0.3260 - val_acc: 0.9135
Epoch 30/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1636 - acc: 0.9474 - val_loss:
0.3792 - val_acc: 0.9077
```

In [20]:

```python
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%
```





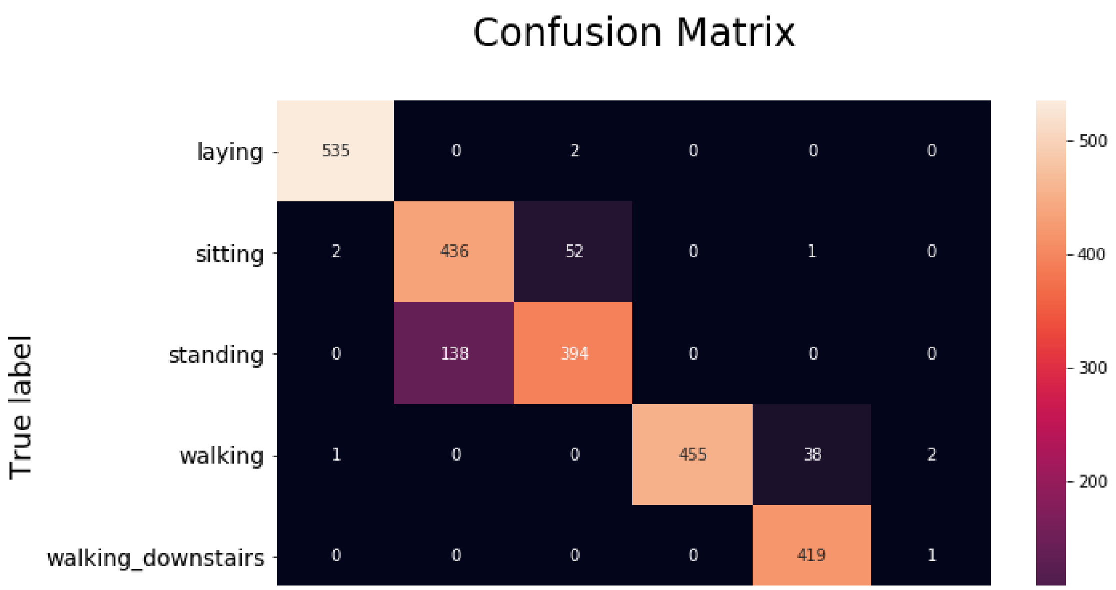
Observation:

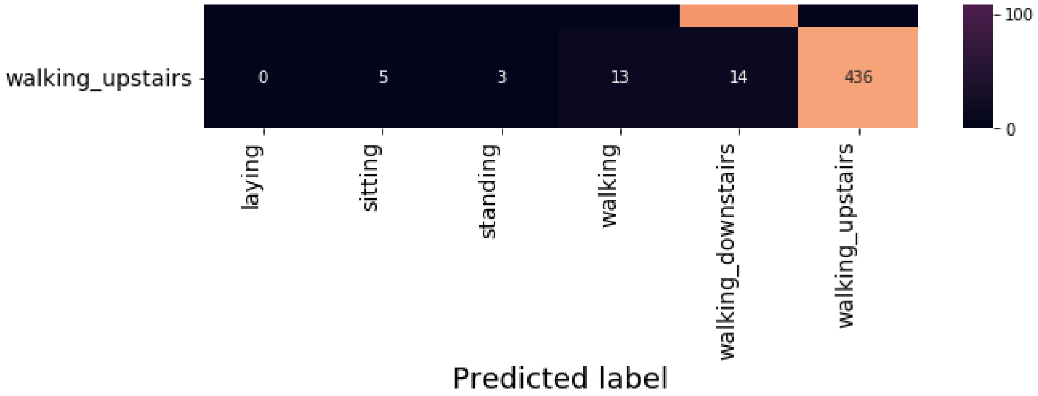
- From above plot,it can be diagnosied that model is performing overfitting.
- The training error graph is reducing continuously and Validation graph is descreasing upto inflection point and later it's increasing.

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.predict(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: 1.3700 - acc: 0.4207 - val_loss:
1.3605 - val_acc: 0.3858
Epoch 2/30
7352/7352 [==============================] - 24s 3ms/step - loss: 1.2399 - acc: 0.4460 - val_loss:
1.3068 - val_acc: 0.4150
Epoch 3/30
7352/7352 [==============================] - 24s 3ms/step - loss: 1.1828 - acc: 0.4551 - val_loss:
1.1227 - val_acc: 0.4645
Epoch 4/30
7352/7352 [==============================] - 24s 3ms/step - loss: 1.1022 - acc: 0.5275 - val_loss:
1.0100 - val_acc: 0.6016
Epoch 5/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.8570 - acc: 0.6140 - val_loss:
0.9916 - val_acc: 0.5836
Epoch 6/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.8873 - acc: 0.6019 - val_loss:
0.8917 - val_acc: 0.6281
Epoch 7/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.9034 - acc: 0.6064 - val_loss:
```

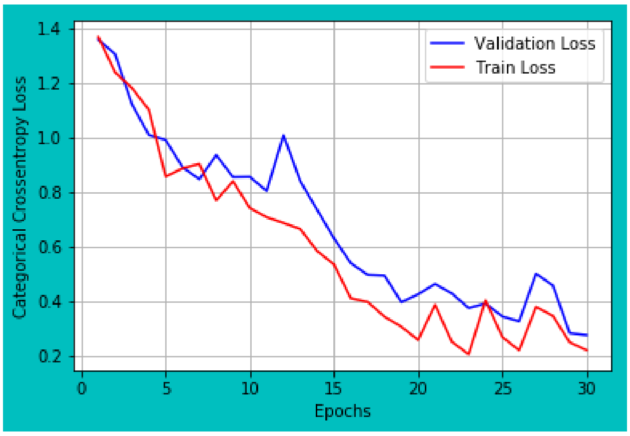
```
0.8474 - val_acc: 0.6135
Epoch 8/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.7693 - acc: 0.6442 - val_loss:
0.9360 - val_acc: 0.5786
Epoch 9/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.8396 - acc: 0.6164 - val_loss:
0.8555 - val_acc: 0.6250
Epoch 10/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.7415 - acc: 0.6522 - val_loss:
0.8564 - val_acc: 0.6298
Epoch 11/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.7076 - acc: 0.6581 - val_loss:
0.8041 - val_acc: 0.6454
Epoch 12/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.6867 - acc: 0.6712 - val_loss:
1.0092 - val_acc: 0.6118
Epoch 13/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.6644 - acc: 0.6999 - val_loss:
0.8403 - val_acc: 0.6960
Epoch 14/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.5832 - acc: 0.7692 - val_loss:
0.7345 - val_acc: 0.7723
Epoch 15/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.5339 - acc: 0.8195 - val_loss:
0.6299 - val_acc: 0.7825
Epoch 16/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.4096 - acc: 0.8643 - val_loss:
0.5387 - val_acc: 0.8409
Epoch 17/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.3963 - acc: 0.8713 - val_loss:
0.4963 - val_acc: 0.8514
Epoch 18/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.3419 - acc: 0.9038 - val_loss:
0.4934 - val_acc: 0.8690
Epoch 19/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.3055 - acc: 0.9011 - val_loss:
0.3953 - val_acc: 0.8816
Epoch 20/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.2563 - acc: 0.9172 - val_loss:
0.4246 - val_acc: 0.8918
Epoch 21/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.3852 - acc: 0.8768 - val_loss:
0.4623 - val_acc: 0.8595
Epoch 22/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2499 - acc: 0.9227 - val_loss:
0.4276 - val_acc: 0.8860
Epoch 23/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2039 - acc: 0.9310 - val_loss:
0.3738 - val_acc: 0.8863
Epoch 24/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.4023 - acc: 0.8913 - val_loss:
0.3902 - val_acc: 0.8904
Epoch 25/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2673 - acc: 0.9215 - val_loss:
0.3429 - val_acc: 0.8877
Epoch 26/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2183 - acc: 0.9305 - val_loss:
0.3252 - val_acc: 0.8968
Epoch 27/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.3781 - acc: 0.8656 - val_loss:
0.5003 - val_acc: 0.8300
Epoch 28/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.3450 - acc: 0.8724 - val_loss:
0.4568 - val_acc: 0.8836
Epoch 29/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.2477 - acc: 0.9135 - val_loss:
0.2817 - val_acc: 0.8856
Epoch 30/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2192 - 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)
```

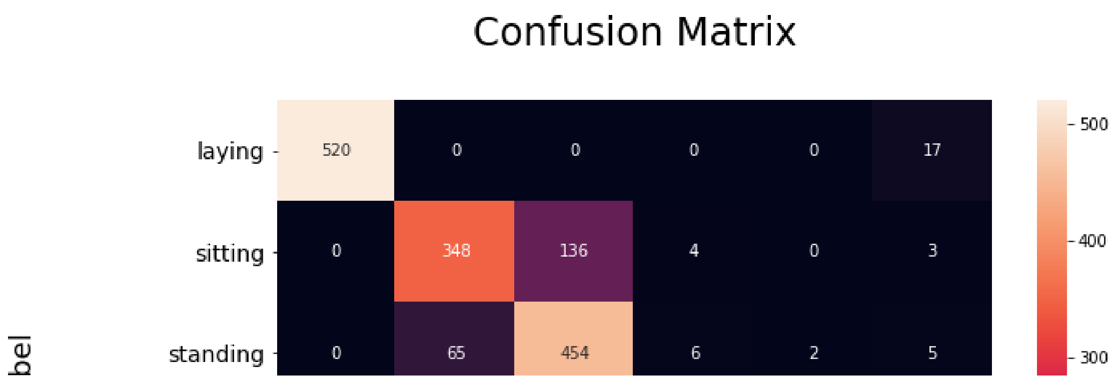
```
Test Score: 0.274399
Train Accuracy: 93.103917%
Test Accuracy: 89.989820%
```

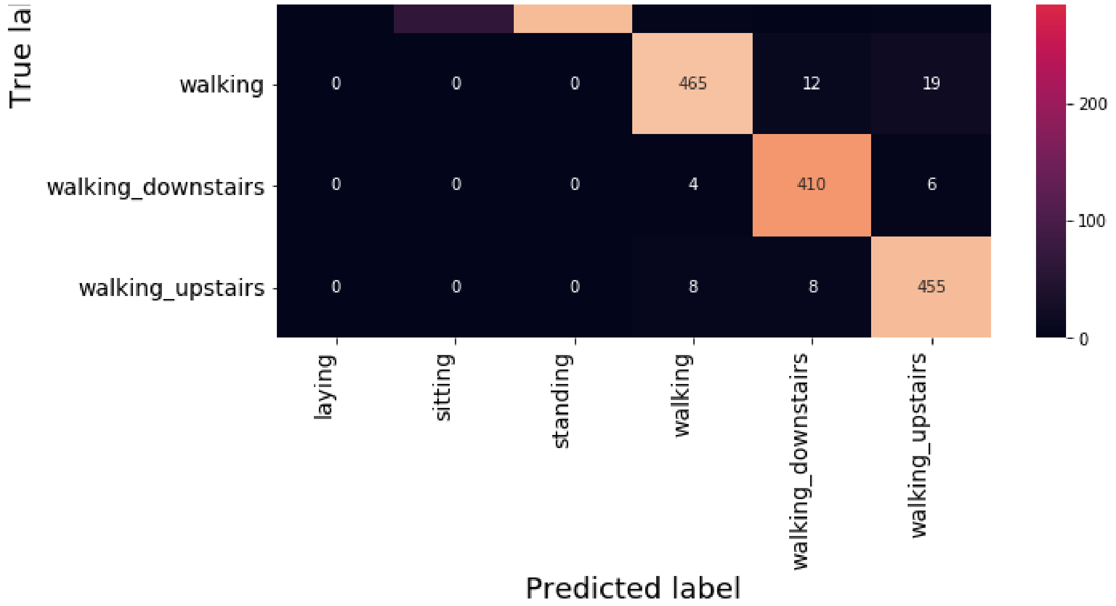


- Above model performs overfitting.

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.predict(X_test), axis=1)])

# 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")

#  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: 1.2827 - acc: 0.4340 - val_loss:
1.1831 - val_acc: 0.4574
Epoch 2/30
7352/7352 [==============================] - 30s 4ms/step - loss: 1.0405 - acc: 0.5345 - val_loss:
0.9734 - val_acc: 0.5667
Epoch 3/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.7448 - acc: 0.6507 - val_loss:
0.8002 - val_acc: 0.6759
Epoch 4/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.7964 - acc: 0.6745 - val_loss:
1.0634 - val_acc: 0.5660
```

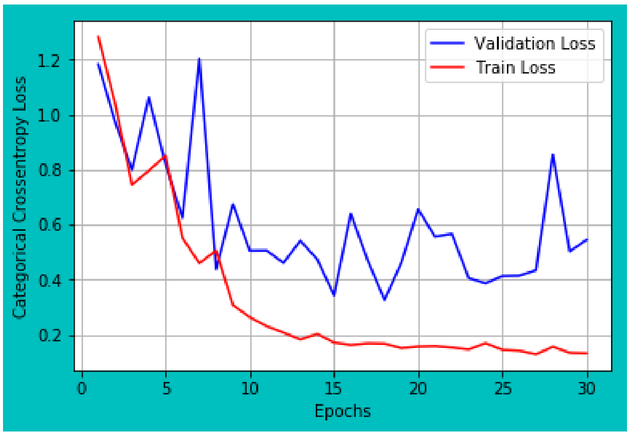
```
Epoch 5/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.8507 - acc: 0.6462 - val_loss:
0.8188 - val_acc: 0.6240
Epoch 6/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.5516 - acc: 0.7889 - val_loss:
0.6243 - val_acc: 0.7628
Epoch 7/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.4592 - acc: 0.8497 - val_loss:
1.2035 - val_acc: 0.6240
Epoch 8/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.5055 - acc: 0.8305 - val_loss:
0.4375 - val_acc: 0.8548
Epoch 9/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.3067 - acc: 0.9008 - val_loss:
0.6739 - val_acc: 0.8202
Epoch 10/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2629 - acc: 0.9162 - val_loss:
0.5049 - val_acc: 0.8694
Epoch 11/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2303 - acc: 0.9278 - val_loss:
0.5054 - val_acc: 0.8707
Epoch 12/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2069 - acc: 0.9302 - val_loss:
0.4603 - val_acc: 0.8768
Epoch 13/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1819 - acc: 0.9393 - val_loss:
0.5414 - val_acc: 0.8904
Epoch 14/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2020 - acc: 0.9344 - val_loss:
0.4737 - val_acc: 0.8795
Epoch 15/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1707 - acc: 0.9422 - val_loss:
0.3429 - val_acc: 0.9040
Epoch 16/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1617 - acc: 0.9433 - val_loss:
0.6396 - val_acc: 0.8622
Epoch 17/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1676 - acc: 0.9452 - val_loss:
0.4702 - val_acc: 0.8823
Epoch 18/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1664 - acc: 0.9437 - val_loss:
0.3252 - val_acc: 0.9060
Epoch 19/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1506 - acc: 0.9448 - val_loss:
0.4614 - val_acc: 0.8945
Epoch 20/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1561 - acc: 0.9460 - val_loss:
0.6557 - val_acc: 0.8870
Epoch 21/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1574 - acc: 0.9446 - val_loss:
0.5562 - val_acc: 0.8907
Epoch 22/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1527 - acc: 0.9478 - val_loss:
0.5662 - val_acc: 0.8928
Epoch 23/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1457 - acc: 0.9459 - val_loss:
0.4055 - val_acc: 0.9013
Epoch 24/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1682 - acc: 0.9437 - val_loss:
0.3864 - val_acc: 0.9043
Epoch 25/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1443 - acc: 0.9499 - val_loss:
0.4129 - val_acc: 0.9091
Epoch 26/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1408 - acc: 0.9513 - val_loss:
0.4141 - val_acc: 0.9030
Epoch 27/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1278 - acc: 0.9491 - val_loss:
0.4337 - val_acc: 0.8996
Epoch 28/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1555 - acc: 0.9494 - val_loss:
0.8541 - val_acc: 0.8711
Epoch 29/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1332 - acc: 0.9520 - val_loss:
0.5022 - val_acc: 0.9023
Epoch 30/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1315 - acc: 0.9493 - val_loss:
```

```
0.5443 - val_acc: 0.9019
```

In [26]:

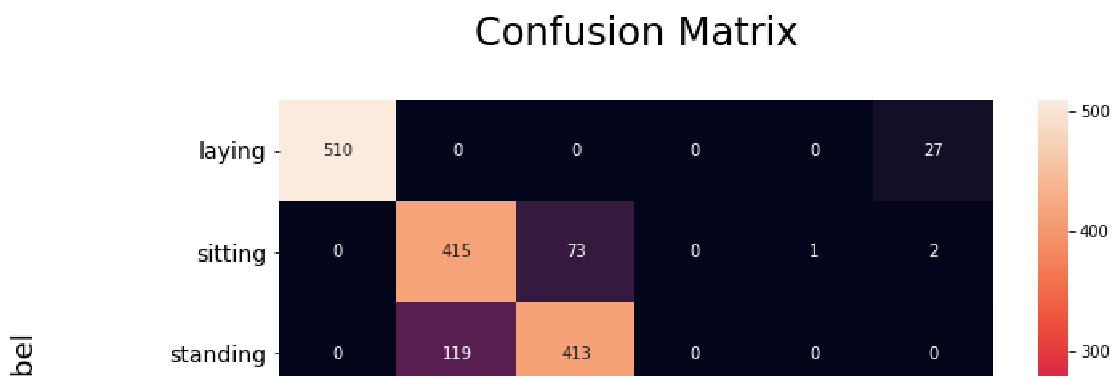
```
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores[0]))
test_acc3= scores[1]*100
train_acc3=(max(hist3.history['acc']))* 100
print("Train Accuracy: %f%%"% (train_acc3))

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)
```

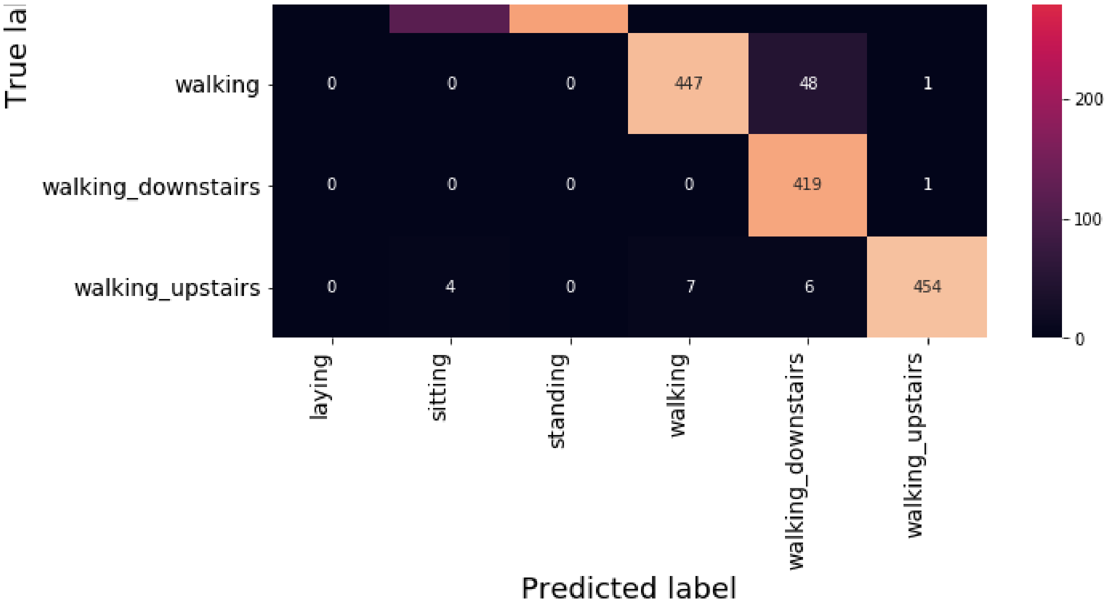
```
Test Score: 0.544287
Train Accuracy: 95.198585%
Test Accuracy: 90.193417%
```



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.predict(X_test), axis=1)])

# 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")

#  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: 1.3861 - acc: 0.3980 - val_loss: 1.3311 - val_acc: 0.3987
Epoch 2/30
7352/7352 [==============================] - 29s 4ms/step - loss: 1.2646 - acc: 0.4353 - val_loss: 1.3222 - val_acc: 0.4506
Epoch 3/30
7352/7352 [==============================] - 29s 4ms/step - loss: 1.3928 - acc: 0.3628 - val_loss: 1.3819 - val_acc: 0.3519
Epoch 4/30
7352/7352 [==============================] - 29s 4ms/step - loss: 1.3383 - acc: 0.3727 - val_loss: 1.3513 - val_acc: 0.3482
```

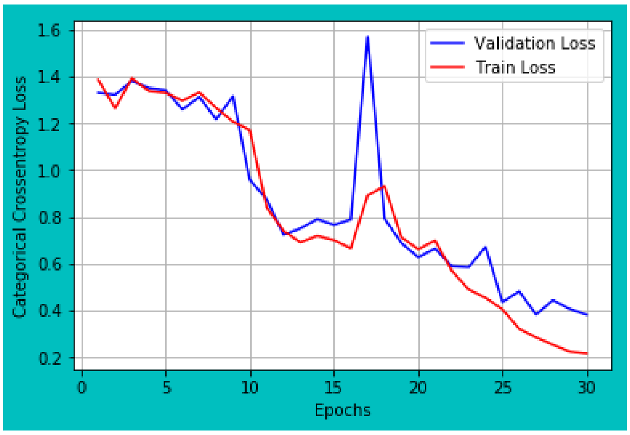
```
Epoch 5/30
7352/7352 [==============================] - 29s 4ms/step - loss: 1.3306 - acc: 0.3740 - val_loss:
1.3417 - val_acc: 0.3482
Epoch 6/30
7352/7352 [==============================] - 29s 4ms/step - loss: 1.2971 - acc: 0.3943 - val_loss:
1.2594 - val_acc: 0.4299
Epoch 7/30
7352/7352 [==============================] - 29s 4ms/step - loss: 1.3327 - acc: 0.3868 - val_loss:
1.3132 - val_acc: 0.4038
Epoch 8/30
7352/7352 [==============================] - 29s 4ms/step - loss: 1.2667 - acc: 0.4391 - val_loss:
1.2172 - val_acc: 0.4995
Epoch 9/30
7352/7352 [==============================] - 29s 4ms/step - loss: 1.2073 - acc: 0.4894 - val_loss:
1.3158 - val_acc: 0.3858
Epoch 10/30
7352/7352 [==============================] - 29s 4ms/step - loss: 1.1716 - acc: 0.4988 - val_loss:
0.9594 - val_acc: 0.5453
Epoch 11/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.8404 - acc: 0.6019 - val_loss:
0.8761 - val_acc: 0.5938
Epoch 12/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.7407 - acc: 0.6356 - val_loss:
0.7237 - val_acc: 0.6342
Epoch 13/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.6914 - acc: 0.6619 - val_loss:
0.7510 - val_acc: 0.6098
Epoch 14/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.7192 - acc: 0.6585 - val_loss:
0.7900 - val_acc: 0.6149
Epoch 15/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.7008 - acc: 0.6564 - val_loss:
0.7663 - val_acc: 0.6518
Epoch 16/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.6659 - acc: 0.6955 - val_loss:
0.7885 - val_acc: 0.7048
Epoch 17/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.8925 - acc: 0.5839 - val_loss:
1.5695 - val_acc: 0.2945
Epoch 18/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.9318 - acc: 0.5718 - val_loss:
0.7928 - val_acc: 0.6325
Epoch 19/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.7132 - acc: 0.6574 - val_loss:
0.6886 - val_acc: 0.6814
Epoch 20/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.6619 - acc: 0.6903 - val_loss:
0.6277 - val_acc: 0.7139
Epoch 21/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.6992 - acc: 0.7240 - val_loss:
0.6648 - val_acc: 0.7126
Epoch 22/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.5699 - acc: 0.7561 - val_loss:
0.5894 - val_acc: 0.7570
Epoch 23/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.4902 - acc: 0.7930 - val_loss:
0.5857 - val_acc: 0.7706
Epoch 24/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.4547 - acc: 0.8192 - val_loss:
0.6714 - val_acc: 0.7258
Epoch 25/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.4055 - acc: 0.8542 - val_loss:
0.4363 - val_acc: 0.8364
Epoch 26/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.3216 - acc: 0.8862 - val_loss:
0.4823 - val_acc: 0.8490
Epoch 27/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2854 - acc: 0.9032 - val_loss:
0.3832 - val_acc: 0.8782
Epoch 28/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2542 - acc: 0.9121 - val_loss:
0.4442 - val_acc: 0.8524
Epoch 29/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2243 - acc: 0.9232 - val_loss:
0.4070 - val_acc: 0.8660
Epoch 30/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2165 - acc: 0.9208 - val_loss:
```

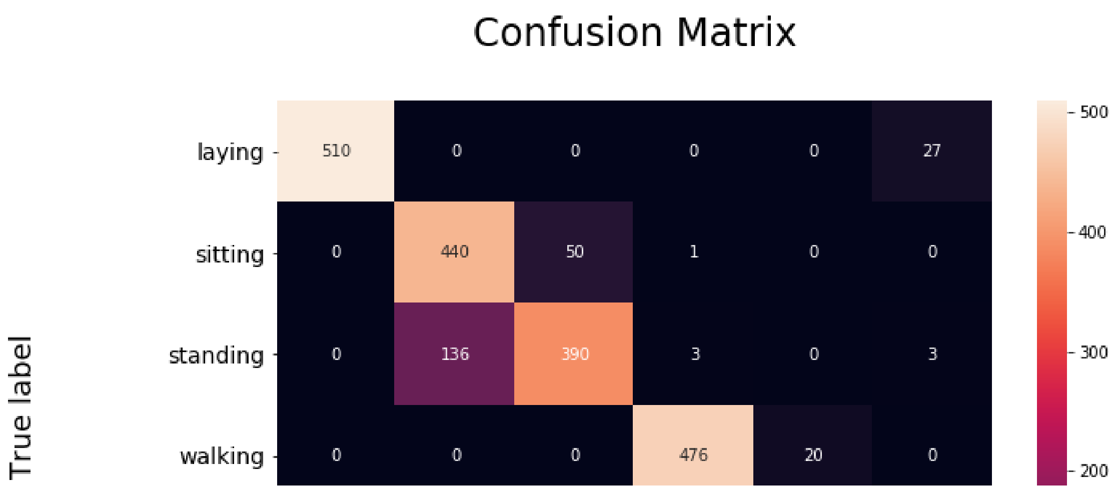
```
0.3825 - val_acc: 0.8928
```

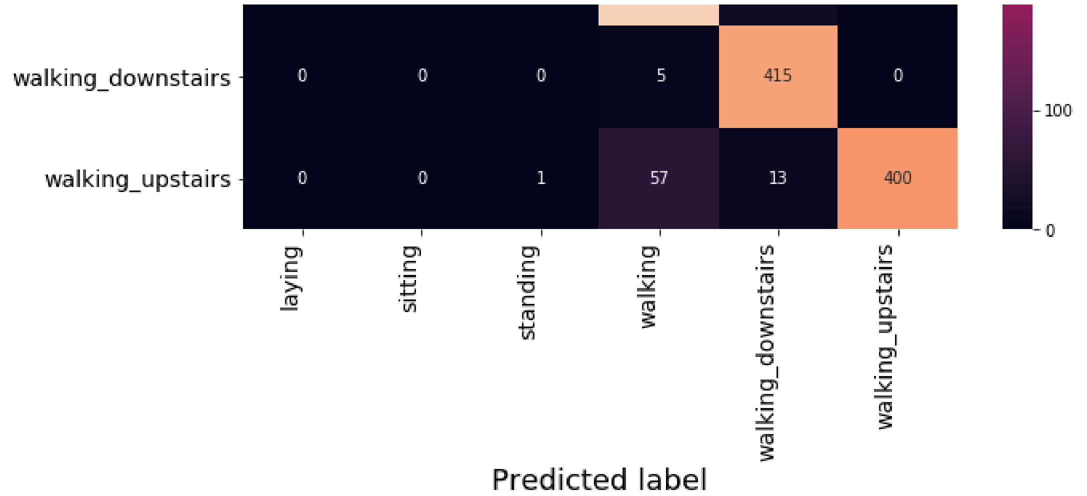
In [29]:

```python
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.predict(X_test), axis=1)])

# 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")

#  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%
```







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
7352/7352 [==============================] - 54s 7ms/step - loss: 1.3149 - acc: 0.4533 - val_loss:
1.0325 - val_acc: 0.4913
Epoch 2/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.9204 - acc: 0.5929 - val_loss:
0.8322 - val_acc: 0.6108
Epoch 3/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.8085 - acc: 0.6246 - val_loss:
```

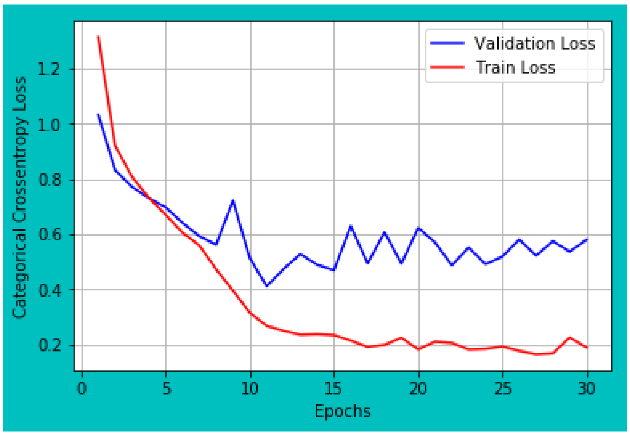
```
0.7718 - val_acc: 0.6030
Epoch 4/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.7318 - acc: 0.6541 - val_loss:
0.7303 - val_acc: 0.6420
Epoch 5/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.6696 - acc: 0.6748 - val_loss:
0.6972 - val_acc: 0.6641
Epoch 6/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.6052 - acc: 0.7114 - val_loss:
0.6408 - val_acc: 0.7160
Epoch 7/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.5585 - acc: 0.7816 - val_loss:
0.5922 - val_acc: 0.7435
Epoch 8/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.4723 - acc: 0.8357 - val_loss:
0.5611 - val_acc: 0.8483
Epoch 9/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.3951 - acc: 0.8814 - val_loss:
0.7218 - val_acc: 0.8229
Epoch 10/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.3148 - acc: 0.9072 - val_loss:
0.5134 - val_acc: 0.8758
Epoch 11/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.2669 - acc: 0.9223 - val_loss:
0.4106 - val_acc: 0.9009
Epoch 12/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.2489 - acc: 0.9286 - val_loss:
0.4730 - val_acc: 0.8901
Epoch 13/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.2347 - acc: 0.9359 - val_loss:
0.5271 - val_acc: 0.8863
Epoch 14/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.2367 - acc: 0.9342 - val_loss:
0.4882 - val_acc: 0.9023
Epoch 15/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.2332 - acc: 0.9343 - val_loss:
0.4688 - val_acc: 0.9036
Epoch 16/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.2135 - acc: 0.9372 - val_loss:
0.6286 - val_acc: 0.8911
Epoch 17/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.1901 - acc: 0.9399 - val_loss:
0.4936 - val_acc: 0.9074
Epoch 18/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.1978 - acc: 0.9388 - val_loss:
0.6069 - val_acc: 0.8901
Epoch 19/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.2236 - acc: 0.9344 - val_loss:
0.4921 - val_acc: 0.9141
Epoch 20/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.1813 - acc: 0.9382 - val_loss:
0.6226 - val_acc: 0.8880
Epoch 21/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.2096 - acc: 0.9358 - val_loss:
0.5694 - val_acc: 0.9074
Epoch 22/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.2049 - acc: 0.9363 - val_loss:
0.4863 - val_acc: 0.9067
Epoch 23/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.1810 - acc: 0.9415 - val_loss:
0.5510 - val_acc: 0.9023
Epoch 24/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.1833 - acc: 0.9392 - val_loss:
0.4904 - val_acc: 0.9104
Epoch 25/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.1917 - acc: 0.9419 - val_loss:
0.5181 - val_acc: 0.9009
Epoch 26/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.1761 - acc: 0.9421 - val_loss:
0.5803 - val_acc: 0.9213
Epoch 27/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.1637 - acc: 0.9455 - val_loss:
0.5213 - val_acc: 0.9050
Epoch 28/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.1671 - acc: 0.9418 - val_loss:
0.5739 - val_acc: 0.9128
Epoch 29/30
```

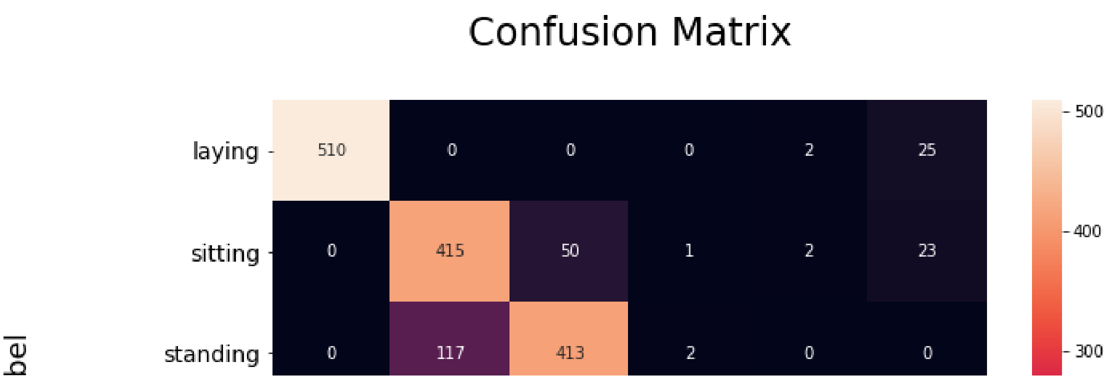
```
7352/7352 [==============================] - 54s 7ms/step - loss: 0.2246 - acc: 0.9381 - val_loss:
0.5360 - val_acc: 0.9101
Epoch 30/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.1883 - acc: 0.9402 - val_loss:
0.5793 - val_acc: 0.9009
```

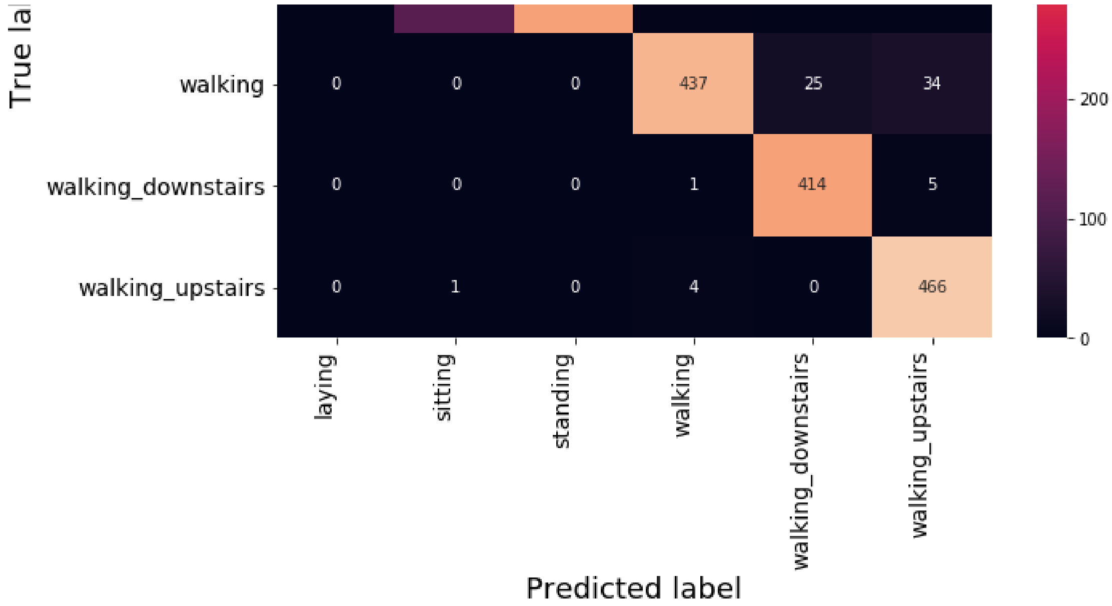
In [31]:

```python
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.predict(X_test), axis=1)])

# 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")

#  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: 1.3817 - acc: 0.4410 - val_loss:
1.3614 - val_acc: 0.3651
Epoch 2/30
```

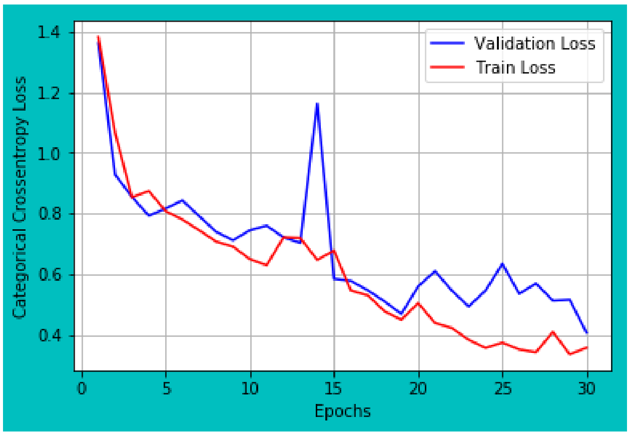
```
7352/7352 [==============================] - 55s 7ms/step - loss: 1.0656 - acc: 0.5241 - val_loss:
0.9279 - val_acc: 0.5803
Epoch 3/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.8534 - acc: 0.5861 - val_loss:
0.8546 - val_acc: 0.5405
Epoch 4/30
7352/7352 [==============================] - 55s 7ms/step - loss: 0.8739 - acc: 0.5762 - val_loss:
0.7923 - val_acc: 0.6067
Epoch 5/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.8074 - acc: 0.5754 - val_loss:
0.8162 - val_acc: 0.5792
Epoch 6/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.7797 - acc: 0.5896 - val_loss:
0.8430 - val_acc: 0.6135
Epoch 7/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.7443 - acc: 0.6247 - val_loss:
0.7904 - val_acc: 0.6115
Epoch 8/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.7071 - acc: 0.6469 - val_loss:
0.7394 - val_acc: 0.6250
Epoch 9/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.6906 - acc: 0.6620 - val_loss:
0.7115 - val_acc: 0.6233
Epoch 10/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.6493 - acc: 0.6789 - val_loss:
0.7445 - val_acc: 0.6227
Epoch 11/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.6291 - acc: 0.6971 - val_loss:
0.7597 - val_acc: 0.6403
Epoch 12/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.7211 - acc: 0.6330 - val_loss:
0.7211 - val_acc: 0.6203
Epoch 13/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.7190 - acc: 0.6421 - val_loss:
0.7027 - val_acc: 0.6291
Epoch 14/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.6469 - acc: 0.6729 - val_loss:
1.1626 - val_acc: 0.4822
Epoch 15/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.6771 - acc: 0.6659 - val_loss:
0.5845 - val_acc: 0.6332
Epoch 16/30
7352/7352 [==============================] - 55s 7ms/step - loss: 0.5461 - acc: 0.7035 - val_loss:
0.5777 - val_acc: 0.6315
Epoch 17/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.5301 - acc: 0.7149 - val_loss:
0.5467 - val_acc: 0.7499
Epoch 18/30
7352/7352 [==============================] - 55s 7ms/step - loss: 0.4774 - acc: 0.7743 - val_loss:
0.5099 - val_acc: 0.7581
Epoch 19/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.4494 - acc: 0.7881 - val_loss:
0.4685 - val_acc: 0.7391
Epoch 20/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.5042 - acc: 0.7654 - val_loss:
0.5593 - val_acc: 0.7394
Epoch 21/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.4389 - acc: 0.7833 - val_loss:
0.6096 - val_acc: 0.7520
Epoch 22/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.4223 - acc: 0.7911 - val_loss:
0.5456 - val_acc: 0.7513
Epoch 23/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.3832 - acc: 0.8074 - val_loss:
0.4929 - val_acc: 0.7845
Epoch 24/30
7352/7352 [==============================] - 53s 7ms/step - loss: 0.3567 - acc: 0.8104 - val_loss:
0.5456 - val_acc: 0.7608
Epoch 25/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.3738 - acc: 0.8138 - val_loss:
0.6342 - val_acc: 0.7679
Epoch 26/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.3516 - acc: 0.8160 - val_loss:
0.5351 - val_acc: 0.7621
Epoch 27/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.3423 - acc: 0.8305 - val_loss:
0.5691 - val_acc: 0.7642
```

```
Epoch 28/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.4092 - acc: 0.8251 - val_loss:
0.5124 - val_acc: 0.8385
Epoch 29/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.3355 - acc: 0.8434 - val_loss:
0.5154 - val_acc: 0.8690
Epoch 30/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.3573 - acc: 0.8507 - val_loss:
0.4068 - val_acc: 0.8663
```

In [33]:

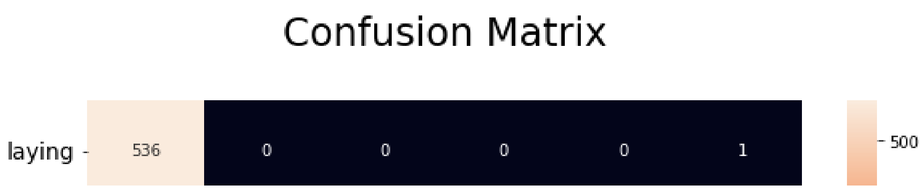
```
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores[0]))
test_acc6= scores[1]*100
train_acc6=(max(hist6.history['acc']))* 100
print("Train Accuracy: %f%%"% (train_acc6))

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)
```

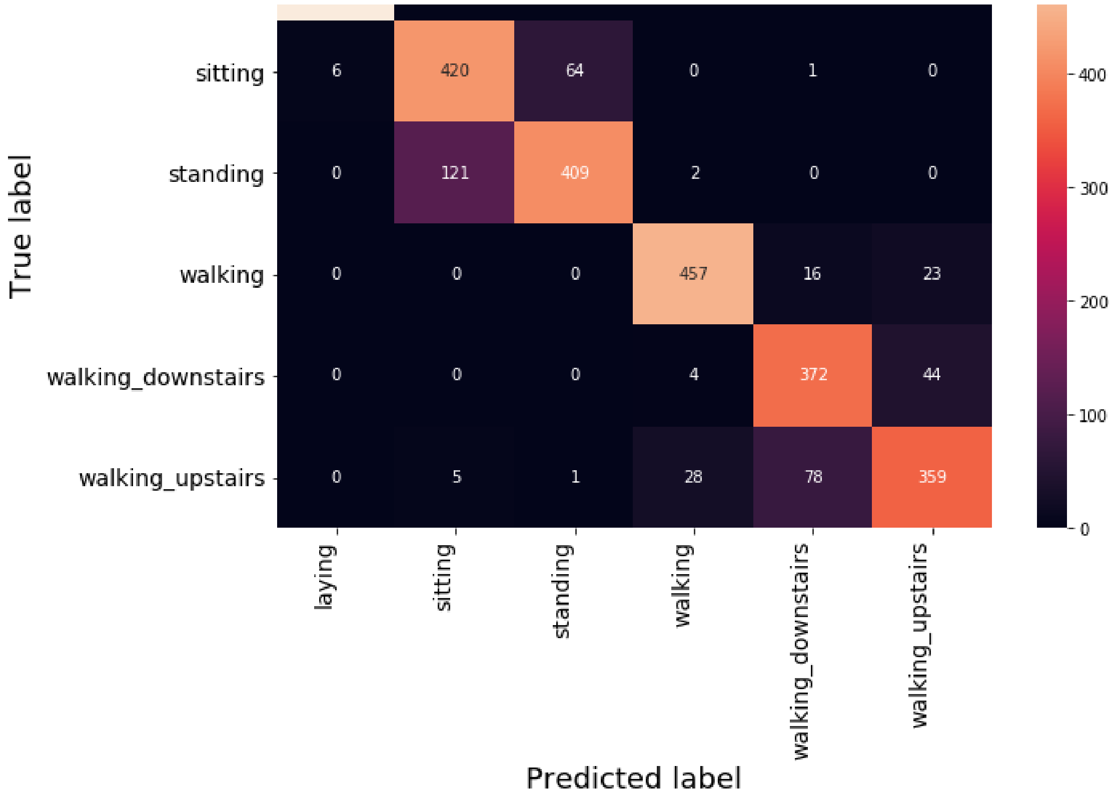
```
Test Score: 0.406773
Train Accuracy: 85.065288%
Test Accuracy: 86.630472%
```



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.predict(X_test), axis=1)])

# 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")

#  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()
# 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
```

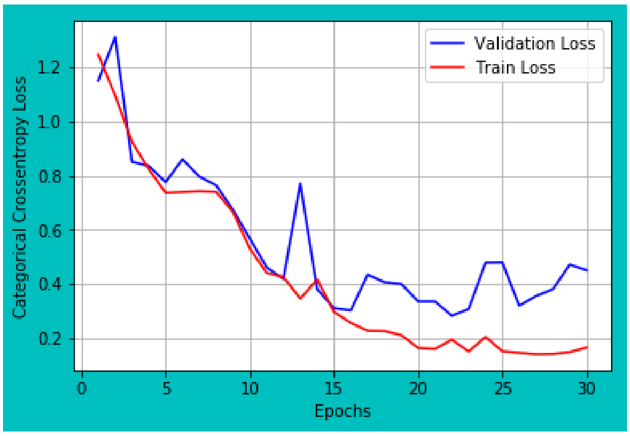
```
Non-trainable params: 0
_________________________________________________________________
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 75s 10ms/step - loss: 1.2460 - acc: 0.4650 - val_loss
: 1.1507 - val_acc: 0.4608
Epoch 2/30
7352/7352 [==============================] - 72s 10ms/step - loss: 1.0960 - acc: 0.5037 - val_loss
: 1.3114 - val_acc: 0.5053
Epoch 3/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.9248 - acc: 0.5747 - val_loss
: 0.8509 - val_acc: 0.5935
Epoch 4/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.8230 - acc: 0.5872 - val_loss
: 0.8356 - val_acc: 0.5921
Epoch 5/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.7364 - acc: 0.6193 - val_loss
: 0.7757 - val_acc: 0.6359
Epoch 6/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.7391 - acc: 0.6291 - val_loss
: 0.8603 - val_acc: 0.5042
Epoch 7/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.7422 - acc: 0.6066 - val_loss
: 0.7962 - val_acc: 0.6037
Epoch 8/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.7399 - acc: 0.6348 - val_loss
: 0.7643 - val_acc: 0.6108
Epoch 9/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.6658 - acc: 0.6737 - val_loss
: 0.6736 - val_acc: 0.6335
Epoch 10/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.5310 - acc: 0.7695 - val_loss
: 0.5680 - val_acc: 0.7923
Epoch 11/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.4409 - acc: 0.8456 - val_loss
: 0.4612 - val_acc: 0.8595
Epoch 12/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.4254 - acc: 0.8546 - val_loss
: 0.4187 - val_acc: 0.8728
Epoch 13/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.3459 - acc: 0.8916 - val_loss
: 0.7711 - val_acc: 0.6875
Epoch 14/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.4164 - acc: 0.8493 - val_loss
: 0.3807 - val_acc: 0.8890
Epoch 15/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.2967 - acc: 0.9066 - val_loss
: 0.3106 - val_acc: 0.8911
Epoch 16/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.2565 - acc: 0.9098 - val_loss
: 0.3027 - val_acc: 0.8975
Epoch 17/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.2276 - acc: 0.9264 - val_loss
: 0.4344 - val_acc: 0.8843
Epoch 18/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.2262 - acc: 0.9328 - val_loss
: 0.4064 - val_acc: 0.8884
Epoch 19/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.2113 - acc: 0.9295 - val_loss
: 0.4000 - val_acc: 0.9067
Epoch 20/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.1637 - acc: 0.9463 - val_loss
: 0.3357 - val_acc: 0.9067
Epoch 21/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1605 - acc: 0.9418 - val_loss
: 0.3353 - val_acc: 0.9053
Epoch 22/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1943 - acc: 0.9369 - val_loss
: 0.2827 - val_acc: 0.9155
Epoch 23/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1510 - acc: 0.9484 - val_loss
: 0.3076 - val_acc: 0.9169
Epoch 24/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.2044 - acc: 0.9301 - val_loss
: 0.4783 - val_acc: 0.8856
Epoch 25/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1516 - acc: 0.9440 - val_loss
```

```
: 0.4793 - val_acc: 0.9033
Epoch 26/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1460 - acc: 0.9482 - val_loss
: 0.3199 - val_acc: 0.9169
Epoch 27/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.1405 - acc: 0.9531 - val_loss
: 0.3554 - val_acc: 0.9223
Epoch 28/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.1414 - acc: 0.9464 - val_loss
: 0.3807 - val_acc: 0.9141
Epoch 29/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.1487 - acc: 0.9489 - val_loss
: 0.4713 - val_acc: 0.8982
Epoch 30/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.1654 - acc: 0.9372 - val_loss
: 0.4509 - val_acc: 0.8955
```

In [36]:

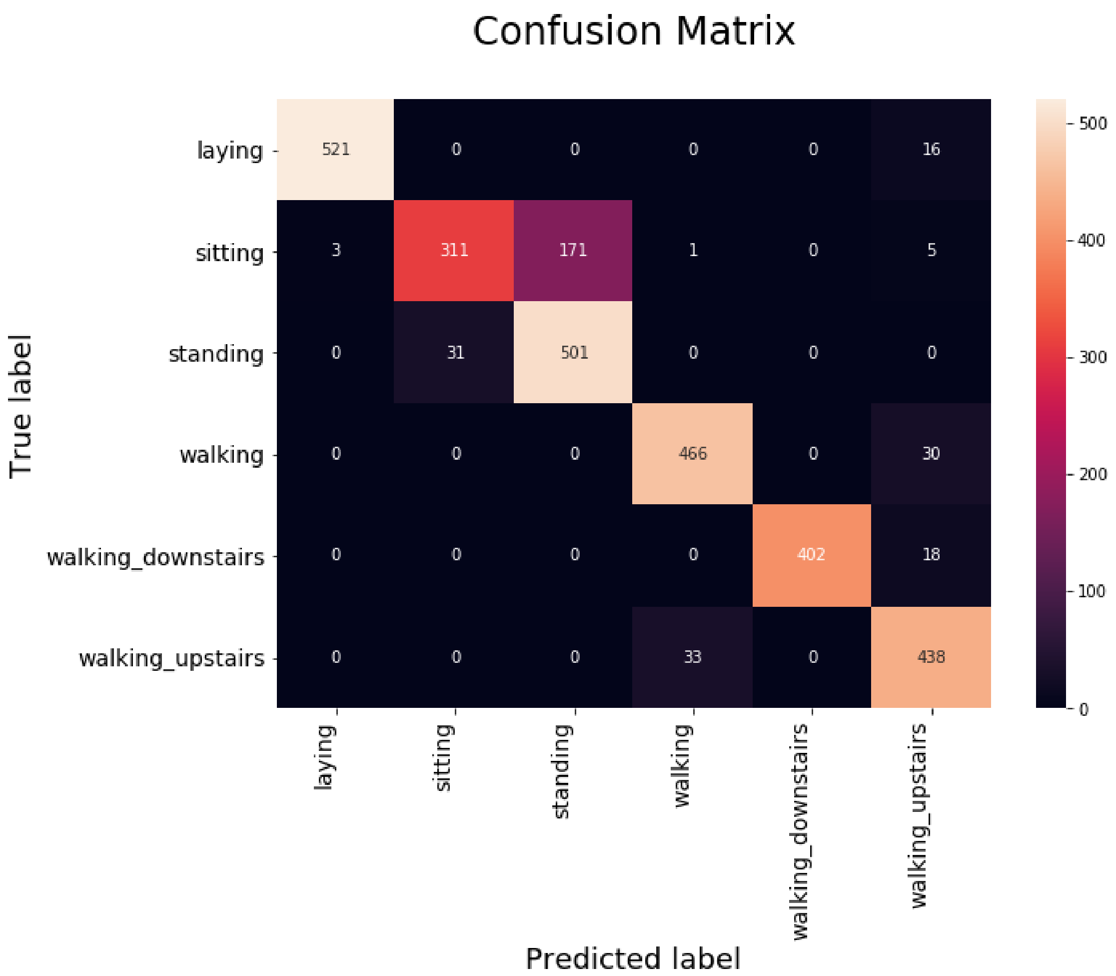
```
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores[0]))
test_acc7= scores[1]*100
train_acc7=(max(hist7.history['acc']))* 100
print("Train Accuracy: %f%%"% (train_acc7))

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.predict(X_test), axis=1)])

# 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")

#  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()
# 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
```

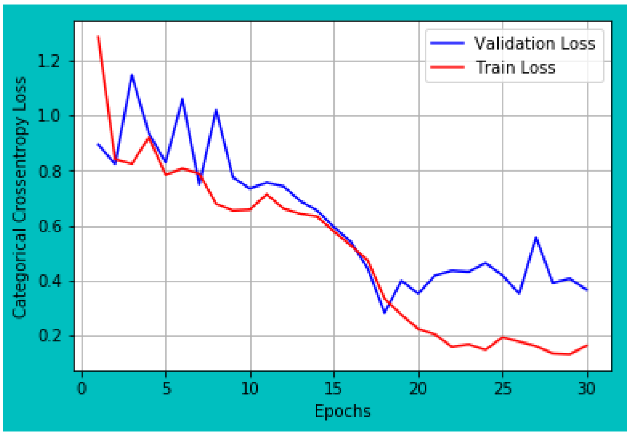
```
_________________________________________________________________
lstm_12 (LSTM)               (None, 64)                33024
_________________________________________________________________
dropout_12 (Dropout)         (None, 64)                0
_________________________________________________________________
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: 1.2849 - acc: 0.4374 - val_loss
: 0.8938 - val_acc: 0.5351
Epoch 2/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.8402 - acc: 0.5975 - val_loss
: 0.8225 - val_acc: 0.6016
Epoch 3/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.8233 - acc: 0.6019 - val_loss
: 1.1463 - val_acc: 0.4937
Epoch 4/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.9199 - acc: 0.5400 - val_loss
: 0.9346 - val_acc: 0.5402
Epoch 5/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.7838 - acc: 0.5828 - val_loss
: 0.8286 - val_acc: 0.5592
Epoch 6/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.8073 - acc: 0.5964 - val_loss
: 1.0598 - val_acc: 0.5083
Epoch 7/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.7881 - acc: 0.6172 - val_loss
: 0.7484 - val_acc: 0.6013
Epoch 8/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.6785 - acc: 0.6458 - val_loss
: 1.0201 - val_acc: 0.5606
Epoch 9/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.6542 - acc: 0.6480 - val_loss
: 0.7750 - val_acc: 0.6233
Epoch 10/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.6570 - acc: 0.6518 - val_loss
: 0.7340 - val_acc: 0.6328
Epoch 11/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.7131 - acc: 0.6432 - val_loss
: 0.7556 - val_acc: 0.6166
Epoch 12/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.6611 - acc: 0.6495 - val_loss
: 0.7423 - val_acc: 0.6172
Epoch 13/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.6419 - acc: 0.6619 - val_loss
: 0.6886 - val_acc: 0.6278
Epoch 14/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.6321 - acc: 0.6634 - val_loss
: 0.6538 - val_acc: 0.6261
Epoch 15/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.5774 - acc: 0.6809 - val_loss
: 0.5937 - val_acc: 0.6301
Epoch 16/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.5268 - acc: 0.7296 - val_loss
: 0.5404 - val_acc: 0.7774
Epoch 17/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.4730 - acc: 0.8115 - val_loss
: 0.4436 - val_acc: 0.8619
Epoch 18/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.3336 - acc: 0.8942 - val_loss
: 0.2814 - val_acc: 0.8972
Epoch 19/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.2748 - acc: 0.9079 - val_loss
: 0.3992 - val_acc: 0.8768
Epoch 20/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.2237 - acc: 0.9316 - val_loss
: 0.3516 - val_acc: 0.8931
Epoch 21/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.2033 - acc: 0.9350 - val_loss
: 0.4181 - val_acc: 0.8683
Epoch 22/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.1583 - acc: 0.9478 - val_loss
: 0.4353 - val_acc: 0.8911
```

```
: 0.4555 - val_acc: 0.8911
Epoch 23/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1670 - acc: 0.9452 - val_loss
: 0.4313 - val_acc: 0.9030
Epoch 24/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1474 - acc: 0.9489 - val_loss
: 0.4626 - val_acc: 0.9013
Epoch 25/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1930 - acc: 0.9374 - val_loss
: 0.4189 - val_acc: 0.8958
Epoch 26/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1771 - acc: 0.9445 - val_loss
: 0.3515 - val_acc: 0.9019
Epoch 27/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1606 - acc: 0.9414 - val_loss
: 0.5563 - val_acc: 0.8884
Epoch 28/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.1341 - acc: 0.9499 - val_loss
: 0.3909 - val_acc: 0.8999
Epoch 29/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.1311 - acc: 0.9514 - val_loss
: 0.4066 - val_acc: 0.8999
Epoch 30/30
7352/7352 [==============================] - 76s 10ms/step - loss: 0.1624 - 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)
```

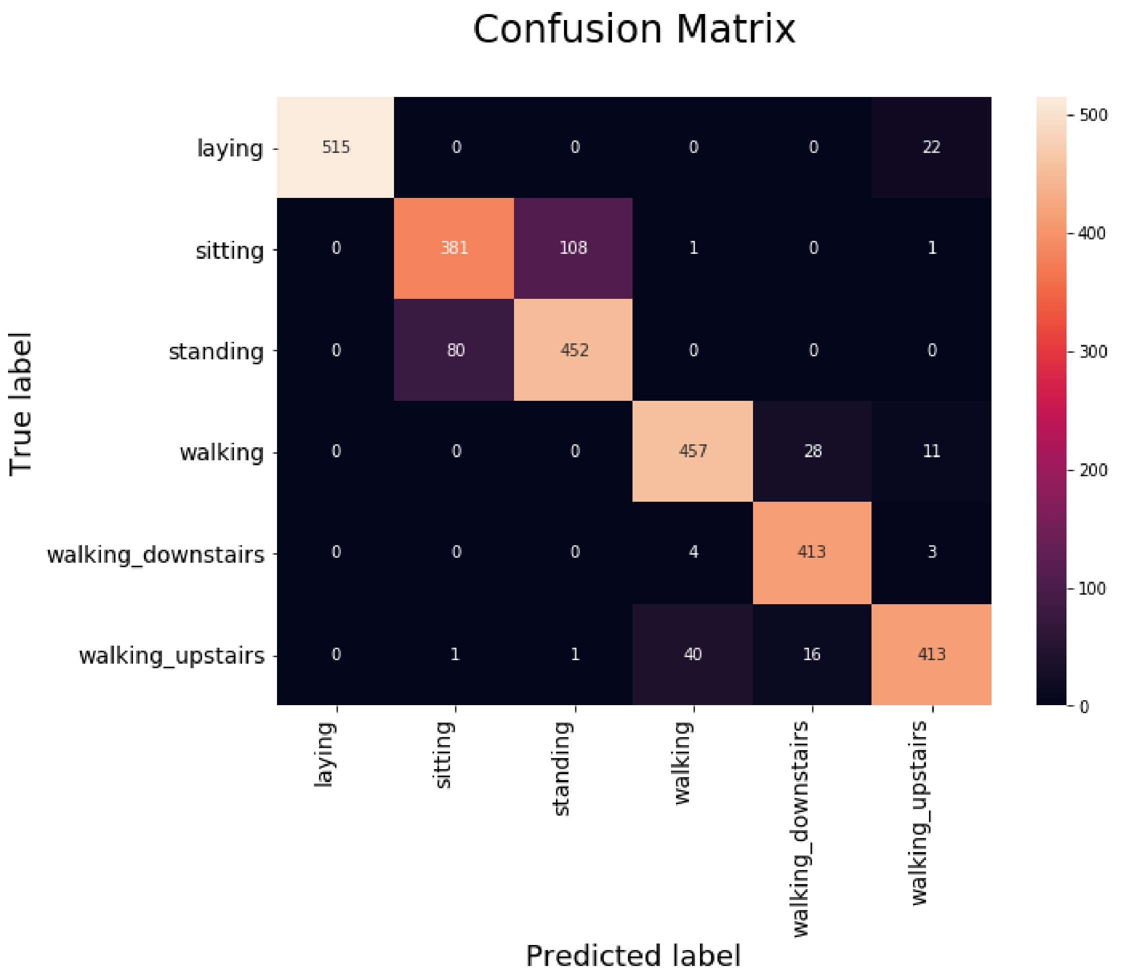
```
Test Score: 0.366313
Train Accuracy: 95.144178%
Test Accuracy: 89.277231%
```



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.predict(X_test), axis=1)])

# 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")

#  heatmap
heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(),
```

```
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 ACCURAC
|
+---------+----------------------------------------------------+-------------------+-----------------
|    1    |         32LSTM+1layerLSTM +rmsprop_optimizer       | 94.73612622415669 |
90.77027485578554 |
|    2    |          32LSTM+1layerLSTM +adam_optimizer         | 93.10391730141458 |
89.98982015609094 |
|    3    |         64LSTM+1layerLSTM +rmsprop_optimizer       | 95.19858541893362 |
90.19341703427214 |
|    4    |          64LSTM+1layerLSTM +adam_optimizer         | 92.31501632208922 |
89.27723108245674 |
|    5    | 32LSTM+2layerLSTM +rmsprop_optimizer+0.65drop_out | 94.54570184983679 |
90.09161859518154 |
|    6    |   32LSTM+2layerLSTM +adam_optimizer+0.65drop_out  | 85.06528835690969 | 86.6304716661011
2 |
|    7    |   64LSTM+2layerLSTM+adam_optimizer+0.65drop_out   | 95.30739934711643 | 89.5486935866983
3 |
|    8    |  64LSTM+2layerLSTM+rmsprop_optimizer+0.65drop_out | 95.14417845484222 |
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.