Importing Libraries

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sn
```

In [2]:

```
%matplotlib inline
import matplotlib.pyplot as plt
def plt_dynamic(x, vy, ty, ax, colors=['b']):
    ax.plot(x, vy, 'b', label="Validation Loss")
    ax.plot(x, ty, 'r', label="Train Loss")
    plt.legend()
    plt.grid()
    fig.canvas.draw()
```

In [3]:

```
# 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)])

    df_cm= pd.crosstab(Y_true, Y_pred, rownames=['True'], colnames=['Pred'])
    plt.figure(figsize = (10,7))
    return sn.heatmap(df_cm, annot=True)
```

Data

In [4]:

```
# Data directory
DATADIR = 'E:\\Machine Learning\\Assignments\\Activity Recognition\\HAR\\UCI_HAR_Dataset'
```

In [5]:

```
# 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",
```

In [6]:

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

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

    for signal in SIGNALS:
        filename = f'E:\\Machine Learning\\Assignments\\Activity
Recognition\\HAR\\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]:

```
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'E:\\Machine Learning\\Assignments\\Activity Recognition\\HAR\\UCI_HAR_Dataset/{su
bset}/y_{subset}.txt'
    y = _read_csv(filename)[0]

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

In [8]:

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

    return X_train, X_test, y_train, y_test
```

In [9]:

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

In [10]:

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

In [11]:

```
# Import Keras
from keras import backend as K
```

```
Using TensorFlow backend.
```

In [12]:

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

In [13]:

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

In [14]:

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

In [15]:

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

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

In [16]:

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

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

```
128
9
7352
```

- Defining the Architecture of LSTM

In [17]:

```
# 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\Utkarsh Sri\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.
```

```
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]:

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

In [19]:

```
# 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\Utkarsh Sri\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 [==============================] - 42s 6ms/step - loss: 1.3312 - acc: 0.4323 - val_loss:
1.1683 - val_acc: 0.4846
Epoch 2/30
7352/7352 [==============================] - 40s 5ms/step - loss: 1.0265 - acc: 0.5618 - val_loss:
0.9187 - val_acc: 0.5945
Epoch 3/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.8139 - acc: 0.6488 - val_loss:
0.7853 - val_acc: 0.6193
Epoch 4/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.7096 - acc: 0.6689 - val_loss:
0.7408 - val_acc: 0.6159
Epoch 5/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.6401 - acc: 0.6880 - val_loss:
0.7020 - val_acc: 0.6664
Epoch 6/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.6230 - acc: 0.6979 - val_loss:
0.7287 - val_acc: 0.7017
Epoch 7/30
7352/7352 [==============================] - 41s 6ms/step - loss: 0.5834 - acc: 0.7262 - val_loss:
0.6355 - val_acc: 0.7268
Epoch 8/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.5487 - acc: 0.7481 - val_loss:
0.6564 - val_acc: 0.7306
Epoch 9/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.4930 - acc: 0.7803 - val_loss:
0.6182 - val_acc: 0.7360
Epoch 10/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.4556 - acc: 0.7900 - val_loss:
0.5948 - val_acc: 0.7112
Epoch 11/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.4074 - acc: 0.8039 - val_loss:
```

```
7352/7352 [==============================] - 40s 5ms/step - loss: 0.3793 - acc: 0.8252 - val_loss:
0.4603 - val_acc: 0.7808
Epoch 13/30
7352/7352 [==============================] - 39s 5ms/step - loss: 0.3665 - acc: 0.8630 - val_loss:
0.5039 - val_acc: 0.8612
Epoch 14/30
7352/7352 [==============================] - 38s 5ms/step - loss: 0.3789 - acc: 0.8876 - val_loss:
0.4645 - val_acc: 0.8480
Epoch 15/30
7352/7352 [==============================] - 39s 5ms/step - loss: 0.3372 - acc: 0.9055 - val_loss:
0.4003 - val_acc: 0.8687
Epoch 16/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.2673 - acc: 0.9210 - val_loss:
0.4191 - val_acc: 0.8473
Epoch 17/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.2368 - acc: 0.9236 - val_loss:
0.3641 - val_acc: 0.8867
Epoch 18/30
7352/7352 [==============================] - 39s 5ms/step - loss: 0.2131 - acc: 0.9323 - val_loss:
0.4253 - val_acc: 0.8833
Epoch 19/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.2135 - acc: 0.9347 - val_loss:
0.3123 - val_acc: 0.8985
Epoch 20/30
7352/7352 [==============================] - 39s 5ms/step - loss: 0.1983 - acc: 0.9407 - val_loss:
0.4502 - val_acc: 0.8914
Epoch 21/30
7352/7352 [==============================] - 38s 5ms/step - loss: 0.2383 - acc: 0.9308 - val_loss:
0.3462 - val_acc: 0.8931
Epoch 22/30
7352/7352 [==============================] - 39s 5ms/step - loss: 0.1764 - acc: 0.9441 - val_loss:
0.4002 - val_acc: 0.8958
Epoch 23/30
7352/7352 [==============================] - 39s 5ms/step - loss: 0.1920 - acc: 0.9406 - val_loss:
0.5881 - val_acc: 0.8850
Epoch 24/30
7352/7352 [==============================] - 39s 5ms/step - loss: 0.2227 - acc: 0.9361 - val_loss:
0.3583 - val_acc: 0.8992
Epoch 25/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.1895 - acc: 0.9421 - val_loss:
0.4919 - val_acc: 0.8748
Epoch 26/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.1905 - acc: 0.9438 - val_loss:
0.3812 - val_acc: 0.8826
Epoch 27/30
7352/7352 [==============================] - 39s 5ms/step - loss: 0.1918 - acc: 0.9416 - val_loss:
0.4873 - val_acc: 0.8951
Epoch 28/30
7352/7352 [==============================] - 39s 5ms/step - loss: 0.1648 - acc: 0.9463 - val_loss:
0.3695 - val_acc: 0.8992
Epoch 29/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.1799 - acc: 0.9442 - val_loss:
0.4771 - val_acc: 0.8884
Epoch 30/30
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1605 - acc: 0.9465 - val_loss:
0.4689 - val_acc: 0.8965
```

Out[19]:

```
<keras.callbacks.History at 0x25ae53cc828>
```

In [20]:

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

Pred	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	\
True						
LAYING	510	0	1	0	0	
SITTING	0	419	51	0	1	
STANDING	0	132	397	2	0	
WALKING	0	0	0	466	29	
WALKING_DOWNSTAIRS	0	0	0	1	416	

```
Pred                WALKING_UPSTAIRS
True
LAYING                           26
SITTING                          20
STANDING                          1
WALKING                           1
WALKING_DOWNSTAIRS                3
WALKING_UPSTAIRS                434
```

In [21]:

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

```
2947/2947 [==============================] - 2s 570us/step
```

In [22]:

```
score
```

Out[22]:

```
[0.46892408261934193, 0.8965049202578894]
```

Observations

- With a simple 2 layer architecture we got 89.65% accuracy and a loss of 0.46
- We can further imporve the performace with Hyperparameter tuning

In [ ]:

```
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, verbose=1, va
lidation_data=(X_test, Y_test))

# 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 of epochs

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

Assignment

64 Node LSTM with 50% Dropout

In [17]:

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

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

