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 = '{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 = 'y_{subset}.txt'
    y = _read_csv(filename)[0]

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

In [8]:

```
signals_data = []

filename = ["body_acc_x_train.txt",
    "body_acc_y_train.txt",
    "body_acc_z_train.txt",
    "body_gyro_x_train.txt",
    "body_gyro_y_train.txt",
    "body_gyro_z_train.txt",
    "total_acc_x_train.txt",
    "total_acc_y_train.txt",
    "total_acc_z_train.txt"]
for files in filename:
    signals_data.append( _read_csv(files).as_matrix())


X_train = np.transpose(signals_data, (1, 2, 0))
```

```
/home/shanud6711/.local/lib/python3.5/site-packages/ipykernel_launcher.py:1
3: FutureWarning: Method .as_matrix will be removed in a future version. Use
.values instead.
  del sys.path[0]
```

In [9]:

```
X_train.shape
```

Out[9]:

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

In [10]:

```
signals_data_test = []
filename = ["body_acc_x_test.txt",
    "body_acc_y_test.txt",
    "body_acc_z_test.txt",
    "body_gyro_x_test.txt",
    "body_gyro_y_test.txt",
    "body_gyro_z_test.txt",
    "total_acc_x_test.txt",
    "total_acc_y_test.txt",
    "total_acc_z_test.txt"]
for files in filename:
    signals_data_test.append( _read_csv(files).as_matrix())


X_test = np.transpose(signals_data_test,(1,2,0))
```

```
/home/shanud6711/.local/lib/python3.5/site-packages/ipykernel_launcher.py:1
2: FutureWarning: Method .as_matrix will be removed in a future version. Use
.values instead.
  if sys.path[0] == '':
```

In [11]:

```
X_test.shape
```

Out[11]:

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

In [12]:

```
y = _read_csv('y_train.txt')[0]
Y_train = pd.get_dummies(y).as_matrix()
```

```
/home/shanud6711/.local/lib/python3.5/site-packages/ipykernel_launcher.py:2:
FutureWarning: Method .as_matrix will be removed in a future version. Use .v
alues instead.
```

In [13]:

```
y_ = _read_csv('y_test.txt')[0]
Y_test = pd.get_dummies(y_).as_matrix()
```

```
/home/shanud6711/.local/lib/python3.5/site-packages/ipykernel_launcher.py:2:
FutureWarning: Method .as_matrix will be removed in a future version. Use .v
alues instead.
```

In [14]:

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

```
/home/shanud6711/.local/lib/python3.5/site-packages/tensorflow/python/framew
ork/dtypes.py:516: FutureWarning: Passing (type, 1) or '1type' as a synonym
of type is deprecated; in a future version of numpy, it will be understood a
s (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/home/shanud6711/.local/lib/python3.5/site-packages/tensorflow/python/framew
ork/dtypes.py:517: FutureWarning: Passing (type, 1) or '1type' as a synonym
of type is deprecated; in a future version of numpy, it will be understood a
s (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/home/shanud6711/.local/lib/python3.5/site-packages/tensorflow/python/framew
ork/dtypes.py:518: FutureWarning: Passing (type, 1) or '1type' as a synonym
of type is deprecated; in a future version of numpy, it will be understood a
s (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/home/shanud6711/.local/lib/python3.5/site-packages/tensorflow/python/framew
ork/dtypes.py:519: FutureWarning: Passing (type, 1) or '1type' as a synonym
of type is deprecated; in a future version of numpy, it will be understood a
s (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/home/shanud6711/.local/lib/python3.5/site-packages/tensorflow/python/framew
ork/dtypes.py:520: FutureWarning: Passing (type, 1) or '1type' as a synonym
of type is deprecated; in a future version of numpy, it will be understood a
s (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/home/shanud6711/.local/lib/python3.5/site-packages/tensorflow/python/framew
ork/dtypes.py:525: FutureWarning: Passing (type, 1) or '1type' as a synonym
of type is deprecated; in a future version of numpy, it will be understood a
s (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
/home/shanud6711/.local/lib/python3.5/site-packages/tensorboard/compat/tenso
rflow_stub/dtypes.py:541: FutureWarning: Passing (type, 1) or '1type' as a s
ynonym of type is deprecated; in a future version of numpy, it will be under
stood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/home/shanud6711/.local/lib/python3.5/site-packages/tensorboard/compat/tenso
rflow_stub/dtypes.py:542: FutureWarning: Passing (type, 1) or '1type' as a s
ynonym of type is deprecated; in a future version of numpy, it will be under
stood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/home/shanud6711/.local/lib/python3.5/site-packages/tensorboard/compat/tenso
rflow_stub/dtypes.py:543: FutureWarning: Passing (type, 1) or '1type' as a s
ynonym of type is deprecated; in a future version of numpy, it will be under
stood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/home/shanud6711/.local/lib/python3.5/site-packages/tensorboard/compat/tenso
rflow_stub/dtypes.py:544: FutureWarning: Passing (type, 1) or '1type' as a s
ynonym of type is deprecated; in a future version of numpy, it will be under
stood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/home/shanud6711/.local/lib/python3.5/site-packages/tensorboard/compat/tenso
rflow_stub/dtypes.py:545: FutureWarning: Passing (type, 1) or '1type' as a s
ynonym of type is deprecated; in a future version of numpy, it will be under
```

```
stood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/home/shanud6711/.local/lib/python3.5/site-packages/tensorboard/compat/tenso
rflow_stub/dtypes.py:550: FutureWarning: Passing (type, 1) or '1type' as a s
ynonym of type is deprecated; in a future version of numpy, it will be under
stood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

In [15]:

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

In [16]:

```
# 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 [17]:

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

In [18]:

```
# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 32
```

In [19]:

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

In [20]:

```
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 [21]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 32)                5376
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 198
=================================================================
Total params: 5,574
Trainable params: 5,574
Non-trainable params: 0
_________________________________________________________________
```

In [22]:

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

In [23]:

```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
```

```
WARNING:tensorflow:From /home/shanud6711/.local/lib/python3.5/site-packages/
tensorflow/python/ops/math_grad.py:1250: add_dispatch_support.<locals>.wrapp
er (from tensorflow.python.ops.array_ops) is deprecated and will be removed
in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /home/shanud6711/.local/lib/python3.5/site-packages/
keras/backend/tensorflow_backend.py:422: The name tf.global_variables is dep
recated. Please use tf.compat.v1.global_variables instead.

Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 19s 3ms/step - loss: 1.3008 - a
ccuracy: 0.4645 - val_loss: 1.0891 - val_accuracy: 0.5565
Epoch 2/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.9056 - a
ccuracy: 0.6171 - val_loss: 0.8277 - val_accuracy: 0.5942
Epoch 3/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.7434 - a
ccuracy: 0.6549 - val_loss: 0.7569 - val_accuracy: 0.6230
Epoch 4/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.6725 - a
ccuracy: 0.6800 - val_loss: 0.6941 - val_accuracy: 0.6651
Epoch 5/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.6236 - a
ccuracy: 0.7116 - val_loss: 0.6568 - val_accuracy: 0.7326
Epoch 6/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.5866 - a
ccuracy: 0.7333 - val_loss: 0.7696 - val_accuracy: 0.6763
Epoch 7/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.5055 - a
ccuracy: 0.7748 - val_loss: 0.6162 - val_accuracy: 0.7272
Epoch 8/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.4769 - a
ccuracy: 0.7791 - val_loss: 0.5323 - val_accuracy: 0.7465
Epoch 9/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.4243 - a
ccuracy: 0.7979 - val_loss: 0.6893 - val_accuracy: 0.7167
Epoch 10/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.4033 - a
ccuracy: 0.8096 - val_loss: 0.5631 - val_accuracy: 0.7326
Epoch 11/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.3785 - a
ccuracy: 0.8391 - val_loss: 0.5226 - val_accuracy: 0.7937
Epoch 12/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.3327 - a
ccuracy: 0.8791 - val_loss: 0.5721 - val_accuracy: 0.8694
Epoch 13/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.2857 - a
ccuracy: 0.9132 - val_loss: 0.4536 - val_accuracy: 0.8812
Epoch 14/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.2537 - a
```

```
ccuracy: 0.9207 - val_loss: 0.5414 - val_accuracy: 0.8748
Epoch 15/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.2421 - a
ccuracy: 0.9293 - val_loss: 0.4205 - val_accuracy: 0.8968
Epoch 16/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.2077 - a
ccuracy: 0.9331 - val_loss: 0.4868 - val_accuracy: 0.8785
Epoch 17/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1950 - a
ccuracy: 0.9384 - val_loss: 0.5625 - val_accuracy: 0.8833
Epoch 18/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1844 - a
ccuracy: 0.9419 - val_loss: 0.6079 - val_accuracy: 0.8738
Epoch 19/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1846 - a
ccuracy: 0.9414 - val_loss: 0.4497 - val_accuracy: 0.8999
Epoch 20/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1701 - a
ccuracy: 0.9459 - val_loss: 0.5215 - val_accuracy: 0.8795
Epoch 21/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1723 - a
ccuracy: 0.9450 - val_loss: 0.4698 - val_accuracy: 0.8887
Epoch 22/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1813 - a
ccuracy: 0.9448 - val_loss: 0.4783 - val_accuracy: 0.8795
Epoch 23/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1579 - a
ccuracy: 0.9465 - val_loss: 0.4126 - val_accuracy: 0.8968
Epoch 24/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1754 - a
ccuracy: 0.9448 - val_loss: 0.3679 - val_accuracy: 0.9111
Epoch 25/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1827 - a
ccuracy: 0.9425 - val_loss: 0.9810 - val_accuracy: 0.8426
Epoch 26/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1565 - a
ccuracy: 0.9489 - val_loss: 0.3639 - val_accuracy: 0.9043
Epoch 27/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1508 - a
ccuracy: 0.9484 - val_loss: 0.3858 - val_accuracy: 0.8996
Epoch 28/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1461 - a
ccuracy: 0.9470 - val_loss: 0.4374 - val_accuracy: 0.9013
Epoch 29/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1558 - a
ccuracy: 0.9449 - val_loss: 0.3596 - val_accuracy: 0.9043
Epoch 30/30
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1712 - a
ccuracy: 0.9433 - val_loss: 0.4166 - val_accuracy: 0.8955
```

Out[23]:

```
<keras.callbacks.callbacks.History at 0x7fd3b86acc88>
```

In [24]:

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

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS
\
True
LAYING                 510        0        27        0                   0
SITTING                  2      381       105        1                   1
STANDING                 0       86       446        0                   0
WALKING                  0        0         0      454                  15
WALKING_DOWNSTAIRS       0        0         0        0                 419
WALKING_UPSTAIRS         0        7         0        4                  31

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            1
STANDING                           0
WALKING                           27
WALKING_DOWNSTAIRS                 1
WALKING_UPSTAIRS                 429
```

In [28]:

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

```
2947/2947 [==============================] - 1s 307us/step
```

In [29]:

```
score
```

Out[29]:

```
[0.41655886096877154, 0.8954869508743286]
```

- With a simple 2 layer architecture we got 89.54% accuracy and a loss of 0.416
- We can further imporve the performace with Hyperparameter tuning

hyperparameter tuning LSTM Model

model 1

In [30]:

```
from keras.layers.normalization import BatchNormalization

model1 = Sequential()
# Configuring the parameters
model1.add(LSTM(64, input_shape=(timesteps, input_dim)))
model1.add(BatchNormalization())
# Adding a dropout layer
model1.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model1.add(Dense(n_classes, activation='sigmoid'))
model1.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 64)                18944
_________________________________________________________________
batch_normalization_2 (Batch (None, 64)                256
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 390
=================================================================
Total params: 19,590
Trainable params: 19,462
Non-trainable params: 128
_________________________________________________________________
```

In [31]:

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

In [32]:

```
model1.fit(X_train,
          Y_train,
          batch_size=16,
          validation_data=(X_test, Y_test),
          epochs=30)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 25s 3ms/step - loss: 1.1227 - a
ccuracy: 0.5359 - val_loss: 0.9233 - val_accuracy: 0.6328
Epoch 2/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.8160 - a
ccuracy: 0.6487 - val_loss: 0.7654 - val_accuracy: 0.6871
Epoch 3/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.7554 - a
ccuracy: 0.6662 - val_loss: 0.7253 - val_accuracy: 0.6804
Epoch 4/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.6246 - a
ccuracy: 0.7371 - val_loss: 0.5614 - val_accuracy: 0.8062
Epoch 5/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.4200 - a
ccuracy: 0.8757 - val_loss: 0.3841 - val_accuracy: 0.8843
Epoch 6/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.3393 - a
ccuracy: 0.8925 - val_loss: 0.3294 - val_accuracy: 0.8856
Epoch 7/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.3520 - a
ccuracy: 0.8777 - val_loss: 0.3121 - val_accuracy: 0.8938
Epoch 8/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.3676 - a
ccuracy: 0.8837 - val_loss: 0.5714 - val_accuracy: 0.8551
Epoch 9/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.3391 - a
ccuracy: 0.8917 - val_loss: 0.3430 - val_accuracy: 0.9013
Epoch 10/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.1988 - a
ccuracy: 0.9313 - val_loss: 0.3869 - val_accuracy: 0.9006
Epoch 11/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1775 - a
ccuracy: 0.9384 - val_loss: 0.3957 - val_accuracy: 0.9040
Epoch 12/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.3241 - a
ccuracy: 0.8945 - val_loss: 0.4345 - val_accuracy: 0.8663
Epoch 13/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.3077 - a
ccuracy: 0.8979 - val_loss: 0.2863 - val_accuracy: 0.9087
Epoch 14/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.2005 - a
ccuracy: 0.9290 - val_loss: 0.3125 - val_accuracy: 0.8972
Epoch 15/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.2611 - a
ccuracy: 0.9159 - val_loss: 0.2801 - val_accuracy: 0.9114
Epoch 16/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.1664 - a
ccuracy: 0.9402 - val_loss: 0.3583 - val_accuracy: 0.9013
Epoch 17/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.1681 - a
ccuracy: 0.9407 - val_loss: 0.2866 - val_accuracy: 0.9223
Epoch 18/30
```

```
7352/7352 [==============================] - 24s 3ms/step - loss: 0.1709 - a
ccuracy: 0.9361 - val_loss: 0.3174 - val_accuracy: 0.9199
Epoch 19/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.1852 - a
ccuracy: 0.9348 - val_loss: 0.3717 - val_accuracy: 0.9026
Epoch 20/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.1833 - a
ccuracy: 0.9340 - val_loss: 0.4566 - val_accuracy: 0.8426
Epoch 21/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.2605 - a
ccuracy: 0.9158 - val_loss: 0.3185 - val_accuracy: 0.9084
Epoch 22/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.2202 - a
ccuracy: 0.9233 - val_loss: 0.4125 - val_accuracy: 0.8985
Epoch 23/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.2080 - a
ccuracy: 0.9283 - val_loss: 0.2724 - val_accuracy: 0.9162
Epoch 24/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.1731 - a
ccuracy: 0.9358 - val_loss: 0.2522 - val_accuracy: 0.9230
Epoch 25/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.1590 - a
ccuracy: 0.9399 - val_loss: 0.2070 - val_accuracy: 0.9240
Epoch 26/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.1934 - a
ccuracy: 0.9274 - val_loss: 0.2364 - val_accuracy: 0.9084
Epoch 27/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.1454 - a
ccuracy: 0.9412 - val_loss: 0.2487 - val_accuracy: 0.9158
Epoch 28/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.1742 - a
ccuracy: 0.9295 - val_loss: 0.2555 - val_accuracy: 0.9125
Epoch 29/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.1569 - a
ccuracy: 0.9392 - val_loss: 0.2654 - val_accuracy: 0.9213
Epoch 30/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.1484 - a
ccuracy: 0.9453 - val_loss: 0.2650 - val_accuracy: 0.9264
```

Out[32]:

```
<keras.callbacks.callbacks.History at 0x7fd3a4d7dda0>
```

In [33]:

```
# Confusion Matrix
print(confusion_matrix(Y_test, model1.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS
\
True
LAYING                 537        0         0        0                   0
SITTING                  0      407        77        0                   0
STANDING                 0       90       442        0                   0
WALKING                  0        0         0      469                  25
WALKING_DOWNSTAIRS       0        0         0        2                 418
WALKING_UPSTAIRS         0        0         0        9                   5

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            7
STANDING                           0
WALKING                            2
WALKING_DOWNSTAIRS                 0
WALKING_UPSTAIRS                 457
```

In [34]:

```
score1 = model1.evaluate(X_test, Y_test)
score1
```

```
2947/2947 [==============================] - 1s 416us/step
```

Out[34]:

```
[0.26511486830740727, 0.9263657927513123]
```

adding hidden layer

In [45]:

```
from keras.layers.normalization import BatchNormalization
# Initiliazing the sequential model
model2 = Sequential()
# Configuring the parameters
model2.add(LSTM(64, return_sequences=True, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model2.add(BatchNormalization())
model2.add(Dropout(0.2))

model2.add(LSTM(64))
model2.add(BatchNormalization())
# Adding a dropout layer
model2.add(Dropout(0.2))
# Adding a dense output layer with sigmoid activation
model2.add(Dense(n_classes, activation='sigmoid'))
model2.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_10 (LSTM)               (None, 128, 64)           18944
_________________________________________________________________
batch_normalization_6 (Batch (None, 128, 64)           256
_________________________________________________________________
dropout_10 (Dropout)         (None, 128, 64)           0
_________________________________________________________________
lstm_11 (LSTM)               (None, 64)                33024
_________________________________________________________________
batch_normalization_7 (Batch (None, 64)                256
_________________________________________________________________
dropout_11 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_7 (Dense)              (None, 6)                 390
=================================================================
Total params: 52,870
Trainable params: 52,614
Non-trainable params: 256
_________________________________________________________________
```

In [46]:

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

In [47]:

```
model2.fit(X_train,
          Y_train,
          batch_size=16,
          validation_data=(X_test, Y_test),
          epochs=30)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 57s 8ms/step - loss: 0.8074 - a
ccuracy: 0.6692 - val_loss: 0.8367 - val_accuracy: 0.6580
Epoch 2/30
7352/7352 [==============================] - 55s 8ms/step - loss: 0.6483 - a
ccuracy: 0.7025 - val_loss: 0.6003 - val_accuracy: 0.7418
Epoch 3/30
7352/7352 [==============================] - 56s 8ms/step - loss: 0.5080 - a
ccuracy: 0.7983 - val_loss: 0.4321 - val_accuracy: 0.8612
Epoch 4/30
7352/7352 [==============================] - 56s 8ms/step - loss: 0.3615 - a
ccuracy: 0.8886 - val_loss: 0.3008 - val_accuracy: 0.8972
Epoch 5/30
7352/7352 [==============================] - 56s 8ms/step - loss: 0.2293 - a
ccuracy: 0.9297 - val_loss: 0.2766 - val_accuracy: 0.9080
Epoch 6/30
7352/7352 [==============================] - 56s 8ms/step - loss: 0.1940 - a
ccuracy: 0.9324 - val_loss: 0.2366 - val_accuracy: 0.9148
Epoch 7/30
7352/7352 [==============================] - 56s 8ms/step - loss: 0.1744 - a
ccuracy: 0.9370 - val_loss: 0.2485 - val_accuracy: 0.9141
Epoch 8/30
7352/7352 [==============================] - 55s 8ms/step - loss: 0.1633 - a
ccuracy: 0.9388 - val_loss: 0.2392 - val_accuracy: 0.9121
Epoch 9/30
7352/7352 [==============================] - 56s 8ms/step - loss: 0.1589 - a
ccuracy: 0.9410 - val_loss: 0.2305 - val_accuracy: 0.9267
Epoch 10/30
7352/7352 [==============================] - 56s 8ms/step - loss: 0.1655 - a
ccuracy: 0.9411 - val_loss: 0.2021 - val_accuracy: 0.9131
Epoch 11/30
7352/7352 [==============================] - 56s 8ms/step - loss: 0.1387 - a
ccuracy: 0.9471 - val_loss: 0.2198 - val_accuracy: 0.9192
Epoch 12/30
7352/7352 [==============================] - 56s 8ms/step - loss: 0.1371 - a
ccuracy: 0.9483 - val_loss: 0.2253 - val_accuracy: 0.9209
Epoch 13/30
7352/7352 [==============================] - 56s 8ms/step - loss: 0.1402 - a
ccuracy: 0.9484 - val_loss: 0.2175 - val_accuracy: 0.9311
Epoch 14/30
7352/7352 [==============================] - 56s 8ms/step - loss: 0.1416 - a
ccuracy: 0.9453 - val_loss: 0.2329 - val_accuracy: 0.9206
Epoch 15/30
7352/7352 [==============================] - 55s 8ms/step - loss: 0.1357 - a
ccuracy: 0.9493 - val_loss: 0.2560 - val_accuracy: 0.9087
Epoch 16/30
7352/7352 [==============================] - 55s 8ms/step - loss: 0.1390 - a
ccuracy: 0.9464 - val_loss: 0.1996 - val_accuracy: 0.9257
Epoch 17/30
7352/7352 [==============================] - 55s 8ms/step - loss: 0.1351 - a
ccuracy: 0.9499 - val_loss: 0.2467 - val_accuracy: 0.9121
Epoch 18/30
```

```
7352/7352 [==============================] - 55s 8ms/step - loss: 0.1319 - a
ccuracy: 0.9483 - val_loss: 0.2088 - val_accuracy: 0.9220
Epoch 19/30
7352/7352 [==============================] - 55s 7ms/step - loss: 0.1291 - a
ccuracy: 0.9484 - val_loss: 0.2085 - val_accuracy: 0.9220
Epoch 20/30
7352/7352 [==============================] - 55s 8ms/step - loss: 0.1311 - a
ccuracy: 0.9461 - val_loss: 0.2183 - val_accuracy: 0.9257
Epoch 21/30
7352/7352 [==============================] - 55s 8ms/step - loss: 0.1290 - a
ccuracy: 0.9487 - val_loss: 0.2282 - val_accuracy: 0.9233
Epoch 22/30
7352/7352 [==============================] - 55s 8ms/step - loss: 0.1307 - a
ccuracy: 0.9487 - val_loss: 0.2218 - val_accuracy: 0.9196
Epoch 23/30
7352/7352 [==============================] - 55s 8ms/step - loss: 0.1341 - a
ccuracy: 0.9445 - val_loss: 0.2142 - val_accuracy: 0.9182
Epoch 24/30
7352/7352 [==============================] - 55s 8ms/step - loss: 0.1311 - a
ccuracy: 0.9480 - val_loss: 0.2125 - val_accuracy: 0.9284
Epoch 25/30
7352/7352 [==============================] - 55s 8ms/step - loss: 0.1264 - a
ccuracy: 0.9512 - val_loss: 0.3688 - val_accuracy: 0.8951
Epoch 26/30
7352/7352 [==============================] - 55s 8ms/step - loss: 0.1573 - a
ccuracy: 0.9411 - val_loss: 0.2283 - val_accuracy: 0.9226
Epoch 27/30
7352/7352 [==============================] - 55s 7ms/step - loss: 0.1243 - a
ccuracy: 0.9472 - val_loss: 0.2044 - val_accuracy: 0.9216
Epoch 28/30
7352/7352 [==============================] - 55s 8ms/step - loss: 0.1192 - a
ccuracy: 0.9521 - val_loss: 0.2352 - val_accuracy: 0.9233
Epoch 29/30
7352/7352 [==============================] - 56s 8ms/step - loss: 0.1271 - a
ccuracy: 0.9494 - val_loss: 0.2112 - val_accuracy: 0.9250
Epoch 30/30
7352/7352 [==============================] - 56s 8ms/step - loss: 0.1268 - a
ccuracy: 0.9497 - val_loss: 0.2354 - val_accuracy: 0.9104
```

Out[47]:

```
<keras.callbacks.callbacks.History at 0x7fd37bfab240>
```

In [48]:

```
# Confusion Matrix
print(confusion_matrix(Y_test, model2.predict(X_test)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS
\
True
LAYING                 537        0         0        0                   0
SITTING                  6      317       161        0                   0
STANDING                 0       45       486        1                   0
WALKING                  0        0         0      471                   5
WALKING_DOWNSTAIRS       0        0         0        0                 416
WALKING_UPSTAIRS         0        2         0        0                  13

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            7
STANDING                           0
WALKING                           20
WALKING_DOWNSTAIRS                 4
WALKING_UPSTAIRS                 456
```

In [49]:

```
score2 = model2.evaluate(X_test, Y_test)
score2
```

```
2947/2947 [==============================] - 3s 982us/step
```

Out[49]:

```
[0.23537055982549487, 0.910417377948761]
```

by changing the hyperparameters we can easily gain higher accuracy of 92.43

In [50]:

```
import pandas as pd
from matplotlib import pyplot
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Dropout
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers.core import Dense, Dropout
from keras.regularizers import l2,l1
import keras
from keras.layers import BatchNormalization
```

In [51]:

```
from sklearn.base import BaseEstimator, TransformerMixin
class scaling_tseries_data(BaseEstimator, TransformerMixin):
    from sklearn.preprocessing import StandardScaler
    def __init__(self):
        self.scale = None

    def transform(self, X):
        temp_X1 = X.reshape((X.shape[0] * X.shape[1], X.shape[2]))
        temp_X1 = self.scale.transform(temp_X1)
        return temp_X1.reshape(X.shape)

    def fit(self, X):
        # remove overlaping
        remove = int(X.shape[1] / 2)
        temp_X = X[:, -remove:, :]
        # flatten data
        temp_X = temp_X.reshape((temp_X.shape[0] * temp_X.shape[1], temp_X.shape[2]))
        scale = StandardScaler()
        scale.fit(temp_X)
        self.scale = scale
        return self
```

In [52]:

