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 [12]:

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

In [23]:

```
# Initializing parameters
```

```
epochs = 30
batch_size = 16
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 [24]:

```
# Initiliazing the sequential model
from keras import regularizers
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden,input_shape=(timesteps, input_dim),kernel_regularizer=regularizers.l2(1e-4
)))
model.add(BatchNormalization())
# Adding a dropout layer
model.add(Dropout(0.2))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_2 (LSTM)                (None, 128)               70656     
_________________________________________________________________
batch_normalization_2 (Batch (None, 128)               512       
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 774       
=================================================================
Total params: 71,942
Trainable params: 71,686
Non-trainable params: 256
```

____

In [25]:

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

In [26]:

```
# 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 [==============================] - 125s 17ms/step - loss: 0.9490 - acc: 0.5926 - val_los
s: 0.7374 - val_acc: 0.6461
Epoch 2/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.6967 - acc: 0.6766 - val_los
s: 0.8233 - val_acc: 0.6451
Epoch 3/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.4318 - acc: 0.8402 - val_los
s: 0.7318 - val_acc: 0.7112
Epoch 4/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.4208 - acc: 0.8371 - val_los
s: 0.3834 - val_acc: 0.8741
Epoch 5/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.1957 - acc: 0.9286 - val_los
s: 0.2748 - val_acc: 0.9094
Epoch 6/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.1869 - acc: 0.9286 - val_los
s: 0.2424 - val_acc: 0.9084
Epoch 7/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.1828 - acc: 0.9340 - val_los
s: 0.2359 - val_acc: 0.9175
Epoch 8/30
7352/7352 [==============================] - 124s 17ms/step - loss: 0.1557 - acc: 0.9397 - val_los
s: 0.3304 - val_acc: 0.8850
Epoch 9/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.1918 - acc: 0.9252 - val_los
s: 0.3118 - val_acc: 0.9063
Epoch 10/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.1722 - acc: 0.9336 - val_los
s: 0.2821 - val_acc: 0.9046
Epoch 11/30
7352/7352 [==============================] - 124s 17ms/step - loss: 0.1419 - acc: 0.9434 - val_los
s: 0.2459 - val_acc: 0.9155
Epoch 12/30
7352/7352 [==============================] - 123s 17ms/step - loss: 0.1474 - acc: 0.9446 - val_los
s: 0.2156 - val_acc: 0.9155
Epoch 13/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.1524 - acc: 0.9441 - val_los
s: 0.2741 - val_acc: 0.9148
Epoch 14/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.1318 - acc: 0.9474 - val_los
s: 0.2568 - val_acc: 0.9274
Epoch 15/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.1260 - acc: 0.9521 - val_los
s: 0.2593 - val_acc: 0.9274
Epoch 16/30
7352/7352 [==============================] - 123s 17ms/step - loss: 0.1381 - acc: 0.9449 - val_los
s: 0.2024 - val_acc: 0.9158
Epoch 17/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.1485 - acc: 0.9425 - val_los
s: 0.2398 - val_acc: 0.9074
Epoch 18/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.1328 - acc: 0.9463 - val_los
s: 0.2459 - val_acc: 0.9243
Epoch 19/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.1402 - acc: 0.9436 - val_los
```

```
s: 0.2256 - val_acc: 0.9304
Epoch 20/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.1402 - acc: 0.9464 - val_los
s: 0.2505 - val_acc: 0.9216
Epoch 21/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.1556 - acc: 0.9412 - val_los
s: 0.3855 - val_acc: 0.8894
Epoch 22/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.1296 - acc: 0.9482 - val_los
s: 0.2030 - val_acc: 0.9355
Epoch 23/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.1332 - acc: 0.9483 - val_los
s: 0.2785 - val_acc: 0.9091
Epoch 24/30
7352/7352 [==============================] - 121s 16ms/step - loss: 0.1440 - acc: 0.9467 - val_los
s: 0.2270 - val_acc: 0.9281
Epoch 25/30
7352/7352 [==============================] - 122s 17ms/step - loss: 0.1233 - acc: 0.9493 - val_los
s: 0.2883 - val_acc: 0.9186
Epoch 26/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.1196 - acc: 0.9508 - val_los
s: 0.2630 - val_acc: 0.9169
Epoch 27/30
7352/7352 [==============================] - 129s 18ms/step - loss: 0.1194 - acc: 0.9491 - val_los
s: 0.1988 - val_acc: 0.9308
Epoch 28/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.1268 - acc: 0.9525 - val_los
s: 0.2684 - val_acc: 0.9233
Epoch 29/30
7352/7352 [==============================] - 129s 17ms/step - loss: 0.1245 - acc: 0.9508 - val_los
s: 0.2613 - val_acc: 0.9291
Epoch 30/30
7352/7352 [==============================] - 124s 17ms/step - loss: 0.1190 - acc: 0.9532 - val_los
s: 0.2125 - val_acc: 0.9362
```

Out[26]:

```
<keras.callbacks.History at 0x24ed756c9e8>
```

In [27]:

```
# 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                  2      373       115        0                   0
STANDING                 0       60       471        1                   0
WALKING                  0        0         0      493                   2
WALKING_DOWNSTAIRS       0        0         0        0                 418
WALKING_UPSTAIRS         0        0         0        0                   4

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            1
STANDING                           0
WALKING                            1
WALKING_DOWNSTAIRS                 2
WALKING_UPSTAIRS                 467
```

In [28]:

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

```
2947/2947 [==============================] - 7s 2ms/step
```

In [29]:

```
score
```

Out[29]:

```
[0.21251263201785342, 0.9362063115032236]
```

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