In [0]:
```
# Importing Libraries
```

In [1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_i
d=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redi
rect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20h
ttps%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleap
is.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.
readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly (http
s://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pf
ee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%
3a2.0%3aoob&response_type=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2
fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%
2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2fwww.goog
leapis.com%2fauth%2fpeopleapi.readonly)

Enter your authorization code:
··········
Mounted at /content/drive
```

In [2]:
```
!pip3 install patool
import patoolib
patoolib.extract_archive('/content/drive/My Drive/HumanActivityRecognition.zip')
```

```
Collecting patool
  Downloading https://files.pythonhosted.org/packages/43/94/52243ddff508780dd2d
8110964320ab4851134a55ab102285b46e740f76a/patool-1.12-py2.py3-none-any.whl (htt
ps://files.pythonhosted.org/packages/43/94/52243ddff508780dd2d8110964320ab48511
34a55ab102285b46e740f76a/patool-1.12-py2.py3-none-any.whl) (77kB)
     |████████████████████████████████| 81kB 2.5MB/s eta 0:00:011
Installing collected packages: patool
Successfully installed patool-1.12
patool: Extracting /content/drive/My Drive/HumanActivityRecognition.zip ...
patool: running /usr/bin/7z x -o./Unpack_wny74lr4 -- "/content/drive/My Drive/H
umanActivityRecognition.zip"
patool: ... /content/drive/My Drive/HumanActivityRecognition.zip extracted to `
HumanActivityRecognition' (multiple files in root).
```

Out[2]:
```
'HumanActivityRecognition'
```

In [0]:
```
import pandas as pd
import numpy as np
```

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'])
```

## Data

In [0]:
```
# Data directory
DATADIR = 'UCI_HAR_Dataset'
```

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'/content/HumanActivityRecognition/HAR/UCI_HAR_Dataset/{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 [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.ht
    """

    filename = f'/content/HumanActivityRecognition/HAR/UCI_HAR_Dataset/{subset}/y
    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 = 28
batch_size = 16
n_hidden1 = 31
n_hidden2 = 34
```

In [0]:
```
# Utility function to count the number of classes
def _count_classes(y):
    return len(set([tuple(category) for category in y]))
```

In [120]:
```
# 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:13: FutureWarning:
Method .as_matrix will be removed in a future version. Use .values instead.
  del sys.path[0]
```

In [121]:
```
timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = _count_classes(Y_train)

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

```
128
9
7352
```

- Defining the Architecture of LSTM

In [122]:

```
# Initiliazing the sequential model
model = Sequential()
from keras.callbacks import EarlyStopping
from keras.callbacks import ModelCheckpoint
from keras.models import load_model

# Configuring the parameters
model.add(LSTM(n_hidden1, input_shape=(timesteps, input_dim),return_sequences = 
# Adding a dropout layer
model.add(Dropout(0.4))

model.add(LSTM(n_hidden2, input_shape=(timesteps, input_dim)))
# # Adding a dropout layer
model.add(Dropout(0.7))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_27 (LSTM)               (None, 128, 31)           5084      
_________________________________________________________________
dropout_27 (Dropout)         (None, 128, 31)           0         
_________________________________________________________________
lstm_28 (LSTM)               (None, 34)                8976      
_________________________________________________________________
dropout_28 (Dropout)         (None, 34)                0         
_________________________________________________________________
dense_14 (Dense)             (None, 6)                 210       
=================================================================
Total params: 14,270
Trainable params: 14,270
Non-trainable params: 0
_________________________________________________________________
```

In [0]:

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

In [124]:
```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs,)


# evaluate the model
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/28
7352/7352 [==============================] - 80s 11ms/step - loss: 1.3133 - ac
c: 0.4544 - val_loss: 1.0180 - val_acc: 0.5779
Epoch 2/28
7352/7352 [==============================] - 72s 10ms/step - loss: 0.9327 - ac
c: 0.5975 - val_loss: 0.8259 - val_acc: 0.6271
Epoch 3/28
7352/7352 [==============================] - 73s 10ms/step - loss: 0.8002 - ac
c: 0.6285 - val_loss: 0.8054 - val_acc: 0.6227
Epoch 4/28
7352/7352 [==============================] - 74s 10ms/step - loss: 0.7564 - ac
c: 0.6409 - val_loss: 0.7818 - val_acc: 0.6210
Epoch 5/28
7352/7352 [==============================] - 73s 10ms/step - loss: 0.7475 - ac
c: 0.6457 - val_loss: 0.7894 - val_acc: 0.6223
Epoch 6/28
7352/7352 [==============================] - 74s 10ms/step - loss: 0.7315 - ac
c: 0.6499 - val_loss: 0.9027 - val_acc: 0.6054
Epoch 7/28
7352/7352 [==============================] - 73s 10ms/step - loss: 0.6993 - ac
c: 0.6568 - val_loss: 0.7988 - val_acc: 0.6200
Epoch 8/28
7352/7352 [==============================] - 73s 10ms/step - loss: 0.6968 - ac
c: 0.6536 - val_loss: 0.8979 - val_acc: 0.6125
Epoch 9/28
7352/7352 [==============================] - 73s 10ms/step - loss: 0.6515 - ac
c: 0.6643 - val_loss: 0.7197 - val_acc: 0.6210
Epoch 10/28
7352/7352 [==============================] - 73s 10ms/step - loss: 0.6020 - ac
c: 0.6712 - val_loss: 0.7570 - val_acc: 0.6152
Epoch 11/28
7352/7352 [==============================] - 73s 10ms/step - loss: 0.5975 - ac
c: 0.6808 - val_loss: 0.6153 - val_acc: 0.6284
Epoch 12/28
7352/7352 [==============================] - 72s 10ms/step - loss: 0.5582 - ac
c: 0.7157 - val_loss: 0.6488 - val_acc: 0.6196
Epoch 13/28
7352/7352 [==============================] - 74s 10ms/step - loss: 0.5150 - ac
c: 0.7946 - val_loss: 0.6463 - val_acc: 0.8782
Epoch 14/28
7352/7352 [==============================] - 74s 10ms/step - loss: 0.4822 - ac
c: 0.8357 - val_loss: 0.5451 - val_acc: 0.8911
Epoch 15/28
7352/7352 [==============================] - 72s 10ms/step - loss: 0.4076 - ac
c: 0.8828 - val_loss: 0.4545 - val_acc: 0.8850
```

```
Epoch 16/28
7352/7352 [==============================] - 72s 10ms/step - loss: 0.3360 - ac
c: 0.9063 - val_loss: 0.5114 - val_acc: 0.8578
Epoch 17/28
7352/7352 [==============================] - 74s 10ms/step - loss: 0.3161 - ac
c: 0.9053 - val_loss: 0.5672 - val_acc: 0.8775
Epoch 18/28
7352/7352 [==============================] - 75s 10ms/step - loss: 0.2639 - ac
c: 0.9241 - val_loss: 0.5785 - val_acc: 0.8975
Epoch 19/28
7352/7352 [==============================] - 72s 10ms/step - loss: 0.2611 - ac
c: 0.9274 - val_loss: 0.8392 - val_acc: 0.8649
Epoch 20/28
7352/7352 [==============================] - 72s 10ms/step - loss: 0.2579 - ac
c: 0.9234 - val_loss: 0.5132 - val_acc: 0.8839
Epoch 21/28
7352/7352 [==============================] - 72s 10ms/step - loss: 0.2531 - ac
c: 0.9297 - val_loss: 0.5169 - val_acc: 0.8924
Epoch 22/28
7352/7352 [==============================] - 72s 10ms/step - loss: 0.2867 - ac
c: 0.9275 - val_loss: 0.6168 - val_acc: 0.8887
Epoch 23/28
7352/7352 [==============================] - 72s 10ms/step - loss: 0.2268 - ac
c: 0.9334 - val_loss: 0.7760 - val_acc: 0.8697
Epoch 24/28
7352/7352 [==============================] - 72s 10ms/step - loss: 0.2254 - ac
c: 0.9331 - val_loss: 0.8598 - val_acc: 0.8697
Epoch 25/28
7352/7352 [==============================] - 72s 10ms/step - loss: 0.2294 - ac
c: 0.9368 - val_loss: 0.7258 - val_acc: 0.8843
Epoch 26/28
7352/7352 [==============================] - 74s 10ms/step - loss: 0.1978 - ac
c: 0.9329 - val_loss: 0.6014 - val_acc: 0.8887
Epoch 27/28
7352/7352 [==============================] - 72s 10ms/step - loss: 0.2246 - ac
c: 0.9316 - val_loss: 0.7234 - val_acc: 0.8985
Epoch 28/28
7352/7352 [==============================] - 72s 10ms/step - loss: 0.2124 - ac
c: 0.9336 - val_loss: 0.5799 - val_acc: 0.9043
```

Out[124]:
```
<keras.callbacks.History at 0x7f976d5e9518>
```

In [125]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  ...  WALKING_DOWNSTAIRS  WALKING_UPSTAIRS
True                                 ...
LAYING                 510        0  ...                   0                 0
SITTING                  1      382  ...                   5                 3
STANDING                 0       68  ...                   0                 0
WALKING                  0        0  ...                   8                29
WALKING_DOWNSTAIRS       0        0  ...                 408                11
WALKING_UPSTAIRS         0        0  ...                   8               448

[6 rows x 6 columns]
```

In [126]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 2s 836us/step
```

In [127]:
```
score
```

Out[127]:
```
[0.5798575633825492, 0.9043094672548354]
```

### Summary

- With a two layer architecture we got 90.4% accuracy and a loss of 0.57