In [0]:

```python
import zipfile
```

In [0]:

```python
zip_ref = zipfile.ZipFile("/content/drive/My Drive/HAR data/HumanActivityRecognition.zip", 'r')
zip_ref.extractall("/content/drive/My Drive/HAR data")
zip_ref.close()
```

In [0]:

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

In [0]:

```python
import matplotlib.pyplot as plt
import numpy as np
import time
# https://gist.github.com/greydanus/f6eee59eaf1d90fcb3b534a25362cea4
# https://stackoverflow.com/a/14434334
# this function is used to update the plots for each epoch and error
def plt_dynamic(x, vy, ty, 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 [0]:

```python
# 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 [0]:

```python
# Data directory
DATADIR = '/content/drive/My Drive/HAR data/HAR/UCI_HAR_Dataset'
```

In [0]:

```python
# 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 [0]:

```
# 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'/content/drive/My Drive/HAR data/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 [0]:

```
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'/content/drive/My Drive/HAR data/HAR/UCI_HAR_Dataset/{subset}/y_{subset}.txt'
    y = _read_csv(filename)[0]

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

In [0]:

```
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 [0]:

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

In [0]:

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

,

In [0]:

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

In [0]:

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

In [0]:

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

In [0]:

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

Train Test Split

In [0]:

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

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:11: FutureWarning: Method .as_matrix
will be removed in a future version. Use .values instead.
  # This is added back by InteractiveShellApp.init_path()
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:12: FutureWarning: Method .as_matrix
will be removed in a future version. Use .values instead.
  if sys.path[0] == '':
```

In [18]:

```
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
```

Store into pickle

In [0]:

```
%%time
import pickle

dbfile_1 = open('/content/drive/My Drive/HAR data/Xtrain', 'ab')
dbfile_2 = open('/content/drive/My Drive/HAR data/Xtest', 'ab')
```

```
dbfile_3 = open('/content/drive/My Drive/HAR data/ytrain', 'ab')
dbfile_4 = open('/content/drive/My Drive/HAR data/ytest', 'ab')

# source, destination
pickle.dump(X_train, dbfile_1)
pickle.dump(X_test, dbfile_2)
pickle.dump(Y_train, dbfile_3)
pickle.dump(Y_test, dbfile_4)

dbfile_1.close()
dbfile_2.close()
dbfile_3.close()
dbfile_4.close()
```

```
CPU times: user 128 ms, sys: 65 ms, total: 193 ms
Wall time: 356 ms
```

In [0]:

```
'''import pickle
dbfile_1 = open('/content/drive/My Drive/HAR data/Xtrain', 'rb')
X_train = pickle.load(dbfile_1)
dbfile_1.close()

dbfile_2 = open('/content/drive/My Drive/HAR data/Xtest', 'rb')
X_test = pickle.load(dbfile_2)
dbfile_2.close()

dbfile_3 = open('/content/drive/My Drive/HAR data/ytrain', 'rb')
Y_train = pickle.load(dbfile_3)
dbfile_3.close()

dbfile_4 = open('/content/drive/My Drive/HAR data/ytest', 'rb')
Y_test = pickle.load(dbfile_4)
dbfile_4.close()'''
```

Model 1

- Defining the Architecture of LSTM

In [0]:

```
# 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()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 32)                5376
_________________________________________________________________
dropout_3 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 198
=================================================================
Total params: 5,574
Trainable params: 5,574
Non-trainable params: 0
_________________________________________________________________
```

In [0]:

```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
```

```
              metrics=['accuracy'])
```

In [0]:

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

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 92s 13ms/step - loss: 1.3018 - acc: 0.4395 - val_loss
: 1.1254 - val_acc: 0.4662
Epoch 2/30
7352/7352 [==============================] - 94s 13ms/step - loss: 0.9666 - acc: 0.5880 - val_loss
: 0.9491 - val_acc: 0.5714
Epoch 3/30
7352/7352 [==============================] - 97s 13ms/step - loss: 0.7812 - acc: 0.6408 - val_loss
: 0.8286 - val_acc: 0.5850
Epoch 4/30
7352/7352 [==============================] - 95s 13ms/step - loss: 0.6941 - acc: 0.6574 - val_loss
: 0.7297 - val_acc: 0.6128
Epoch 5/30
7352/7352 [==============================] - 92s 13ms/step - loss: 0.6336 - acc: 0.6912 - val_loss
: 0.7359 - val_acc: 0.6787
Epoch 6/30
7352/7352 [==============================] - 94s 13ms/step - loss: 0.5859 - acc: 0.7134 - val_loss
: 0.7015 - val_acc: 0.6939
Epoch 7/30
7352/7352 [==============================] - 95s 13ms/step - loss: 0.5692 - acc: 0.7477 - val_loss
: 0.5995 - val_acc: 0.7387
Epoch 8/30
7352/7352 [==============================] - 96s 13ms/step - loss: 0.4899 - acc: 0.7809 - val_loss
: 0.5762 - val_acc: 0.7387
Epoch 9/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.4482 - acc: 0.7886 - val_loss
: 0.7413 - val_acc: 0.7126
Epoch 10/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.4132 - acc: 0.8077 - val_loss
: 0.5048 - val_acc: 0.7513
Epoch 11/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.3985 - acc: 0.8274 - val_loss
: 0.5234 - val_acc: 0.7452
Epoch 12/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.3378 - acc: 0.8638 - val_loss
: 0.4114 - val_acc: 0.8833
Epoch 13/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.2947 - acc: 0.9051 - val_loss
: 0.4386 - val_acc: 0.8731
Epoch 14/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.2448 - acc: 0.9291 - val_loss
: 0.3768 - val_acc: 0.8921
Epoch 15/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.2157 - acc: 0.9331 - val_loss
: 0.4441 - val_acc: 0.8931
Epoch 16/30
7352/7352 [==============================] - 90s 12ms/step - loss: 0.2053 - acc: 0.9366 - val_loss
: 0.4162 - val_acc: 0.8968
Epoch 17/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.2028 - acc: 0.9404 - val_loss
: 0.4538 - val_acc: 0.8962
Epoch 18/30
7352/7352 [==============================] - 93s 13ms/step - loss: 0.1911 - acc: 0.9419 - val_loss
: 0.3964 - val_acc: 0.8999
Epoch 19/30
7352/7352 [==============================] - 96s 13ms/step - loss: 0.1912 - acc: 0.9407 - val_loss
: 0.3165 - val_acc: 0.9030
Epoch 20/30
7352/7352 [==============================] - 96s 13ms/step - loss: 0.1732 - acc: 0.9446 - val_loss
: 0.4546 - val_acc: 0.8904
Epoch 21/30
7352/7352 [==============================] - 94s 13ms/step - loss: 0.1782 - acc: 0.9444 - val_loss
: 0.3346 - val_acc: 0.9063
```

```
Epoch 22/30
7352/7352 [==============================] - 95s 13ms/step - loss: 0.1812 - acc: 0.9418 - val_loss
: 0.8164 - val_acc: 0.8582
Epoch 23/30
7352/7352 [==============================] - 95s 13ms/step - loss: 0.1824 - acc: 0.9426 - val_loss
: 0.4240 - val_acc: 0.9036
Epoch 24/30
7352/7352 [==============================] - 94s 13ms/step - loss: 0.1726 - acc: 0.9429 - val_loss
: 0.4067 - val_acc: 0.9148
Epoch 25/30
7352/7352 [==============================] - 96s 13ms/step - loss: 0.1737 - acc: 0.9411 - val_loss
: 0.3396 - val_acc: 0.9074
Epoch 26/30
7352/7352 [==============================] - 96s 13ms/step - loss: 0.1650 - acc: 0.9461 - val_loss
: 0.3806 - val_acc: 0.9019
Epoch 27/30
7352/7352 [==============================] - 89s 12ms/step - loss: 0.1925 - acc: 0.9415 - val_loss
: 0.6464 - val_acc: 0.8850
Epoch 28/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.1965 - acc: 0.9425 - val_loss
: 0.3363 - val_acc: 0.9203
Epoch 29/30
7352/7352 [==============================] - 92s 12ms/step - loss: 0.1889 - acc: 0.9431 - val_loss
: 0.3737 - val_acc: 0.9158
Epoch 30/30
7352/7352 [==============================] - 95s 13ms/step - loss: 0.1945 - acc: 0.9414 - val_loss
: 0.3088 - val_acc: 0.9097
```

Out[0]:

```
<keras.callbacks.History at 0x29b5ee36a20>
```

In [0]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 512        0        25        0                   0
SITTING                  3      410        75        0                   0
STANDING                 0       87       445        0                   0
WALKING                  0        0         0      481                   2
WALKING_DOWNSTAIRS       0        0         0        0                 382
WALKING_UPSTAIRS         0        0         0        2                  18

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            3
STANDING                           0
WALKING                           13
WALKING_DOWNSTAIRS                38
WALKING_UPSTAIRS                 451
```

In [0]:

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

```
2947/2947 [==============================] - 4s 2ms/step
```

In [0]:

```
score
```

Out[0]:

```
[0.3087582236972612, 0.9097387173396675]
```

Model 2

In [30]:

```
# With One LSTM Layer Model 1

model = Sequential()

# 1 LSTM layer
model.add(LSTM(n_hidden, input_shape = (timesteps, input_dim)))
model.add(Dropout(0.25))
model.add(Dense(n_classes, activation = 'relu'))
model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
print(model.summary())
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 32)                5376
_________________________________________________________________
dropout_2 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 198
=================================================================
Total params: 5,574
Trainable params: 5,574
Non-trainable params: 0
_________________________________________________________________
None
```

In [31]:

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

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 10s 1ms/step - loss: 0.6114 - acc: 0.8331 - val_loss: 0.4062 - val_acc: 0.8537
Epoch 2/30
7352/7352 [==============================] - 9s 1ms/step - loss: 0.3978 - acc: 0.8531 - val_loss: 0.3853 - val_acc: 0.8613
Epoch 3/30
7352/7352 [==============================] - 9s 1ms/step - loss: 0.3711 - acc: 0.8528 - val_loss: 0.3690 - val_acc: 0.8606
Epoch 4/30
7352/7352 [==============================] - 9s 1ms/step - loss: 0.3640 - acc: 0.8465 - val_loss: 0.3767 - val_acc: 0.8463
Epoch 5/30
7352/7352 [==============================] - 9s 1ms/step - loss: 0.3483 - acc: 0.8488 - val_loss: 0.3553 - val_acc: 0.8466
Epoch 6/30
7352/7352 [==============================] - 10s 1ms/step - loss: 0.3224 - acc: 0.8537 - val_loss: 0.3445 - val_acc: 0.8430
Epoch 7/30
7352/7352 [==============================] - 10s 1ms/step - loss: 0.3085 - acc: 0.8558 - val_loss: 0.3386 - val_acc: 0.8489
Epoch 8/30
7352/7352 [==============================] - 10s 1ms/step - loss: 0.3019 - acc: 0.8552 - val_loss: 0.3288 - val_acc: 0.8465
Epoch 9/30
7352/7352 [==============================] - 10s 1ms/step - loss: 0.3060 - acc: 0.8530 - val_loss: 0.3330 - val_acc: 0.8414
Epoch 10/30
7352/7352 [==============================] - 9s 1ms/step - loss: 0.2876 - acc: 0.8563 - val_loss: 0.2848 - val_acc: 0.8536
Epoch 11/30
7352/7352 [==============================] - 9s 1ms/step - loss: 0.2385 - acc: 0.8687 - val_loss: 0.2610 - val_acc: 0.8743
Epoch 12/30
7352/7352 [==============================] - 10s 1ms/step - loss: 0.2214 - acc: 0.8774 - val_loss:
```

```
0.2659 - val_acc: 0.8687
Epoch 13/30
7352/7352 [==============================] - 10s 1ms/step - loss: 0.2202 - acc: 0.8725 - val_loss:
0.2505 - val_acc: 0.8721
Epoch 14/30
7352/7352 [==============================] - 10s 1ms/step - loss: 0.2060 - acc: 0.8751 - val_loss:
0.2575 - val_acc: 0.8678
Epoch 15/30
7352/7352 [==============================] - 10s 1ms/step - loss: 0.2164 - acc: 0.8714 - val_loss:
0.2541 - val_acc: 0.8685
Epoch 16/30
7352/7352 [==============================] - 10s 1ms/step - loss: 0.2013 - acc: 0.8754 - val_loss:
0.2426 - val_acc: 0.8730
Epoch 17/30
7352/7352 [==============================] - 10s 1ms/step - loss: 0.4133 - acc: 0.8520 - val_loss:
0.2799 - val_acc: 0.8526
Epoch 18/30
7352/7352 [==============================] - 10s 1ms/step - loss: 0.2380 - acc: 0.8553 - val_loss:
0.2532 - val_acc: 0.8554
Epoch 19/30
7352/7352 [==============================] - 10s 1ms/step - loss: 0.2155 - acc: 0.8639 - val_loss:
0.2320 - val_acc: 0.8756
Epoch 20/30
7352/7352 [==============================] - 10s 1ms/step - loss: 0.2046 - acc: 0.8751 - val_loss:
0.2358 - val_acc: 0.8738
Epoch 21/30
7352/7352 [==============================] - 10s 1ms/step - loss: 0.1891 - acc: 0.8772 - val_loss:
0.2349 - val_acc: 0.8748
Epoch 22/30
7352/7352 [==============================] - 10s 1ms/step - loss: 0.2241 - acc: 0.8765 - val_loss:
0.4929 - val_acc: 0.8196
Epoch 23/30
7352/7352 [==============================] - 10s 1ms/step - loss: 0.4119 - acc: 0.8355 - val_loss:
0.3371 - val_acc: 0.8367
Epoch 24/30
7352/7352 [==============================] - 9s 1ms/step - loss: 0.2626 - acc: 0.8525 - val_loss:
0.3211 - val_acc: 0.8519
Epoch 25/30
7352/7352 [==============================] - 10s 1ms/step - loss: 0.2514 - acc: 0.8576 - val_loss:
0.3072 - val_acc: 0.8558
Epoch 26/30
7352/7352 [==============================] - 9s 1ms/step - loss: 0.2366 - acc: 0.8631 - val_loss:
0.2963 - val_acc: 0.8589
Epoch 27/30
7352/7352 [==============================] - 10s 1ms/step - loss: 0.2270 - acc: 0.8681 - val_loss:
0.2749 - val_acc: 0.8649
Epoch 28/30
7352/7352 [==============================] - 10s 1ms/step - loss: 0.2103 - acc: 0.8725 - val_loss:
0.2722 - val_acc: 0.8684
Epoch 29/30
7352/7352 [==============================] - 10s 1ms/step - loss: 0.2023 - acc: 0.8798 - val_loss:
0.2991 - val_acc: 0.8760
Epoch 30/30
7352/7352 [==============================] - 10s 1ms/step - loss: 0.2483 - acc: 0.8713 - val_loss:
0.2719 - val_acc: 0.8700
```

In [32]:

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

```
2947/2947 [==============================] - 1s 302us/step
[0.2719094553862105, 0.8700373047159452]
```

In [0]:

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

```
Pred                LAYING  SITTING  ...  WALKING_DOWNSTAIRS  WALKING_UPSTAIRS
True                                 ...
LAYING                 506        0  ...                   4                27
SITTING                  0      222  ...                   0                26
STANDING                 0        9  ...                   0               111
```

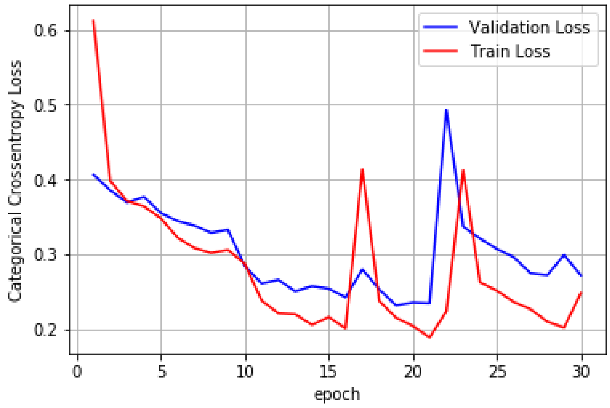
```
WALKING                   0       28  ...                 28                184
WALKING_DOWNSTAIRS        0        6  ...                363                 26
WALKING_UPSTAIRS          0        0  ...                 40                428

[6 rows x 6 columns]
```

In [0]:

```
%matplotlib inline
```

In [36]:

```
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)
```



Model 3

In [0]:

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

model = Sequential()

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

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

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
```

```
=================================================================
lstm_4 (LSTM)                (None, 80)                28800
_________________________________________________________________
dropout_4 (Dropout)          (None, 80)                0
_________________________________________________________________
dense_4 (Dense)              (None, 6)                 486
=================================================================
Total params: 29,286
Trainable params: 29,286
Non-trainable params: 0
_________________________________________________________________
None
```

In [0]:

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

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.4190 - acc: 0.8390 - val_loss: 0.3644 - val_acc: 0.8571
Epoch 2/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.3535 - acc: 0.8616 - val_loss: 0.3331 - val_acc: 0.8613
Epoch 3/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.3183 - acc: 0.8736 - val_loss: 0.3162 - val_acc: 0.8704
Epoch 4/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2887 - acc: 0.8776 - val_loss: 0.2728 - val_acc: 0.8806
Epoch 5/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2342 - acc: 0.8948 - val_loss: 0.2279 - val_acc: 0.8955
Epoch 6/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2182 - acc: 0.8994 - val_loss: 0.2412 - val_acc: 0.8933
Epoch 7/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2087 - acc: 0.9015 - val_loss: 0.2105 - val_acc: 0.9009
Epoch 8/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1874 - acc: 0.9109 - val_loss: 0.2073 - val_acc: 0.9037
Epoch 9/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1813 - acc: 0.9161 - val_loss: 0.1885 - val_acc: 0.9164
Epoch 10/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1476 - acc: 0.9429 - val_loss: 0.1629 - val_acc: 0.9406
Epoch 11/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1349 - acc: 0.9464 - val_loss: 0.1546 - val_acc: 0.9390
Epoch 12/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1186 - acc: 0.9572 - val_loss: 0.1608 - val_acc: 0.9467
Epoch 13/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0865 - acc: 0.9713 - val_loss: 0.1619 - val_acc: 0.9440
Epoch 14/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0809 - acc: 0.9730 - val_loss: 0.1272 - val_acc: 0.9545
Epoch 15/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0792 - acc: 0.9726 - val_loss: 0.1019 - val_acc: 0.9637
Epoch 16/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1704 - acc: 0.9347 - val_loss: 0.1586 - val_acc: 0.9412
Epoch 17/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0992 - acc: 0.9674 - val_loss: 0.1695 - val_acc: 0.9481
Epoch 18/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1203 - acc: 0.9567 - val_loss:
```

```
0.1166 - val_acc: 0.9592
Epoch 19/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0920 - acc: 0.9701 - val_loss:
0.1421 - val_acc: 0.9531
Epoch 20/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1147 - acc: 0.9616 - val_loss:
0.1311 - val_acc: 0.9570
Epoch 21/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0738 - acc: 0.9737 - val_loss:
0.1042 - val_acc: 0.9641
Epoch 22/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.0671 - acc: 0.9752 - val_loss:
0.1112 - val_acc: 0.9606
Epoch 23/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1143 - acc: 0.9567 - val_loss:
0.1159 - val_acc: 0.9583
Epoch 24/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.0743 - acc: 0.9738 - val_loss:
0.1186 - val_acc: 0.9601
Epoch 25/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0684 - acc: 0.9777 - val_loss:
0.1046 - val_acc: 0.9650
Epoch 26/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0576 - acc: 0.9798 - val_loss:
0.0958 - val_acc: 0.9663
Epoch 27/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0481 - acc: 0.9819 - val_loss:
0.0974 - val_acc: 0.9646
Epoch 28/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0474 - acc: 0.9816 - val_loss:
0.0986 - val_acc: 0.9676
Epoch 29/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0617 - acc: 0.9770 - val_loss:
0.1099 - val_acc: 0.9645
Epoch 30/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.0488 - acc: 0.9810 - val_loss:
0.0985 - val_acc: 0.9663
```

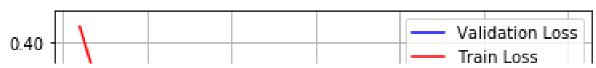
In [0]:

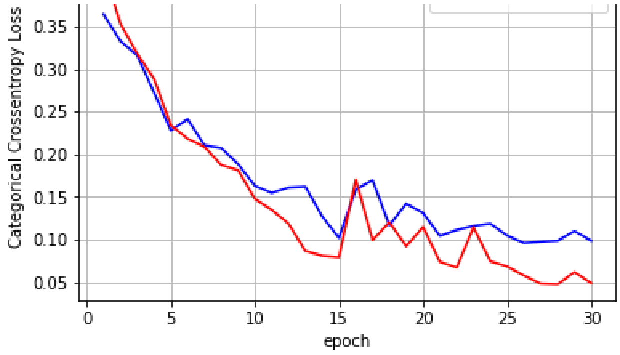
```
score = model.evaluate(X_test, Y_test)
print(score)
```

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

In [0]:

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

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

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





Model 4

In [0]:

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

model = Sequential()

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

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

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

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:66: The name tf.get_default_graph is deprecated. Plea
se use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:541: The name tf.placeholder is deprecated. Please us
e tf.compat.v1.placeholder instead.

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

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:148: The name tf.placeholder_with_default is
deprecated. Please use tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:3733: 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`.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793: The name t
f.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:3657: The name tf.log is deprecated. Please use tf.ma
th.log instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow/python/ops/nn_impl.py:180: add_dispatch_support.<locals>.wrapper (from
tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_1 (LSTM)                (None, 128, 80)           28800     
_________________________________________________________________
dropout_1 (Dropout)          (None, 128, 80)           0         
```

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

In [0]:

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

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0341 - acc: 0.9861 - val_loss:
0.0949 - val_acc: 0.9742
Epoch 2/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0371 - acc: 0.9852 - val_loss:
0.1142 - val_acc: 0.9724
Epoch 3/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0352 - acc: 0.9856 - val_loss:
0.1288 - val_acc: 0.9688
Epoch 4/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0390 - acc: 0.9845 - val_loss:
0.0962 - val_acc: 0.9751
Epoch 5/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0366 - acc: 0.9847 - val_loss:
0.1185 - val_acc: 0.9722
Epoch 6/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0370 - acc: 0.9849 - val_loss:
0.1057 - val_acc: 0.9723
Epoch 7/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0331 - acc: 0.9867 - val_loss:
0.1143 - val_acc: 0.9738
Epoch 8/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0332 - acc: 0.9860 - val_loss:
0.1110 - val_acc: 0.9671
Epoch 9/30
7352/7352 [==============================] - 59s 8ms/step - loss: 0.0319 - acc: 0.9864 - val_loss:
0.1210 - val_acc: 0.9723
Epoch 10/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0354 - acc: 0.9867 - val_loss:
0.1088 - val_acc: 0.9736
Epoch 11/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0304 - acc: 0.9886 - val_loss:
0.1426 - val_acc: 0.9748
Epoch 12/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0331 - acc: 0.9860 - val_loss:
0.2099 - val_acc: 0.9475
Epoch 13/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0332 - acc: 0.9870 - val_loss:
0.1255 - val_acc: 0.9623
Epoch 14/30
7352/7352 [==============================] - 59s 8ms/step - loss: 0.0299 - acc: 0.9883 - val_loss:
0.1483 - val_acc: 0.9718
Epoch 15/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0304 - acc: 0.9877 - val_loss:
0.1413 - val_acc: 0.9688
Epoch 16/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0317 - acc: 0.9885 - val_loss:
0.1288 - val_acc: 0.9744
Epoch 17/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0308 - acc: 0.9879 - val_loss:
0.1279 - val_acc: 0.9727
Epoch 18/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0301 - acc: 0.9886 - val_loss:
```

```
0.1383 - val_acc: 0.9695
Epoch 19/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0303 - acc: 0.9891 - val_loss:
0.1219 - val_acc: 0.9738
Epoch 20/30
7352/7352 [==============================] - 59s 8ms/step - loss: 0.0332 - acc: 0.9872 - val_loss:
0.1546 - val_acc: 0.9686
Epoch 21/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0343 - acc: 0.9877 - val_loss:
0.1032 - val_acc: 0.9755
Epoch 22/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0329 - acc: 0.9874 - val_loss:
0.1271 - val_acc: 0.9674
Epoch 23/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0297 - acc: 0.9876 - val_loss:
0.1098 - val_acc: 0.9737
Epoch 24/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0290 - acc: 0.9892 - val_loss:
0.1386 - val_acc: 0.9716
Epoch 25/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0336 - acc: 0.9884 - val_loss:
0.1342 - val_acc: 0.9760
Epoch 26/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0327 - acc: 0.9875 - val_loss:
0.1325 - val_acc: 0.9699
Epoch 27/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0342 - acc: 0.9871 - val_loss:
0.1163 - val_acc: 0.9701
Epoch 28/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0291 - acc: 0.9890 - val_loss:
0.1486 - val_acc: 0.9640
Epoch 29/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.0303 - acc: 0.9890 - val_loss:
0.1134 - val_acc: 0.9759
Epoch 30/30
7352/7352 [==============================] - 59s 8ms/step - loss: 0.0272 - acc: 0.9901 - val_loss:
0.1277 - val_acc: 0.9695
CPU times: user 32min 43s, sys: 1min 43s, total: 34min 27s
Wall time: 29min 1s
```

In [0]:

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

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

In [0]:

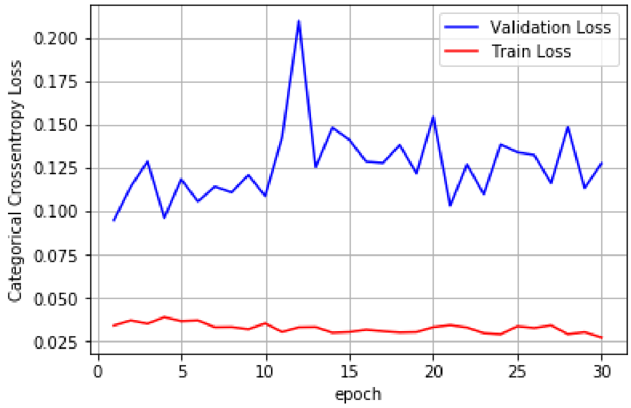
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

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

[6 rows x 6 columns]
```

In [0]:

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

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

# print(history.history.keys())
```

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



Conclusion

In [3]:

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

x.add_row(["Lstm + dropout(0.5)","32","sigmoid","rmsprop" ,"90.97%"])
x.add_row(["Lstm + dropout(0.25)","32","relu","Adam" ,"87%"])
x.add_row(["Lstm + dropout(0.25)","80","sigmoid" ,"Adam", "96.62%"])
x.add_row(["Lstm + dropout(0.25)","80","sigmoid","rmsprop" ,"97.16%"])


print(x)
```

```
+----------------------+--------------+------------+-----------+--------------------+
|        Model         | Hidden layer | activation | Optimizer | Test accuracy in % |
+----------------------+--------------+------------+-----------+--------------------+
| Lstm + dropout(0.5)  |      32      |  sigmoid   |  rmsprop  |       90.97%       |
| Lstm + dropout(0.25) |      32      |    relu    |    Adam   |        87%         |
| Lstm + dropout(0.25) |      80      |  sigmoid   |    Adam   |       96.62%       |
| Lstm + dropout(0.25) |      80      |  sigmoid   |  rmsprop  |       97.16%       |
+----------------------+--------------+------------+-----------+--------------------+
```

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