```
Scale = scaling_tseries_data()
Scale.fit(X_train)
X_train_sc = Scale.transform(X_train)
X_test_sc = Scale.transform(X_test)
```

In [103]:

```
model = Sequential()
model.add(Conv1D(filters=128, kernel_size=5, activation='relu',kernel_initializer='he_unifo
                 kernel_regularizer=l2(0.01),input_shape=(128,9)))
model.add(Conv1D(filters=32, kernel_size=5, activation='relu',kernel_regularizer=l2(0.06),k
model.add(Dropout(0.3))
model.add(MaxPooling1D(pool_size=3))
model.add(Flatten())
model.add(Dense(16, activation='relu'))
model.add(Dense(6, activation='softmax'))
model.summary()
```

```
Model: "sequential_20"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_25 (Conv1D)           (None, 124, 128)          5888
_________________________________________________________________
conv1d_26 (Conv1D)           (None, 120, 32)           20512
_________________________________________________________________
dropout_24 (Dropout)         (None, 120, 32)           0
_________________________________________________________________
max_pooling1d_13 (MaxPooling (None, 40, 32)            0
_________________________________________________________________
flatten_13 (Flatten)         (None, 1280)              0
_________________________________________________________________
dense_32 (Dense)             (None, 16)                20496
_________________________________________________________________
dense_33 (Dense)             (None, 6)                 102
=================================================================
Total params: 46,998
Trainable params: 46,998
Non-trainable params: 0
_________________________________________________________________
```

In [104]:

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

In [105]:

```
model.fit(X_train_sc,Y_train, epochs=30, batch_size=16,validation_data=(X_test_sc, Y_test),
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 9s 1ms/step - loss: 2.5121 - ac
curacy: 0.8433 - val_loss: 1.0140 - val_accuracy: 0.8487
Epoch 2/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.4978 - ac
curacy: 0.9385 - val_loss: 0.5578 - val_accuracy: 0.8717
Epoch 3/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.3053 - ac
curacy: 0.9369 - val_loss: 0.4350 - val_accuracy: 0.8928
Epoch 4/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.2584 - ac
curacy: 0.9404 - val_loss: 0.3485 - val_accuracy: 0.9050
Epoch 5/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.2301 - ac
curacy: 0.9427 - val_loss: 0.3784 - val_accuracy: 0.8833
Epoch 6/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.2113 - ac
curacy: 0.9449 - val_loss: 0.3349 - val_accuracy: 0.8989
Epoch 7/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1983 - ac
curacy: 0.9470 - val_loss: 0.3507 - val_accuracy: 0.8985
Epoch 8/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1813 - ac
curacy: 0.9504 - val_loss: 0.3130 - val_accuracy: 0.8955
Epoch 9/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1875 - ac
curacy: 0.9449 - val_loss: 0.2937 - val_accuracy: 0.9118
Epoch 10/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1672 - ac
curacy: 0.9531 - val_loss: 0.2825 - val_accuracy: 0.9094
Epoch 11/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1732 - ac
curacy: 0.9531 - val_loss: 0.2862 - val_accuracy: 0.9016
Epoch 12/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1569 - ac
curacy: 0.9557 - val_loss: 0.2625 - val_accuracy: 0.9097
Epoch 13/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1544 - ac
curacy: 0.9538 - val_loss: 0.2553 - val_accuracy: 0.9169
Epoch 14/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1648 - ac
curacy: 0.9512 - val_loss: 0.3326 - val_accuracy: 0.9033
Epoch 15/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1505 - ac
curacy: 0.9548 - val_loss: 0.2807 - val_accuracy: 0.9050
Epoch 16/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1477 - ac
curacy: 0.9548 - val_loss: 0.2768 - val_accuracy: 0.9165
Epoch 17/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1427 - ac
curacy: 0.9559 - val_loss: 0.2882 - val_accuracy: 0.9125
Epoch 18/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1520 - ac
curacy: 0.9550 - val_loss: 0.2644 - val_accuracy: 0.9067
Epoch 19/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1442 - ac
```

```
curacy: 0.9550 - val_loss: 0.3108 - val_accuracy: 0.8975
Epoch 20/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1418 - ac
curacy: 0.9580 - val_loss: 0.2589 - val_accuracy: 0.9230
Epoch 21/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1491 - ac
curacy: 0.9531 - val_loss: 0.2729 - val_accuracy: 0.9152
Epoch 22/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1392 - ac
curacy: 0.9563 - val_loss: 0.2601 - val_accuracy: 0.9203
Epoch 23/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1311 - ac
curacy: 0.9567 - val_loss: 0.2882 - val_accuracy: 0.9189
Epoch 24/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1288 - ac
curacy: 0.9592 - val_loss: 0.3151 - val_accuracy: 0.9169
Epoch 25/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1499 - ac
curacy: 0.9551 - val_loss: 0.2920 - val_accuracy: 0.9226
Epoch 26/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1378 - ac
curacy: 0.9570 - val_loss: 0.2951 - val_accuracy: 0.9121
Epoch 27/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1321 - ac
curacy: 0.9566 - val_loss: 0.3059 - val_accuracy: 0.9074
Epoch 28/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1258 - ac
curacy: 0.9581 - val_loss: 0.2781 - val_accuracy: 0.9203
Epoch 29/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1275 - ac
curacy: 0.9581 - val_loss: 0.3058 - val_accuracy: 0.9087
Epoch 30/30
7352/7352 [==============================] - 8s 1ms/step - loss: 0.1342 - ac
curacy: 0.9584 - val_loss: 0.2954 - val_accuracy: 0.9145
```

Out[105]:

```
<keras.callbacks.callbacks.History at 0x7fd368461438>
```

In [106]:

```
print(confusion_matrix(Y_test, model.predict(X_test_sc)))
```

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS
\
True
LAYING                 537        0         0        0                   0
SITTING                  0      373       115        0                   0
STANDING                 0       70       462        0                   0
WALKING                  0        0         0      492                   3
WALKING_DOWNSTAIRS       0        0         0       13                 404
WALKING_UPSTAIRS         0       12         0       13                  19

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            3
STANDING                           0
WALKING                            1
WALKING_DOWNSTAIRS                 3
WALKING_UPSTAIRS                 427
```

In [107]:

```
score3 = model.evaluate(X_test_sc,Y_test)
```

```
2947/2947 [==============================] - 1s 216us/step
```

In [108]:

```
score3
```

Out[108]:

```
[0.2953713467495701, 0.9144893288612366]
```

Divide conquer algorithm for the CNN model

- Classification on static and dynamic activities

In [662]:

```
y = _read_csv('y_train.txt')[0]
y[y<=3] = 0
y[y>3] = 1
Y_train_sd = pd.get_dummies(y).as_matrix()
```

```
/home/shanud6711/.local/lib/python3.5/site-packages/ipykernel_launcher.py:4:
FutureWarning: Method .as_matrix will be removed in a future version. Use .v
alues instead.
  after removing the cwd from sys.path.
```

In [663]:

```
y = _read_csv('y_test.txt')[0]
y[y<=3] = 0
y[y>3] = 1
Y_test_sd = pd.get_dummies(y).as_matrix()
```

```
/home/shanud6711/.local/lib/python3.5/site-packages/ipykernel_launcher.py:4:
FutureWarning: Method .as_matrix will be removed in a future version. Use .v
alues instead.
  after removing the cwd from sys.path.
```

In [664]:

```
K.clear_session()
np.random.seed(0)
tf.set_random_seed(0)
sess = tf.Session(graph=tf.get_default_graph())
K.set_session(sess)
model = Sequential()
model.add(Conv1D(filters=32, kernel_size=3, activation='relu',kernel_initializer='he_unifor
model.add(Conv1D(filters=32, kernel_size=3, activation='relu',kernel_initializer='he_unifor
model.add(Dropout(0.6))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(50, activation='relu'))
model.add(Dense(2, activation='softmax'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 126, 32)           896
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 124, 32)           3104
_________________________________________________________________
dropout_1 (Dropout)          (None, 124, 32)           0
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 62, 32)            0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1984)              0
_________________________________________________________________
dense_1 (Dense)              (None, 50)                99250
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 102
=================================================================
Total params: 103,352
Trainable params: 103,352
Non-trainable params: 0
_________________________________________________________________
```

In [665]:

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

In [666]:

```
model.fit(X_train_sc,Y_train_sd, epochs=30, batch_size=16,validation_data=(X_test_sc, Y_tes
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 4s 510us/step - loss: 0.0539 -
accuracy: 0.9786 - val_loss: 0.0119 - val_accuracy: 0.9980
Epoch 2/30
7352/7352 [==============================] - 3s 471us/step - loss: 0.0016 -
accuracy: 0.9993 - val_loss: 0.0167 - val_accuracy: 0.9956
Epoch 3/30
7352/7352 [==============================] - 4s 480us/step - loss: 9.4747e-0
4 - accuracy: 0.9996 - val_loss: 0.0116 - val_accuracy: 0.9976
Epoch 4/30
7352/7352 [==============================] - 3s 446us/step - loss: 0.0013 -
accuracy: 0.9996 - val_loss: 0.0114 - val_accuracy: 0.9980
Epoch 5/30
7352/7352 [==============================] - 3s 455us/step - loss: 4.1275e-0
5 - accuracy: 1.0000 - val_loss: 0.0118 - val_accuracy: 0.9983
Epoch 6/30
7352/7352 [==============================] - 3s 455us/step - loss: 1.4602e-0
5 - accuracy: 1.0000 - val_loss: 0.0130 - val_accuracy: 0.9983
Epoch 7/30
7352/7352 [==============================] - 4s 516us/step - loss: 2.6964e-0
5 - accuracy: 1.0000 - val_loss: 0.0126 - val_accuracy: 0.9983
Epoch 8/30
7352/7352 [==============================] - 3s 473us/step - loss: 5.1072e-0
6 - accuracy: 1.0000 - val_loss: 0.0138 - val_accuracy: 0.9983
Epoch 9/30
7352/7352 [==============================] - 3s 440us/step - loss: 2.8857e-0
6 - accuracy: 1.0000 - val_loss: 0.0140 - val_accuracy: 0.9983
Epoch 10/30
7352/7352 [==============================] - 3s 445us/step - loss: 4.1877e-0
6 - accuracy: 1.0000 - val_loss: 0.0154 - val_accuracy: 0.9983
Epoch 11/30
7352/7352 [==============================] - 3s 460us/step - loss: 4.5057e-0
6 - accuracy: 1.0000 - val_loss: 0.0142 - val_accuracy: 0.9983
Epoch 12/30
7352/7352 [==============================] - 3s 457us/step - loss: 1.4870e-0
6 - accuracy: 1.0000 - val_loss: 0.0144 - val_accuracy: 0.9983
Epoch 13/30
7352/7352 [==============================] - 3s 445us/step - loss: 1.1154e-0
6 - accuracy: 1.0000 - val_loss: 0.0155 - val_accuracy: 0.9983
Epoch 14/30
7352/7352 [==============================] - 3s 450us/step - loss: 6.6128e-0
7 - accuracy: 1.0000 - val_loss: 0.0160 - val_accuracy: 0.9983
Epoch 15/30
7352/7352 [==============================] - 3s 448us/step - loss: 1.3831e-0
6 - accuracy: 1.0000 - val_loss: 0.0158 - val_accuracy: 0.9983
Epoch 16/30
7352/7352 [==============================] - 3s 433us/step - loss: 5.1323e-0
7 - accuracy: 1.0000 - val_loss: 0.0171 - val_accuracy: 0.9983
Epoch 17/30
7352/7352 [==============================] - 3s 441us/step - loss: 5.0681e-0
7 - accuracy: 1.0000 - val_loss: 0.0174 - val_accuracy: 0.9983
Epoch 18/30
7352/7352 [==============================] - 3s 446us/step - loss: 3.6351e-0
7 - accuracy: 1.0000 - val_loss: 0.0189 - val_accuracy: 0.9983
Epoch 19/30
7352/7352 [==============================] - 3s 435us/step - loss: 0.0344 -
```

```
accuracy: 0.9989 - val_loss: 0.0106 - val_accuracy: 0.9976
Epoch 20/30
7352/7352 [==============================] - 3s 433us/step - loss: 0.0017 -
accuracy: 0.9997 - val_loss: 0.0071 - val_accuracy: 0.9986
Epoch 21/30
7352/7352 [==============================] - 3s 430us/step - loss: 4.2396e-0
6 - accuracy: 1.0000 - val_loss: 0.0061 - val_accuracy: 0.9986
Epoch 22/30
7352/7352 [==============================] - 3s 431us/step - loss: 1.8650e-0
6 - accuracy: 1.0000 - val_loss: 0.0069 - val_accuracy: 0.9986
Epoch 23/30
7352/7352 [==============================] - 3s 438us/step - loss: 1.1383e-0
6 - accuracy: 1.0000 - val_loss: 0.0072 - val_accuracy: 0.9986
Epoch 24/30
7352/7352 [==============================] - 3s 440us/step - loss: 7.7350e-0
7 - accuracy: 1.0000 - val_loss: 0.0068 - val_accuracy: 0.9986
Epoch 25/30
7352/7352 [==============================] - 3s 444us/step - loss: 9.2907e-0
7 - accuracy: 1.0000 - val_loss: 0.0074 - val_accuracy: 0.9986
Epoch 26/30
7352/7352 [==============================] - 3s 445us/step - loss: 6.1673e-0
7 - accuracy: 1.0000 - val_loss: 0.0078 - val_accuracy: 0.9986
Epoch 27/30
7352/7352 [==============================] - 3s 446us/step - loss: 2.2000e-0
7 - accuracy: 1.0000 - val_loss: 0.0080 - val_accuracy: 0.9986
Epoch 28/30
7352/7352 [==============================] - 3s 451us/step - loss: 1.2798e-0
7 - accuracy: 1.0000 - val_loss: 0.0082 - val_accuracy: 0.9986
Epoch 29/30
7352/7352 [==============================] - 3s 446us/step - loss: 1.1533e-0
7 - accuracy: 1.0000 - val_loss: 0.0082 - val_accuracy: 0.9986
Epoch 30/30
7352/7352 [==============================] - 3s 445us/step - loss: 2.0668e-0
7 - accuracy: 1.0000 - val_loss: 0.0085 - val_accuracy: 0.9986
```

Out[666]:

```
<keras.callbacks.callbacks.History at 0x7fd258e53e48>
```

In [667]:

```
score4 = model.evaluate(X_test_sc,Y_test_sd)
score4
```

```
2947/2947 [==============================] - 0s 84us/step
```

Out[667]:

```
[0.008455163890011538, 0.9986426830291748]
```

In [668]:

```
model.save('model_2_activity.h5')
```

- classifiaction on static activities

In [669]:

```
y = _read_csv('y_train.txt')[0]
y_tr_sb = y>3
y = y[y>3]
Y_train_stat = pd.get_dummies(y).as_matrix()
```

```
/home/shanud6711/.local/lib/python3.5/site-packages/ipykernel_launcher.py:4:
FutureWarning: Method .as_matrix will be removed in a future version. Use .v
alues instead.
  after removing the cwd from sys.path.
```

In [670]:

```
y = _read_csv('y_test.txt')[0]
y_te_sb = y>3
y = y[y>3]
Y_test_stat = pd.get_dummies(y).as_matrix()
```

```
/home/shanud6711/.local/lib/python3.5/site-packages/ipykernel_launcher.py:4:
FutureWarning: Method .as_matrix will be removed in a future version. Use .v
alues instead.
  after removing the cwd from sys.path.
```

In [671]:

```
X_train_stat = X_train_sc[y_tr_sb]
X_test_stat = X_test_sc[y_te_sb]
```

In [672]:

```
X_train_stat.shape , Y_train_stat.shape
```

Out[672]:

```
((4067, 128, 9), (4067, 3))
```

In [673]:

```
X_test_stat.shape
```

Out[673]:

```
(1560, 128, 9)
```

In [674]:

```
Y_test_stat.shape
```

Out[674]:

```
(1560, 3)
```

In [675]:

```
K.clear_session()
#np.random.seed(10)
#tf.set_random_seed(10)
sess = tf.Session(graph=tf.get_default_graph())
K.set_session(sess)
model1 = Sequential()
model1.add(Conv1D(filters=30, kernel_size=3, activation='relu', kernel_initializer='he_unif
model1.add(Conv1D(filters=50, kernel_size=5, activation='relu', kernel_initializer='he_unif
model1.add(Conv1D(filters=100, kernel_size=3, activation='relu', kernel_initializer='he_uni
model1.add(Flatten())
model1.add(Dense(50, activation='relu'))
model1.add(Dropout(0.5))
model1.add(Dense(3, activation='softmax'))
model1.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 126, 30)           840
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 122, 50)           7550
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 120, 100)          15100
_________________________________________________________________
flatten_1 (Flatten)          (None, 12000)             0
_________________________________________________________________
dense_1 (Dense)              (None, 50)                600050
_________________________________________________________________
dropout_1 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 153
=================================================================
Total params: 623,693
Trainable params: 623,693
Non-trainable params: 0
_________________________________________________________________
```

In [676]:

```
adam = keras.optimizers.Adam(lr=0.001)
from keras.optimizers import SGD
adadelta = keras.optimizers.Adadelta(learning_rate=1.0, rho=0.95)
nadam = keras.optimizers.Nadam(learning_rate=0.002, beta_1=0.9, beta_2=0.999)
opt = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model1.compile(loss='categorical_crossentropy', optimizer= adam, metrics=['accuracy'])
```

In [677]:

```
model1.fit(X_train_stat,Y_train_stat, epochs=50, batch_size=32,validation_data=(X_test_stat
```

```
accuracy: 0.9825 - val_loss: 2.3767 - val_accuracy: 0.8885
Epoch 28/50
4067/4067 [==============================] - 4s 1ms/step - loss: 0.0624 -
accuracy: 0.9786 - val_loss: 2.0915 - val_accuracy: 0.8859
Epoch 29/50
4067/4067 [==============================] - 4s 1ms/step - loss: 0.0723 -
accuracy: 0.9717 - val_loss: 1.7648 - val_accuracy: 0.9058
Epoch 30/50
4067/4067 [==============================] - 4s 1ms/step - loss: 0.0904 -
accuracy: 0.9636 - val_loss: 1.2428 - val_accuracy: 0.9006
Epoch 31/50
4067/4067 [==============================] - 4s 1ms/step - loss: 0.0684 -
accuracy: 0.9747 - val_loss: 1.7873 - val_accuracy: 0.8936
Epoch 32/50
4067/4067 [==============================] - 4s 1ms/step - loss: 0.0607 -
accuracy: 0.9769 - val_loss: 2.2159 - val_accuracy: 0.9077
Epoch 33/50
4067/4067 [==============================] - 4s 1ms/step - loss: 0.0492 -
accuracy: 0.9818 - val_loss: 2.2873 - val_accuracy: 0.9096
Epoch 34/50
```

In [678]:

```
model1.save('model_2_static.h5')
```

- classifiaction on dynamic activities

In [679]:

```
y = _read_csv('y_train.txt')[0]
y_tr_dy = y<=3
y = y[y<=3]
Y_train_dynamic = pd.get_dummies(y).as_matrix()
```

```
/home/shanud6711/.local/lib/python3.5/site-packages/ipykernel_launcher.py:4:
FutureWarning: Method .as_matrix will be removed in a future version. Use .v
alues instead.
  after removing the cwd from sys.path.
```

In [680]:

```
y = _read_csv('y_test.txt')[0]
y_te_dy = y<=3
y = y[y<=3]
Y_test_dynamic = pd.get_dummies(y).as_matrix()
```

```
/home/shanud6711/.local/lib/python3.5/site-packages/ipykernel_launcher.py:4:
FutureWarning: Method .as_matrix will be removed in a future version. Use .v
alues instead.
  after removing the cwd from sys.path.
```

In [681]:

```
X_train_dynamic = X_train_sc[y_tr_dy]
X_test_dynamic = X_test_sc[y_te_dy]
```

In [682]:

```
X_train_dynamic.shape, Y_train_dynamic.shape
```

Out[682]:

```
((3285, 128, 9), (3285, 3))
```

In [683]:

```
X_test_dynamic.shape, Y_test_dynamic.shape
```

Out[683]:

```
((1387, 128, 9), (1387, 3))
```

In [684]:

```
K.clear_session()
np.random.seed(10)
tf.set_random_seed(10)
sess = tf.Session(graph=tf.get_default_graph())
K.set_session(sess)
model = Sequential()
model.add(Conv1D(filters=30, kernel_size=3, activation='relu',kernel_initializer='he_unifor
model.add(Conv1D(filters=50, kernel_size=5, activation='relu',kernel_initializer='he_unifor
model.add(Conv1D(filters=100, kernel_size=3, activation='relu',kernel_initializer='he_unifo
model.add(Flatten())
model.add(Dense(50, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(3, activation='softmax'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 126, 30)           840
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 122, 50)           7550
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 120, 100)          15100
_________________________________________________________________
flatten_1 (Flatten)          (None, 12000)             0
_________________________________________________________________
dense_1 (Dense)              (None, 50)                600050
_________________________________________________________________
dropout_1 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 153
=================================================================
Total params: 623,693
Trainable params: 623,693
Non-trainable params: 0
_________________________________________________________________
```

In [685]:

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

In [686]:

```
model.fit(X_train_dynamic,Y_train_dynamic, epochs=50, batch_size=32,validation_data=(X_test
```

```
Train on 3285 samples, validate on 1387 samples
Epoch 1/50
3285/3285 [==============================] - 4s 1ms/step - loss: 1.0717 -
accuracy: 0.4557 - val_loss: 0.7277 - val_accuracy: 0.6482
Epoch 2/50
3285/3285 [==============================] - 4s 1ms/step - loss: 0.6345 -
accuracy: 0.6685 - val_loss: 0.3201 - val_accuracy: 0.9286
Epoch 3/50
3285/3285 [==============================] - 4s 1ms/step - loss: 0.2033 -
accuracy: 0.9205 - val_loss: 0.2088 - val_accuracy: 0.9524
Epoch 4/50
3285/3285 [==============================] - 4s 1ms/step - loss: 0.0643 -
accuracy: 0.9711 - val_loss: 0.3454 - val_accuracy: 0.9524
Epoch 5/50
3285/3285 [==============================] - 4s 1ms/step - loss: 0.0547 -
accuracy: 0.9735 - val_loss: 0.2960 - val_accuracy: 0.9640
Epoch 6/50
3285/3285 [==============================] - 4s 1ms/step - loss: 0.0449 -
accuracy: 0.9799 - val_loss: 0.2723 - val_accuracy: 0.9676
```

In [687]:

```
model.save('model_2_dynamic.h5')
```

In [688]:

```
from keras.models import load_model
import pickle
model_2class = load_model('model_2_activity.h5')
model_static = load_model('model_2_static.h5')
model_dynamic = load_model('model_2_dynamic.h5')
```

In [689]:

```
def predict_activity(X):
    predict_st_dy = model_2class.predict(X)
    y_pred_st_dy =  np.argmax(predict_st_dy, axis=1)

  #static data
    X_static = X[y_pred_st_dy==1]
  #dynamic data
    X_dynamic = X[y_pred_st_dy==0]

    predict_st = model_static.predict(X_static)
    predict_static = np.argmax(predict_st,axis=1)
    predict_static_class_label = predict_static + 4

    predict_dy = model_dynamic.predict(X_dynamic)
    predict_dynamic = np.argmax(predict_dy,axis=1)
    predict_dynamic_class_label = predict_dynamic + 1

    i,j = 0,0
    final_pred = []
    for pred in y_pred_st_dy:
        if pred == 1:
            final_pred.append(predict_static_class_label[i])
            i = i + 1
        else:
            final_pred.append(predict_dynamic_class_label[j])
            j = j + 1

    return final_pred
```

In [690]:

```
X_test_sc.shape,X_train_sc.shape
```

Out[690]:

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

In [691]:

```
Y_train = _read_csv('y_train.txt')[0]
Y_test = _read_csv('y_test.txt')[0]
```

In [692]:

```
Y_train.shape,Y_test.shape
```

Out[692]:

```
((7352,), (2947,))
```

In [693]:

```
##predicting
final_pred_test = predict_activity(X_test_sc)
final_pred_train = predict_activity(X_train_sc)
```

In [694]:

```
##accuracy of train and test
from sklearn.metrics import accuracy_score
print('Accuracy of train data',accuracy_score(Y_train,final_pred_train))
print('Accuracy of test data',accuracy_score(Y_test,final_pred_test))
```

```
Accuracy of train data 0.9948313384113167
Accuracy of test data 0.9395995928062436
```

In [695]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, final_pred_test)
cm
```

Out[695]:

```
array([[495,   0,   1,   0,   0,   0],
       [  1, 442,  28,   0,   0,   0],
       [  2,   0, 418,   0,   0,   0],
       [  0,   3,   0, 406,  82,   0],
       [  1,   0,   0,  56, 475,   0],
       [  0,   0,   0,   0,   4, 533]])
```

by divide and conquer approach we get the max tuned accuracy for the model as 93.95%

In [ ]: