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

In [24]:
```
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 [25]:
```
!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 1.4MB/s
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_awsrkuu_ -- "/content/drive/My Drive/H
umanActivityRecognition.zip"
patool: ... /content/drive/My Drive/HumanActivityRecognition.zip extracted to `
HumanActivityRecognition' (multiple files in root).
```

Out[25]:
```
'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.h
    """

    filename = f'/content/HumanActivityRecognition/HAR/UCI_HAR_Dataset/{subset}/
    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 = 32
batch_size = 16
n_hidden1 = 66
# n_hidden2 = 66
```

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

In [46]:
```
# 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 [47]:
```
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 [48]:
```
# 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_sequenc
# Adding a dropout layer
model.add(Dropout(0.2))

# 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_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_2 (LSTM)                (None, 66)                20064     
_________________________________________________________________
dropout_2 (Dropout)          (None, 66)                0         
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 402       
=================================================================
Total params: 20,466
Trainable params: 20,466
Non-trainable params: 0
_________________________________________________________________
```

In [0]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [50]:
```
# 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 1/32
7352/7352 [==============================] - 42s 6ms/step - loss: 1.1468 - acc:
0.4974 - val_loss: 0.9718 - val_acc: 0.5779
Epoch 2/32
7352/7352 [==============================] - 40s 5ms/step - loss: 0.8351 - acc:
0.6129 - val_loss: 0.8328 - val_acc: 0.6074
Epoch 3/32
7352/7352 [==============================] - 40s 5ms/step - loss: 0.6960 - acc:
0.6748 - val_loss: 0.7534 - val_acc: 0.7085
Epoch 4/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.5724 - acc:
0.7871 - val_loss: 0.5295 - val_acc: 0.8127
Epoch 5/32
7352/7352 [==============================] - 40s 5ms/step - loss: 0.3648 - acc:
0.8799 - val_loss: 0.5936 - val_acc: 0.8303
Epoch 6/32
7352/7352 [==============================] - 40s 5ms/step - loss: 0.2763 - acc:
0.9052 - val_loss: 0.4078 - val_acc: 0.8493
Epoch 7/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.2198 - acc:
0.9252 - val_loss: 0.3227 - val_acc: 0.8931
Epoch 8/32
7352/7352 [==============================] - 40s 5ms/step - loss: 0.1823 - acc:
0.9374 - val_loss: 0.2567 - val_acc: 0.9138
Epoch 9/32
7352/7352 [==============================] - 40s 5ms/step - loss: 0.1705 - acc:
0.9388 - val_loss: 0.2990 - val_acc: 0.9046
Epoch 10/32
7352/7352 [==============================] - 40s 5ms/step - loss: 0.1581 - acc:
0.9403 - val_loss: 0.2954 - val_acc: 0.9016
Epoch 11/32
7352/7352 [==============================] - 40s 5ms/step - loss: 0.1477 - acc:
0.9484 - val_loss: 0.3717 - val_acc: 0.9057
Epoch 12/32
7352/7352 [==============================] - 40s 5ms/step - loss: 0.1602 - acc:
0.9391 - val_loss: 0.3181 - val_acc: 0.8972
Epoch 13/32
7352/7352 [==============================] - 40s 5ms/step - loss: 0.1407 - acc:
0.9502 - val_loss: 0.2872 - val_acc: 0.8992
Epoch 14/32
7352/7352 [==============================] - 40s 5ms/step - loss: 0.1376 - acc:
0.9478 - val_loss: 0.3902 - val_acc: 0.8999
Epoch 15/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1338 - acc:
0.9491 - val_loss: 0.6023 - val_acc: 0.8789
```

```
Epoch 16/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1339 - acc:
0.9501 - val_loss: 0.4209 - val_acc: 0.8979
Epoch 17/32
7352/7352 [==============================] - 40s 6ms/step - loss: 0.1693 - acc:
0.9384 - val_loss: 0.3419 - val_acc: 0.8989
Epoch 18/32
7352/7352 [==============================] - 40s 5ms/step - loss: 0.1356 - acc:
0.9513 - val_loss: 0.5303 - val_acc: 0.8850
Epoch 19/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1378 - acc:
0.9498 - val_loss: 0.4059 - val_acc: 0.8968
Epoch 20/32
7352/7352 [==============================] - 40s 5ms/step - loss: 0.1368 - acc:
0.9512 - val_loss: 0.4248 - val_acc: 0.9023
Epoch 21/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1331 - acc:
0.9486 - val_loss: 0.4898 - val_acc: 0.8829
Epoch 22/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1323 - acc:
0.9516 - val_loss: 0.5119 - val_acc: 0.8951
Epoch 23/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1397 - acc:
0.9501 - val_loss: 0.3859 - val_acc: 0.9019
Epoch 24/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1327 - acc:
0.9501 - val_loss: 0.3200 - val_acc: 0.9135
Epoch 25/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1290 - acc:
0.9498 - val_loss: 0.2496 - val_acc: 0.9294
Epoch 26/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1279 - acc:
0.9494 - val_loss: 0.3840 - val_acc: 0.9097
Epoch 27/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1546 - acc:
0.9456 - val_loss: 0.3094 - val_acc: 0.9057
Epoch 28/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1279 - acc:
0.9521 - val_loss: 0.3303 - val_acc: 0.9057
Epoch 29/32
7352/7352 [==============================] - 40s 6ms/step - loss: 0.1188 - acc:
0.9520 - val_loss: 0.3774 - val_acc: 0.9101
Epoch 30/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1120 - acc:
0.9553 - val_loss: 0.3410 - val_acc: 0.9172
Epoch 31/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1245 - acc:
0.9521 - val_loss: 0.2935 - val_acc: 0.9070
Epoch 32/32
7352/7352 [==============================] - 40s 5ms/step - loss: 0.1219 - acc:
0.9521 - val_loss: 0.3748 - val_acc: 0.9162
```

Out[50]:
```
<keras.callbacks.History at 0x7fc0cb4d3a58>
```

In [52]:
```
# 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                  0      415  ...                   2                 0
STANDING                 0      100  ...                   0                 0
WALKING                  0        1  ...                  17                 8
WALKING_DOWNSTAIRS       0        0  ...                 419                 0
WALKING_UPSTAIRS         0        0  ...                   0               454

[6 rows x 6 columns]
```

In [53]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 2s 560us/step
```

In [54]:
```
score
```

Out[54]:
```
[0.37476612998923525, 0.9161859518154055]
```

Summary

- With a simple single layer architecture and a dropout of 0.2 we got 92.94% accuracy and a loss of 0.24 from the best model.

In [ ]: