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_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly (https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.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/43/94/52243ddff508780dd2d8110964320ab4851134a55ab102285b46e740f76a/patool-1.12-py2.py3-none-any.whl (https://files.pythonhosted.org/packages/43/94/52243ddff508780dd2d8110964320ab4851134a55ab102285b46e740f76a/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/HumanActivityRecognition.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/{subse
        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 = 32
batch_size = 16
n_hidden1 = 66
# n_hidden2 = 33
```

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

In [72]:
```
# 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 [73]:
```
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 [74]:

```
# 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(0.6))

# 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()
```

```
WARNING:tensorflow:Large dropout rate: 0.6 (>0.5). In TensorFlow 2.x, dropout()
uses dropout rate instead of keep_prob. Please ensure that this is intended.
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_5 (LSTM)                (None, 66)                20064
_________________________________________________________________
dropout_5 (Dropout)          (None, 66)                0
_________________________________________________________________
dense_5 (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 [76]:
```
# 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 [==============================] - 43s 6ms/step - loss: 1.2165 - acc:
0.4804 - val_loss: 1.0231 - val_acc: 0.5351
Epoch 2/32
7352/7352 [==============================] - 42s 6ms/step - loss: 0.9784 - acc:
0.5722 - val_loss: 1.1168 - val_acc: 0.4679
Epoch 3/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.8013 - acc:
0.6260 - val_loss: 0.8258 - val_acc: 0.6067
Epoch 4/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.8027 - acc:
0.6340 - val_loss: 0.9166 - val_acc: 0.6498
Epoch 5/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.7224 - acc:
0.6887 - val_loss: 0.8064 - val_acc: 0.7011
Epoch 6/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.7062 - acc:
0.6751 - val_loss: 0.8184 - val_acc: 0.7089
Epoch 7/32
7352/7352 [==============================] - 42s 6ms/step - loss: 0.5461 - acc:
0.7769 - val_loss: 0.6499 - val_acc: 0.7184
Epoch 8/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.4655 - acc:
0.8428 - val_loss: 0.9957 - val_acc: 0.7228
Epoch 9/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.4051 - acc:
0.8619 - val_loss: 0.7026 - val_acc: 0.8117
Epoch 10/32
7352/7352 [==============================] - 42s 6ms/step - loss: 0.3175 - acc:
0.8995 - val_loss: 0.3685 - val_acc: 0.8653
Epoch 11/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.2710 - acc:
0.9074 - val_loss: 0.5330 - val_acc: 0.8717
Epoch 12/32
7352/7352 [==============================] - 42s 6ms/step - loss: 0.2749 - acc:
0.9151 - val_loss: 0.4381 - val_acc: 0.8768
Epoch 13/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.2383 - acc:
0.9256 - val_loss: 0.4444 - val_acc: 0.8721
Epoch 14/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1921 - acc:
0.9347 - val_loss: 0.4037 - val_acc: 0.8687
Epoch 15/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1919 - acc:
0.9374 - val_loss: 0.3120 - val_acc: 0.8992
```

```
Epoch 16/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1958 - acc:
0.9400 - val_loss: 0.2919 - val_acc: 0.9057
Epoch 17/32
7352/7352 [==============================] - 43s 6ms/step - loss: 0.2576 - acc:
0.9267 - val_loss: 0.4852 - val_acc: 0.8531
Epoch 18/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1717 - acc:
0.9387 - val_loss: 0.3191 - val_acc: 0.9026
Epoch 19/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1755 - acc:
0.9382 - val_loss: 0.3675 - val_acc: 0.9009
Epoch 20/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1639 - acc:
0.9416 - val_loss: 0.2869 - val_acc: 0.9084
Epoch 21/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1648 - acc:
0.9399 - val_loss: 0.3324 - val_acc: 0.8945
Epoch 22/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1687 - acc:
0.9391 - val_loss: 0.5019 - val_acc: 0.9053
Epoch 23/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1644 - acc:
0.9429 - val_loss: 0.3353 - val_acc: 0.9206
Epoch 24/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1592 - acc:
0.9425 - val_loss: 0.3182 - val_acc: 0.9209
Epoch 25/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1568 - acc:
0.9467 - val_loss: 0.2759 - val_acc: 0.9240
Epoch 26/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1363 - acc:
0.9461 - val_loss: 0.3511 - val_acc: 0.8918
Epoch 27/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1583 - acc:
0.9472 - val_loss: 0.3457 - val_acc: 0.9067
Epoch 28/32
7352/7352 [==============================] - 42s 6ms/step - loss: 0.1756 - acc:
0.9446 - val_loss: 0.7317 - val_acc: 0.8738
Epoch 29/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1448 - acc:
0.9517 - val_loss: 0.3776 - val_acc: 0.9125
Epoch 30/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1397 - acc:
0.9476 - val_loss: 0.4637 - val_acc: 0.9101
Epoch 31/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1483 - acc:
0.9490 - val_loss: 0.3766 - val_acc: 0.9036
Epoch 32/32
7352/7352 [==============================] - 41s 6ms/step - loss: 0.1437 - acc:
0.9509 - val_loss: 0.4356 - val_acc: 0.8972
```

Out[76]: `<keras.callbacks.History at 0x7fc0c9adf518>`

In [78]:
```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                LAYING  SITTING  ...  WALKING_DOWNSTAIRS  WALKING_UPSTAIRS
True                                 ...
LAYING                 505       32  ...                   0                 0
SITTING                  5      382  ...                   0                 4
STANDING                 0      110  ...                   0                 1
WALKING                  0        1  ...                   3                22
WALKING_DOWNSTAIRS       0        0  ...                 414                 4
WALKING_UPSTAIRS         0        0  ...                   9               453

[6 rows x 6 columns]
```

In [79]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 2s 568us/step
```

In [80]:
```
score
```

Out[80]:
```
[0.4355118162471876, 0.8971835765184933]
```

- With a simple single layer architecture we got 92.4% accuracy and a loss of 0.37 from the best model.