In [1]:

```
# Importing Libraries
```

In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

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

# Utility function to print the confusion matrix
def confusion_matrix(Y_true, Y_pred):
    Y_true = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_true, axis=1)])
    Y_pred = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_pred, axis=1)])

    return pd.crosstab(Y_true, Y_pred, rownames=['True'], colnames=['Pred'])
```

Data

In [3]:

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

In [4]:

```
# 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 [5]:

```
import os
os.chdir('C:/Users/kingsubham27091995/Desktop/AppliedAiCouse/CASE
STUDIES/HumanActivityRecognition/HAR')
```

In [6]:

```
# Utility function to read the data from csv file
def _read_csv(filename):
    return pd.read_csv(filename, delim_whitespace=True, header=None)
```

```python
# Utility function to load the load
def load_signals(subset):
    signals_data = []

    for signal in SIGNALS:
        filename = f'UCI_HAR_Dataset/{subset}/Inertial Signals/{signal}_{subset}.txt'
        signals_data.append(
            _read_csv(filename).as_matrix()
        )

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

In [7]:

```python
def load_y(subset):
    """
    The objective that we are trying to predict is a integer, from 1 to 6,
    that represents a human activity. We return a binary representation of
    every sample objective as a 6 bits vector using One Hot Encoding
    (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html)
    """
    filename = f'UCI_HAR_Dataset/{subset}/y_{subset}.txt'
    y = _read_csv(filename)[0]

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

In [8]:

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

    return X_train, X_test, y_train, y_test
```

In [9]:

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

In [10]:

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

In [11]:

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

```
Using TensorFlow backend.
```

In [12]:

```python
# Importing libraries
from keras.models import Sequential
```

```python
from keras.layers import LSTM
from keras.layers.core import Dense, Dropout
```

In [13]:

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

In [14]:

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

In [15]:

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

```
C:\Users\kingsubham27091995\Anaconda3\lib\site-packages\ipykernel_launcher.py:12: FutureWarning: M
ethod .as_matrix will be removed in a future version. Use .values instead.
  if sys.path[0] == '':
```

In [16]:

```python
timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = _count_classes(Y_train)

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

```
128
9
7352
```

1- Layer LSTM

In [17]:

```python
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden, 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()
```

```
WARNING:tensorflow:From C:\Users\kingsubham27091995\Anaconda3\lib\site-
packages\tensorflow\python\framework\op_def_library.py:263: colocate_with (from
tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From C:\Users\kingsubham27091995\Anaconda3\lib\site-
packages\keras\backend\tensorflow_backend.py:3445: calling dropout (from
tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed in a future
version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 32)                5376
```

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

In [18]:

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

In [19]:

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

```
WARNING:tensorflow:From C:\Users\kingsubham27091995\Anaconda3\lib\site-packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 41s 6ms/step - loss: 1.3306 - acc: 0.4361 - val_loss: 1.1743 - val_acc: 0.4723
Epoch 2/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.9716 - acc: 0.5788 - val_loss: 0.9656 - val_acc: 0.5263
Epoch 3/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.7787 - acc: 0.6510 - val_loss: 0.7841 - val_acc: 0.6135
Epoch 4/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.6911 - acc: 0.6587 - val_loss: 0.7109 - val_acc: 0.6203
Epoch 5/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.6496 - acc: 0.6794 - val_loss: 0.8000 - val_acc: 0.6362
Epoch 6/30
7352/7352 [==============================] - 44s 6ms/step - loss: 0.8295 - acc: 0.6221 - val_loss: 0.7970 - val_acc: 0.6318
Epoch 7/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.5920 - acc: 0.7122 - val_loss: 0.6816 - val_acc: 0.7038
Epoch 8/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.5472 - acc: 0.7594 - val_loss: 0.6473 - val_acc: 0.7258
Epoch 9/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.4838 - acc: 0.7805 - val_loss: 0.6033 - val_acc: 0.7445
Epoch 10/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.4212 - acc: 0.7907 - val_loss: 0.5396 - val_acc: 0.7472
Epoch 11/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.3967 - acc: 0.8084 - val_loss: 0.5399 - val_acc: 0.7526
Epoch 12/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.3609 - acc: 0.8407 - val_loss: 0.5420 - val_acc: 0.8222
Epoch 13/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.3312 - acc: 0.8798 - val_loss: 0.5038 - val_acc: 0.8673
Epoch 14/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.3000 - acc: 0.9053 - val_loss: 0.4823 - val_acc: 0.8758
Epoch 15/30
```

```
7352/7352 [==============================] - 42s 6ms/step - loss: 0.2574 - acc: 0.9230 - val_loss:
0.5465 - val_acc: 0.8731
Epoch 16/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.2383 - acc: 0.9293 - val_loss:
0.5054 - val_acc: 0.8694
Epoch 17/30
7352/7352 [==============================] - 39s 5ms/step - loss: 0.2271 - acc: 0.9313 - val_loss:
0.5409 - val_acc: 0.8653
Epoch 18/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.2121 - acc: 0.9353 - val_loss:
0.5611 - val_acc: 0.8677
Epoch 19/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.2129 - acc: 0.9355 - val_loss:
0.5293 - val_acc: 0.8887
Epoch 20/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1825 - acc: 0.9425 - val_loss:
0.4945 - val_acc: 0.8656
Epoch 21/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1857 - acc: 0.9400 - val_loss:
0.4470 - val_acc: 0.8836
Epoch 22/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1827 - acc: 0.9445 - val_loss:
0.7009 - val_acc: 0.8517
Epoch 23/30
7352/7352 [==============================] - 44s 6ms/step - loss: 0.1909 - acc: 0.9391 - val_loss:
0.5290 - val_acc: 0.8884
Epoch 24/30
7352/7352 [==============================] - 39s 5ms/step - loss: 0.1925 - acc: 0.9399 - val_loss:
0.6053 - val_acc: 0.8812
Epoch 25/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.1821 - acc: 0.9440 - val_loss:
0.4675 - val_acc: 0.8765
Epoch 26/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.1773 - acc: 0.9448 - val_loss:
0.5316 - val_acc: 0.8826
Epoch 27/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1817 - acc: 0.9406 - val_loss:
0.5282 - val_acc: 0.8843
Epoch 28/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.1966 - acc: 0.9402 - val_loss:
0.5390 - val_acc: 0.8931
Epoch 29/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.2257 - acc: 0.9291 - val_loss:
0.4739 - val_acc: 0.8948
Epoch 30/30
7352/7352 [==============================] - 44s 6ms/step - loss: 0.1745 - acc: 0.9442 - val_loss:
0.4692 - val_acc: 0.8806
```

Out[19]:

```
<keras.callbacks.History at 0x8cc27a668>
```

In [20]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 510        0         0        0                   0
SITTING                  0      398        66        0                   0
STANDING                 0      100       418        2                   0
WALKING                  0        3         0      465                   8
WALKING_DOWNSTAIRS       0        0         0        0                 360
WALKING_UPSTAIRS         0        3         0       23                   1

Pred                WALKING_UPSTAIRS
True
LAYING                            27
SITTING                           27
STANDING                          12
WALKING                           20
WALKING_DOWNSTAIRS                60
WALKING_UPSTAIRS                 444
```

In [21]:

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

```
2947/2947 [==============================] - 2s 532us/step
```

In [22]:

```
score
```

Out[22]:

```
[0.46922748649472224, 0.8805564981336953]
```

- With a simple 2 layer architecture we got 88.05% accuracy and a loss of 0.46
- We can further imporve the performace with Hyperparameter tuning

Hyperparameter Tuning with different DropOut Rates keeping LSTM Units=32

In [29]:

```
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV
from keras.models import Sequential
from keras.layers.core import Dense, Dropout
from keras.layers import LSTM
```

In [30]:

```
def build_model(units, rate):
    model = Sequential() # Initiliazing the sequential model
    model.add(LSTM(units=units, input_shape=(timesteps, input_dim))) # Configuring the parameters
    model.add(Dropout(rate=rate)) # Adding a dropout layer
    model.add(Dense(units=6, kernel_initializer='he_normal', activation='sigmoid')) # Adding a dens
e output layer with sigmoid activation
    model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])
#Compiling the model
    #model.summary()
    return model
```

In [35]:

```
model = KerasClassifier(build_fn = build_model)
parameters = {'units': [32],
              'rate': [0.25, 0.5,0.7]
              }
```

In [39]:

```
grid_search_CV = GridSearchCV(estimator = model,
                           param_grid = parameters,
                           cv = 3,
                           n_jobs=4)
grid_search = grid_search_CV.fit(X_train, Y_train, epochs=30)
```

```
Epoch 1/30
7352/7352 [==============================] - 25s 3ms/step - loss: 1.3526 - acc: 0.4433
Epoch 2/30
7352/7352 [==============================] - 20s 3ms/step - loss: 1.0738 - acc: 0.5226
Epoch 3/30
7352/7352 [==============================] - 20s 3ms/step - loss: 1.0310 - acc: 0.5574
Epoch 4/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.8162 - acc: 0.6265
Epoch 5/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.7050 - acc: 0.6714
Epoch 6/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.6713 - acc: 0.7178
```

```
Epoch 7/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.6267 - acc: 0.7594
Epoch 8/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.5190 - acc: 0.8051
Epoch 9/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.4214 - acc: 0.8628
Epoch 10/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.3776 - acc: 0.8848
Epoch 11/30
7352/7352 [==============================] - 21s 3ms/step - loss: 0.2947 - acc: 0.9117
Epoch 12/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.2533 - acc: 0.9197
Epoch 13/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.2463 - acc: 0.9267
Epoch 14/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.2160 - acc: 0.9320
Epoch 15/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.2013 - acc: 0.9327
Epoch 16/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.1869 - acc: 0.9355
Epoch 17/30
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1922 - acc: 0.9357
Epoch 18/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.1783 - acc: 0.9384
Epoch 19/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.1793 - acc: 0.9372
Epoch 20/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.1665 - acc: 0.9429
Epoch 21/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.1643 - acc: 0.9399
Epoch 22/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.1585 - acc: 0.9412
Epoch 23/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.1409 - acc: 0.9419
Epoch 24/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.1416 - acc: 0.9455
Epoch 25/30
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1419 - acc: 0.9482
Epoch 26/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.1490 - acc: 0.9490
Epoch 27/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.1401 - acc: 0.9495
Epoch 28/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.1365 - acc: 0.9482
Epoch 29/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.1320 - acc: 0.9510
Epoch 30/30
7352/7352 [==============================] - 20s 3ms/step - loss: 0.1304 - acc: 0.9510
```

In [41]:

```
print("Best: %f using %s" % (grid_search.best_score_, grid_search.best_params_))
means = grid_search.cv_results_['mean_test_score']
stds = grid_search.cv_results_['std_test_score']
params = grid_search.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

```
Best: 0.900027 using {'rate': 0.25, 'units': 32}
0.900027 (0.011693) with: {'rate': 0.25, 'units': 32}
0.803047 (0.064273) with: {'rate': 0.5, 'units': 32}
0.787677 (0.059909) with: {'rate': 0.7, 'units': 32}
```

After hyperparameter tuning :

1. Accuracy=90%
2. DropOut Rate=0.25(Best)
3. LSTM Units= 32

Defining few Functions

Plotting Confusion Matrix

In [42]:

```
# Plot Confusion Matrix
def plot_confusion_matrix_lstm(y_test, y_predict):
    result = confusion_matrix(y_test, y_predict)

    plt.figure(figsize=(12, 10))
    sns.heatmap(result,
                xticklabels= list(ACTIVITIES.values()),
                yticklabels=list(ACTIVITIES.values()),
                annot=True, fmt="d");
    plt.title("Confusion matrix")
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.show()
```

Plotting epochs vs Loss

In [43]:

```
# Plot train and cross validation loss
def plot_train_cv_loss(trained_model, epochs, colors=['b']):
    fig, ax = plt.subplots(1,1)
    ax.set_xlabel('epoch')
    ax.set_ylabel('Categorical Crossentropy Loss')
    x_axis_values = list(range(1,epochs+1))

    validation_loss = trained_model.history['val_loss']
    train_loss = trained_model.history['loss']

    ax.plot(x_axis_values, validation_loss, 'b', label="Validation Loss")
    ax.plot(x_axis_values, train_loss, 'r', label="Train Loss")
    plt.legend()
    plt.grid()
    fig.canvas.draw()
```

2- Layer LSTM's

In [46]:

```
# Initializing parameters
n_epochs = 30
n_batch = 16
n_classes = _count_classes(Y_train)

# Bias regularizer value - we will use elasticnet
# https://machinelearningmastery.com/use-weight-regularization-lstm-networks-time-series-forecasti
ng/
from keras.regularizers import L1L2
reg = L1L2(0.01, 0.01)
```

LSTM units=32 , Dropout= 0.50

In [51]:

```
from keras.layers import LSTM , BatchNormalization
from keras.layers.core import Dense, Dropout
# Model execution
model = Sequential()
model.add(LSTM(32, input_shape=(timesteps, input_dim), return_sequences=True,bias_regularizer=reg ))
```

```
model.add(LSTM(32, input_shape=(timesteps, input_dim), return_sequences=True,bias_regularizer=reg )
)
model.add(BatchNormalization())
model.add(Dropout(0.50))
model.add(LSTM(32))
model.add(Dropout(0.50))
model.add(Dense(n_classes, activation='sigmoid'))
model.compile(loss='categorical_crossentropy',
          optimizer='adam',
          metrics=['accuracy'])
# Training the model
trained_model  = model.fit(X_train,
                           Y_train,
                           batch_size=n_batch,
                           validation_data=(X_test, Y_test),
                           epochs=epochs)
```

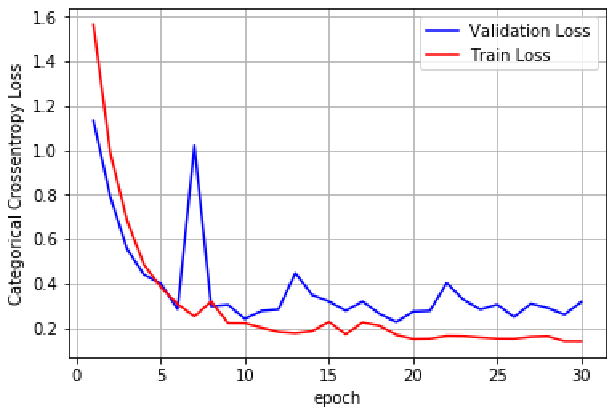
```
Model Summary:
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 91s 12ms/step - loss: 1.5640 - acc: 0.5994 - val_loss
: 1.1322 - val_acc: 0.6695
Epoch 2/30
7352/7352 [==============================] - 86s 12ms/step - loss: 0.9878 - acc: 0.7040 - val_loss
: 0.7904 - val_acc: 0.7323
Epoch 3/30
7352/7352 [==============================] - 84s 11ms/step - loss: 0.6829 - acc: 0.7848 - val_loss
: 0.5545 - val_acc: 0.7788
Epoch 4/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.4845 - acc: 0.8428 - val_loss
: 0.4397 - val_acc: 0.8602
Epoch 5/30
7352/7352 [==============================] - 93s 13ms/step - loss: 0.3833 - acc: 0.8862 - val_loss
: 0.4002 - val_acc: 0.8599
Epoch 6/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.3076 - acc: 0.9115 - val_loss
: 0.2842 - val_acc: 0.8985
Epoch 7/30
7352/7352 [==============================] - 88s 12ms/step - loss: 0.2519 - acc: 0.9261 - val_loss
: 1.0203 - val_acc: 0.7116
Epoch 8/30
7352/7352 [==============================] - 86s 12ms/step - loss: 0.3189 - acc: 0.9037 - val_loss
: 0.2970 - val_acc: 0.8958
Epoch 9/30
7352/7352 [==============================] - 86s 12ms/step - loss: 0.2219 - acc: 0.9268 - val_loss
: 0.3042 - val_acc: 0.8958
Epoch 10/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.2215 - acc: 0.9302 - val_loss
: 0.2411 - val_acc: 0.9101
Epoch 11/30
7352/7352 [==============================] - 86s 12ms/step - loss: 0.2006 - acc: 0.9339 - val_loss
: 0.2770 - val_acc: 0.9036
Epoch 12/30
7352/7352 [==============================] - 87s 12ms/step - loss: 0.1817 - acc: 0.9378 - val_loss
: 0.2839 - val_acc: 0.9125
Epoch 13/30
7352/7352 [==============================] - 87s 12ms/step - loss: 0.1762 - acc: 0.9403 - val_loss
: 0.4473 - val_acc: 0.8728
Epoch 14/30
7352/7352 [==============================] - 93s 13ms/step - loss: 0.1854 - acc: 0.9389 - val_loss
: 0.3484 - val_acc: 0.8867
Epoch 15/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.2271 - acc: 0.9293 - val_loss
: 0.3191 - val_acc: 0.8911
Epoch 16/30
7352/7352 [==============================] - 94s 13ms/step - loss: 0.1714 - acc: 0.9391 - val_loss
: 0.2779 - val_acc: 0.8958
Epoch 17/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.2248 - acc: 0.9174 - val_loss
: 0.3189 - val_acc: 0.8951
Epoch 18/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.2096 - acc: 0.9295 - val_loss
: 0.2639 - val_acc: 0.9087
Epoch 19/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1689 - acc: 0.9406 - val_loss
: 0.2267 - val_acc: 0.9192
Epoch 20/30
```

```
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1505 - acc: 0.9423 - val_loss
: 0.2738 - val_acc: 0.9118
Epoch 21/30
7352/7352 [==============================] - 83s 11ms/step - loss: 0.1518 - acc: 0.9426 - val_loss
: 0.2770 - val_acc: 0.9101
Epoch 22/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1641 - acc: 0.9423 - val_loss
: 0.4021 - val_acc: 0.8643
Epoch 23/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1631 - acc: 0.9429 - val_loss
: 0.3267 - val_acc: 0.9104
Epoch 24/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1572 - acc: 0.9433 - val_loss
: 0.2833 - val_acc: 0.9172
Epoch 25/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1519 - acc: 0.9444 - val_loss
: 0.3053 - val_acc: 0.9141
Epoch 26/30
7352/7352 [==============================] - 83s 11ms/step - loss: 0.1513 - acc: 0.9426 - val_loss
: 0.2493 - val_acc: 0.9165
Epoch 27/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1599 - acc: 0.9423 - val_loss
: 0.3090 - val_acc: 0.9060
Epoch 28/30
7352/7352 [==============================] - 83s 11ms/step - loss: 0.1628 - acc: 0.9436 - val_loss
: 0.2902 - val_acc: 0.8975
Epoch 29/30
7352/7352 [==============================] - 81s 11ms/step - loss: 0.1402 - acc: 0.9484 - val_loss
: 0.2604 - val_acc: 0.9114
Epoch 30/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1399 - acc: 0.9479 - val_loss
: 0.3167 - val_acc: 0.9050
```

Plotting Epochs vs Loss

In [54]:

```
% matplotlib inline
import matplotlib.pyplot as plt

# Plot train and cross validation error
plot_train_cv_loss(trained_model, epochs)
```



From epoch 10, we starts to overfit the model, so best value for epoch is 10

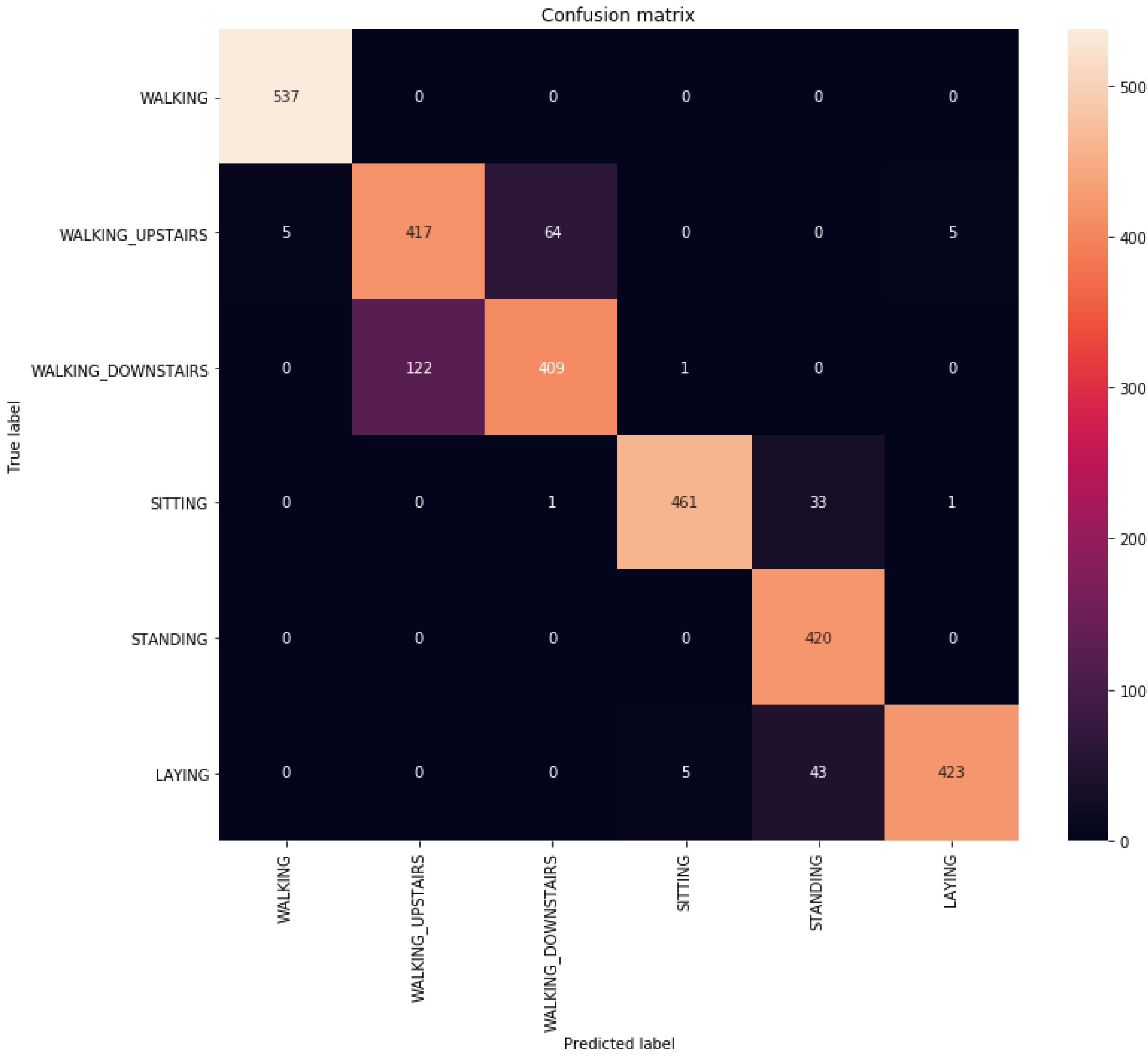
Plotting Confusion Matrix

In [61]:

```
import seaborn as sns

scores = model.evaluate(X_test, Y_test, verbose=0)
print("Test Accuracy: %f%%" % (scores[1]*100))

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

```
Y_predictions = pd.Series([ACTIVITIES[y] for y in np.argmax(model.predict(X_test), axis=1)])

# Confusion Matrix
plot_confusion_matrix_lstm(Y_true, Y_predictions)
```

```
Test Accuracy: 90.498812%
```



> **LSTM Units= 32 , DropOut= 0.25**

In [62]:

```
from keras.layers import LSTM , BatchNormalization
from keras.layers.core import Dense, Dropout
# Model execution
model = Sequential()
model.add(LSTM(32, input_shape=(timesteps, input_dim), return_sequences=True,bias_regularizer=reg )
)
model.add(BatchNormalization())
model.add(Dropout(0.25))
model.add(LSTM(32))
model.add(Dropout(0.25))
model.add(Dense(n_classes, activation='sigmoid'))
model.compile(loss='categorical_crossentropy',
          optimizer='adam',
          metrics=['accuracy'])
# Training the model
trained_model  = model.fit(X_train,
                           Y_train,
                           batch_size=n_batch,
                           validation_data=(X_test, Y_test),
```

```
                            epochs=epochs)
```

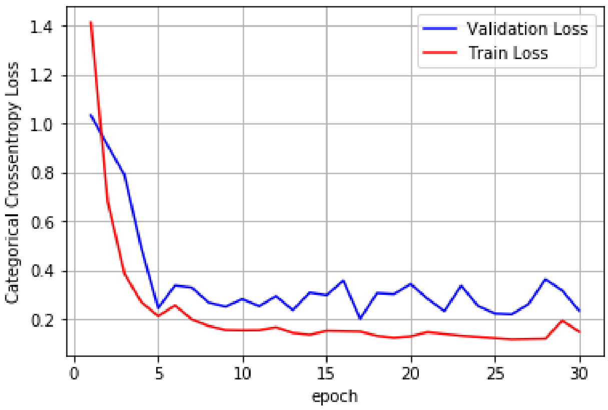
```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 85s 12ms/step - loss: 1.4139 - acc: 0.6511 - val_loss
: 1.0345 - val_acc: 0.7733
Epoch 2/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.6820 - acc: 0.8478 - val_loss
: 0.9108 - val_acc: 0.7353
Epoch 3/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.3863 - acc: 0.9063 - val_loss
: 0.7907 - val_acc: 0.7251
Epoch 4/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.2706 - acc: 0.9234 - val_loss
: 0.4915 - val_acc: 0.8409
Epoch 5/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.2132 - acc: 0.9317 - val_loss
: 0.2458 - val_acc: 0.9108
Epoch 6/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.2573 - acc: 0.9195 - val_loss
: 0.3387 - val_acc: 0.8778
Epoch 7/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.2000 - acc: 0.9347 - val_loss
: 0.3301 - val_acc: 0.8785
Epoch 8/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1730 - acc: 0.9393 - val_loss
: 0.2687 - val_acc: 0.8989
Epoch 9/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1561 - acc: 0.9425 - val_loss
: 0.2519 - val_acc: 0.9053
Epoch 10/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1549 - acc: 0.9423 - val_loss
: 0.2831 - val_acc: 0.9013
Epoch 11/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1554 - acc: 0.9406 - val_loss
: 0.2537 - val_acc: 0.9067
Epoch 12/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1676 - acc: 0.9381 - val_loss
: 0.2953 - val_acc: 0.8992
Epoch 13/30
7352/7352 [==============================] - 83s 11ms/step - loss: 0.1447 - acc: 0.9475 - val_loss
: 0.2375 - val_acc: 0.9192
Epoch 14/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1370 - acc: 0.9472 - val_loss
: 0.3096 - val_acc: 0.9036
Epoch 15/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1534 - acc: 0.9395 - val_loss
: 0.2989 - val_acc: 0.9030
Epoch 16/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1520 - acc: 0.9412 - val_loss
: 0.3587 - val_acc: 0.8846
Epoch 17/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1505 - acc: 0.9402 - val_loss
: 0.2024 - val_acc: 0.9169
Epoch 18/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1323 - acc: 0.9472 - val_loss
: 0.3078 - val_acc: 0.9040
Epoch 19/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1245 - acc: 0.9513 - val_loss
: 0.3032 - val_acc: 0.9043
Epoch 20/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1304 - acc: 0.9489 - val_loss
: 0.3449 - val_acc: 0.8765
Epoch 21/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1481 - acc: 0.9437 - val_loss
: 0.2843 - val_acc: 0.9118
Epoch 22/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1401 - acc: 0.9486 - val_loss
: 0.2339 - val_acc: 0.9111
Epoch 23/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1330 - acc: 0.9490 - val_loss
: 0.3379 - val_acc: 0.8856
Epoch 24/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1280 - acc: 0.9504 - val_loss
: 0.2549 - val_acc: 0.9148
Epoch 25/30
```

```
7352/7352 [==============================] - 83s 11ms/step - loss: 0.1229 - acc: 0.9482 - val_loss
: 0.2237 - val_acc: 0.9284
Epoch 26/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1180 - acc: 0.9524 - val_loss
: 0.2205 - val_acc: 0.9332
Epoch 27/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1195 - acc: 0.9510 - val_loss
: 0.2624 - val_acc: 0.9179
Epoch 28/30
7352/7352 [==============================] - 83s 11ms/step - loss: 0.1209 - acc: 0.9504 - val_loss
: 0.3642 - val_acc: 0.8700
Epoch 29/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1951 - acc: 0.9328 - val_loss
: 0.3181 - val_acc: 0.9043
Epoch 30/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.1496 - acc: 0.9460 - val_loss
: 0.2358 - val_acc: 0.9148
```

Plotting epochs vs Loss

In [63]:

```
# Plot train and cross validation error
plot_train_cv_loss(trained_model, epochs)
```

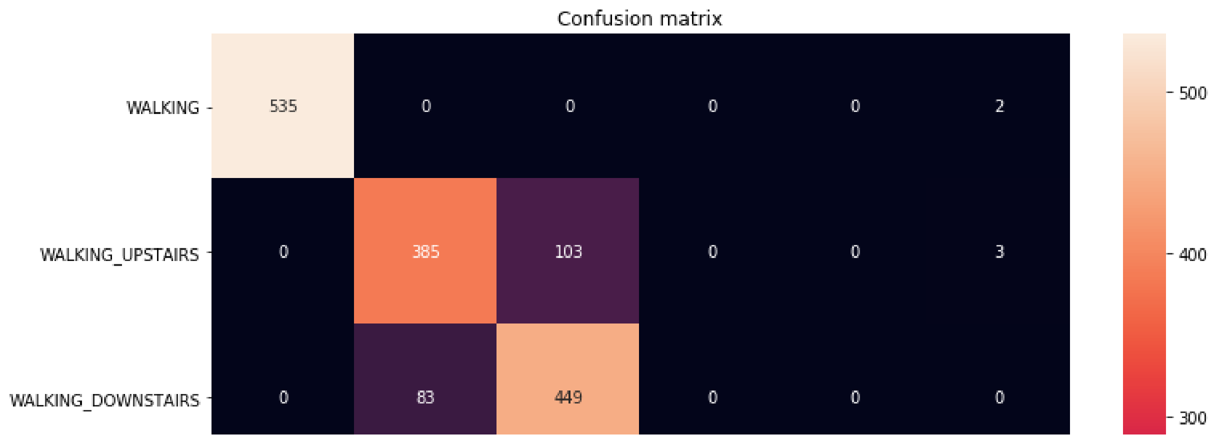


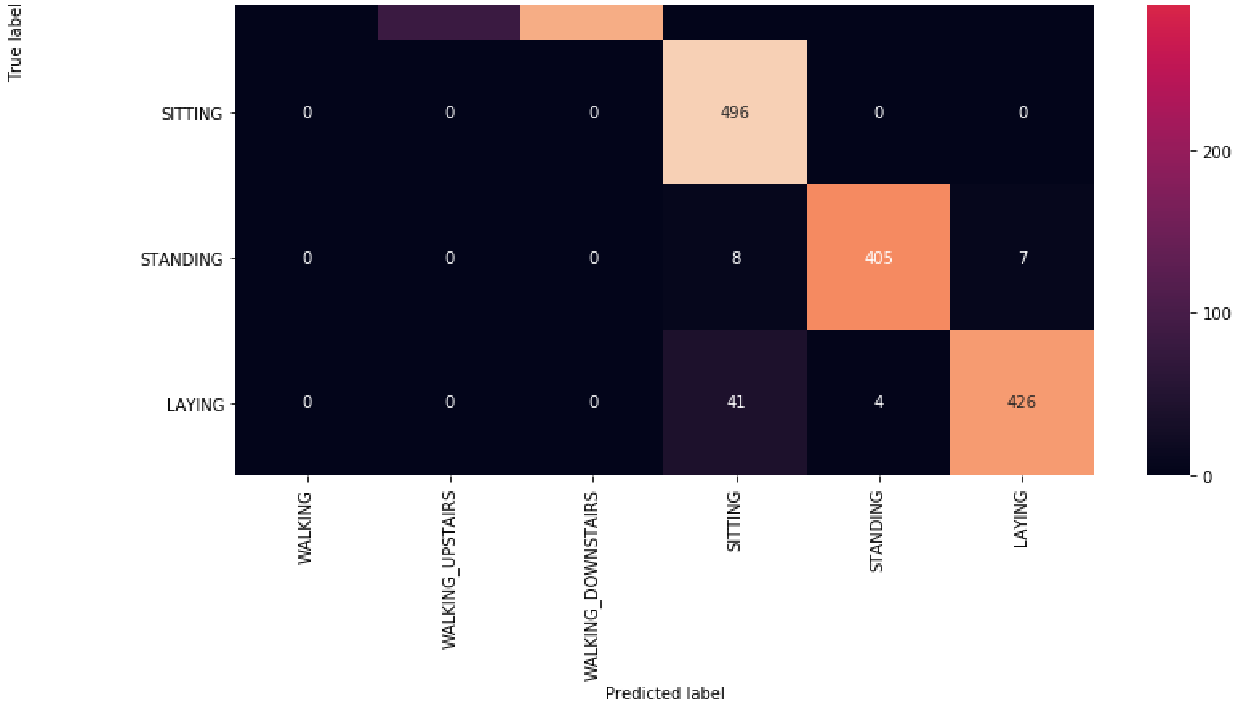
Plotting Confusion Matrix

In [64]:

```
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Test Accuracy: %f%%" % (scores[1]*100))
Y_true = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_test, axis=1)])
Y_predictions = pd.Series([ACTIVITIES[y] for y in np.argmax(model.predict(X_test), axis=1)])

# Confusion Matrix
plot_confusion_matrix_lstm(Y_true, Y_predictions)
```

```
Test Accuracy: 91.482864%
```





> **LSTM Units= 64, Dropout =0.50**

In [65]:

```
# Model execution
model = Sequential()
model.add(LSTM(64, input_shape=(timesteps, input_dim), return_sequences=True, bias_regularizer=reg)
)
model.add(BatchNormalization())
model.add(Dropout(0.50))
model.add(LSTM(50))
model.add(Dropout(0.50))
model.add(Dense(n_classes, activation='sigmoid'))
model.compile(loss='categorical_crossentropy',
          optimizer='adam',
          metrics=['accuracy'])
# Training the model
trained_model  = model.fit(X_train,
                           Y_train,
                           batch_size=n_batch,
                           validation_data=(X_test, Y_test),
                           epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 135s 18ms/step - loss: 1.8012 - acc: 0.6477 - val_los
s: 1.3358 - val_acc: 0.6956
Epoch 2/30
7352/7352 [==============================] - 109s 15ms/step - loss: 0.8502 - acc: 0.8311 - val_los
s: 0.5815 - val_acc: 0.8833
Epoch 3/30
7352/7352 [==============================] - 106s 14ms/step - loss: 0.3816 - acc: 0.9064 - val_los
s: 0.3977 - val_acc: 0.8778
Epoch 4/30
7352/7352 [==============================] - 105s 14ms/step - loss: 0.2546 - acc: 0.9170 - val_los
s: 0.2446 - val_acc: 0.9138
Epoch 5/30
7352/7352 [==============================] - 106s 14ms/step - loss: 0.2292 - acc: 0.9223 - val_los
s: 0.5429 - val_acc: 0.8578
Epoch 6/30
7352/7352 [==============================] - 109s 15ms/step - loss: 0.2293 - acc: 0.9212 - val_los
s: 0.2791 - val_acc: 0.8989
Epoch 7/30
7352/7352 [==============================] - 128s 17ms/step - loss: 0.1958 - acc: 0.9300 - val_los
s: 0.3770 - val_acc: 0.8972
Epoch 8/30
```

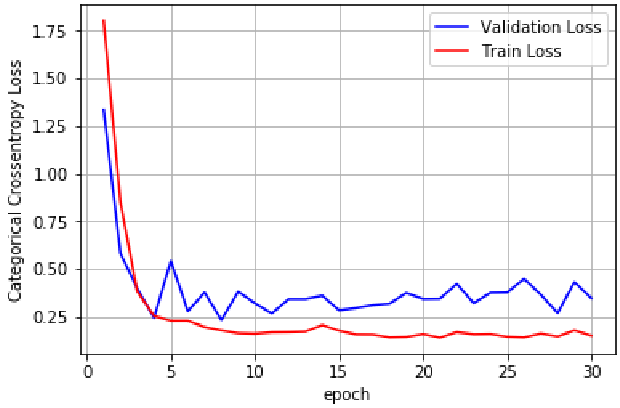
```
7352/7352 [==============================] - 120s 16ms/step - loss: 0.1788 - acc: 0.9357 - val_loss: 0.2333 - val_acc: 0.9114
Epoch 9/30
7352/7352 [==============================] - 114s 16ms/step - loss: 0.1640 - acc: 0.9400 - val_loss: 0.3812 - val_acc: 0.8894
Epoch 10/30
7352/7352 [==============================] - 116s 16ms/step - loss: 0.1614 - acc: 0.9411 - val_loss: 0.3206 - val_acc: 0.9040
Epoch 11/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1692 - acc: 0.9328 - val_loss: 0.2678 - val_acc: 0.9128
Epoch 12/30
7352/7352 [==============================] - 105s 14ms/step - loss: 0.1701 - acc: 0.9342 - val_loss: 0.3423 - val_acc: 0.8979
Epoch 13/30
7352/7352 [==============================] - 105s 14ms/step - loss: 0.1727 - acc: 0.9310 - val_loss: 0.3423 - val_acc: 0.9046
Epoch 14/30
7352/7352 [==============================] - 106s 14ms/step - loss: 0.2059 - acc: 0.9240 - val_loss: 0.3600 - val_acc: 0.9063
Epoch 15/30
7352/7352 [==============================] - 105s 14ms/step - loss: 0.1771 - acc: 0.9366 - val_loss: 0.2842 - val_acc: 0.9148
Epoch 16/30
7352/7352 [==============================] - 105s 14ms/step - loss: 0.1573 - acc: 0.9354 - val_loss: 0.2961 - val_acc: 0.9145
Epoch 17/30
7352/7352 [==============================] - 105s 14ms/step - loss: 0.1566 - acc: 0.9425 - val_loss: 0.3109 - val_acc: 0.9067
Epoch 18/30
7352/7352 [==============================] - 105s 14ms/step - loss: 0.1420 - acc: 0.9450 - val_loss: 0.3188 - val_acc: 0.9175
Epoch 19/30
7352/7352 [==============================] - 104s 14ms/step - loss: 0.1440 - acc: 0.9453 - val_loss: 0.3745 - val_acc: 0.9104
Epoch 20/30
7352/7352 [==============================] - 105s 14ms/step - loss: 0.1590 - acc: 0.9387 - val_loss: 0.3429 - val_acc: 0.9067
Epoch 21/30
7352/7352 [==============================] - 104s 14ms/step - loss: 0.1404 - acc: 0.9437 - val_loss: 0.3440 - val_acc: 0.9084
Epoch 22/30
7352/7352 [==============================] - 105s 14ms/step - loss: 0.1696 - acc: 0.9381 - val_loss: 0.4221 - val_acc: 0.9080
Epoch 23/30
7352/7352 [==============================] - 104s 14ms/step - loss: 0.1579 - acc: 0.9416 - val_loss: 0.3209 - val_acc: 0.9023
Epoch 24/30
7352/7352 [==============================] - 104s 14ms/step - loss: 0.1594 - acc: 0.9376 - val_loss: 0.3756 - val_acc: 0.8945
Epoch 25/30
7352/7352 [==============================] - 106s 14ms/step - loss: 0.1450 - acc: 0.9433 - val_loss: 0.3771 - val_acc: 0.9026
Epoch 26/30
7352/7352 [==============================] - 105s 14ms/step - loss: 0.1417 - acc: 0.9408 - val_loss: 0.4485 - val_acc: 0.9009
Epoch 27/30
7352/7352 [==============================] - 104s 14ms/step - loss: 0.1621 - acc: 0.9391 - val_loss: 0.3651 - val_acc: 0.9046
Epoch 28/30
7352/7352 [==============================] - 105s 14ms/step - loss: 0.1465 - acc: 0.9444 - val_loss: 0.2696 - val_acc: 0.9158
Epoch 29/30
7352/7352 [==============================] - 105s 14ms/step - loss: 0.1790 - acc: 0.9392 - val_loss: 0.4313 - val_acc: 0.8778
Epoch 30/30
7352/7352 [==============================] - 105s 14ms/step - loss: 0.1511 - acc: 0.9440 - val_loss: 0.3471 - val_acc: 0.9128
```

Plotting Epochs vs Loss

In [66]:

```
# Plot train and cross validation error
plot_train_cv_loss(trained_model, epochs)
```

```
plot_train_cv_loss(trained_model, epochs)
```



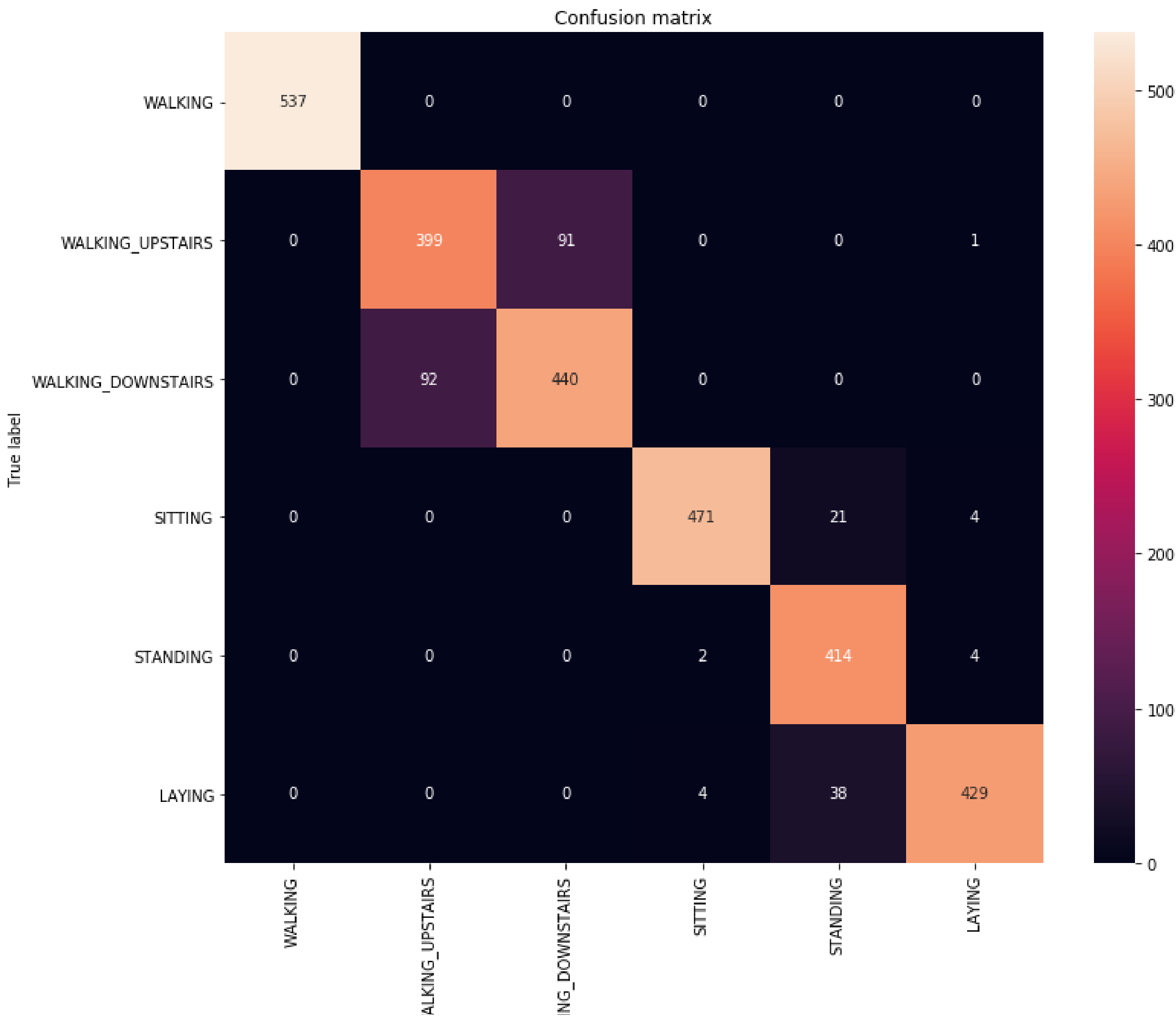
From epoch 8, we starts to overfit the model, so best value for epoch is 8

Plotting Confusion Matrix

In [67]:

```
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Test Accuracy: %f%%" % (scores[1]*100))
Y_true = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_test, axis=1)])
Y_predictions = pd.Series([ACTIVITIES[y] for y in np.argmax(model.predict(X_test), axis=1)])

# Confusion Matrix
plot_confusion_matrix_lstm(Y_true, Y_predictions)
```

```
Test Accuracy: 91.279267%
```



Let's try by increasing the Dropout Rate to 0.7

> **LSTM Units= 32, Dropout =0.7**

In [68]:

```
# Model execution
model = Sequential()
model.add(LSTM(32, input_shape=(timesteps, input_dim), return_sequences=True, bias_regularizer=reg)
)
model.add(BatchNormalization())
model.add(Dropout(0.70))
model.add(LSTM(32))
model.add(Dropout(0.70))
model.add(Dense(n_classes, activation='sigmoid'))
model.compile(loss='categorical_crossentropy',
          optimizer='adam',
          metrics=['accuracy'])
# Training the model
trained_model  = model.fit(X_train,
                           Y_train,
                           batch_size=n_batch,
                           validation_data=(X_test, Y_test),
                           epochs=epochs)
```

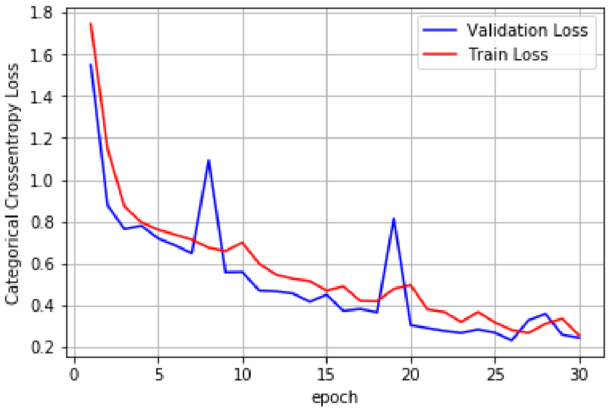
```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 102s 14ms/step - loss: 1.7457 - acc: 0.5113 - val_los
s: 1.5490 - val_acc: 0.5439
Epoch 2/30
7352/7352 [==============================] - 90s 12ms/step - loss: 1.1467 - acc: 0.5977 - val_loss
: 0.8771 - val_acc: 0.6759
Epoch 3/30
7352/7352 [==============================] - 97s 13ms/step - loss: 0.8728 - acc: 0.6338 - val_loss
: 0.7650 - val_acc: 0.6793
Epoch 4/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.7970 - acc: 0.6375 - val_loss
: 0.7806 - val_acc: 0.6861
Epoch 5/30
7352/7352 [==============================] - 101s 14ms/step - loss: 0.7615 - acc: 0.6462 - val_los
s: 0.7199 - val_acc: 0.6162
Epoch 6/30
7352/7352 [==============================] - 86s 12ms/step - loss: 0.7379 - acc: 0.6513 - val_loss
: 0.6871 - val_acc: 0.6695
Epoch 7/30
7352/7352 [==============================] - 85s 12ms/step - loss: 0.7141 - acc: 0.6759 - val_loss
: 0.6486 - val_acc: 0.6644
Epoch 8/30
7352/7352 [==============================] - 84s 11ms/step - loss: 0.6753 - acc: 0.7042 - val_loss
: 1.0930 - val_acc: 0.6006
Epoch 9/30
7352/7352 [==============================] - 84s 11ms/step - loss: 0.6596 - acc: 0.7254 - val_loss
: 0.5583 - val_acc: 0.7933
Epoch 10/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.6998 - acc: 0.7379 - val_loss
: 0.5596 - val_acc: 0.7998
Epoch 11/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.5990 - acc: 0.7820 - val_loss
: 0.4708 - val_acc: 0.8331
Epoch 12/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.5468 - acc: 0.7947 - val_loss
: 0.4673 - val_acc: 0.8263
Epoch 13/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.5276 - acc: 0.7998 - val_loss
: 0.4570 - val_acc: 0.8453
Epoch 14/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.5161 - acc: 0.8066 - val_loss
: 0.4172 - val_acc: 0.8351
```

```
Epoch 15/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.4707 - acc: 0.8334 - val_loss
: 0.4509 - val_acc: 0.8456
Epoch 16/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.4905 - acc: 0.8176 - val_loss
: 0.3733 - val_acc: 0.8734
Epoch 17/30
7352/7352 [==============================] - 79s 11ms/step - loss: 0.4215 - acc: 0.8515 - val_loss
: 0.3827 - val_acc: 0.8802
Epoch 18/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.4197 - acc: 0.8538 - val_loss
: 0.3661 - val_acc: 0.8694
Epoch 19/30
7352/7352 [==============================] - 84s 11ms/step - loss: 0.4790 - acc: 0.8413 - val_loss
: 0.8157 - val_acc: 0.7309
Epoch 20/30
7352/7352 [==============================] - 95s 13ms/step - loss: 0.4973 - acc: 0.8194 - val_loss
: 0.3054 - val_acc: 0.8975
Epoch 21/30
7352/7352 [==============================] - 87s 12ms/step - loss: 0.3802 - acc: 0.8709 - val_loss
: 0.2906 - val_acc: 0.8965
Epoch 22/30
7352/7352 [==============================] - 82s 11ms/step - loss: 0.3681 - acc: 0.8743 - val_loss
: 0.2776 - val_acc: 0.9050
Epoch 23/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.3200 - acc: 0.8985 - val_loss
: 0.2678 - val_acc: 0.9101
Epoch 24/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.3680 - acc: 0.8781 - val_loss
: 0.2844 - val_acc: 0.9050
Epoch 25/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.3177 - acc: 0.9026 - val_loss
: 0.2692 - val_acc: 0.9172
Epoch 26/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.2819 - acc: 0.9089 - val_loss
: 0.2317 - val_acc: 0.9277
Epoch 27/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.2675 - acc: 0.9163 - val_loss
: 0.3284 - val_acc: 0.8856
Epoch 28/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.3114 - acc: 0.8998 - val_loss
: 0.3591 - val_acc: 0.8778
Epoch 29/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.3362 - acc: 0.9000 - val_loss
: 0.2591 - val_acc: 0.9128
Epoch 30/30
7352/7352 [==============================] - 80s 11ms/step - loss: 0.2562 - acc: 0.9163 - val_loss
: 0.2447 - val_acc: 0.9172
```

Plotting epochs vs Loss

In [69]:

```
# Plot train and cross validation error
plot_train_cv_loss(trained_model, epochs)
```



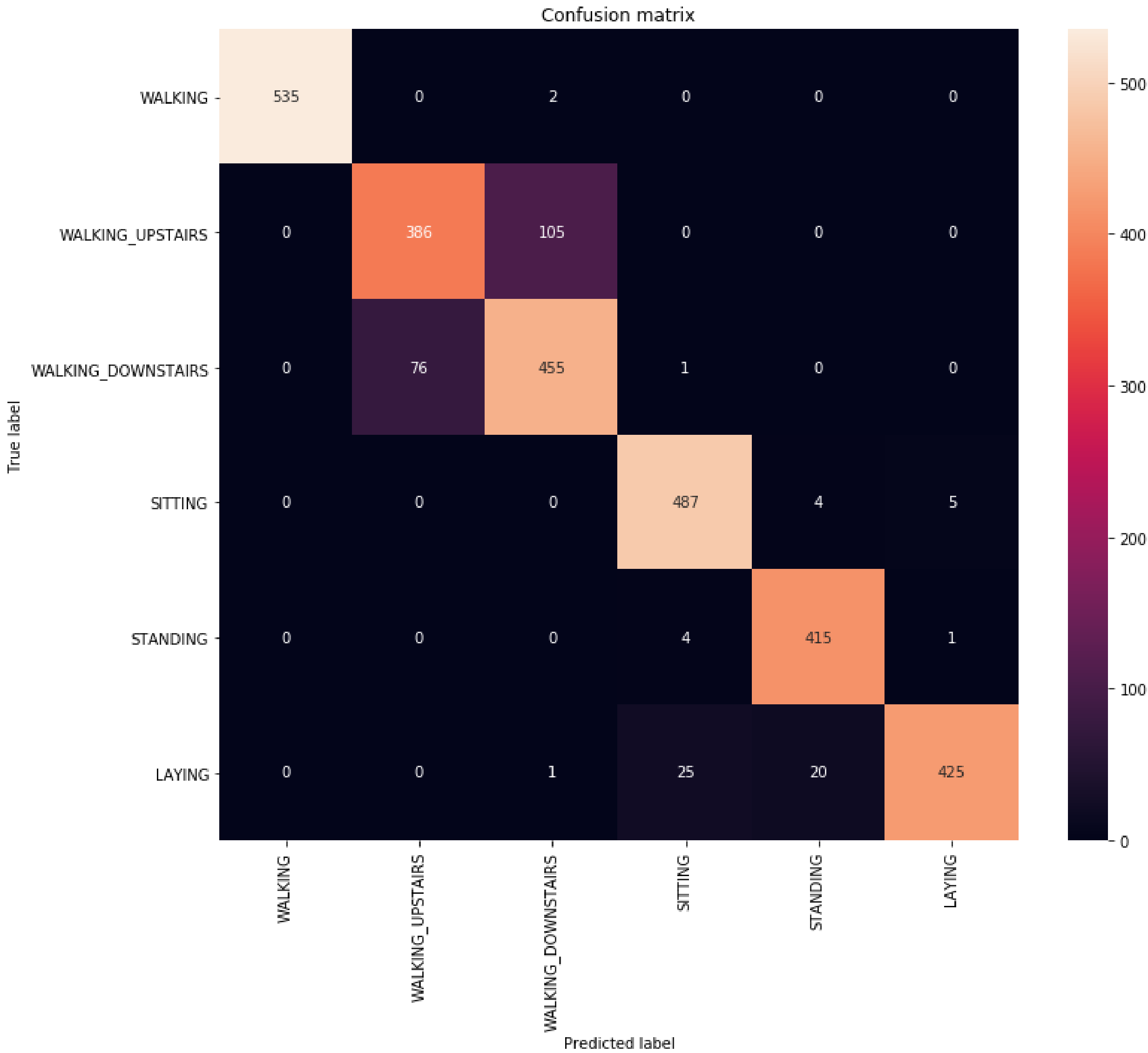
From epoch 9, we starts to overfit the model, so best value for epoch is 9

Plotting Confusion Matrix

In [70]:

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

Y_true = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_test, axis=1)])
Y_predictions = pd.Series([ACTIVITIES[y] for y in np.argmax(model.predict(X_test), axis=1)])

# Confusion Matrix
plot_confusion_matrix_lstm(Y_true, Y_predictions)
```

```
Test Accuracy: 91.720394%
```



> **LSTM Units = 64 with Dropout= 0.7**

In [71]:

```
# Model execution
model = Sequential()
model.add(LSTM(64, input_shape=(timesteps, input_dim), return_sequences=True, bias_regularizer=reg)
)
model.add(BatchNormalization())
model.add(Dropout(0.70))
model.add(LSTM(50))
```

```
model.add(Dropout(0.70))
model.add(Dense(n_classes, activation='sigmoid'))
model.compile(loss='categorical_crossentropy',
          optimizer='adam',
          metrics=['accuracy'])
# Training the model
trained_model  = model.fit(X_train,
                           Y_train,
                           batch_size=n_batch,
                           validation_data=(X_test, Y_test),
                           epochs=epochs)
```

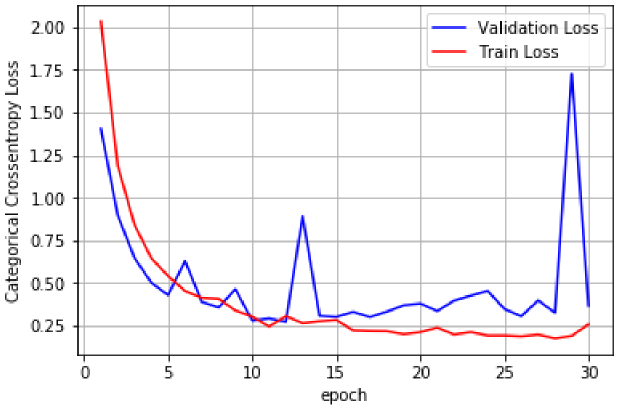
```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 113s 15ms/step - loss: 2.0342 - acc: 0.5740 - val_los
s: 1.4051 - val_acc: 0.6586
Epoch 2/30
7352/7352 [==============================] - 106s 14ms/step - loss: 1.1879 - acc: 0.6488 - val_los
s: 0.8965 - val_acc: 0.7048
Epoch 3/30
7352/7352 [==============================] - 119s 16ms/step - loss: 0.8405 - acc: 0.6725 - val_los
s: 0.6462 - val_acc: 0.7543
Epoch 4/30
7352/7352 [==============================] - 113s 15ms/step - loss: 0.6449 - acc: 0.7360 - val_los
s: 0.5011 - val_acc: 0.8107
Epoch 5/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.5405 - acc: 0.8112 - val_los
s: 0.4277 - val_acc: 0.8429
Epoch 6/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.4521 - acc: 0.8570 - val_los
s: 0.6293 - val_acc: 0.7587
Epoch 7/30
7352/7352 [==============================] - 124s 17ms/step - loss: 0.4124 - acc: 0.8770 - val_los
s: 0.3884 - val_acc: 0.8599
Epoch 8/30
7352/7352 [==============================] - 117s 16ms/step - loss: 0.4065 - acc: 0.8648 - val_los
s: 0.3564 - val_acc: 0.8846
Epoch 9/30
7352/7352 [==============================] - 107s 15ms/step - loss: 0.3385 - acc: 0.8949 - val_los
s: 0.4629 - val_acc: 0.8331
Epoch 10/30
7352/7352 [==============================] - 117s 16ms/step - loss: 0.3020 - acc: 0.9066 - val_los
s: 0.2786 - val_acc: 0.9080
Epoch 11/30
7352/7352 [==============================] - 111s 15ms/step - loss: 0.2441 - acc: 0.9246 - val_los
s: 0.2918 - val_acc: 0.9043
Epoch 12/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.3051 - acc: 0.9041 - val_los
s: 0.2715 - val_acc: 0.8951
Epoch 13/30
7352/7352 [==============================] - 110s 15ms/step - loss: 0.2637 - acc: 0.9161 - val_los
s: 0.8909 - val_acc: 0.7913
Epoch 14/30
7352/7352 [==============================] - 117s 16ms/step - loss: 0.2754 - acc: 0.9129 - val_los
s: 0.3068 - val_acc: 0.8945
Epoch 15/30
7352/7352 [==============================] - 113s 15ms/step - loss: 0.2824 - acc: 0.9149 - val_los
s: 0.3008 - val_acc: 0.8985
Epoch 16/30
7352/7352 [==============================] - 107s 15ms/step - loss: 0.2209 - acc: 0.9275 - val_los
s: 0.3298 - val_acc: 0.8945
Epoch 17/30
7352/7352 [==============================] - 107s 15ms/step - loss: 0.2179 - acc: 0.9251 - val_los
s: 0.2996 - val_acc: 0.9084
Epoch 18/30
7352/7352 [==============================] - 107s 15ms/step - loss: 0.2168 - acc: 0.9248 - val_los
s: 0.3311 - val_acc: 0.9019
Epoch 19/30
7352/7352 [==============================] - 106s 14ms/step - loss: 0.1991 - acc: 0.9306 - val_los
s: 0.3686 - val_acc: 0.8860
Epoch 20/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.2114 - acc: 0.9317 - val_los
s: 0.3788 - val_acc: 0.8945
Epoch 21/30
7352/7352 [==============================] - 107s 15ms/step - loss: 0.2372 - acc: 0.9225 - val_los
s: 0.3350 - val_acc: 0.8989
Epoch 22/30
```

```
Epoch 22/30
7352/7352 [==============================] - 107s 14ms/step - loss: 0.1970 - acc: 0.9382 - val_los
s: 0.3956 - val_acc: 0.8744
Epoch 23/30
7352/7352 [==============================] - 107s 15ms/step - loss: 0.2116 - acc: 0.9320 - val_los
s: 0.4254 - val_acc: 0.8653
Epoch 24/30
7352/7352 [==============================] - 107s 15ms/step - loss: 0.1911 - acc: 0.9380 - val_los
s: 0.4528 - val_acc: 0.8867
Epoch 25/30
7352/7352 [==============================] - 107s 15ms/step - loss: 0.1911 - acc: 0.9351 - val_los
s: 0.3471 - val_acc: 0.8958
Epoch 26/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1862 - acc: 0.9368 - val_los
s: 0.3041 - val_acc: 0.9057
Epoch 27/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1965 - acc: 0.9357 - val_los
s: 0.3977 - val_acc: 0.8992
Epoch 28/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1743 - acc: 0.9414 - val_los
s: 0.3247 - val_acc: 0.9131
Epoch 29/30
7352/7352 [==============================] - 108s 15ms/step - loss: 0.1890 - acc: 0.9359 - val_los
s: 1.7274 - val_acc: 0.6518
Epoch 30/30
7352/7352 [==============================] - 107s 15ms/step - loss: 0.2571 - acc: 0.9180 - val_los
s: 0.3656 - val_acc: 0.8968
```

Plotting epochs vs Loss

In [72]:

```
# Plot train and cross validation error
plot_train_cv_loss(trained_model, epochs)
```



Plotting Confusion Matrix

In [ ]:

```
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Test Accuracy: %f%%" % (scores[1]*100))
Y_true = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_test, axis=1)])
Y_predictions = pd.Series([ACTIVITIES[y] for y in np.argmax(model.predict(Y_test), axis=1)])

# Confusion Matrix
plot_confusion_matrix_lstm(Y_true, Y_predictions)
```

Pretty Table

In [ ]:

```
from prettytable import PrettyTable
ptable = PrettyTable()
ptable.title = " Model Comparision "
```

```
ptable.field_names = ["LSTM Layers",'LSTM Units','Dropout', 'Test Accuracy']
ptable.add_row(["1","32","0.50","88%"])
ptable.add_row(["1","32","0.25","90%"])
ptable.add_row(["2","32","0.50","90.49%"])
ptable.add_row(["2","32","0.25","91.48%"])
ptable.add_row(["2","64","0.50","91.28%"])
ptable.add_row(["2","32","0.70","91.72%"])
ptable.add_row(["2","64","0.70","89.68"])
print(ptable)
```

Conclusion:

1. It is very easy to Overfit in LSTM, since the number of datapoints is small and No. of Parameters is Large. So, we can't train much complex networks.
2. To prevent Overfiting we use the DropOut Layer. We checked using different DropOut Rates, to make out model perform well.
3. The best Accuracy=91.72% which is produced when we gave LSTM Units=32, LSTM Layers=2, DropOut rate=0.70.
4. Tuning the Hyperparameter is very important to get better results.
5. Deep Learning Models performed fairly well, but Feature engineered ML Models gave better accuracy.