In [1]:
```
# Importing Libraries
```

In [1]:
```
import pandas as pd
import numpy as np
```

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

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

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

Data

In [3]:
```
# Data directory
DATADIR = 'UCI_HAR_Dataset'
```

In [4]:
```
# Raw data signals
# Signals are from Accelerometer and Gyroscope
# The signals are in x,y,z directions
# Sensor signals are filtered to have only body acceleration
# excluding the acceleration due to gravity
# Triaxial acceleration from the accelerometer is total acceleration
SIGNALS = [
    "body_acc_x",
    "body_acc_y",
    "body_acc_z",
    "body_gyro_x",
    "body_gyro_y",
    "body_gyro_z",
    "total_acc_x",
    "total_acc_y",
    "total_acc_z"
]
```

In [5]:
```
# 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 = 'UCI_HAR_Dataset/'+subset+'/Inertial Signals/'+signal+'_'+subs
        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 [6]:
```
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.htr
    """
    filename = 'UCI_HAR_Dataset/'+subset+'/y_'+subset+'.txt'
    y = _read_csv(filename)[0]

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

In [7]:
```
def load_data():
    """
    Obtain the dataset from multiple files.
    Returns: X_train, X_test, y_train, y_test
    """
    X_train, X_test = load_signals('train'), load_signals('test')
    y_train, y_test = load_y('train'), load_y('test')

    return X_train, X_test, y_train, y_test
```

In [10]:
```
# Importing tensorflow
np.random.seed(42)
import tensorflow as tf
tf.set_random_seed(42)
```

```
/home/j_choudhary1001/anaconda3/lib/python3.6/site-packages/h5py/__init__.py:3
6: FutureWarning: Conversion of the second argument of issubdtype from `float`
to `np.floating` is deprecated. In future, it will be treated as `np.float64 ==
np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
```

In [11]:
```
# Configuring a session
session_conf = tf.ConfigProto(
    intra_op_parallelism_threads=1,
    inter_op_parallelism_threads=1
)
```

In [13]:
```
# Import Keras
from keras import backend as K
sess = tf.Session(graph=tf.get_default_graph(), config=session_conf)
K.set_session(sess)
```

```
Using TensorFlow backend.
```

In [14]:
```
# Importing libraries
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers.core import Dense, Dropout
```

In [15]:
```
# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 32
```

In [16]:
```
# Utility function to count the number of classes
def _count_classes(y):
    return len(set([tuple(category) for category in y]))
```

In [17]:
```
# Loading the train and test data
X_train, X_test, Y_train, Y_test = load_data()
```

```
/home/j_choudhary1001/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.
py:12: FutureWarning: Method .as_matrix will be removed in a future version. Us
e .values instead.
  if sys.path[0] == '':
```

In [20]:
```
timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = _count_classes(Y_train)

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

```
128
9
7352
```

1 layer architecture

In [32]:
```
# 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()
```

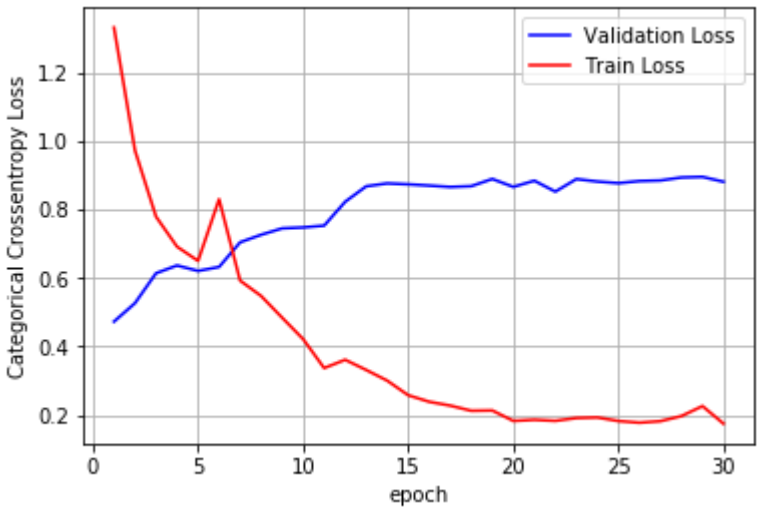
```
WARNING:tensorflow:From C:\Program Files\Anaconda3\lib\site-packages\tensorflow
\python\framework\op_def_library.py:263: colocate_with (from tensorflow.python.
framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From C:\Program Files\Anaconda3\lib\site-packages\keras\back
end\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 - kee
p_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 [33]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [34]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
7352/7352 [==============================] - 46s 6ms/step - loss: 0.6496 - ac
c: 0.6794 - val_loss: 0.8000 - val_acc: 0.6362
Epoch 6/30
7352/7352 [==============================] - 47s 6ms/step - loss: 0.8295 - ac
c: 0.6221 - val_loss: 0.7970 - val_acc: 0.6318
Epoch 7/30
7352/7352 [==============================] - 46s 6ms/step - loss: 0.5920 - ac
c: 0.7122 - val_loss: 0.6816 - val_acc: 0.7038
Epoch 8/30
7352/7352 [==============================] - 47s 6ms/step - loss: 0.5472 - ac
c: 0.7594 - val_loss: 0.6473 - val_acc: 0.7258
Epoch 9/30
7352/7352 [==============================] - 47s 6ms/step - loss: 0.4838 - ac
c: 0.7805 - val_loss: 0.6033 - val_acc: 0.7445
Epoch 10/30
7352/7352 [==============================] - 49s 7ms/step - loss: 0.4212 - ac
c: 0.7907 - val_loss: 0.5396 - val_acc: 0.7472
Epoch 11/30
7352/7352 [==============================] - 49s 7ms/step - loss: 0.3967 - ac
c: 0.8084 - val_loss: 0.5399 - val_acc: 0.7526
```

In [41]:
```
epochs = 30
%matplotlib inline
import matplotlib.pyplot as plt
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
x = list(range(1,epochs+1))
vy = val_loss #history.history['val_loss']
ty = train_loss #history.history['loss']
# plt_dynamic(x, vy, ty, ax)
ax.plot(x, vy, 'b', label="Validation Loss")
ax.plot(x, ty, 'r', label="Train Loss")
plt.legend()
plt.grid()
fig.canvas.draw()
```



In [35]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

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

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

In [36]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 2s 526us/step
```

In [37]:
```
score
```

Out[37]:
```
[0.4692274864947224, 0.8805564981336953]
```

With 64 hidden units and 1 layer, dropout = 0.5

In [38]:
```
# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 64
```

In [39]:
```
# 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()
```

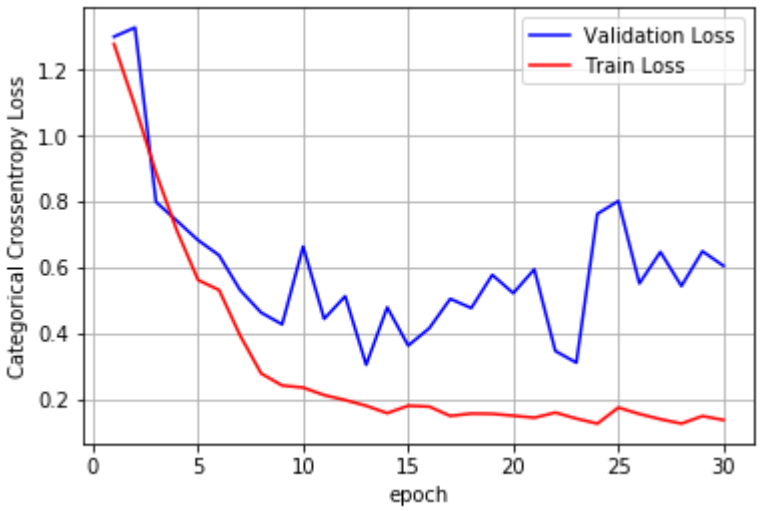
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_2 (LSTM)                (None, 64)                18944     
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 390       
=================================================================
Total params: 19,334
Trainable params: 19,334
Non-trainable params: 0
_________________________________________________________________
```

In [40]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [41]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Epoch 15/30
7352/7352 [==============================] - 60s 8ms/step - loss: 0.1820 - ac
c: 0.9348 - val_loss: 0.3643 - val_acc: 0.8982
Epoch 16/30
7352/7352 [==============================] - 60s 8ms/step - loss: 0.1794 - ac
c: 0.9381 - val_loss: 0.4166 - val_acc: 0.9006
Epoch 17/30
7352/7352 [==============================] - 66s 9ms/step - loss: 0.1511 - ac
c: 0.9437 - val_loss: 0.5058 - val_acc: 0.8890
Epoch 18/30
7352/7352 [==============================] - 61s 8ms/step - loss: 0.1583 - ac
c: 0.9463 - val_loss: 0.4778 - val_acc: 0.8904
Epoch 19/30
7352/7352 [==============================] - 62s 8ms/step - loss: 0.1575 - ac
c: 0.9449 - val_loss: 0.5783 - val_acc: 0.9019
Epoch 20/30
7352/7352 [==============================] - 36433s 5s/step - loss: 0.1519 -
acc: 0.9444 - val_loss: 0.5230 - val_acc: 0.8911
Epoch 21/30
7352/7352 [==============================] - 57s 8ms/step - loss: 0.1457 - ac
```

In [43]:
```
epochs = 30
import matplotlib.pyplot as plt
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
x = list(range(1,epochs+1))
vy = val_loss #history.history['val_loss']
ty = train_loss #history.history['loss']
# plt_dynamic(x, vy, ty, ax)
ax.plot(x, vy, 'b', label="Validation Loss")
ax.plot(x, ty, 'r', label="Train Loss")
plt.legend()
plt.grid()
fig.canvas.draw()
```



In [42]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

Pred	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	\
True						
LAYING	510	0	0	0	0	
SITTING	0	371	101	0	1	
STANDING	0	55	477	0	0	
WALKING	0	0	0	453	30	
WALKING_DOWNSTAIRS	0	0	0	1	418	
WALKING_UPSTAIRS	0	0	0	0	7	

Pred	WALKING_UPSTAIRS
True	
LAYING	27
SITTING	18
STANDING	0
WALKING	13
WALKING_DOWNSTAIRS	1
WALKING_UPSTAIRS	464

In [43]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 2s 765us/step
```

In [44]:
```
score
```

Out[44]:
```
[0.6057351109406184, 0.9138106549032915]
```

LSTM layer with droput rate = 0.3

In [51]:
```
# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 64
```

In [52]:
```
# 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(0.3))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

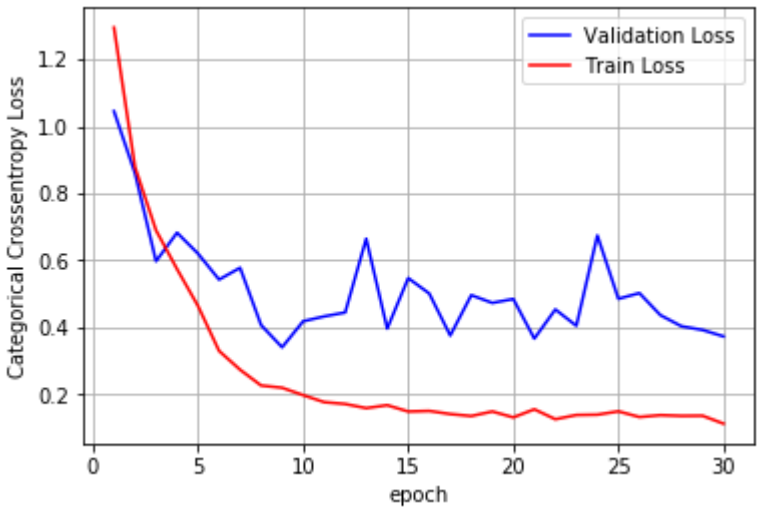
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_4 (LSTM)                (None, 64)                18944
_________________________________________________________________
dropout_4 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_4 (Dense)              (None, 6)                 390
=================================================================
Total params: 19,334
Trainable params: 19,334
Non-trainable params: 0
_________________________________________________________________
```

In [53]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [54]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Epoch 15/30
7352/7352 [==============================] - 54s 7ms/step - loss: 0.1481 - ac
c: 0.9430 - val_loss: 0.5465 - val_acc: 0.8599
Epoch 16/30
7352/7352 [==============================] - 56s 8ms/step - loss: 0.1498 - ac
c: 0.9457 - val_loss: 0.5006 - val_acc: 0.8931
Epoch 17/30
7352/7352 [==============================] - 56s 8ms/step - loss: 0.1405 - ac
c: 0.9450 - val_loss: 0.3751 - val_acc: 0.8938
Epoch 18/30
7352/7352 [==============================] - 57s 8ms/step - loss: 0.1346 - ac
c: 0.9487 - val_loss: 0.4956 - val_acc: 0.8955
Epoch 19/30
7352/7352 [==============================] - 55s 8ms/step - loss: 0.1482 - ac
c: 0.9434 - val_loss: 0.4725 - val_acc: 0.8751
Epoch 20/30
7352/7352 [==============================] - 57s 8ms/step - loss: 0.1306 - ac
c: 0.9498 - val_loss: 0.4841 - val_acc: 0.8965
Epoch 21/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.1547 - ac
```

In [45]:
```
epochs = 30
import matplotlib.pyplot as plt
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
x = list(range(1,epochs+1))
vy = val_loss #history.history['val_loss']
ty = train_loss #history.history['loss']
# plt_dynamic(x, vy, ty, ax)
ax.plot(x, vy, 'b', label="Validation Loss")
ax.plot(x, ty, 'r', label="Train Loss")
plt.legend()
plt.grid()
fig.canvas.draw()
```



In [55]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 536        0         0        0                   0
SITTING                  8      435        48        0                   0
STANDING                 0      149       383        0                   0
WALKING                  0        0         0      459                  24
WALKING_DOWNSTAIRS       0        0         0        0                 417
WALKING_UPSTAIRS         0        1         0       14                   9

Pred                WALKING_UPSTAIRS
True
LAYING                             1
SITTING                            0
STANDING                           0
WALKING                           13
WALKING_DOWNSTAIRS                 3
WALKING_UPSTAIRS                 447
```

In [56]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 3s 928us/step
```

In [57]:
```
score
```

Out[57]:
```
[0.3724719570703537, 0.9083814048184594]
```

2 LSTM layers with 32 units and dropouts (rate = 0.3)

In [58]:
```
# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 32
```

In [66]:
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden, return_sequences=True, input_shape=(timesteps, input_dim
# Adding a dropout layer
model.add(Dropout(0.3))
# Configuring the parameters
model.add(LSTM(64, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.3))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

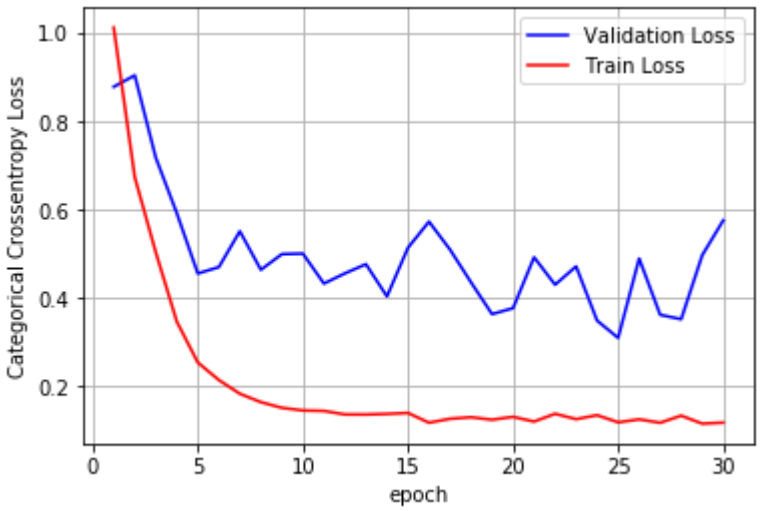
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_15 (LSTM)               (None, 128, 32)           5376
_________________________________________________________________
dropout_11 (Dropout)         (None, 128, 32)           0
_________________________________________________________________
lstm_16 (LSTM)               (None, 64)                24832
_________________________________________________________________
dropout_12 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_7 (Dense)              (None, 6)                 390
=================================================================
Total params: 30,598
Trainable params: 30,598
Non-trainable params: 0
_________________________________________________________________
```

In [67]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [68]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
Epoch 15/30
7352/7352 [==============================] - 94s 13ms/step - loss: 0.1402 - a
cc: 0.9494 - val_loss: 0.5143 - val_acc: 0.8951
Epoch 16/30
7352/7352 [==============================] - 91s 12ms/step - loss: 0.1181 - a
cc: 0.9506 - val_loss: 0.5735 - val_acc: 0.8945
Epoch 17/30
7352/7352 [==============================] - 93s 13ms/step - loss: 0.1271 - a
cc: 0.9524 - val_loss: 0.5102 - val_acc: 0.8982
Epoch 18/30
7352/7352 [==============================] - 92s 13ms/step - loss: 0.1305 - a
cc: 0.9505 - val_loss: 0.4360 - val_acc: 0.8955
Epoch 19/30
7352/7352 [==============================] - 98s 13ms/step - loss: 0.1249 - a
cc: 0.9495 - val_loss: 0.3640 - val_acc: 0.8992
Epoch 20/30
7352/7352 [==============================] - 106s 14ms/step - loss: 0.1314 -
acc: 0.9532 - val_loss: 0.3775 - val_acc: 0.8996
Epoch 21/30
7352/7352 [==============================] - 106s 14ms/step - loss: 0.1207 -
```

In [48]:
```
epochs = 30
import matplotlib.pyplot as plt
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
x = list(range(1,epochs+1))
vy = val_loss #history.history['val_loss']
ty = train_loss #history.history['loss']
# plt_dynamic(x, vy, ty, ax)
ax.plot(x, vy, 'b', label="Validation Loss")
ax.plot(x, ty, 'r', label="Train Loss")
plt.legend()
plt.grid()
fig.canvas.draw()
```



In [69]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                 509        1         0        0                   0
SITTING                  2      415        68        1                   4
STANDING                 0      113       419        0                   0
WALKING                  0        0         0      445                  48
WALKING_DOWNSTAIRS       0        0         0        0                 418
WALKING_UPSTAIRS         0        2         0        3                  22

Pred                WALKING_UPSTAIRS
True
LAYING                            27
SITTING                            1
STANDING                           0
WALKING                            3
WALKING_DOWNSTAIRS                 2
WALKING_UPSTAIRS                 444
```

In [70]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 4s 1ms/step
```

In [71]:
```
score
```

Out[71]:
```
[0.5762635302961742, 0.8992195453003053]
```

32 hidden units with 1 layer and dropout = 0.8 and optimizer = Adam

In [18]:
```
# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 32
```

In [21]:
```
# 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(0.8))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

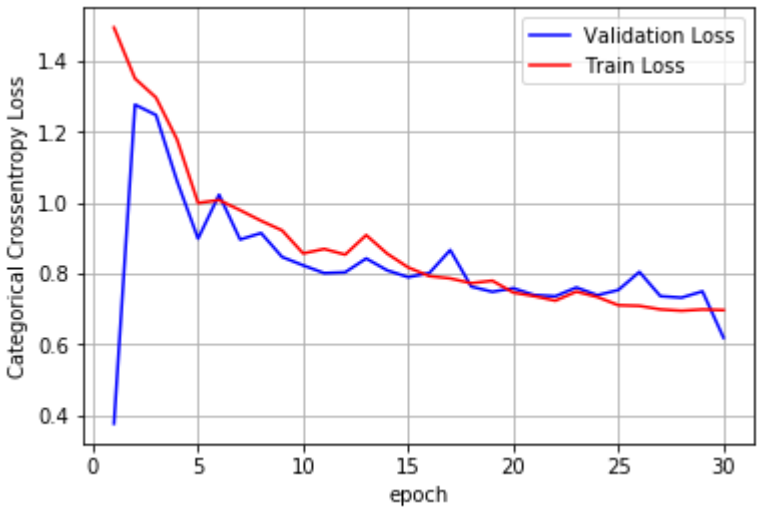
```
WARNING:tensorflow:From /home/j_choudhary1001/anaconda3/lib/python3.6/site-pack
ages/tensorflow/python/framework/op_def_library.py:263: colocate_with (from ten
sorflow.python.framework.ops) is deprecated and will be removed in a future ver
sion.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /home/j_choudhary1001/anaconda3/lib/python3.6/site-pack
ages/keras/backend/tensorflow_backend.py:3445: calling dropout (from tensorflo
w.python.ops.nn_ops) with keep_prob is deprecated and will be removed in a futu
re version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - kee
p_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 [22]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

In [23]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
c: 0.6122 - val_loss: 0.8087 - val_acc: 0.6057
Epoch 15/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.8169 - ac
c: 0.6073 - val_loss: 0.7894 - val_acc: 0.6064
Epoch 16/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7924 - ac
c: 0.6197 - val_loss: 0.8009 - val_acc: 0.5999
Epoch 17/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.7859 - ac
c: 0.6164 - val_loss: 0.8661 - val_acc: 0.5942
Epoch 18/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7723 - ac
c: 0.6254 - val_loss: 0.7627 - val_acc: 0.6125
Epoch 19/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7794 - ac
c: 0.6258 - val_loss: 0.7480 - val_acc: 0.6223
Epoch 20/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.7459 - ac
c: 0.6337 - val_loss: 0.7581 - val_acc: 0.6128
```

In [50]:
```
epochs = 30
import matplotlib.pyplot as plt
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
x = list(range(1,epochs+1))
vy = val_loss #history.history['val_loss']
ty = train_loss #history.history['loss']
# plt_dynamic(x, vy, ty, ax)
ax.plot(x, vy, 'b', label="Validation Loss")
ax.plot(x, ty, 'r', label="Train Loss")
plt.legend()
plt.grid()
fig.canvas.draw()
```



In [24]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

Pred	LAYING	SITTING	STANDING	WALKING
True				
LAYING	510	0	27	0
SITTING	0	394	95	2
STANDING	0	106	422	4
WALKING	0	0	0	496
WALKING_DOWNSTAIRS	0	0	1	419
WALKING_UPSTAIRS	0	1	2	468

In [25]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 1s 415us/step
```

In [26]:
```
score
```

Out[26]:
```
[0.7457214260519986, 0.6182558534102477]
```

2 layer with 32 units and dropout = 0.5

In [28]:
```
# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 32
```

In [32]:
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden, return_sequences = True, input_shape=(timesteps, input_d
# Adding a dropout layer
model.add(Dropout(0.5))
# Configuring the parameters
model.add(LSTM(64))
# 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()
```

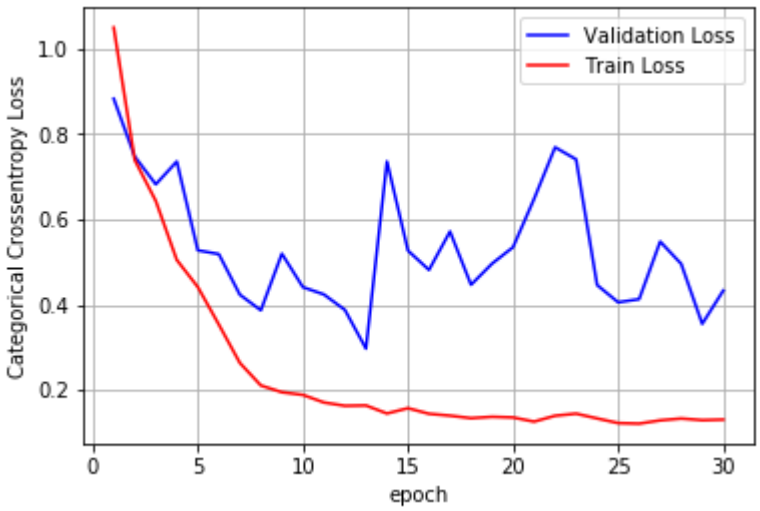
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 128, 32)           5376
_________________________________________________________________
dropout_2 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
lstm_3 (LSTM)                (None, 64)                24832
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 390
=================================================================
Total params: 30,598
Trainable params: 30,598
Non-trainable params: 0
_________________________________________________________________
```

In [33]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [34]:
```
# 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 [==============================] - 76s 10ms/step - loss: 1.0496 - a
cc: 0.5273 - val_loss: 0.8828 - val_acc: 0.5887
Epoch 2/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.7390 - a
cc: 0.6517 - val_loss: 0.7472 - val_acc: 0.6892
Epoch 3/30
7352/7352 [==============================] - 74s 10ms/step - loss: 0.6433 - a
cc: 0.7055 - val_loss: 0.6822 - val_acc: 0.7068
Epoch 4/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.5050 - a
cc: 0.7746 - val_loss: 0.7357 - val_acc: 0.7122
Epoch 5/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.4415 - a
cc: 0.8039 - val_loss: 0.5275 - val_acc: 0.7974
Epoch 6/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.3536 - a
cc: 0.8701 - val_loss: 0.5187 - val_acc: 0.8303
```

In [52]:
```
epochs = 30
import matplotlib.pyplot as plt
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
x = list(range(1,epochs+1))
vy = val_loss #history.history['val_loss']
ty = train_loss #history.history['loss']
# plt_dynamic(x, vy, ty, ax)
ax.plot(x, vy, 'b', label="Validation Loss")
ax.plot(x, ty, 'r', label="Train Loss")
plt.legend()
plt.grid()
fig.canvas.draw()
```



In [42]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

Pred	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	\
True						
LAYING	510	0	0	0	0	
SITTING	0	371	101	0	1	
STANDING	0	55	477	0	0	
WALKING	0	0	0	453	30	
WALKING_DOWNSTAIRS	0	0	0	1	418	
WALKING_UPSTAIRS	0	0	0	0	7	

Pred	WALKING_UPSTAIRS
True	
LAYING	27
SITTING	18
STANDING	0
WALKING	13
WALKING_DOWNSTAIRS	1
WALKING_UPSTAIRS	464

In [43]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 2s 765us/step
```

In [44]:
```
score
```

Out[44]:
```
[0.6057351109406184, 0.9138106549032915]
```

In [54]:
```
from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Layers", "Hidden Units", "Dropout", "Loss", "Accuracy"]

x.add_row(["1 layer", 32, 0.5, 0.45, 0.88])
x.add_row(["1 layer", 64, 0.5, 0.60, 0.9138])
x.add_row(["1 layer", 64, 0.3, 0.37, 0.9083])
x.add_row(["2 layer", 32, 0.3, 0.57, 0.8992])
x.add_row(["1 layer, optimizer = adam", 32, 0.8, 0.74, 0.6182])
x.add_row(["2 layer", 32, 0.5, 0.60, 0.9138])
print(x)
```

```
+---------------------------+--------------+---------+------+----------+
|           Layers          | Hidden Units | Dropout | Loss | Accuracy |
+---------------------------+--------------+---------+------+----------+
|          1 layer          |      32      |   0.5   | 0.45 |   0.88   |
|          1 layer          |      64      |   0.5   | 0.6  |  0.9138  |
|          1 layer          |      64      |   0.3   | 0.37 |  0.9083  |
|          2 layer          |      32      |   0.3   | 0.57 |  0.8992  |
| 1 layer, optimizer = adam |      32      |   0.8   | 0.74 |  0.6182  |
|          2 layer          |      32      |   0.5   | 0.6  |  0.9138  |
+---------------------------+--------------+---------+------+----------+
```

Steps followed

1. Loaded the train and test datasets
2. Defined the architecture of LSTM
3. Trained the network with different configurations