In [0]:

```
# Importing libraries
import pandas as pd
import numpy as np

from keras.models import Sequential, Model
from keras.layers import LSTM, Conv1D, MaxPooling1D, Flatten, LSTM, BatchNormalization, Input
from keras.layers.core import Dense, Dropout
from keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint
import keras

from keras.regularizers import l1, l2, l1_l2
from sklearn.model_selection import train_test_split
from keras.models import load_model
from sklearn.metrics import accuracy_score

from prettytable import PrettyTable
```

In [0]:

```
pt = PrettyTable()
pt.field_names = ['Model', 'Loss', 'Test Accuracy']
```

In [2]:

```
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&scope=email%20https%3A%2F%2Fwww.googlea
%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20https%3A%2
.googleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code

Enter your authorization code:
··········
Mounted at /content/drive
```

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 = 'drive/My Drive/CoLab/Human Activity Recognition/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 = []
```

```python
    for signal in SIGNALS:
        filename = f'/'.join([DATADIR,'{0}/Inertial Signals/{1}_{0}.txt'.format(subset, signal)])
        signals_data.append(
            _read_csv(filename).values
        )

    # 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]:

```python
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'/'.join([DATADIR,'{0}/y_{0}.txt'.format(subset)])
    y = _read_csv(filename)[0]

    return pd.get_dummies(y).values
```

In [0]:

```python
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]:

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

In [0]:

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

In [12]:

```
timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = _count_classes(Y_train)

n_hidden = 32
np.random.seed(23)

print(timesteps)
print(input_dim)
print(len(X_train))
```

```
128
9
7352
```

- Defining the Architecture of LSTM

Model - 1 Using multilayer LSTM

- Splitting the data into Train and Validation data

In [0]:

```
trainX, valX, trainy, valy = train_test_split(X_train, Y_train, test_size=.33, random_state=23)
```

In [0]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(CuDNNLSTM(64, return_sequences=True, input_shape=(timesteps, input_dim)))
model.add(Dropout(rate=0.7))

model.add(CuDNNLSTM(32, input_shape=(timesteps, input_dim)))
model.add(Dropout(rate=0.5))

model.add(Dense(100, activation='relu'))
model.add(BatchNormalization())
# model.add(Dropout(rate=0.7))

model.add(Dense(n_classes, activation='sigmoid'))

# model.summary()
```

```
model.compile(loss='categorical_crossentropy', optimizer=keras.optimizers.Adam(), metrics=['accuracy'])
```

In [0]:

```
# Training the model
model.fit(trainX,
          trainy,
          batch_size=8,
          validation_data=(valX, valy),
          epochs=50,
          callbacks=[EarlyStopping(monitor='val_acc', patience=25), ReduceLROnPlateau(monitor='val_acc', factor=0.2, patience=5, min_lr=.0001)])
```

```
Train on 4925 samples, validate on 2427 samples
Epoch 1/50
4925/4925 [==============================] - 17s 3ms/step - loss: 1.2205 - acc: 0.4721 - val_loss: 0.9024 - val_acc: 0.4800
Epoch 2/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.8310 - acc: 0.5949 - val_loss: 0.6706 - val_acc: 0.6551
Epoch 3/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.7441 - acc: 0.6311 - val_loss: 0.6190 - val_acc: 0.6593
Epoch 4/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.6962 - acc: 0.6499 - val_loss: 0.6139 - val_acc: 0.6642
Epoch 5/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.7006 - acc: 0.6493 - val_loss: 0.6304 - val_acc: 0.6708
Epoch 6/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.7043 - acc: 0.6491 - val_loss: 0.6064 - val_acc: 0.6576
Epoch 7/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.6802 - acc: 0.6502 - val_loss: 0.6152 - val_acc: 0.6716
Epoch 8/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.7519 - acc: 0.6337 - val_loss: 1.9287 - val_acc: 0.4569
Epoch 9/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.8927 - acc: 0.5848 - val_loss: 0.6260 - val_acc: 0.6568
Epoch 10/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.6979 - acc: 0.6445 - val_loss: 0.6561 - val_acc: 0.6539
Epoch 11/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.6845 - acc: 0.6475 - val_loss: 0.6186 - val_acc: 0.6663
Epoch 12/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.6696 - acc: 0.6589 - val_loss: 0.6209 - val_acc: 0.6728
Epoch 13/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.6523 - acc: 0.6615 - val_loss: 0.6036 - val_acc: 0.6782
Epoch 14/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.6346 - acc: 0.6908 - val_loss: 0.5123 - val_acc: 0.7553
Epoch 15/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.6431 - acc: 0.7232 - val_loss: 0.4792 - val_acc: 0.7895
Epoch 16/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.4728 - acc: 0.7963 - val_loss: 0.3330 - val_acc: 0.8896
Epoch 17/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.4983 - acc: 0.8110 - val_loss: 0.3650 - val_acc: 0.8826
Epoch 18/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.4729 - acc: 0.8185 - val_loss: 0.3075 - val_acc: 0.8649
Epoch 19/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.4875 - acc: 0.8037 - val_loss: 0.3148 - val_acc: 0.9188
```

```
Epoch 20/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.4763 - acc: 0.8162 - val_loss: 0.3779 - val_acc: 0.8739
Epoch 21/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.3595 - acc: 0.8650 - val_loss: 0.1515 - val_acc: 0.9526
Epoch 22/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.3260 - acc: 0.8930 - val_loss: 0.1731 - val_acc: 0.9324
Epoch 23/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.2551 - acc: 0.9127 - val_loss: 0.1466 - val_acc: 0.9419
Epoch 24/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.1867 - acc: 0.9350 - val_loss: 0.1181 - val_acc: 0.9547
Epoch 25/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.1739 - acc: 0.9340 - val_loss: 0.1176 - val_acc: 0.9506
Epoch 26/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.2980 - acc: 0.9003 - val_loss: 0.1437 - val_acc: 0.9312
Epoch 27/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.1972 - acc: 0.9267 - val_loss: 0.1317 - val_acc: 0.9547
Epoch 28/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.2369 - acc: 0.9198 - val_loss: 0.1266 - val_acc: 0.9411
Epoch 29/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.1773 - acc: 0.9373 - val_loss: 0.1109 - val_acc: 0.9592
Epoch 30/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.2227 - acc: 0.9155 - val_loss: 0.3197 - val_acc: 0.8199
Epoch 31/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.2268 - acc: 0.9170 - val_loss: 0.1545 - val_acc: 0.9444
Epoch 32/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.1865 - acc: 0.9330 - val_loss: 0.1287 - val_acc: 0.9485
Epoch 33/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.2302 - acc: 0.9241 - val_loss: 0.1236 - val_acc: 0.9580
Epoch 34/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.1845 - acc: 0.9358 - val_loss: 0.1237 - val_acc: 0.9580
Epoch 35/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.1512 - acc: 0.9482 - val_loss: 0.1369 - val_acc: 0.9477
Epoch 36/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.1494 - acc: 0.9462 - val_loss: 0.1134 - val_acc: 0.9584
Epoch 37/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.1428 - acc: 0.9484 - val_loss: 0.1159 - val_acc: 0.9539
Epoch 38/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.1363 - acc: 0.9488 - val_loss: 0.1395 - val_acc: 0.9576
Epoch 39/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.1361 - acc: 0.9521 - val_loss: 0.1105 - val_acc: 0.9584
Epoch 40/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.1420 - acc: 0.9486 - val_loss: 0.1129 - val_acc: 0.9584
Epoch 41/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.1297 - acc: 0.9527 - val_loss: 0.1146 - val_acc: 0.9588
Epoch 42/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.1343 - acc: 0.9519 - val_loss: 0.1112 - val_acc: 0.9580
Epoch 43/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.1272 - acc: 0.9515 - val_loss: 0.1103 - val_acc: 0.9543
Epoch 44/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.1311 - acc: 0.9513 - val_loss: 0.1143 - val_acc: 0.9555
Epoch 45/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.1294 - acc: 0.9519 - val_loss: 0.1092 - val_acc: 0.9584
Epoch 46/50
```

```
Epoch 46/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.1258 - acc: 0.9501 - val_loss: 0.1079 - val_acc: 0.9604
Epoch 47/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.1288 - acc: 0.9539 - val_loss: 0.1061 - val_acc: 0.9604
Epoch 48/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.1445 - acc: 0.9480 - val_loss: 0.1143 - val_acc: 0.9576
Epoch 49/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.1337 - acc: 0.9517 - val_loss: 0.1117 - val_acc: 0.9592
Epoch 50/50
4925/4925 [==============================] - 10s 2ms/step - loss: 0.1252 - acc: 0.9513 - val_loss: 0.1048 - val_acc: 0.9613
```

Out[0]:

```
<keras.callbacks.History at 0x7fbcf2813240>
```

In [0]:

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

```
2947/2947 [==============================] - 1s 263us/step
```

Out[0]:

```
[0.42299219827817475, 0.9002375296912114]
```

In [0]:

```
print('Loss : {} | Accuracy : {} %'.format(*np.round([0.42299219827817475, 0.9002375296912114*100.], 3)))
```

```
Loss : 0.423 | Accuracy : 90.024 %
```

- With a simple 2 LSTM layers architecture we got **90.024% test accuracy** and a **loss of 0.423**

In [0]:

```
pt.add_row(['2 Layers LSTM', 0.423, '90.024 %'])
```

In [0]:

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

Out[0]:

Pred	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS

Pred	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
True						
LAYING	514	0	1	0	0	22
SITTING	0	394	94	0	0	3
STANDING	0	100	430	2	0	0
WALKING	0	0	0	488	0	8
WALKING_DOWNSTAIRS	0	0	0	1	418	1
WALKING_UPSTAIRS	0	0	0	43	19	409

Model - 2 Using Conv nets

Tried with various hyper parameter values by hit and trail.

In [0]:

```
model = Sequential()
model.add(Conv1D(filters=32, kernel_size=3, activation='relu', kernel_initializer='he_normal', kernel_regularizer=l2(.001), input_shape=(timesteps,
input_dim)))
model.add(Conv1D(filters=32, kernel_size=3, activation='relu', kernel_initializer='he_normal', kernel_regularizer=l2(.0001)))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dropout(0.7))
model.add(Dense(128, activation='relu', kernel_initializer='he_normal', kernel_regularizer=l2(.001)))
model.add(BatchNormalization())
model.add(Dropout(0.7))
model.add(Dense(6, activation='softmax'))
# model.summary()

model.compile(loss='categorical_crossentropy', optimizer=keras.optimizers.Adam(), metrics=['accuracy'])
```

In [0]:

```
!mkdir ./bestModel
```

In [0]:

```
!rm ./bestModel/*
```

In [0]:

```
# Training the model
model.fit(trainX,
          trainy,
          batch_size=8,
```

```
          validation_data=(valX, valy),
          epochs=50,
          callbacks=[ModelCheckpoint('bestModel/bestmodel.hdf5', monitor='val_acc', save_best_only=True),\
                     EarlyStopping(monitor='val_acc', patience=15),\
                     ReduceLROnPlateau(monitor='val_acc', factor=0.2, patience=5, min_lr=.0001)])
```

```
Train on 4925 samples, validate on 2427 samples
Epoch 1/50
4925/4925 [==============================] - 5s 926us/step - loss: 0.2105 - acc: 0.9511 - val_loss: 0.1696 - val_acc: 0.9629
Epoch 2/50
4925/4925 [==============================] - 4s 899us/step - loss: 0.2159 - acc: 0.9486 - val_loss: 0.1637 - val_acc: 0.9633
Epoch 3/50
4925/4925 [==============================] - 4s 896us/step - loss: 0.2086 - acc: 0.9535 - val_loss: 0.1631 - val_acc: 0.9621
Epoch 4/50
4925/4925 [==============================] - 4s 878us/step - loss: 0.2033 - acc: 0.9551 - val_loss: 0.1629 - val_acc: 0.9637
Epoch 5/50
4925/4925 [==============================] - 4s 889us/step - loss: 0.2037 - acc: 0.9539 - val_loss: 0.1610 - val_acc: 0.9633
Epoch 6/50
4925/4925 [==============================] - 4s 903us/step - loss: 0.2063 - acc: 0.9507 - val_loss: 0.1590 - val_acc: 0.9633
Epoch 7/50
4925/4925 [==============================] - 4s 881us/step - loss: 0.2016 - acc: 0.9521 - val_loss: 0.1566 - val_acc: 0.9642
Epoch 8/50
4925/4925 [==============================] - 4s 879us/step - loss: 0.2067 - acc: 0.9529 - val_loss: 0.1575 - val_acc: 0.9629
Epoch 9/50
4925/4925 [==============================] - 4s 907us/step - loss: 0.2039 - acc: 0.9492 - val_loss: 0.1552 - val_acc: 0.9646
Epoch 10/50
4925/4925 [==============================] - 4s 904us/step - loss: 0.2042 - acc: 0.9519 - val_loss: 0.1564 - val_acc: 0.9637
Epoch 11/50
4925/4925 [==============================] - 4s 882us/step - loss: 0.2085 - acc: 0.9541 - val_loss: 0.1523 - val_acc: 0.9625
Epoch 12/50
4925/4925 [==============================] - 4s 898us/step - loss: 0.2010 - acc: 0.9501 - val_loss: 0.1523 - val_acc: 0.9637
Epoch 13/50
4925/4925 [==============================] - 4s 893us/step - loss: 0.1947 - acc: 0.9533 - val_loss: 0.1511 - val_acc: 0.9637
Epoch 14/50
4925/4925 [==============================] - 4s 912us/step - loss: 0.2002 - acc: 0.9531 - val_loss: 0.1498 - val_acc: 0.9646
Epoch 15/50
4925/4925 [==============================] - 4s 894us/step - loss: 0.2020 - acc: 0.9529 - val_loss: 0.1513 - val_acc: 0.9646
Epoch 16/50
4925/4925 [==============================] - 4s 904us/step - loss: 0.1872 - acc: 0.9568 - val_loss: 0.1482 - val_acc: 0.9646
Epoch 17/50
4925/4925 [==============================] - 5s 950us/step - loss: 0.1828 - acc: 0.9576 - val_loss: 0.1497 - val_acc: 0.9621
Epoch 18/50
4925/4925 [==============================] - 5s 969us/step - loss: 0.1976 - acc: 0.9498 - val_loss: 0.1483 - val_acc: 0.9637
Epoch 19/50
4925/4925 [==============================] - 4s 907us/step - loss: 0.1923 - acc: 0.9561 - val_loss: 0.1470 - val_acc: 0.9642
Epoch 20/50
4925/4925 [==============================] - 4s 908us/step - loss: 0.1981 - acc: 0.9511 - val_loss: 0.1453 - val_acc: 0.9658
Epoch 21/50
4925/4925 [==============================] - 4s 889us/step - loss: 0.1826 - acc: 0.9574 - val_loss: 0.1454 - val_acc: 0.9666
Epoch 22/50
4925/4925 [==============================] - 4s 911us/step - loss: 0.1887 - acc: 0.9557 - val_loss: 0.1418 - val_acc: 0.9646
Epoch 23/50
```

```
4925/4925 [==============================] - 4s 888us/step - loss: 0.1927 - acc: 0.9521 - val_loss: 0.1439 - val_acc: 0.9670
Epoch 24/50
4925/4925 [==============================] - 4s 907us/step - loss: 0.1799 - acc: 0.9539 - val_loss: 0.1437 - val_acc: 0.9621
Epoch 25/50
4925/4925 [==============================] - 4s 873us/step - loss: 0.1810 - acc: 0.9576 - val_loss: 0.1410 - val_acc: 0.9646
Epoch 26/50
4925/4925 [==============================] - 4s 888us/step - loss: 0.1760 - acc: 0.9586 - val_loss: 0.1452 - val_acc: 0.9658
Epoch 27/50
4925/4925 [==============================] - 4s 885us/step - loss: 0.1821 - acc: 0.9543 - val_loss: 0.1405 - val_acc: 0.9674
Epoch 28/50
4925/4925 [==============================] - 4s 879us/step - loss: 0.1836 - acc: 0.9549 - val_loss: 0.1377 - val_acc: 0.9674
Epoch 29/50
4925/4925 [==============================] - 4s 901us/step - loss: 0.1873 - acc: 0.9551 - val_loss: 0.1370 - val_acc: 0.9670
Epoch 30/50
4925/4925 [==============================] - 4s 891us/step - loss: 0.1858 - acc: 0.9555 - val_loss: 0.1378 - val_acc: 0.9666
Epoch 31/50
4925/4925 [==============================] - 4s 872us/step - loss: 0.1741 - acc: 0.9588 - val_loss: 0.1358 - val_acc: 0.9650
Epoch 32/50
4925/4925 [==============================] - 4s 896us/step - loss: 0.1757 - acc: 0.9602 - val_loss: 0.1353 - val_acc: 0.9658
Epoch 33/50
4925/4925 [==============================] - 4s 893us/step - loss: 0.1682 - acc: 0.9630 - val_loss: 0.1353 - val_acc: 0.9687
Epoch 34/50
4925/4925 [==============================] - 4s 885us/step - loss: 0.1807 - acc: 0.9537 - val_loss: 0.1374 - val_acc: 0.9674
Epoch 35/50
4925/4925 [==============================] - 4s 884us/step - loss: 0.1661 - acc: 0.9606 - val_loss: 0.1341 - val_acc: 0.9691
Epoch 36/50
4925/4925 [==============================] - 4s 908us/step - loss: 0.1746 - acc: 0.9570 - val_loss: 0.1321 - val_acc: 0.9674
Epoch 37/50
4925/4925 [==============================] - 4s 881us/step - loss: 0.1714 - acc: 0.9570 - val_loss: 0.1314 - val_acc: 0.9654
Epoch 38/50
4925/4925 [==============================] - 4s 893us/step - loss: 0.1697 - acc: 0.9592 - val_loss: 0.1324 - val_acc: 0.9679
Epoch 39/50
4925/4925 [==============================] - 4s 896us/step - loss: 0.1624 - acc: 0.9614 - val_loss: 0.1291 - val_acc: 0.9699
Epoch 40/50
4925/4925 [==============================] - 4s 884us/step - loss: 0.1677 - acc: 0.9582 - val_loss: 0.1308 - val_acc: 0.9707
Epoch 41/50
4925/4925 [==============================] - 4s 907us/step - loss: 0.1845 - acc: 0.9529 - val_loss: 0.1290 - val_acc: 0.9720
Epoch 42/50
4925/4925 [==============================] - 4s 884us/step - loss: 0.1775 - acc: 0.9557 - val_loss: 0.1311 - val_acc: 0.9703
Epoch 43/50
4925/4925 [==============================] - 4s 892us/step - loss: 0.1667 - acc: 0.9588 - val_loss: 0.1279 - val_acc: 0.9679
Epoch 44/50
4925/4925 [==============================] - 4s 877us/step - loss: 0.1652 - acc: 0.9598 - val_loss: 0.1368 - val_acc: 0.9716
Epoch 45/50
4925/4925 [==============================] - 4s 898us/step - loss: 0.1726 - acc: 0.9563 - val_loss: 0.1286 - val_acc: 0.9687
Epoch 46/50
4925/4925 [==============================] - 4s 888us/step - loss: 0.1660 - acc: 0.9598 - val_loss: 0.1270 - val_acc: 0.9699
Epoch 47/50
4925/4925 [==============================] - 4s 888us/step - loss: 0.1616 - acc: 0.9612 - val_loss: 0.1266 - val_acc: 0.9699
Epoch 48/50
4925/4925 [==============================] - 4s 894us/step - loss: 0.1642 - acc: 0.9614 - val_loss: 0.1280 - val_acc: 0.9670
Epoch 49/50
4925/4925 [==============================] - 4s 885us/step - loss: 0.1657 - acc: 0.9610 - val_loss: 0.1241 - val_acc: 0.9707
```

```
Epoch 50/50
4925/4925 [==============================] - 4s 900us/step - loss: 0.1690 - acc: 0.9578 - val_loss: 0.1332 - val_acc: 0.9674
```

Out[0]:

```
<keras.callbacks.History at 0x7fbcf7dbc630>
```

Loading best model so far

In [0]:

```
#Loading best model at epoch with high val accuracy
model = load_model('bestModel/bestmodel.hdf5')
```

In [0]:

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

```
2947/2947 [==============================] - 0s 80us/step
[0.343903040034936, 0.9229725144214456]
```

In [0]:

```
print('Loss : {} | Accuracy : {} %'.format(*np.round([0.343903040034936, 0.9229725144214456*100.], 2)))
```

```
Loss : 0.344 | Accuracy : 92.297 %
```

- With a simple 2 convolution layers and 1 max-pooling architecture we got **92.297% test accuracy** and a **loss of 0.344**

In [0]:

```
pt.add_row(['2 Layers 1D Convolutions', 0.344, '92.297 %'])
```

In [0]:

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

Out[0]:

Pred	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
True						

Pred	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
True						
LAYING	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
SITTING	0	399	68	0	0	24
STANDING	0	72	456	0	0	4
WALKING	0	0	0	481	14	1
WALKING_DOWNSTAIRS	0	0	0	0	420	0
WALKING_UPSTAIRS	0	0	0	2	20	449

Model - 3 Using Divide and conquer Conv nets

References:

- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5949027/
- https://github.com/maxpumperla/hyperas

Data Preparation Basic

In [0]:

```
from keras.layers import multiply
import keras
import keras.backend as K
```

In [0]:

```
def load_y_new(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'/'.join([DATADIR,'{0}/y_{0}.txt'.format(subset)])
    y = _read_csv(filename)[0]

    return y.values
```

In [0]:

```
X_train, X_test = load_signals('train'), load_signals('test')
y_train, y_test = load_y_new('train'), load_y_new('test')
```

In [0]:

In [0]:

```
y_train_bin, y_test_bin = pd.Series(y_train).map(dict(zip(range(1,7), [1]*3+[0]*3))).values,\
                           pd.Series(y_test).map(dict(zip(range(1,7), [1]*3+[0]*3))).values
```

In [0]:

```
# Dynamic class data
X_train_dynamic, X_test_dynamic = X_train[y_train_bin==1], X_test[y_test_bin==1]

y_train_dynamic, y_test_dynamic = y_train[y_train_bin==1], y_test[y_test_bin==1]

# Static class data
X_train_static, X_test_static = X_train[y_train_bin==0], X_test[y_test_bin==0]

y_train_static, y_test_static = y_train[y_train_bin==0], y_test[y_test_bin==0]
```

In [0]:

```
y_train_bin, y_test_bin = pd.get_dummies(y_train_bin).values,\
                                     pd.get_dummies(y_test_bin).values

y_train_dynamic, y_test_dynamic = pd.get_dummies(y_train_dynamic).values,\
                                                 pd.get_dummies(y_test_dynamic).values

y_train_static, y_test_static = pd.get_dummies(y_train_static).values,\
                                              pd.get_dummies(y_test_static).values
```

Base Binary Model

In [0]:

```
main_input = Input(shape=(timesteps, input_dim), name='main_input')

# Base Binary Model
x = Conv1D(filters=16, kernel_size=3, activation='relu', kernel_initializer='he_normal', kernel_regularizer=l2(.0001))(main_input)
x = MaxPooling1D(pool_size=2)(x)
x = Flatten()(x)
x = Dropout(rate=.5)(x)
x = Dense(16, activation='relu', kernel_initializer='he_normal', kernel_regularizer=l2(.001))(x)
x = BatchNormalization()(x)
x = Dropout(rate=.65)(x)
out = Dense(2, activation='softmax', name='binary_out')(x)

bin_model = Model(inputs=main_input, outputs=out)

bin_model.compile(loss='binary_crossentropy', optimizer=keras.optimizers.Adam(.01), metrics=['accuracy'])
```

In [0]:

```python
bin_model.fit(X_train,
              y_train_bin,
              batch_size=8,
              validation_data=(X_test, y_test_bin),
              epochs=20,
              callbacks=[EarlyStopping(monitor='val_acc', patience=10), ReduceLROnPlateau(monitor='val_acc', factor=0.2, patience=3, min_lr=.0001)])
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/20
7352/7352 [==============================] - 15s 2ms/step - loss: 0.2863 - acc: 0.9479 - val_loss: 0.1700 - val_acc: 0.9986
Epoch 2/20
7352/7352 [==============================] - 8s 1ms/step - loss: 0.2714 - acc: 0.9686 - val_loss: 0.1841 - val_acc: 0.9854
Epoch 3/20
7352/7352 [==============================] - 8s 1ms/step - loss: 0.2654 - acc: 0.9767 - val_loss: 0.1852 - val_acc: 0.9946
Epoch 4/20
7352/7352 [==============================] - 8s 1ms/step - loss: 0.2800 - acc: 0.9720 - val_loss: 0.1329 - val_acc: 0.9986
Epoch 5/20
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1795 - acc: 0.9789 - val_loss: 0.0813 - val_acc: 0.9986
Epoch 6/20
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1202 - acc: 0.9839 - val_loss: 0.0524 - val_acc: 0.9993
Epoch 7/20
7352/7352 [==============================] - 8s 1ms/step - loss: 0.0806 - acc: 0.9902 - val_loss: 0.0423 - val_acc: 0.9986
Epoch 8/20
7352/7352 [==============================] - 8s 1ms/step - loss: 0.0986 - acc: 0.9861 - val_loss: 0.0416 - val_acc: 0.9986
Epoch 9/20
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1006 - acc: 0.9850 - val_loss: 0.0400 - val_acc: 0.9993
Epoch 10/20
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1014 - acc: 0.9791 - val_loss: 0.0355 - val_acc: 0.9993
Epoch 11/20
7352/7352 [==============================] - 8s 1ms/step - loss: 0.0702 - acc: 0.9893 - val_loss: 0.0330 - val_acc: 0.9993
Epoch 12/20
7352/7352 [==============================] - 8s 1ms/step - loss: 0.0825 - acc: 0.9865 - val_loss: 0.0301 - val_acc: 0.9990
Epoch 13/20
7352/7352 [==============================] - 7s 1ms/step - loss: 0.0714 - acc: 0.9849 - val_loss: 0.0287 - val_acc: 0.9990
Epoch 14/20
7352/7352 [==============================] - 8s 1ms/step - loss: 0.0721 - acc: 0.9898 - val_loss: 0.0281 - val_acc: 0.9990
Epoch 15/20
7352/7352 [==============================] - 8s 1ms/step - loss: 0.0761 - acc: 0.9861 - val_loss: 0.0273 - val_acc: 0.9990
Epoch 16/20
7352/7352 [==============================] - 8s 1ms/step - loss: 0.0737 - acc: 0.9868 - val_loss: 0.0272 - val_acc: 0.9990
```

Out[0]:

```
<keras.callbacks.History at 0x7f98b29eef28>
```

In [0]:

```python
bin_loss, bin_acc = bin_model.evaluate(X_test, y_test_bin)
```

```
2947/2947 [==============================] - 0s 141us/step
```

In [0]:

```
print('Loss : {} | Accuracy : {} %'.format(bin_loss, bin_acc*100.))
```

```
Loss : 0.027248992813763112 | Accuracy : 99.8982015609094 %
```

In [0]:

```
# Utility function to print the confusion matrix
def confusion_matrix_bin(Y_true, Y_pred, ACTIVITIES):
    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'])
```

In [0]:

```
confusion_matrix_bin(y_test_bin, bin_model.predict(X_test), {
    0: 'Static',
    1: 'Dynamic',
})
```

Out[0]:

Pred	Dynamic	Static
True		
Dynamic	1387	0
Static	3	1557

In [0]:

```
bin_model.save('Base_Binary_model.hdf5')
```

Dynamic class Model

In [0]:

```
# Model - 1
dynamic_input = Input(shape=(timesteps, input_dim), name='dynamic_input')

x = Conv1D(filters=64, kernel_size=3, activation='relu', )(dynamic_input)
```

```
y = Conv1D(filters=64, kernel_size=3, activation='relu',)(dynamic_input)
y = Conv1D(filters=32, kernel_size=3, activation='relu',)(y)
y = MaxPooling1D(pool_size=2)(y)
y = Flatten()(y)
y = Dropout(rate=.6)(y)
y = Dense(32, activation='relu', kernel_regularizer=l2(.001))(y)
y = BatchNormalization()(y)
y = Dropout(.6)(y)
dynamic_output = Dense(3, activation='softmax', name='dynamic_out')(y)

dynamic_model = Model(inputs=dynamic_input, outputs=dynamic_output)

dynamic_model.compile(loss='categorical_crossentropy', optimizer=keras.optimizers.rmsprop(.01), metrics=['accuracy'])
```

In [0]:

```
dynamic_model.fit(X_train_dynamic,
              y_train_dynamic,
              batch_size=8,
              validation_data=(X_test_dynamic, y_test_dynamic),
              epochs=50,
              callbacks=[EarlyStopping(monitor='val_acc', patience=25), ReduceLROnPlateau(monitor='val_acc', factor=0.2, patience=5, min_lr=.001)])
```

```
Train on 3285 samples, validate on 1387 samples
Epoch 1/50
3285/3285 [==============================] - 3s 787us/step - loss: 0.1978 - acc: 0.9522 - val_loss: 0.1291 - val_acc: 0.9647
Epoch 2/50
3285/3285 [==============================] - 3s 791us/step - loss: 0.2071 - acc: 0.9482 - val_loss: 0.1693 - val_acc: 0.9603
Epoch 3/50
3285/3285 [==============================] - 3s 791us/step - loss: 0.2372 - acc: 0.9434 - val_loss: 0.1323 - val_acc: 0.9733
Epoch 4/50
3285/3285 [==============================] - 3s 782us/step - loss: 0.2026 - acc: 0.9470 - val_loss: 0.1143 - val_acc: 0.9740
Epoch 5/50
3285/3285 [==============================] - 3s 780us/step - loss: 0.2680 - acc: 0.9376 - val_loss: 0.1669 - val_acc: 0.9625
Epoch 6/50
3285/3285 [==============================] - 3s 809us/step - loss: 0.2039 - acc: 0.9528 - val_loss: 0.1494 - val_acc: 0.9668
Epoch 7/50
3285/3285 [==============================] - 3s 789us/step - loss: 0.2005 - acc: 0.9440 - val_loss: 0.1628 - val_acc: 0.9625
Epoch 8/50
3285/3285 [==============================] - 3s 783us/step - loss: 0.2084 - acc: 0.9470 - val_loss: 0.1412 - val_acc: 0.9690
Epoch 9/50
3285/3285 [==============================] - 3s 804us/step - loss: 0.2095 - acc: 0.9516 - val_loss: 0.1473 - val_acc: 0.9697
Epoch 10/50
3285/3285 [==============================] - 3s 784us/step - loss: 0.2003 - acc: 0.9513 - val_loss: 0.1707 - val_acc: 0.9690
Epoch 11/50
3285/3285 [==============================] - 3s 798us/step - loss: 0.1752 - acc: 0.9580 - val_loss: 0.1792 - val_acc: 0.9640
Epoch 12/50
3285/3285 [==============================] - 3s 780us/step - loss: 0.1962 - acc: 0.9549 - val_loss: 0.1443 - val_acc: 0.9690
Epoch 13/50
3285/3285 [==============================] - 3s 812us/step - loss: 0.2125 - acc: 0.9537 - val_loss: 0.2340 - val_acc: 0.9625
Epoch 14/50
3285/3285 [==============================] - 3s 793us/step - loss: 0.1583 - acc: 0.9598 - val_loss: 0.2017 - val_acc: 0.9661
```

```
3285/3285 [==============================] - 3s 793us/step - loss: 0.1585 - acc: 0.9598 - val_loss: 0.2017 - val_acc: 0.9661
Epoch 15/50
3285/3285 [==============================] - 3s 783us/step - loss: 0.1709 - acc: 0.9647 - val_loss: 0.1951 - val_acc: 0.9726
Epoch 16/50
3285/3285 [==============================] - 3s 805us/step - loss: 0.1303 - acc: 0.9674 - val_loss: 0.2214 - val_acc: 0.9654
Epoch 17/50
3285/3285 [==============================] - 3s 817us/step - loss: 0.2010 - acc: 0.9528 - val_loss: 0.2585 - val_acc: 0.9640
Epoch 18/50
3285/3285 [==============================] - 3s 818us/step - loss: 0.1694 - acc: 0.9647 - val_loss: 0.2897 - val_acc: 0.9510
Epoch 19/50
3285/3285 [==============================] - 3s 821us/step - loss: 0.1643 - acc: 0.9665 - val_loss: 0.4056 - val_acc: 0.9286
Epoch 20/50
3285/3285 [==============================] - 3s 818us/step - loss: 0.2001 - acc: 0.9543 - val_loss: 0.3493 - val_acc: 0.9452
Epoch 21/50
3285/3285 [==============================] - 3s 812us/step - loss: 0.1556 - acc: 0.9656 - val_loss: 0.2541 - val_acc: 0.9697
Epoch 22/50
3285/3285 [==============================] - 3s 798us/step - loss: 0.1520 - acc: 0.9686 - val_loss: 0.2686 - val_acc: 0.9668
Epoch 23/50
3285/3285 [==============================] - 3s 787us/step - loss: 0.1669 - acc: 0.9632 - val_loss: 0.2475 - val_acc: 0.9683
Epoch 24/50
3285/3285 [==============================] - 3s 791us/step - loss: 0.1708 - acc: 0.9616 - val_loss: 0.2433 - val_acc: 0.9690
Epoch 25/50
3285/3285 [==============================] - 3s 796us/step - loss: 0.1600 - acc: 0.9686 - val_loss: 0.2373 - val_acc: 0.9712
Epoch 26/50
3285/3285 [==============================] - 3s 789us/step - loss: 0.1810 - acc: 0.9586 - val_loss: 0.3030 - val_acc: 0.9510
Epoch 27/50
3285/3285 [==============================] - 3s 787us/step - loss: 0.1833 - acc: 0.9623 - val_loss: 0.2875 - val_acc: 0.9575
Epoch 28/50
3285/3285 [==============================] - 3s 793us/step - loss: 0.1666 - acc: 0.9589 - val_loss: 0.2197 - val_acc: 0.9697
Epoch 29/50
3285/3285 [==============================] - 3s 803us/step - loss: 0.1933 - acc: 0.9586 - val_loss: 0.2960 - val_acc: 0.9640
```

Out[0]:

```
<keras.callbacks.History at 0x7f7c353bd978>
```

In [0]:

```python
dynamic_loss, dynamic_acc = dynamic_model.evaluate(X_test_dynamic, y_test_dynamic)
```

```
1387/1387 [==============================] - 0s 91us/step
```

In [0]:

```python
print('Loss : {} | Accuracy : {} %'.format(dynamic_loss, dynamic_acc*100.))
```

```
Loss : 0.2959768250621533 | Accuracy : 96.39509733237203 %
```

In [0]:

In [0]:

```
confusion_matrix_bin(y_test_dynamic, dynamic_model.predict(X_test_dynamic), { 0:'WALKING', 1:'WALKING_UPSTAIRS', 2:'WALKING_DOWNSTAIRS'})
```

Out[0]:

Pred	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
True			
WALKING	482	5	9
WALKING_DOWNSTAIRS	9	405	6
WALKING_UPSTAIRS	2	24	445

In [0]:

```
dynamic_model.save('Dynamic_class_model.hdf5')
```

Static class model

In [0]:

```
# Model - 2
static_input = Input(shape=(timesteps, input_dim), name='static_input')

z = Conv1D(filters=32, kernel_size=3, activation='relu', kernel_regularizer=l2(.001))(static_input)
z = Conv1D(filters=64, kernel_size=3, activation='relu', kernel_regularizer=l2(.1))(z)
z = Conv1D(filters=128, kernel_size=3, activation='relu', kernel_regularizer=l2(.001))(z)
z = MaxPooling1D(pool_size=2)(z)
z = Flatten()(z)
z = Dropout(rate=.6)(z)
z = Dense(32, activation='relu', kernel_regularizer=l2())(z)
z = BatchNormalization()(z)
z = Dropout(.7)(z)
static_output = Dense(3, activation='softmax', name='static_out')(z)

static_model = Model(inputs=static_input, outputs=static_output)

static_model.compile(loss='categorical_crossentropy', optimizer=keras.optimizers.rmsprop(.001), metrics=['accuracy'])
```

In [0]:

```
static_model.fit(X_train_static,
              y_train_static,
              batch_size=8,
              validation_data=(X_test_static, y_test_static),
              epochs=100,
              callbacks=[EarlyStopping(monitor='val_acc', patience=10), ReduceLROnPlateau(monitor='val_acc', factor=0.2, patience=5, min_lr=.001)])
```

```
          callbacks=[EarlyStopping(monitor='val_acc', patience=10), ReduceLROnPlateau(monitor='val_acc', factor=0.2, patience=5, min_lr=.001)])
```

```
Train on 4067 samples, validate on 1560 samples
Epoch 1/100
4067/4067 [==============================] - 7s 2ms/step - loss: 0.6775 - acc: 0.8190 - val_loss: 0.6473 - val_acc: 0.8423
Epoch 2/100
4067/4067 [==============================] - 3s 828us/step - loss: 0.6001 - acc: 0.8394 - val_loss: 0.4327 - val_acc: 0.8731
Epoch 3/100
4067/4067 [==============================] - 3s 848us/step - loss: 0.5590 - acc: 0.8591 - val_loss: 0.5350 - val_acc: 0.8397
Epoch 4/100
4067/4067 [==============================] - 3s 832us/step - loss: 0.6069 - acc: 0.8439 - val_loss: 0.4331 - val_acc: 0.8833
Epoch 5/100
4067/4067 [==============================] - 3s 827us/step - loss: 0.5809 - acc: 0.8522 - val_loss: 0.5184 - val_acc: 0.8718
Epoch 6/100
4067/4067 [==============================] - 3s 838us/step - loss: 0.5679 - acc: 0.8596 - val_loss: 0.4533 - val_acc: 0.8750
Epoch 7/100
4067/4067 [==============================] - 3s 830us/step - loss: 0.6576 - acc: 0.8289 - val_loss: 0.4280 - val_acc: 0.8782
Epoch 8/100
4067/4067 [==============================] - 3s 827us/step - loss: 0.6018 - acc: 0.8542 - val_loss: 0.3973 - val_acc: 0.8795
Epoch 9/100
4067/4067 [==============================] - 3s 819us/step - loss: 0.5465 - acc: 0.8640 - val_loss: 0.3853 - val_acc: 0.8744
Epoch 10/100
4067/4067 [==============================] - 3s 836us/step - loss: 0.4603 - acc: 0.8812 - val_loss: 0.3426 - val_acc: 0.8878
Epoch 11/100
4067/4067 [==============================] - 3s 855us/step - loss: 0.3887 - acc: 0.8901 - val_loss: 0.2883 - val_acc: 0.9006
Epoch 12/100
4067/4067 [==============================] - 3s 823us/step - loss: 0.4278 - acc: 0.8837 - val_loss: 0.3677 - val_acc: 0.8769
Epoch 13/100
4067/4067 [==============================] - 3s 821us/step - loss: 0.3768 - acc: 0.8945 - val_loss: 0.3443 - val_acc: 0.8891
Epoch 14/100
4067/4067 [==============================] - 3s 833us/step - loss: 0.4090 - acc: 0.8903 - val_loss: 0.3097 - val_acc: 0.9038
Epoch 15/100
4067/4067 [==============================] - 3s 851us/step - loss: 0.3586 - acc: 0.8928 - val_loss: 0.2868 - val_acc: 0.9115
Epoch 16/100
4067/4067 [==============================] - 3s 839us/step - loss: 0.3436 - acc: 0.9051 - val_loss: 0.2872 - val_acc: 0.8929
Epoch 17/100
4067/4067 [==============================] - 3s 829us/step - loss: 0.4000 - acc: 0.8965 - val_loss: 0.2715 - val_acc: 0.9071
Epoch 18/100
4067/4067 [==============================] - 3s 821us/step - loss: 0.3699 - acc: 0.9066 - val_loss: 0.3236 - val_acc: 0.8756
Epoch 19/100
4067/4067 [==============================] - 3s 854us/step - loss: 0.3494 - acc: 0.9048 - val_loss: 0.2486 - val_acc: 0.9186
Epoch 20/100
4067/4067 [==============================] - 3s 818us/step - loss: 0.3097 - acc: 0.9139 - val_loss: 0.3031 - val_acc: 0.9000
Epoch 21/100
4067/4067 [==============================] - 3s 830us/step - loss: 0.3285 - acc: 0.9171 - val_loss: 0.2414 - val_acc: 0.9212
Epoch 22/100
4067/4067 [==============================] - 3s 852us/step - loss: 0.3598 - acc: 0.9147 - val_loss: 0.3131 - val_acc: 0.8962
Epoch 23/100
4067/4067 [==============================] - 3s 822us/step - loss: 0.3422 - acc: 0.9098 - val_loss: 0.2341 - val_acc: 0.9147
Epoch 24/100
4067/4067 [==============================] - 3s 833us/step - loss: 0.3300 - acc: 0.9105 - val_loss: 0.2997 - val_acc: 0.9096
Epoch 25/100
4067/4067 [==============================] - 3s 825us/step - loss: 0.3333 - acc: 0.9147 - val_loss: 0.4211 - val_acc: 0.8891
```

```
Epoch 26/100
4067/4067 [==============================] - 3s 832us/step - loss: 0.3374 - acc: 0.9144 - val_loss: 0.2650 - val_acc: 0.9179
Epoch 27/100
4067/4067 [==============================] - 3s 847us/step - loss: 0.3347 - acc: 0.9171 - val_loss: 0.2659 - val_acc: 0.9250
Epoch 28/100
4067/4067 [==============================] - 3s 859us/step - loss: 0.2788 - acc: 0.9299 - val_loss: 0.3297 - val_acc: 0.8981
Epoch 29/100
4067/4067 [==============================] - 3s 818us/step - loss: 0.3128 - acc: 0.9223 - val_loss: 0.2863 - val_acc: 0.9173
Epoch 30/100
4067/4067 [==============================] - 3s 809us/step - loss: 0.2784 - acc: 0.9329 - val_loss: 0.2427 - val_acc: 0.9244
Epoch 31/100
4067/4067 [==============================] - 3s 847us/step - loss: 0.3202 - acc: 0.9225 - val_loss: 0.2444 - val_acc: 0.9218
Epoch 32/100
4067/4067 [==============================] - 3s 821us/step - loss: 0.2459 - acc: 0.9334 - val_loss: 0.2705 - val_acc: 0.9205
Epoch 33/100
4067/4067 [==============================] - 3s 830us/step - loss: 0.3265 - acc: 0.9223 - val_loss: 0.2791 - val_acc: 0.9250
Epoch 34/100
4067/4067 [==============================] - 3s 848us/step - loss: 0.2761 - acc: 0.9294 - val_loss: 0.2870 - val_acc: 0.9231
Epoch 35/100
4067/4067 [==============================] - 3s 852us/step - loss: 0.2685 - acc: 0.9243 - val_loss: 0.2538 - val_acc: 0.9199
Epoch 36/100
4067/4067 [==============================] - 3s 835us/step - loss: 0.2763 - acc: 0.9334 - val_loss: 0.3314 - val_acc: 0.9013
Epoch 37/100
4067/4067 [==============================] - 3s 830us/step - loss: 0.2677 - acc: 0.9329 - val_loss: 0.2328 - val_acc: 0.9237
```

Out[0]:

```
<keras.callbacks.History at 0x7f7c3855d240>
```

In [0]:

```
static_loss, static_acc = static_model.evaluate(X_test_static, y_test_static)
```

```
1560/1560 [==============================] - 0s 92us/step
```

In [0]:

```
print('Loss : {} | Accuracy : {} %'.format(static_loss, static_acc*100.))
```

```
Loss : 0.23280242435061016 | Accuracy : 92.37179487179488 %
```

In [0]:

```
confusion_matrix_bin(y_test_static, static_model.predict(X_test_static), { 0:'SITTING', 1:'STANDING', 2:'LAYING'})
```

Out[0]:

Pred	LAYING	SITTING	STANDING
True			
LAYING	537	0	0
SITTING	0	402	89
STANDING	0	30	502

In [0]:

```
static_model.save('Static_class_model.hdf5')
```

Final Prediction

In [0]:

```
t = np.zeros((10), dtype='int')

t[[2,6,8]] = [23,54,9]

t
```

Out[0]:

```
array([ 0,  0, 23,  0,  0,  0, 54,  0,  9,  0])
```

In [0]:

```
from keras.models import load_model
from scipy.ndimage import gaussian_filter

class PredictActivity:
  def __init__(self):
    self.binary_model = None
    self.dynamic_model = None
    self.static_model = None

  def loadModels(self, binModelPath, dynamicModelpath, staticModelPath):
    self.binary_model = load_model(binModelPath)
    self.dynamic_model = load_model(dynamicModelpath)
    self.static_model = load_model(staticModelPath)

  def predict(self, X):
    y_bin = np.argmax(self.binary_model.predict(X), axis=1)

    X_dynamic = X[y_bin==1]
    X_static = X[y_bin==0]
```

```python
    # X_dynamic = X_dynamic + .007*(X_dynamic - gaussian_filter(X_dynamic, sigma=8))
    # X_static = X_static + .007*(X_static - gaussian_filter(X_static, sigma=8))

    y_dynamic = np.argmax(self.dynamic_model.predict(X_dynamic), axis=1)
    y_static = np.argmax(self.static_model.predict(X_static), axis=1)

    y_dynamic = y_dynamic + 1
    y_static = y_static + 4

    output = np.zeros((X.shape[0]), dtype='int')

    output[np.where(y_bin==1)[0]] = y_dynamic

    output[np.where(y_bin==0)[0]] = y_static

    return output
```

In [0]:

```python
predictactivity = PredictActivity()

predictactivity.loadModels('./Base_Binary_model.hdf5', './Dynamic_class_model.hdf5', './Static_class_model.hdf5')
```

In [0]:

```python
accuracy_score(y_test, predictactivity.predict(X_test))
```

Out[0]:

```
0.9416355615880556
```

In [0]:

```python
print('Accuracy : {} %'.format(round(accuracy_score(y_test, predictactivity.predict(X_test))*100., 1)))
```

```
Accuracy : 94.2 %
```

In [0]:

```python
pt.add_row(['Divide and Conquer(CNNs)', '-', '94.2 %'])
```

In [100]:

```python
print(pt)
```

```
+--------------------------+-------+---------------+
```

Model	Loss	Test Accuracy
2 Layers LSTM	0.423	90.024 %
2 Layers 1D Convolutions	0.344	92.297 %
Divide and Conquer(CNNs)	-	94.2 %