Human activity recognition using LSTM

In [ ]:

```
# Importing Libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
from time import time
from datetime import datetime
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.wrappers.scikit_learn import KerasClassifier
from keras.constraints import maxnorm
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers.core import Dense, Dropout
from keras.layers.normalization import BatchNormalization
```

In [0]:

```
# 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'])
```

1.0 Data

In [0]:

```
# 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'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'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.random.set_seed(42)
```

In [0]:

```
# Configuring a session
session_conf = tf.compat.v1.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]:

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

In [0]:

```
# Loading the train and test data
import warnings
warnings.filterwarnings("ignore")

X_train, X_test, Y_train, Y_test = load_data()
```

In [0]:

```
type(X_train)
```

Out[0]:

```
numpy.ndarray
```

In [0]:

```
print((X_train[0][0]))
```

```
[ 1.808515e-04  1.076681e-02  5.556068e-02  3.019122e-02  6.601362e-02
  2.285864e-02  1.012817e+00 -1.232167e-01  1.029341e-01]
```

In [0]:

```
print((X_train[0]))
```

```
[[ 1.808515e-04  1.076681e-02  5.556068e-02 ...  1.012817e+00
  -1.232167e-01  1.029341e-01]
 [ 1.013856e-02  6.579480e-03  5.512483e-02 ...  1.022833e+00
  -1.268756e-01  1.056872e-01]
 [ 9.275574e-03  8.928878e-03  4.840473e-02 ...  1.022028e+00
  -1.240037e-01  1.021025e-01]
 ...
 [-1.147484e-03  1.714439e-04  2.647864e-03 ...  1.018445e+00
  -1.240696e-01  1.003852e-01]
 [-2.222655e-04  1.574181e-03  2.381057e-03 ...  1.019372e+00
  -1.227451e-01  9.987355e-02]
 [ 1.575500e-03  3.070189e-03 -2.269757e-03 ...  1.021171e+00
  -1.213260e-01  9.498741e-02]]
```

In [0]:

```
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
```

In [0]:

```
print(n_classes)
```

```
6
```

In [0]:

```
np.save('X_train', X_train)
np.save('X_test', X_test)
np.save('Y_train', Y_train)
np.save('Y_test', Y_test)
```

In [3]:

```
from zipfile import ZipFile
file_name="/content/Colab.zip"

with ZipFile(file_name,'r') as zip:
    zip.extractall()
    print('Done')
```

```
Done
```

In [0]:

```
X_train= np.load('/content/Colab/X_train.npy')
X_test= np.load('/content/Colab/X_test.npy')
Y_train= np.load('/content/Colab/Y_train.npy')
Y_test= np.load('/content/Colab/Y_test.npy')
```

In [0]:

```
Y_test= np.load('/content/Colab/Y_test.npy')
```

2.0 Simple base model without hyperparameter tuning

In [0]:

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

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]
```

- With a simple 2 layer architecture we got 90.09% accuracy and a loss of 0.30
- We can further imporve the performace with Hyperparameter tuning

3.0 Hyperparameter tuning a single layered LSTM using KerasClassifier & Grid search

In [5]:

```
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
```

In [0]:

```
# Credits: https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning
-models-python-keras/
# Function to create model, required for KerasClassifier
def create_model(cells=1,dropout_rate=0.0):
    # create model
    model = Sequential()
    model.add(LSTM(cells, input_shape=(timesteps, input_dim)))
    model.add(Dropout(dropout_rate))
    model.add(Dense(n_classes, activation='sigmoid'))
    # Compile model
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accurac
y'])
    return model
```

In [0]:

```
model = KerasClassifier(build_fn=create_model, epochs=20, batch_size=50, verbose=0)
```

3.1 Grid Search

In [0]:

```
# defining the search parameters
import warnings
warnings.filterwarnings("ignore")
start = datetime.now()
cells=[64,128,150]
dropout_rate = [0.25, 0.35, 0.50]
param_grid = dict(cells= cells, dropout_rate=dropout_rate)
grid = GridSearchCV(estimator=model,param_grid=param_grid,cv=3)
grid_result = grid.fit(X_train, Y_train)
print('Time taken :', datetime.now() - start)
```

```
Time taken : 3:50:57.960481
```

3.2 Best estimator

In [0]:

```
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```

```
Best: 0.653972 using {'cells': 64, 'dropout_rate': 0.35}
```

In [0]:

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

```
0.607726 (0.021996) with: {'cells': 64, 'dropout_rate': 0.25}
0.653972 (0.015658) with: {'cells': 64, 'dropout_rate': 0.35}
0.517818 (0.150945) with: {'cells': 64, 'dropout_rate': 0.5}
0.520947 (0.088254) with: {'cells': 128, 'dropout_rate': 0.25}
0.560800 (0.086814) with: {'cells': 128, 'dropout_rate': 0.35}
0.432127 (0.293107) with: {'cells': 128, 'dropout_rate': 0.5}
0.632345 (0.114650) with: {'cells': 150, 'dropout_rate': 0.25}
0.574129 (0.046813) with: {'cells': 150, 'dropout_rate': 0.35}
0.542029 (0.108520) with: {'cells': 150, 'dropout_rate': 0.5}
```

3.3 3-D Plot to visualize the metric for different values of hyperparameters

In [0]:

```
df=pd.DataFrame(grid.cv_results_)
df.head(2)
```

Out[0]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_cells	paran
0	473.090310	5.601252	5.374086	0.078879	64	0.25
1	476.258571	1.768062	5.696839	0.041143	64	0.35

In [0]:

```
df.to_csv('hyp.csv')
```

In [0]:

```
%matplotlib notebook
%matplotlib inline
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np
def enable_plotly_in_cell():
    import IPython
    from plotly.offline import init_notebook_mode
    display(IPython.core.display.HTML('''<script src="/static/components/requirejs/requ
ire.js"></script>'''))
    init_notebook_mode(connected=False)
```

In [ ]:

```
# https://plot.ly/python/3d-axes/
#trace1 = go.Scatter3d(x=df['param_cells'],y=df['param_dropout_rate'],z=df['mean_test_s
core'], name = 'train')
trace2 = go.Scatter3d(x=df['param_cells'],y=df['param_dropout_rate'],z=df['mean_test_sc
ore'], name = 'Cross validation')
data = [trace2]
enable_plotly_in_cell()

layout = go.Layout(scene = dict(
        xaxis = dict(title='Number of LSTM cells'),
        yaxis = dict(title='Drop-out rate'),
        zaxis = dict(title='Accuracy'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

PLot

3.4 Applying the best hyperparameters on the network

In [0]:

```
n_hidden= 64
dropout_rate= 0.35
```

Architecture

In [10]:

```
# Initiliazing the sequential model
model1 = Sequential()

model1.add(LSTM(n_hidden,input_shape=(timesteps, input_dim)))
model1.add(BatchNormalization())
model1.add(Dropout(dropout_rate))

model1.add(Dense(n_classes, activation='sigmoid'))
model1.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_2 (LSTM)                (None, 64)                18944     
_________________________________________________________________
batch_normalization_2 (Batch (None, 64)                256       
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 390       
=================================================================
Total params: 19,590
Trainable params: 19,462
Non-trainable params: 128
_________________________________________________________________
```

In [0]:

```
# Compiling the model
model1.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

3.5 Checkpointing the model and creating the callback list

In [0]:

```
from keras.callbacks import ModelCheckpoint
from keras.callbacks import CSVLogger
import matplotlib.pyplot as plt
from keras.callbacks import TensorBoard
import tensorflow as tf
import datetime
import keras


filepath="weights-{epoch:02d}-{val_accuracy:.2f}.hdf5"
checkpoints = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')
train_results = CSVLogger('train_results_2.log') #storing the training results in a pandas dataframe
callbacks_list = [checkpoints, train_results]
```

3.6 Fitting the model in batches

In [17]:

```
history= model1.fit(X_train,Y_train,batch_size=50,validation_data=(X_test, Y_test),nb_e
poch=30,verbose=1,
                    callbacks =callbacks_list)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: UserWarnin
g: The `nb_epoch` argument in `fit` has been renamed `epochs`.


Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 44s 6ms/step - loss: 0.8737 -
acc: 0.5903 - val_loss: 0.8794 - val_acc: 0.5148
Epoch 2/30

/usr/local/lib/python3.6/dist-packages/keras/callbacks.py:707: RuntimeWarn
ing: Can save best model only with val_accuracy available, skipping.
  'skipping.' % (self.monitor), RuntimeWarning)
```

```
7352/7352 [==============================] - 44s 6ms/step - loss: 0.7757 -
acc: 0.6091 - val_loss: 0.8388 - val_acc: 0.6257
Epoch 3/30
7352/7352 [==============================] - 44s 6ms/step - loss: 0.7354 -
acc: 0.6138 - val_loss: 0.8674 - val_acc: 0.5589
Epoch 4/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.7585 -
acc: 0.5871 - val_loss: 0.7672 - val_acc: 0.5809
Epoch 5/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.7604 -
acc: 0.5632 - val_loss: 0.8021 - val_acc: 0.5107
Epoch 6/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.7076 -
acc: 0.5717 - val_loss: 0.7230 - val_acc: 0.5701
Epoch 7/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.7145 -
acc: 0.5690 - val_loss: 0.7387 - val_acc: 0.5304
Epoch 8/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.7102 -
acc: 0.5690 - val_loss: 0.7256 - val_acc: 0.5073
Epoch 9/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.6796 -
acc: 0.5846 - val_loss: 0.7081 - val_acc: 0.6091
Epoch 10/30
7352/7352 [==============================] - 44s 6ms/step - loss: 0.7015 -
acc: 0.6204 - val_loss: 0.6679 - val_acc: 0.6637
Epoch 11/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.5928 -
acc: 0.6959 - val_loss: 0.6539 - val_acc: 0.6610
Epoch 12/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.6295 -
acc: 0.7047 - val_loss: 1.9109 - val_acc: 0.4917
Epoch 13/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.6305 -
acc: 0.7311 - val_loss: 0.5935 - val_acc: 0.7448
Epoch 14/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.4553 -
acc: 0.8402 - val_loss: 0.4621 - val_acc: 0.8626
Epoch 15/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.2337 -
acc: 0.9241 - val_loss: 0.6692 - val_acc: 0.8544
Epoch 16/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1950 -
acc: 0.9300 - val_loss: 0.4736 - val_acc: 0.8758
Epoch 17/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.2293 -
acc: 0.9221 - val_loss: 0.5560 - val_acc: 0.8690
Epoch 18/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.2485 -
acc: 0.9144 - val_loss: 0.3738 - val_acc: 0.8907
Epoch 19/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1847 -
acc: 0.9309 - val_loss: 0.2657 - val_acc: 0.9053
Epoch 20/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1825 -
acc: 0.9338 - val_loss: 0.2923 - val_acc: 0.9128
Epoch 21/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1516 -
acc: 0.9411 - val_loss: 0.2962 - val_acc: 0.9111
Epoch 22/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1465 -
```

```
acc: 0.9436 - val_loss: 0.2487 - val_acc: 0.9074
Epoch 23/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1433 -
acc: 0.9396 - val_loss: 0.3190 - val_acc: 0.9094
Epoch 24/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1792 -
acc: 0.9310 - val_loss: 0.2996 - val_acc: 0.9121
Epoch 25/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1683 -
acc: 0.9353 - val_loss: 0.3410 - val_acc: 0.8819
Epoch 26/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1682 -
acc: 0.9377 - val_loss: 0.2552 - val_acc: 0.9019
Epoch 27/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1430 -
acc: 0.9402 - val_loss: 0.2351 - val_acc: 0.9141
Epoch 28/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1309 -
acc: 0.9444 - val_loss: 0.2480 - val_acc: 0.9040
Epoch 29/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1266 -
acc: 0.9463 - val_loss: 0.2544 - val_acc: 0.9070
Epoch 30/30
7352/7352 [==============================] - 43s 6ms/step - loss: 0.1200 -
acc: 0.9517 - val_loss: 0.2620 - val_acc: 0.9128
```

3.7 Confusion matrix

In [102]:

```
cm=confusion_matrix(Y_test, model1.predict(X_test))
cm
```

Out[102]:

Pred	LAYING	SITTING	STANDING	WALKING	WALKING_DOWN
True					
LAYING	537	0	0	0	0
SITTING	0	371	116	1	0
STANDING	0	77	452	2	0
WALKING	0	8	0	467	15
WALKING_DOWNSTAIRS	0	0	0	4	412
WALKING_UPSTAIRS	0	0	0	13	7

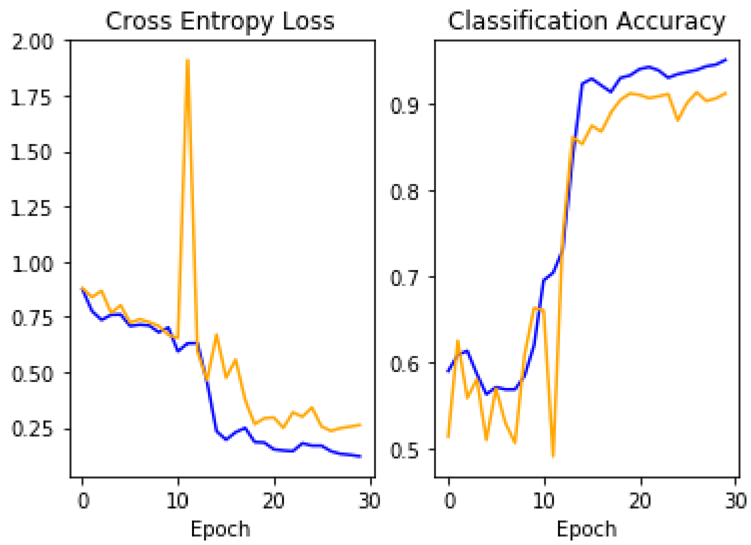
3.8 Plots on training results

In [0]:

```
# function to plot epoch vs loss
%matplotlib notebook
%matplotlib inline
from matplotlib import pyplot
def plot(history):
    # plot loss
    pyplot.subplot(121)
    pyplot.title('Cross Entropy Loss')
    pyplot.xlabel('Epoch')
    pyplot.plot(history.history['loss'], color='blue', label='train')
    pyplot.plot(history.history['val_loss'], color='orange', label='test')
    # plot accuracy
    pyplot.subplot(122)
    pyplot.title('\nClassification Accuracy')
    pyplot.xlabel('Epoch')
    pyplot.plot(history.history['acc'], color='blue', label='train')
    pyplot.plot(history.history['val_acc'], color='orange', label='test')
```

In [41]:

```
plot(history)
```



3.9 Model Testing

In [20]:

```
score = model1.evaluate(X_test, Y_test, verbose=1)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
2947/2947 [==============================] - 10s 3ms/step
Test loss: 0.2620189085359711
Test accuracy: 0.9127926705123854
```

4.0 Deep LSTM model

In [0]:

```
epochs = 50
batch_size= 50
n_hidden1 = 64
n_hidden2 =128
d1 = 0.50
d2 = 0.60 #using higher dropout rates
```

In [0]:

```
import keras.backend as K
K.clear_session()
```

4.1 Architecture

In [126]:

```
# Initiliazing the sequential model
model2 = Sequential()

model2.add(LSTM(n_hidden1,return_sequences=True,input_shape=(timesteps, input_dim)))
model2.add(BatchNormalization())
model2.add(Dropout(d1))

model2.add(LSTM(n_hidden2))
model2.add(BatchNormalization())
model2.add(Dropout(d2))

model2.add(Dense(n_classes, activation='sigmoid'))
model2.summary()
```

```
WARNING:tensorflow:Large dropout rate: 0.6 (>0.5). In TensorFlow 2.x, drop
out() uses dropout rate instead of keep_prob. Please ensure that this is i
ntended.
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_1 (LSTM)                (None, 128, 64)           18944     
_________________________________________________________________
batch_normalization_1 (Batch (None, 128, 64)           256       
_________________________________________________________________
dropout_1 (Dropout)          (None, 128, 64)           0         
_________________________________________________________________
lstm_2 (LSTM)                (None, 128)               98816     
_________________________________________________________________
batch_normalization_2 (Batch (None, 128)               512       
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 774       
=================================================================
Total params: 119,302
Trainable params: 118,918
Non-trainable params: 384
_________________________________________________________________
```

4.2 Compiling

In [0]:

```
# Compiling the model
model2.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

4.3 Checkpointing the model and creating the callback list

In [0]:

```
from keras.callbacks import ModelCheckpoint
from keras.callbacks import CSVLogger
import matplotlib.pyplot as plt
from keras.callbacks import TensorBoard
import tensorflow as tf
import datetime
import keras


filepath='model-ep{epoch:03d}-val_acc{val_acc:.3f}.h5'
checkpoints = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1, save_best_on
ly=True, mode='max')
train_results = CSVLogger('train_results_model2.log') #storing the training results in
 a pandas dataframe
callbacks_list = [checkpoints, train_results]
```

4.4 Fitting the model in batches

In [129]:

```
# Fitting the model
history1= model2.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test, Y_t
est),epochs=epochs)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/50
7352/7352 [==============================] - 88s 12ms/step - loss: 1.0211
- acc: 0.6208 - val_loss: 0.8160 - val_acc: 0.6953
Epoch 2/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.7524
- acc: 0.6862 - val_loss: 0.7849 - val_acc: 0.6661
Epoch 3/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.7182
- acc: 0.6865 - val_loss: 0.7363 - val_acc: 0.7316
Epoch 4/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.6304
- acc: 0.7300 - val_loss: 0.9483 - val_acc: 0.6956
Epoch 5/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.5353
- acc: 0.8070 - val_loss: 0.5818 - val_acc: 0.8368
Epoch 6/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.3677
- acc: 0.8641 - val_loss: 0.4695 - val_acc: 0.8341
Epoch 7/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.2595
- acc: 0.8483 - val_loss: 0.4434 - val_acc: 0.7842
Epoch 8/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.2833
- acc: 0.8303 - val_loss: 0.3670 - val_acc: 0.7920
Epoch 9/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.2319
- acc: 0.8347 - val_loss: 0.4086 - val_acc: 0.7764
Epoch 10/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.2277
- acc: 0.8312 - val_loss: 0.3435 - val_acc: 0.8035
Epoch 11/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.2197
- acc: 0.8402 - val_loss: 0.3575 - val_acc: 0.7978
Epoch 12/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.2174
- acc: 0.8860 - val_loss: 0.3930 - val_acc: 0.9114
Epoch 13/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.1803
- acc: 0.9338 - val_loss: 0.4490 - val_acc: 0.8894
Epoch 14/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.1561
- acc: 0.9410 - val_loss: 0.4746 - val_acc: 0.8548
Epoch 15/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.1666
- acc: 0.9391 - val_loss: 0.2934 - val_acc: 0.9104
Epoch 16/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.1702
- acc: 0.9355 - val_loss: 0.3931 - val_acc: 0.8873
Epoch 17/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.1906
- acc: 0.9290 - val_loss: 0.3184 - val_acc: 0.9077
Epoch 18/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.1631
- acc: 0.9361 - val_loss: 0.2773 - val_acc: 0.9135
Epoch 19/50
7352/7352 [==============================] - 86s 12ms/step - loss: 0.1359
- acc: 0.9407 - val_loss: 0.3139 - val_acc: 0.9060
Epoch 20/50
7352/7352 [==============================] - 86s 12ms/step - loss: 0.1375
- acc: 0.9479 - val_loss: 0.3403 - val_acc: 0.9097
```

```
Epoch 21/50
7352/7352 [==============================] - 86s 12ms/step - loss: 0.1440
- acc: 0.9430 - val_loss: 0.3217 - val_acc: 0.9148
Epoch 22/50
7352/7352 [==============================] - 85s 12ms/step - loss: 0.1313
- acc: 0.9484 - val_loss: 0.3400 - val_acc: 0.9097
Epoch 23/50
7352/7352 [==============================] - 85s 12ms/step - loss: 0.1913
- acc: 0.9340 - val_loss: 0.2570 - val_acc: 0.9186
Epoch 24/50
7352/7352 [==============================] - 86s 12ms/step - loss: 0.1379
- acc: 0.9412 - val_loss: 0.2645 - val_acc: 0.9281
Epoch 25/50
7352/7352 [==============================] - 85s 12ms/step - loss: 0.1649
- acc: 0.9415 - val_loss: 0.2581 - val_acc: 0.9046
Epoch 26/50
7352/7352 [==============================] - 85s 12ms/step - loss: 0.1326
- acc: 0.9478 - val_loss: 0.2355 - val_acc: 0.9355
Epoch 27/50
7352/7352 [==============================] - 86s 12ms/step - loss: 0.1320
- acc: 0.9490 - val_loss: 0.2499 - val_acc: 0.9253
Epoch 28/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.1220
- acc: 0.9489 - val_loss: 0.2754 - val_acc: 0.9257
Epoch 29/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.1227
- acc: 0.9486 - val_loss: 0.2694 - val_acc: 0.9209
Epoch 30/50
7352/7352 [==============================] - 88s 12ms/step - loss: 0.1287
- acc: 0.9463 - val_loss: 0.2407 - val_acc: 0.9281
Epoch 31/50
7352/7352 [==============================] - 88s 12ms/step - loss: 0.1671
- acc: 0.9306 - val_loss: 0.2330 - val_acc: 0.9175
Epoch 32/50
7352/7352 [==============================] - 88s 12ms/step - loss: 0.1439
- acc: 0.9369 - val_loss: 0.3069 - val_acc: 0.9074
Epoch 33/50
7352/7352 [==============================] - 88s 12ms/step - loss: 0.1429
- acc: 0.9392 - val_loss: 0.3173 - val_acc: 0.9104
Epoch 34/50
7352/7352 [==============================] - 88s 12ms/step - loss: 0.1222
- acc: 0.9506 - val_loss: 0.2809 - val_acc: 0.9318
Epoch 35/50
7352/7352 [==============================] - 88s 12ms/step - loss: 0.1227
- acc: 0.9521 - val_loss: 0.2797 - val_acc: 0.9233
Epoch 36/50
7352/7352 [==============================] - 88s 12ms/step - loss: 0.1186
- acc: 0.9504 - val_loss: 0.3137 - val_acc: 0.9226
Epoch 37/50
7352/7352 [==============================] - 88s 12ms/step - loss: 0.1596
- acc: 0.9354 - val_loss: 0.3006 - val_acc: 0.9128
Epoch 38/50
7352/7352 [==============================] - 88s 12ms/step - loss: 0.1533
- acc: 0.9351 - val_loss: 0.3289 - val_acc: 0.8965
Epoch 39/50
7352/7352 [==============================] - 88s 12ms/step - loss: 0.1540
- acc: 0.9370 - val_loss: 0.2790 - val_acc: 0.9243
Epoch 40/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.1287
- acc: 0.9464 - val_loss: 0.2605 - val_acc: 0.9284
Epoch 41/50
```

```
7352/7352 [==============================] - 88s 12ms/step - loss: 0.1227
- acc: 0.9479 - val_loss: 0.2856 - val_acc: 0.9260
Epoch 42/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.1214
- acc: 0.9467 - val_loss: 0.3178 - val_acc: 0.9274
Epoch 43/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.1218
- acc: 0.9493 - val_loss: 0.3100 - val_acc: 0.9270
Epoch 44/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.1222
- acc: 0.9497 - val_loss: 0.3382 - val_acc: 0.9182
Epoch 45/50
7352/7352 [==============================] - 89s 12ms/step - loss: 0.1255
- acc: 0.9509 - val_loss: 0.3199 - val_acc: 0.9230
Epoch 46/50
7352/7352 [==============================] - 89s 12ms/step - loss: 0.1120
- acc: 0.9532 - val_loss: 0.3275 - val_acc: 0.9213
Epoch 47/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.1225
- acc: 0.9487 - val_loss: 0.3052 - val_acc: 0.9247
Epoch 48/50
7352/7352 [==============================] - 88s 12ms/step - loss: 0.1304
- acc: 0.9421 - val_loss: 0.3078 - val_acc: 0.9165
Epoch 49/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.1237
- acc: 0.9484 - val_loss: 0.3364 - val_acc: 0.9186
Epoch 50/50
7352/7352 [==============================] - 87s 12ms/step - loss: 0.1196
- acc: 0.9524 - val_loss: 0.3126 - val_acc: 0.9308
```

4.5 Confusion matrix

In [132]:

```
cm1= confusion_matrix(Y_test, model2.predict(X_test))
cm1
```

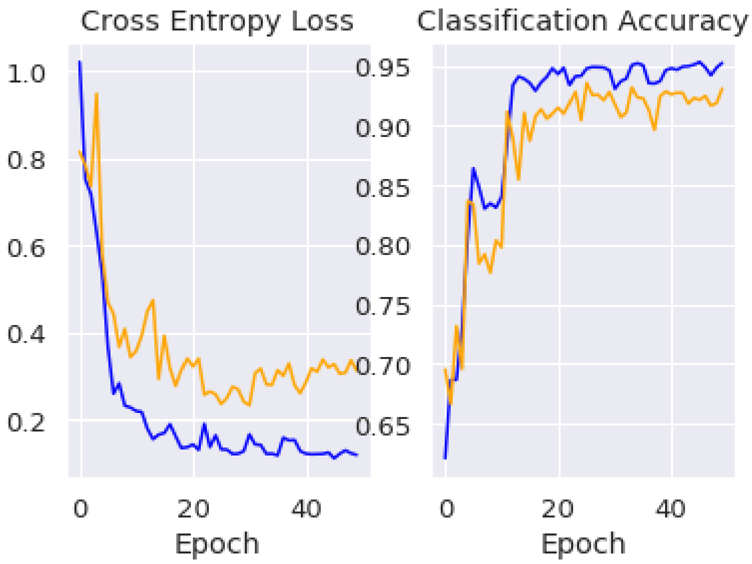
Out[132]:

Pred	LAYING	SITTING	STANDING	WALKING	WALKING_DOWN
True					
LAYING	537	0	0	0	0
SITTING	0	370	118	0	0
STANDING	0	50	482	0	0
WALKING	0	0	0	466	27
WALKING_DOWNSTAIRS	0	0	0	1	418
WALKING_UPSTAIRS	0	0	0	1	0

4.6 Plots on training results

In [134]:

```
plot(history1)
```



4.7 Model Testing

In [135]:

```
score = model2.evaluate(X_test, Y_test, verbose=1)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
2947/2947 [==============================] - 19s 6ms/step
Test loss: 0.312556509635759 87
Test accuracy: 0.9307770614183916
```

5.0 Summary

In [3]:

```
#Ref: http://zetcode.com/python/prettytable/

from prettytable import PrettyTable
x=PrettyTable()
x.field_names=["Model","Test loss","Test accuracy"]

x.add_row(["1 layered LSTM without hyp tuning","0.3088","90.97%"])
x.add_row(["1 layered LSTM with hyp tuning","0.2620","91.30%"])
x.add_row(["Deep 2 layered LSTM","0.3126","93.08%"])

print(x)
```

```
+-----------------------------------+-----------+---------------+
|               Model               | Test loss | Test accuracy |
+-----------------------------------+-----------+---------------+
| 1 layered LSTM without hyp tuning |   0.3088  |     90.97%    |
|   1 layered LSTM with hyp tuning  |   0.2620  |     91.30%    |
|        Deep 2 layered LSTM        |   0.3126  |     93.08%    |
+-----------------------------------+-----------+---------------+
```