CLASSIFICATON OF BRAIN CANCER IMAGES USING BIGDL

A PROJECT REPORT

Submitted by

DHRUBANKA DUTTA(17BCE1019)
ARNAV TRIPATHY(17BCE1026)
AMAN SHAH(17BCE1221)
SHIKAR BHARADWAJ(17BCE1250)

B.Tech

Computer Science and Engineering

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computing Science and Engineering

ACKNOWLEDGEMENT

We would like to express my special thanks of gratitude to my professor Prof. Ramesh Ragala, who gave me this opportunity to do this project of Large Scale Data Processing on “CLASSIFICATON OF BRAIN CANCER IMAGES USING BIGDL”, who also helped me in the completion of my project. I came to know about many new things and I’m really thankful to them.

I would also like to thank my friends who helped me a lot in finalizing this project within the limited time frame.

Dhrubanka Dutta
17BCE1019

CONTENTS

INTRODUCTION

Classification of brain cancer images is important in the medical field for quick evaluation and fast reports. The identification, segmentation and detection of infecting area in brain tumor MRI images are a tedious and time-consuming task. This becomes a problem when there are large number of scanned images to go through. Here a solution is provided in form of BigDL which is a platform for processing large amont of data in deep learning.

The dataset consists of aronf 4700 brain cancer images that will classified as brain having tumor or not.
The Cancer Genome Atlas Sarcoma (TCGA-SARC) data collection is part of a larger effort to build a research community focused on connecting cancer phenotypes to genotypes by providing clinical images matched to subjects from The Cancer Genome Atlas (TCGA). Clinical, genetic, and pathological data resides in the Genomic Data Commons (GDC) Data Portal while the radiological data is stored on The Cancer Imaging Archive (TCIA).

Matched TCGA patient identifiers allow researchers to explore the TCGA/TCIA databases for correlations between tissue genotype, radiological phenotype and patient outcomes. Tissues for TCGA were collected from many sites all over the world in order to reach their accrual targets, usually around 500 specimens per cancer type. For this reason the image data sets are also extremely heterogeneous in terms of scanner modalities, manufacturers and acquisition protocols. In most cases the images were acquired as part of routine care and not as part of a controlled research study or clinical trial.

Hadoop is a n open-source software utility that facilitates using a network of many computers to solve problems involving massive amounts of data and computation. It provides a software framework for distributed storage and processing of big data using the MapReduce

Apache Spark is an open-source distributed general-purpose cluster-computing framework. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. Apache Spark has as its architectural foundation the resilient distributed dataset (RDD), a read-only multiset of data items distributed over a cluster of machines, that is maintained in a fault-tolerant way. All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common occurrences and should be automatically handled by the framework.

PySpark is a python API for spark released by Apache Spark community to support python with Spark. Using PySpark, one can easily integrate and work with RDD in python programming language too. There are numerous features that make PySpark such an amazing framework when it comes to working with huge datasets.

BigDL is a distributed deep learning framework for Apache Spark, created by Jason Dai at Intel. igDL is a distributed deep learning library for Apache Spark; with BigDL, users can write their deep learning applications as standard Spark programs, which can directly run on top of existing Spark or Hadoop clusters

Tensorflow Inception v3 is a widely-used image recognition model that has been shown to attain greater than 78.1% accuracy on the ImageNet dataset. The model itself is made up of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concats, dropouts, and fully connected layers. Batchnorm is used extensively throughout the model and applied to activation inputs. Loss is computed via Softmax. Here this model is retrained using a separate dataset to be able to classify

OpenCV is a library of programming functions mainly aimed at real-time computer vision. This module will be mainly used to support image transformations necessary for executing tensorflow code.

OVERVIEW

The main aim of this projects is to work through a large dataset and perform deep learning on the dataset. The dataset consists of more than four thousand images. These are dicom images that contain a lot of valuable information layered on top of the actual image. These underlying images are extracted and uploaded to the hadoop file system. These are uploaded in bytes fomat to preserve the content of the image. Other formats such as jpeg and png in hdfs leads to data loss and therefore corruption of dataset.

A classifier model is trained using google inception v3 model to classify brain scans as brain having a tumor or not having a tumor. The datatset used to train this model is a separate dataset acquires from kaggle which had divided images into yes or no. The model loads the trained weights and forms a graph from it and passes the images through the graph. The graph then extracts features of the image and classifies the image as yes or no.

REQUIRED MODULES

Hadoop -
Apache Hadoop is a collection of open-source software utilities that facilitate using a network of many computers to solve problems involving massive amounts of data and computation. It provides a software framework for distributed storage and processing of big data using the MapReduce programming model.

Installation -

To check, use command - $java -version

```
aman@ubuntu:~$ java -version
openjdk version "11.0.4" 2019-07-16
OpenJDK Runtime Environment (build 11.0.4+11-post-Ubuntu-1u
buntu218.04.3)
OpenJDK 64-Bit Server VM (build 11.0.4+11-post-Ubuntu-1ubun
tu218.04.3, mixed mode, sharing)
aman@ubuntu:~$
```

1) Download Hadoop file using terminal or from website

2) Use command $ tar xzf hadoop-2.7.3.tar.gz to extract file

```
aman@ubuntu:~$ su
Password:
root@ubuntu:/home/aman# tar xzf hadoop-2.7.3.tar.gz
root@ubuntu:/home/aman# mv hadoop-2.7.3 hadoop/
root@ubuntu:/home/aman# exit
exit
```

3) Install Hadoop in Pseudo Distributed Mode

4) Move Hadoop folder to usr/local/hadoop

5) Set Hadoop Environment variables by adding commands to ~/.bashrc file

6) Use $ gedit ~/.bashrc to edit the file

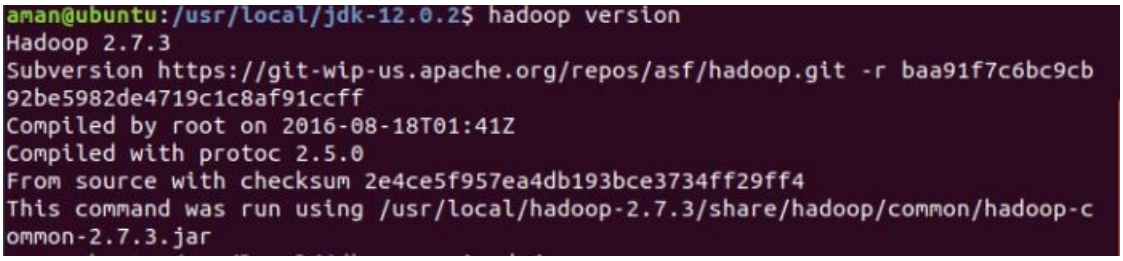
```
aman@ubuntu:~$ gedit ~/.bashrc
```

7) Add the following commands

```
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_INSTALL=$HADOOP_HOME
```

8) Apply changes using $ source ~/.bashrc

9) Check Hadoop Installation

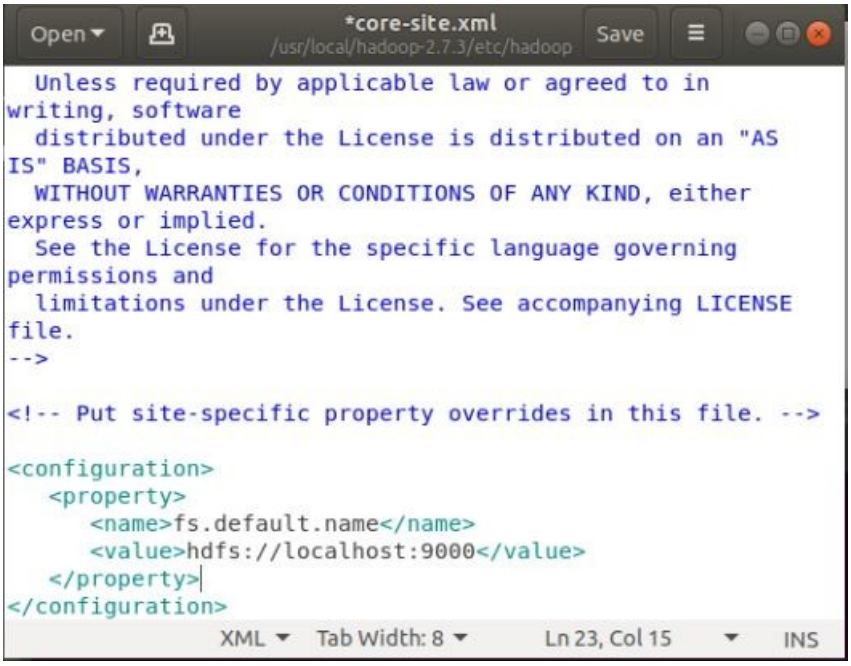


10) Use $ cd $HADOOP_HOME/etc/hadoop

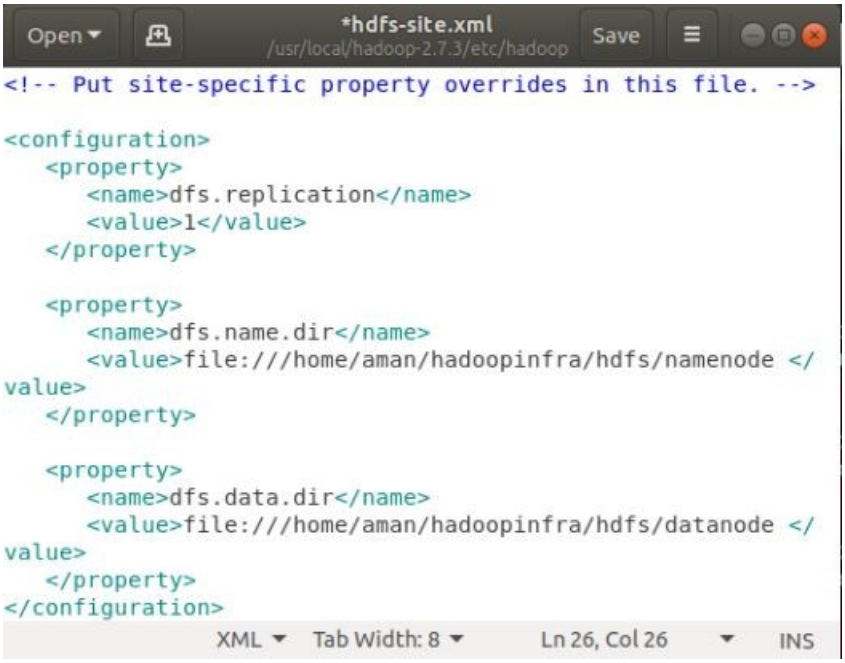
11) Edit the Hadoop-env.sh file and replace JAVA_HOME value by export

12) JAVA_HOME=/usr/local/jdk12.0.2

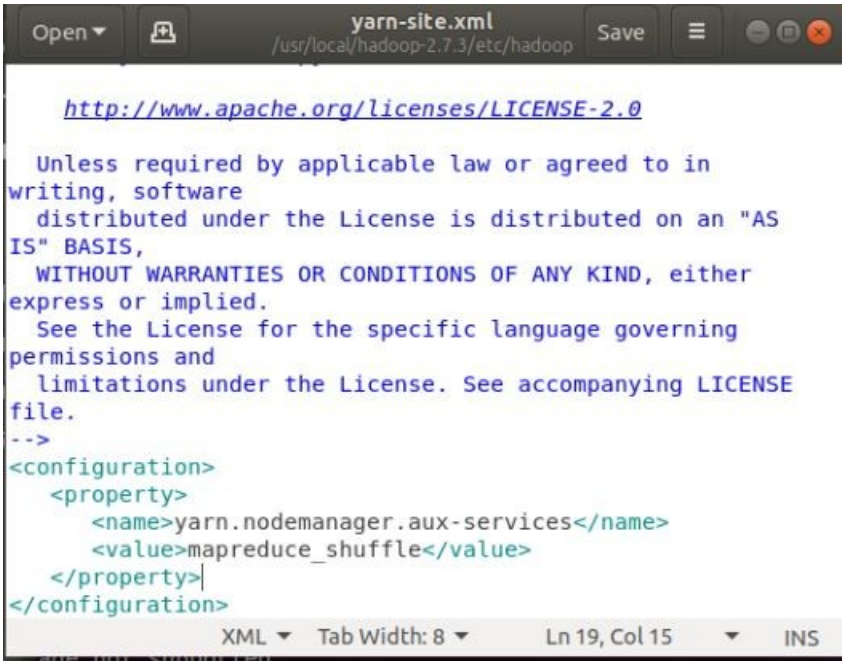
13) Edit the core-site.xml



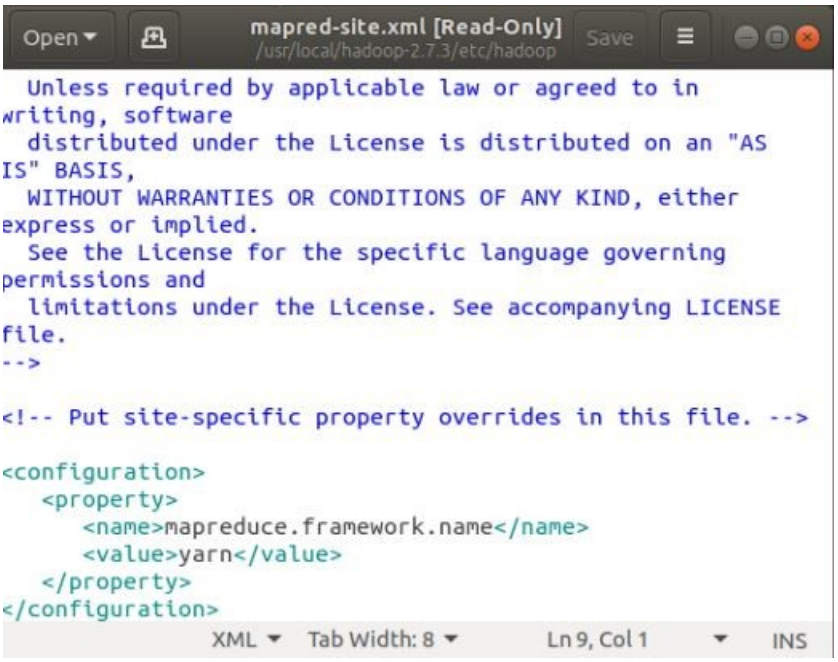
14) Edit hdfs-site.xml



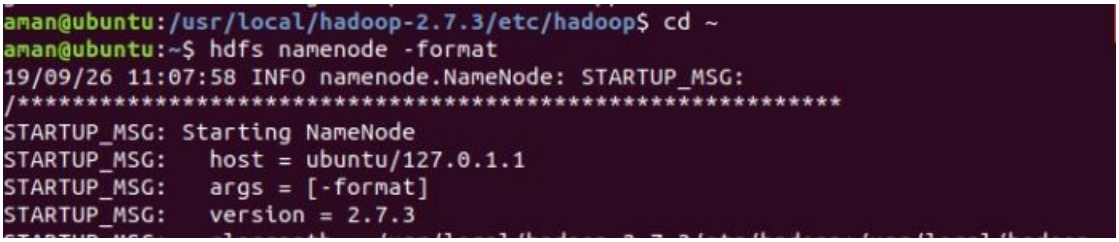
15) Edit yarn-site.xml

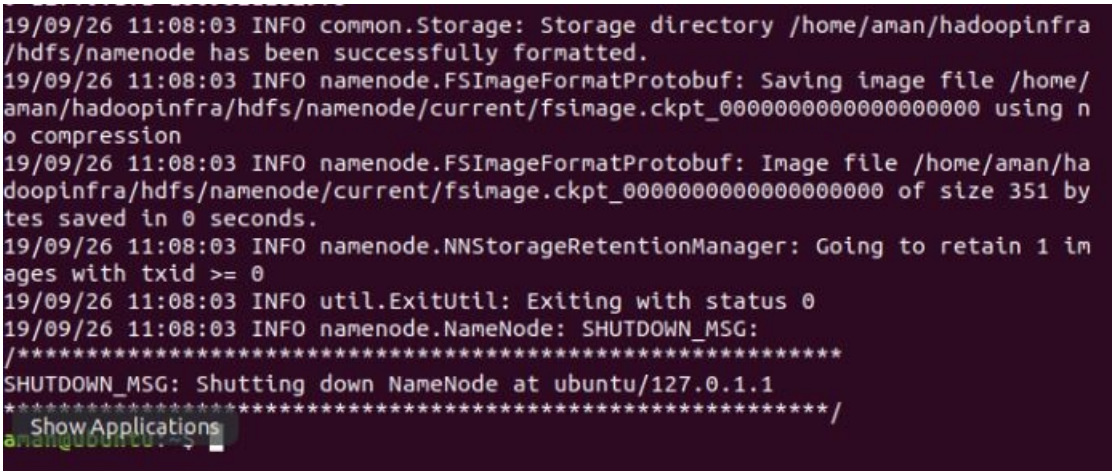


16) Edit Mapred-site.xml



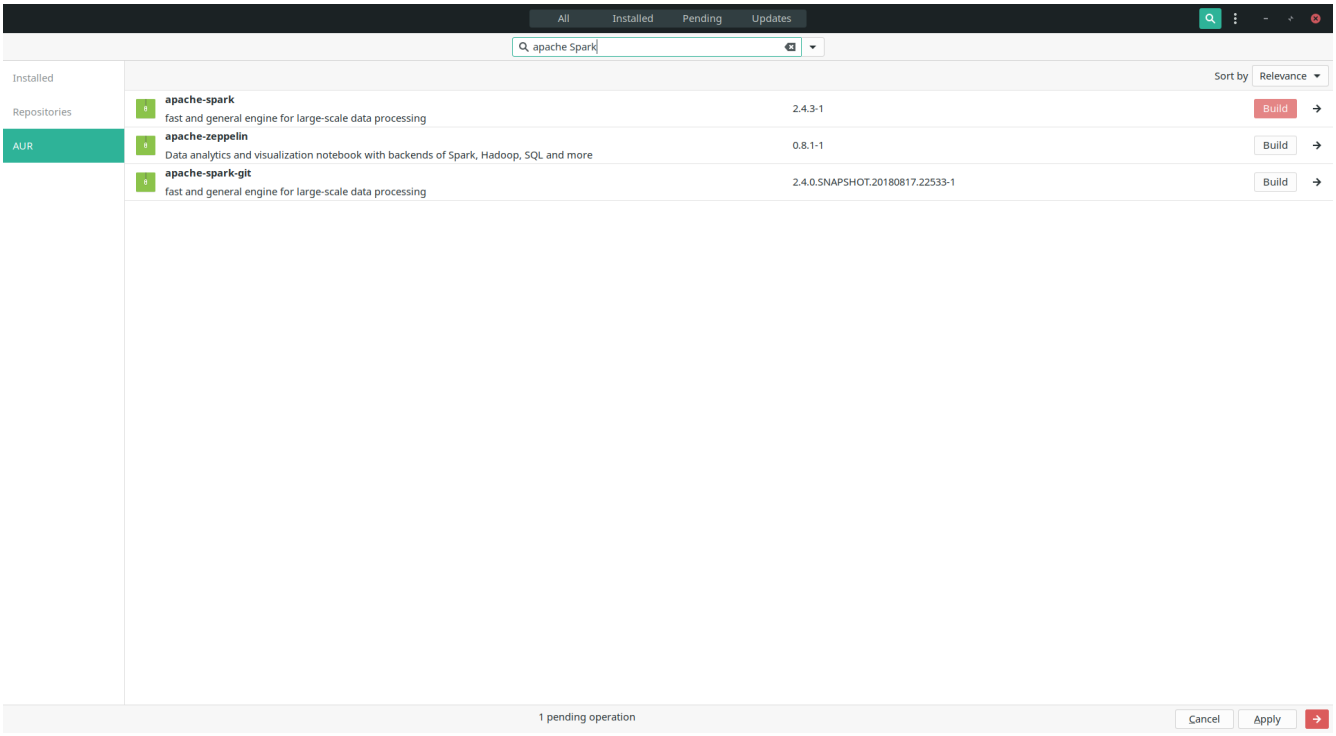
17) Setup namenode using $ hdfs namenode -format

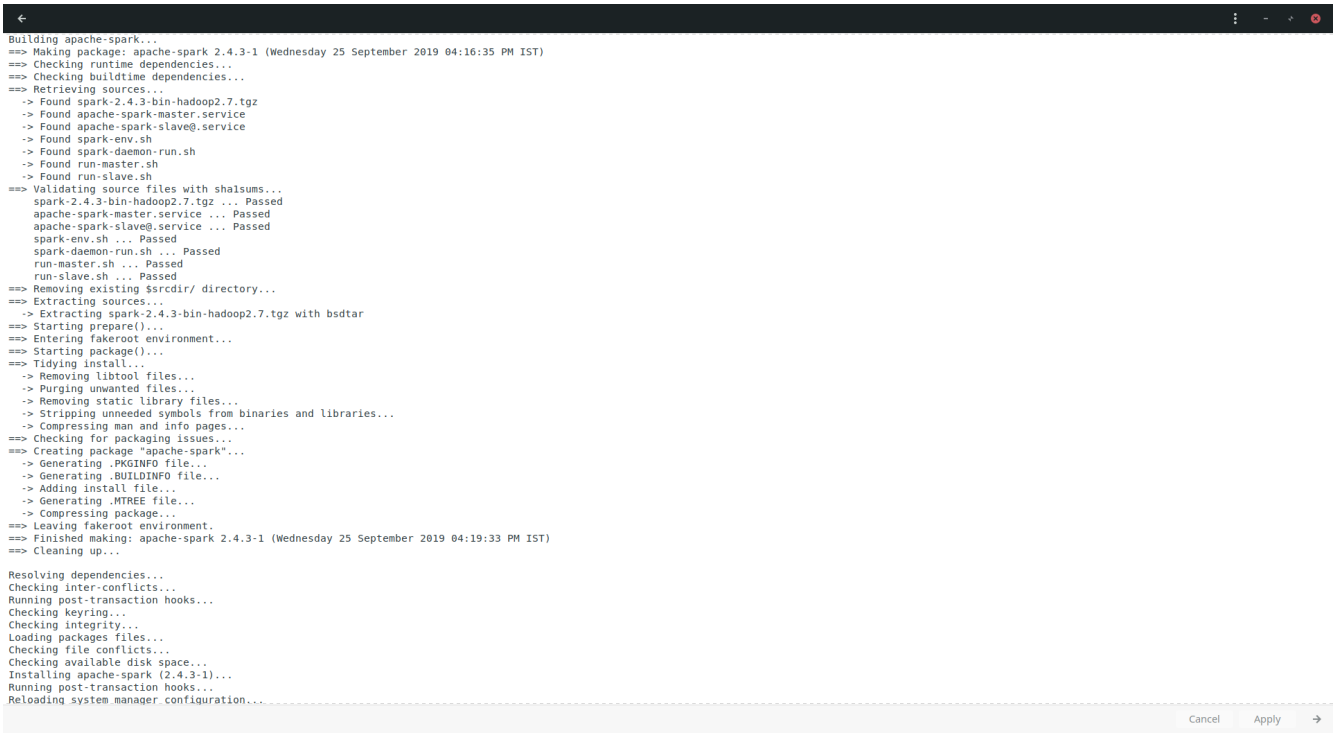


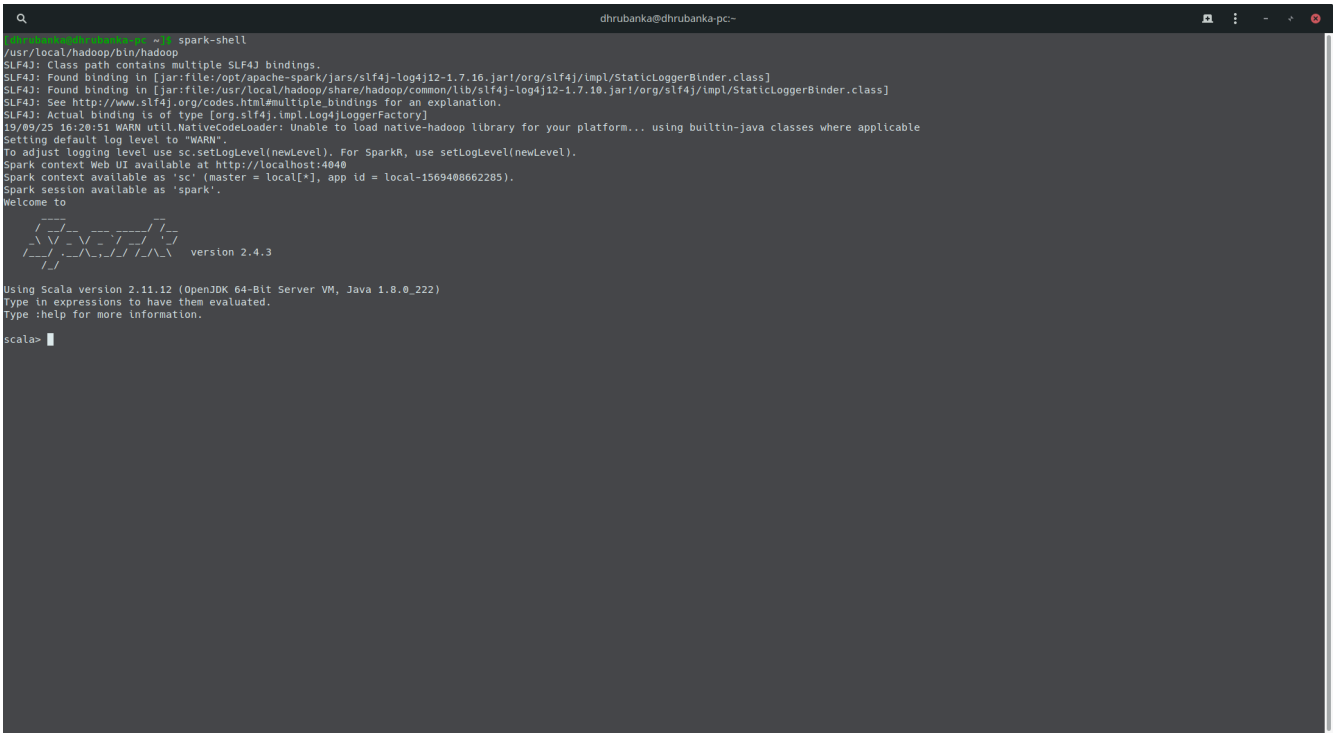


<u>Apache Spark</u> –
Apache Spark is an open-source distributed general-purpose cluster-computing framework. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. Apache Spark has as its architectural foundation the resilient distributed dataset (RDD), a read-only multiset of data items distributed over a cluster of machines, that is maintained in a fault-tolerant way. All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common occurrences and should be automatically handled by the framework.

<u>Installation -</u>
In manjaro arch-linux, apache spark is directly available as a package in the package manager.


All
Installed
Pending
Updates
apache Spark
Installed
Repositories
AUR
Sort by
Relevance
apache-spark
fast and general engine for large-scale data processing
2.4.3-1
Build
apache-zeppelin
Data analytics and visualization notebook with backends of Spark, Hadoop, SQL and more
0.8.1-1
Build
apache-spark-git
fast and general engine for large-scale data processing
2.4.0.SNAPSHOT.20180817.22533-1
Build
1 pending operation
Cancel
Apply


Building apache-spark...
==> Making package: apache-spark 2.4.3-1 (Wednesday 25 September 2019 04:16:35 PM IST)
==> Checking runtime dependencies...
==> Checking buildtime dependencies...
==> Retrieving sources...
-> Found spark-2.4.3-bin-hadoop2.7.tgz
-> Found apache-spark-master.service
-> Found apache-spark-slave@.service
-> Found spark-env.sh
-> Found spark-daemon-run.sh
-> Found run-master.sh
-> Found run-slave.sh
==> Validating source files with sha1sums...
spark-2.4.3-bin-hadoop2.7.tgz ... Passed
apache-spark-master.service ... Passed
apache-spark-slave@.service ... Passed
spark-env.sh ... Passed
spark-daemon-run.sh ... Passed
run-master.sh ... Passed
run-slave.sh ... Passed
==> Removing existing $srcdir/ directory...
==> Extracting sources...
-> Extracting spark-2.4.3-bin-hadoop2.7.tgz with bsdtar
==> Starting prepare()...
==> Entering fakeroot environment...
==> Starting package()...
==> Tidying install...
-> Removing libtool files...
-> Purging unwanted files...
-> Removing static library files...
-> Stripping unneeded symbols from binaries and libraries...
-> Compressing man and info pages...
==> Checking for packaging issues...
==> Creating package "apache-spark"...
-> Generating .PKGINFO file...
-> Generating .BUILDINFO file...
-> Adding install file...
-> Generating .MTREE file...
-> Compressing package...
==> Leaving fakeroot environment.
==> Finished making: apache-spark 2.4.3-1 (Wednesday 25 September 2019 04:19:33 PM IST)
==> Cleaning up...
Resolving dependencies...
Checking inter-conflicts...
Running post-transaction hooks...
Checking keyring...
Checking integrity...
Loading packages files...
Checking file conflicts...
Checking available disk space...
Installing apache-spark (2.4.3-1)...
Running post-transaction hooks...
Reloading system manager configuration...
Cancel
Apply

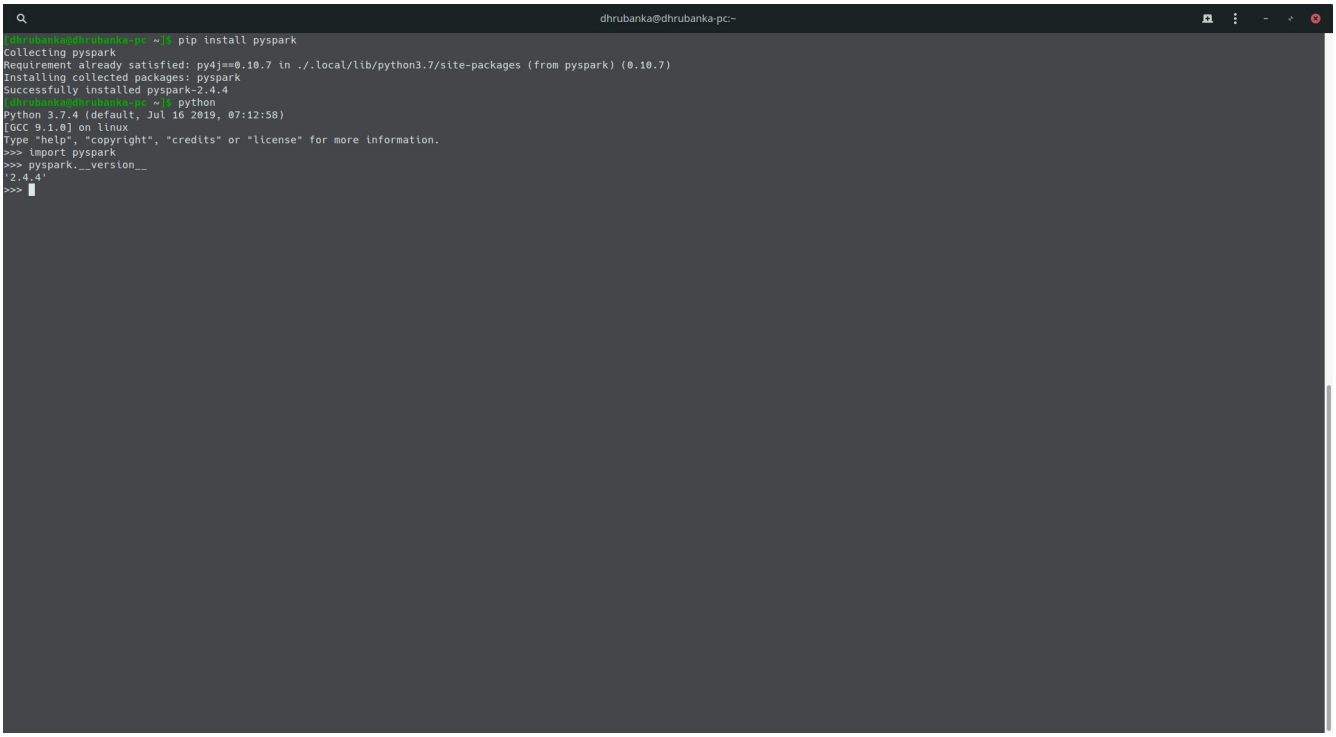


PySpark -

PySpark is a python API for spark released by Apache Spark community to support python with Spark. Using PySpark, one can easily integrate and work with RDD in python programming language too. There are numerous features that make PySpark such an amazing framework when it comes to working with huge datasets.

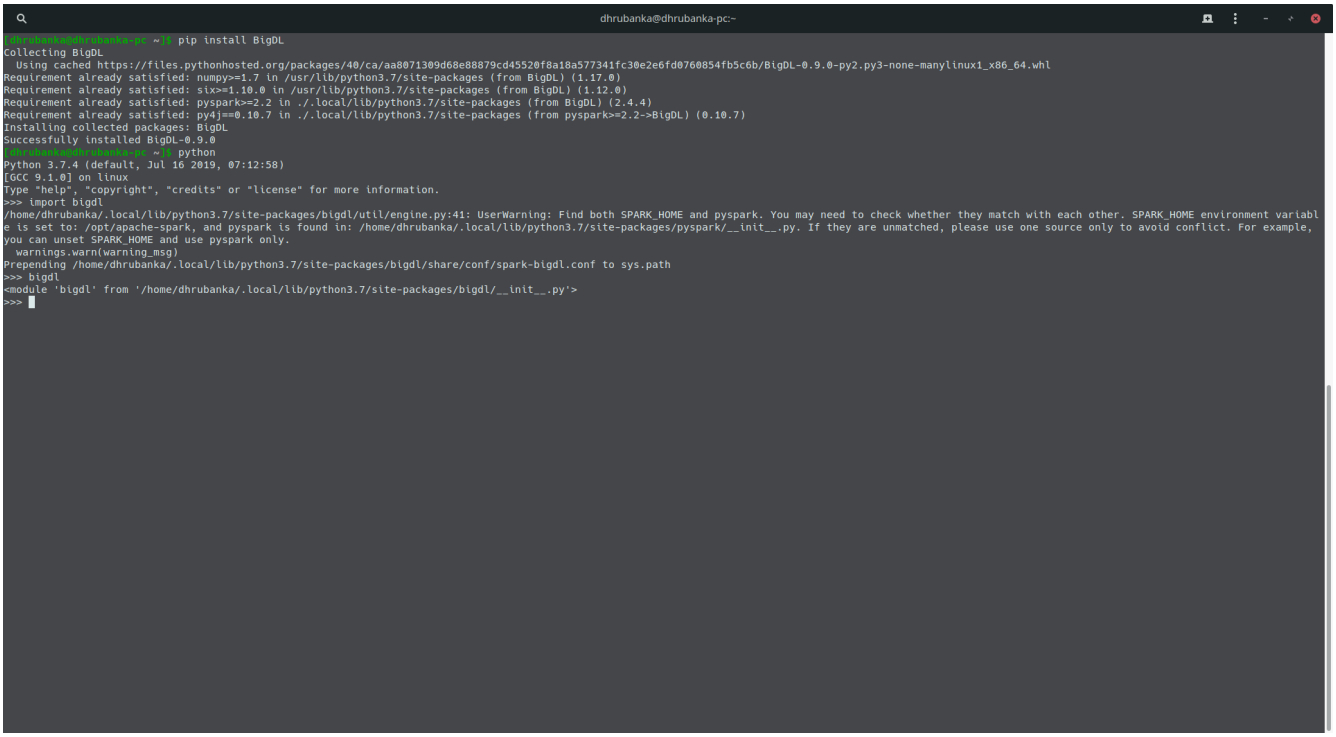
Installation -

The package pyspark can be installed using command, pip install pyspark –user.



BigDL -

BigDL is a distributed deep learning framework for Apache Spark, created by Jason Dai at Intel. igDL is a distributed deep learning library for Apache Spark; with BigDL, users can write their deep learning applications as standard Spark programs, which can directly run on top of existing Spark or Hadoop clusters

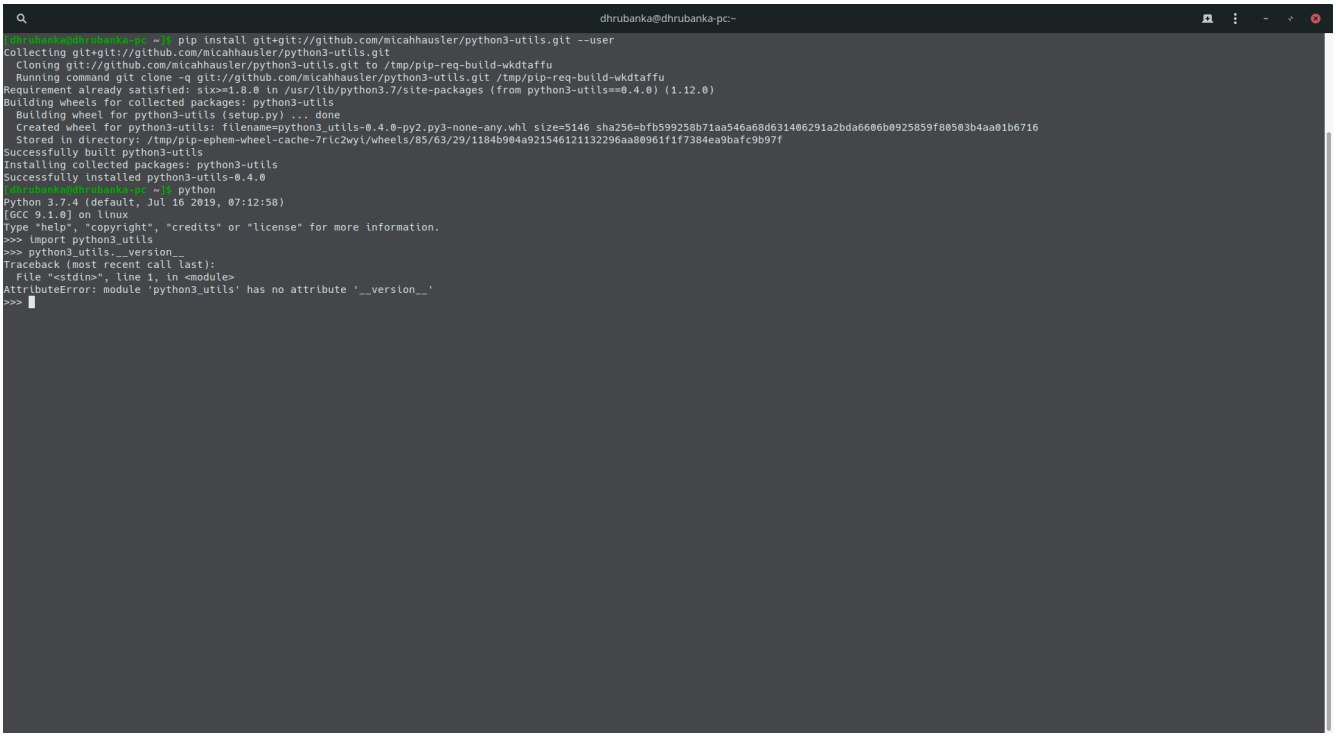


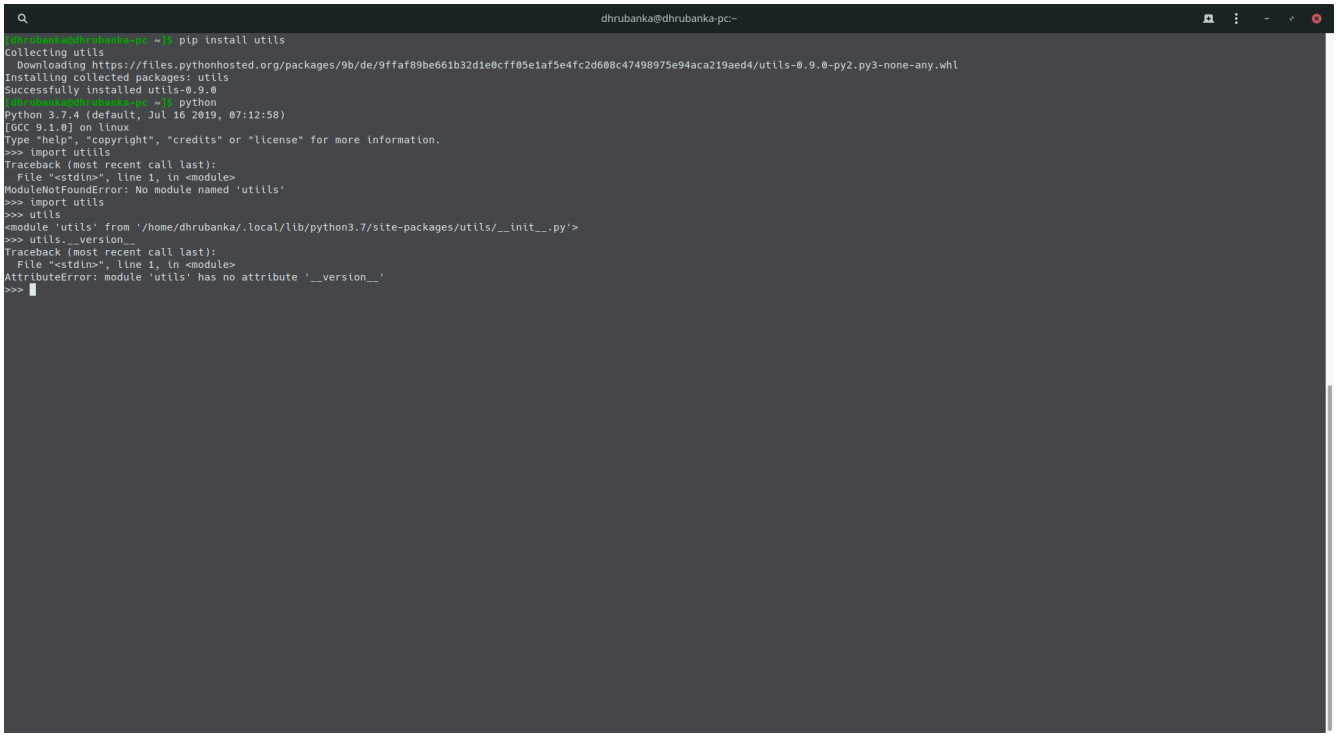
Installation - The package bigdl can be installed using command, pip install bigdl –user.

Utils -

Installation -

The package utils can be installed using command, pip install utils –user.





OpenCV:
OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source BSD license.

It can be installed using pip install opencv-python==”version” --user

```
[dhrubanka@dhrubanka-pc ~]$ pip install opencv-python==4.1.0.25 --user
Requirement already satisfied: opencv-python==4.1.0.25 in /usr/lib/python3.7/site-packages (4.1.0.25)
Requirement already satisfied: numpy>=1.14.5 in /usr/lib/python3.7/site-packages (from opencv-python==4.1.0.25) (1.17.3)
[dhrubanka@dhrubanka-pc ~]$ python
Python 3.7.4 (default, Oct  4 2019, 06:57:26)
[GCC 9.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'4.1.0'
>>>
```

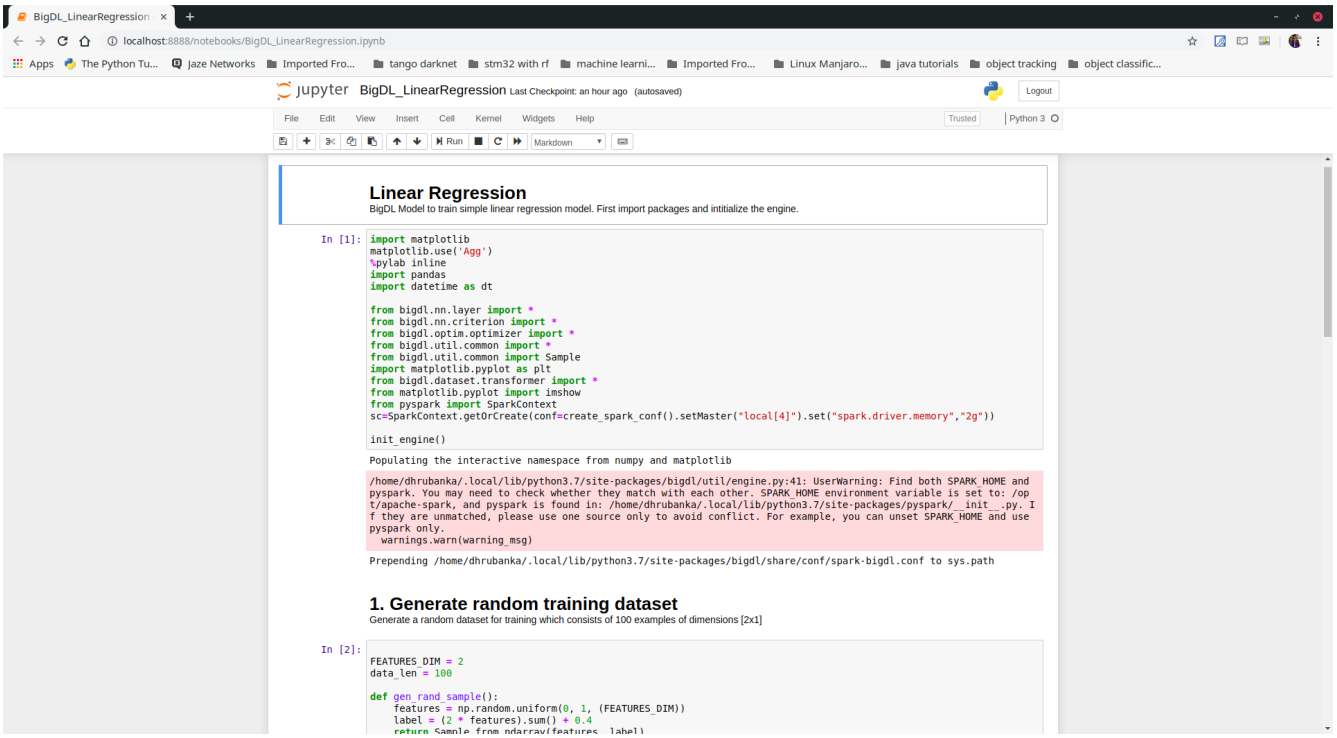
Tensorflow:
TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks.

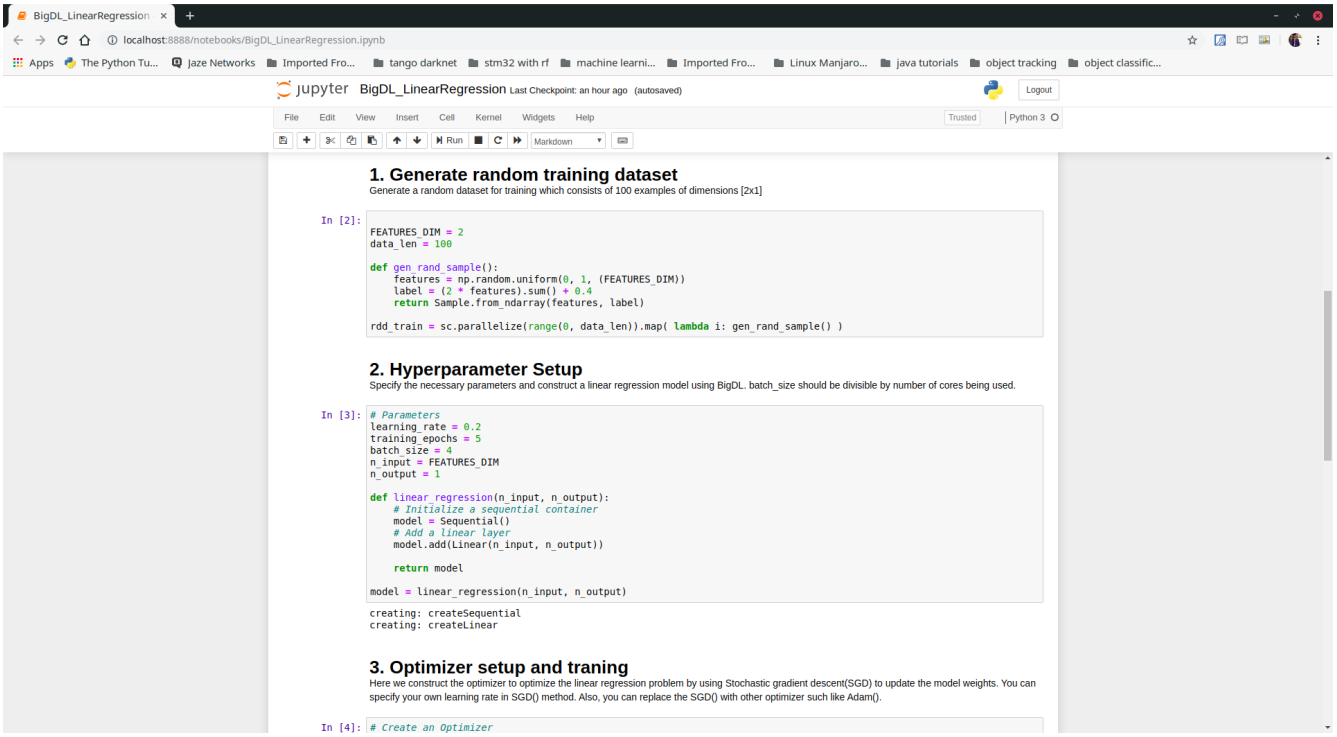
It can be installed using pip install tensorflow==”version” --user

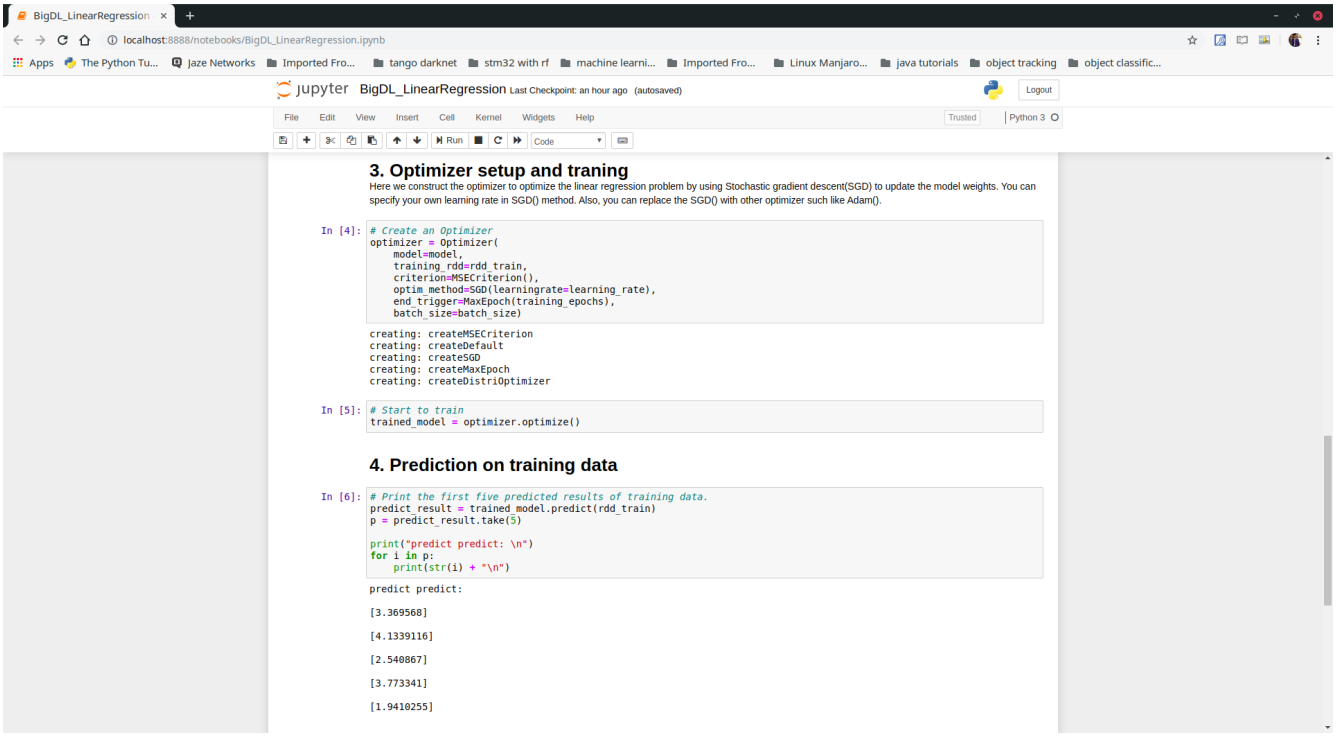
```
[dhrubanka@dhrubanka-pc ~]$ pip install tensorflow==1.14.0
Requirement already satisfied: tensorflow==1.14.0 in ./.local/lib/python3.7/site-packages (1.14.0)
Requirement already satisfied: grpcio>=1.8.6 in /usr/lib/python3.7/site-packages (from tensorflow==1.14.0) (1.24.3)
Requirement already satisfied: gast>=0.2.0 in /usr/lib/python3.7/site-packages (from tensorflow==1.14.0) (0.3.2)
Requirement already satisfied: wheel>=0.26 in /usr/lib/python3.7/site-packages (from tensorflow==1.14.0) (0.33.6)
Requirement already satisfied: astor>=0.6.0 in /usr/lib/python3.7/site-packages (from tensorflow==1.14.0) (0.8.0)
Requirement already satisfied: numpy<2.0,>=1.14.5 in /usr/lib/python3.7/site-packages (from tensorflow==1.14.0) (1.17.3)
Requirement already satisfied: wrapt>=1.11.1 in /usr/lib/python3.7/site-packages (from tensorflow==1.14.0) (1.11.2)
Requirement already satisfied: keras-preprocessing>=1.0.5 in /usr/lib/python3.7/site-packages (from tensorflow==1.14.0) (1.1.0)
Requirement already satisfied: google-pasta>=0.1.6 in ./.local/lib/python3.7/site-packages (from tensorflow==1.14.0) (0.1.7)
Requirement already satisfied: tensorboard<1.15.0,>=1.14.0 in ./.local/lib/python3.7/site-packages (from tensorflow==1.14.0) (1.14.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/lib/python3.7/site-packages (from tensorflow==1.14.0) (1.1.0)
Requirement already satisfied: protobuf>=3.6.1 in /usr/lib/python3.7/site-packages (from tensorflow==1.14.0) (3.10.0)
Requirement already satisfied: absl-py>=0.7.0 in /usr/lib/python3.7/site-packages (from tensorflow==1.14.0) (0.8.1)
Requirement already satisfied: six>=1.10.0 in /usr/lib/python3.7/site-packages (from tensorflow==1.14.0) (1.12.0)
Requirement already satisfied: keras-applications>=1.0.6 in /usr/lib/python3.7/site-packages (from tensorflow==1.14.0) (1.0.8)
Requirement already satisfied: tensorflow-estimator<1.15.0rc0,>=1.14.0rc0 in ./.local/lib/python3.7/site-packages (from tensorflow==1.14.0) (1.14.0)
Requirement already satisfied: markdown>=2.6.8 in /usr/lib/python3.7/site-packages (from tensorboard<1.15.0,>=1.14.0->tensorflow==1.14.0) (3.1.1)
Requirement already satisfied: werkzeug>=0.11.15 in /usr/lib/python3.7/site-packages (from tensorboard<1.15.0,>=1.14.0->tensorflow==1.14.0) (0.16.0)
Requirement already satisfied: setuptools>=41.0.0 in /usr/lib/python3.7/site-packages (from tensorboard<1.15.0,>=1.14.0->tensorflow==1.14.0) (41.2.0)
Requirement already satisfied: h5py in /usr/lib/python3.7/site-packages (from keras-applications>=1.0.6->tensorflow==1.14.0) (2.9.0)
```

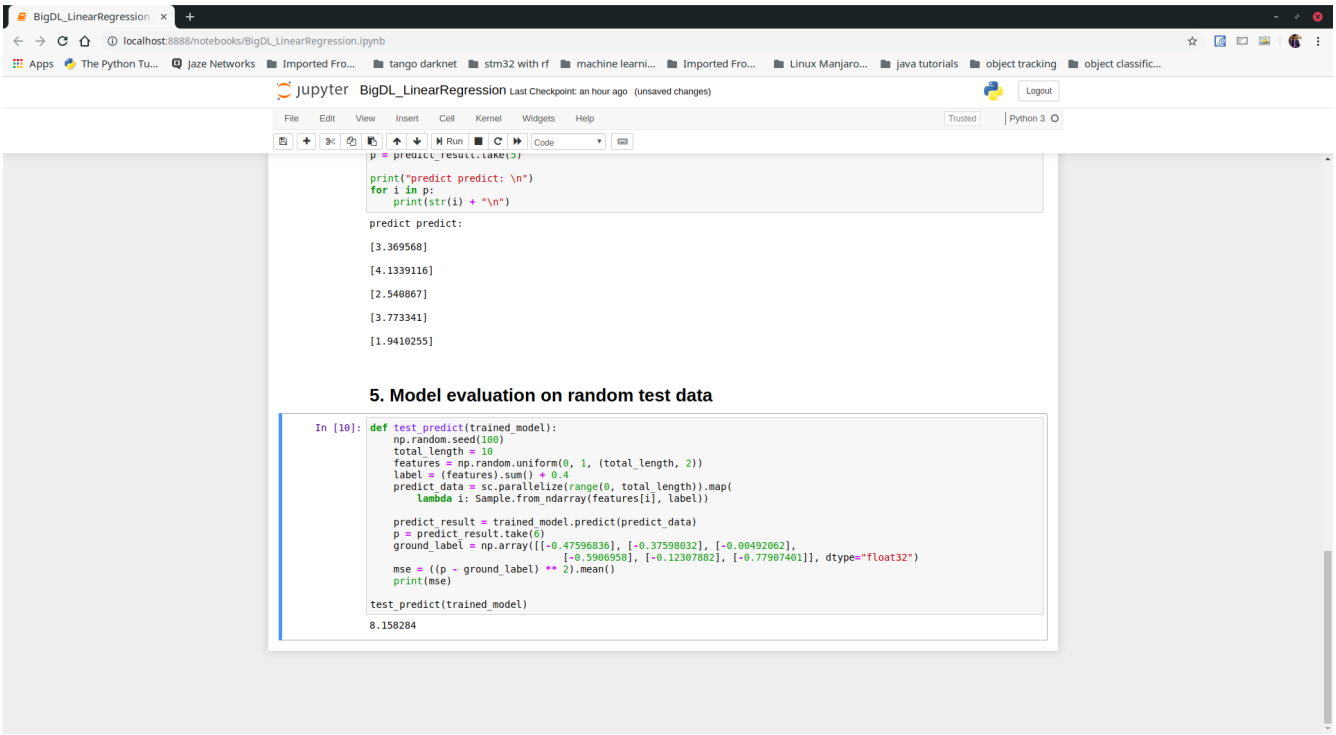
```
[dhrubanka@dhrubanka-pc ~]$ python
Python 3.7.4 (default, Oct  4 2019, 06:57:26)
[GCC 9.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
/home/dhrubanka/.local/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:516: FutureWarning: 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_quint8 = np.dtype([("quint8", 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_quint16 = np.dtype([("quint16", 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/tensorboard/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/tensorboard/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_quint8 = np.dtype([("quint8", np.uint8, 1)])
/home/dhrubanka/.local/lib/python3.7/site-packages/tensorboard/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/tensorboard/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_quint16 = np.dtype([("quint16", np.uint16, 1)])
/home/dhrubanka/.local/lib/python3.7/site-packages/tensorboard/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/tensorboard/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 understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
>>> tf.__version__
'1.14.0'
>>>
```

Example Code :
This is an example code to run linear regression on teh the BigDL platform.



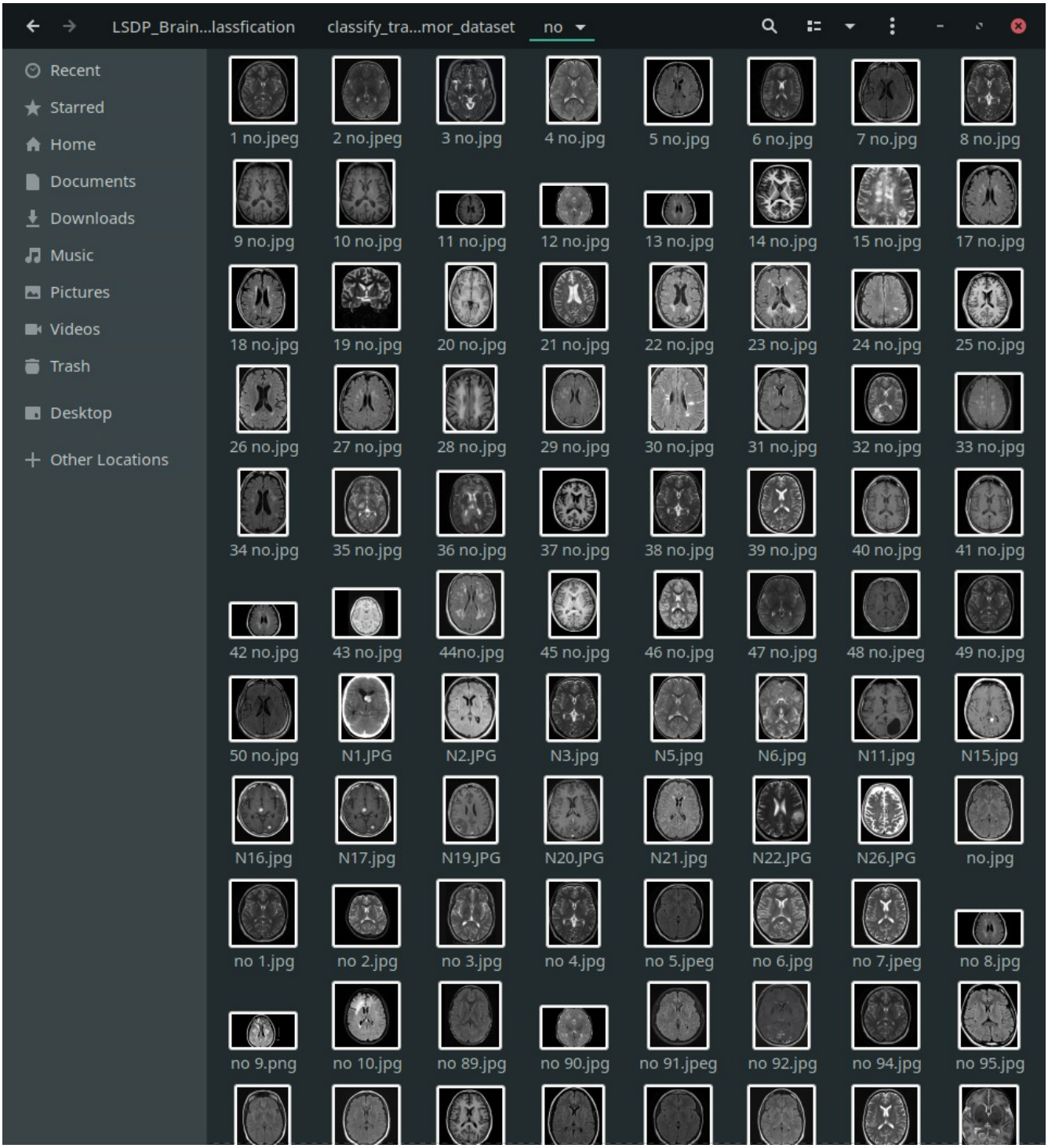


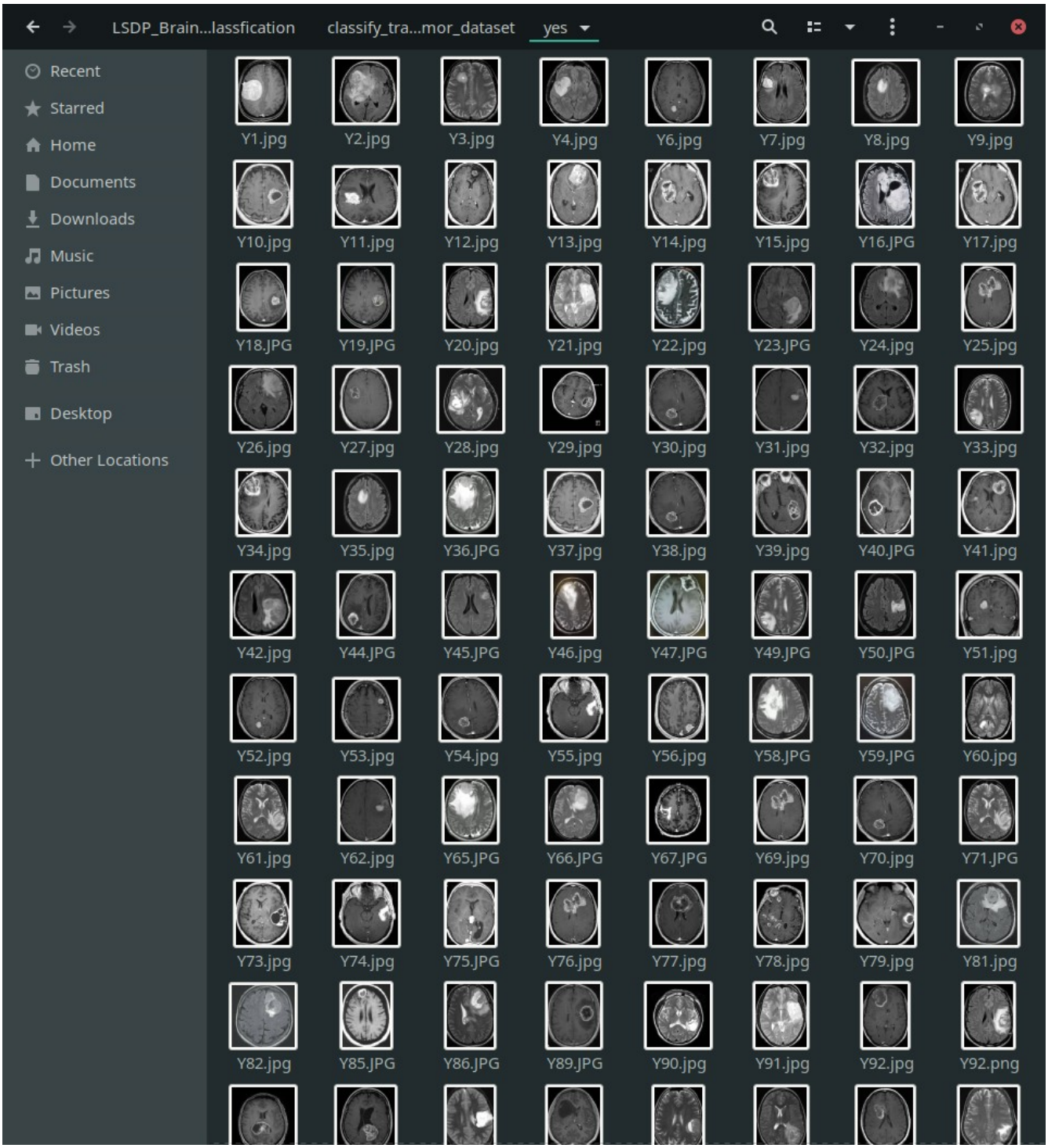

BigDL_LinearRegression
localhost:8888/notebooks/BigDL_LinearRegression.ipynb
jupyter BigDL_LinearRegression Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
3. Optimizer setup and traning
Here we construct the optimizer to optimize the linear regression problem by using Stochastic gradient descent(SGD) to update the model weights. You can specify your own learning rate in SGD() method. Also, you can replace the SGD() with other optimizer such like Adam().
In [4]:
# Create an Optimizer
optimizer = Optimizer(
model=model,
training_rdd=rdd_train,
criterion=MSECriterion(),
optim_method=SGD(learningrate=learning_rate),
end_trigger=MaxEpoch(training_epochs),
batch_size=batch_size)
creating: createMSECriterion
creating: createDefault
creating: createSGD
creating: createMaxEpoch
creating: createDistriOptimizer
In [5]:
# Start to train
trained_model = optimizer.optimize()
4. Prediction on training data
In [6]:
# Print the first five predicted results of training data.
predict_result = trained_model.predict(rdd_train)
p = predict_result.take(5)
print("predict predict: \n")
for i in p:
print(str(i) + "\n")
predict predict:
[3.369568]
[4.1339116]
[2.540867]
[3.773341]
[1.9410255]


BigDL_LinearRegression
localhost:8888/notebooks/BigDL_LinearRegression.ipynb
jupyter BigDL_LinearRegression Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
print("predict predict: \n")
for i in p:
print(str(i) + "\n")
predict predict:
[3.369568]
[4.1339116]
[2.540867]
[3.773341]
[1.9410255]
5. Model evaluation on random test data
In [10]:
def test_predict(trained_model):
np.random.seed(100)
total_length = 10
features = np.random.uniform(0, 1, (total_length, 2))
label = (features).sum() + 0.4
predict_data = sc.parallelize(range(0, total_length)).map(
lambda i: Sample.from_ndarray(features[i], label))
predict_result = trained_model.predict(predict_data)
p = predict_result.take(6)
ground_label = np.array([[-0.47596836], [-0.37598032], [-0.00492062],
[-0.5906958], [-0.12307882], [-0.77907401]], dtype="float32")
mse = ((p - ground_label) ** 2).mean()
print(mse)
test_predict(trained_model)
8.158284

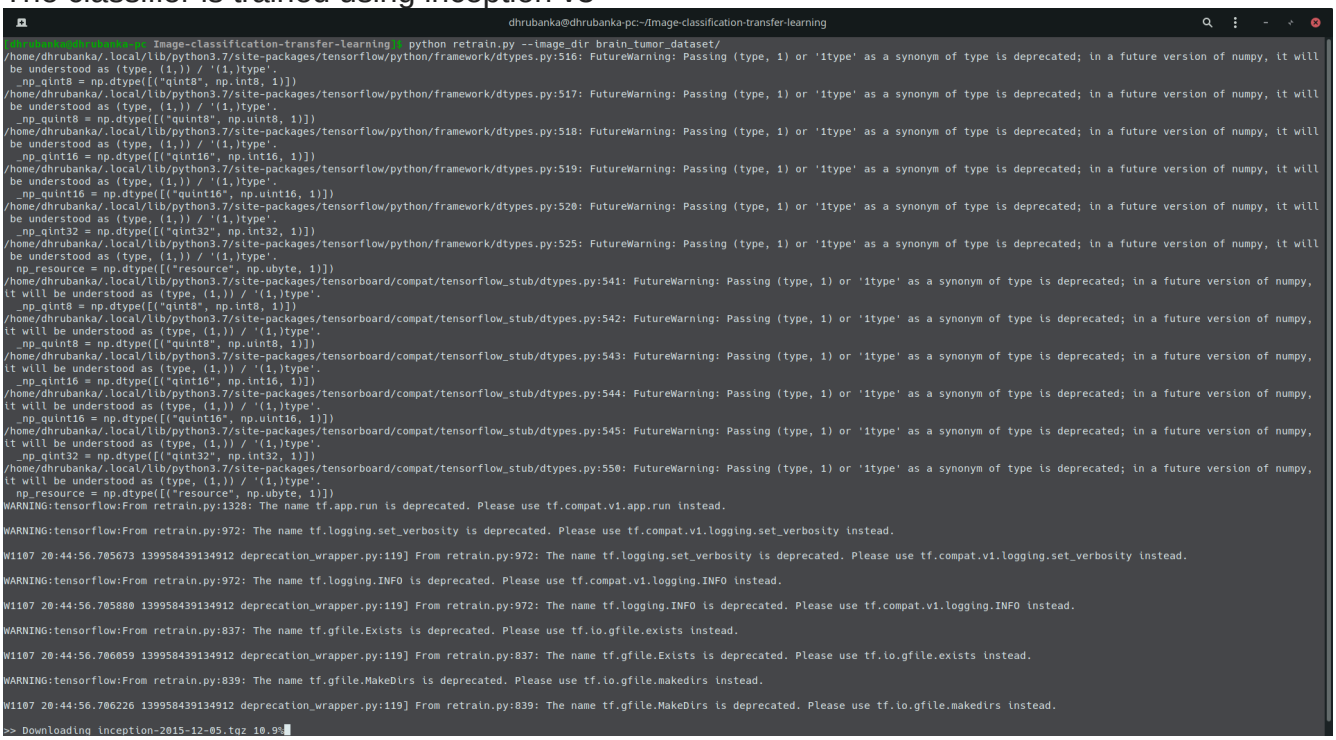
IMPLEMENTATION

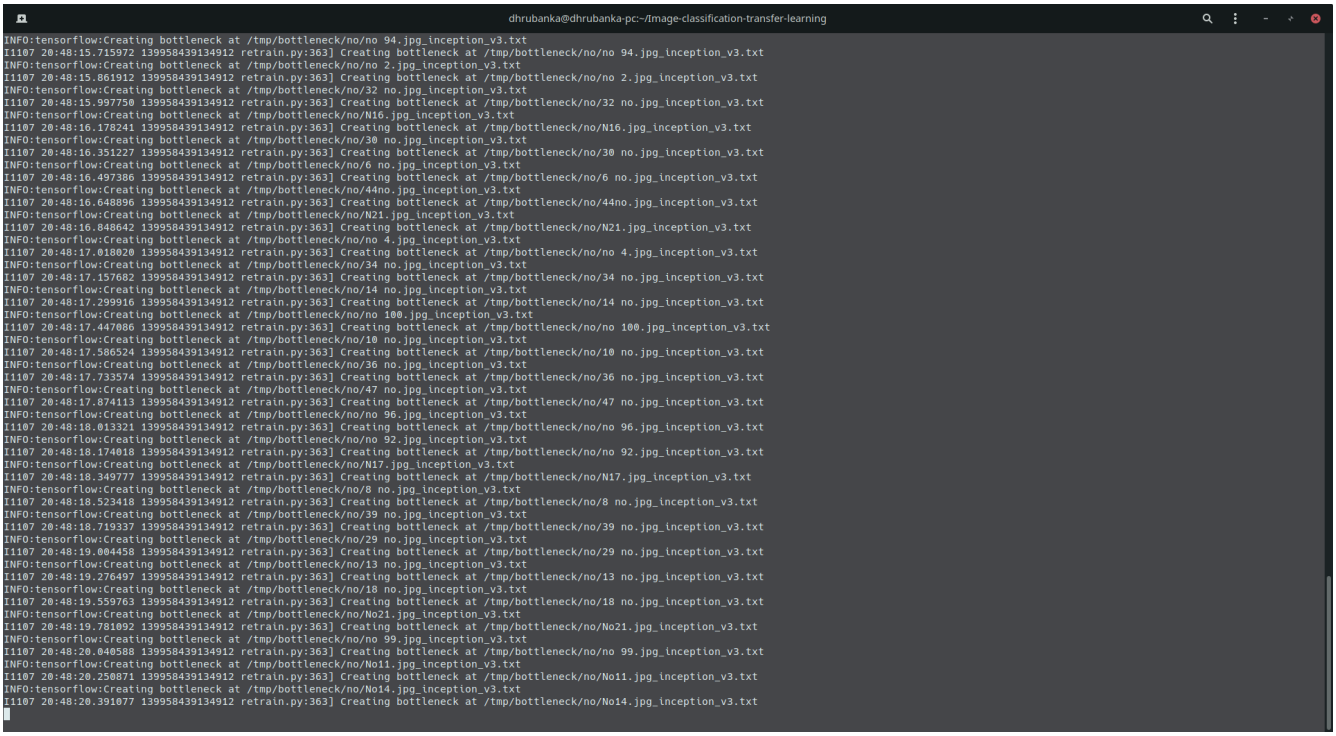
Form dataset for classifier :

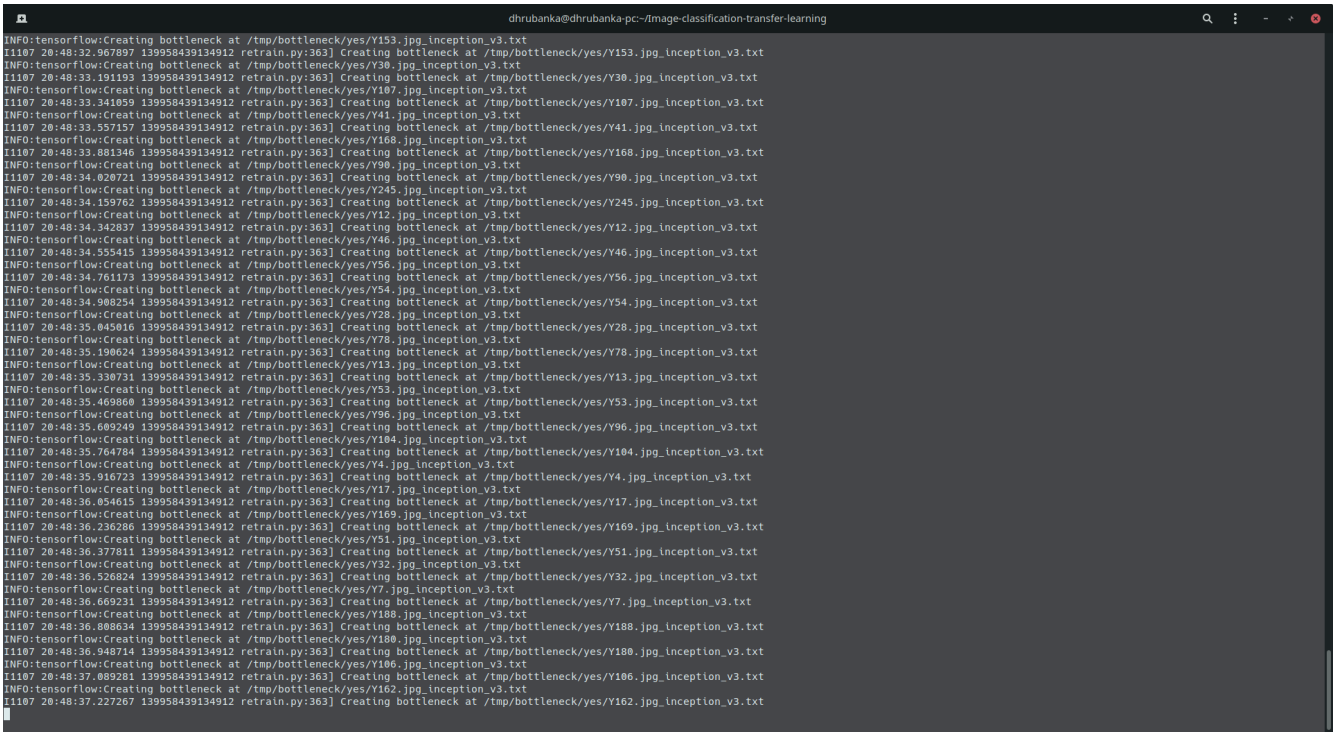


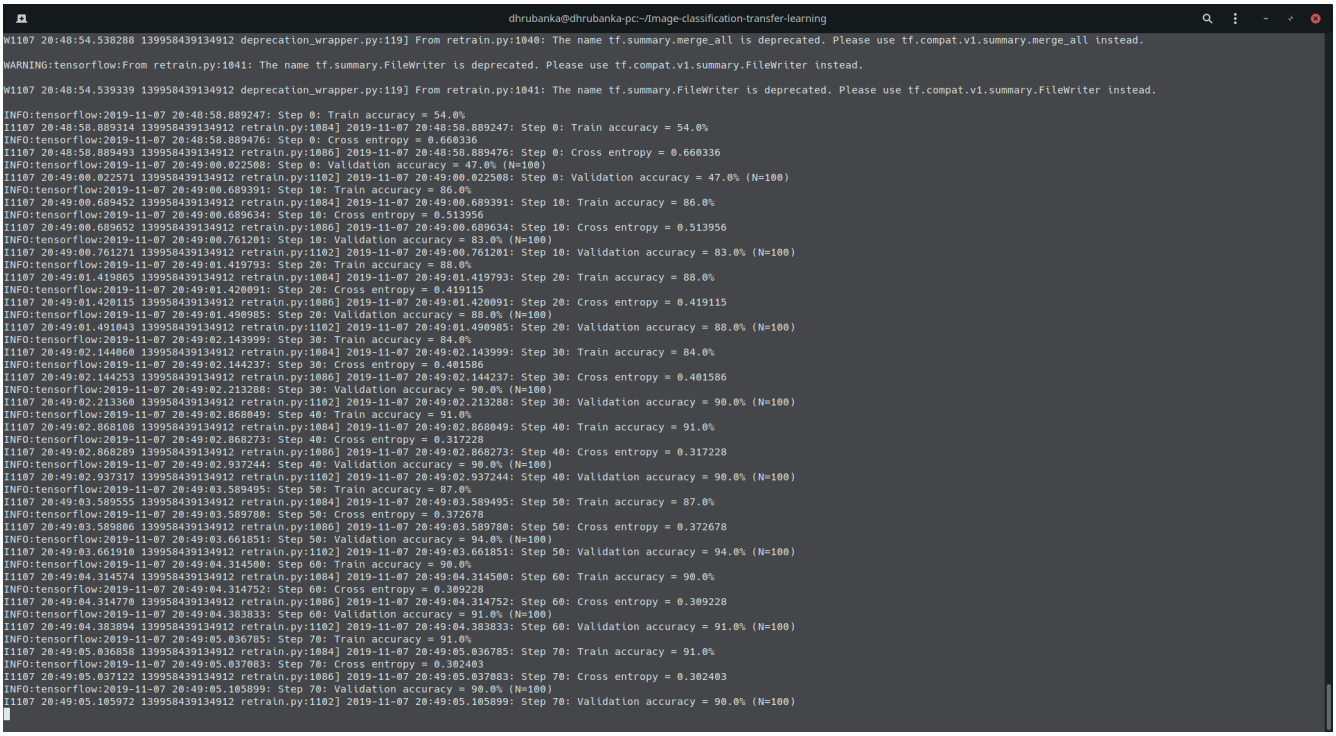

LSDP_Brain...lassfication
classify_tra...mor_dataset
yes
Recent
Starred
Home
Documents
Downloads
Music
Pictures
Videos
Trash
Desktop
Other Locations
Y1.jpg
Y2.jpg
Y3.jpg
Y4.jpg
Y6.jpg
Y7.jpg
Y8.jpg
Y9.jpg
Y10.jpg
Y11.jpg
Y12.jpg
Y13.jpg
Y14.jpg
Y15.jpg
Y16.JPG
Y17.jpg
Y18.JPG
Y19.JPG
Y20.jpg
Y21.jpg
Y22.jpg
Y23.JPG
Y24.jpg
Y25.jpg
Y26.jpg
Y27.jpg
Y28.jpg
Y29.jpg
Y30.jpg
Y31.jpg
Y32.jpg
Y33.jpg
Y34.jpg
Y35.jpg
Y36.JPG
Y37.jpg
Y38.jpg
Y39.jpg
Y40.JPG
Y41.jpg
Y42.jpg
Y44.JPG
Y45.JPG
Y46.jpg
Y47.JPG
Y49.JPG
Y50.JPG
Y51.jpg
Y52.jpg
Y53.jpg
Y54.jpg
Y55.jpg
Y56.jpg
Y58.JPG
Y59.JPG
Y60.jpg
Y61.jpg
Y62.jpg
Y65.JPG
Y66.JPG
Y67.JPG
Y69.jpg
Y70.jpg
Y71.JPG
Y73.jpg
Y74.jpg
Y75.JPG
Y76.jpg
Y77.jpg
Y78.jpg
Y79.jpg
Y81.jpg
Y82.jpg
Y85.JPG
Y86.JPG
Y89.JPG
Y90.jpg
Y91.jpg
Y92.jpg
Y92.png

The classifier is trained using inception v3





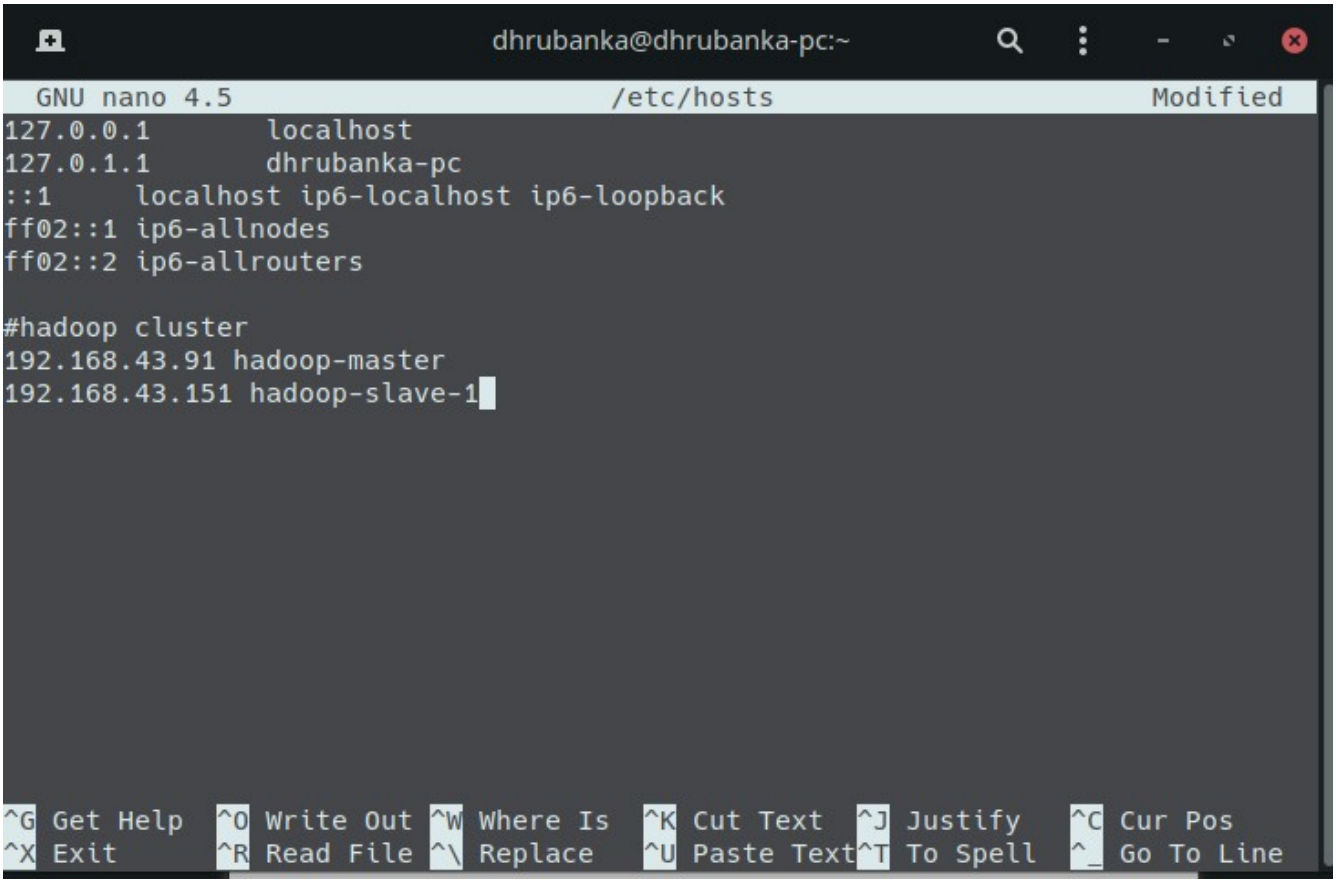

dhrubanka@dhrubanka-pc:~/Image-classification-transfer-learning
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y153.jpg_inception_v3.txt
I1107 20:48:32.967897 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y153.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y30.jpg_inception_v3.txt
I1107 20:48:33.191193 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y30.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y107.jpg_inception_v3.txt
I1107 20:48:33.341059 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y107.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y41.jpg_inception_v3.txt
I1107 20:48:33.557157 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y41.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y168.jpg_inception_v3.txt
I1107 20:48:33.881346 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y168.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y90.jpg_inception_v3.txt
I1107 20:48:34.020721 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y90.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y245.jpg_inception_v3.txt
I1107 20:48:34.159762 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y245.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y12.jpg_inception_v3.txt
I1107 20:48:34.342837 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y12.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y46.jpg_inception_v3.txt
I1107 20:48:34.555415 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y46.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y56.jpg_inception_v3.txt
I1107 20:48:34.761173 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y56.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y54.jpg_inception_v3.txt
I1107 20:48:34.908254 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y54.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y28.jpg_inception_v3.txt
I1107 20:48:35.045016 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y28.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y78.jpg_inception_v3.txt
I1107 20:48:35.190624 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y78.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y13.jpg_inception_v3.txt
I1107 20:48:35.330731 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y13.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y53.jpg_inception_v3.txt
I1107 20:48:35.469860 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y53.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y96.jpg_inception_v3.txt
I1107 20:48:35.609249 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y96.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y104.jpg_inception_v3.txt
I1107 20:48:35.764784 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y104.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y4.jpg_inception_v3.txt
I1107 20:48:35.916723 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y4.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y17.jpg_inception_v3.txt
I1107 20:48:36.054615 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y17.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y169.jpg_inception_v3.txt
I1107 20:48:36.236286 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y169.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y51.jpg_inception_v3.txt
I1107 20:48:36.377811 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y51.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y32.jpg_inception_v3.txt
I1107 20:48:36.526824 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y32.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y7.jpg_inception_v3.txt
I1107 20:48:36.669231 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y7.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y188.jpg_inception_v3.txt
I1107 20:48:36.808634 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y188.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y180.jpg_inception_v3.txt
I1107 20:48:36.948714 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y180.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y106.jpg_inception_v3.txt
I1107 20:48:37.089281 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y106.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/yes/Y162.jpg_inception_v3.txt
I1107 20:48:37.227267 139958439134912 retrain.py:363] Creating bottleneck at /tmp/bottleneck/yes/Y162.jpg_inception_v3.txt


dhrubanka@dhrubanka-pc:~/Image-classification-transfer-learning
W1107 20:48:54.538288 139958439134912 deprecation_wrapper.py:119] From retrain.py:1040: The name tf.summary.merge_all is deprecated. Please use tf.compat.v1.summary.merge_all instead.
WARNING:tensorflow:From retrain.py:1041: The name tf.summary.FileWriter is deprecated. Please use tf.compat.v1.summary.FileWriter instead.
W1107 20:48:54.539339 139958439134912 deprecation_wrapper.py:119] From retrain.py:1041: The name tf.summary.FileWriter is deprecated. Please use tf.compat.v1.summary.FileWriter instead.
INFO:tensorflow:2019-11-07 20:48:58.889247: Step 0: Train accuracy = 54.0%
I1107 20:48:58.889314 139958439134912 retrain.py:1084] 2019-11-07 20:48:58.889247: Step 0: Train accuracy = 54.0%
INFO:tensorflow:2019-11-07 20:48:58.889476: Step 0: Cross entropy = 0.660336
I1107 20:48:58.889493 139958439134912 retrain.py:1086] 2019-11-07 20:48:58.889476: Step 0: Cross entropy = 0.660336
INFO:tensorflow:2019-11-07 20:49:00.022508: Step 0: Validation accuracy = 47.0% (N=100)
I1107 20:49:00.022571 139958439134912 retrain.py:1102] 2019-11-07 20:49:00.022508: Step 0: Validation accuracy = 47.0% (N=100)
INFO:tensorflow:2019-11-07 20:49:00.689391: Step 10: Train accuracy = 86.0%
I1107 20:49:00.689452 139958439134912 retrain.py:1084] 2019-11-07 20:49:00.689391: Step 10: Train accuracy = 86.0%
INFO:tensorflow:2019-11-07 20:49:00.689634: Step 10: Cross entropy = 0.513956
I1107 20:49:00.689652 139958439134912 retrain.py:1086] 2019-11-07 20:49:00.689634: Step 10: Cross entropy = 0.513956
INFO:tensorflow:2019-11-07 20:49:00.761201: Step 10: Validation accuracy = 83.0% (N=100)
I1107 20:49:00.761271 139958439134912 retrain.py:1102] 2019-11-07 20:49:00.761201: Step 10: Validation accuracy = 83.0% (N=100)
INFO:tensorflow:2019-11-07 20:49:01.419793: Step 20: Train accuracy = 88.0%
I1107 20:49:01.419865 139958439134912 retrain.py:1084] 2019-11-07 20:49:01.419793: Step 20: Train accuracy = 88.0%
INFO:tensorflow:2019-11-07 20:49:01.420091: Step 20: Cross entropy = 0.419115
I1107 20:49:01.420115 139958439134912 retrain.py:1086] 2019-11-07 20:49:01.420091: Step 20: Cross entropy = 0.419115
INFO:tensorflow:2019-11-07 20:49:01.490985: Step 20: Validation accuracy = 88.0% (N=100)
I1107 20:49:01.491043 139958439134912 retrain.py:1102] 2019-11-07 20:49:01.490985: Step 20: Validation accuracy = 88.0% (N=100)
INFO:tensorflow:2019-11-07 20:49:02.143999: Step 30: Train accuracy = 84.0%
I1107 20:49:02.144060 139958439134912 retrain.py:1084] 2019-11-07 20:49:02.143999: Step 30: Train accuracy = 84.0%
INFO:tensorflow:2019-11-07 20:49:02.144237: Step 30: Cross entropy = 0.401586
I1107 20:49:02.144253 139958439134912 retrain.py:1086] 2019-11-07 20:49:02.144237: Step 30: Cross entropy = 0.401586
INFO:tensorflow:2019-11-07 20:49:02.213288: Step 30: Validation accuracy = 90.0% (N=100)
I1107 20:49:02.213360 139958439134912 retrain.py:1102] 2019-11-07 20:49:02.213288: Step 30: Validation accuracy = 90.0% (N=100)
INFO:tensorflow:2019-11-07 20:49:02.868049: Step 40: Train accuracy = 91.0%
I1107 20:49:02.868108 139958439134912 retrain.py:1084] 2019-11-07 20:49:02.868049: Step 40: Train accuracy = 91.0%
INFO:tensorflow:2019-11-07 20:49:02.868273: Step 40: Cross entropy = 0.317228
I1107 20:49:02.868289 139958439134912 retrain.py:1086] 2019-11-07 20:49:02.868273: Step 40: Cross entropy = 0.317228
INFO:tensorflow:2019-11-07 20:49:02.937244: Step 40: Validation accuracy = 90.0% (N=100)
I1107 20:49:02.937317 139958439134912 retrain.py:1102] 2019-11-07 20:49:02.937244: Step 40: Validation accuracy = 90.0% (N=100)
INFO:tensorflow:2019-11-07 20:49:03.589495: Step 50: Train accuracy = 87.0%
I1107 20:49:03.589555 139958439134912 retrain.py:1084] 2019-11-07 20:49:03.589495: Step 50: Train accuracy = 87.0%
INFO:tensorflow:2019-11-07 20:49:03.589780: Step 50: Cross entropy = 0.372678
I1107 20:49:03.589806 139958439134912 retrain.py:1086] 2019-11-07 20:49:03.589780: Step 50: Cross entropy = 0.372678
INFO:tensorflow:2019-11-07 20:49:03.661851: Step 50: Validation accuracy = 94.0% (N=100)
I1107 20:49:03.661910 139958439134912 retrain.py:1102] 2019-11-07 20:49:03.661851: Step 50: Validation accuracy = 94.0% (N=100)
INFO:tensorflow:2019-11-07 20:49:04.314500: Step 60: Train accuracy = 90.0%
I1107 20:49:04.314574 139958439134912 retrain.py:1084] 2019-11-07 20:49:04.314500: Step 60: Train accuracy = 90.0%
INFO:tensorflow:2019-11-07 20:49:04.314752: Step 60: Cross entropy = 0.309228
I1107 20:49:04.314770 139958439134912 retrain.py:1086] 2019-11-07 20:49:04.314752: Step 60: Cross entropy = 0.309228
INFO:tensorflow:2019-11-07 20:49:04.383833: Step 60: Validation accuracy = 91.0% (N=100)
I1107 20:49:04.383894 139958439134912 retrain.py:1102] 2019-11-07 20:49:04.383833: Step 60: Validation accuracy = 91.0% (N=100)
INFO:tensorflow:2019-11-07 20:49:05.036785: Step 70: Train accuracy = 91.0%
I1107 20:49:05.036858 139958439134912 retrain.py:1084] 2019-11-07 20:49:05.036785: Step 70: Train accuracy = 91.0%
INFO:tensorflow:2019-11-07 20:49:05.037083: Step 70: Cross entropy = 0.302403
I1107 20:49:05.037122 139958439134912 retrain.py:1086] 2019-11-07 20:49:05.037083: Step 70: Cross entropy = 0.302403
INFO:tensorflow:2019-11-07 20:49:05.105899: Step 70: Validation accuracy = 90.0% (N=100)
I1107 20:49:05.105972 139958439134912 retrain.py:1102] 2019-11-07 20:49:05.105899: Step 70: Validation accuracy = 90.0% (N=100)

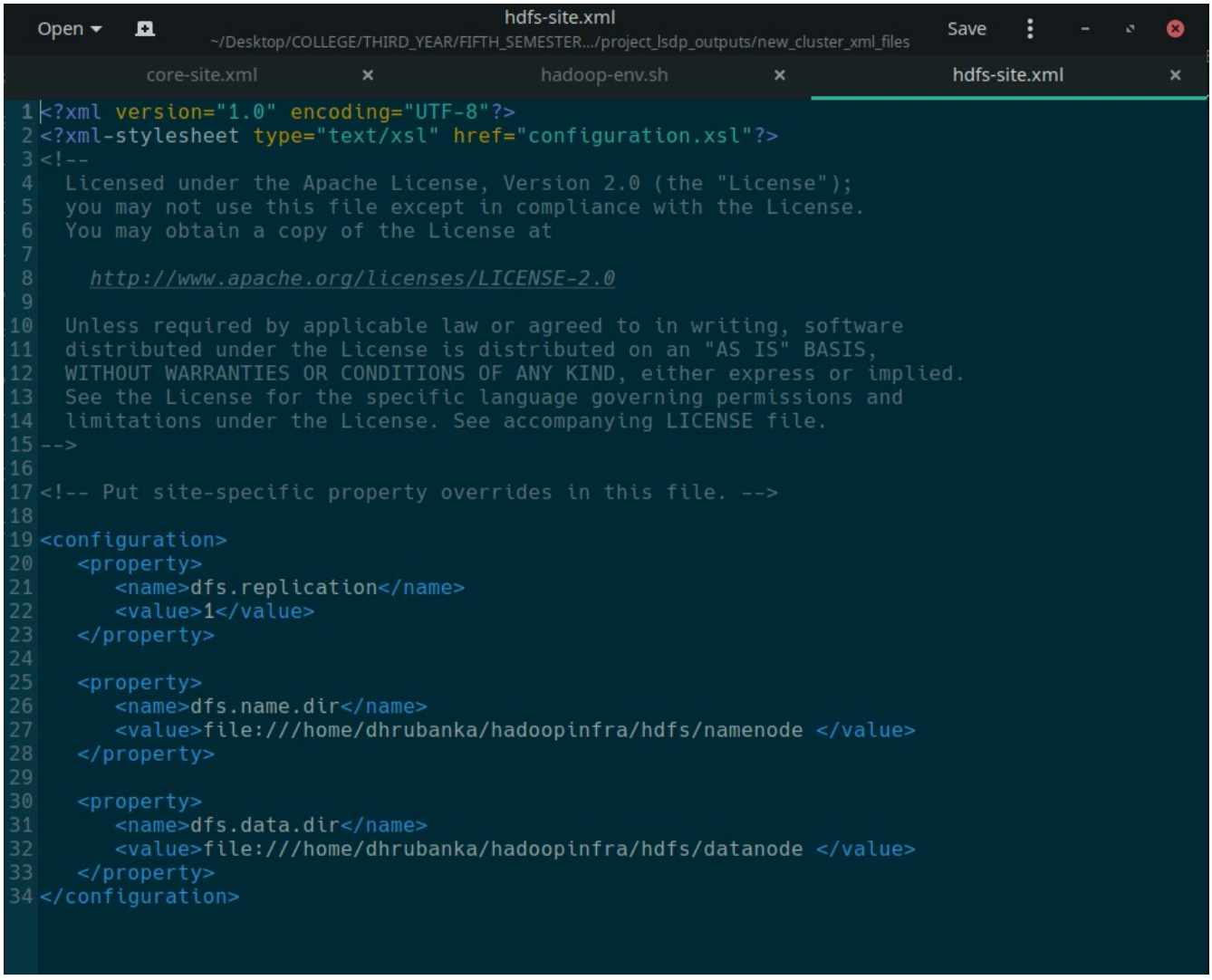
```
INFO:tensorflow:2019-11-07 20:59:24.434115: Step 7980: Validation accuracy = 96.0% (N=100)
I1107 20:59:24.434189 139958439134912 retrain.py:1102] 2019-11-07 20:59:24.434115: Step 7980: Validation accuracy = 96.0% (N=100)
INFO:tensorflow:2019-11-07 20:59:25.136510: Step 7990: Train accuracy = 100.0%
I1107 20:59:25.136571 139958439134912 retrain.py:1084] 2019-11-07 20:59:25.136510: Step 7990: Train accuracy = 100.0%
INFO:tensorflow:2019-11-07 20:59:25.136717: Step 7990: Cross entropy = 0.012224
I1107 20:59:25.136734 139958439134912 retrain.py:1086] 2019-11-07 20:59:25.136717: Step 7990: Cross entropy = 0.012224
INFO:tensorflow:2019-11-07 20:59:25.211638: Step 7990: Validation accuracy = 90.0% (N=100)
I1107 20:59:25.211701 139958439134912 retrain.py:1102] 2019-11-07 20:59:25.211638: Step 7990: Validation accuracy = 90.0% (N=100)
INFO:tensorflow:2019-11-07 20:59:25.865223: Step 7999: Train accuracy = 100.0%
I1107 20:59:25.865285 139958439134912 retrain.py:1084] 2019-11-07 20:59:25.865223: Step 7999: Train accuracy = 100.0%
INFO:tensorflow:2019-11-07 20:59:25.865462: Step 7999: Cross entropy = 0.010826
I1107 20:59:25.865482 139958439134912 retrain.py:1086] 2019-11-07 20:59:25.865462: Step 7999: Cross entropy = 0.010826
INFO:tensorflow:2019-11-07 20:59:25.940358: Step 7999: Validation accuracy = 98.0% (N=100)
I1107 20:59:25.940423 139958439134912 retrain.py:1102] 2019-11-07 20:59:25.940358: Step 7999: Validation accuracy = 98.0% (N=100)
INFO:tensorflow:Final test accuracy = 91.7% (N=24)
I1107 20:59:26.916956 139958439134912 retrain.py:1128] Final test accuracy = 91.7% (N=24)
WARNING:tensorflow:From retrain.py:829: convert_variables_to_constants (from tensorflow.python.framework.graph_util_impl) is deprecated and will be removed in a future version.
Instructions for updating:
Use `tf.compat.v1.graph_util.convert_variables_to_constants`
W1107 20:59:27.025642 139958439134912 deprecation.py:323] From retrain.py:829: convert_variables_to_constants (from tensorflow.python.framework.graph_util_impl) is deprecated and will be removed in a future version.
Instructions for updating:
Use `tf.compat.v1.graph_util.convert_variables_to_constants`
WARNING:tensorflow:From /home/dhrubanka/.local/lib/python3.7/site-packages/tensorflow/python/framework/graph_util_impl.py:270: extract_sub_graph (from tensorflow.python.framework.graph_util_impl) is deprecated and will be removed in a future version.
Instructions for updating:
Use `tf.compat.v1.graph_util.extract_sub_graph`
W1107 20:59:27.025857 139958439134912 deprecation.py:323] From /home/dhrubanka/.local/lib/python3.7/site-packages/tensorflow/python/framework/graph_util_impl.py:270: extract_sub_graph (from tensorflow.python.framework.graph_util_impl) is deprecated and will be removed in a future version.
Instructions for updating:
Use `tf.compat.v1.graph_util.extract_sub_graph`
INFO:tensorflow:Froze 2 variables.
I1107 20:59:27.126499 139958439134912 graph_util_impl.py:311] Froze 2 variables.
INFO:tensorflow:Converted 2 variables to const ops.
I1107 20:59:27.166027 139958439134912 graph_util_impl.py:364] Converted 2 variables to const ops.
(dhrubanka@dhrubanka-pc Image-classification-transfer-learning)$
```

Forming a cluster:

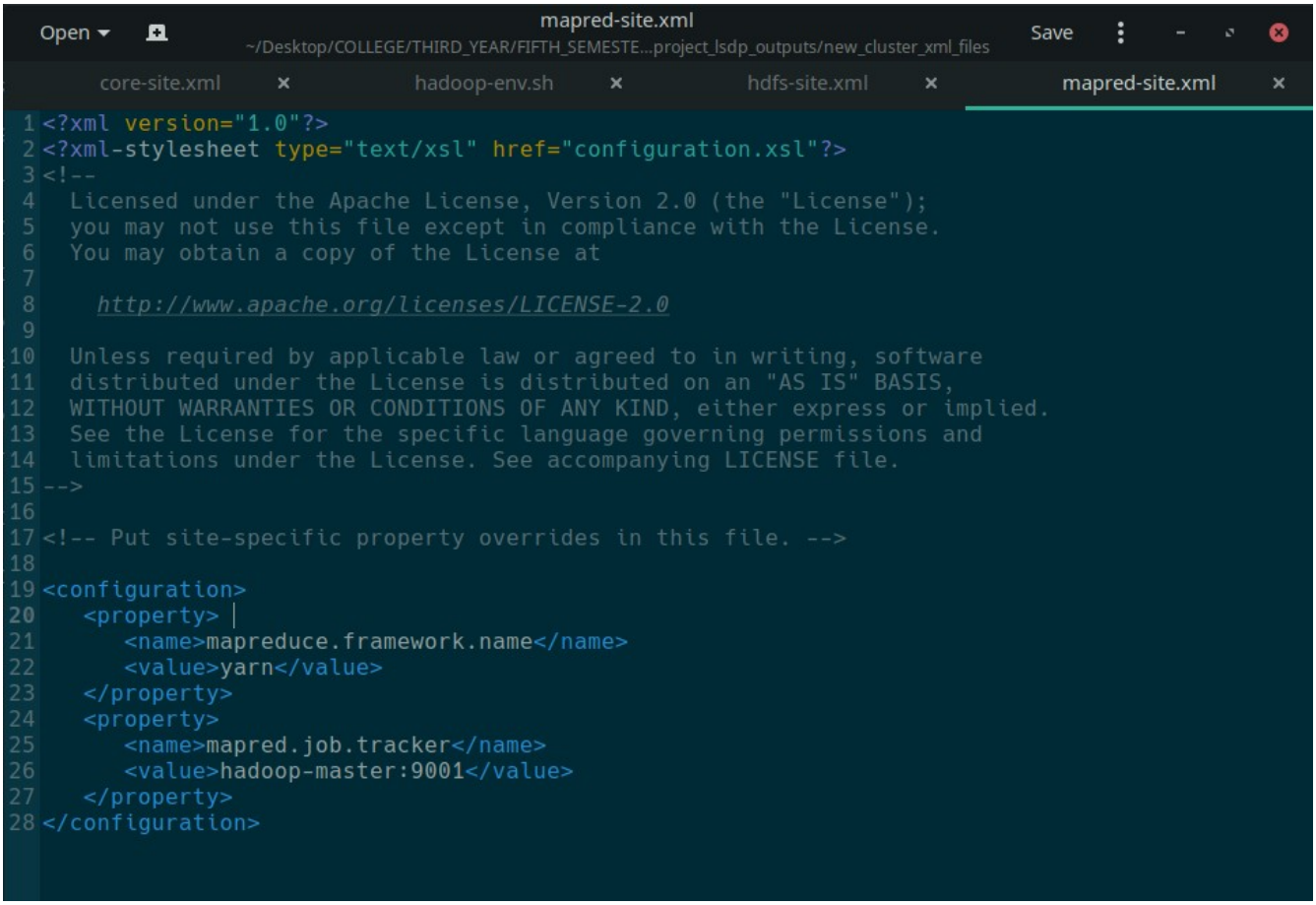
Add the ips of the machines to be put together in the cluster in the */etc /hosts* file.



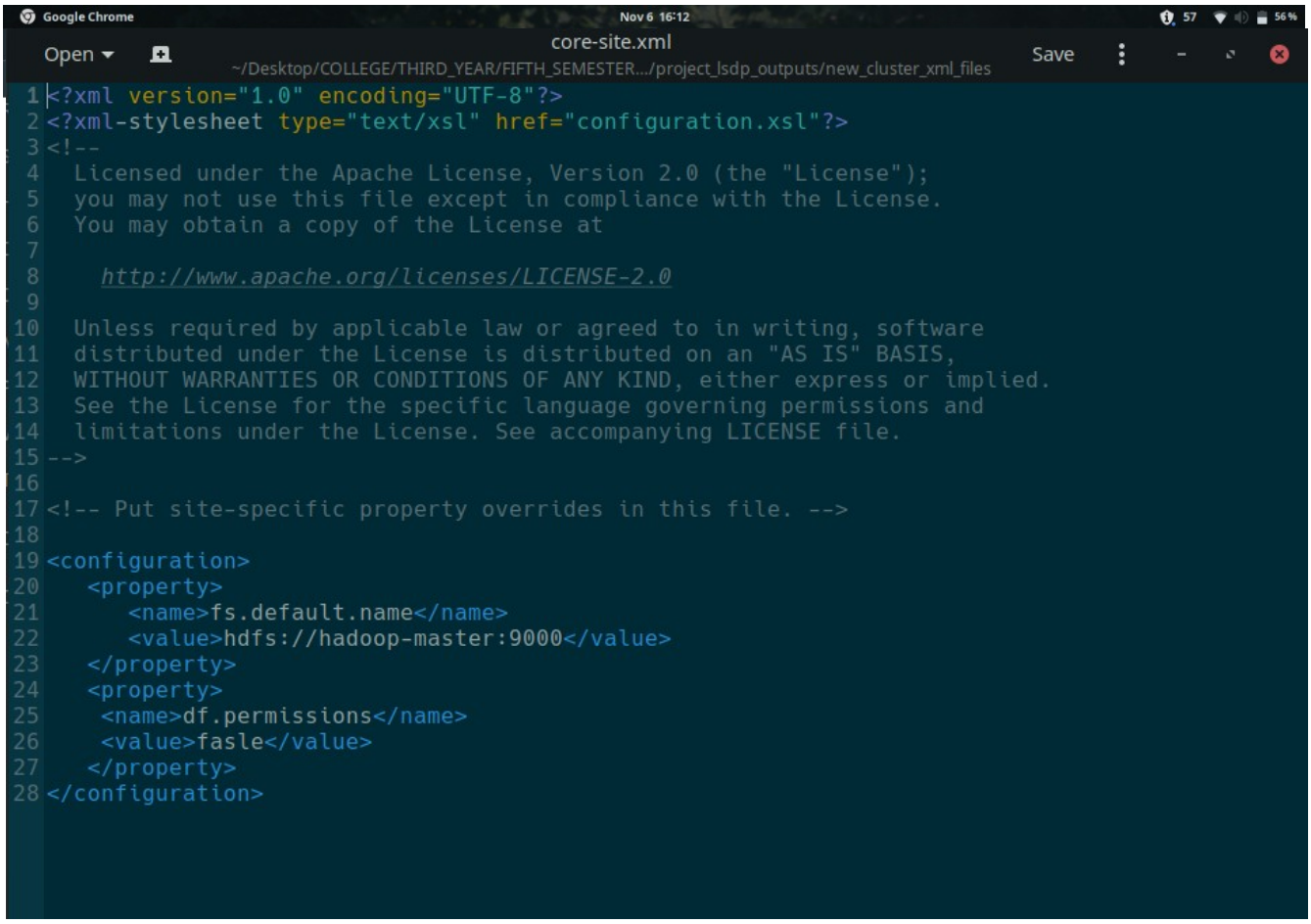
Edit hdfs-site.xml as follows:



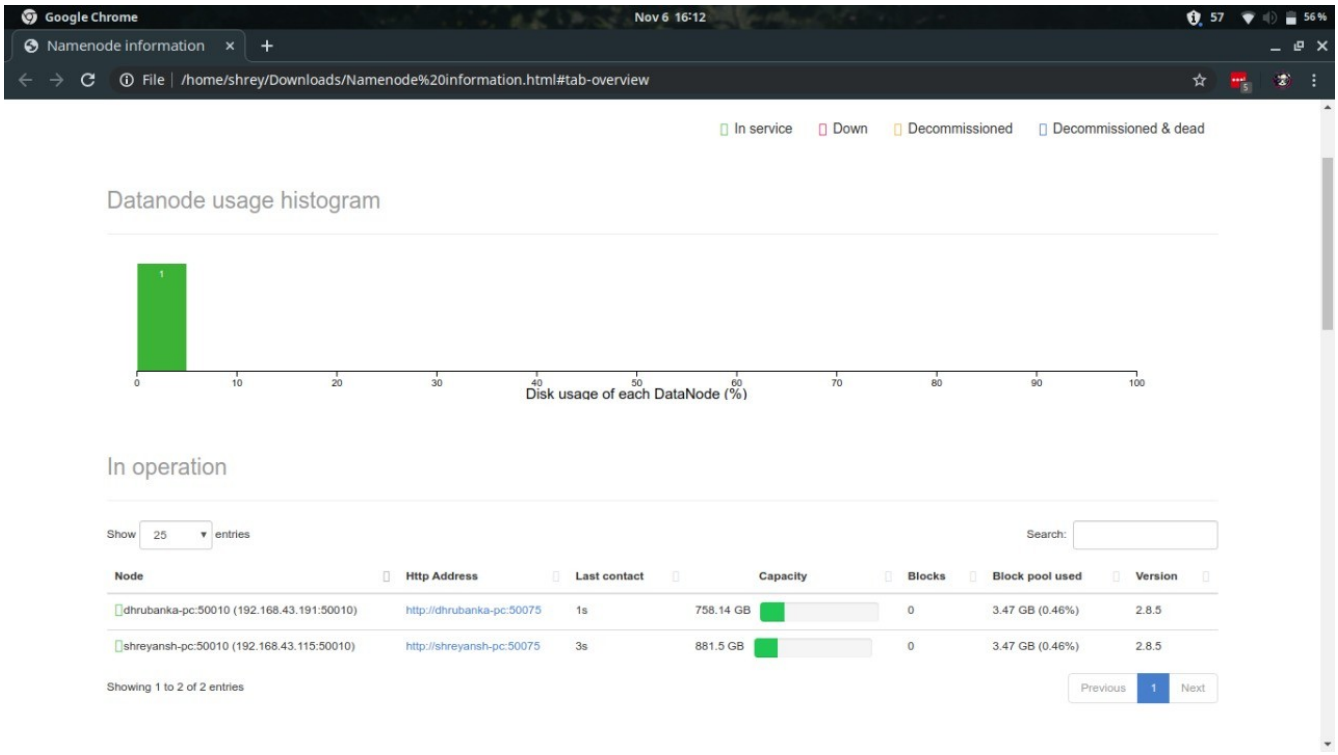
Edit mapred-site.xml as follows:

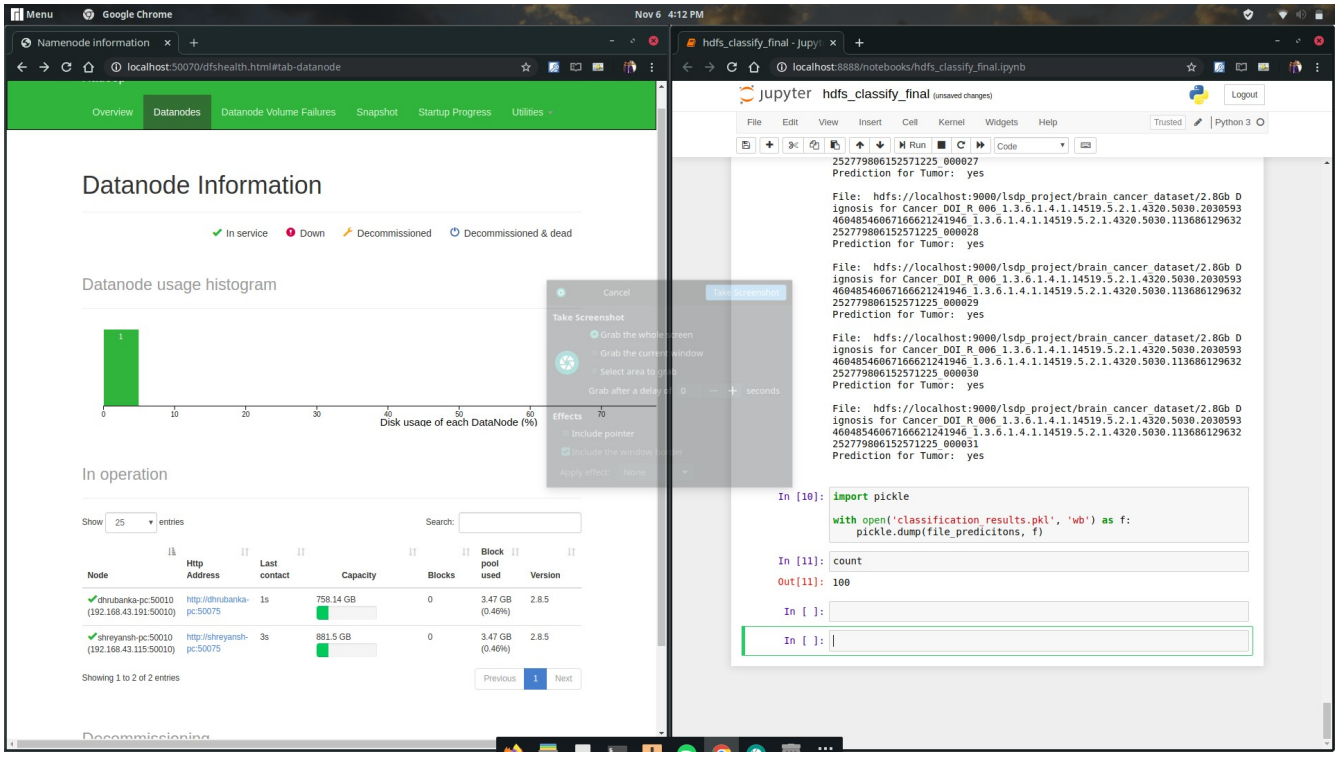


Edit core-site.xml as follows:



Then run **`$ bin/hadoop namenode –format`** in the hadoop bin folder


Google Chrome
Nov 6 16:12
Namenode information
File | /home/shrey/Downloads/Namenode%20information.html#tab-overview
In service
Down
Decommissioned
Decommissioned & dead
Datanode usage histogram
Disk usage of each DataNode (%)
In operation
Show 25 entries
Search:
Node
Http Address
Last contact
Capacity
Blocks
Block pool used
Version
dhrubanka-pc:50010 (192.168.43.191:50010)
http://dhrubanka-pc:50075
1s
758.14 GB
0
3.47 GB (0.46%)
2.8.5
shreyansh-pc:50010 (192.168.43.115:50010)
http://shreyansh-pc:50075
3s
881.5 GB
0
3.47 GB (0.46%)
2.8.5
Showing 1 to 2 of 2 entries
Previous
1
Next


Menu
Google Chrome
Nov 6 4:12 PM
Namenode information
localhost:50070/dfshealth.html#tab-datanode
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities
Datanode Information
In service
Down
Decommissioned
Decommissioned & dead
Datanode usage histogram
Disk usage of each DataNode (%)
In operation
Show 25 entries
Search:
Node
Http Address
Last contact
Capacity
Blocks
Block pool used
Version
dhrubanka-pc:50010 (192.168.43.191:50010)
http://dhrubanka-pc:50075
1s
758.14 GB
0
3.47 GB (0.46%)
2.8.5
shreyansh-pc:50010 (192.168.43.115:50010)
http://shreyansh-pc:50075
3s
881.5 GB
0
3.47 GB (0.46%)
2.8.5
Showing 1 to 2 of 2 entries
Previous
1
Next
Decommissioning
hdfs_classify_final - Jupyter
localhost:8888/notebooks/hdfs_classify_final.ipynb
jupyter hdfs_classify_final (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code
Prediction for Tumor: yes
In [10]:
import pickle
with open('classification_results.pkl', 'wb') as f:
pickle.dump(file_predicitons, f)
In [11]:
count
Out[11]:
100
In [ ]:
In [ ]:

RESULT

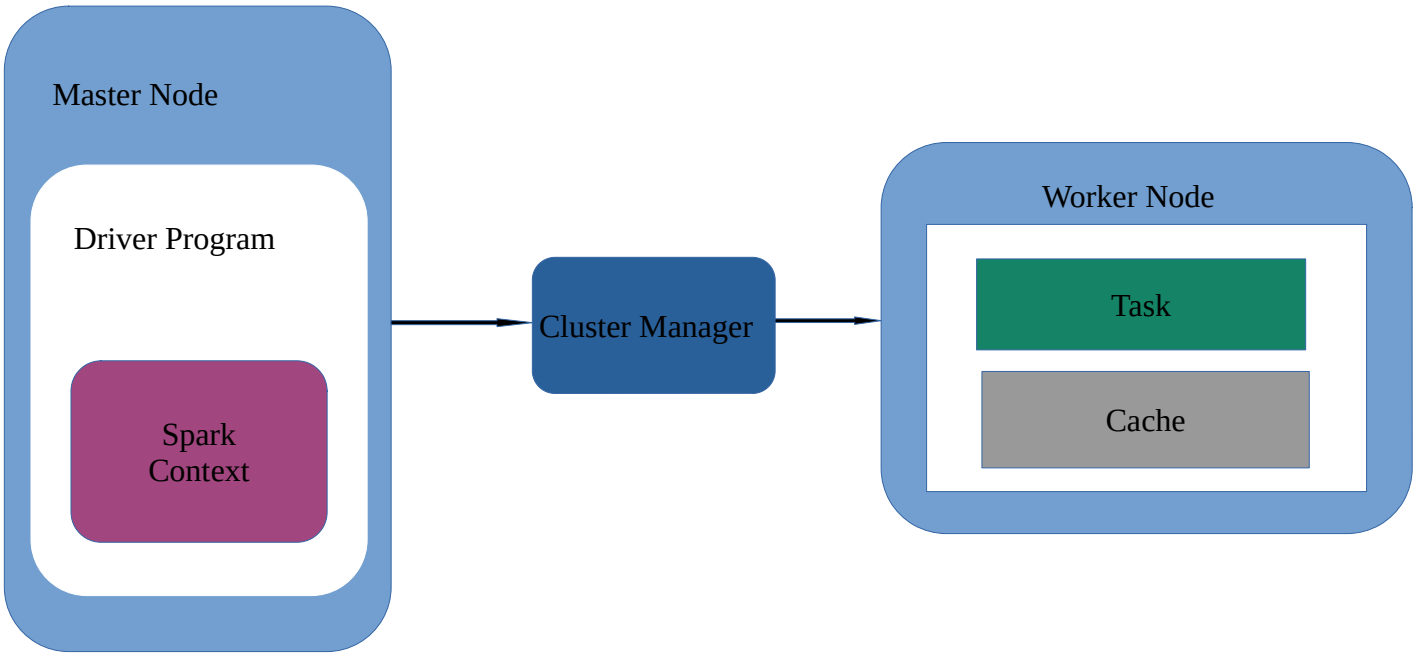
The classier ran on the instance of PySpark and classifed the input images from the dataset one by one. The images were stored in the HDFS in form of binary files. These were then extracted to the PySpark instance using SparkContext. The tumor prediction of the images were given as output by the classifier. The script classified the images into Yes or No categories using the trained classifier. This data was available in the hdfs and can be accessed by all nodes of the cluster.

Model	Training Time	Training accuracy	Testing accuracy
Google Incpetion v3	20 minutes	97%	96%

The following hardware setup was used in the cluster. It led to fast computation with 200 images classified in 3.5 minutes.

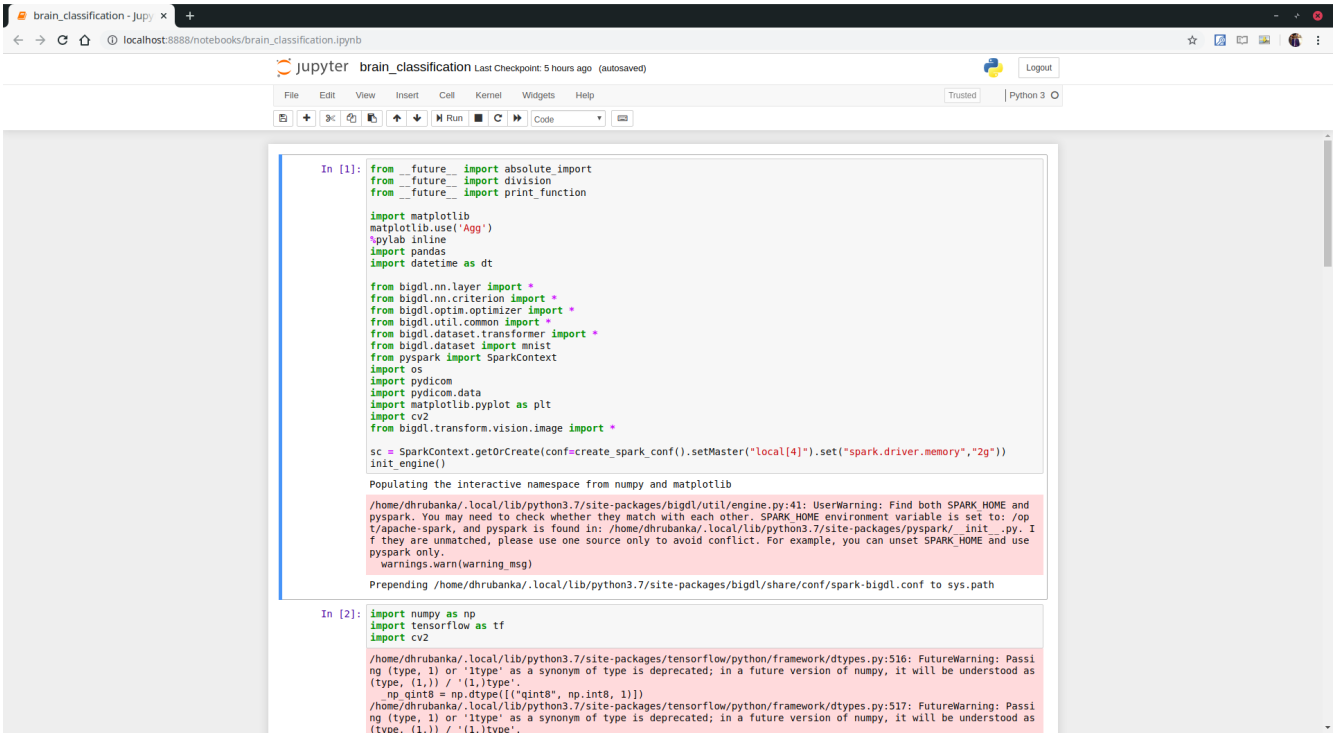
Machine	Processor	GPU	RAM	OS	HARD DISK
Master	Intel i5	AMD Radeon 530	8 GB	Manjaro Arch-Linux	1 TB
Slave1	Intel i7	Nvidia GTX 1050	12 GB	Ubuntu 18.04	1 TB

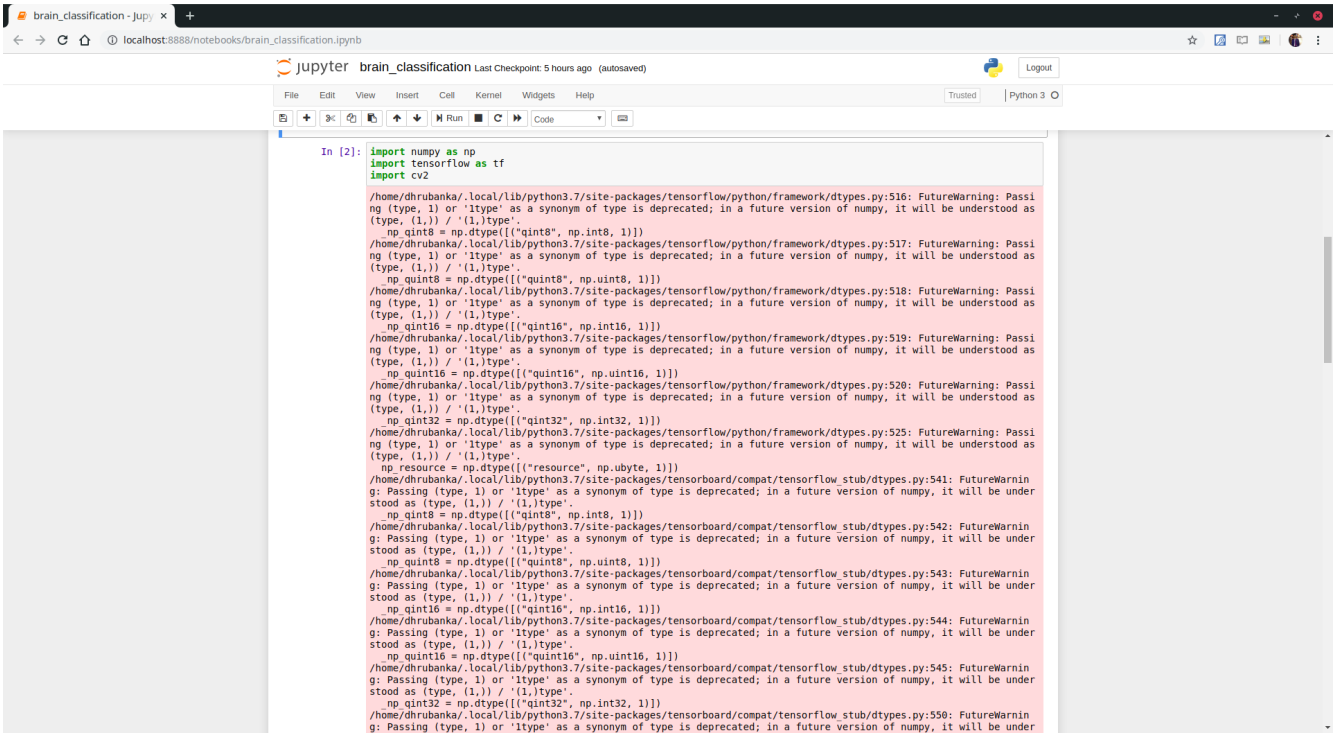
A cluster architechture was established as follows :

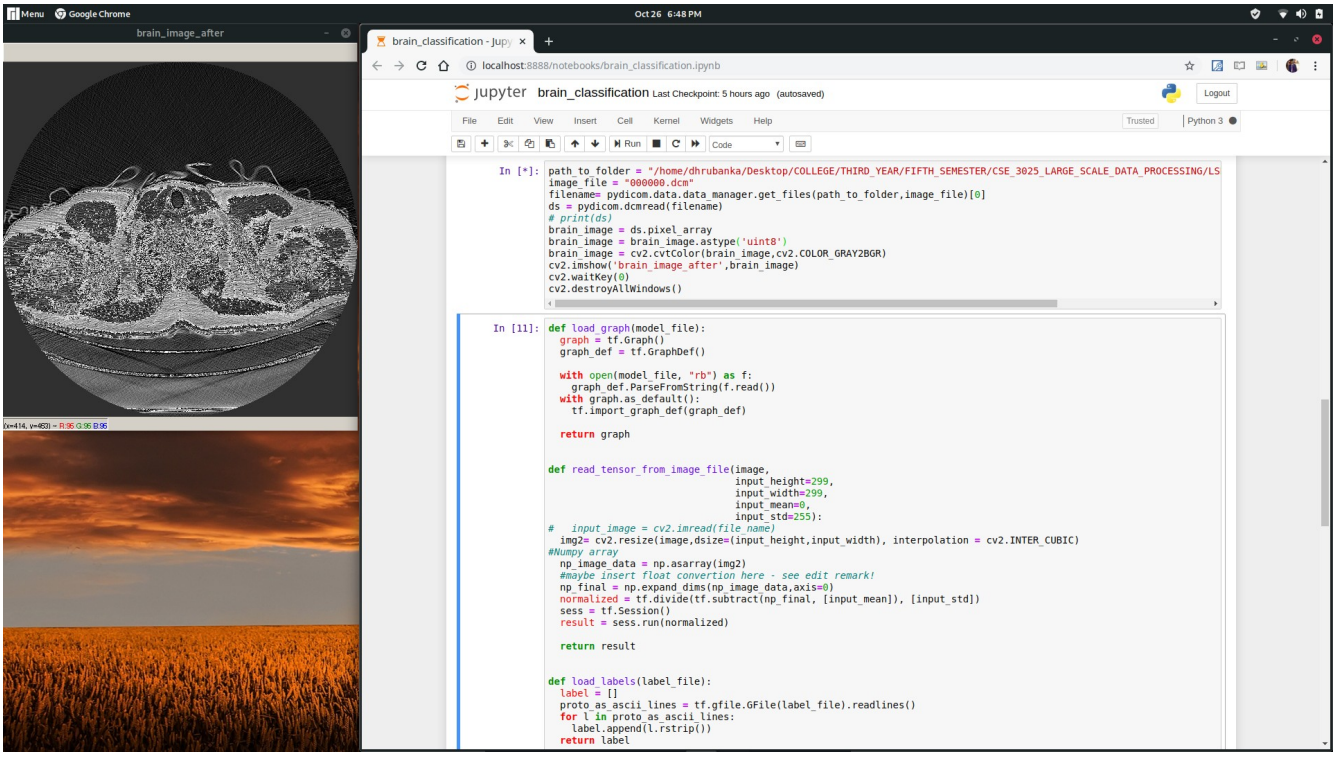


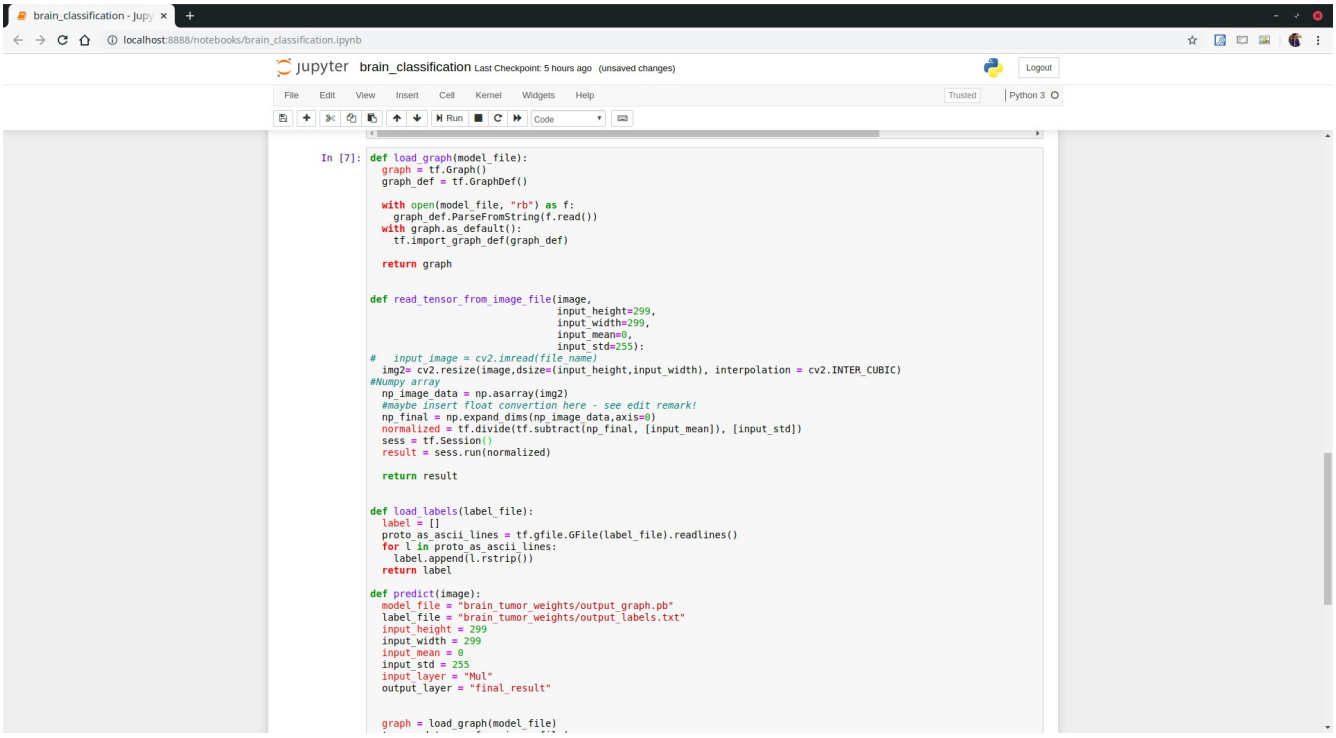
OUTPUTS

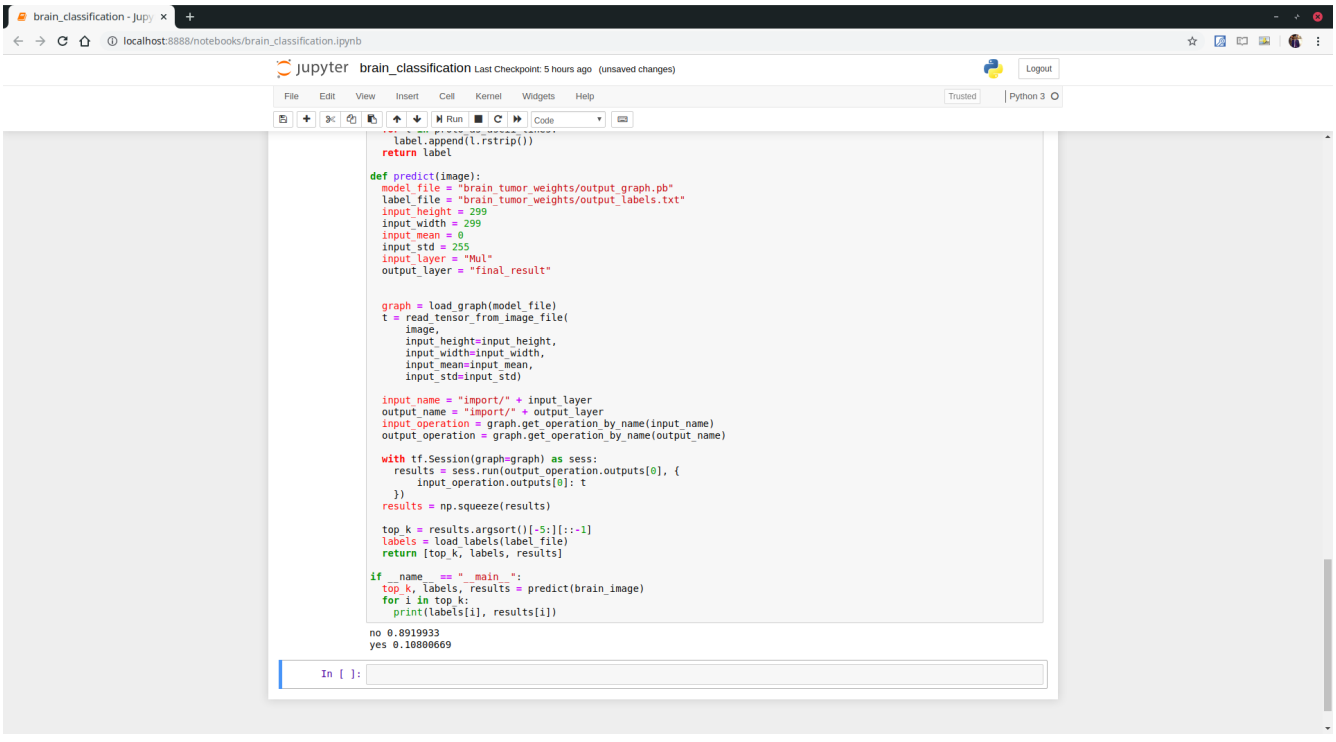
For single image :



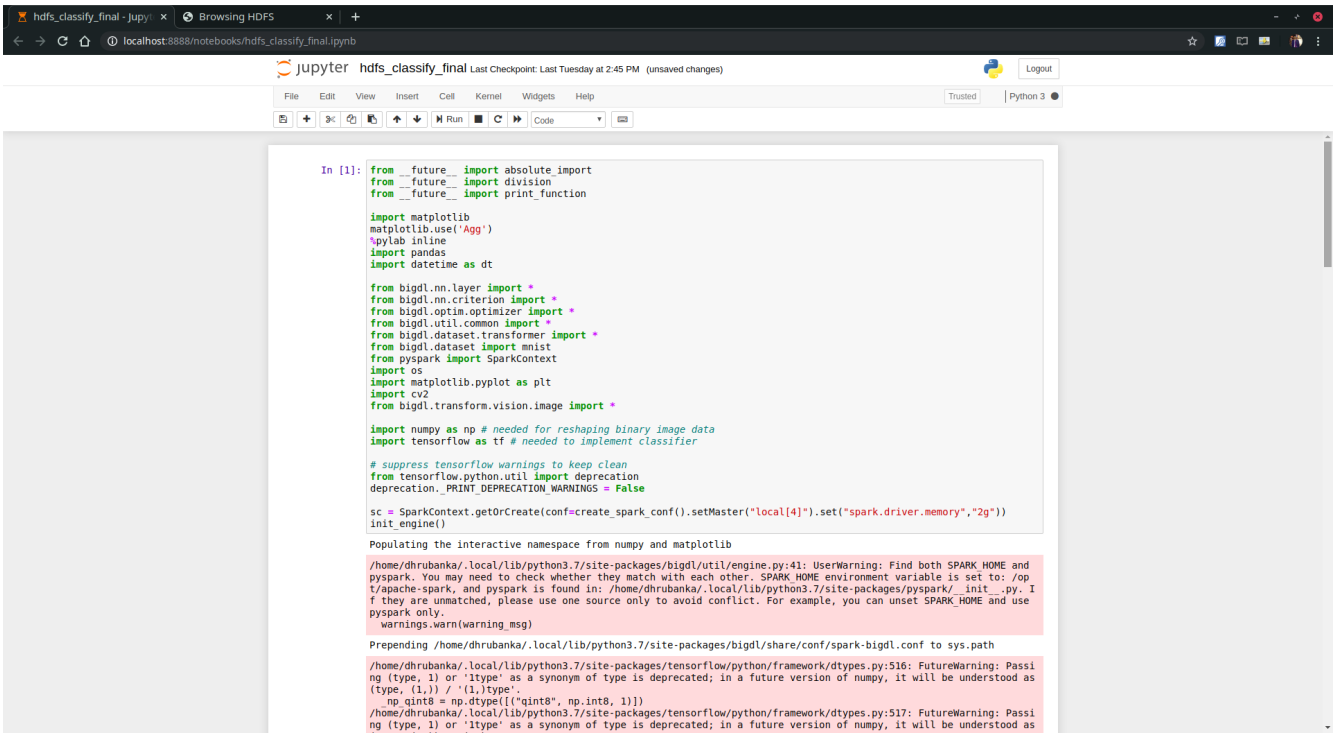

brain_classification - Jupy
localhost:8888/notebooks/brain_classification.ipynb
jupyter brain_classification Last Checkpoint: 5 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [2]:
import numpy as np
import tensorflow as tf
import cv2
/home/dhrubanka/.local/lib/python3.7/site-packages/tensorflow/python/framework/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_quint8 = np.dtype([("quint8", 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_quint16 = np.dtype([("quint16", 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/tensorboard/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/tensorboard/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_quint8 = np.dtype([("quint8", np.uint8, 1)])
/home/dhrubanka/.local/lib/python3.7/site-packages/tensorboard/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/tensorboard/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_quint16 = np.dtype([("quint16", np.uint16, 1)])
/home/dhrubanka/.local/lib/python3.7/site-packages/tensorboard/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/tensorboard/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

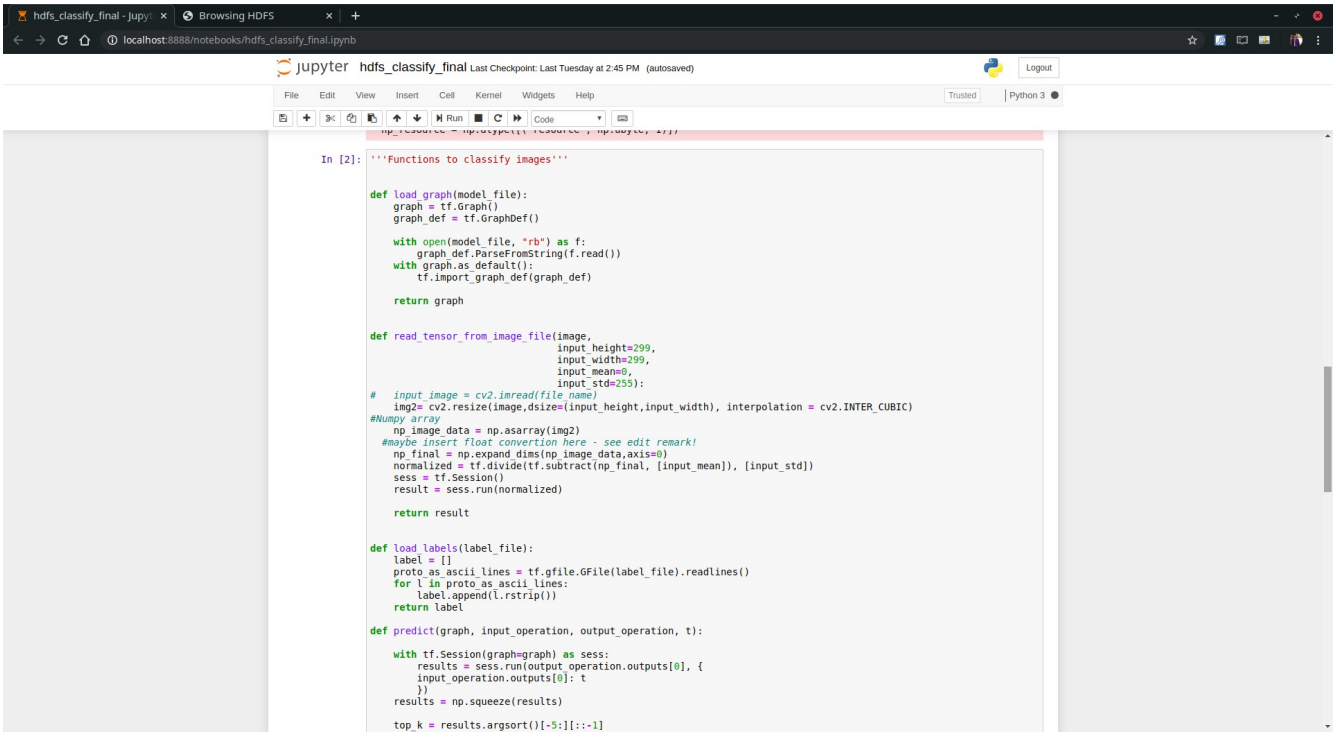

Menu Google Chrome
Oct 26 6:48 PM
brain_image_after
brain_classification - Jupy
localhost:8888/notebooks/brain_classification.ipynb
jupyter brain_classification Last Checkpoint: 5 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [*]:
path_to_folder = "/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 convertion 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

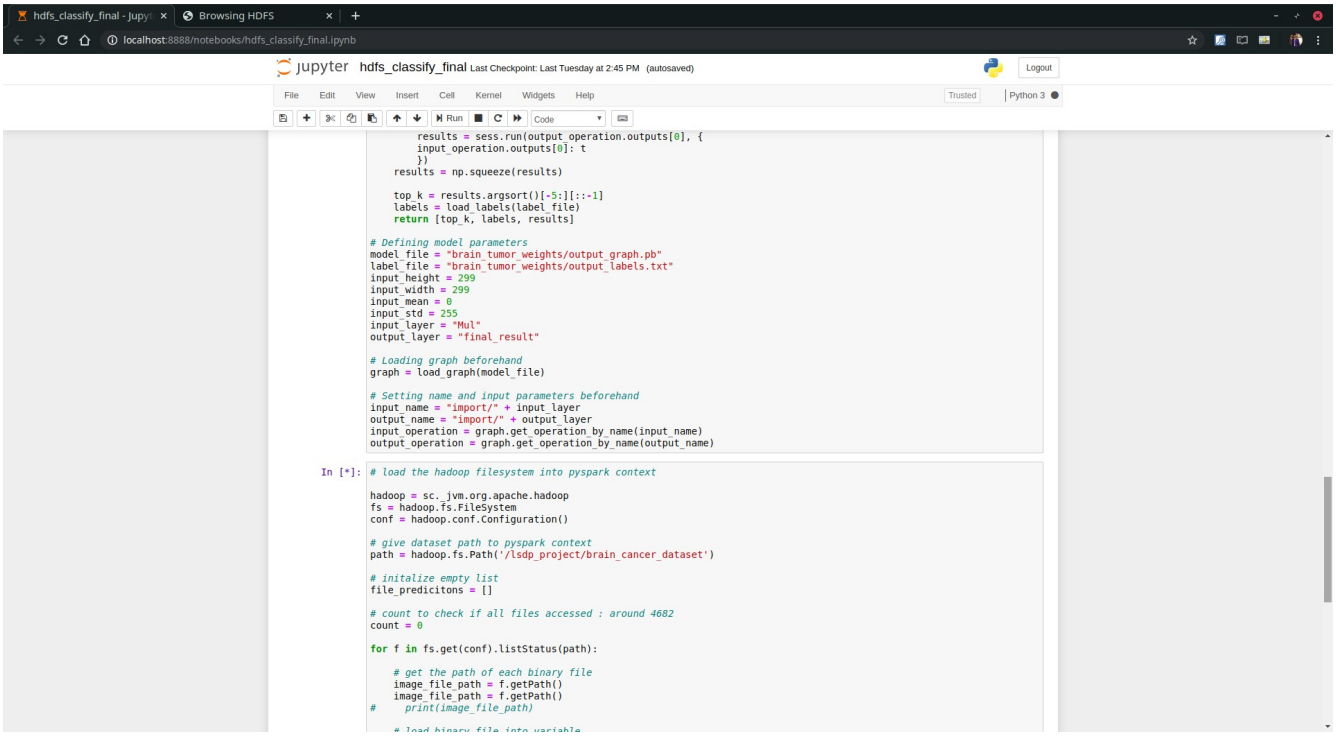

brain_classification - Jupy
localhost:8888/notebooks/brain_classification.ipynb
jupyter brain_classification Last Checkpoint: 5 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [7]:
def load_graph(model_file):
graph = tf.Graph()
graph_def = 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,dsize=(input_height,input_width), interpolation = cv2.INTER_CUBIC)
#Numpy_array
np_image_data = np.asarray(img2)
#maybe insert float convertion 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)


brain_classification - Jupy
localhost:8888/notebooks/brain_classification.ipynb
jupyter brain_classification Last Checkpoint: 5 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
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()[-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 0.10880669
In [ ]:

For multiple images :



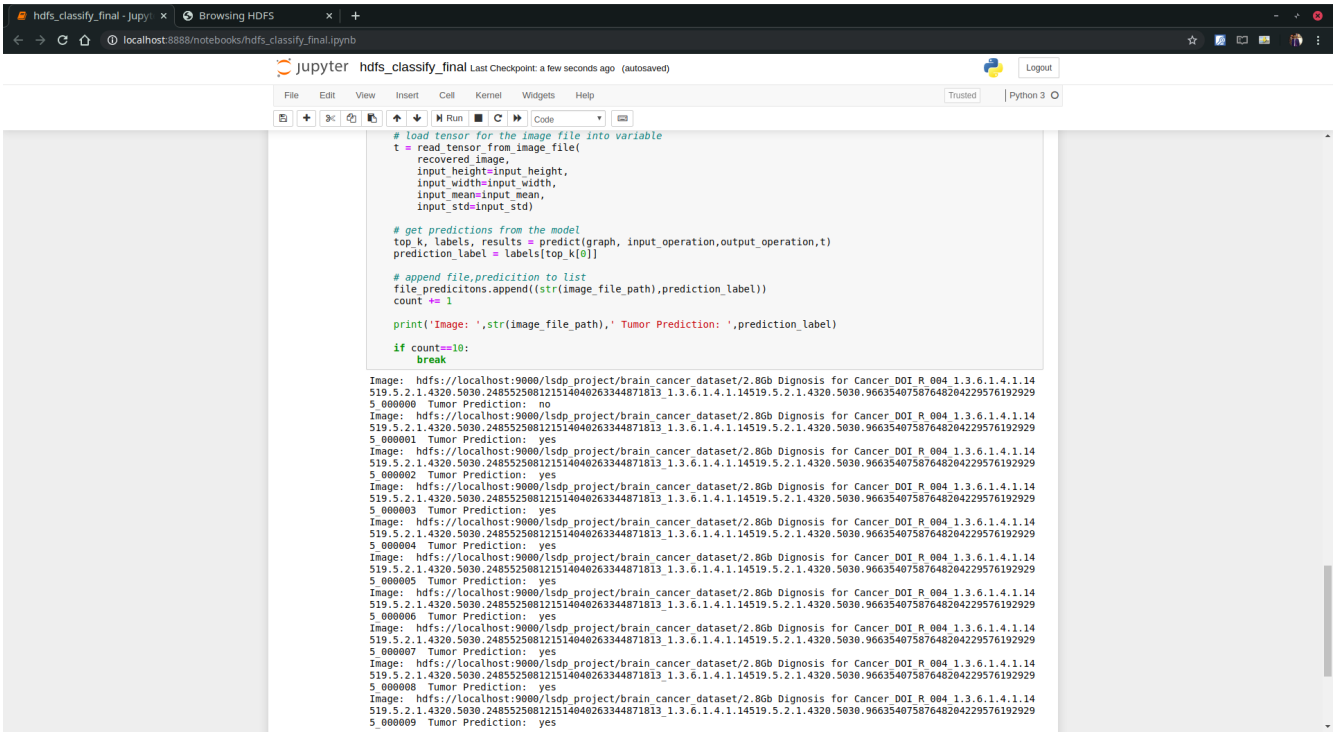




```
        results = sess.run(output_operation.outputs[0], {
        input_operation.outputs[0]: t
        })
    results = np.squeeze(results)

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

# Defining model parameters
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"

# Loading graph beforehand
graph = load_graph(model_file)

# Setting name and input parameters beforehand
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)
```

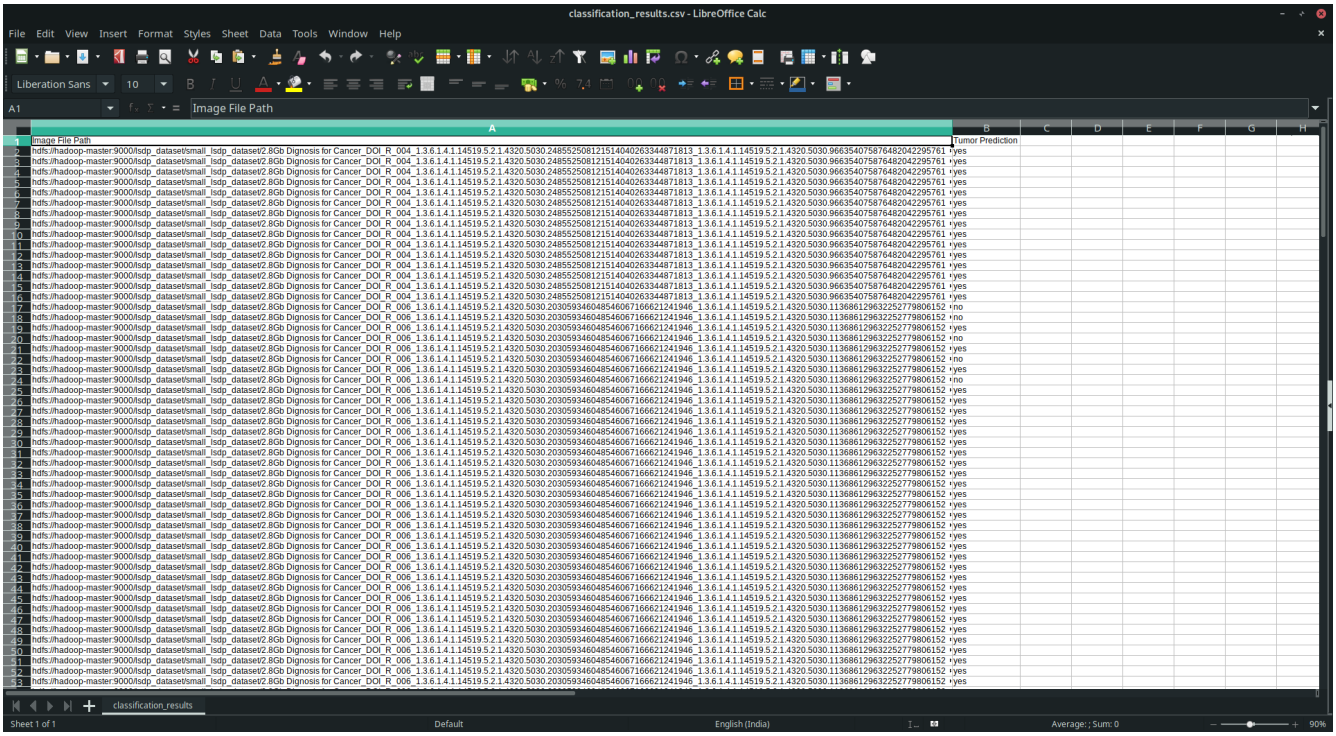
In [*]:

```
# load the hadoop filesystem into pyspark context

hadoop = sc._jvm.org.apache.hadoop
fs = hadoop.fs.FileSystem
conf = hadoop.conf.Configuration()

# give dataset path to pyspark context
path = hadoop.fs.Path('/lsdp_project/brain_cancer_dataset')

# initalize empty list
file_predicitons = []

# count to check if all files accessed : around 4682
count = 0

for f in fs.get(conf).listStatus(path):

    # get the path of each binary file
    image_file_path = f.getPath()
    image_file_path = f.getPath()
#     print(image_file_path)

    # load binary file into variable
```



```
    # load tensor for the image file into variable
    t = read_tensor_from_image_file(
        recovered_image,
        input_height=input_height,
        input_width=input_width,
        input_mean=input_mean,
        input_std=input_std)

    # get predictions from the model
    top_k, labels, results = predict(graph, input_operation,output_operation,t)
    prediction_label = labels[top_k[0]]

    # append file,predicition to list
    file_predicitons.append((str(image_file_path),prediction_label))
    count += 1

    print('Image: ',str(image_file_path),' Tumor Prediction: ',prediction_label)

    if count==10:
        break
```

```
Image:  hdfs://localhost:9000/lsdp_project/brain_cancer_dataset/2.8Gb Dignosis for Cancer_DOI_R_004_1.3.6.1.4.1.14519.5.2.1.4320.5030.248552508121514040263344871813_1.3.6.1.4.1.14519.5.2.1.4320.5030.966354075876482042295761929295_000000  Tumor Prediction:  no
Image:  hdfs://localhost:9000/lsdp_project/brain_cancer_dataset/2.8Gb Dignosis for Cancer_DOI_R_004_1.3.6.1.4.1.14519.5.2.1.4320.5030.248552508121514040263344871813_1.3.6.1.4.1.14519.5.2.1.4320.5030.966354075876482042295761929295_000001  Tumor Prediction:  yes
Image:  hdfs://localhost:9000/lsdp_project/brain_cancer_dataset/2.8Gb Dignosis for Cancer_DOI_R_004_1.3.6.1.4.1.14519.5.2.1.4320.5030.248552508121514040263344871813_1.3.6.1.4.1.14519.5.2.1.4320.5030.966354075876482042295761929295_000002  Tumor Prediction:  yes
Image:  hdfs://localhost:9000/lsdp_project/brain_cancer_dataset/2.8Gb Dignosis for Cancer_DOI_R_004_1.3.6.1.4.1.14519.5.2.1.4320.5030.248552508121514040263344871813_1.3.6.1.4.1.14519.5.2.1.4320.5030.966354075876482042295761929295_000003  Tumor Prediction:  yes
Image:  hdfs://localhost:9000/lsdp_project/brain_cancer_dataset/2.8Gb Dignosis for Cancer_DOI_R_004_1.3.6.1.4.1.14519.5.2.1.4320.5030.248552508121514040263344871813_1.3.6.1.4.1.14519.5.2.1.4320.5030.966354075876482042295761929295_000004  Tumor Prediction:  yes
Image:  hdfs://localhost:9000/lsdp_project/brain_cancer_dataset/2.8Gb Dignosis for Cancer_DOI_R_004_1.3.6.1.4.1.14519.5.2.1.4320.5030.248552508121514040263344871813_1.3.6.1.4.1.14519.5.2.1.4320.5030.966354075876482042295761929295_000005  Tumor Prediction:  yes
Image:  hdfs://localhost:9000/lsdp_project/brain_cancer_dataset/2.8Gb Dignosis for Cancer_DOI_R_004_1.3.6.1.4.1.14519.5.2.1.4320.5030.248552508121514040263344871813_1.3.6.1.4.1.14519.5.2.1.4320.5030.966354075876482042295761929295_000006  Tumor Prediction:  yes
Image:  hdfs://localhost:9000/lsdp_project/brain_cancer_dataset/2.8Gb Dignosis for Cancer_DOI_R_004_1.3.6.1.4.1.14519.5.2.1.4320.5030.248552508121514040263344871813_1.3.6.1.4.1.14519.5.2.1.4320.5030.966354075876482042295761929295_000007  Tumor Prediction:  yes
Image:  hdfs://localhost:9000/lsdp_project/brain_cancer_dataset/2.8Gb Dignosis for Cancer_DOI_R_004_1.3.6.1.4.1.14519.5.2.1.4320.5030.248552508121514040263344871813_1.3.6.1.4.1.14519.5.2.1.4320.5030.966354075876482042295761929295_000008  Tumor Prediction:  yes
Image:  hdfs://localhost:9000/lsdp_project/brain_cancer_dataset/2.8Gb Dignosis for Cancer_DOI_R_004_1.3.6.1.4.1.14519.5.2.1.4320.5030.248552508121514040263344871813_1.3.6.1.4.1.14519.5.2.1.4320.5030.966354075876482042295761929295_000009  Tumor Prediction:  yes
```



CONCLUSION

1. Hadoop and PySpark were setup in the machines.
2. BigDL was setup in all the machines.
3. The dataset was uploaded to the HDFS.
4. An instance of PySpark was started and integrated with BigDL.
5. The image classifier was trained using inception with validation metric of .
6. The classifier model was loaded into the PySpark instance using BigDL.
7. The dataset was iterated through and each image was passed to the classifier.
8. The classifier classfied the images and the image path along with prediction were listed.
9. The image paths along with their predcition were written to a pickle file.
10. A hadoop cluster was setup.
11. The script was run on the cluster and speed improvement was noticed.