```
WARNING:tensorflow:From C:\Users\Utkarsh Sri\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\Utkarsh Sri\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, 64)                18944
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 390
=================================================================
Total params: 19,334
Trainable params: 19,334
Non-trainable params: 0
_________________________________________________________________
```

In [18]:

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

In [19]:

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

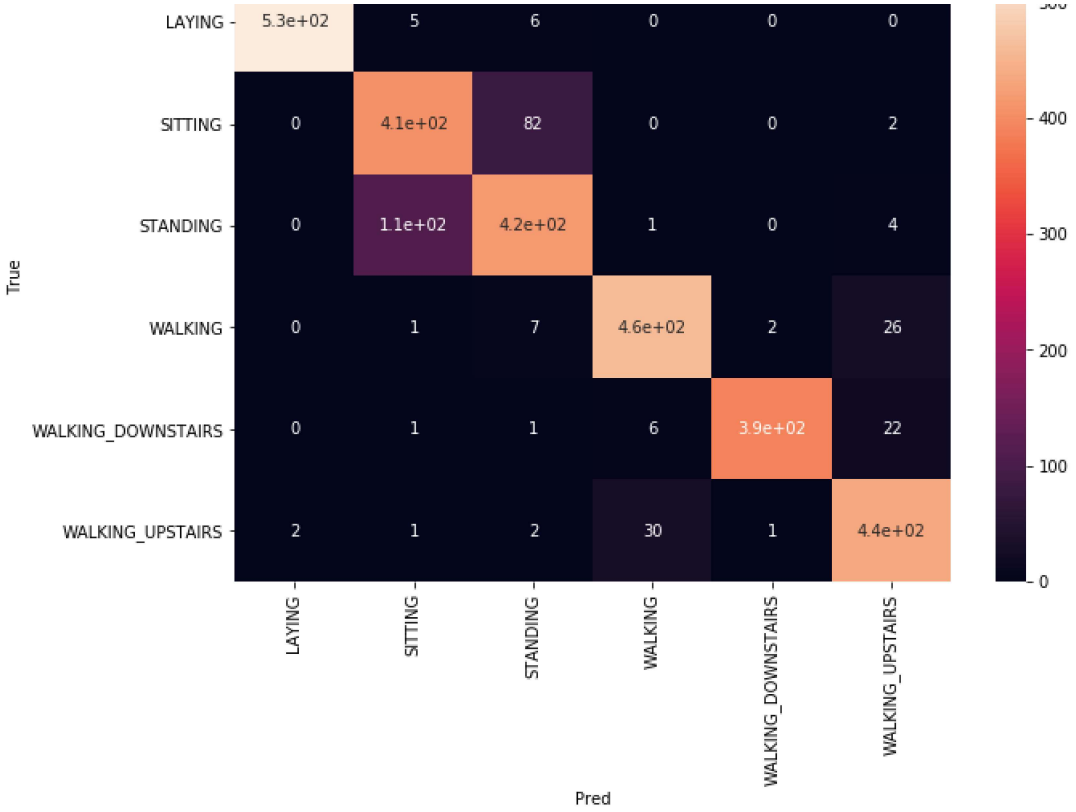
```
WARNING:tensorflow:From C:\Users\Utkarsh Sri\Anaconda3\lib\site-
packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 153s 21ms/step - loss: 1.2745 - acc: 0.4493 - val_los
s: 1.1803 - val_acc: 0.4798
Epoch 2/30
7352/7352 [==============================] - 146s 20ms/step - loss: 1.0554 - acc: 0.5328 - val_los
s: 0.9924 - val_acc: 0.5914
Epoch 3/30
7352/7352 [==============================] - 150s 20ms/step - loss: 0.8490 - acc: 0.6304 - val_los
s: 0.8917 - val_acc: 0.6315
Epoch 4/30
7352/7352 [==============================] - 154s 21ms/step - loss: 0.7967 - acc: 0.6753 - val_los
s: 0.8117 - val_acc: 0.7129
Epoch 5/30
7352/7352 [==============================] - 152s 21ms/step - loss: 0.6605 - acc: 0.7568 - val_los
s: 0.7433 - val_acc: 0.7384
Epoch 6/30
7352/7352 [==============================] - 167s 23ms/step - loss: 0.4971 - acc: 0.8354 - val_los
s: 0.6230 - val_acc: 0.8314
Epoch 7/30
7352/7352 [==============================] - 156s 21ms/step - loss: 0.4033 - acc: 0.8739 - val_los
s: 0.4898 - val_acc: 0.8683
Epoch 8/30
```

```
Epoch 9/30
7352/7352 [==============================] - 147s 20ms/step - loss: 0.2629 - acc: 0.9163 - val_los
s: 0.7544 - val_acc: 0.8188
Epoch 10/30
7352/7352 [==============================] - 158s 22ms/step - loss: 0.2210 - acc: 0.9272 - val_los
s: 0.3708 - val_acc: 0.8850
Epoch 11/30
7352/7352 [==============================] - 150s 20ms/step - loss: 0.2183 - acc: 0.9309 - val_los
s: 0.4459 - val_acc: 0.8806
Epoch 12/30
7352/7352 [==============================] - 143s 19ms/step - loss: 0.1917 - acc: 0.9351 - val_los
s: 0.4205 - val_acc: 0.8850
Epoch 13/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.1764 - acc: 0.9418 - val_los
s: 0.4568 - val_acc: 0.8975
Epoch 14/30
7352/7352 [==============================] - 144s 20ms/step - loss: 0.1689 - acc: 0.9393 - val_los
s: 0.4119 - val_acc: 0.9030
Epoch 15/30
7352/7352 [==============================] - 151s 21ms/step - loss: 0.1696 - acc: 0.9392 - val_los
s: 0.5453 - val_acc: 0.8839
Epoch 16/30
7352/7352 [==============================] - 148s 20ms/step - loss: 0.1761 - acc: 0.9373 - val_los
s: 0.4915 - val_acc: 0.8982
Epoch 17/30
7352/7352 [==============================] - 149s 20ms/step - loss: 0.1700 - acc: 0.9359 - val_los
s: 0.4387 - val_acc: 0.8938
Epoch 18/30
7352/7352 [==============================] - 138s 19ms/step - loss: 0.1506 - acc: 0.9389 - val_los
s: 0.3992 - val_acc: 0.8982
Epoch 19/30
7352/7352 [==============================] - 145s 20ms/step - loss: 0.1528 - acc: 0.9475 - val_los
s: 0.5344 - val_acc: 0.8951
Epoch 20/30
7352/7352 [==============================] - 139s 19ms/step - loss: 0.1462 - acc: 0.9471 - val_los
s: 0.3748 - val_acc: 0.9101
Epoch 21/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.1407 - acc: 0.9479 - val_los
s: 0.4278 - val_acc: 0.8996
Epoch 22/30
7352/7352 [==============================] - 139s 19ms/step - loss: 0.1343 - acc: 0.9495 - val_los
s: 0.4068 - val_acc: 0.9046
Epoch 23/30
7352/7352 [==============================] - 139s 19ms/step - loss: 0.1546 - acc: 0.9453 - val_los
s: 0.4926 - val_acc: 0.8948
Epoch 24/30
7352/7352 [==============================] - 138s 19ms/step - loss: 0.1406 - acc: 0.9471 - val_los
s: 0.5629 - val_acc: 0.8921
Epoch 25/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.1496 - acc: 0.9474 - val_los
s: 0.4126 - val_acc: 0.9118
Epoch 26/30
7352/7352 [==============================] - 139s 19ms/step - loss: 0.1578 - acc: 0.9431 - val_los
s: 0.3190 - val_acc: 0.9165
Epoch 27/30
7352/7352 [==============================] - 139s 19ms/step - loss: 0.1472 - acc: 0.9459 - val_los
s: 0.3117 - val_acc: 0.9169
Epoch 28/30
7352/7352 [==============================] - 138s 19ms/step - loss: 0.1512 - acc: 0.9456 - val_los
s: 0.2773 - val_acc: 0.9175
Epoch 29/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.1637 - acc: 0.9406 - val_los
s: 0.4662 - val_acc: 0.8999
Epoch 30/30
7352/7352 [==============================] - 139s 19ms/step - loss: 0.1381 - acc: 0.9516 - val_los
s: 0.3640 - val_acc: 0.8941
```

In [20]:

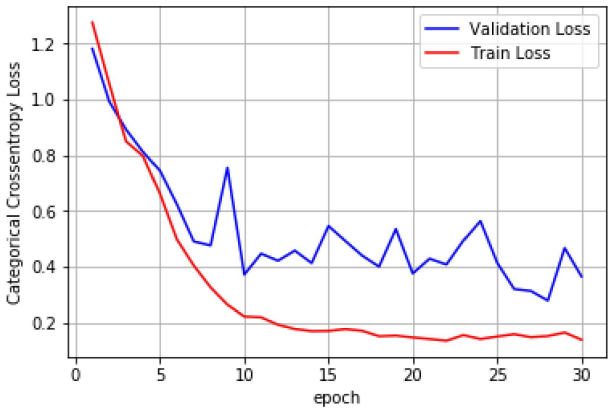
```
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
AxesSubplot(0.125,0.125;0.62x0.755)
```



In [21]:

```
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, verbose=1, va
lidation_data=(X_test, Y_test))

# 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 of epochs

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



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

```
2947/2947 [==============================] - 10s 3ms/step
```

In [23]:

```
score
```

Out[23]:

```
[0.3639915957540732, 0.8941296233457754]
```

Observations

- With a simple 2 layer architecture we got 90.30% accuracy and a loss of 0.34

32 Node LSTM with 25% Dropout

In [17]:

```
model = Sequential()
# Configuring the parameters
model.add(LSTM(32, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.25))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
WARNING:tensorflow:From C:\Users\Utkarsh Sri\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\Utkarsh Sri\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]:

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

In [19]:

