

Configuration Manual

MSc Research Project
Data Analytics

Aboli Kapadnis
Student ID: x19218451

School of Computing
National College of Ireland

Supervisor: Prof. Hicham Rifai

**National College of Ireland
Project Submission Sheet
School of Computing**



Student Name:	Aboli Kapadnis
Student ID:	x19218451
Programme:	MSc Data Analytics
Year:	2020-21
Module:	MSc Research Project
Supervisor:	Prof. Hicham Rifai
Submission Due Date:	16/08/2021
Project Title:	Configuration Manual
Word Count:	405
Page Count:	9

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	17th September 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission , to each project (including multiple copies).	<input type="checkbox"/>
You must ensure that you retain a HARD COPY of the project , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Aboli Kapadnis
x19218451

1 Introduction

The objective for configuration manual is provide details of the system setup, software specification, hardware platforms and steps for the implementing the research project: Brain Tumour Detection using Deep Learning with AlexNet

2 System Configuration

2.1 Hardware

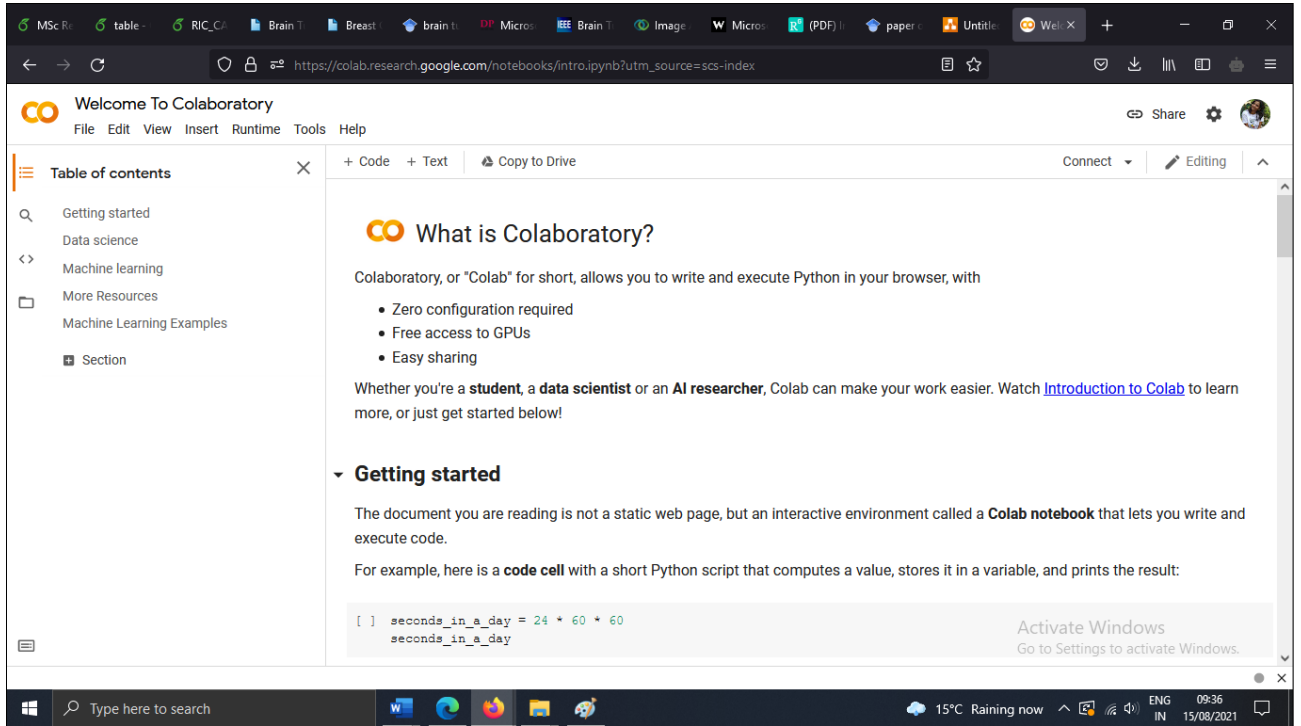
OS	Windows 10
RAM	Minimum 8GB (2.14 GB from Colaboratory)
Hard Disk Space	Minimum 100GB (100 GB drive space)

2.2 Software

Programming Language Tools	Google Colaboratory (Cloud based Jupyter notebook environment), Python version 3, Microsoft Excel, Overleaf
Web Browser	Google Chrome or Mozilla Firefox
Email	Access to a Gmail account

2.2.1 Google colab environment:

CNN and other models consume long time to process the visuals. The experiment is conducted on a Google Collaboratory environment with a 100 GB hard drive, 12.72 GB RAM, and a 48.97 GB run-time GPU. Because all models have additional layers, they take longer to execute on big picture data. The use of a GPU in real time speeds up the execution of CNN and AlexNet. The python libraries Keras and TensorFlow are utilized to implement all three models. On Google Collaboratory Notebook, Python version 3 is used. Google Drive has been used to store the data. Numpy and Keras packages from Python are used for image normalization, argumentation, cropping and up-sampling. 1. Sign in to Google colab with your gmail.



3 Project Development

This project follows a Cross Industry Standard Process for Data Mining (CRISP- DM) Methodology and the details of the research process such as data collection, processing techniques and algorithms are discussed below.

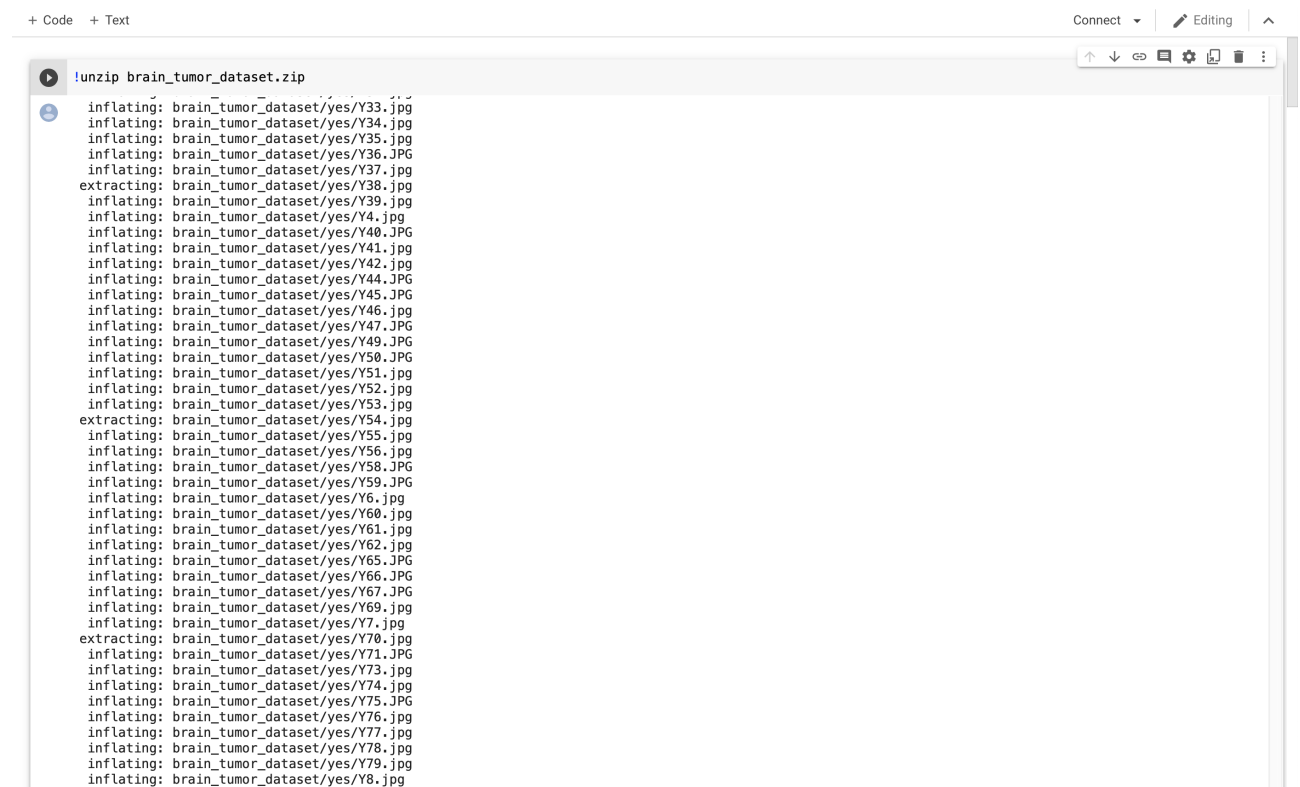
3.1 Google Colaboratory Environment Setup

The Google Colaboratory environment is used for perform the experiments. Google drive access is required for accessing Google Colaboratory and hence a valid Gmail account is essential. Once the URL is followed, a code is visible using which access will be granted to use the Google Drive for mounting the files and data.

We have used Python 3 notebooks for all Experiments

3.2 Data Preparation

Step 1: Unzipping the dataset

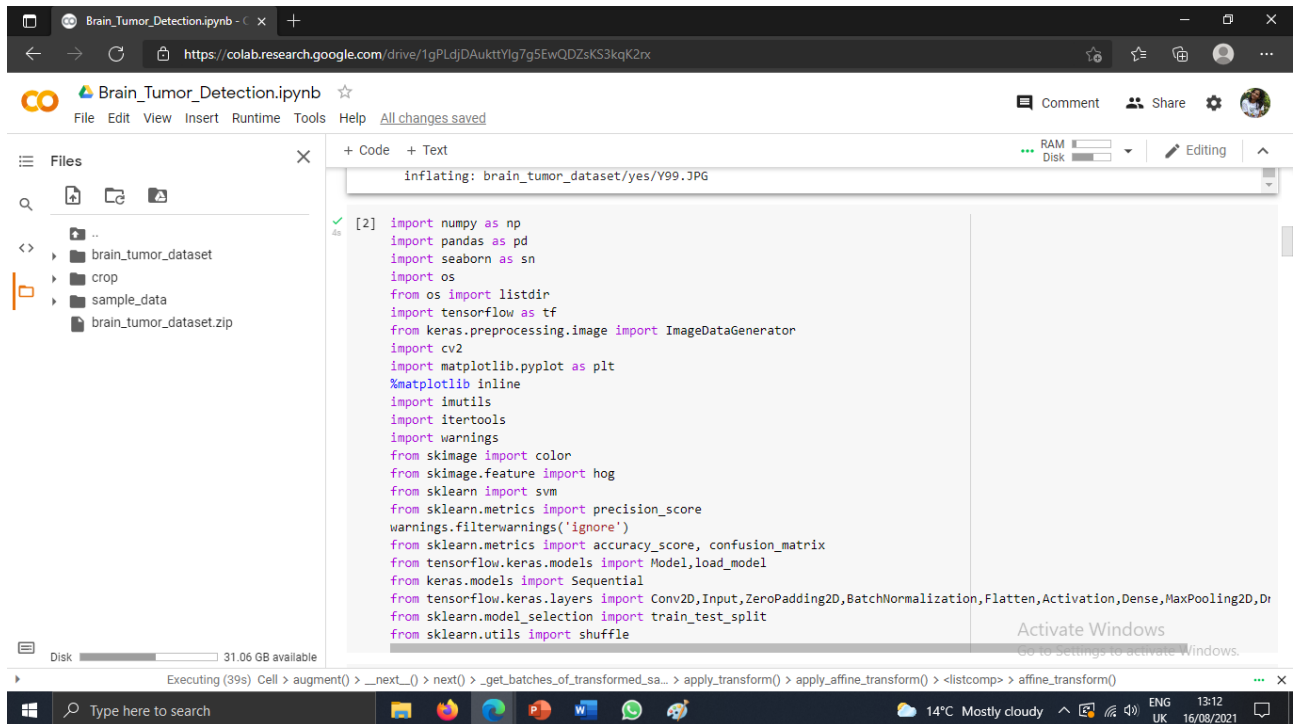


```
+ Code + Text Connect Editing ^
```

```
!unzip brain_tumor_dataset.zip
```

```
inflating: brain_tumor_dataset/yes/Y33.jpg
inflating: brain_tumor_dataset/yes/Y34.jpg
inflating: brain_tumor_dataset/yes/Y35.jpg
inflating: brain_tumor_dataset/yes/Y36.JPG
inflating: brain_tumor_dataset/yes/Y37.jpg
extracting: brain_tumor_dataset/yes/Y38.jpg
inflating: brain_tumor_dataset/yes/Y39.jpg
inflating: brain_tumor_dataset/yes/Y4.jpg
inflating: brain_tumor_dataset/yes/Y40.JPG
inflating: brain_tumor_dataset/yes/Y41.jpg
inflating: brain_tumor_dataset/yes/Y42.jpg
inflating: brain_tumor_dataset/yes/Y44.JPG
inflating: brain_tumor_dataset/yes/Y45.JPG
inflating: brain_tumor_dataset/yes/Y46.jpg
inflating: brain_tumor_dataset/yes/Y47.JPG
inflating: brain_tumor_dataset/yes/Y49.JPG
inflating: brain_tumor_dataset/yes/Y50.JPG
inflating: brain_tumor_dataset/yes/Y51.jpg
inflating: brain_tumor_dataset/yes/Y52.jpg
inflating: brain_tumor_dataset/yes/Y53.jpg
extracting: brain_tumor_dataset/yes/Y54.jpg
inflating: brain_tumor_dataset/yes/Y55.jpg
inflating: brain_tumor_dataset/yes/Y56.jpg
inflating: brain_tumor_dataset/yes/Y58.JPG
inflating: brain_tumor_dataset/yes/Y59.JPG
inflating: brain_tumor_dataset/yes/Y6.jpg
inflating: brain_tumor_dataset/yes/Y60.jpg
inflating: brain_tumor_dataset/yes/Y61.jpg
inflating: brain_tumor_dataset/yes/Y62.jpg
inflating: brain_tumor_dataset/yes/Y65.JPG
inflating: brain_tumor_dataset/yes/Y66.JPG
inflating: brain_tumor_dataset/yes/Y67.JPG
inflating: brain_tumor_dataset/yes/Y69.jpg
inflating: brain_tumor_dataset/yes/Y7.jpg
extracting: brain_tumor_dataset/yes/Y70.jpg
inflating: brain_tumor_dataset/yes/Y71.JPG
inflating: brain_tumor_dataset/yes/Y73.jpg
inflating: brain_tumor_dataset/yes/Y74.jpg
inflating: brain_tumor_dataset/yes/Y75.JPG
inflating: brain_tumor_dataset/yes/Y76.jpg
inflating: brain_tumor_dataset/yes/Y77.jpg
inflating: brain_tumor_dataset/yes/Y78.jpg
inflating: brain_tumor_dataset/yes/Y79.jpg
inflating: brain_tumor_dataset/yes/Y8.jpg
```

Step 2: Import Libraries



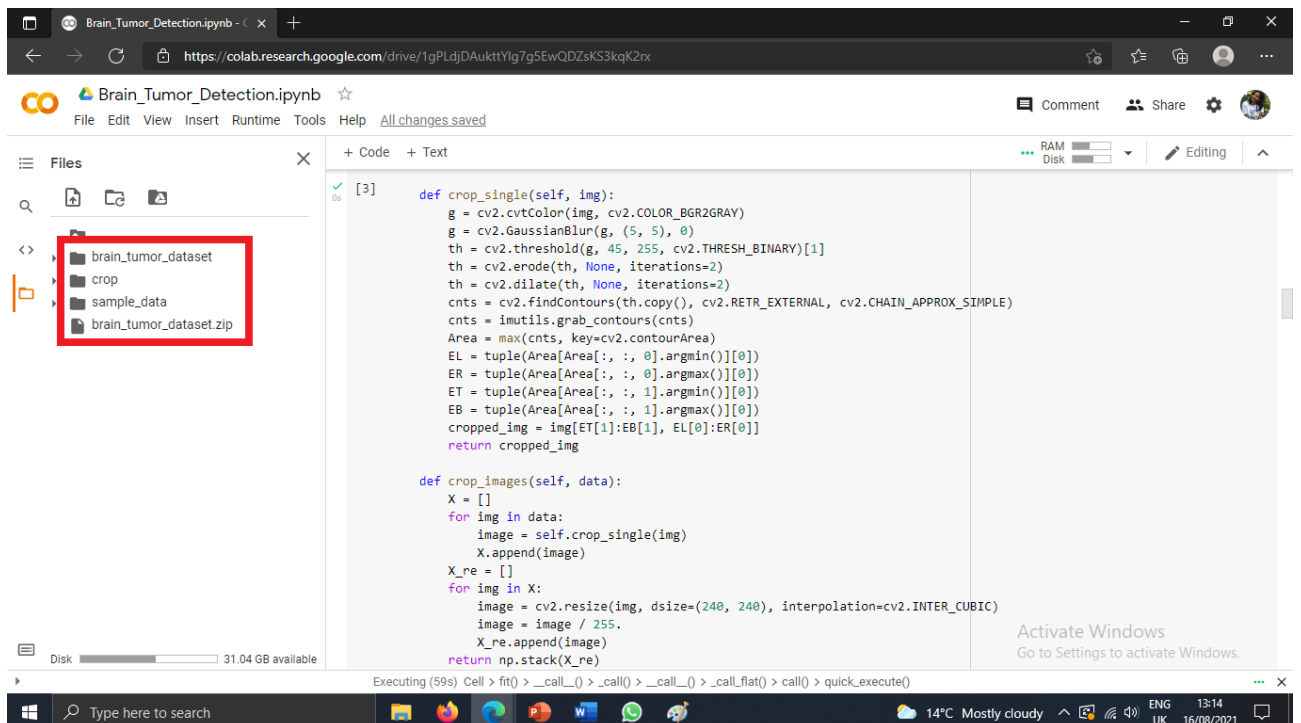
The screenshot shows a Google Colab notebook titled "Brain_Tumor_Detection.ipynb". The left sidebar displays the file explorer with a folder named "brain_tumor_dataset" containing subfolders "crop" and "sample_data", and a file "brain_tumor_dataset.zip". The main code area shows the execution of a cell with the following code:

```
inflatng: brain_tumor_dataset/yes/Y99.JPG

[2] import numpy as np
import pandas as pd
import seaborn as sn
import os
from os import listdir
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
import cv2
import matplotlib.pyplot as plt
%matplotlib inline
import imutils
import itertools
import warnings
from skimage import color
from skimage.feature import hog
from sklearn import svm
from sklearn.metrics import precision_score
warnings.filterwarnings('ignore')
from sklearn.metrics import accuracy_score, confusion_matrix
from tensorflow.keras.models import Model, load_model
from keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Input, ZeroPadding2D, BatchNormalization, Flatten, Activation, Dense, MaxPooling2D, Dropout
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
```

The bottom status bar indicates the execution time is 39 seconds and the cell is running the command: `> _next__() > next() > _get_batches_of_transformed_samples() > apply_transform() > apply_affine_transform() > <listcomp> > affine_transform()`.

Step 3: Data Augmentation



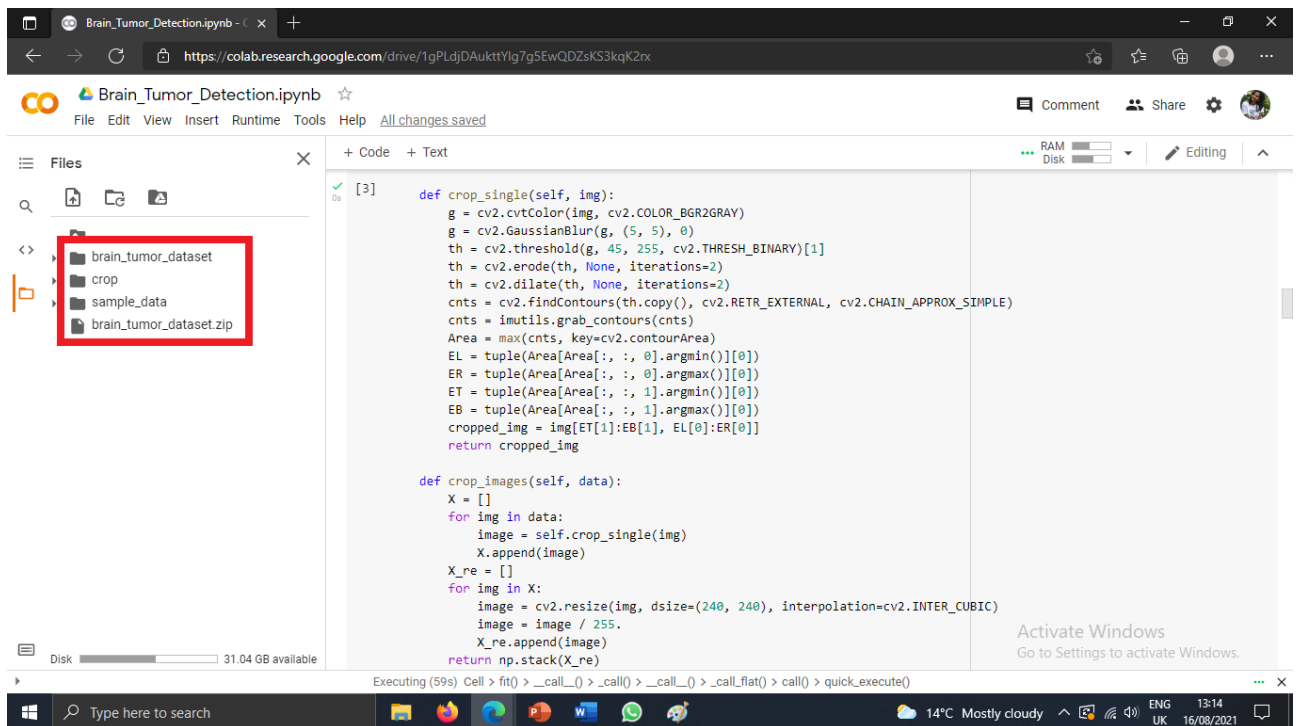
The screenshot shows the same Google Colab notebook. The left sidebar now highlights the "brain_tumor_dataset" folder with a red box. The main code area shows the execution of a cell with the following code:

```
[3] def crop_single(self, img):
    g = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    g = cv2.GaussianBlur(g, (5, 5), 0)
    th = cv2.threshold(g, 45, 255, cv2.THRESH_BINARY)[1]
    th = cv2.erode(th, None, iterations=2)
    th = cv2.dilate(th, None, iterations=2)
    cnts = cv2.findContours(th.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cnts = imutils.grab_contours(cnts)
    Area = max(cnts, key=cv2.contourArea)
    EL = tuple(Area[Area[:, :, 0].argmin()][0])
    ER = tuple(Area[Area[:, :, 0].argmax()][0])
    ET = tuple(Area[Area[:, :, 1].argmin()][0])
    EB = tuple(Area[Area[:, :, 1].argmax()][0])
    cropped_img = img[ET[1]:EB[1], EL[0]:ER[0]]
    return cropped_img

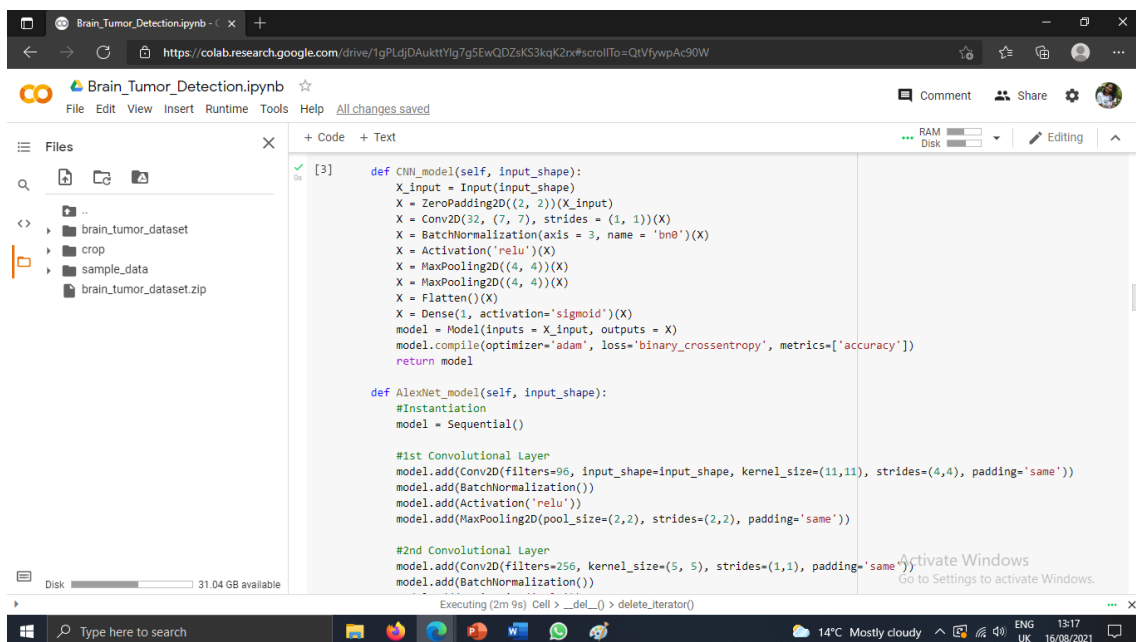
def crop_images(self, data):
    X = []
    for img in data:
        image = self.crop_single(img)
        X.append(image)
    X_re = []
    for img in X:
        image = cv2.resize(img, dsize=(240, 240), interpolation=cv2.INTER_CUBIC)
        image = image / 255.
        X_re.append(image)
    return np.stack(X_re)
```

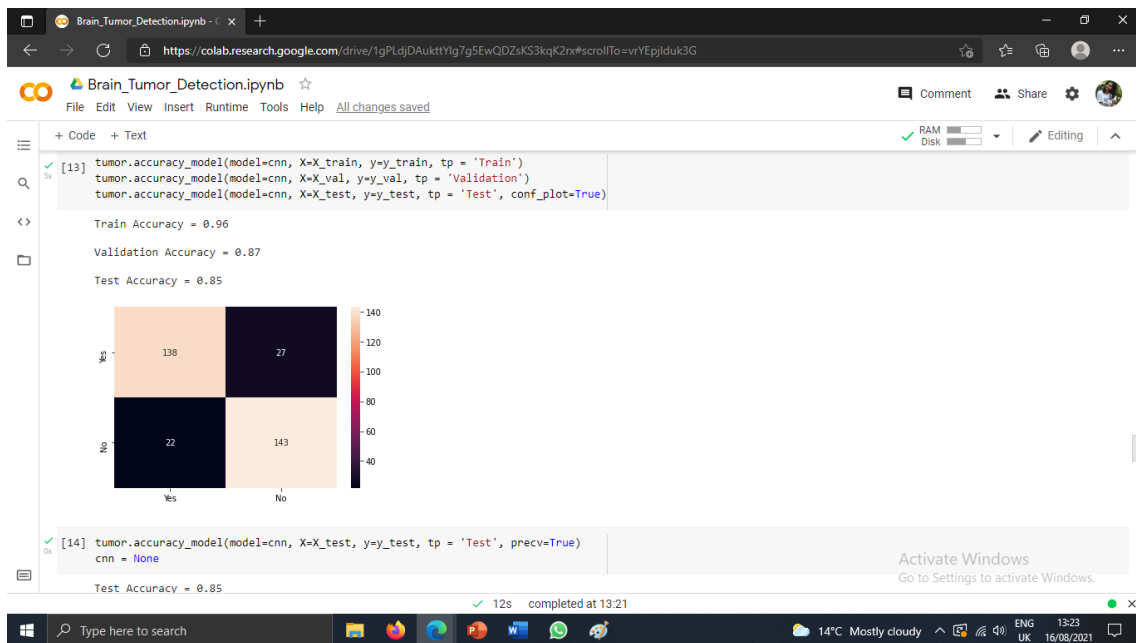
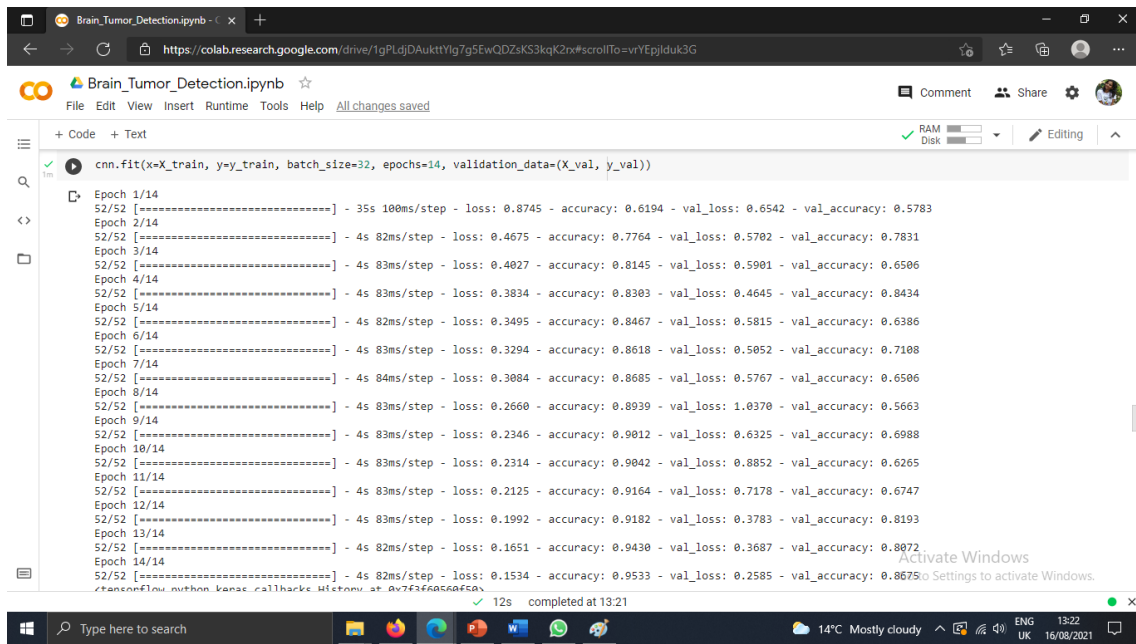
The bottom status bar indicates the execution time is 59 seconds and the cell is running the command: `> fit() > __call__() > __call__() > __call_flat() > call() > quick_execute()`.

Step 4: Create Crop Folder



Step 5: CNN Model





Step 5: AlexNet Model


```

def AlexNet_model(self, input_shape):
    #Instantiation
    model = Sequential()

    #1st Convolutional Layer
    model.add(Conv2D(filters=96, input_shape=input_shape, kernel_size=(11,11), strides=(4,4), padding='same'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'))

    #2nd Convolutional Layer
    model.add(Conv2D(filters=256, kernel_size=(5, 5), strides=(1,1), padding='same'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'))

    #3rd Convolutional Layer
    model.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), padding='same'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))

    #4th Convolutional Layer
    model.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), padding='same'))
    model.add(BatchNormalization())

```

Brain_Tumor_Detection.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```

def accuracy_model(self, model, X, y, tp, conf_plot = False, precv = False):
    predictions = model.predict(X)
    predictions = [1 if x>0.5 else 0 for x in predictions]
    accuracy = accuracy_score(y, predictions)
    print(tp+ ' Accuracy = %.2f\n' % accuracy)
    if precv:
        prec = precision_score(y, predictions, average=None)
        print('Precision Matrix')
        print(prec)
    if conf_plot:
        cm = confusion_matrix(y, predictions)
        df_cm = pd.DataFrame(cm, index = ["Yes", "No"], columns = ["Yes", "No"])
        sn.heatmap(df_cm, annot=True, fmt='g')

```

[4] image_dir = "brain_tumor_dataset/"
os.makedirs('crop', exist_ok = True)
os.makedirs('crop/yes', exist_ok = True)
os.makedirs('crop/no', exist_ok = True)
aug_dp = 'crop/'
aug_yes = aug_dp+'yes'
aug_no = aug_dp+'no'

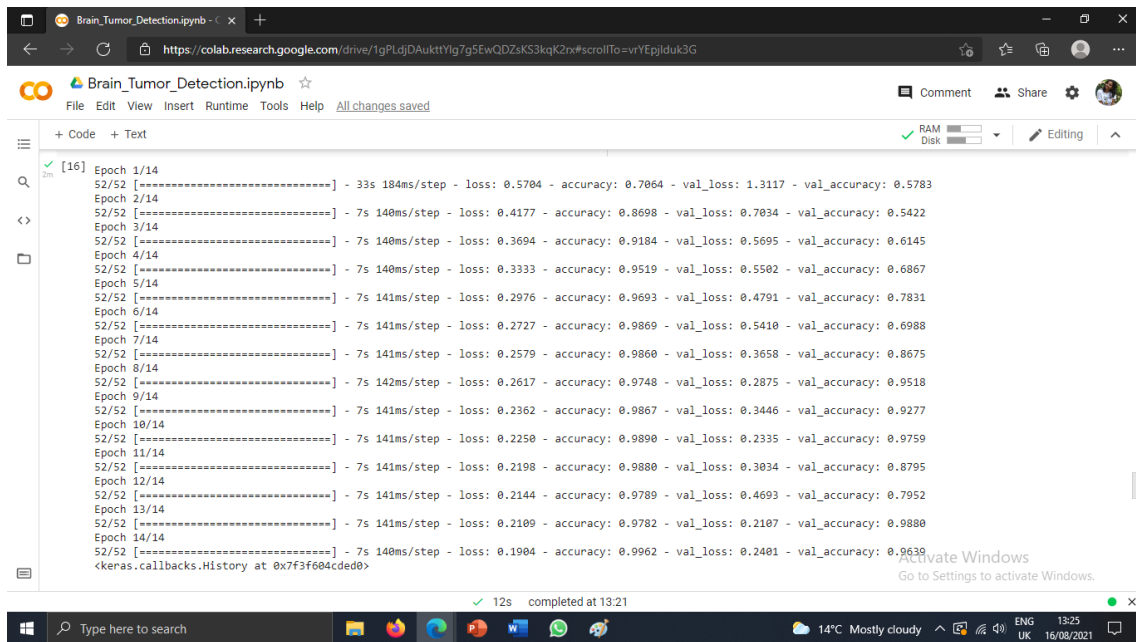
[5] tumor = Tumor()

[6] tumor.augment(inout_dir=image_dir+'yes', samples=6, output_dir=aug_yes)

12s completed at 13:21

Activate Windows
Go to Settings to activate Windows.

14°C Mostly cloudy 13:21 16/08/2021

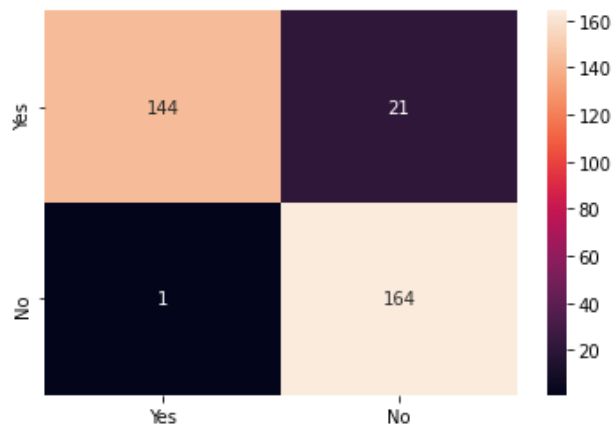


```
tumor.accuracy_model(model=alexnet, X=X_train, y=y_train, tp = 'Train')
tumor.accuracy_model(model=alexnet, X=X_val, y=y_val, tp = 'Validation')
tumor.accuracy_model(model=alexnet, X=X_test, y=y_test, tp = 'Test', conf_plot=True)
```

Train Accuracy = 0.97

Validation Accuracy = 0.96

Test Accuracy = 0.93



Step 6: Accuracy Model

```
def accuracy_model(self, model, X, y, tp, conf_plot = False, precv = False):
    predictions = model.predict(X)
    predictions = [1 if x>0.5 else 0 for x in predictions]
    accuracy = accuracy_score(y, predictions)
    print(tp+ ' Accuracy = %.2f\n' % accuracy)
    if precv:
        prec = precision_score(y, predictions, average=None)
        print('Precision Matrix')
        print(prec)
    if conf_plot:
        cm = confusion_matrix(y, predictions)
        df_cm = pd.DataFrame(cm, index = ["Yes", "No"], columns = ["Yes", "No"])
        sn.heatmap(df_cm, annot=True, fmt='g')
```

```
[4] image_dir = "brain_tumor_dataset/"
os.makedirs('crop', exist_ok = True)
os.makedirs('crop/yes', exist_ok = True)
os.makedirs('crop/no', exist_ok = True)
aug_dp = 'crop/'
aug_yes = aug_dp+'yes'
aug_no = aug_dp+'no'
```

```
[5] tumor = Tumor()
```

```
[6] tumor.augment(inout_dir=image_dir+'yes', samples=6, outout_dir=aug_yes)
```

12s completed at 13:21

Step 7: Histogram of Gradient and Support Vector Machine (HOG + SVM)

```
[19] X_gray = [color.rgb2gray(i) for i in X]
```

```
[20] ppc = 16
hog_images = []
hog_features = []

for image in X_gray:
    fd, hog_image = hog(image, orientations=8, pixels_per_cell=(ppc, ppc), cells_per_block=(4, 4), block_norm= 'L2', visualize=True)
    hog_images.append(hog_image)
    hog_features.append(fd)
```

```
[21] clf = svm.SVC()
hog_features = np.array(hog_features)
data_frame = np.hstack((hog_features, y))
np.random.seed(42)
np.random.shuffle(data_frame)
```

```
[22] #What percentage of data you want to keep for training
percentage = 80
partition = int(len(hog_features)*percentage/100)
x_train, x_test = data_frame[:partition, :-1], data_frame[partition: , :-1]
y_train, y_test = data_frame[:partition, -1:].ravel(), data_frame[partition: , -1:].ravel()
```

12s completed at 13:21