In [1]:

```
# Importing Libraries
```

In [2]:

```
import pandas as pd
import numpy as np
```

In [3]:

```
# 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 [4]:

```
# Data directory
DATADIR = 'UCI_HAR_Dataset'
```

In [5]:

```
# 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 [6]:

```
# 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 [7]:

```
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 [8]:

```
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 [9]:

```
# Importing tensorflow
np.random.seed(42)
import tensorflow as tf
tf.set_random_seed(42)
```

In [10]:

```
# Configuring a session
session_conf = tf.ConfigProto(
    intra_op_parallelism_threads=1,
    inter_op_parallelism_threads=1
)
```

In [11]:

```
# 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 [22]:

```
# Importing libraries
from keras.models import Sequential
from keras.layers import LSTM,BatchNormalization
from keras.layers.core import Dense, Dropout,Flatten
from keras.layers import Conv1D, MaxPooling2D
```

In [130]:

```
# Initializing parameters
epochs = 15
batch_size = 64
n_hidden =128
```

In [14]:

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

In [15]:

```
# Loading the train and test data
X_train, X_test, Y_train, Y_test = load_data()
```

```
C:\Users\mchetankumar\AppData\Local\Continuum\anaconda3\envs\TaxiEnv\lib\site-
packages\ipykernel_launcher.py:12: FutureWarning: Method .as_matrix will be removed in a future ve
rsion. Use .values instead.
  if sys.path[0] == '':
C:\Users\mchetankumar\AppData\Local\Continuum\anaconda3\envs\TaxiEnv\lib\site-
packages\ipykernel_launcher.py:11: FutureWarning: Method .as_matrix will be removed in a future ve
rsion. Use .values instead.
  # This is added back by InteractiveShellApp.init_path()
```

In [16]:

```
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 [137]:

```
# Initiliazing the sequential model
from keras import regularizers
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden,return_sequences=True,input_shape=(timesteps,
input_dim),kernel_regularizer=regularizers.l2(1e-4)))
model.add(Conv1D(8, 2, padding="same", activation="relu"))
model.add(Flatten())
# Adding a dropout layer
model.add(Dense(8))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_41 (LSTM)               (None, 128, 128)          70656
_________________________________________________________________
conv1d_73 (Conv1D)           (None, 128, 8)            2056
_________________________________________________________________
flatten_34 (Flatten)         (None, 1024)              0
_________________________________________________________________
dense_92 (Dense)             (None, 8)                 8200
_________________________________________________________________
dense_93 (Dense)             (None, 6)                 54
=================================================================
```

```
Total params: 80,966
Trainable params: 80,966
Non-trainable params: 0
_________________________________________________________________
```

In [138]:

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

In [140]:

```
# 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 1/15
7352/7352 [==============================] - 145s 20ms/step - loss: 0.1105 - acc: 0.9514 - val_los
s: 0.4019 - val_acc: 0.9009
Epoch 2/15
7352/7352 [==============================] - 147s 20ms/step - loss: 0.1071 - acc: 0.9546 - val_los
s: 0.2647 - val_acc: 0.9155
Epoch 3/15
7352/7352 [==============================] - 146s 20ms/step - loss: 0.1208 - acc: 0.9483 - val_los
s: 0.2840 - val_acc: 0.9226
Epoch 4/15
7352/7352 [==============================] - 147s 20ms/step - loss: 0.1072 - acc: 0.9531 - val_los
s: 0.3017 - val_acc: 0.9114
Epoch 5/15
7352/7352 [==============================] - 147s 20ms/step - loss: 0.1056 - acc: 0.9532 - val_los
s: 0.3101 - val_acc: 0.9101
Epoch 6/15
7352/7352 [==============================] - 148s 20ms/step - loss: 0.1062 - acc: 0.9547 - val_los
s: 0.2629 - val_acc: 0.9165
Epoch 7/15
7352/7352 [==============================] - 147s 20ms/step - loss: 0.1103 - acc: 0.9517 - val_los
s: 0.3366 - val_acc: 0.9094
Epoch 8/15
7352/7352 [==============================] - 142s 19ms/step - loss: 0.1072 - acc: 0.9539 - val_los
s: 0.2741 - val_acc: 0.9152
Epoch 9/15
7352/7352 [==============================] - 142s 19ms/step - loss: 0.1059 - acc: 0.9548 - val_los
s: 0.2995 - val_acc: 0.9152
Epoch 10/15
7352/7352 [==============================] - 138s 19ms/step - loss: 0.0990 - acc: 0.9551 - val_los
s: 0.3499 - val_acc: 0.9108
Epoch 11/15
7352/7352 [==============================] - 64s 9ms/step - loss: 0.1027 - acc: 0.9548 - val_loss:
0.3437 - val_acc: 0.9152
Epoch 12/15
7352/7352 [==============================] - 64s 9ms/step - loss: 0.1046 - acc: 0.9542 - val_loss:
nan - val_acc: 0.9006
Epoch 13/15
7352/7352 [==============================] - 65s 9ms/step - loss: 0.1063 - acc: 0.9553 - val_loss:
0.2776 - val_acc: 0.9108
Epoch 14/15
7352/7352 [==============================] - 65s 9ms/step - loss: 0.1002 - acc: 0.9576 - val_loss:
0.2808 - val_acc: 0.9013
Epoch 15/15
7352/7352 [==============================] - 68s 9ms/step - loss: 0.0951 - acc: 0.9573 - val_loss:
0.2329 - val_acc: 0.9250
```

Out[140]:

```
<keras.callbacks.History at 0x2248153a390>
```

In [141]:

In [141]:

```
# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
Pred                 LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True
LAYING                  537        0         0        0                   0
SITTING                   6      401        84        0                   0
STANDING                  0       71       461        0                   0
WALKING                   0        0         0      471                  25
WALKING_DOWNSTAIRS        0        0         0        7                 411
WALKING_UPSTAIRS          0        0         0        1                  25

Pred                 WALKING_UPSTAIRS
True
LAYING                              0
SITTING                             0
STANDING                            0
WALKING                             0
WALKING_DOWNSTAIRS                  2
WALKING_UPSTAIRS                  445
```

In [142]:

```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 5s 2ms/step
```

In [143]:

```
score
```

Out[143]:

```
[0.23294797681913876, 0.9250084832032576]
```

In [144]:

```
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Architecture", "Hidden LSTM Units", "LSTM Layers", "Dropout","Test Loss","Accurac
y"]

x.add_row(["1", "32","NA",0.50,0.41,0.91])
x.add_row(["2", "64","NA",0.25,0.30,0.92])
x.add_row(["3", "100","NA",0.75,0.56,0.90])
x.add_row(["4", "128","1",0.50,0.36,0.93])
x.add_row(["5", "128","2",0.75,0.45,0.91])
x.add_row(["6", "250","2",0.75,1.28,0.84])
x.add_row(["7", "180","2",0.25,0.50,0.90])
x.add_row(["8", "200","1",0.75,0.47,0.89])
x.add_row(["9", "128","1",0.2,0.21,0.94])
x.add_row(["10", "128","1","NA",0.23,0.93])
print(x)
```

```
+--------------+-------------------+-------------+---------+-----------+----------+
| Architecture | Hidden LSTM Units | LSTM Layers | Dropout | Test Loss | Accuracy |
+--------------+-------------------+-------------+---------+-----------+----------+
|      1       |         32        |      NA     |   0.5   |    0.41   |   0.91   |
|      2       |         64        |      NA     |   0.25  |    0.3    |   0.92   |
|      3       |        100        |      NA     |   0.75  |    0.56   |   0.9    |
|      4       |        128        |      1      |   0.5   |    0.36   |   0.93   |
|      5       |        128        |      2      |   0.75  |    0.45   |   0.91   |
|      6       |        250        |      2      |   0.75  |    1.28   |   0.84   |
|      7       |        180        |      2      |   0.25  |    0.5    |   0.9    |
|      8       |        200        |      1      |   0.75  |    0.47   |   0.89   |
|      9       |        128        |      1      |   0.2   |    0.21   |   0.94   |
|      10      |        128        |      1      |    NA   |    0.23   |   0.93   |
+--------------+-------------------+-------------+---------+-----------+----------+
```

- I was able to improve the accuracy from 90.09% to 93.62% and loss from 0.30 to 0.21