```
history=model.fit(X_train,
          Y_train,
          batch_size=batch_size,
```

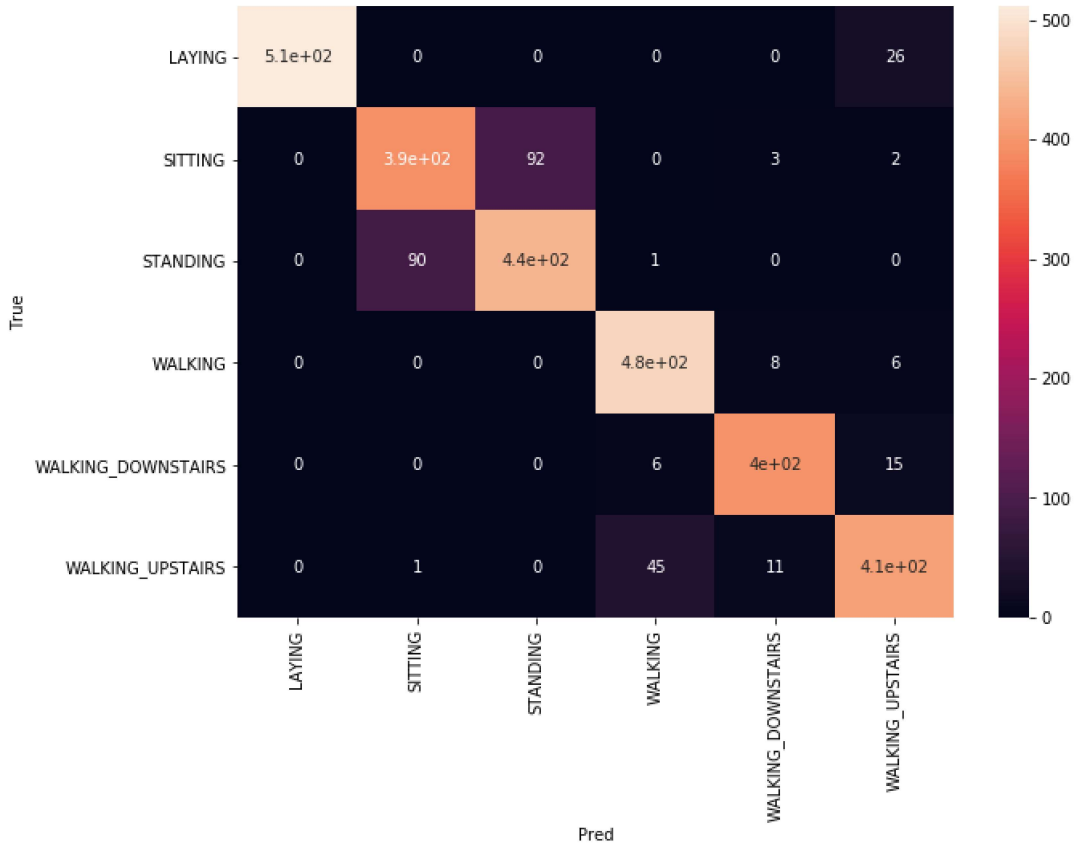
```
WARNING:tensorflow:From C:\Users\Utkarsh Sri\Anaconda3\lib\site-
packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 190s 26ms/step - loss: 1.2903 - acc: 0.4619 - val_los
s: 1.2222 - val_acc: 0.4367
Epoch 2/30
7352/7352 [==============================] - 174s 24ms/step - loss: 0.9752 - acc: 0.5691 - val_los
s: 0.9171 - val_acc: 0.5942
Epoch 3/30
7352/7352 [==============================] - 168s 23ms/step - loss: 0.7803 - acc: 0.6790 - val_los
s: 0.7922 - val_acc: 0.6627
Epoch 4/30
7352/7352 [==============================] - 186s 25ms/step - loss: 0.6672 - acc: 0.6899 - val_los
s: 0.7281 - val_acc: 0.6895
Epoch 5/30
7352/7352 [==============================] - 175s 24ms/step - loss: 0.5766 - acc: 0.7595 - val_los
s: 0.5976 - val_acc: 0.7448
Epoch 6/30
7352/7352 [==============================] - 165s 22ms/step - loss: 0.4731 - acc: 0.7952 - val_los
s: 0.7735 - val_acc: 0.7078
Epoch 7/30
7352/7352 [==============================] - 157s 21ms/step - loss: 0.4095 - acc: 0.8411 - val_los
s: 0.6538 - val_acc: 0.8052
Epoch 8/30
7352/7352 [==============================] - 148s 20ms/step - loss: 0.3041 - acc: 0.9052 - val_los
s: 0.4605 - val_acc: 0.8677
Epoch 9/30
7352/7352 [==============================] - 143s 19ms/step - loss: 0.2436 - acc: 0.9218 - val_los
s: 0.4323 - val_acc: 0.8765
Epoch 10/30
7352/7352 [==============================] - 144s 20ms/step - loss: 0.2256 - acc: 0.9300 - val_los
s: 0.3891 - val_acc: 0.8823
Epoch 11/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.2106 - acc: 0.9309 - val_los
s: 0.4177 - val_acc: 0.8877
Epoch 12/30
7352/7352 [==============================] - 146s 20ms/step - loss: 0.1929 - acc: 0.9374 - val_los
s: 0.4130 - val_acc: 0.8612
Epoch 13/30
7352/7352 [==============================] - 138s 19ms/step - loss: 0.1711 - acc: 0.9423 - val_los
s: 0.2882 - val_acc: 0.8965
Epoch 14/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.1666 - acc: 0.9438 - val_los
s: 0.4002 - val_acc: 0.8880
Epoch 15/30
7352/7352 [==============================] - 142s 19ms/step - loss: 0.1617 - acc: 0.9440 - val_los
s: 0.4144 - val_acc: 0.8863
Epoch 16/30
7352/7352 [==============================] - 152s 21ms/step - loss: 0.1626 - acc: 0.9436 - val_los
s: 0.3580 - val_acc: 0.9002
Epoch 17/30
7352/7352 [==============================] - 149s 20ms/step - loss: 0.1446 - acc: 0.9470 - val_los
s: 0.3485 - val_acc: 0.8941
Epoch 18/30
7352/7352 [==============================] - 137s 19ms/step - loss: 0.1628 - acc: 0.9468 - val_los
s: 0.3410 - val_acc: 0.8992
Epoch 19/30
7352/7352 [==============================] - 145s 20ms/step - loss: 0.1648 - acc: 0.9456 - val_los
s: 0.5908 - val_acc: 0.8758
Epoch 20/30
7352/7352 [==============================] - 145s 20ms/step - loss: 0.1511 - acc: 0.9504 - val_los
s: 0.3802 - val_acc: 0.8907
Epoch 21/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.1612 - acc: 0.9482 - val_los
s: 0.3715 - val_acc: 0.8999
Epoch 22/30
7352/7352 [==============================] - 161s 22ms/step - loss: 0.1519 - acc: 0.9475 - val_los
s: 0.7402 - val_acc: 0.8504
Epoch 23/30
```

```
Epoch 24/30
7352/7352 [==============================] - 155s 21ms/step - loss: 0.1412 - acc: 0.9499 - val_los
s: 0.8429 - val_acc: 0.8466
Epoch 25/30
7352/7352 [==============================] - 152s 21ms/step - loss: 0.1360 - acc: 0.9528 - val_los
s: 0.3804 - val_acc: 0.8938
Epoch 26/30
7352/7352 [==============================] - 151s 20ms/step - loss: 0.1425 - acc: 0.9508 - val_los
s: 0.5873 - val_acc: 0.8626
Epoch 27/30
7352/7352 [==============================] - 151s 21ms/step - loss: 0.1349 - acc: 0.9508 - val_los
s: 0.4106 - val_acc: 0.9125
Epoch 28/30
7352/7352 [==============================] - 152s 21ms/step - loss: 0.1287 - acc: 0.9524 - val_los
s: 0.3821 - val_acc: 0.9050
Epoch 29/30
7352/7352 [==============================] - 147s 20ms/step - loss: 0.1410 - acc: 0.9506 - val_los
s: 0.3348 - val_acc: 0.9192
Epoch 30/30
7352/7352 [==============================] - 184s 25ms/step - loss: 0.1353 - acc: 0.9523 - val_los
s: 0.3653 - val_acc: 0.8962
```

In [20]:

```
print(confusion_matrix(Y_test, model.predict(X_test)))
```

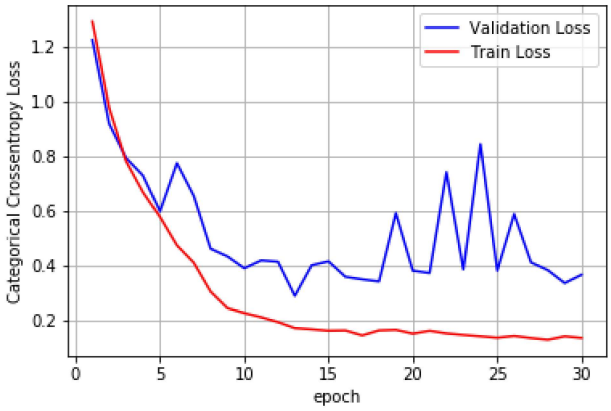
```
AxesSubplot(0.125,0.125;0.62x0.755)
```



In [21]:

```
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, verbose=1, va
```

```
# we will get val_loss and val_acc only when you pass the parameter 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 of epochs

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



In [22]:

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

```
2947/2947 [==============================] - 11s 4ms/step
```

In [23]:

```
score
```

Out[23]:

```
[0.36530350676606693, 0.8961655921275874]
```

Observations

- With a simple 2 layer architecture we got 89.48% accuracy and a loss of 0.42

32 Node LSTM with 75% Dropout + 16 Node LSTM with 50% Dropout

In [17]:

```
model = Sequential()
# Configuring the parameters
model.add(LSTM(32, input_shape=(timesteps, input_dim),return_sequences = True))
# Adding a dropout layer
model.add(Dropout(0.75))
model.add(LSTM(16, 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\Utkarsh Sri\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:
```

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, 128, 32)           5376
_________________________________________________________________
dropout_1 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
lstm_2 (LSTM)                (None, 16)                3136
_________________________________________________________________
dropout_2 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 102
=================================================================
Total params: 8,614
Trainable params: 8,614
Non-trainable params: 0
_________________________________________________________________
```

In [18]:

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

In [19]:

```
history=model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=15)
```

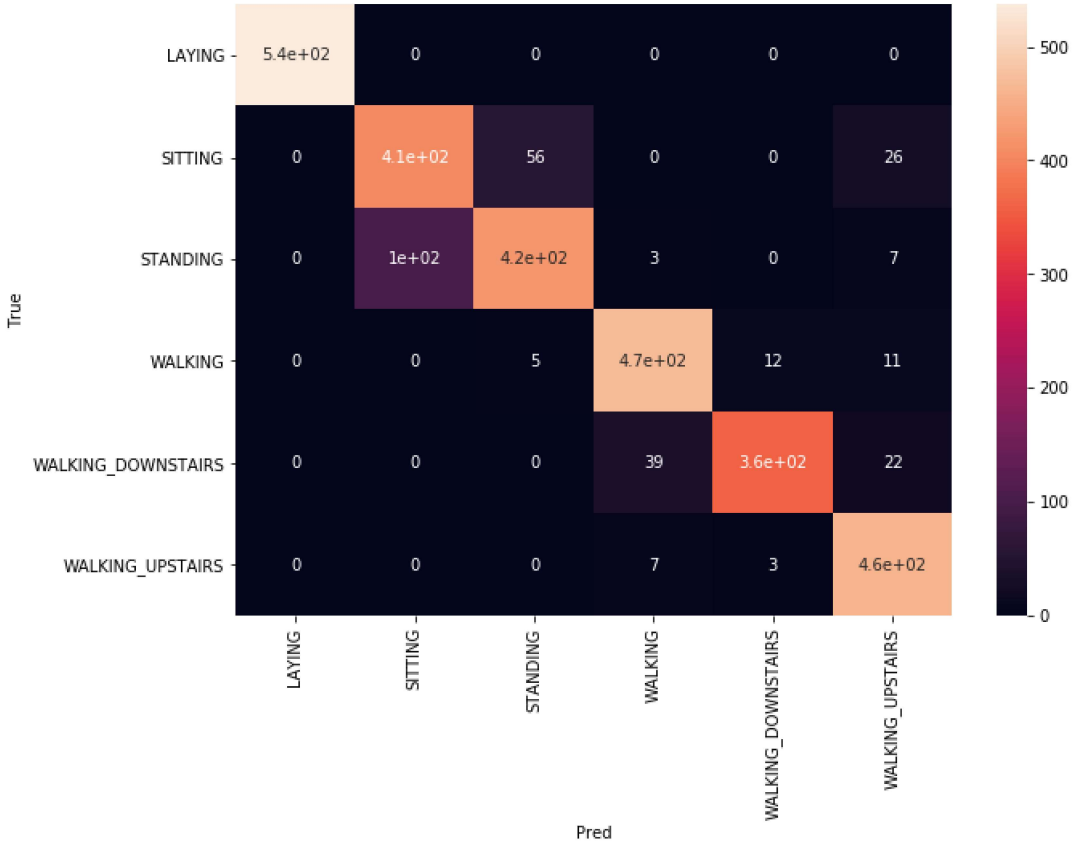
WARNING:tensorflow:From C:\Users\Utkarsh Sri\Anaconda3\lib\site-packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 7352 samples, validate on 2947 samples
Epoch 1/15
7352/7352 [==============================] - 276s 38ms/step - loss: 1.4008 - acc: 0.4754 - val_loss: 1.1915 - val_acc: 0.5429
Epoch 2/15
7352/7352 [==============================] - 254s 35ms/step - loss: 1.0557 - acc: 0.5930 - val_loss: 0.9078 - val_acc: 0.6213
Epoch 3/15
7352/7352 [==============================] - 253s 34ms/step - loss: 0.8967 - acc: 0.6430 - val_loss: 0.7278 - val_acc: 0.6990
Epoch 4/15
7352/7352 [==============================] - 255s 35ms/step - loss: 0.7760 - acc: 0.7065 - val_loss: 0.6566 - val_acc: 0.7211
Epoch 5/15
7352/7352 [==============================] - 253s 34ms/step - loss: 0.6708 - acc: 0.7452 - val_loss: 0.5729 - val_acc: 0.7394
Epoch 6/15
7352/7352 [==============================] - 253s 34ms/step - loss: 0.6052 - acc: 0.7677 - val_loss: 0.5868 - val_acc: 0.7377
Epoch 7/15
7352/7352 [==============================] - 255s 35ms/step - loss: 0.5586 - acc: 0.7775 - val_loss: 0.5246 - val_acc: 0.7547
Epoch 8/15
7352/7352 [==============================] - 253s 34ms/step - loss: 0.5191 - acc: 0.8024 - val_loss: 0.4154 - val_acc: 0.7757
Epoch 9/15
7352/7352 [==============================] - 254s 34ms/step - loss: 0.4833 - acc: 0.8360 - val_loss: 0.5571 - val_acc: 0.8351
Epoch 10/15
7352/7352 [==============================] - 253s 34ms/step - loss: 0.4417 - acc: 0.8603 - val_loss:

```
7352/7352 [==============================] - 253s 34ms/step - loss: 0.4329 - acc: 0.8694 - val_los
s: 0.3827 - val_acc: 0.8850
Epoch 12/15
7352/7352 [==============================] - 255s 35ms/step - loss: 0.3763 - acc: 0.8798 - val_los
s: 0.3998 - val_acc: 0.8951
Epoch 13/15
7352/7352 [==============================] - 253s 34ms/step - loss: 0.3463 - acc: 0.9008 - val_los
s: 0.3443 - val_acc: 0.9067
Epoch 14/15
7352/7352 [==============================] - 254s 35ms/step - loss: 0.3156 - acc: 0.9102 - val_los
s: 0.3472 - val_acc: 0.9009
Epoch 15/15
7352/7352 [==============================] - 253s 34ms/step - loss: 0.2981 - acc: 0.9153 - val_los
s: 0.3591 - val_acc: 0.9013
```

In [20]:

```
print(confusion_matrix(Y_test, model.predict(X_test)))
```

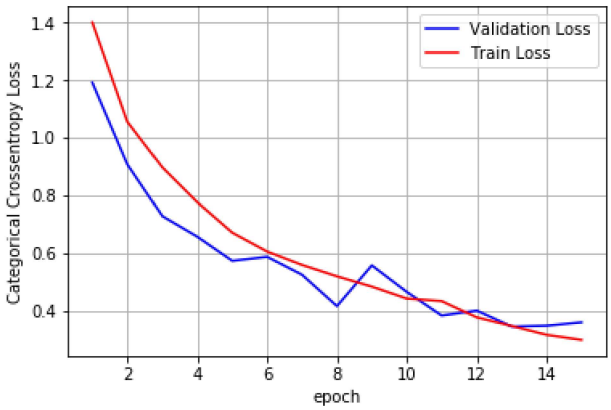
```
AxesSubplot(0.125,0.125;0.62x0.755)
```



In [22]:

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

# list of epoch numbers
x = list(range(1,15+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, verbose=1, va
lidation_data=(X_test, Y_test))

# 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
```

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



In [23]:

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

```
2947/2947 [==============================] - 19s 7ms/step
```

In [24]:

```
score
```

Out[24]:

```
[0.3591378914267533, 0.9012555140821175]
```

Observations

- With a simple 2 layer architecture we got 90.70% accuracy and a loss of 0.47

64 Node LSTM with 25% Dropout

In [17]:

```
model = Sequential()
# Configuring the parameters
model.add(LSTM(64, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.25))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
WARNING:tensorflow:From C:\Users\Utkarsh Sri\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\Utkarsh Sri\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 #
```

```
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 390
=================================================================
Total params: 19,334
Trainable params: 19,334
Non-trainable params: 0
_________________________________________________________________
```

In [18]:

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

In [19]:

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

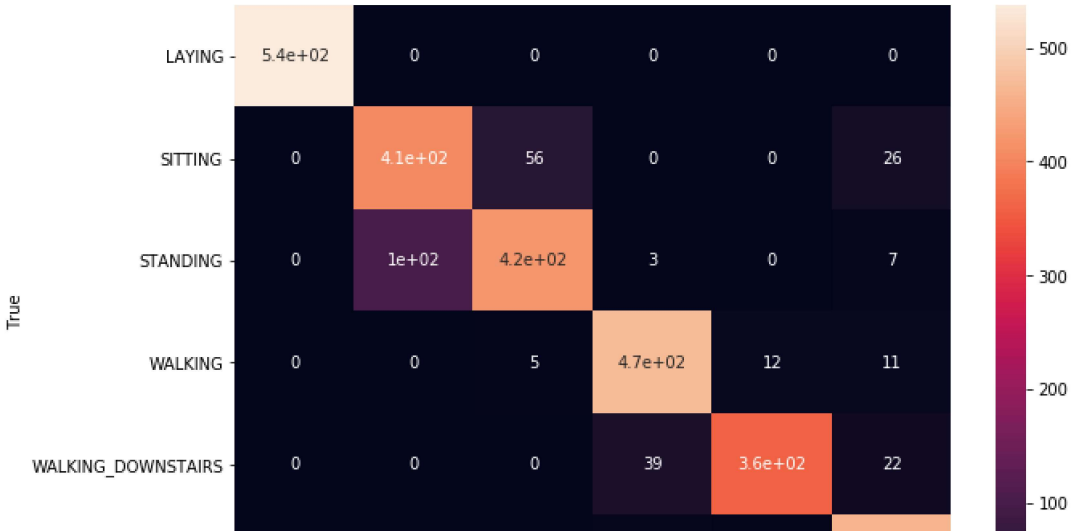
```
WARNING:tensorflow:From C:\Users\Utkarsh Sri\Anaconda3\lib\site-
packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 162s 22ms/step - loss: 1.2910 - acc: 0.4147 - val_los
s: 1.2095 - val_acc: 0.4605
Epoch 2/30
7352/7352 [==============================] - 127s 17ms/step - loss: 1.0924 - acc: 0.5091 - val_los
s: 1.1413 - val_acc: 0.4696
Epoch 3/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.8651 - acc: 0.6347 - val_los
s: 0.8807 - val_acc: 0.6172
Epoch 4/30
7352/7352 [==============================] - 145s 20ms/step - loss: 0.6883 - acc: 0.7127 - val_los
s: 0.7194 - val_acc: 0.7011
Epoch 5/30
7352/7352 [==============================] - 133s 18ms/step - loss: 0.5777 - acc: 0.7828 - val_los
s: 0.6578 - val_acc: 0.7906
Epoch 6/30
7352/7352 [==============================] - 133s 18ms/step - loss: 0.5146 - acc: 0.8221 - val_los
s: 0.6614 - val_acc: 0.7737
Epoch 7/30
7352/7352 [==============================] - 149s 20ms/step - loss: 0.4403 - acc: 0.8464 - val_los
s: 0.5433 - val_acc: 0.8120
Epoch 8/30
7352/7352 [==============================] - 147s 20ms/step - loss: 0.3350 - acc: 0.8867 - val_los
s: 0.5037 - val_acc: 0.8381
Epoch 9/30
7352/7352 [==============================] - 146s 20ms/step - loss: 0.3091 - acc: 0.8954 - val_los
s: 0.4071 - val_acc: 0.8504
Epoch 10/30
7352/7352 [==============================] - 146s 20ms/step - loss: 0.3433 - acc: 0.8874 - val_los
s: 0.4285 - val_acc: 0.8449
Epoch 11/30
7352/7352 [==============================] - 146s 20ms/step - loss: 0.2503 - acc: 0.9147 - val_los
s: 0.4332 - val_acc: 0.8592
Epoch 12/30
7352/7352 [==============================] - 148s 20ms/step - loss: 0.2255 - acc: 0.9261 - val_los
s: 0.4328 - val_acc: 0.8714
Epoch 13/30
7352/7352 [==============================] - 147s 20ms/step - loss: 0.2150 - acc: 0.9260 - val_los
s: 0.3099 - val_acc: 0.8911
Epoch 14/30
7352/7352 [==============================] - 147s 20ms/step - loss: 0.1905 - acc: 0.9327 - val_los
s: 0.2847 - val_acc: 0.9030
```

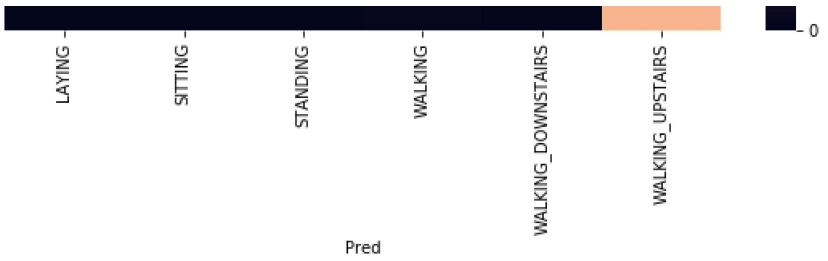
```
s: 0.3964 - val_acc: 0.8921
Epoch 16/30
7352/7352 [==============================] - 147s 20ms/step - loss: 0.2121 - acc: 0.9310 - val_los
s: 0.2323 - val_acc: 0.9206
Epoch 17/30
7352/7352 [==============================] - 148s 20ms/step - loss: 0.1607 - acc: 0.9423 - val_los
s: 0.3125 - val_acc: 0.9013
Epoch 18/30
7352/7352 [==============================] - 148s 20ms/step - loss: 0.1594 - acc: 0.9460 - val_los
s: 0.3082 - val_acc: 0.9091
Epoch 19/30
7352/7352 [==============================] - 148s 20ms/step - loss: 0.1542 - acc: 0.9460 - val_los
s: 0.4292 - val_acc: 0.9063
Epoch 20/30
7352/7352 [==============================] - 148s 20ms/step - loss: 0.1422 - acc: 0.9491 - val_los
s: 0.3046 - val_acc: 0.9141
Epoch 21/30
7352/7352 [==============================] - 149s 20ms/step - loss: 0.1449 - acc: 0.9487 - val_los
s: 0.3540 - val_acc: 0.9080
Epoch 22/30
7352/7352 [==============================] - 149s 20ms/step - loss: 0.1303 - acc: 0.9457 - val_los
s: 0.2957 - val_acc: 0.9023
Epoch 23/30
7352/7352 [==============================] - 148s 20ms/step - loss: 0.1276 - acc: 0.9489 - val_los
s: 0.3380 - val_acc: 0.9104
Epoch 24/30
7352/7352 [==============================] - 149s 20ms/step - loss: 0.1475 - acc: 0.9495 - val_los
s: 0.2874 - val_acc: 0.9152
Epoch 25/30
7352/7352 [==============================] - 148s 20ms/step - loss: 0.1358 - acc: 0.9495 - val_los
s: 0.2401 - val_acc: 0.9175
Epoch 26/30
7352/7352 [==============================] - 148s 20ms/step - loss: 0.1273 - acc: 0.9529 - val_los
s: 0.2943 - val_acc: 0.9220
Epoch 27/30
7352/7352 [==============================] - 148s 20ms/step - loss: 0.1320 - acc: 0.9516 - val_los
s: 0.3164 - val_acc: 0.9026
Epoch 28/30
7352/7352 [==============================] - 147s 20ms/step - loss: 0.1278 - acc: 0.9474 - val_los
s: 0.3515 - val_acc: 0.9111
Epoch 29/30
7352/7352 [==============================] - 147s 20ms/step - loss: 0.1282 - acc: 0.9514 - val_los
s: 0.2780 - val_acc: 0.9216
Epoch 30/30
7352/7352 [==============================] - 151s 20ms/step - loss: 0.1200 - acc: 0.9536 - val_los
s: 0.2614 - val_acc: 0.9155
```

In [25]:

```
print(confusion_matrix(Y_test, model.predict(X_test)))
```

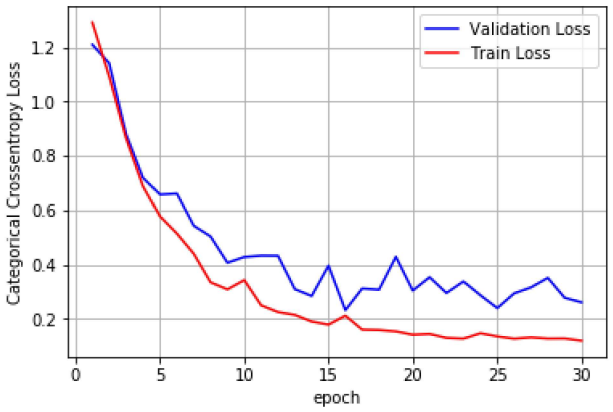
```
AxesSubplot(0.125,0.125;0.62x0.755)
```





In [24]:

```
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, verbose=1, va
lidation_data=(X_test, Y_test))

# 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 of epochs

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



In [25]:

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

```
2947/2947 [==============================] - 22s 8ms/step
```

In [26]:

```
score
```

Out[26]:

```
[1.289391368334278, 0.38819138106549034]
```

Observations

128 Node LSTM with 25% Dropout

In [25]:

```
model = Sequential()
# Configuring the parameters
model.add(LSTM(128, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.25))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 128)               70656
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 774
=================================================================
Total params: 71,430
Trainable params: 71,430
Non-trainable params: 0
_________________________________________________________________
```

In [28]:

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

In [29]:

```
history=model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=30)
```

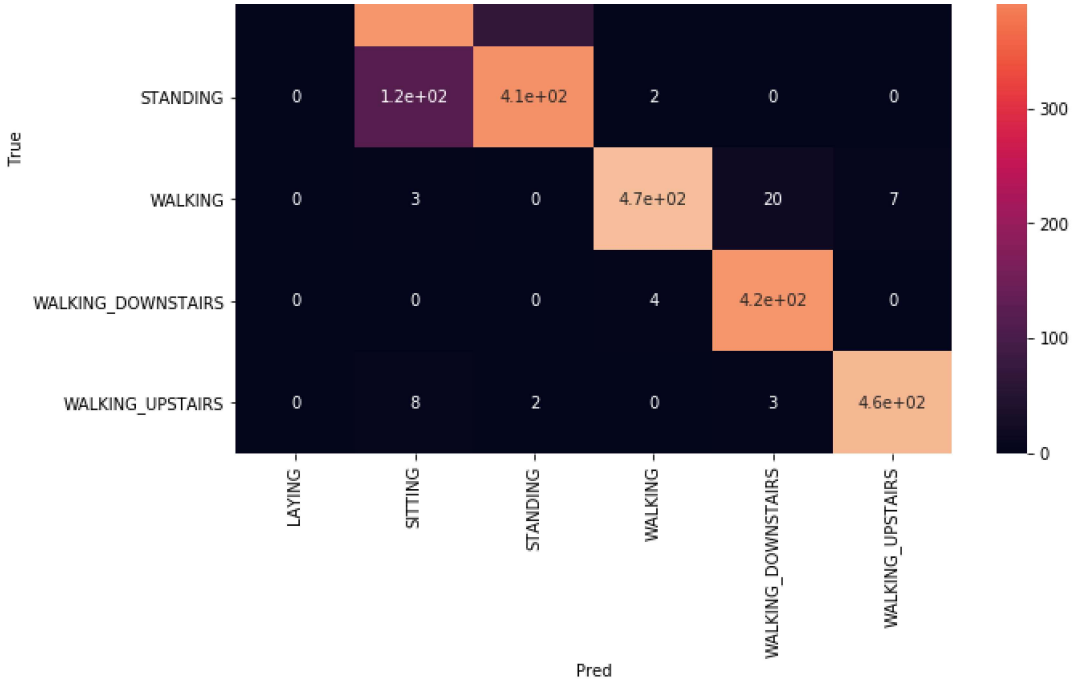
```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 131s 18ms/step - loss: 0.8246 - acc: 0.6144 - val_los
s: 0.8130 - val_acc: 0.5914
Epoch 2/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.5628 - acc: 0.7466 - val_los
s: 0.5878 - val_acc: 0.8022
Epoch 3/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.3061 - acc: 0.8950 - val_los
s: 0.3314 - val_acc: 0.8823
Epoch 4/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.2318 - acc: 0.9233 - val_los
s: 0.5525 - val_acc: 0.8497
Epoch 5/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.1953 - acc: 0.9317 - val_los
s: 0.3166 - val_acc: 0.9040
Epoch 6/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.1846 - acc: 0.9377 - val_los
s: 0.3573 - val_acc: 0.8863
Epoch 7/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.1634 - acc: 0.9414 - val_los
s: 0.3461 - val_acc: 0.8633
Epoch 8/30
7352/7352 [==============================] - 128s 17ms/step - loss: 0.1612 - acc: 0.9416 - val_los
s: 0.3927 - val_acc: 0.9019
Epoch 9/30
7352/7352 [==============================] - 128s 17ms/step - loss: 0.1461 - acc: 0.9464 - val_los
s: 0.4263 - val_acc: 0.8914
```

```
s: 0.3316 - val_acc: 0.8850
Epoch 11/30
7352/7352 [==============================] - 128s 17ms/step - loss: 0.1378 - acc: 0.9453 - val_los
s: 0.2321 - val_acc: 0.9264
Epoch 12/30
7352/7352 [==============================] - 128s 17ms/step - loss: 0.1485 - acc: 0.9400 - val_los
s: 0.5237 - val_acc: 0.8928
Epoch 13/30
7352/7352 [==============================] - 128s 17ms/step - loss: 0.1626 - acc: 0.9408 - val_los
s: 0.3634 - val_acc: 0.9148
Epoch 14/30
7352/7352 [==============================] - 130s 18ms/step - loss: 0.1361 - acc: 0.9479 - val_los
s: 0.4242 - val_acc: 0.9050
Epoch 15/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.1522 - acc: 0.9476 - val_los
s: 0.2847 - val_acc: 0.9230
Epoch 16/30
7352/7352 [==============================] - 130s 18ms/step - loss: 0.1460 - acc: 0.9475 - val_los
s: 0.4842 - val_acc: 0.8948
Epoch 17/30
7352/7352 [==============================] - 127s 17ms/step - loss: 0.1352 - acc: 0.9459 - val_los
s: 0.2915 - val_acc: 0.9223
Epoch 18/30
7352/7352 [==============================] - 127s 17ms/step - loss: 0.1432 - acc: 0.9495 - val_los
s: 0.3070 - val_acc: 0.9097
Epoch 19/30
7352/7352 [==============================] - 127s 17ms/step - loss: 0.1435 - acc: 0.9468 - val_los
s: 0.2541 - val_acc: 0.9226
Epoch 20/30
7352/7352 [==============================] - 129s 17ms/step - loss: 0.1301 - acc: 0.9520 - val_los
s: 0.2877 - val_acc: 0.9182
Epoch 21/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.1401 - acc: 0.9494 - val_los
s: 0.3673 - val_acc: 0.9125
Epoch 22/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.1426 - acc: 0.9482 - val_los
s: 0.2825 - val_acc: 0.9128
Epoch 23/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.1288 - acc: 0.9521 - val_los
s: 0.2567 - val_acc: 0.9287
Epoch 24/30
7352/7352 [==============================] - 133s 18ms/step - loss: 0.1268 - acc: 0.9516 - val_los
s: 0.4492 - val_acc: 0.9135
Epoch 25/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.1290 - acc: 0.9510 - val_los
s: 0.2380 - val_acc: 0.9301
Epoch 26/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.1312 - acc: 0.9535 - val_los
s: 0.2425 - val_acc: 0.9264
Epoch 27/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.1376 - acc: 0.9504 - val_los
s: 0.2465 - val_acc: 0.9148
Epoch 28/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.1360 - acc: 0.9518 - val_los
s: 0.3004 - val_acc: 0.9226
Epoch 29/30
7352/7352 [==============================] - 130s 18ms/step - loss: 0.1319 - acc: 0.9539 - val_los
s: 0.3095 - val_acc: 0.9026
Epoch 30/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.1086 - acc: 0.9561 - val_los
s: 0.3873 - val_acc: 0.9155
```

In [30]:

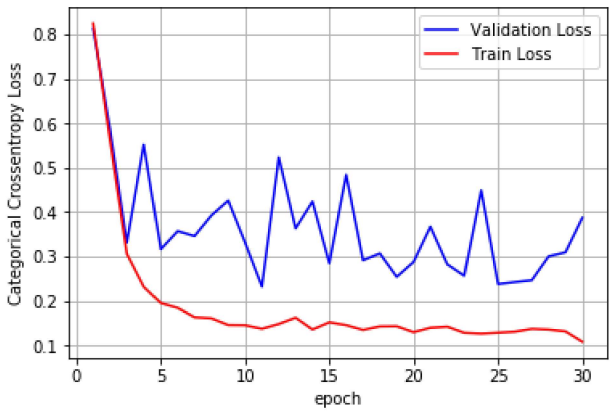
```
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
AxesSubplot(0.125,0.125;0.62x0.755)
```



In [31]:

```
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, verbose=1, va
lidation_data=(X_test, Y_test))

# 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 of epochs

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



In [32]:

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

In [33]:

```
score
```

Out[33]:

```
[0.3873248515814697, 0.9155072955548015]
```

In [ ]:

```

```

Observations

- With a architecture we got 81.61% accuracy and a loss of 0.36

256 Node LSTM with 50% Dropout

In [17]:

```
model = Sequential()
# Configuring the parameters
model.add(LSTM(48, 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\Utkarsh Sri\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\Utkarsh Sri\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, 48)                11136
_________________________________________________________________
dropout_1 (Dropout)          (None, 48)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 294
=================================================================
Total params: 11,430
Trainable params: 11,430
Non-trainable params: 0
_________________________________________________________________
```

In [18]:

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

In [19]:

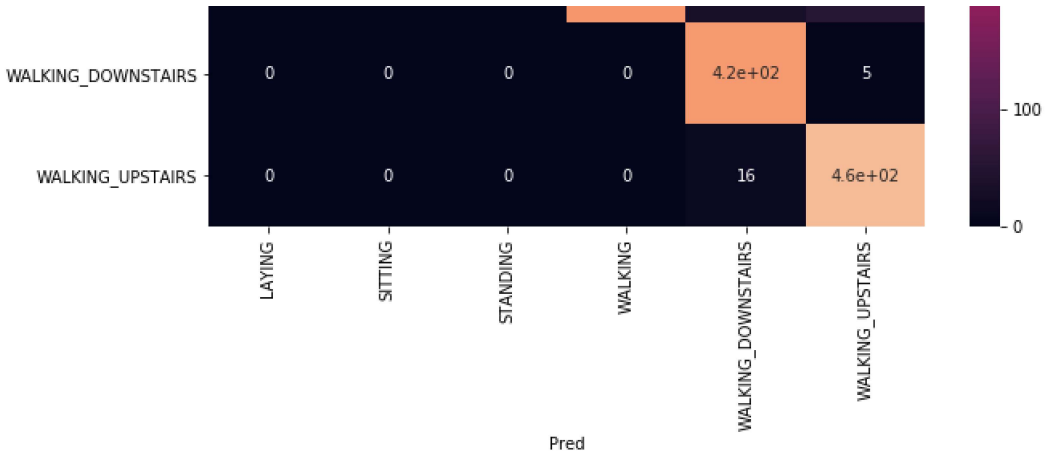
```
history=model.fit(X_train,
          Y_train,
          batch_size=batch_size,
```

```
WARNING:tensorflow:From C:\Users\Utkarsh Sri\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 1/15
7352/7352 [==============================] - 144s 20ms/step - loss: 1.2951 - acc: 0.4487 - val_los
s: 1.2627 - val_acc: 0.4174
Epoch 2/15
7352/7352 [==============================] - 132s 18ms/step - loss: 0.9968 - acc: 0.5509 - val_los
s: 0.8794 - val_acc: 0.6064
Epoch 3/15
7352/7352 [==============================] - 127s 17ms/step - loss: 0.7534 - acc: 0.6804 - val_los
s: 0.7449 - val_acc: 0.7146
Epoch 4/15
7352/7352 [==============================] - 127s 17ms/step - loss: 0.6350 - acc: 0.7372 - val_los
s: 0.6397 - val_acc: 0.7784
Epoch 5/15
7352/7352 [==============================] - 127s 17ms/step - loss: 0.5202 - acc: 0.8089 - val_los
s: 0.6026 - val_acc: 0.7896
Epoch 6/15
7352/7352 [==============================] - 129s 17ms/step - loss: 0.4465 - acc: 0.8592 - val_los
s: 0.4893 - val_acc: 0.8500
Epoch 7/15
7352/7352 [==============================] - 128s 17ms/step - loss: 0.3945 - acc: 0.8799 - val_los
s: 0.5466 - val_acc: 0.8337
Epoch 8/15
7352/7352 [==============================] - 128s 17ms/step - loss: 0.3614 - acc: 0.8898 - val_los
s: 0.6207 - val_acc: 0.8259
Epoch 9/15
7352/7352 [==============================] - 129s 18ms/step - loss: 0.2786 - acc: 0.9143 - val_los
s: 0.3230 - val_acc: 0.8907
Epoch 10/15
7352/7352 [==============================] - 130s 18ms/step - loss: 0.2349 - acc: 0.9259 - val_los
s: 0.3283 - val_acc: 0.8975
Epoch 11/15
7352/7352 [==============================] - 128s 17ms/step - loss: 0.2340 - acc: 0.9246 - val_los
s: 0.3062 - val_acc: 0.8924
Epoch 12/15
7352/7352 [==============================] - 139s 19ms/step - loss: 0.2025 - acc: 0.9323 - val_los
s: 0.4153 - val_acc: 0.8870
Epoch 13/15
7352/7352 [==============================] - 140s 19ms/step - loss: 0.1941 - acc: 0.9342 - val_los
s: 0.3844 - val_acc: 0.8962
Epoch 14/15
7352/7352 [==============================] - 140s 19ms/step - loss: 0.1835 - acc: 0.9385 - val_los
s: 0.3694 - val_acc: 0.8911
Epoch 15/15
7352/7352 [==============================] - 135s 18ms/step - loss: 0.1983 - acc: 0.9368 - val_los
s: 0.3809 - val_acc: 0.8921
```

In [20]:

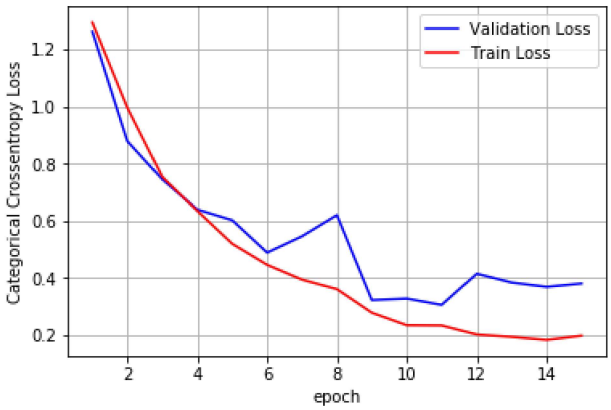
```
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
AxesSubplot(0.125,0.125;0.62x0.755)
```



In [22]:

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

# list of epoch numbers
x = list(range(1,15+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, verbose=1, va
lidation_data=(X_test, Y_test))

# 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 of epochs

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



In [23]:

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

```
2947/2947 [==============================] - 10s 3ms/step
```

In [24]:

```
score
```

Out[24]:

