In [1]:
```
!unzip UCI_HAR_Dataset.zip
```

```
Archive:  UCI_HAR_Dataset.zip
  inflating: UCI_HAR_Dataset/.DS_Store
  inflating: UCI_HAR_Dataset/_DS_Store
  inflating: UCI_HAR_Dataset/activity_labels.txt
   creating: UCI_HAR_Dataset/csv_files/
  inflating: UCI_HAR_Dataset/csv_files/test.csv
  inflating: UCI_HAR_Dataset/csv_files/train.csv
  inflating: UCI_HAR_Dataset/features.txt
  inflating: UCI_HAR_Dataset/features_info.txt
  inflating: UCI_HAR_Dataset/README.txt
   creating: UCI_HAR_Dataset/test/
   creating: UCI_HAR_Dataset/test/Inertial Signals/
  inflating: UCI_HAR_Dataset/test/Inertial Signals/body_acc_x_test.txt
  inflating: UCI_HAR_Dataset/test/Inertial Signals/body_acc_y_test.txt
  inflating: UCI_HAR_Dataset/test/Inertial Signals/body_acc_z_test.txt
  inflating: UCI_HAR_Dataset/test/Inertial Signals/body_gyro_x_test.txt
  inflating: UCI_HAR_Dataset/test/Inertial Signals/body_gyro_y_test.txt
  inflating: UCI_HAR_Dataset/test/Inertial Signals/body_gyro_z_test.txt
  inflating: UCI_HAR_Dataset/test/Inertial Signals/total_acc_x_test.txt
  inflating: UCI_HAR_Dataset/test/Inertial Signals/total_acc_y_test.txt
  inflating: UCI_HAR_Dataset/test/Inertial Signals/total_acc_z_test.txt
  inflating: UCI_HAR_Dataset/test/subject_test.txt
  inflating: UCI_HAR_Dataset/test/X_test.txt
  inflating: UCI_HAR_Dataset/test/y_test.txt
   creating: UCI_HAR_Dataset/train/
  inflating: UCI_HAR_Dataset/train/.DS_Store
   creating: UCI_HAR_Dataset/train/Inertial Signals/
  inflating: UCI_HAR_Dataset/train/Inertial Signals/body_acc_x_train.txt
  inflating: UCI_HAR_Dataset/train/Inertial Signals/body_acc_y_train.txt
  inflating: UCI_HAR_Dataset/train/Inertial Signals/body_acc_z_train.txt
  inflating: UCI_HAR_Dataset/train/Inertial Signals/body_gyro_x_train.txt
  inflating: UCI_HAR_Dataset/train/Inertial Signals/body_gyro_y_train.txt
  inflating: UCI_HAR_Dataset/train/Inertial Signals/body_gyro_z_train.txt
  inflating: UCI_HAR_Dataset/train/Inertial Signals/total_acc_x_train.txt
  inflating: UCI_HAR_Dataset/train/Inertial Signals/total_acc_y_train.txt
  inflating: UCI_HAR_Dataset/train/Inertial Signals/total_acc_z_train.txt
  inflating: UCI_HAR_Dataset/train/subject_train.txt
  inflating: UCI_HAR_Dataset/train/X_train.txt
  inflating: UCI_HAR_Dataset/train/y_train.txt
```

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

In [2]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")
import random
# Importing tensorflow
np.random.seed(42)
import tensorflow as tf
tf.set_random_seed(42)


# Importing libraries
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers.core import Dense, Dropout
from keras.layers.normalization import BatchNormalization
from keras.optimizers import Adam
from keras import backend as K

from keras.layers import Conv1D, MaxPooling1D, Dense, Flatten, Dropout,Conv2D,MaxPooling2D
```

```
Using TensorFlow backend.
```

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',
}


ACTIVITIES1 = {
    0: 'WALKING',
    1: 'WALKING_UPSTAIRS',
    2: 'WALKING_DOWNSTAIRS',
}

ACTIVITIES2 = {
    0: 'SITTING',
    1: 'STANDING',
    2: 'LAYING',
}

ACTIVITIES3 = {
    0: 'SITTING',
    1: 'STANDING',
}
```

```
# 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'])

# Utility function to print the confusion matrix
def confusion_matrix_1(Y_true, Y_pred):
    Y_true = pd.Series([ACTIVITIES1[y] for y in np.argmax(Y_true, axis=1)])
    Y_pred = pd.Series([ACTIVITIES1[y] for y in np.argmax(Y_pred, axis=1)])

    return pd.crosstab(Y_true, Y_pred, rownames=['True'], colnames=['Pred'])


# Utility function to print the confusion matrix
def confusion_matrix_2(Y_true, Y_pred):
    Y_true = pd.Series([ACTIVITIES2[y] for y in np.argmax(Y_true, axis=1)])
    Y_pred = pd.Series([ACTIVITIES2[y] for y in np.argmax(Y_pred, axis=1)])

    return pd.crosstab(Y_true, Y_pred, rownames=['True'], colnames=['Pred'])

# Utility function to print the confusion matrix
def confusion_matrix_3(Y_true, Y_pred):
    Y_true = pd.Series([ACTIVITIES3[y] for y in np.argmax(Y_true, axis=1)])
    Y_pred = pd.Series([ACTIVITIES3[y] for y in np.argmax(Y_pred, axis=1)])

    return pd.crosstab(Y_true, Y_pred, rownames=['True'], colnames=['Pred'])
```

Data Gathering

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'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 [0]:
```
def load_y(subset):
    global dir_path
    if subset is "train":
        y_path = DATADIR + '/train/y_train.txt'
    elif subset is "test":
        y_path = DATADIR + '/test/y_test.txt'

    with open(y_path) as f:
        container = f.readlines()

    result = []
    for line in container:
        num_str = line.strip()
        result.append(int(num_str))
    return np.array(result)
```

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

In [0]:
```
# Import Keras
sess = tf.Session(graph=tf.get_default_graph(), config=session_conf)
K.set_session(sess)
```

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

In [0]:
```
# Loading the train and test data
X_train, X_test, Y_train, Y_test = load_data()
```

In [14]:
```
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
```

In [0]:
```
unique, counts = np.unique(Y_test, return_counts=True)
```

In [16]:
```
print("Unique classes and their counts--")
print(unique)
print(counts)
```

```
Unique classes and their counts--
[1 2 3 4 5 6]
[496 471 420 491 532 537]
```

Loading Data

In [0]:
```
# Load all train and test data (* dynamic and static data are mixed.)

X_train_all = load_signals("train")   # at this stage, the data includes both dynamic and static HAR data
Y_train_all = load_y("train")
```

In [0]:
```
X_test_all = load_signals("test")
Y_test_all = load_y("test")
```

In [19]:
```
print(X_train_all.shape, Y_test_all.shape)
print(X_test_all.shape, Y_test_all.shape) #[samples, timesteps, features]
```

```
(7352, 128, 9) (2947,)
(2947, 128, 9) (2947,)
```

Separating Data- Dynamic (WALKING, WALKING UPSTAIRS, WALKING DOWNSTAIRS)

In [0]:
```
import random

# Select dynamic HAR train data

dynamic_1 = np.where(Y_train_all == 1)[0]
dynamic_2 = np.where(Y_train_all == 2)[0]
dynamic_3 = np.where(Y_train_all == 3)[0]
dynamic = np.concatenate([dynamic_1, dynamic_2, dynamic_3])
dynamic_list = dynamic.tolist()

# Shuffle dynamic data index
r = random.random()
random.shuffle(dynamic_list, lambda: r)

dynamic = np.array(dynamic_list)

X_train = X_train_all[dynamic]
Y_train = Y_train_all[dynamic]
```

In [0]:
```
print ("train_dynamic shape: ", Y_train.shape)

np.unique(Y_train)
```

```
train_dynamic shape:  (3285,)
```

Out[287]:
```
array([1, 2, 3])
```

In [0]:
```
X_train.shape
```

Out[288]:
```
(3285, 128, 9)
```

In [0]:
```
dynamic_1 = np.where(Y_test_all == 1)[0]
dynamic_2 = np.where(Y_test_all == 2)[0]
dynamic_3 = np.where(Y_test_all == 3)[0]
dynamic = np.concatenate([dynamic_1, dynamic_2, dynamic_3])

X_test = X_test_all[dynamic]
Y_test = Y_test_all[dynamic]
```

In [0]:
```
print ("test_dynamic shape: ", Y_test.shape)

np.unique(Y_test)
```

```
test_dynamic shape:  (1387,)
```

Out[290]:
```
array([1, 2, 3])
```

In [0]:
```
# Convert to one hot encoding vector

from keras.utils import np_utils

Y_train_dynamic_ohe=np.asarray(pd.get_dummies(Y_train))
Y_test_dynamic_ohe=np.asarray(pd.get_dummies(Y_test))
```

In [0]:
```
Y_train_dynamic_ohe
```

Out[292]:
```
array([[1, 0, 0],
       [1, 0, 0],
       [0, 0, 1],
       ...,
       [0, 0, 1],
       [0, 0, 1],
       [0, 0, 1]], dtype=uint8)
```

In [0]:
```
Y_test_dynamic_ohe
```

Out[293]:
```
array([[1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       ...,
       [0, 0, 1],
       [0, 0, 1],
       [0, 0, 1]], dtype=uint8)
```

Building Model for Dynamic Data

In [0]:
```
X_train.shape
```

Out[294]:
```
(3285, 128, 9)
```

In [73]:
```
model1 = Sequential()

model1.add(Conv1D(32, 3, input_shape=(128, 9), activation='relu',kernel_initializer = 'he_normal'))
model1.add(MaxPooling1D(2))
model1.add(Dropout(0.2))
model1.add(BatchNormalization())


model1.add(Conv1D(64, 3, activation='relu',kernel_initializer = 'he_normal'))
model1.add(MaxPooling1D(2))
model1.add(BatchNormalization())
#model1.add(Dropout(0.2))

model1.add(Conv1D(80, 3,  activation='relu',kernel_initializer = 'he_normal'))
model1.add(MaxPooling1D(2))
model1.add(Dropout(0.2))
model1.add(BatchNormalization())

model1.add(Flatten())

model1.add(Dense(3, activation='softmax'))


model1.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer = 'adam')

model1.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_29 (Conv1D)           (None, 126, 32)           896
_________________________________________________________________
max_pooling1d_26 (MaxPooling (None, 63, 32)            0
_________________________________________________________________
dropout_44 (Dropout)         (None, 63, 32)            0
_________________________________________________________________
batch_normalization_25 (Batc (None, 63, 32)            128
_________________________________________________________________
conv1d_30 (Conv1D)           (None, 61, 64)            6208
_________________________________________________________________
max_pooling1d_27 (MaxPooling (None, 30, 64)            0
_________________________________________________________________
batch_normalization_26 (Batc (None, 30, 64)            256
_________________________________________________________________
conv1d_31 (Conv1D)           (None, 28, 80)            15440
_________________________________________________________________
max_pooling1d_28 (MaxPooling (None, 14, 80)            0
_________________________________________________________________
dropout_45 (Dropout)         (None, 14, 80)            0
_________________________________________________________________
batch_normalization_27 (Batc (None, 14, 80)            320
_________________________________________________________________
flatten_13 (Flatten)         (None, 1120)              0
_________________________________________________________________
dense_14 (Dense)             (None, 3)                 3363
=================================================================
Total params: 26,611
Trainable params: 26,259
Non-trainable params: 352
_________________________________________________________________
```

In [74]:
```
model1.fit(X_train, Y_train_dynamic_ohe,batch_size=32, epochs=50, verbose=2, validation_data=(X_test, Y_test_dynamic_ohe)
```

```
Train on 3285 samples, validate on 1387 samples
Epoch 1/50
 - 7s - loss: 0.7209 - acc: 0.7081 - val_loss: 0.5022 - val_acc: 0.8198
Epoch 2/50
 - 1s - loss: 0.1686 - acc: 0.9385 - val_loss: 0.3290 - val_acc: 0.8666
Epoch 3/50
 - 1s - loss: 0.0781 - acc: 0.9723 - val_loss: 0.3665 - val_acc: 0.8609
Epoch 4/50
 - 1s - loss: 0.0442 - acc: 0.9896 - val_loss: 0.4038 - val_acc: 0.8335
Epoch 5/50
 - 1s - loss: 0.0312 - acc: 0.9903 - val_loss: 0.3472 - val_acc: 0.8645
Epoch 6/50
 - 1s - loss: 0.0270 - acc: 0.9909 - val_loss: 0.2836 - val_acc: 0.8904
Epoch 7/50
 - 1s - loss: 0.0204 - acc: 0.9927 - val_loss: 0.1611 - val_acc: 0.9373
Epoch 8/50
 - 1s - loss: 0.0176 - acc: 0.9954 - val_loss: 0.1315 - val_acc: 0.9459
Epoch 9/50
 - 1s - loss: 0.0174 - acc: 0.9936 - val_loss: 0.1859 - val_acc: 0.9214
Epoch 10/50
 - 1s - loss: 0.0080 - acc: 0.9976 - val_loss: 0.1842 - val_acc: 0.9301
Epoch 11/50
 - 1s - loss: 0.0053 - acc: 0.9985 - val_loss: 0.0990 - val_acc: 0.9582
Epoch 12/50
 - 1s - loss: 0.0106 - acc: 0.9973 - val_loss: 0.1755 - val_acc: 0.9423
Epoch 13/50
 - 1s - loss: 0.0096 - acc: 0.9973 - val_loss: 0.0507 - val_acc: 0.9813
Epoch 14/50
 - 1s - loss: 0.0081 - acc: 0.9979 - val_loss: 0.0486 - val_acc: 0.9798
Epoch 15/50
 - 1s - loss: 0.0049 - acc: 0.9985 - val_loss: 0.0728 - val_acc: 0.9719
Epoch 16/50
 - 1s - loss: 0.0036 - acc: 0.9988 - val_loss: 0.0589 - val_acc: 0.9733
Epoch 17/50
 - 1s - loss: 0.0047 - acc: 0.9985 - val_loss: 0.0413 - val_acc: 0.9805
Epoch 18/50
 - 1s - loss: 0.0059 - acc: 0.9985 - val_loss: 0.0826 - val_acc: 0.9683
Epoch 19/50
 - 1s - loss: 0.0020 - acc: 0.9997 - val_loss: 0.0554 - val_acc: 0.9827
Epoch 20/50
 - 1s - loss: 0.0029 - acc: 0.9991 - val_loss: 0.0564 - val_acc: 0.9733
Epoch 21/50
 - 1s - loss: 0.0039 - acc: 0.9994 - val_loss: 0.0644 - val_acc: 0.9719
Epoch 22/50
 - 1s - loss: 0.0019 - acc: 0.9991 - val_loss: 0.0462 - val_acc: 0.9784
Epoch 23/50
 - 1s - loss: 0.0011 - acc: 1.0000 - val_loss: 0.0257 - val_acc: 0.9899
Epoch 24/50
 - 1s - loss: 0.0025 - acc: 0.9988 - val_loss: 0.0312 - val_acc: 0.9856
Epoch 25/50
 - 1s - loss: 0.0040 - acc: 0.9985 - val_loss: 0.0258 - val_acc: 0.9921
Epoch 26/50
 - 1s - loss: 0.0022 - acc: 0.9994 - val_loss: 0.0639 - val_acc: 0.9697
Epoch 27/50
 - 1s - loss: 0.0049 - acc: 0.9982 - val_loss: 0.0587 - val_acc: 0.9784
Epoch 28/50
 - 1s - loss: 0.0013 - acc: 1.0000 - val_loss: 0.0754 - val_acc: 0.9690
Epoch 29/50
 - 1s - loss: 8.9887e-04 - acc: 0.9997 - val_loss: 0.0262 - val_acc: 0.9913
Epoch 30/50
 - 1s - loss: 5.7741e-04 - acc: 1.0000 - val_loss: 0.0275 - val_acc: 0.9892
Epoch 31/50
 - 1s - loss: 0.0034 - acc: 0.9988 - val_loss: 0.0659 - val_acc: 0.9740
Epoch 32/50
 - 1s - loss: 0.0036 - acc: 0.9988 - val_loss: 0.0218 - val_acc: 0.9942
Epoch 33/50
 - 1s - loss: 0.0030 - acc: 0.9988 - val_loss: 0.0270 - val_acc: 0.9885
Epoch 34/50
 - 1s - loss: 0.0013 - acc: 0.9997 - val_loss: 0.0251 - val_acc: 0.9921
Epoch 35/50
 - 1s - loss: 0.0047 - acc: 0.9985 - val_loss: 0.0163 - val_acc: 0.9928
Epoch 36/50
 - 1s - loss: 0.0060 - acc: 0.9982 - val_loss: 0.0244 - val_acc: 0.9906
Epoch 37/50
 - 1s - loss: 5.4143e-04 - acc: 1.0000 - val_loss: 0.0192 - val_acc: 0.9928
Epoch 38/50
 - 1s - loss: 8.1451e-04 - acc: 1.0000 - val_loss: 0.0412 - val_acc: 0.9856
Epoch 39/50
 - 1s - loss: 7.8809e-04 - acc: 0.9997 - val_loss: 0.0683 - val_acc: 0.9791
Epoch 40/50
 - 1s - loss: 0.0052 - acc: 0.9982 - val_loss: 0.0650 - val_acc: 0.9769
Epoch 41/50
 - 1s - loss: 0.0117 - acc: 0.9960 - val_loss: 0.0366 - val_acc: 0.9870
Epoch 42/50
 - 1s - loss: 0.0036 - acc: 0.9991 - val_loss: 0.0533 - val_acc: 0.9755
```

```
Epoch 43/50
 - 1s - loss: 0.0039 - acc: 0.9988 - val_loss: 0.0373 - val_acc: 0.9841
Epoch 44/50
 - 1s - loss: 4.8511e-04 - acc: 1.0000 - val_loss: 0.0400 - val_acc: 0.9863
Epoch 45/50
 - 1s - loss: 5.8625e-04 - acc: 1.0000 - val_loss: 0.0313 - val_acc: 0.9856
Epoch 46/50
 - 1s - loss: 0.0023 - acc: 0.9994 - val_loss: 0.0499 - val_acc: 0.9827
Epoch 47/50
 - 2s - loss: 0.0012 - acc: 0.9997 - val_loss: 0.0788 - val_acc: 0.9712
Epoch 48/50
 - 2s - loss: 0.0024 - acc: 0.9994 - val_loss: 0.0410 - val_acc: 0.9856
Epoch 49/50
 - 2s - loss: 2.2058e-04 - acc: 1.0000 - val_loss: 0.0484 - val_acc: 0.9834
Epoch 50/50
 - 1s - loss: 2.3963e-04 - acc: 1.0000 - val_loss: 0.0437 - val_acc: 0.9805
```

Out[74]:
```
<keras.callbacks.History at 0x7f8eb06c81d0>
```

In [75]:
```
# Confusion Matrix
print(confusion_matrix(Y_test_dynamic_ohe, model1.predict(X_test)))
```

```
Pred                WALKING  WALKING_DOWNSTAIRS  WALKING_UPSTAIRS
True
WALKING                 482                  10                 4
WALKING_DOWNSTAIRS        0                 409                11
WALKING_UPSTAIRS          0                   2               469
```

In [76]:
```
score1 = model1.evaluate(X_test, Y_test_dynamic_ohe)
```

```
1387/1387 [==============================] - 0s 152us/step
```

In [77]:
```
score1
```

Out[77]:
```
[0.043732418487859935, 0.9805335255948089]
```

Separating Data- Static (SITTING, STANDING, LAYING)

In [0]:
```
import random

# Select static HAR train data

static_1 = np.where(Y_train_all == 4)[0]
static_2 = np.where(Y_train_all == 5)[0]
static_3 = np.where(Y_train_all == 6)[0]
static = np.concatenate([static_1, static_2, static_3])
static_list = static.tolist()

# Shuffle static data index
r = random.random()
random.shuffle(static_list, lambda: r)

static = np.array(static_list)

X_train = X_train_all[static]
Y_train = Y_train_all[static]
```

In [21]:
```
print ("train_static shape: ", Y_train.shape)

np.unique(Y_train)
```

```
train_static shape:  (4067,)
```

Out[21]:
```
array([4, 5, 6])
```

In [0]:
```
X_train.shape
```

Out[143]:
```
(4067, 128, 9)
```

In [0]:
```
# Select static HAR test data


static_1 = np.where(Y_test_all == 4)[0]
static_2 = np.where(Y_test_all == 5)[0]
static_3 = np.where(Y_test_all == 6)[0]
static = np.concatenate([static_1, static_2, static_3])

X_test = X_test_all[static]
Y_test = Y_test_all[static]
```

In [23]:
```
print ("static shape: ", Y_test.shape)

np.unique(Y_test)
```

```
static shape:  (1560,)
```

Out[23]:
```
array([4, 5, 6])
```

In [0]:
```
# Convert to one hot encoding vector

from keras.utils import np_utils

Y_train_static_ohe=np.asarray(pd.get_dummies(Y_train))
Y_test_static_ohe=np.asarray(pd.get_dummies(Y_test))
```

In [25]:
```
Y_train_static_ohe
```

Out[25]:
```
array([[1, 0, 0],
       [0, 1, 0],
       [1, 0, 0],
       ...,
       [0, 1, 0],
       [0, 0, 1],
       [0, 1, 0]], dtype=uint8)
```

In [26]:
```
Y_test_static_ohe
```

Out[26]:
```
array([[1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       ...,
       [0, 0, 1],
       [0, 0, 1],
       [0, 0, 1]], dtype=uint8)
```

Building Model for Static Data

In [28]:
```
X_train.shape
```

Out[28]:
```
(4067, 128, 9)
```

In [56]:
```
model2 = Sequential()


model2.add(Conv1D(50, 3, input_shape=(128, 9), activation='tanh',kernel_initializer = 'he_normal'))
model2.add(MaxPooling1D(2))
model2.add(BatchNormalization())
model2.add(Dropout(0.2))


model2.add(Conv1D(100, 3, activation='tanh',kernel_initializer = 'he_normal'))
model2.add(MaxPooling1D(2))
model2.add(BatchNormalization())
model2.add(Dropout(0.2))



model2.add(LSTM(32,activation = 'tanh',kernel_initializer = 'he_normal',return_sequences = True))
model2.add(Dropout(0.5))


model2.add(LSTM(80,activation = 'tanh',kernel_initializer = 'he_normal',return_sequences = True))
model2.add(Dropout(0.5))



model2.add(Flatten())


model2.add(Dense(3, activation='softmax'))



model2.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer = 'adam')

model2.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_20 (Conv1D)           (None, 126, 50)           1400
_________________________________________________________________
max_pooling1d_17 (MaxPooling (None, 63, 50)            0
_________________________________________________________________
batch_normalization_17 (Batc (None, 63, 50)            200
_________________________________________________________________
dropout_34 (Dropout)         (None, 63, 50)            0
_________________________________________________________________
conv1d_21 (Conv1D)           (None, 61, 100)           15100
_________________________________________________________________
max_pooling1d_18 (MaxPooling (None, 30, 100)           0
_________________________________________________________________
batch_normalization_18 (Batc (None, 30, 100)           400
_________________________________________________________________
dropout_35 (Dropout)         (None, 30, 100)           0
_________________________________________________________________
lstm_17 (LSTM)               (None, 30, 32)            17024
_________________________________________________________________
dropout_36 (Dropout)         (None, 30, 32)            0
_________________________________________________________________
lstm_18 (LSTM)               (None, 30, 80)            36160
_________________________________________________________________
dropout_37 (Dropout)         (None, 30, 80)            0
_________________________________________________________________
flatten_10 (Flatten)         (None, 2400)              0
_________________________________________________________________
dense_11 (Dense)             (None, 3)                 7203
=================================================================
Total params: 77,487
Trainable params: 77,187
Non-trainable params: 300
_________________________________________________________________
```

In [57]:
```
model2.fit(X_train, Y_train_static_ohe,batch_size=32, epochs=161, verbose=2, validation_data=(X_test, Y_test_static_ohe))
```

```
Train on 4067 samples, validate on 1560 samples
Epoch 1/161
 - 25s - loss: 0.2918 - acc: 0.8812 - val_loss: 0.4686 - val_acc: 0.8481
Epoch 2/161
 - 19s - loss: 0.2519 - acc: 0.8992 - val_loss: 0.2824 - val_acc: 0.9038
Epoch 3/161
 - 19s - loss: 0.2474 - acc: 0.8962 - val_loss: 0.3019 - val_acc: 0.8808
Epoch 4/161
 - 19s - loss: 0.2167 - acc: 0.9134 - val_loss: 0.2632 - val_acc: 0.8776
Epoch 5/161
 - 19s - loss: 0.2258 - acc: 0.9031 - val_loss: 0.3321 - val_acc: 0.8795
Epoch 6/161
 - 19s - loss: 0.2202 - acc: 0.9073 - val_loss: 0.2725 - val_acc: 0.9064
Epoch 7/161
 - 19s - loss: 0.2049 - acc: 0.9120 - val_loss: 0.3297 - val_acc: 0.8814
Epoch 8/161
 - 19s - loss: 0.2048 - acc: 0.9127 - val_loss: 0.3491 - val_acc: 0.8583
Epoch 9/161
 - 19s - loss: 0.2026 - acc: 0.9127 - val_loss: 0.2849 - val_acc: 0.9103
```

In [58]:
```
# Confusion Matrix
print(confusion_matrix_2(Y_test_static_ohe, model2.predict(X_test)))
```

```
Pred      LAYING  SITTING  STANDING
True
LAYING       537        0         0
SITTING        2      450        39
STANDING       0       22       510
```

In [61]:
```
score2 = model2.evaluate(X_test, Y_test_static_ohe)
```

```
1560/1560 [==============================] - 2s 2ms/step
```

In [64]:
```
score2
```

Out[64]:
```
[0.18009876538986808, 0.9679153846153846]
```

CONCLUSION

In [78]:
```
#http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["MODEL CLASSES","ACCURACY"]

x.add_row(["MODEL 1- DYNAMIC(WALKING, DOWNWARDS, UPWADRS)",score1[1]])
x.add_row(["Model 2- STATIC(SITTING, STANDING, LAYING)",score2[1]])


print(x)
```

```
+-----------------------------------------------+--------------------+
|                MODEL CLASSES                  |      ACCURACY      |
+-----------------------------------------------+--------------------+
| MODEL 1- DYNAMIC(WALKING, DOWNWARDS, UPWADRS) | 0.9805335255948089 |
|   Model 2- STATIC(SITTING, STANDING, LAYING)  | 0.9679153846153846 |
+-----------------------------------------------+--------------------+
```

In [0]:
```

```