

# CSE 3025 – LARGE SCALE DATA PROCESSING

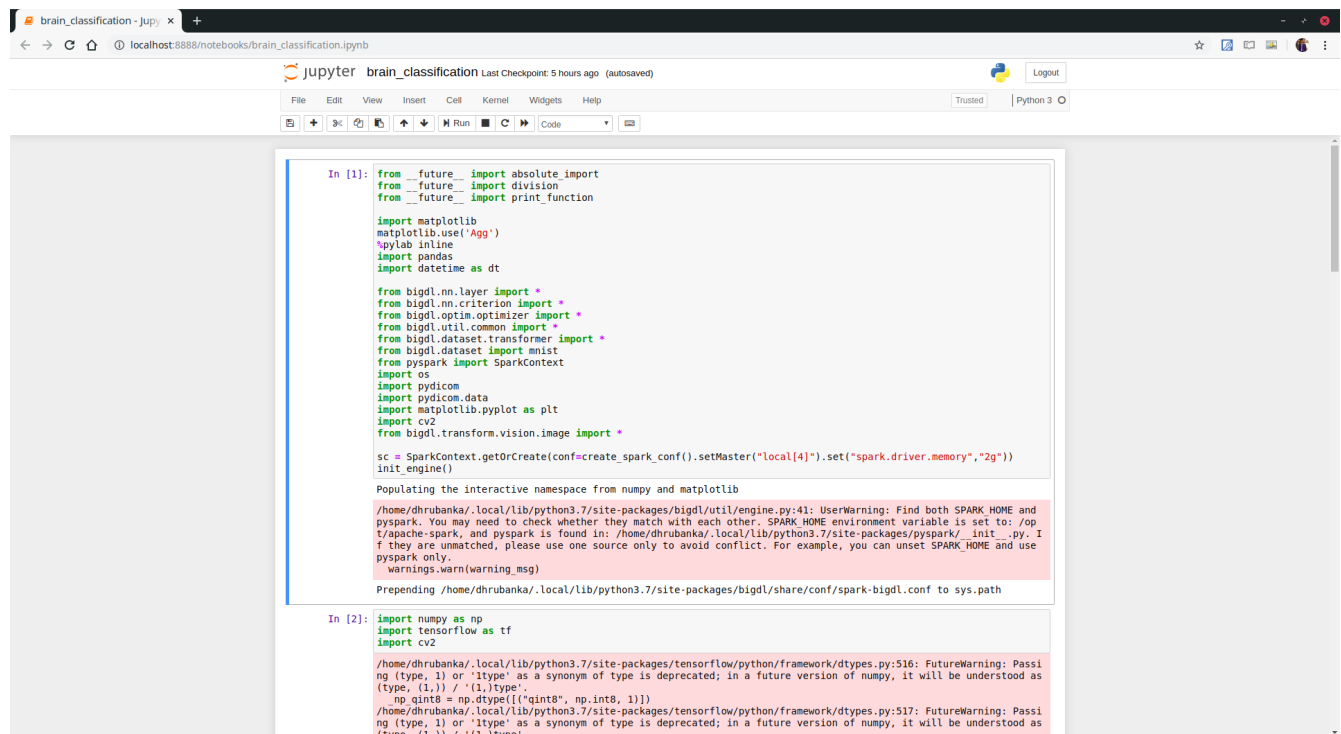
## PROJECT – IMAGE CLASSIFICATION FOR BRAIN MRI SCANS

### FACULTY – PROF. RAMESH RAGALA

TEAM MEMBERS :

DHRUBANKA DUTTA – 17BCE1019  
ARNAV TRIPATHY – 17BCE1026  
AMAN SHAH – 17BCE1221  
SHIKHAR BHARADWAJ – 17BCE1250

PROJECT OUTPUTS :



```
In [1]: from __future__ import absolute import
from __future__ import division
from __future__ import print_function

import matplotlib
matplotlib.use('Agg')
%pylab inline
import pandas
import datetime as dt

from bigdl.nn.layer import *
from bigdl.nn.criterion import *
from bigdl.optim.optimizer import *
from bigdl.util.common import *
from bigdl.dataset.transformer import *
from bigdl.dataset import mnist
from pyspark import SparkContext
import os
import pydicom
import pydicom.data
import matplotlib.pyplot as plt
import cv2
from bigdl.transform.vision.image import *

sc = SparkContext.getOrCreate(conf=create_spark_conf()).setMaster("local[4]").set("spark.driver.memory","2g")
init_engine()

Populating the interactive namespace from numpy and matplotlib

/home/dhrubanka/.local/lib/python3.7/site-packages/bigdl/util/engine.py:41: UserWarning: Find both SPARK_HOME and
pyspark. You may need to check whether they match with each other. SPARK_HOME environment variable is set to: /op
t/apache-spark, and pyspark is found in: /home/dhrubanka/.local/lib/python3.7/site-packages/pyspark/_init_.py. I
f they are unmatched, please use one source only to avoid conflict. For example, you can unset SPARK_HOME and use
pyspark only.
warnings.warn(warning_msg)

Prepending /home/dhrubanka/.local/lib/python3.7/site-packages/bigdl/share/conf/spark-bigdl.conf to sys.path

In [2]: import numpy as np
import tensorflow as tf
import cv2

/home/dhrubanka/.local/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:516: FutureWarning: Passi
ng (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as
(type, (1,)) / '(1,)type'.
  np.qint8 = np.dtype([('qint8', np.int8, 1)])
/home/dhrubanka/.local/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:517: FutureWarning: Passi
ng (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as
(type, (1,)) / '(1,)type'.
```

```
brain_classification - Jupyter
localhost:8888/notebooks/brain_classification.ipynb

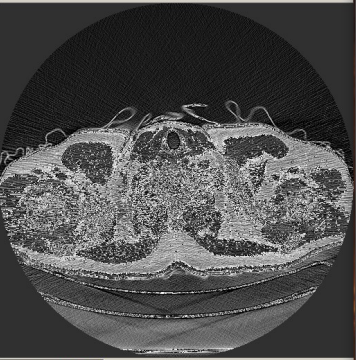
jupyter brain_classification Last Checkpoint: 5 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help
Trust Python 3

In [2]: import numpy as np
import tensorflow as tf
import cv2

/home/dhrubanka/.local/lib/python3.7/site-packages/tensorflow/python/framework/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 as (type, (1,)) / '(1,)type'.
  np.qint8 = np.dtype [("qint8", np.int8, 1)])
/home/dhrubanka/.local/lib/python3.7/site-packages/tensorflow/python/framework/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 as (type, (1,)) / '(1,)type'.
  np.qint8 = np.dtype [("qint8", np.uint8, 1)])
/home/dhrubanka/.local/lib/python3.7/site-packages/tensorflow/python/framework/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 as (type, (1,)) / '(1,)type'.
  np.qint16 = np.dtype [("qint16", np.int16, 1)])
/home/dhrubanka/.local/lib/python3.7/site-packages/tensorflow/python/framework/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 as (type, (1,)) / '(1,)type'.
  np.qint16 = np.dtype [("qint16", np.uint16, 1)])
/home/dhrubanka/.local/lib/python3.7/site-packages/tensorflow/python/framework/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 as (type, (1,)) / '(1,)type'.
  np.qint32 = np.dtype [("qint32", np.int32, 1)])
/home/dhrubanka/.local/lib/python3.7/site-packages/tensorflow/python/framework/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 as (type, (1,)) / '(1,)type'.
  np.resource = np.dtype [("resource", np.ubyte, 1)])
/home/dhrubanka/.local/lib/python3.7/site-packages/tensorflow/compat/tensorflow_stub/dtypes.py:541: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np.qint8 = np.dtype [("qint8", np.int8, 1)])
/home/dhrubanka/.local/lib/python3.7/site-packages/tensorflow/compat/tensorflow_stub/dtypes.py:542: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np.qint8 = np.dtype [("qint8", np.uint8, 1)])
/home/dhrubanka/.local/lib/python3.7/site-packages/tensorflow/compat/tensorflow_stub/dtypes.py:543: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np.qint16 = np.dtype [("qint16", np.int16, 1)])
/home/dhrubanka/.local/lib/python3.7/site-packages/tensorflow/compat/tensorflow_stub/dtypes.py:544: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np.qint16 = np.dtype [("qint16", np.uint16, 1)])
/home/dhrubanka/.local/lib/python3.7/site-packages/tensorflow/compat/tensorflow_stub/dtypes.py:545: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np.qint32 = np.dtype [("qint32", np.int32, 1)])
/home/dhrubanka/.local/lib/python3.7/site-packages/tensorflow/compat/tensorflow_stub/dtypes.py:550: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be under
```

brain\_image\_after



cv2.imread('brain\_image\_after', cv2.IMREAD\_GRAYSCALE)

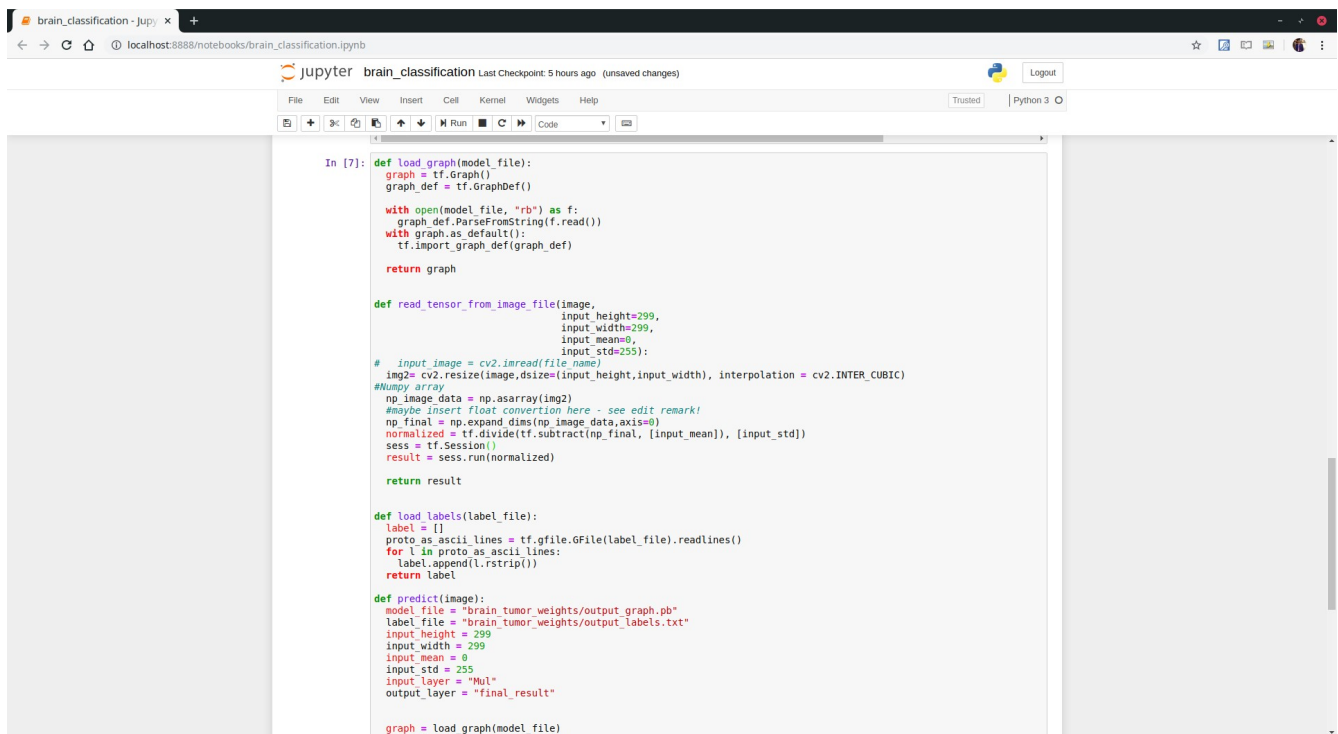
brain\_classification - Jupyter  
localhost:8888/notebooks/brain\_classification.ipynb

jupyter brain\_classification Last Checkpoint: 5 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help  
Trust Python 3

In [1]: path to folder = "/home/dhrubanka/Desktop/COLLEGE/THIRD\_YEAR/FIFTH\_SEMESTER/CSE\_3025\_LARGE\_SCALE\_DATA\_PROCESSING/LS"  
image\_file = "000000.dcm"  
filename = pydicom.data.data\_manager.get\_files(path to folder, image\_file)[0]  
ds = pydicom.dcmread(filename)  
# print(ds)  
brain\_image = ds.pixel\_array  
brain\_image = brain\_image.astype('uint8')  
brain\_image = cv2.cvtColor(brain\_image, cv2.COLOR\_GRAY2BGR)  
cv2.imshow('brain\_image\_after', brain\_image)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

In [11]: def load\_graph(model\_file):  
graph = tf.Graph()  
graph\_def = tf.GraphDef()  
  
with open(model\_file, "rb") as f:  
graph\_def.ParseFromString(f.read())  
with graph.as\_default():  
tf.import\_graph\_def(graph\_def)  
  
return graph  
  
def read\_tensor\_from\_image\_file(image,  
input\_height=299,  
input\_width=299,  
input\_mean=0,  
input\_std=255):  
# input\_image = cv2.imread(file\_name)  
img2 = cv2.resize(image, dsize=(input\_height, input\_width), interpolation = cv2.INTER\_CUBIC)  
#Numpy array  
np\_image\_data = np.asarray(img2)  
#maybe insert float conversion here - see edit remark!  
np\_final = np.expand\_dims(np\_image\_data, axis=0)  
normalized = tf.divide(tf.subtract(np\_final, [input\_mean]), [input\_std])  
sess = tf.Session()  
result = sess.run(normalized)  
  
return result  
  
def load\_labels(label\_file):  
label = []  
proto\_as\_ascii\_lines = tf.gfile.GFile(label\_file).readlines()  
for l in proto\_as\_ascii\_lines:  
label.append(l.rstrip())  
return label



```
In [7]: def load_graph(model_file):
graph = tf.Graph()
graph_def = tf.GraphDef()

with open(model_file, "rb") as f:
    graph_def.ParseFromString(f.read())
with graph.as_default():
    tf.import_graph_def(graph_def)

return graph

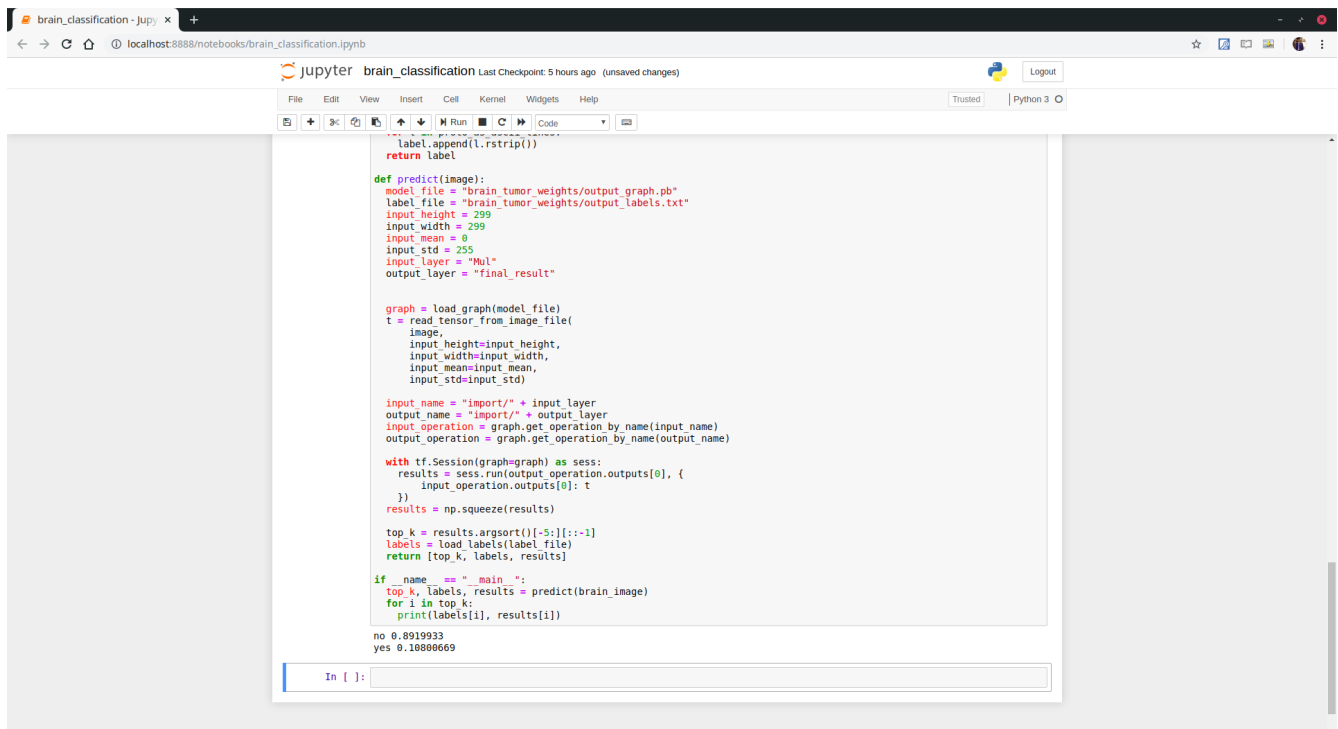
def read_tensor_from_image_file(image,
                                input_height=299,
                                input_width=299,
                                input_mean=0,
                                input_std=255):
    # input_image = cv2.imread(file_name)
    img2= cv2.resize(image,dsi= (input_height,input_width), interpolation = cv2.INTER_CUBIC)
    #Numpy array
    np_image_data = np.asarray(img2)
    #maybe insert float conversion here - see edit remark!
    np_final = np.expand_dims(np_image_data,axis=0)
    normalized = tf.divide(tf.subtract(np_final, [input_mean]), [input_std])
    sess = tf.Session()
    result = sess.run(normalized)

    return result

def load_labels(label_file):
    label = []
    proto_as_ascii_lines = tf.gfile.GFile(label_file).readlines()
    for l in proto_as_ascii_lines:
        label.append(l.rstrip())
    return label

def predict(image):
    model_file = "brain_tumor_weights/output_graph.pb"
    label_file = "brain_tumor_weights/output_labels.txt"
    input_height = 299
    input_width = 299
    input_mean = 0
    input_std = 255
    input_layer = "Mul"
    output_layer = "final_result"

    graph = load_graph(model_file)
```



```
label.append(l.rstrip())
return label

def predict(image):
    model_file = "brain_tumor_weights/output_graph.pb"
    label_file = "brain_tumor_weights/output_labels.txt"
    input_height = 299
    input_width = 299
    input_mean = 0
    input_std = 255
    input_layer = "Mul"
    output_layer = "final_result"

    graph = load_graph(model_file)
    t = read_tensor_from_image_file(
        image,
        input_height=input_height,
        input_width=input_width,
        input_mean=input_mean,
        input_std=input_std)

    input_name = "import/" + input_layer
    output_name = "import/" + output_layer
    input_operation = graph.get_operation_by_name(input_name)
    output_operation = graph.get_operation_by_name(output_name)

    with tf.Session(graph=graph) as sess:
        results = sess.run(output_operation.outputs[0], {
            input_operation.outputs[0]: t
        })
        results = np.squeeze(results)

        top_k = results.argsort()[::-1][-5:][::-1]
        labels = load_labels(label_file)
        return [top_k, labels, results]

if __name__ == "__main__":
    top_k, labels, results = predict(brain_image)
    for i in top_k:
        print(labels[i], results[i])

no 0.8919933
yes 0.10800669

In [ ]:
```

Image classifier model used here was trained using tensorflow.  
Dataset for training brain images classifier – <https://www.kaggle.com/navoneel/brain-mri-images-for-brain-tumor-detection>.  
The classifier was trained on tensorflow with an validation metric of 95%.  
Implemented on single image to show results.  
The classification process runs on a spark-bigdl instance.