64 Node LSTM with 25% Dropout + 32 Node LSTM with 10% Dropout + 16 Node LSTM with 5% Dropout

In [17]:

```
model = Sequential()
# Configuring the parameters
model.add(LSTM(64, input_shape=(timesteps, input_dim),return_sequences = True))
# Adding a dropout layer
model.add(Dropout(0.25))
model.add(LSTM(32, input_shape=(timesteps, input_dim),return_sequences = True))
# Adding a dropout layer
model.add(Dropout(0.10))
model.add(LSTM(16, input_shape=(timesteps, input_dim)))
model.add(Dropout(0.05))
# Adding a dropout layer
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
WARNING:tensorflow:From C:\Users\Utkarsh Sri\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\Utkarsh Sri\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, 128, 64)           18944
_________________________________________________________________
dropout_1 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
lstm_2 (LSTM)                (None, 128, 32)           12416
_________________________________________________________________
dropout_2 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
lstm_3 (LSTM)                (None, 16)                3136
_________________________________________________________________
dropout_3 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 102
=================================================================
Total params: 34,598
Trainable params: 34,598
Non-trainable params: 0
_________________________________________________________________
```

In [18]:

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

In [19]:

```
history=model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=15)
```

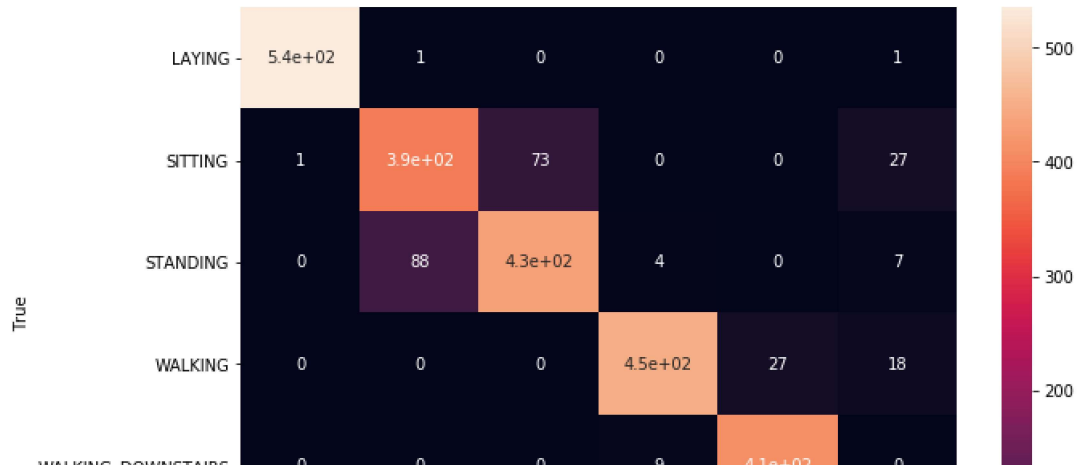
```
WARNING:tensorflow:From C:\Users\Utkarsh Sri\Anaconda3\lib\site-
```

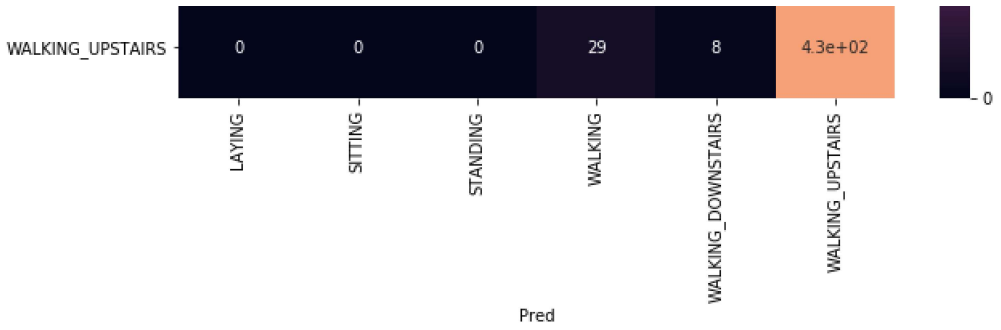
```
Instructions for updating:
Use tf.cast instead.
Train on 7352 samples, validate on 2947 samples
Epoch 1/15
7352/7352 [==============================] - 443s 60ms/step - loss: 1.2461 - acc: 0.4947 - val_los
s: 1.0148 - val_acc: 0.6457
Epoch 2/15
7352/7352 [==============================] - 415s 56ms/step - loss: 0.7504 - acc: 0.7394 - val_los
s: 0.6295 - val_acc: 0.7723
Epoch 3/15
7352/7352 [==============================] - 377s 51ms/step - loss: 0.4425 - acc: 0.8761 - val_los
s: 0.4949 - val_acc: 0.8504
Epoch 4/15
7352/7352 [==============================] - 427s 58ms/step - loss: 0.3018 - acc: 0.9132 - val_los
s: 0.5724 - val_acc: 0.8429
Epoch 5/15
7352/7352 [==============================] - 454s 62ms/step - loss: 0.2493 - acc: 0.9225 - val_los
s: 0.3743 - val_acc: 0.8582
Epoch 6/15
7352/7352 [==============================] - 422s 57ms/step - loss: 0.2093 - acc: 0.9317 - val_los
s: 0.4370 - val_acc: 0.8660
Epoch 7/15
7352/7352 [==============================] - 374s 51ms/step - loss: 0.1752 - acc: 0.9399 - val_los
s: 0.5569 - val_acc: 0.8459
Epoch 8/15
7352/7352 [==============================] - 400s 54ms/step - loss: 0.1572 - acc: 0.9448 - val_los
s: 0.3430 - val_acc: 0.8985
Epoch 9/15
7352/7352 [==============================] - 436s 59ms/step - loss: 0.1436 - acc: 0.9465 - val_los
s: 0.3893 - val_acc: 0.8992
Epoch 10/15
7352/7352 [==============================] - 434s 59ms/step - loss: 0.1469 - acc: 0.9471 - val_los
s: 0.7431 - val_acc: 0.8317
Epoch 11/15
7352/7352 [==============================] - 413s 56ms/step - loss: 0.1430 - acc: 0.9455 - val_los
s: 0.4779 - val_acc: 0.8850
Epoch 12/15
7352/7352 [==============================] - 378s 51ms/step - loss: 0.1416 - acc: 0.9480 - val_los
s: 0.3827 - val_acc: 0.9101
Epoch 13/15
7352/7352 [==============================] - 380s 52ms/step - loss: 0.1296 - acc: 0.9516 - val_los
s: 0.3642 - val_acc: 0.9063
Epoch 14/15
7352/7352 [==============================] - 408s 56ms/step - loss: 0.1291 - acc: 0.9498 - val_los
s: 0.5719 - val_acc: 0.8826
Epoch 15/15
7352/7352 [==============================] - 395s 54ms/step - loss: 0.1293 - acc: 0.9505 - val_los
s: 0.4048 - val_acc: 0.9006
```

In [20]:

```
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
AxesSubplot(0.125,0.125;0.62x0.755)
```





In [21]:

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

```
2947/2947 [==============================] - 26s 9ms/step
```

In [22]:

```
score
```

Out[22]:

```
[0.40479919474501946, 0.9005768578215134]
```

Observations

- With a architecture we got 90.05% accuracy and a loss of 0.40

Conclusion

Observations

The highest accuracy achived is 91.67%.
Models with different Dropouts,Layers & nodes being trained.

Model Comparision

In [32]:

```
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Number of Layers","Number of Nodes", "Dropout", "Validation Loss", "Validation Ac
curacy"]

x.add_row([1, 32,"50%", 0.46, "89.65%"])
x.add_row([1, 64, "50%",0.34, "90.32%"])
x.add_row([1, 32, "25%",0.42, "89.20%"])
x.add_row([1, 64, "25%",0.58, "89.71%"])
x.add_row([1, 128, "25%",0.36, "91.67%"])
x.add_row([1, 256, "50%",0.87, "86.93%"])
x.add_row([2, "32+16", "75+50",0.47, "90.70%"])
x.add_row([3, "64+32+16","25+10+5",0.40,"90.05%"])


print(x)
```

```
+------------------+----------------+---------+----------------+--------------------+
| Number of Layers | Number of Nodes | Dropout | Validation Loss | Validation Accuracy |
```

1	64	50%	0.34	90.32%	
1	32	25%	0.42	89.20%	
1	64	25%	0.58	89.71%	
1	128	25%	0.36	91.67%	
1	256	50%	0.87	86.93%	
2	32+16	75+50	0.47	90.70%	
3	64+32+16	25+10+5	0.4	90.05%	

Steps Taken

Defining Various Models to increase Accuray.
Considering Different values of Nodes,Dropouts.
Plotting Confusion Matrix.
Drawing Conclusions.

In [ ]: