Assignment: Human activity detection

OBJECTIVE - Perform hyperparameter tuning using various models based on following:

1. **Instead of 32 LSTM, use even deeper LSTM, say-64**
2. **Tune dropout rate**
3. **Two LSTM layers and larger dropouts**

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 = '/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'/input/uci_har_dataset/{subset}/Inertial Signals/{signal}_{sub
        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'/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 [ ]:

```
```

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
```

HYPERPARAMETER TUNING

Here we will do hypertuning with following architectures:

1. 32LSTM+1layerLSTM +rmsprop_optimizer
2. 32LSTM+1layerLSTM +adam_optimizer

3. 64LSTM+1layerLSTM +rmsprop_optimizer
4. 64LSTM+1layerLSTM +adam_optimizer
5. 32LSTM+2layerLSTM +rmsprop_optimizer+0.65drop_out
6. 32LSTM+2layerLSTM +adam_optimizer+0.65drop_out
7. 64LSTM+2layerLSTM+adam_optimizer+0.65drop_out
8. 64LSTM+2layerLSTM+rmsprop_optimizer+0.65drop_out

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)
```

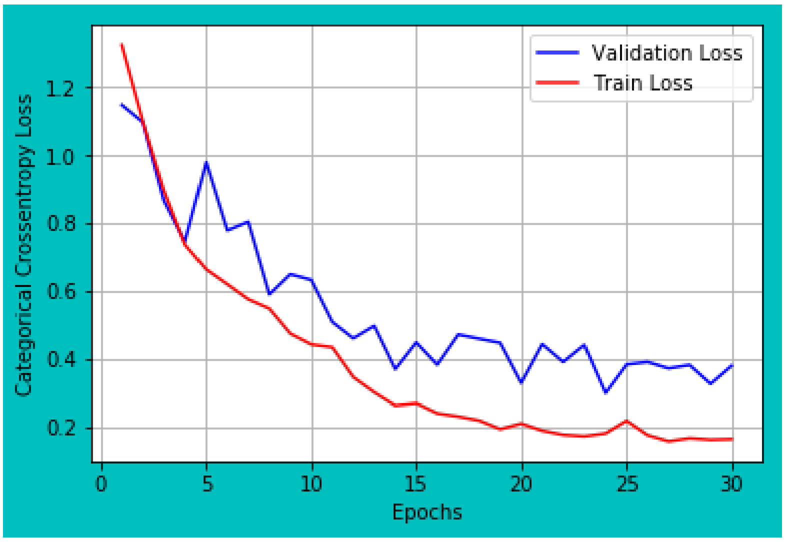
```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 24s 3ms/step - loss: 1.3
247 - acc: 0.4308 - val_loss: 1.1474 - val_acc: 0.4808
Epoch 2/30
7352/7352 [==============================] - 24s 3ms/step - loss: 1.0
995 - acc: 0.5196 - val_loss: 1.0965 - val_acc: 0.5236
Epoch 3/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.8
968 - acc: 0.6092 - val_loss: 0.8670 - val_acc: 0.6312
Epoch 4/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.7
366 - acc: 0.6532 - val_loss: 0.7453 - val_acc: 0.6125
Epoch 5/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.6
636 - acc: 0.6768 - val_loss: 0.9799 - val_acc: 0.5989
Epoch 6/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.6
197 - acc: 0.6989 - val_loss: 0.7784 - val_acc: 0.6498
Epoch 7/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.5
752 - acc: 0.7444 - val_loss: 0.8042 - val_acc: 0.6820
Epoch 8/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.5
479 - acc: 0.7636 - val_loss: 0.5897 - val_acc: 0.7438
Epoch 9/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.4
742 - acc: 0.7856 - val_loss: 0.6495 - val_acc: 0.7258
Epoch 10/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.4
419 - acc: 0.8069 - val_loss: 0.6326 - val_acc: 0.7788
Epoch 11/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.4
336 - acc: 0.8388 - val_loss: 0.5086 - val_acc: 0.8554
Epoch 12/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.3
461 - acc: 0.8980 - val_loss: 0.4601 - val_acc: 0.8605
Epoch 13/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.3
012 - acc: 0.9075 - val_loss: 0.4970 - val_acc: 0.8504
Epoch 14/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2
620 - acc: 0.9210 - val_loss: 0.3690 - val_acc: 0.8884
Epoch 15/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2
688 - acc: 0.9223 - val_loss: 0.4479 - val_acc: 0.8744
Epoch 16/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2
391 - acc: 0.9238 - val_loss: 0.3821 - val_acc: 0.9002
Epoch 17/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2
299 - acc: 0.9280 - val_loss: 0.4713 - val_acc: 0.8856
```

```
Epoch 18/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2
181 - acc: 0.9372 - val_loss: 0.4589 - val_acc: 0.8989
Epoch 19/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1
929 - acc: 0.9377 - val_loss: 0.4467 - val_acc: 0.8992
Epoch 20/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2
092 - acc: 0.9363 - val_loss: 0.3284 - val_acc: 0.8914
Epoch 21/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1
884 - acc: 0.9400 - val_loss: 0.4429 - val_acc: 0.8921
Epoch 22/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1
763 - acc: 0.9412 - val_loss: 0.3904 - val_acc: 0.9043
Epoch 23/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1
721 - acc: 0.9403 - val_loss: 0.4405 - val_acc: 0.9050
Epoch 24/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1
805 - acc: 0.9397 - val_loss: 0.2988 - val_acc: 0.9074
Epoch 25/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2
175 - acc: 0.9402 - val_loss: 0.3831 - val_acc: 0.8996
Epoch 26/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1
753 - acc: 0.9408 - val_loss: 0.3904 - val_acc: 0.8968
Epoch 27/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1
573 - acc: 0.9412 - val_loss: 0.3716 - val_acc: 0.9033
Epoch 28/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1
663 - acc: 0.9421 - val_loss: 0.3816 - val_acc: 0.9094
Epoch 29/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1
619 - acc: 0.9434 - val_loss: 0.3260 - val_acc: 0.9135
Epoch 30/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1
636 - 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%
```



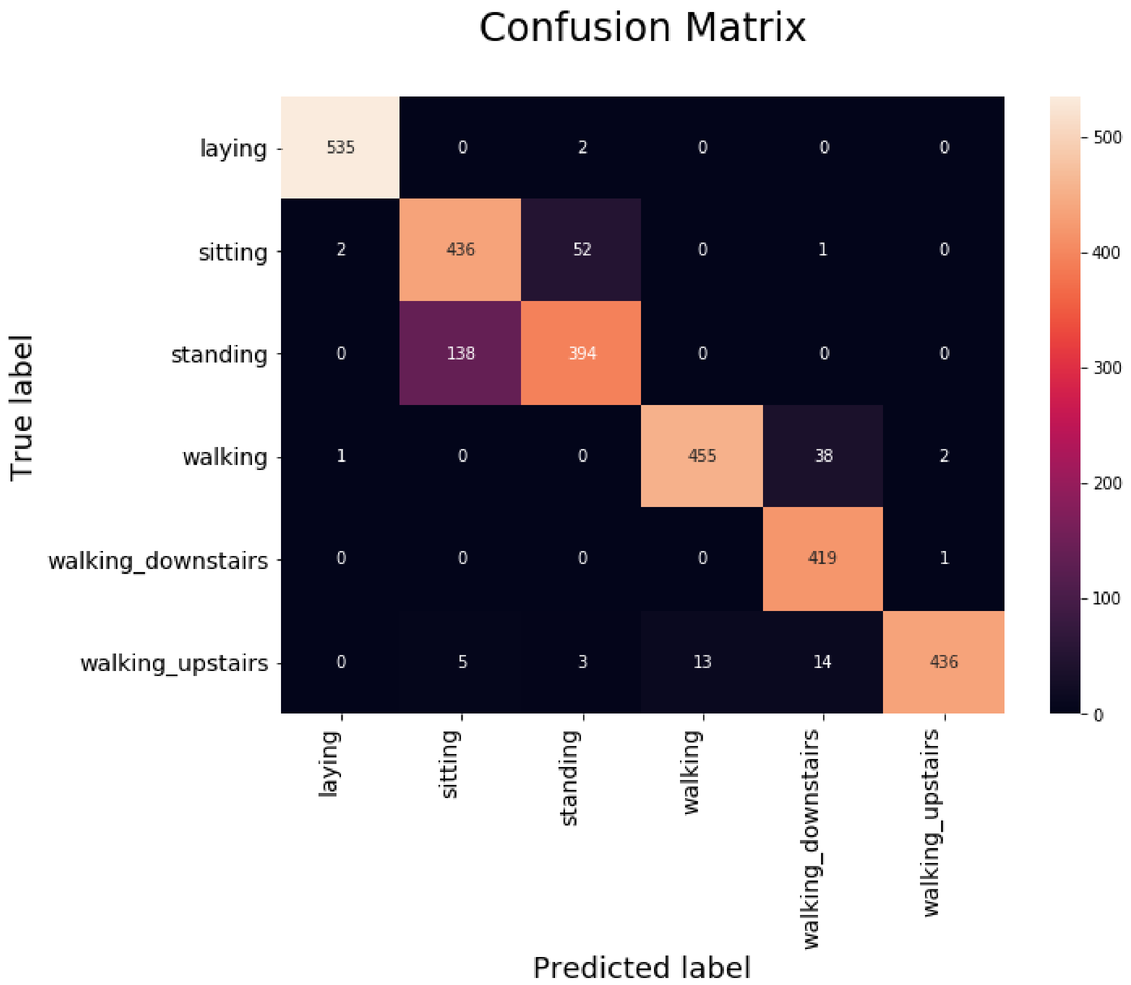
Observation:

- 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.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: 1.3
700 - acc: 0.4207 - val_loss: 1.3605 - val_acc: 0.3858
Epoch 2/30
7352/7352 [==============================] - 24s 3ms/step - loss: 1.2
399 - acc: 0.4460 - val_loss: 1.3068 - val_acc: 0.4150
Epoch 3/30
7352/7352 [==============================] - 24s 3ms/step - loss: 1.1
828 - acc: 0.4551 - val_loss: 1.1227 - val_acc: 0.4645
Epoch 4/30
7352/7352 [==============================] - 24s 3ms/step - loss: 1.1
022 - acc: 0.5275 - val_loss: 1.0100 - val_acc: 0.6016
Epoch 5/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.8
570 - acc: 0.6140 - val_loss: 0.9916 - val_acc: 0.5836
Epoch 6/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.8
873 - acc: 0.6019 - val_loss: 0.8917 - val_acc: 0.6281
Epoch 7/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.9
034 - acc: 0.6064 - val_loss: 0.8474 - val_acc: 0.6135
Epoch 8/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.7
693 - acc: 0.6442 - val_loss: 0.9360 - val_acc: 0.5786
```

```
Epoch 9/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.8
396 - acc: 0.6164 - val_loss: 0.8555 - val_acc: 0.6250
Epoch 10/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.7
415 - acc: 0.6522 - val_loss: 0.8564 - val_acc: 0.6298
Epoch 11/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.7
076 - acc: 0.6581 - val_loss: 0.8041 - val_acc: 0.6454
Epoch 12/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.6
867 - acc: 0.6712 - val_loss: 1.0092 - val_acc: 0.6118
Epoch 13/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.6
644 - acc: 0.6999 - val_loss: 0.8403 - val_acc: 0.6960
Epoch 14/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.5
832 - acc: 0.7692 - val_loss: 0.7345 - val_acc: 0.7723
Epoch 15/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.5
339 - acc: 0.8195 - val_loss: 0.6299 - val_acc: 0.7825
Epoch 16/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.4
096 - acc: 0.8643 - val_loss: 0.5387 - val_acc: 0.8409
Epoch 17/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.3
963 - acc: 0.8713 - val_loss: 0.4963 - val_acc: 0.8514
Epoch 18/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.3
419 - acc: 0.9038 - val_loss: 0.4934 - val_acc: 0.8690
Epoch 19/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.3
055 - acc: 0.9011 - val_loss: 0.3953 - val_acc: 0.8816
Epoch 20/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.2
563 - acc: 0.9172 - val_loss: 0.4246 - val_acc: 0.8918
Epoch 21/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.3
852 - acc: 0.8768 - val_loss: 0.4623 - val_acc: 0.8595
Epoch 22/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2
499 - acc: 0.9227 - val_loss: 0.4276 - val_acc: 0.8860
Epoch 23/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2
039 - acc: 0.9310 - val_loss: 0.3738 - val_acc: 0.8863
Epoch 24/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.4
023 - acc: 0.8913 - val_loss: 0.3902 - val_acc: 0.8904
Epoch 25/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2
673 - acc: 0.9215 - val_loss: 0.3429 - val_acc: 0.8877
Epoch 26/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2
183 - acc: 0.9305 - val_loss: 0.3252 - val_acc: 0.8968
Epoch 27/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.3
781 - acc: 0.8656 - val_loss: 0.5003 - val_acc: 0.8300
Epoch 28/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.3
450 - acc: 0.8724 - val_loss: 0.4568 - val_acc: 0.8836
Epoch 29/30
```

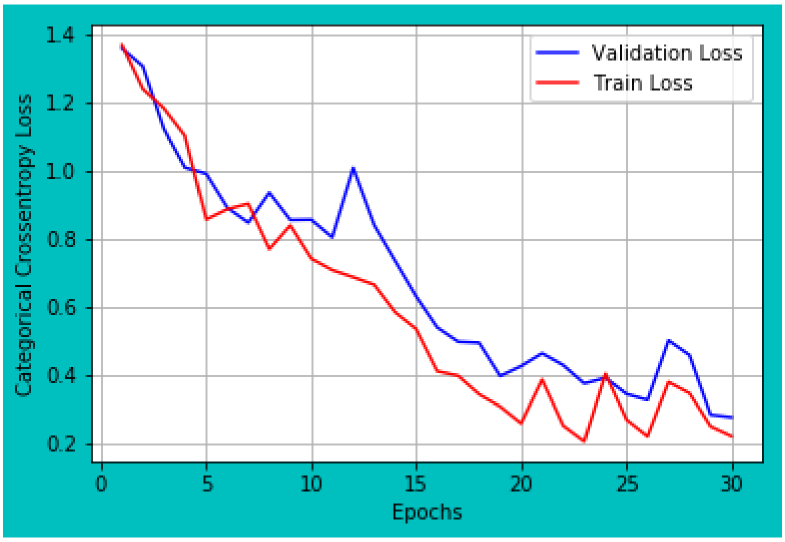
```
7352/7352 [==============================] - 24s 3ms/step - loss: 0.2
477 - acc: 0.9135 - val_loss: 0.2817 - val_acc: 0.8856
Epoch 30/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2
192 - acc: 0.9291 - val_loss: 0.2744 - val_acc: 0.8999
```

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)
```

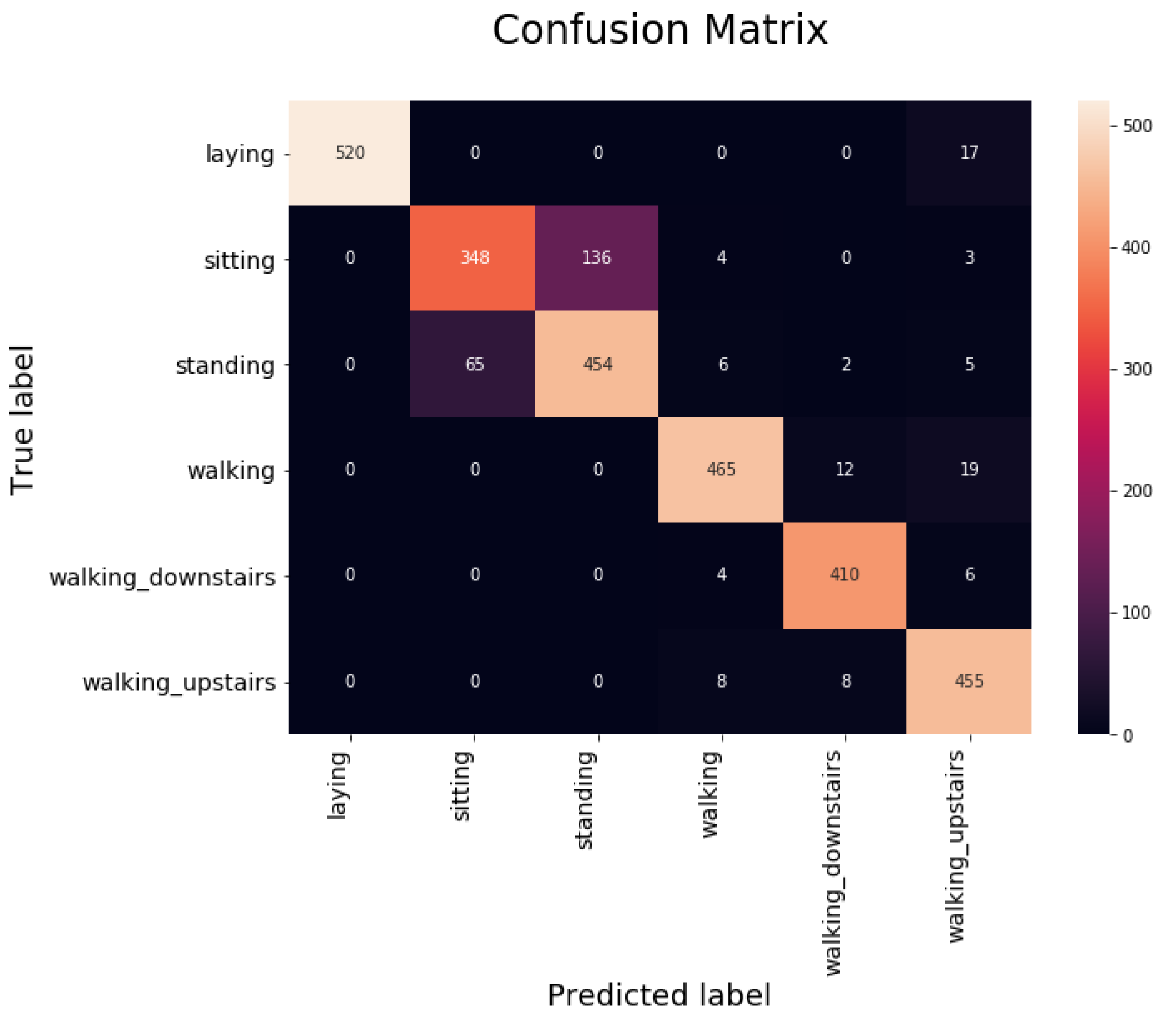
```
Test Score: 0.274399
Train Accuracy: 93.103917%
Test Accuracy: 89.989820%
```



- 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),

# seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downstairs','walkin
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_name
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
```

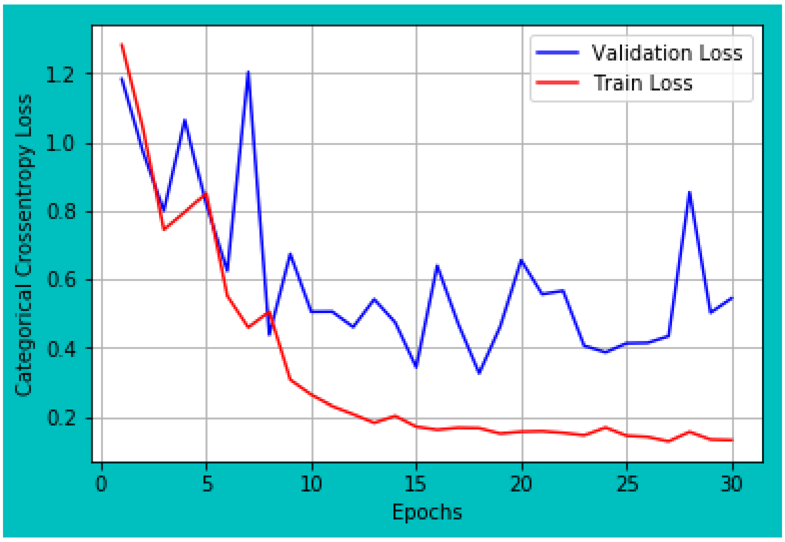
```
Epoch 9/30
7352/7352 [==============================] - 30s 4ms/step - loss:
0.3067 - acc: 0.9008 - val_loss: 0.6739 - val_acc: 0.8202
Epoch 10/30
7352/7352 [==============================] - 30s 4ms/step - loss:
0.2629 - acc: 0.9162 - val_loss: 0.5049 - val_acc: 0.8694
Epoch 11/30
7352/7352 [==============================] - 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]:

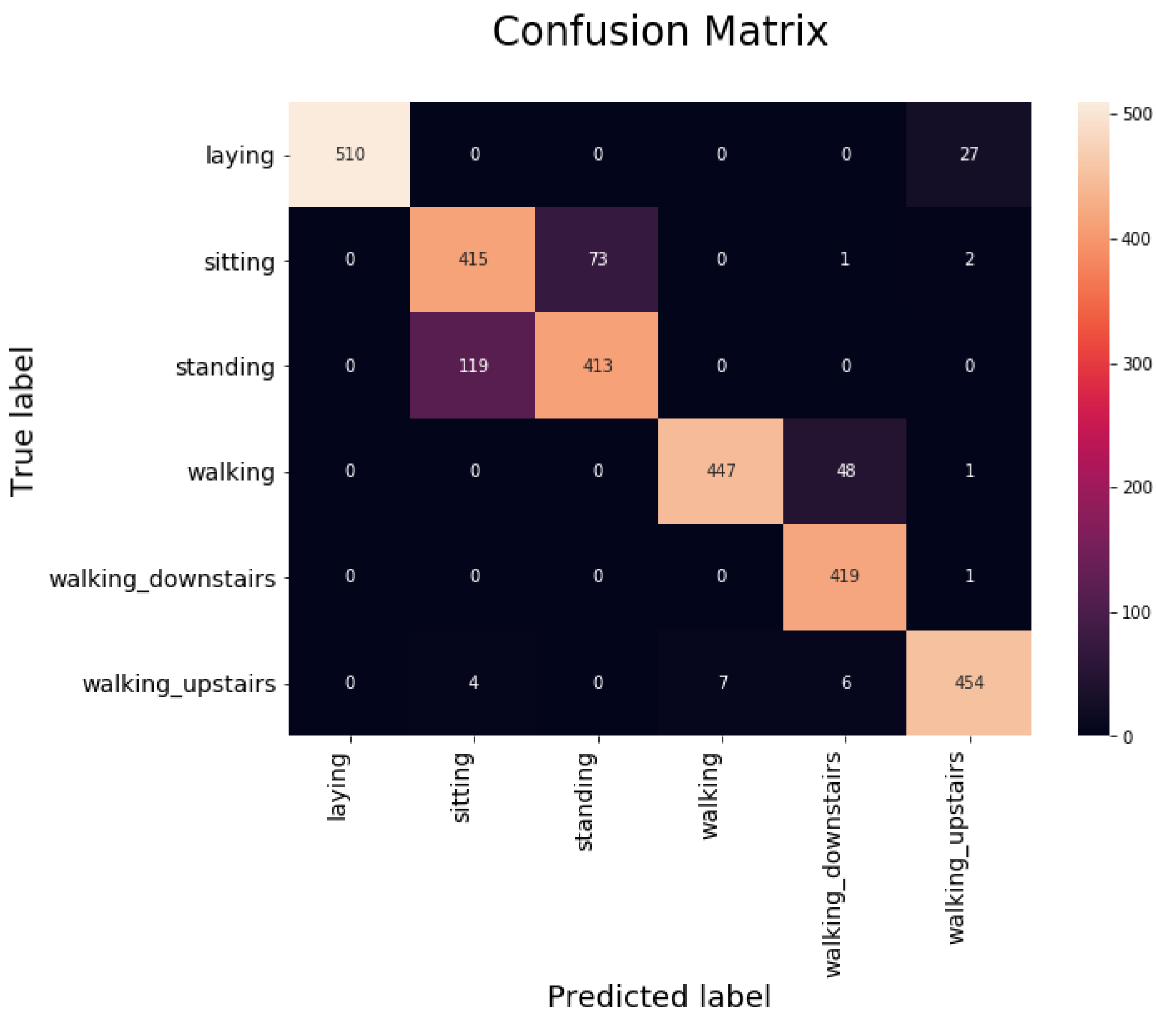
```
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores[0]))
test_acc3= scores[1]*100
train_acc3=(max(hist3.history['acc']))* 100
print("Train Accuracy: %f%%"% (train_acc3))

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),

# seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downstairs','walkin
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_name
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
```

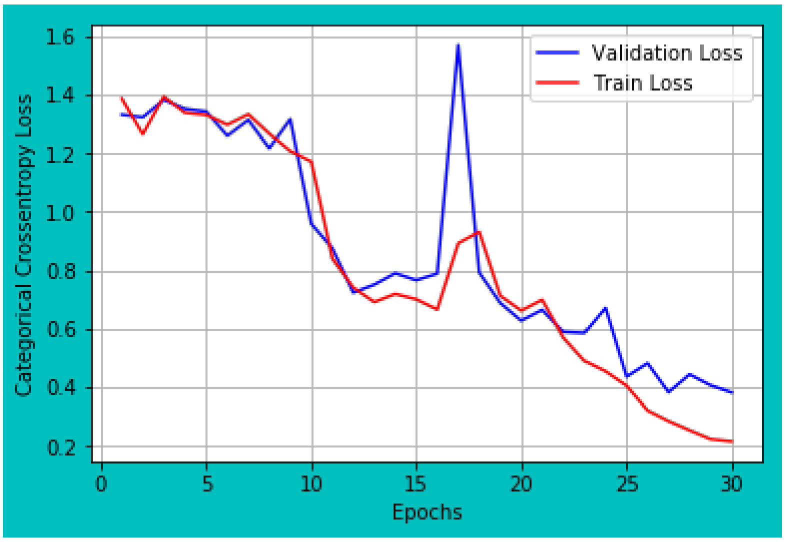
```
Epoch 9/30
7352/7352 [==============================] - 29s 4ms/step - loss:
1.2073 - acc: 0.4894 - val_loss: 1.3158 - val_acc: 0.3858
Epoch 10/30
7352/7352 [==============================] - 29s 4ms/step - loss:
1.1716 - acc: 0.4988 - val_loss: 0.9594 - val_acc: 0.5453
Epoch 11/30
7352/7352 [==============================] - 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
```

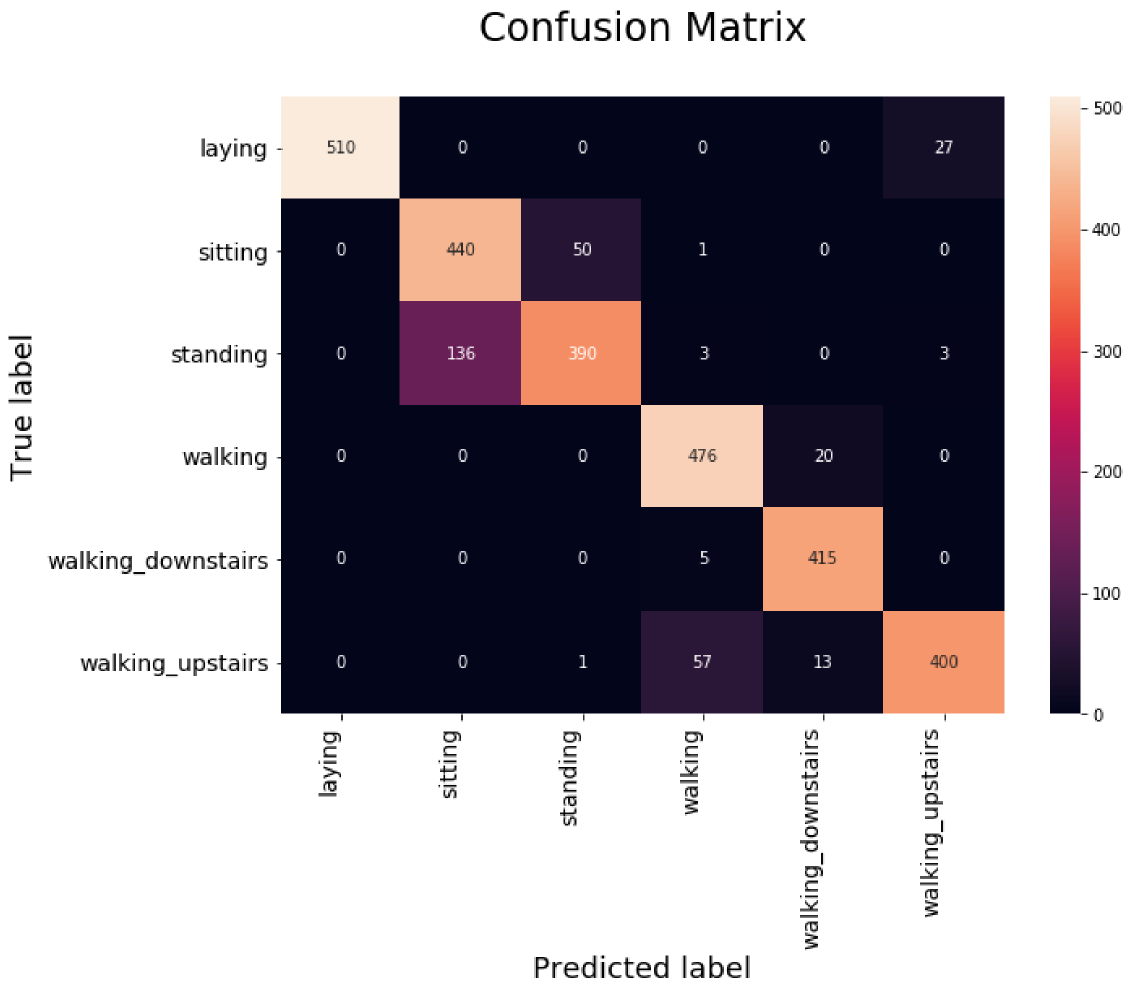
```
7352/7352 [==============================] - 29s 4ms/step - loss:
0.2243 - acc: 0.9232 - val_loss: 0.4070 - val_acc: 0.8660
Epoch 30/30
7352/7352 [==============================] - 29s 4ms/step - loss:
0.2165 - acc: 0.9208 - val_loss: 0.3825 - val_acc: 0.8928
```

In [29]:

```
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),

# seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downstairs','walkin
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_name
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]:

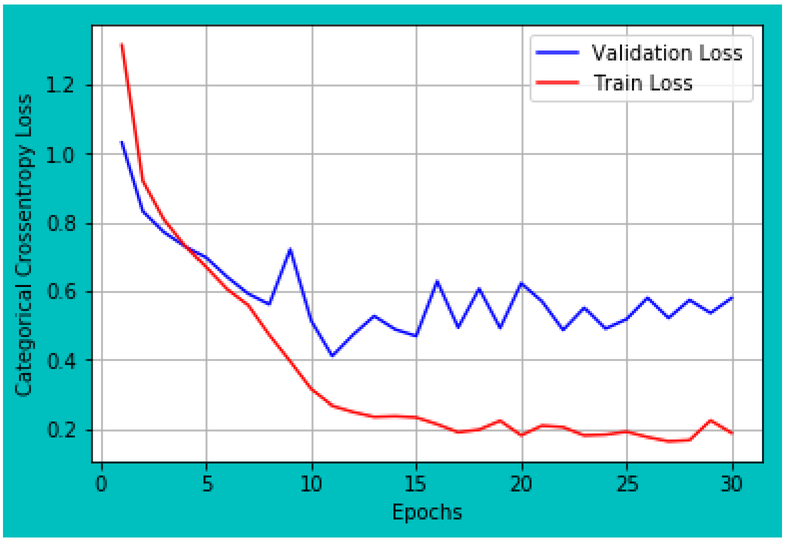
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(32,return_sequences=True,
               input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.65))
# second LSTM layer
model.add(LSTM(32))
model.add(Dropout(0.65))
# Adding 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)
```

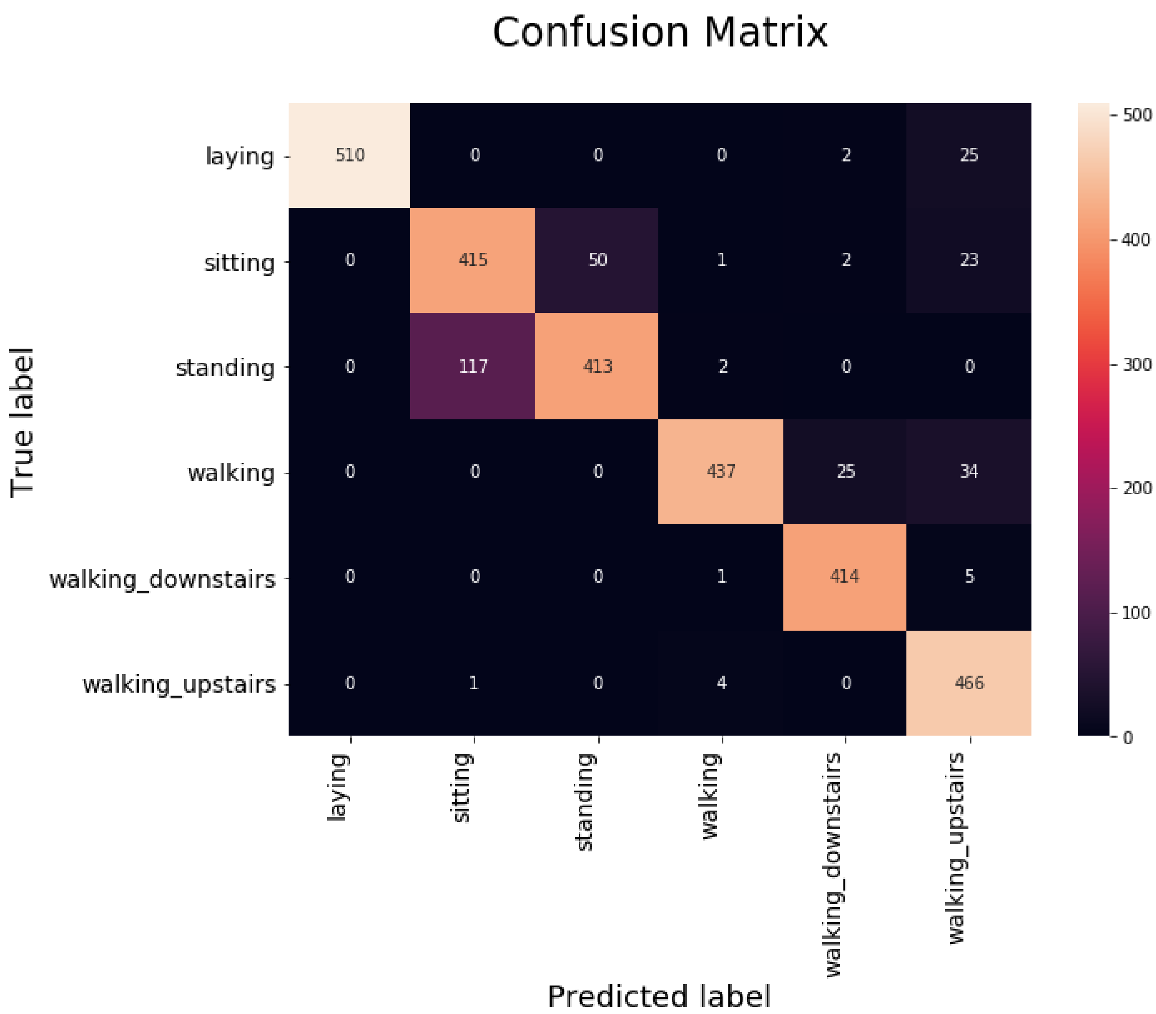
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_5 (LSTM)                (None, 128, 32)           5376
_________________________________________________________________
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
```

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.predict(X_test),

# seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downstairs','walkin
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_name
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]:

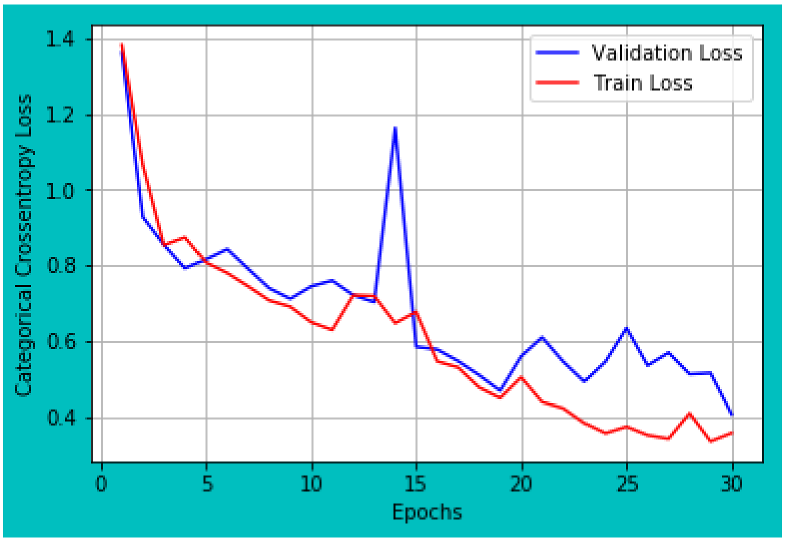
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(32,return_sequences=True,
               input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.65))
# second LSTM layer
model.add(LSTM(32))
model.add(Dropout(0.65))
# Adding 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:
```

In [33]:

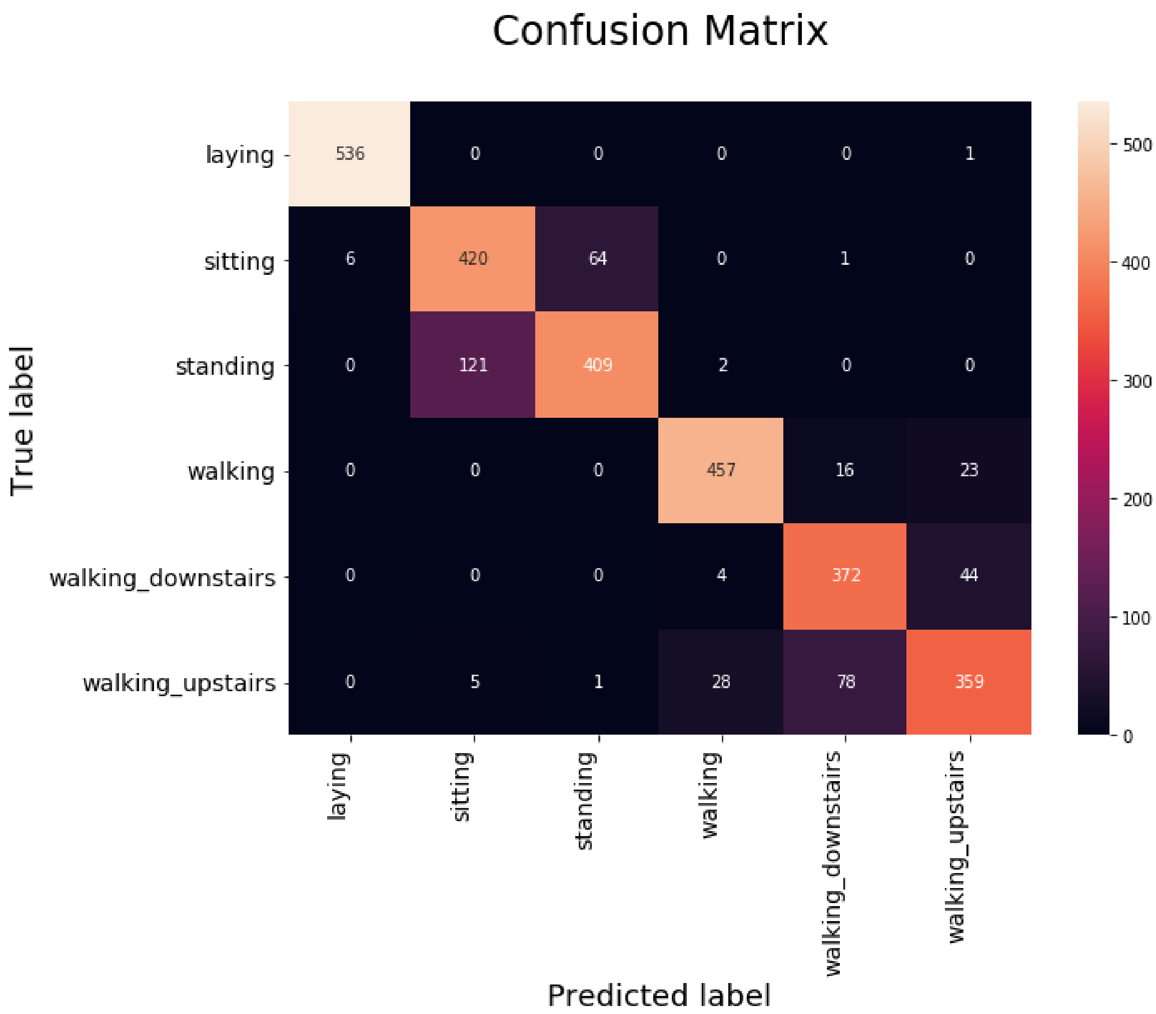
```
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores[0]))
test_acc6= scores[1]*100
train_acc6=(max(hist6.history['acc']))* 100
print("Train Accuracy: %f%%"% (train_acc6))

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),

# seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downstairs','walkin
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_name
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]:

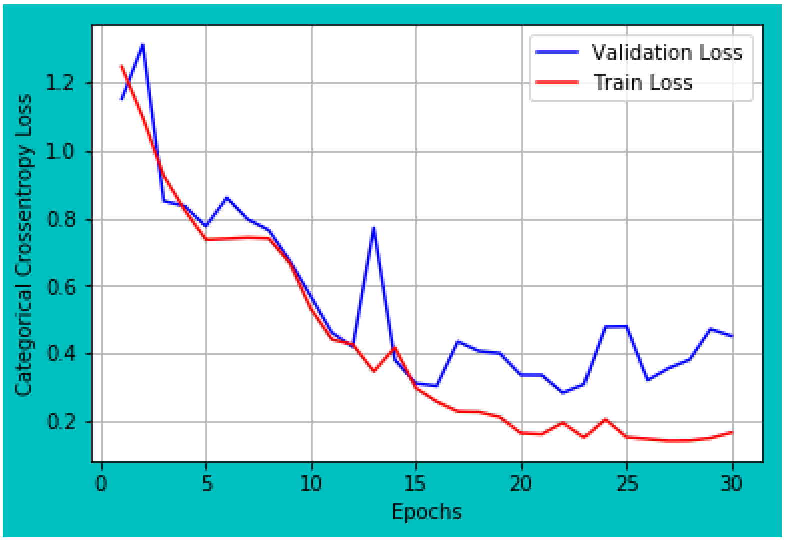
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(64,return_sequences=True,
               input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.65))
# second LSTM layer
model.add(LSTM(64))
model.add(Dropout(0.65))
# Adding 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:
```

In [36]:

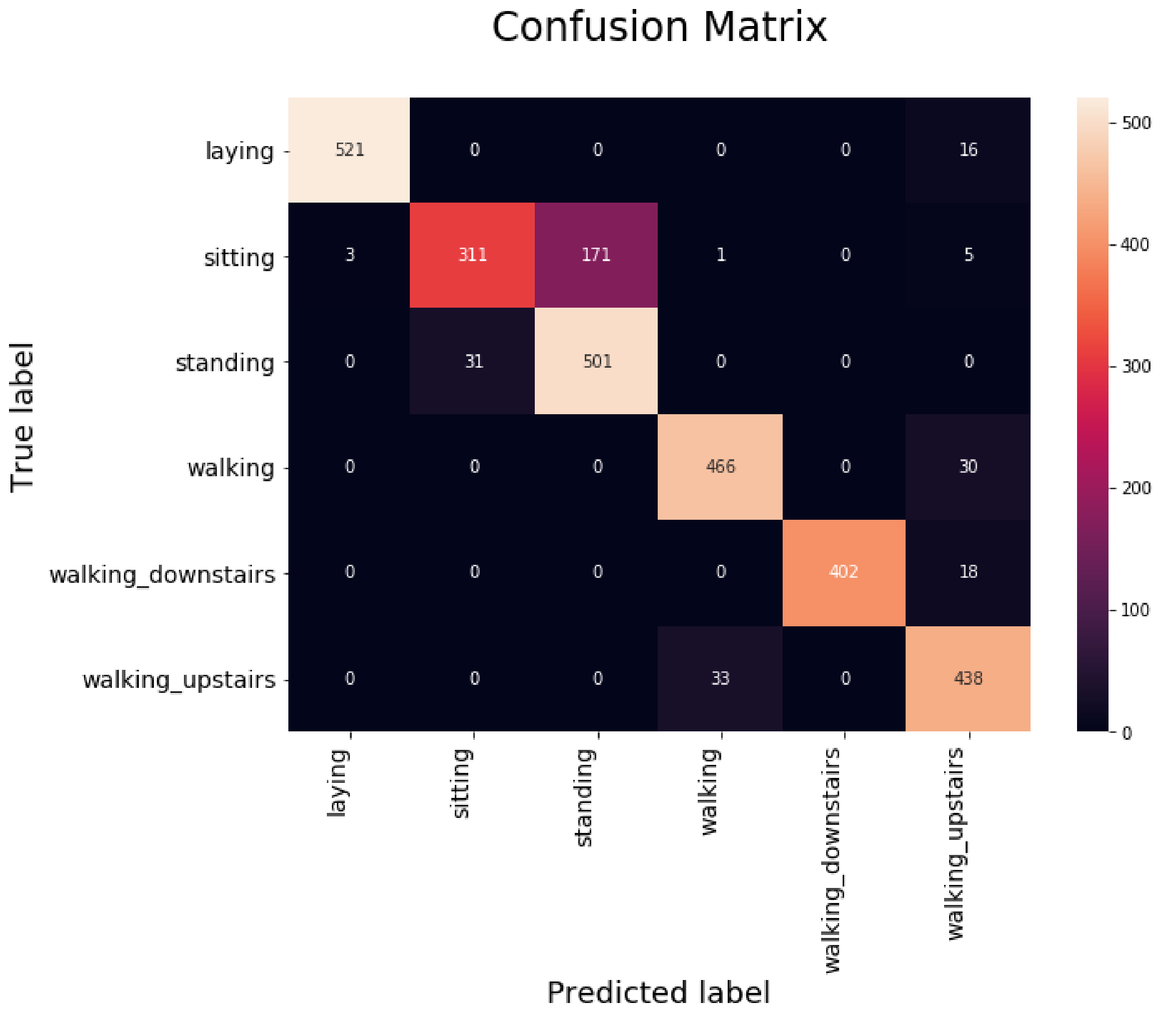
```
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores[0]))
test_acc7= scores[1]*100
train_acc7=(max(hist7.history['acc']))* 100
print("Train Accuracy: %f%%"% (train_acc7))

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),

# seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downstairs','walkin
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_name
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]:

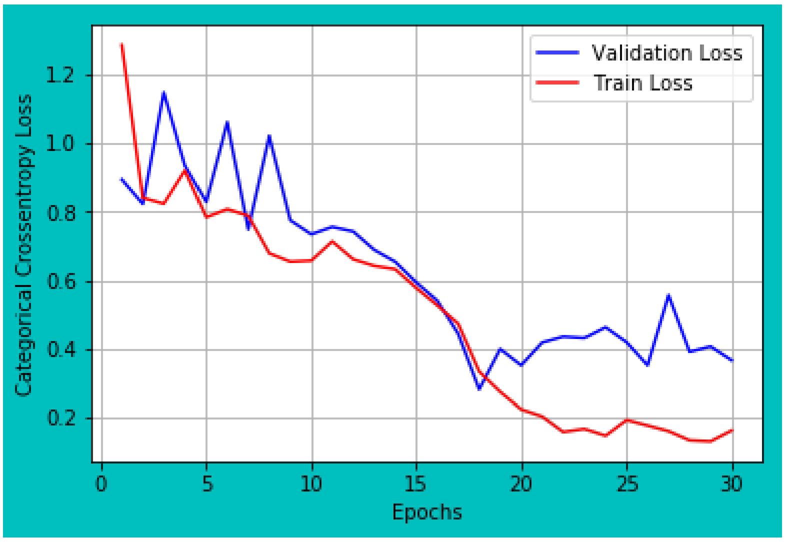
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(64,return_sequences=True,
               input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.65))
# second LSTM layer
model.add(LSTM(64))
model.add(Dropout(0.65))
# Adding 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
```

In [39]:

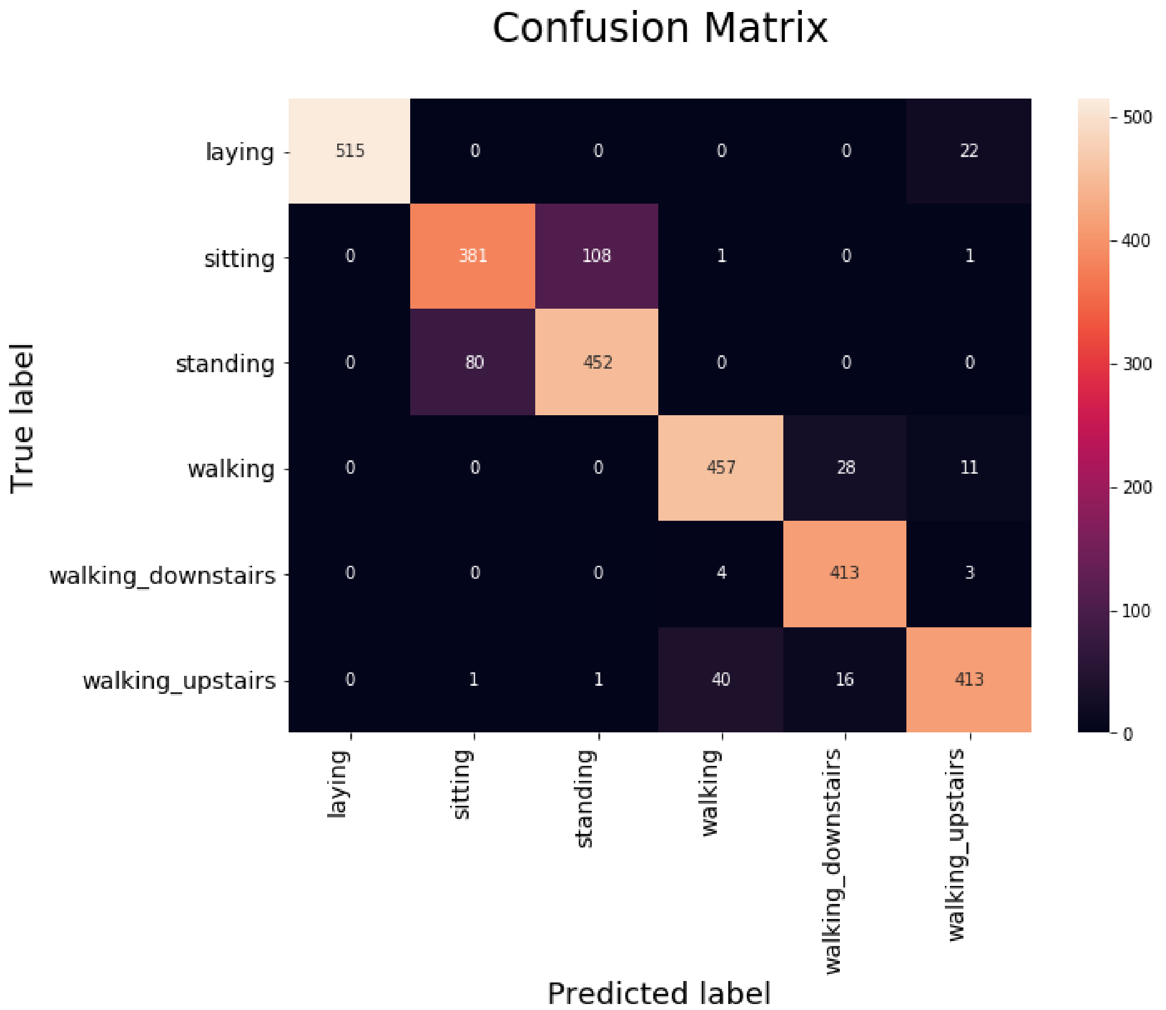
```
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Test Score: %f" % (scores[0]))
test_acc8= scores[1]*100
train_acc8=(max(hist8.history['acc']))* 100
print("Train Accuracy: %f%%"% (train_acc8))

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),

# seaborn heatmaps
class_names = ['laying','sitting','standing','walking','walking_downstairs','walkin
df_heatmap = pd.DataFrame(confusion_matrix(Y_true, Y_predictions), index=class_name
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                    | TRAINI
NG ACCURACY |  TESTING ACCURACY |
+--------+---------------------------------------------------+-------
------------+------------------+
|   1    |        32LSTM+1layerLSTM +rmsprop_optimizer       | 94.736
12622415669 | 90.77027485578554 |
|   2    |         32LSTM+1layerLSTM +adam_optimizer         | 93.103
91730141458 | 89.98982015609094 |
|   3    |        64LSTM+1layerLSTM +rmsprop_optimizer       | 95.198
58541893362 | 90.19341703427214 |
|   4    |         64LSTM+1layerLSTM +adam_optimizer         | 92.315
01632208922 | 89.27723108245674 |
|   5    | 32LSTM+2layerLSTM +rmsprop_optimizer+0.65drop_out | 94.545
70184983679 | 90.09161859518154 |
|   6    |   32LSTM+2layerLSTM +adam_optimizer+0.65drop_out  | 85.065
28835690969 | 86.63047166610112 |
|   7    |   64LSTM+2layerLSTM+adam_optimizer+0.65drop_out   | 95.307
39934711643 | 89.54869358669833 |
|   8    |  64LSTM+2layerLSTM+rmsprop_optimizer+0.65drop_out | 95.144
17845484222 | 89.27723108245674 |
+--------+---------------------------------------------------+-------
------------+------------------+
```

OBSERVATIONS:

- After increasing the hidden layers from 32 to 64 with 1 layer LSTM, model test accuracy has decreased.

- On increasing the number of LSTM layers, we found that model performing good on validation data, but not on test data. And hence is overfitting.
- RMS optimizer is prerforming better than adam optimiser.

AIM - To increase accuracy above 94%

Model 9

In [0]:

```
# With One LSTM Layer Model 1  #
n_hidden = 80

model = Sequential()

# 1 LSTM layer
model.add(LSTM(n_hidden, input_shape = (timesteps, input_dim)))     # 1 LSTM

model.add(Dropout(0.25))
model.add(Dense(n_classes, activation = 'sigmoid'))
model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accurac
print(model.summary())
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_4 (LSTM)                (None, 80)                28800     
_________________________________________________________________
dropout_4 (Dropout)          (None, 80)                0         
_________________________________________________________________
dense_4 (Dense)              (None, 6)                 486       
=================================================================
Total params: 29,286
Trainable params: 29,286
Non-trainable params: 0
_________________________________________________________________
None
```

In [0]:

```
# Training the model
history = model.fit(X_train,
          Y_train,
          batch_size=64,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.4
190 - acc: 0.8390 - val_loss: 0.3644 - val_acc: 0.8571
Epoch 2/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.3
535 - acc: 0.8616 - val_loss: 0.3331 - val_acc: 0.8613
Epoch 3/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.3
183 - acc: 0.8736 - val_loss: 0.3162 - val_acc: 0.8704
Epoch 4/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2
887 - acc: 0.8776 - val_loss: 0.2728 - val_acc: 0.8806
Epoch 5/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2
342 - acc: 0.8948 - val_loss: 0.2279 - val_acc: 0.8955
Epoch 6/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2
182 - acc: 0.8994 - val_loss: 0.2412 - val_acc: 0.8933
Epoch 7/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2
087 - acc: 0.9015 - val_loss: 0.2105 - val_acc: 0.9009
Epoch 8/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1
874 - acc: 0.9109 - val_loss: 0.2073 - val_acc: 0.9037
Epoch 9/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1
813 - acc: 0.9161 - val_loss: 0.1885 - val_acc: 0.9164
Epoch 10/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1
476 - acc: 0.9429 - val_loss: 0.1629 - val_acc: 0.9406
Epoch 11/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1
349 - acc: 0.9464 - val_loss: 0.1546 - val_acc: 0.9390
Epoch 12/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1
186 - acc: 0.9572 - val_loss: 0.1608 - val_acc: 0.9467
Epoch 13/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0
865 - acc: 0.9713 - val_loss: 0.1619 - val_acc: 0.9440
Epoch 14/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0
809 - acc: 0.9730 - val_loss: 0.1272 - val_acc: 0.9545
Epoch 15/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0
792 - acc: 0.9726 - val_loss: 0.1019 - val_acc: 0.9637
Epoch 16/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1
704 - acc: 0.9347 - val_loss: 0.1586 - val_acc: 0.9412
Epoch 17/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0
992 - acc: 0.9674 - val_loss: 0.1695 - val_acc: 0.9481
```

```
Epoch 18/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1
203 - acc: 0.9567 - val_loss: 0.1166 - val_acc: 0.9592
Epoch 19/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0
920 - acc: 0.9701 - val_loss: 0.1421 - val_acc: 0.9531
Epoch 20/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1
147 - acc: 0.9616 - val_loss: 0.1311 - val_acc: 0.9570
Epoch 21/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0
738 - acc: 0.9737 - val_loss: 0.1042 - val_acc: 0.9641
Epoch 22/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.0
671 - acc: 0.9752 - val_loss: 0.1112 - val_acc: 0.9606
Epoch 23/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1
143 - acc: 0.9567 - val_loss: 0.1159 - val_acc: 0.9583
Epoch 24/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.0
743 - acc: 0.9738 - val_loss: 0.1186 - val_acc: 0.9601
Epoch 25/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0
684 - acc: 0.9777 - val_loss: 0.1046 - val_acc: 0.9650
Epoch 26/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0
576 - acc: 0.9798 - val_loss: 0.0958 - val_acc: 0.9663
Epoch 27/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0
481 - acc: 0.9819 - val_loss: 0.0974 - val_acc: 0.9646
Epoch 28/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0
474 - acc: 0.9816 - val_loss: 0.0986 - val_acc: 0.9676
Epoch 29/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0
617 - acc: 0.9770 - val_loss: 0.1099 - val_acc: 0.9645
Epoch 30/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0
488 - acc: 0.9810 - val_loss: 0.0985 - val_acc: 0.9663
```

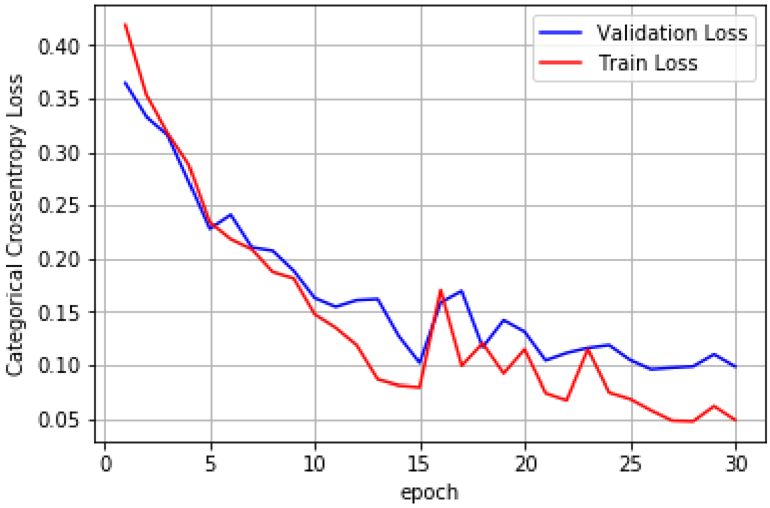
In [0]:

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

```
2947/2947 [==============================] - 8s 3ms/step
[0.09848940819087885, 0.9662934093908977]
```

In [0]:

```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

# list of epoch numbers
x = list(range(1,epochs+1))

# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch

# we will get val_loss and val_acc only when you pass the paramter validation_data
# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
# for each key in histrory.histrory we will have a list of length equal to number o

vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```



In [0]:

```
```

Model 10

In [0]:

```
# With One LSTM Layer Model 1  #
n_hidden = 80

model = Sequential()

# 1 LSTM layer
model.add(LSTM(n_hidden, input_shape = (timesteps, input_dim), return_sequences = T

model.add(Dropout(0.25))
model.add(LSTM(n_hidden))
model.add(Dense(n_classes, activation = 'sigmoid'))

model.compile(loss = 'binary_crossentropy', optimizer = 'rmsprop', metrics = ['accu
print(model.summary())
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/kera
s/backend/tensorflow_backend.py:66: The name tf.get_default_graph i
s deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/kera
s/backend/tensorflow_backend.py:541: The name tf.placeholder is dep
recated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/kera
s/backend/tensorflow_backend.py:4432: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/kera
s/backend/tensorflow_backend.py:148: The name tf.placeholder_with_d
efault is deprecated. Please use tf.compat.v1.placeholder_with_defa
ult instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/kera
s/backend/tensorflow_backend.py:3733: calling dropout (from tensorf
low.python.ops.nn_ops) with keep_prob is deprecated and will be rem
oved in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `ra
te = 1 - keep_prob`.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/kera
s/optimizers.py:793: The name tf.train.Optimizer is deprecated. Ple
ase use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/kera
s/backend/tensorflow_backend.py:3657: The name tf.log is deprecate
d. Please use tf.math.log instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tens
orflow/python/ops/nn_impl.py:180: add_dispatch_support.<locals>.wra
pper (from tensorflow.python.ops.array_ops) is deprecated and will
be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 128, 80)           28800
_________________________________________________________________
```

```
dropout_1 (Dropout)          (None, 128, 80)          0
_________________________________________________________________
lstm_2 (LSTM)                (None, 80)               51520
_________________________________________________________________
dense_1 (Dense)              (None, 6)                486
=================================================================
Total params: 80,806
Trainable params: 80,806
Non-trainable params: 0
_________________________________________________________________
None
```

In [0]:

```
%%time
# Training the model
history = model.fit(X_train,
          Y_train,
          batch_size= 64,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
341 - acc: 0.9861 - val_loss: 0.0949 - val_acc: 0.9742
Epoch 2/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
371 - acc: 0.9852 - val_loss: 0.1142 - val_acc: 0.9724
Epoch 3/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
352 - acc: 0.9856 - val_loss: 0.1288 - val_acc: 0.9688
Epoch 4/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
390 - acc: 0.9845 - val_loss: 0.0962 - val_acc: 0.9751
Epoch 5/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
366 - acc: 0.9847 - val_loss: 0.1185 - val_acc: 0.9722
Epoch 6/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
370 - acc: 0.9849 - val_loss: 0.1057 - val_acc: 0.9723
Epoch 7/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
331 - acc: 0.9867 - val_loss: 0.1143 - val_acc: 0.9738
Epoch 8/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
332 - acc: 0.9860 - val_loss: 0.1110 - val_acc: 0.9671
Epoch 9/30
7352/7352 [==============================] - 59s 8ms/step - loss: 0.0
319 - acc: 0.9864 - val_loss: 0.1210 - val_acc: 0.9723
Epoch 10/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
354 - acc: 0.9867 - val_loss: 0.1088 - val_acc: 0.9736
Epoch 11/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
304 - acc: 0.9886 - val_loss: 0.1426 - val_acc: 0.9748
Epoch 12/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
331 - acc: 0.9860 - val_loss: 0.2099 - val_acc: 0.9475
Epoch 13/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
332 - acc: 0.9870 - val_loss: 0.1255 - val_acc: 0.9623
Epoch 14/30
7352/7352 [==============================] - 59s 8ms/step - loss: 0.0
299 - acc: 0.9883 - val_loss: 0.1483 - val_acc: 0.9718
Epoch 15/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
304 - acc: 0.9877 - val_loss: 0.1413 - val_acc: 0.9688
Epoch 16/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
317 - acc: 0.9885 - val_loss: 0.1288 - val_acc: 0.9744
Epoch 17/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
```

```
308 - acc: 0.9879 - val_loss: 0.1279 - val_acc: 0.9727
Epoch 18/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
301 - acc: 0.9886 - val_loss: 0.1383 - val_acc: 0.9695
Epoch 19/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
303 - acc: 0.9891 - val_loss: 0.1219 - val_acc: 0.9738
Epoch 20/30
7352/7352 [==============================] - 59s 8ms/step - loss: 0.0
332 - acc: 0.9872 - val_loss: 0.1546 - val_acc: 0.9686
Epoch 21/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
343 - acc: 0.9877 - val_loss: 0.1032 - val_acc: 0.9755
Epoch 22/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
329 - acc: 0.9874 - val_loss: 0.1271 - val_acc: 0.9674
Epoch 23/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
297 - acc: 0.9876 - val_loss: 0.1098 - val_acc: 0.9737
Epoch 24/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
290 - acc: 0.9892 - val_loss: 0.1386 - val_acc: 0.9716
Epoch 25/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
336 - acc: 0.9884 - val_loss: 0.1342 - val_acc: 0.9760
Epoch 26/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
327 - acc: 0.9875 - val_loss: 0.1325 - val_acc: 0.9699
Epoch 27/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
342 - acc: 0.9871 - val_loss: 0.1163 - val_acc: 0.9701
Epoch 28/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
291 - acc: 0.9890 - val_loss: 0.1486 - val_acc: 0.9640
Epoch 29/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0
303 - acc: 0.9890 - val_loss: 0.1134 - val_acc: 0.9759
Epoch 30/30
7352/7352 [==============================] - 59s 8ms/step - loss: 0.0
272 - acc: 0.9901 - val_loss: 0.1277 - val_acc: 0.9695
CPU times: user 32min 43s, sys: 1min 43s, total: 34min 27s
Wall time: 29min 1s
```

In [0]:

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

```
2947/2947 [==============================] - 17s 6ms/step
[0.1007159318419772, 0.9716095518248745]
```

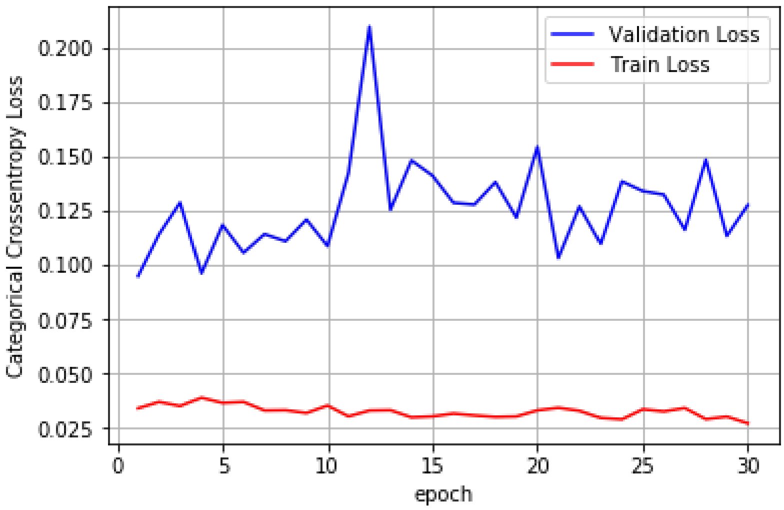
In [0]:

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

```
Pred                LAYING  SITTING  ...  WALKING_DOWNSTAIRS  WALKING
_UPSTAIRS
True                                 ...
LAYING                 511        0  ...                   0
0
SITTING                  0      351  ...                   0
2
STANDING                 0       22  ...                   1
0
WALKING                  0        0  ...                  43
1
WALKING_DOWNSTAIRS       0        0  ...                 420
0
WALKING_UPSTAIRS         0        4  ...                  14
440

[6 rows x 6 columns]
```

In [0]:

```
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')

# list of epoch numbers
x = list(range(1,epochs+1))

# print(history.history.keys())
# dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
# history = model_drop.fit(X_train, Y_train, batch_size=batch_size, epochs=nb_epoch

# we will get val_loss and val_acc only when you pass the paramter validation_data
# val_loss : validation loss
# val_acc : validation accuracy

# loss : training loss
# acc : train accuracy
# for each key in histrory.histrory we will have a list of length equal to number o

vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty, ax)
```



Conclusion

In [2]:

```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Model","Hidden layer","activation","Optimizer", "Test accuracy in

x.add_row(["MODEL-9 : Lstm + dropout(0.25)","80","sigmoid" ,"Adam", "96.62%"])
x.add_row(["MODEL-10 :Lstm + dropout(0.25)","80","sigmoid","rmsprop" ,"97.16%"])


print(x)
```

```
+--------------------------------+-------------+-----------+-------
----+--------------------+
|             Model              | Hidden layer | activation | Optimi
zer | Test accuracy in % |
+--------------------------------+-------------+-----------+-------
----+--------------------+
| MODEL-9 : Lstm + dropout(0.25) |      80      |  sigmoid   |    Ada
m   |       96.62%       |
| MODEL-10 :Lstm + dropout(0.25) |      80      |  sigmoid   |  rmspr
op  |       97.16%       |
+--------------------------------+-------------+-----------+-------
----+--------------------+
```

- By increasing the hidden layers to 80 we got a very good result.
- Lstm + dropout(0.25) model ,with 80 hidden layers with "sigmoid" activation function we got a test acccuracy of 97.16%.