In [1]:

```
# Importing Libraries
```

In [2]:

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

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)])

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

Data

In [4]:

```
# Data directory
DATADIR = '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",
    "total_acc_z"
]
```

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'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'UCI_HAR_Dataset/{subset}/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 [12]:

```
import warnings

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

In [13]:

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

In [14]:

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

In [15]:

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

n_hidden=64 and dropout=0.75

In [16]:

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

# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 64

import warnings
warnings.filterwarnings("ignore")

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

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))

# 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(0.75))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()

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

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

```
128
9
7352
_________________________________________________________________
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
_________________________________________________________________
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 106s 14ms/step - loss: 1.3362 - acc: 0.4219 - val_los
s: 1.1777 - val_acc: 0.4425
Epoch 2/30
7352/7352 [==============================] - 85s 12ms/step - loss: 1.0784 - acc: 0.5286 - val_loss
: 0.9738 - val_acc: 0.5908
Epoch 3/30
7352/7352 [==============================] - 113s 15ms/step - loss: 0.9211 - acc: 0.5933 - val_los
s: 1.0172 - val_acc: 0.5877
Epoch 4/30
7352/7352 [==============================] - 93s 13ms/step - loss: 0.8725 - acc: 0.6109 - val_loss
: 0.8408 - val_acc: 0.5803
```

```
Epoch 5/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.7957 - acc: 0.6296 - val_los
s: 0.7838 - val_acc: 0.6216
Epoch 6/30
7352/7352 [==============================] - 120s 16ms/step - loss: 0.7280 - acc: 0.6586 - val_los
s: 0.7752 - val_acc: 0.6325
Epoch 7/30
7352/7352 [==============================] - 134s 18ms/step - loss: 0.6963 - acc: 0.6946 - val_los
s: 1.0885 - val_acc: 0.5497
Epoch 8/30
7352/7352 [==============================] - 136s 19ms/step - loss: 0.6256 - acc: 0.7433 - val_los
s: 0.6902 - val_acc: 0.7391
Epoch 9/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.5125 - acc: 0.7828 - val_los
s: 0.6471 - val_acc: 0.7933
Epoch 10/30
7352/7352 [==============================] - 147s 20ms/step - loss: 0.5204 - acc: 0.8005 - val_los
s: 0.7481 - val_acc: 0.7927
Epoch 11/30
7352/7352 [==============================] - 146s 20ms/step - loss: 0.5522 - acc: 0.7889 - val_los
s: 0.6910 - val_acc: 0.7703
Epoch 12/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.4643 - acc: 0.8505 - val_los
s: 0.5249 - val_acc: 0.8629
Epoch 13/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.3940 - acc: 0.8806 - val_los
s: 0.5224 - val_acc: 0.8592
Epoch 14/30
7352/7352 [==============================] - 137s 19ms/step - loss: 0.3367 - acc: 0.8991 - val_los
s: 0.4356 - val_acc: 0.8633
Epoch 15/30
7352/7352 [==============================] - 119s 16ms/step - loss: 0.3196 - acc: 0.9030 - val_los
s: 0.4107 - val_acc: 0.8717
Epoch 16/30
7352/7352 [==============================] - 135s 18ms/step - loss: 0.3100 - acc: 0.9108 - val_los
s: 0.3529 - val_acc: 0.8873
Epoch 17/30
7352/7352 [==============================] - 138s 19ms/step - loss: 0.2745 - acc: 0.9248 - val_los
s: 0.5333 - val_acc: 0.8707
Epoch 18/30
7352/7352 [==============================] - 141s 19ms/step - loss: 0.2637 - acc: 0.9264 - val_los
s: 0.4006 - val_acc: 0.8941
Epoch 19/30
7352/7352 [==============================] - 144s 20ms/step - loss: 0.2539 - acc: 0.9293 - val_los
s: 0.4366 - val_acc: 0.8870
Epoch 20/30
7352/7352 [==============================] - 127s 17ms/step - loss: 0.2635 - acc: 0.9260 - val_los
s: 0.6227 - val_acc: 0.8242
Epoch 21/30
7352/7352 [==============================] - 132s 18ms/step - loss: 0.2608 - acc: 0.9302 - val_los
s: 0.3450 - val_acc: 0.9053
Epoch 22/30
7352/7352 [==============================] - 133s 18ms/step - loss: 0.2103 - acc: 0.9362 - val_los
s: 0.4566 - val_acc: 0.9060
Epoch 23/30
7352/7352 [==============================] - 136s 19ms/step - loss: 0.2225 - acc: 0.9342 - val_los
s: 0.4679 - val_acc: 0.8880
Epoch 24/30
7352/7352 [==============================] - 136s 18ms/step - loss: 0.2345 - acc: 0.9342 - val_los
s: 0.4704 - val_acc: 0.9013
Epoch 25/30
7352/7352 [==============================] - 10839s 1s/step - loss: 0.1993 - acc: 0.9382 - val_los
s: 0.3809 - val_acc: 0.9043
Epoch 26/30
7352/7352 [==============================] - 60344s 8s/step - loss: 0.2155 - acc: 0.9359 - val_los
s: 0.4240 - val_acc: 0.8887
Epoch 27/30
7352/7352 [==============================] - 40s 5ms/step - loss: 0.1903 - acc: 0.9393 - val_loss:
0.5555 - val_acc: 0.9040
Epoch 28/30
7352/7352 [==============================] - 56s 8ms/step - loss: 0.1898 - acc: 0.9408 - val_loss:
0.5897 - val_acc: 0.8999
Epoch 29/30
7352/7352 [==============================] - 132s 18ms/step - loss: 0.1798 - acc: 0.9415 - val_los
s: 0.6991 - val_acc: 0.8914
Epoch 30/30
7352/7352 [==============================] - 140s 19ms/step - loss: 0.2200 - acc: 0.9382 - val_los
```

```
s: 0.6202 - val_acc: 0.8904
```

In [17]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 499       11         0        0                   0
SITTING                  1      421        67        1                   0
STANDING                 0      124       402        5                   0
WALKING                  0        1         0      450                  21
WALKING_DOWNSTAIRS       0        0         0        0                 403
WALKING_UPSTAIRS         0        2         0       20                   0

Pred                WALKING_UPSTAIRS
True
LAYING                            27
SITTING                            1
STANDING                           1
WALKING                           24
WALKING_DOWNSTAIRS                17
WALKING_UPSTAIRS                 449
```

In [18]:

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

```
2947/2947 [==============================] - 2s 618us/step
```

In [19]:

```
score
```

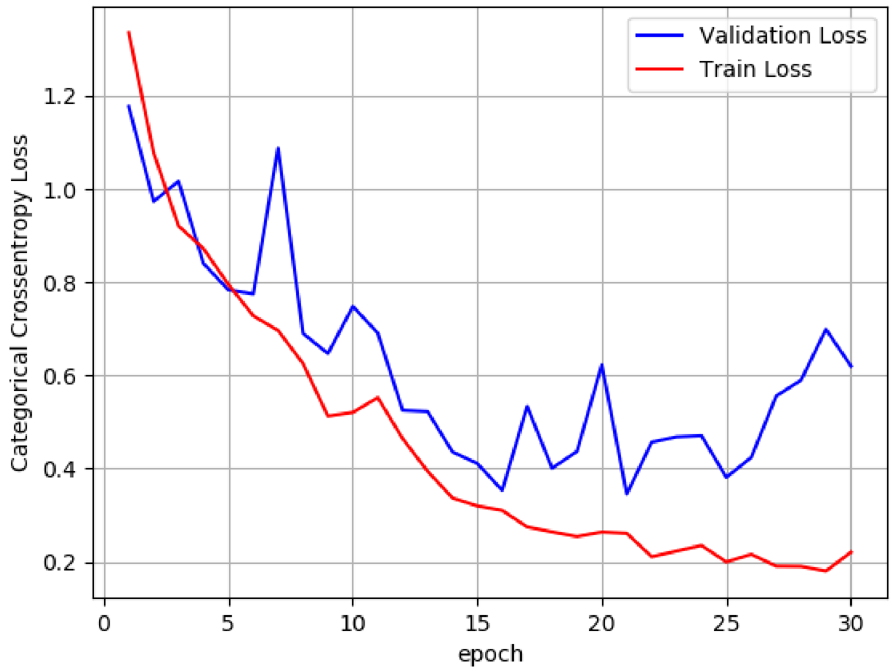
Out[19]:

```
[0.6202012920251873, 0.8903970139124533]
```

In [21]:

```
# this function is used to update the plots for each epoch and error
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()

%matplotlib notebook
import matplotlib.pyplot as plt

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)
```



- With a simple 2 layer architecture and number of hidden layers=64 , we got 89.03% accuracy and a loss of 0.620

n_hidden=256 dropout_rate=0.80

In [22]:

```
# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 256

import warnings
warnings.filterwarnings("ignore")

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

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))

# 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(0.80))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()

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

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

```
128
9
7352
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_2 (LSTM)                (None, 256)               272384    
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 1542      
=================================================================
Total params: 273,926
Trainable params: 273,926
Non-trainable params: 0
_________________________________________________________________
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 351s 48ms/step - loss: 1.3458 - acc: 0.4320 - val_los
s: 1.1183 - val_acc: 0.5640
Epoch 2/30
7352/7352 [==============================] - 435s 59ms/step - loss: 1.2954 - acc: 0.4504 - val_los
s: 1.2204 - val_acc: 0.4374
Epoch 3/30
7352/7352 [==============================] - 428s 58ms/step - loss: 1.1648 - acc: 0.4943 - val_los
s: 0.8888 - val_acc: 0.5955
Epoch 4/30
7352/7352 [==============================] - 509s 69ms/step - loss: 0.8625 - acc: 0.6215 - val_los
s: 0.7889 - val_acc: 0.6624
Epoch 5/30
7352/7352 [==============================] - 923s 126ms/step - loss: 0.9022 - acc: 0.6045 - val_lo
ss: 0.7366 - val_acc: 0.6824
Epoch 6/30
7352/7352 [==============================] - 548s 75ms/step - loss: 0.7698 - acc: 0.6797 - val_los
s: 0.7038 - val_acc: 0.7292
Epoch 7/30
7352/7352 [==============================] - 505s 69ms/step - loss: 0.5612 - acc: 0.7889 - val_los
s: 0.5613 - val_acc: 0.8052
Epoch 8/30
7352/7352 [==============================] - 506s 69ms/step - loss: 0.4134 - acc: 0.8651 - val_los
s: 0.6767 - val_acc: 0.7913
Epoch 9/30
7352/7352 [==============================] - 509s 69ms/step - loss: 0.4659 - acc: 0.8384 - val_los
s: 0.3927 - val_acc: 0.8789
Epoch 10/30
7352/7352 [==============================] - 512s 70ms/step - loss: 0.2404 - acc: 0.9210 - val_los
s: 0.3165 - val_acc: 0.9070
Epoch 11/30
7352/7352 [==============================] - 520s 71ms/step - loss: 0.2328 - acc: 0.9260 - val_los
s: 0.3286 - val_acc: 0.8951
Epoch 12/30
7352/7352 [==============================] - 503s 68ms/step - loss: 0.2022 - acc: 0.9313 - val_los
s: 0.3939 - val_acc: 0.8622
Epoch 13/30
7352/7352 [==============================] - 385s 52ms/step - loss: 0.1957 - acc: 0.9331 - val_los
s: 0.2523 - val_acc: 0.9148
Epoch 14/30
7352/7352 [==============================] - 586s 80ms/step - loss: 0.1718 - acc: 0.9382 - val_los
s: 0.2700 - val_acc: 0.9091
Epoch 15/30
7352/7352 [==============================] - 2671s 363ms/step - loss: 0.2045 - acc: 0.9319 - val_l
oss: 0.2653 - val_acc: 0.9155
Epoch 16/30
7352/7352 [==============================] - 299s 41ms/step - loss: 0.1940 - acc: 0.9329 - val_los
s: 0.2386 - val_acc: 0.9308
Epoch 17/30
7352/7352 [==============================] - 373s 51ms/step - loss: 0.1677 - acc: 0.9433 - val_los
s: 0.5916 - val_acc: 0.9053
Epoch 18/30
```

```
7352/7352 [==============================] - 426s 58ms/step - loss: 0.1900 - acc: 0.9411 - val_los
s: 0.2795 - val_acc: 0.9304
Epoch 19/30
7352/7352 [==============================] - 539s 73ms/step - loss: 0.1725 - acc: 0.9425 - val_los
s: 0.3244 - val_acc: 0.8751
Epoch 20/30
7352/7352 [==============================] - 477s 65ms/step - loss: 0.1772 - acc: 0.9407 - val_los
s: 0.3203 - val_acc: 0.9135
Epoch 21/30
7352/7352 [==============================] - 546s 74ms/step - loss: 0.2150 - acc: 0.9230 - val_los
s: 0.2621 - val_acc: 0.9165
Epoch 22/30
7352/7352 [==============================] - 568s 77ms/step - loss: 0.1560 - acc: 0.9431 - val_los
s: 0.3936 - val_acc: 0.9006
Epoch 23/30
7352/7352 [==============================] - 518s 70ms/step - loss: 0.1721 - acc: 0.9464 - val_los
s: 0.3289 - val_acc: 0.9121
Epoch 24/30
7352/7352 [==============================] - 526s 72ms/step - loss: 0.1978 - acc: 0.9408 - val_los
s: 0.4607 - val_acc: 0.9148
Epoch 25/30
7352/7352 [==============================] - 274s 37ms/step - loss: 0.1503 - acc: 0.9480 - val_los
s: 0.4520 - val_acc: 0.9125
Epoch 26/30
7352/7352 [==============================] - 471s 64ms/step - loss: 0.1677 - acc: 0.9444 - val_los
s: 0.3685 - val_acc: 0.9155
Epoch 27/30
7352/7352 [==============================] - 29366s 4s/step - loss: 0.1688 - acc: 0.9423 - val_los
s: 0.6145 - val_acc: 0.9057
Epoch 28/30
7352/7352 [==============================] - 466s 63ms/step - loss: 0.1685 - acc: 0.9453 - val_los
s: 0.5743 - val_acc: 0.9006
Epoch 29/30
7352/7352 [==============================] - 524s 71ms/step - loss: 0.1972 - acc: 0.9389 - val_los
s: 0.6403 - val_acc: 0.8979
Epoch 30/30
7352/7352 [==============================] - 521s 71ms/step - loss: 0.1971 - acc: 0.9418 - val_los
s: 0.4554 - val_acc: 0.9179
```

In [25]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 535        0         0        0                   0
SITTING                  7      367       116        1                   0
STANDING                 0       70       460        0                   0
WALKING                  0        0         0      478                   0
WALKING_DOWNSTAIRS       0        0         0        0                 396
WALKING_UPSTAIRS         0        0         0        1                   1

Pred                WALKING_UPSTAIRS
True
LAYING                             2
SITTING                            0
STANDING                           2
WALKING                           18
WALKING_DOWNSTAIRS                24
WALKING_UPSTAIRS                 469
```

In [26]:

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

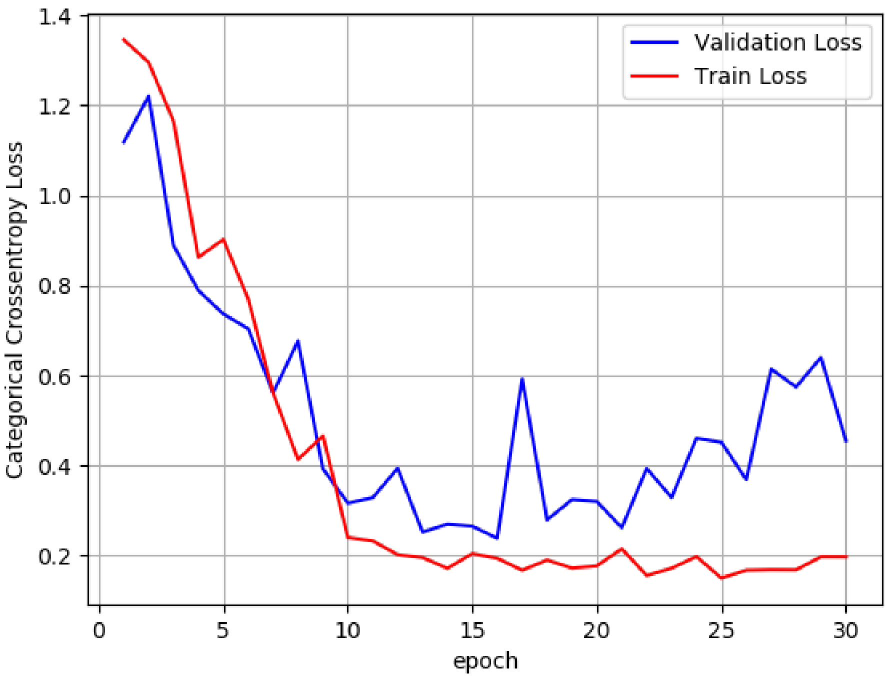
```
2947/2947 [==============================] - 37s 13ms/step
```

Out[26]:

```
[0.4554048873649215, 0.9178825924669155]
```

In [27]:

```
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)
```



- With a simple 2 layer architecture and number of hidden layers=256 and dropout_rate=0.75 , we got 91.78% accuracy and a loss of 0.45

2 LSTM Layers + Larger Dropout

In [29]:

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

import warnings
warnings.filterwarnings("ignore")

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

```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))

# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(128, input_shape=(timesteps, input_dim), return_sequences=True))
model.add(LSTM(64, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.8))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()

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

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

```
128
9
7352
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_3 (LSTM)                (None, 128, 128)          70656     
_________________________________________________________________
lstm_4 (LSTM)                (None, 64)                49408     
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 390       
=================================================================
Total params: 120,454
Trainable params: 120,454
Non-trainable params: 0
_________________________________________________________________
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 433s 59ms/step - loss: 1.1594 - acc: 0.5226 - val_los
s: 1.2993 - val_acc: 0.4462
Epoch 2/30
7352/7352 [==============================] - 410s 56ms/step - loss: 0.8752 - acc: 0.6064 - val_los
s: 0.8001 - val_acc: 0.6013
Epoch 3/30
7352/7352 [==============================] - 434s 59ms/step - loss: 0.8000 - acc: 0.6302 - val_los
s: 0.7883 - val_acc: 0.6213
Epoch 4/30
7352/7352 [==============================] - 11220s 2s/step - loss: 0.7373 - acc: 0.6425 - val_los
s: 0.8382 - val_acc: 0.6003
Epoch 5/30
7352/7352 [==============================] - 40430s 5s/step - loss: 0.7109 - acc: 0.6450 - val_los
s: 0.6489 - val_acc: 0.6261
Epoch 6/30
7352/7352 [==============================] - 206s 28ms/step - loss: 0.6649 - acc: 0.6585 - val_los
s: 2.0368 - val_acc: 0.4703
Epoch 7/30
7352/7352 [==============================] - 238s 32ms/step - loss: 0.6388 - acc: 0.6646 - val_los
s: 0.7069 - val_acc: 0.6250
Epoch 8/30
7352/7352 [==============================] - 404s 55ms/step - loss: 0.6321 - acc: 0.6840 - val_los
s: 0.7945 - val_acc: 0.6271
Epoch 9/30
7352/7352 [==============================] - 493s 67ms/step - loss: 0.7077 - acc: 0.6632 - val_los
```

```
s: 0.6919 - val_acc: 0.6111
Epoch 10/30
7352/7352 [==============================] - 432s 59ms/step - loss: 0.5139 - acc: 0.7631 - val_los
s: 0.4655 - val_acc: 0.7750
Epoch 11/30
7352/7352 [==============================] - 389s 53ms/step - loss: 0.4362 - acc: 0.8547 - val_los
s: 0.3067 - val_acc: 0.8901
Epoch 12/30
7352/7352 [==============================] - 2160s 294ms/step - loss: 0.5382 - acc: 0.8220 - val_l
oss: 0.4236 - val_acc: 0.8772
Epoch 13/30
7352/7352 [==============================] - 421s 57ms/step - loss: 0.3226 - acc: 0.9121 - val_los
s: 0.6795 - val_acc: 0.8171
Epoch 14/30
7352/7352 [==============================] - 66633s 9s/step - loss: 0.4727 - acc: 0.8441 - val_los
s: 0.6040 - val_acc: 0.8310
Epoch 15/30
7352/7352 [==============================] - 263s 36ms/step - loss: 0.2988 - acc: 0.9128 - val_los
s: 0.2825 - val_acc: 0.9060
Epoch 16/30
7352/7352 [==============================] - 246s 33ms/step - loss: 0.2684 - acc: 0.9188 - val_los
s: 0.7409 - val_acc: 0.8510
Epoch 17/30
7352/7352 [==============================] - 423s 57ms/step - loss: 0.2310 - acc: 0.9320 - val_los
s: 0.4245 - val_acc: 0.8972
Epoch 18/30
7352/7352 [==============================] - 427s 58ms/step - loss: 0.2285 - acc: 0.9327 - val_los
s: 0.3312 - val_acc: 0.9175
Epoch 19/30
7352/7352 [==============================] - 427s 58ms/step - loss: 0.3522 - acc: 0.9139 - val_los
s: 0.7232 - val_acc: 0.8663
Epoch 20/30
7352/7352 [==============================] - 422s 57ms/step - loss: 0.2270 - acc: 0.9236 - val_los
s: 0.3252 - val_acc: 0.9077
Epoch 21/30
7352/7352 [==============================] - 400s 54ms/step - loss: 0.2189 - acc: 0.9331 - val_los
s: 0.2765 - val_acc: 0.9155
Epoch 22/30
7352/7352 [==============================] - 434s 59ms/step - loss: 0.1870 - acc: 0.9362 - val_los
s: 0.4661 - val_acc: 0.8941
Epoch 23/30
7352/7352 [==============================] - 440s 60ms/step - loss: 0.2115 - acc: 0.9399 - val_los
s: 0.4888 - val_acc: 0.8985
Epoch 24/30
7352/7352 [==============================] - 435s 59ms/step - loss: 0.2311 - acc: 0.9313 - val_los
s: 0.4473 - val_acc: 0.9030
Epoch 25/30
7352/7352 [==============================] - 453s 62ms/step - loss: 0.2117 - acc: 0.9350 - val_los
s: 0.3518 - val_acc: 0.9162
Epoch 26/30
7352/7352 [==============================] - 482s 66ms/step - loss: 0.2655 - acc: 0.9274 - val_los
s: 0.3591 - val_acc: 0.9050
Epoch 27/30
7352/7352 [==============================] - 429s 58ms/step - loss: 0.2138 - acc: 0.9391 - val_los
s: 0.4368 - val_acc: 0.8877
Epoch 28/30
7352/7352 [==============================] - 425s 58ms/step - loss: 0.1724 - acc: 0.9391 - val_los
s: 0.5297 - val_acc: 0.9030
Epoch 29/30
7352/7352 [==============================] - 428s 58ms/step - loss: 0.1958 - acc: 0.9395 - val_los
s: 2.0814 - val_acc: 0.6994
Epoch 30/30
7352/7352 [==============================] - 431s 59ms/step - loss: 0.1920 - acc: 0.9372 - val_los
s: 0.4494 - val_acc: 0.8999
```

In [30]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 516        0        21        0                   0
SITTING                  0      379       112        0                   0
STANDING                 0       91       441        0                   0
WALKING                  0        0         0      444                  40
```

```
WALKING_DOWNSTAIRS        0         0          0         4                  410
WALKING_UPSTAIRS          0         1          5         0                    3

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            0
STANDING                           0
WALKING                           12
WALKING_DOWNSTAIRS                 6
WALKING_UPSTAIRS                 462
```

In [31]:

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

```
2947/2947 [==============================] - 27s 9ms/step
```

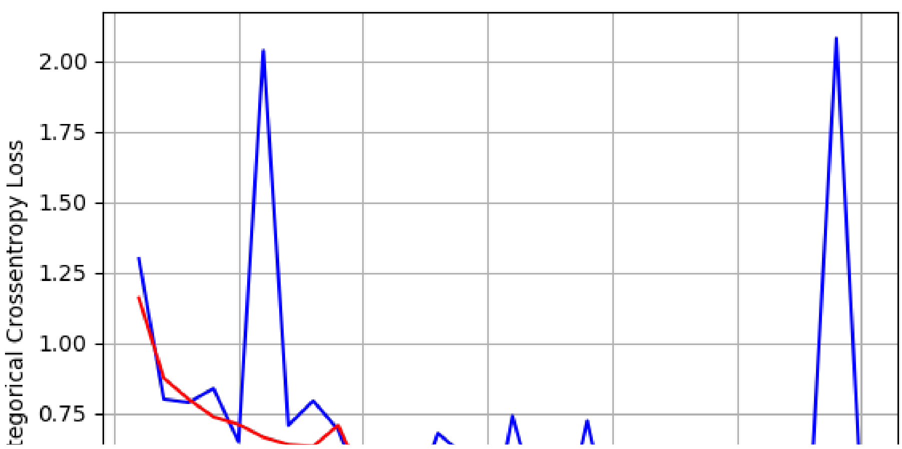
Out[31]:

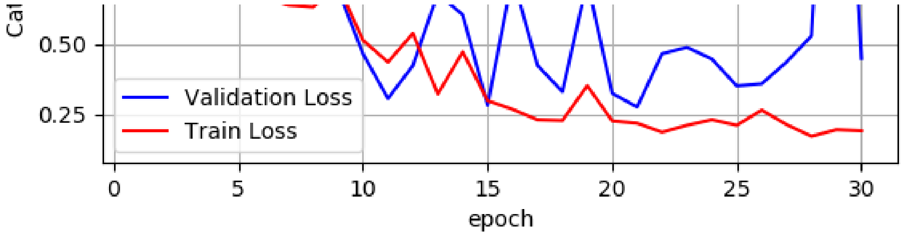
```
[0.44939401513188654, 0.8998982015609094]
```

Observation: With 2 LSTM Layers and larger dropout , accuracy is 89.98% with loss of 0.449

In [32]:

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

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 [40]:

```
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Model", "Num_hidden_layers", "Dropout_rate", "Loss", "Accuracy"]

x.add_row(["LSTM", 64 , 0.75,0.6202012920251873,0.8903970139124533])
x.add_row(["LSTM", 256, 0.80,0.4554048873649215,0.9178825924669155])
x.add_row(["2-LSTM", "128 and 64",0.8,0.44939401513188654, 0.8998982015609094])

print(x)
```

```
+--------+-------------------+--------------+---------------------+--------------------+
| Model  | Num_hidden_layers | Dropout_rate |         Loss        |      Accuracy      |
+--------+-------------------+--------------+---------------------+--------------------+
|  LSTM  |         64        |     0.75     |  0.6202012920251873 | 0.8903970139124533 |
|  LSTM  |        256        |     0.8      |  0.4554048873649215 | 0.9178825924669155 |
| 2-LSTM |     128 and 64    |     0.8      | 0.44939401513188654 | 0.8998982015609094 |
+--------+-------------------+--------------+---------------------+--------------------+
```