DETECTION OF BRAIN TUMORS IN CT-SCANS

A CAPSTONE PROJECT REPORT

Submitted in partial fulfillment of the requirement for the award of the Degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING WITH SPECIALIZATION IN NETWORKS AND SECURITY

by
Gandrapu Sri Sai Lahari (18BCN7120)

Under the guidance of
Dr. G.Muneeswari

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

VIT-AP UNIVERSITY

AMARAVATI- 522237

DECEMBER 2021

CERTIFICATE

This is to certify that the Capstone Project work titled **“DETECTION OF BRAIN TUMORS IN CT SCAN”** that is being submitted by **GANDRAPU SRI SAI LAHARI (18BCN7120)** is in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

[illegible]

(Dr. G,Muneeswari)

Guide

The thesis is satisfactory / ~~unsatisfactory~~

Approved by

[illegible]

PROGRAM CHAIR

B. Tech. CSE-NS

[illegible]

DEAN

School of Computer Science and Engineering

ACKNOWLEDGEMENT

I would like to express my gratitude to my guide Dr. G.Muneeswari, for providing me with her constant motivation, understanding, and encouragement in doing this project. I am truly thankful for her guidance throughout the project. This project would not have been completed without her continuous support.

I would also like to extend my gratitude to Dr. G. Viswanathan, Dr. S. V. Kota Reddy, Dr. CLV Sivakumar, Dr. Jagadish Mudiganti, SCOPE, for providing me an opportunity to carry out this project during the tenure of the course.

Place: Amaravathi
Date: 30-12-2021

Gandrapu Sri Sai Lahari

ABSTRACT

In this paper an efficient brain tumor detection system is designed, in which we can load the image file of the CT Scan and the system runs several image processing algorithms and gives us the result (i.e brain tumor detected or not detected). Brain Tumor is a serious illness and it needs prompt aid. This system can be installed in hospitals for immediate detection of the tumor. After the image dataset is loaded, colored images are converted into grayscale, then the noise is reduced in the images. I have developed this system using Gaussian Filter, adaptive Median Filter, and AMF-CNN-GBML models. These models are compared based on metrics like accuracy and error rate. I have used Tkinter for displaying the outputs in a batch file.

TABLE OF CONTENTS

S.NO	Chapter	Title	Page Number
1.		Acknowledgment	3
2.		Abstract	4
3.		List Of Figures and Table	6
4.	1	Introduction	7
	1.1	Objectives	8
	1.2	Background and Literature Survey	8-9
	1.3	Organization of Report	10
5.	2	DETECTION OF BRAIN TUMORS IN CT-SCANS	11-12
	2.1	Proposed System	12
	2.2	Dataset	13
	2.3	System Details	14
	2.3.1	Software Details	14-16
	2.3.2	Hardware Details	16
6.	3	Result and Discussion	17
7.	4	Conclusion and Future Works	26
8.	5	Appendix	27
9.	6	References	43

LIST OF TABLES

Table No.	**Title**	**Page No**
1.	Accuracy Comparison	**21**
2.	Error rate	**21**

LIST OF FIGURES

Figure No.	**Title**	**Page No.**
1.	Design of the system and Dataset	**12-13**
2.	Implementation	**17**
3.	Dataset loaded in the application	**18**
4.	AMF Output	**19**
5.	Accuracies of different models	**20**
6.	Accuracy graph	**22**
7.	Error rate graph	**23**
8.	Tumor detection image 1	**24**
9.	Tumor detection image 2	**25**

CHAPTER 1

INTRODUCTION

The growth of abnormal cells, cancerous or non-cancerous is a brain tumor. The tumor starts growing in the brain or anywhere else and can spread to the brain. It needs prompt aid otherwise it leads to many complexities. Image processing techniques come to our rescue here, we can use several image processing techniques and deep learning techniques to design a system that can detect the CT Scans. This makes the detection process a lot easier. The algorithms which give us high accuracy must be considered because there is a high chance of false positives and false negatives. Convolutional Neural Networks give high accuracy when it comes to the analysis of images, these are currently the best algorithms for automated image analysis and processing.

Convolutional Neural Networks are a part of deep learning. Neural Networks consist of nodes, input layer, hidden layers, and output layer. Convolutional Neural Networks uses principles from algebra, mainly matrix multiplication CNN, which has a convolutional layer and a pooling layer. Convolutional layer converts the inputs to generate a feature map. The pooling layer (eg. Max pooling, Min pooling, and average pooling) is used to reduce the number of features and to prevent overfitting. CNN is mostly used for image processing because it reduces the size of parameters. CNN is repeated for many epochs to get good accuracy.

1.1 Objectives

The following are the objectives of this project:

- To design an efficient system that can detect brain tumors in CT Scans.
- To find the image processing technique that gives the best accuracy for the CT Scan image dataset.
- Accuracy Comparison of different image processing techniques on the dataset.
- Error rate Comparison of different image processing techniques on the dataset.
- To check if the highly accurate algorithm gives the true positives and true negatives.

1.2 Background and Literature Survey

A similar project was carried out by A.Sivaramakrishnan and Dr.M.Karnan. They used a Fuzzy C-means clustering algorithm and effectively extracted the brain tumor region from the MRI images. They used the C-means clustering because of its simplicity and faster clustering. By using image segmentation they were able to locate the centroid point and it gave a high-resolution result.

Dalia Mahmoud carried out a similar project using Artificial Neural Networks. She used three techniques which were, Feed Forward Back Propagation Neural Network, Recurrent

Neural Network with 250 nodes, and Elman Network with 200 initial nodes. She observed that the Elman neural Network gave high accuracy among the three techniques. She used a dataset that contains all types of MR Scan models so she was able to achieve high performance, yield, and accuracy in detecting the tumors or abnormalities in the images.

Astina Minz and Chandrakant Mahobiya from the Department of CSE, MATS College of Engineering & Technology, Raipur, CG used the Adaboost technique in detecting the brain tumor. Their system consisted of three steps preprocessing, feature extraction, and classification. They preprocessed the dataset to remove any noise in the images. Threshold segmentation and median filtering were implemented on the pre-processed image. Then they used GLCM (Gray Level Co-occurrence Matrix) for the feature extraction process and for classification Adaboost was used.

T. Logeswari and M. Karnan used a clustering based approach which is SOM (Self-organizing map). The detection is done in two phases, firstly the noise in the images is removed and the second phase was to identify the principal tissue structures. They also presented a fuzzy C-means clustering for the process of segmentation. There are many segmentation techniques, among them, C-means clustering was used in their paper.

1.3 Organization of the report

The remaining chapters of the project are described as follows:

- Chapter 2 contains the proposed system, dataset, hardware, and software details.
- Chapter 3 contains the results and their discussion.
- Chapter 4 concludes the report.
- Chapter 5 consists of codes.
- Chapter 6 consists of references.

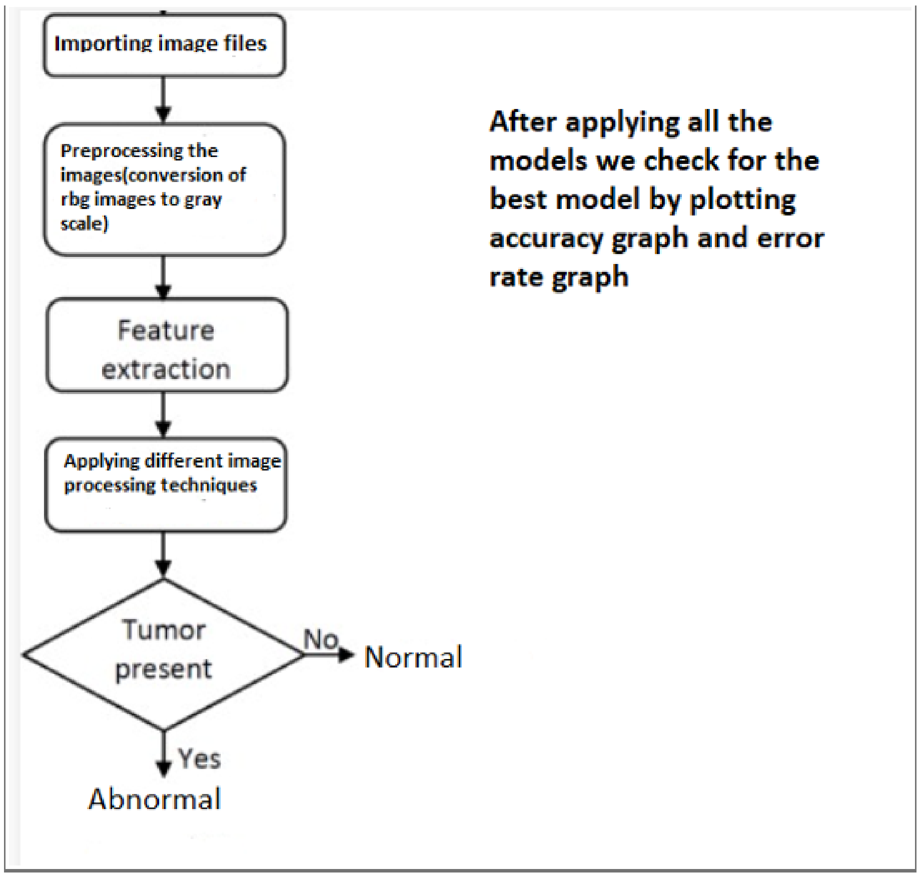
CHAPTER 2

This Chapter describes the proposed system, software, and hardware details of the project

2.1 Proposed System

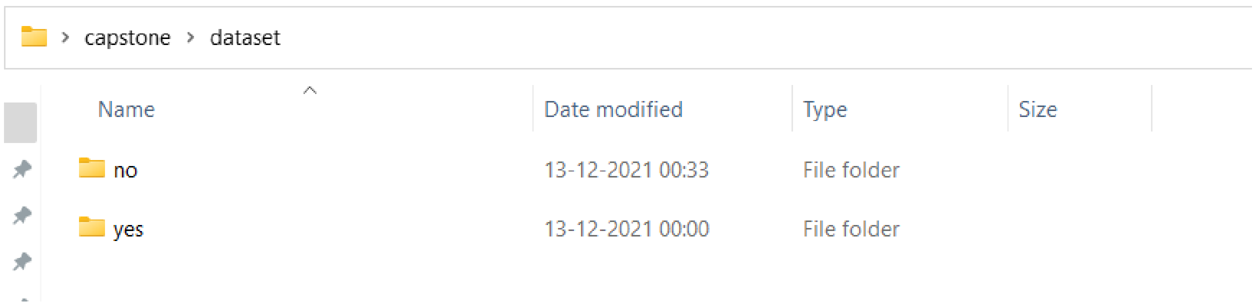
- I have trained the CT Scan images with Gaussian filter, Adaptive median filtering, and AMF gradient boosting algorithms, AMF GB got the highest accuracy among the three techniques.
- I have plotted the graphs for the accuracy and error rate of the three techniques for better understanding.
- I have used Tkinter to display an interface where we can upload the dataset, perform the algorithms on it and see the outputs.
- The final output we get would be if the CT Scan is normal or abnormal, where normal indicates the absence of tumor and abnormal indicates the presence of brain tumor.

The following flowchart (figure 1) shows the design of the project


Importing image files
Preprocessing the images(conversion of rbg images to gray scale)
Feature extraction
Applying different image processing techniques
Tumor present
No
Normal
Yes
Abnormal
After applying all the models we check for the best model by plotting accuracy graph and error rate graph

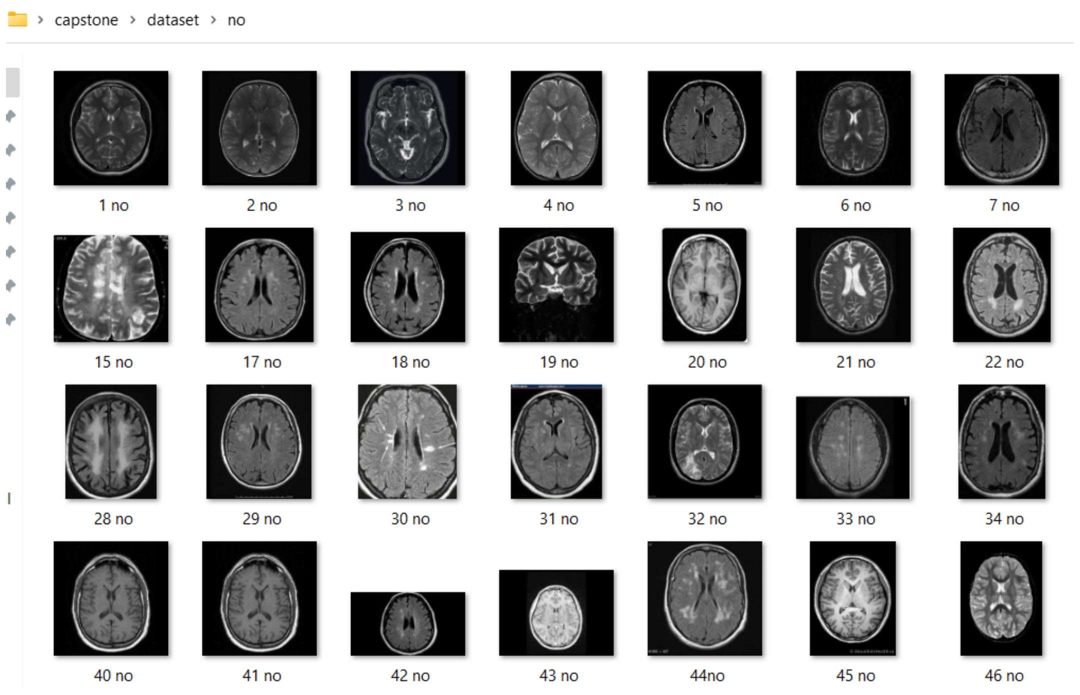
2.2 Dataset

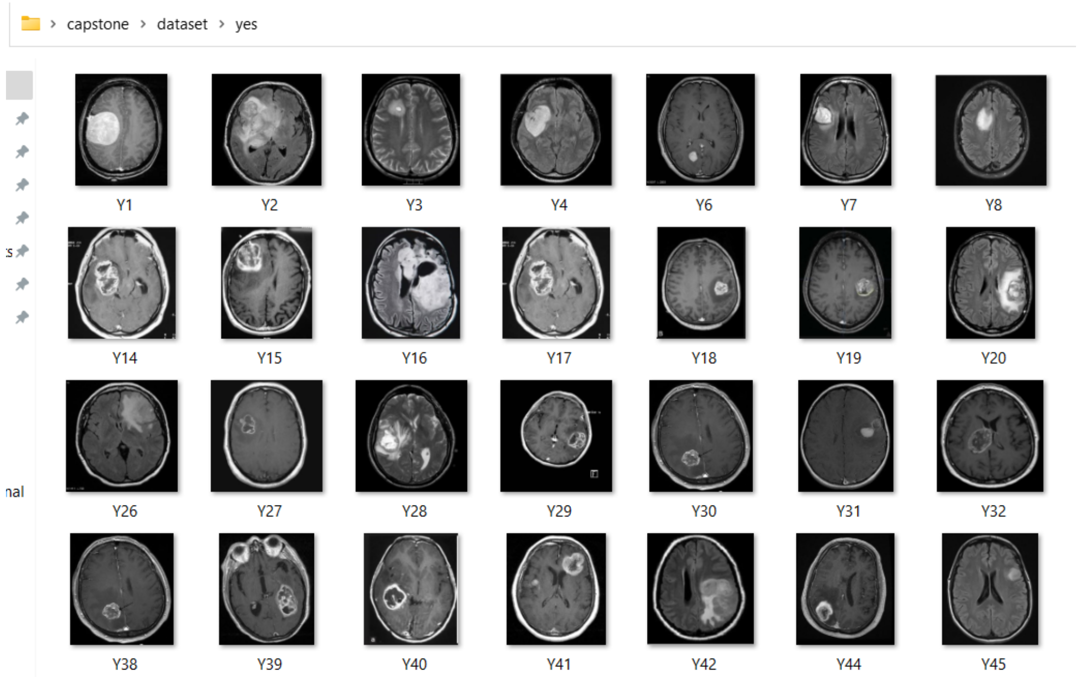
I found the dataset from kaggle and trained my model with both normal and abnormal CT Scan images.



› capstone › dataset

Name	Date modified	Type	Size
no	13-12-2021 00:33	File folder	
yes	13-12-2021 00:00	File folder	


capstone › dataset › no
1 no
2 no
3 no
4 no
5 no
6 no
7 no
15 no
17 no
18 no
19 no
20 no
21 no
22 no
28 no
29 no
30 no
31 no
32 no
33 no
34 no
40 no
41 no
42 no
43 no
44no
45 no
46 no


capstone › dataset › yes
Y1
Y2
Y3
Y4
Y6
Y7
Y8
Y14
Y15
Y16
Y17
Y18
Y19
Y20
Y26
Y27
Y28
Y29
Y30
Y31
Y32
Y38
Y39
Y40
Y41
Y42
Y44
Y45

2.3 System Details

This section contains the software and hardware details of the system:

2.3.1 Software Details

Python programming language, Tkinter, Adaptive Median Filter, Gaussian Filter, and AMF-CNN-GBML. Python Modules like NumPy, Pandas, Keras, TensorFlow, and OpenCV are required in the PC.

i) Python Programming language

I have used the Python 3.7.0 version in this project to write code for the algorithms. Python is a widely-used programming language for Machine Learning and Deep Learning projects because it is easily understandable and can also perform complex tasks quickly.

ii) Tkinter

Tkinter is a Graphical User Interface (GUI) module used in Python to create an interactive page. Using Tkinter we can create GUI applications with ease. To create a Tkinter app we need to import the module (Tkinter), then create the main window and add any widgets and event triggers on widgets. Tkinter consists of many modules like Frame, Listbox, Label, MenuButtons, Buttons, CheckButtons, and many more. We can also set the background color, size of the border, width, and height of the widget. We can also make a cursor appear when the mouse moves over the menubutton.

iii) Adaptive Median Filter

- Median Filter is an image processing technique. It is used to remove noise in the images.
- But the Median filter is only effective with impulse noise (salt and pepper).
- The median filter's output quality deteriorates when the noise level is more than 20 percent.
- So, an Adaptive Median Filter was developed to overcome the drawbacks of the Median Filter Technique.
- In the Adaptive median filter, the size of the kernel is variable so a better output is obtained.
- Adaptive Median Filter first checks the median value of the pixels, then it checks for corrupted or impulse noise image then replaces the impulse noise with the median value.

iv) Gaussian Filter

- Gaussian Filter is a Linear Filter, it is used to reduce noise and to blur images.
- Gaussian Filtering is more efficient in smoothening the images, researchers say that the neurons present in the human brain work the same as gaussian filters while processing the images.
- The most common first step while blurring the images is edge detection.
- The Gaussian smoothing operator is a two dimensional convolutional operator, it is used in the smoothing and blurring of images.

v) AMF-CNN-GBML

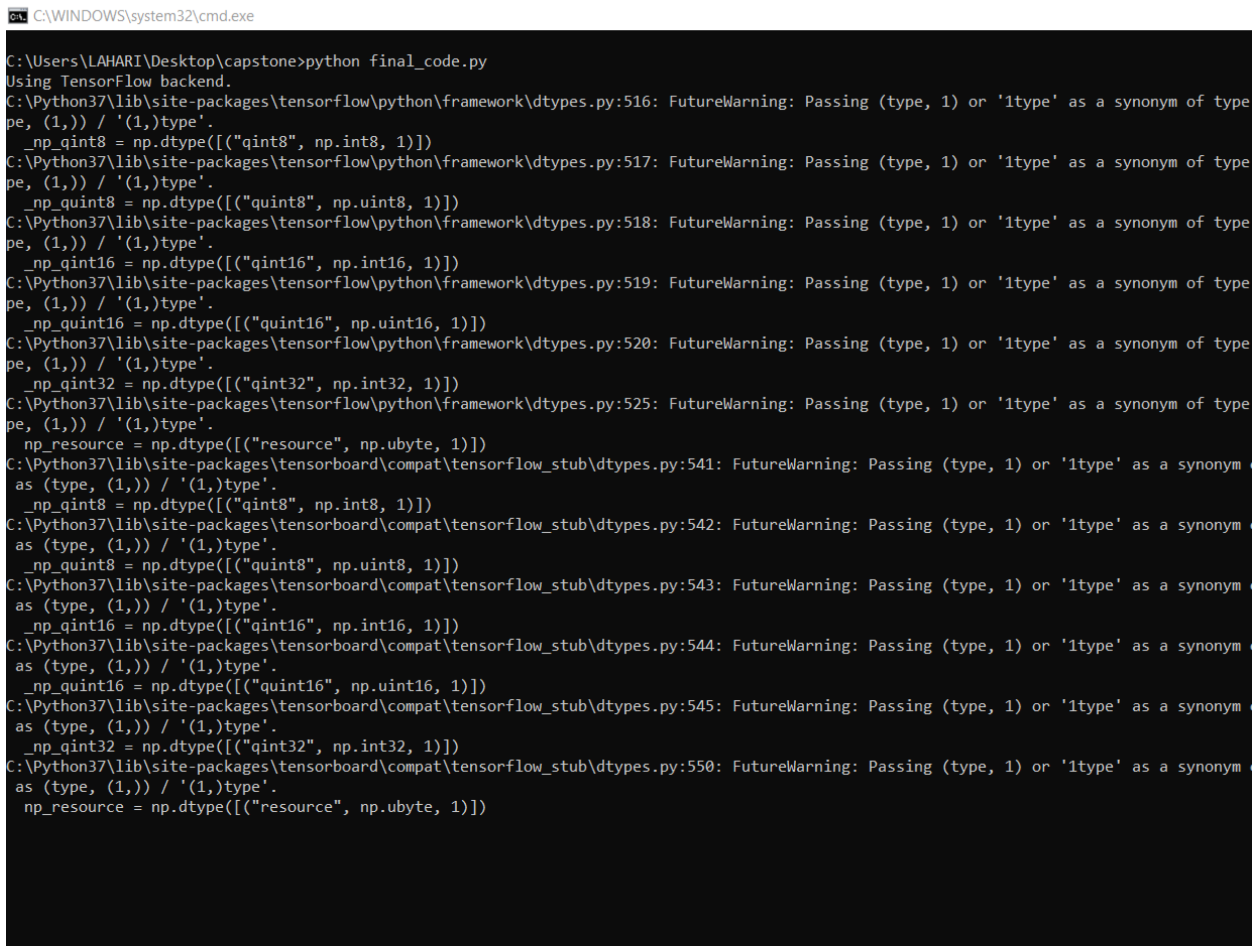
- This algorithm is a combination of Adaptive Median Filter with Convolutional Neural Networks and Gradient Boosting Machine Learning algorithms.
- The Adaptive Median filtering algorithm helps in the reduction of noise in the CT Scan images.
- RBG (colored) images are converted into Gray Scale images.
- Convolutional Neural Network does robust training on the dataset with a large number of epochs.
- In the ending gradient boosting algorithm is applied to the images, Gradient Boosting is an ensembling technique in Machine Learning.
- Gradient Boosting is used in reducing the bias errors in ML models.

2.3.2 Hardware Details

- Laptop Model: ASUS VivoBook S
- Processor: Intel Core i5
- CPU: 8th Gen
- Operating System: Windows 10
- RAM: 8 GB

CHAPTER 3

RESULTS AND DISCUSSION

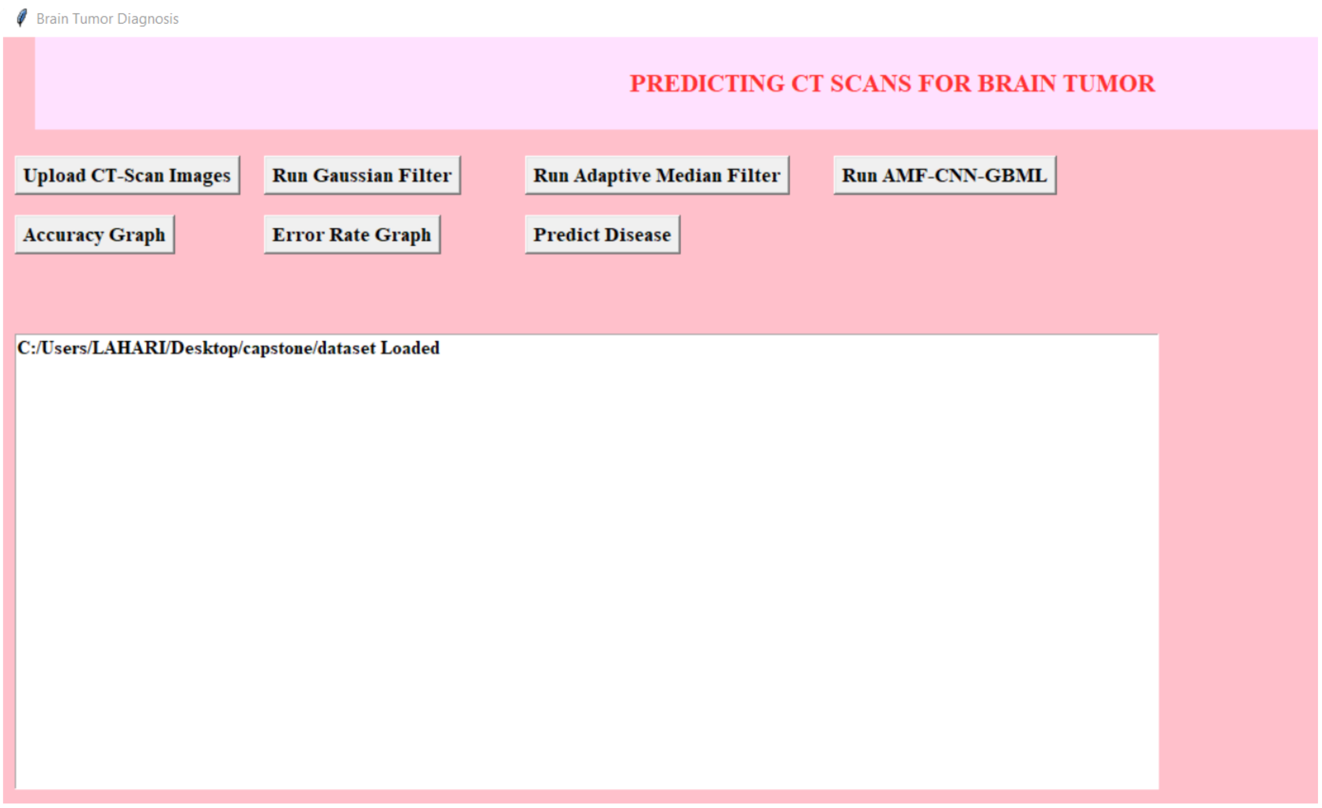


```
C:\WINDOWS\system32\cmd.exe

C:\Users\LAHARI\Desktop\capstone>python final_code.py
Using TensorFlow backend.
C:\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:516: FutureWarning: Passing (type, 1) or '1type' as a synonym of type
pe, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:517: FutureWarning: Passing (type, 1) or '1type' as a synonym of type
pe, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:518: FutureWarning: Passing (type, 1) or '1type' as a synonym of type
pe, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:519: FutureWarning: Passing (type, 1) or '1type' as a synonym of type
pe, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:520: FutureWarning: Passing (type, 1) or '1type' as a synonym of type
pe, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:525: FutureWarning: Passing (type, 1) or '1type' as a synonym of type
pe, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
C:\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:541: FutureWarning: Passing (type, 1) or '1type' as a synonym
 as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:542: FutureWarning: Passing (type, 1) or '1type' as a synonym
 as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:543: FutureWarning: Passing (type, 1) or '1type' as a synonym
 as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:544: FutureWarning: Passing (type, 1) or '1type' as a synonym
 as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:545: FutureWarning: Passing (type, 1) or '1type' as a synonym
 as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:550: FutureWarning: Passing (type, 1) or '1type' as a synonym
 as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

Figure2.

First step is to run the batch file, then the python file in which the algorithm code is present gets executed.

Figure 3.



```
C:/Users/LAHARI/Desktop/capstone/dataset Loaded
```

When the batch file gets executed, a GUI gets opened where we can upload the CT Scan Images.

After uploading the dataset, the above image is obtained.

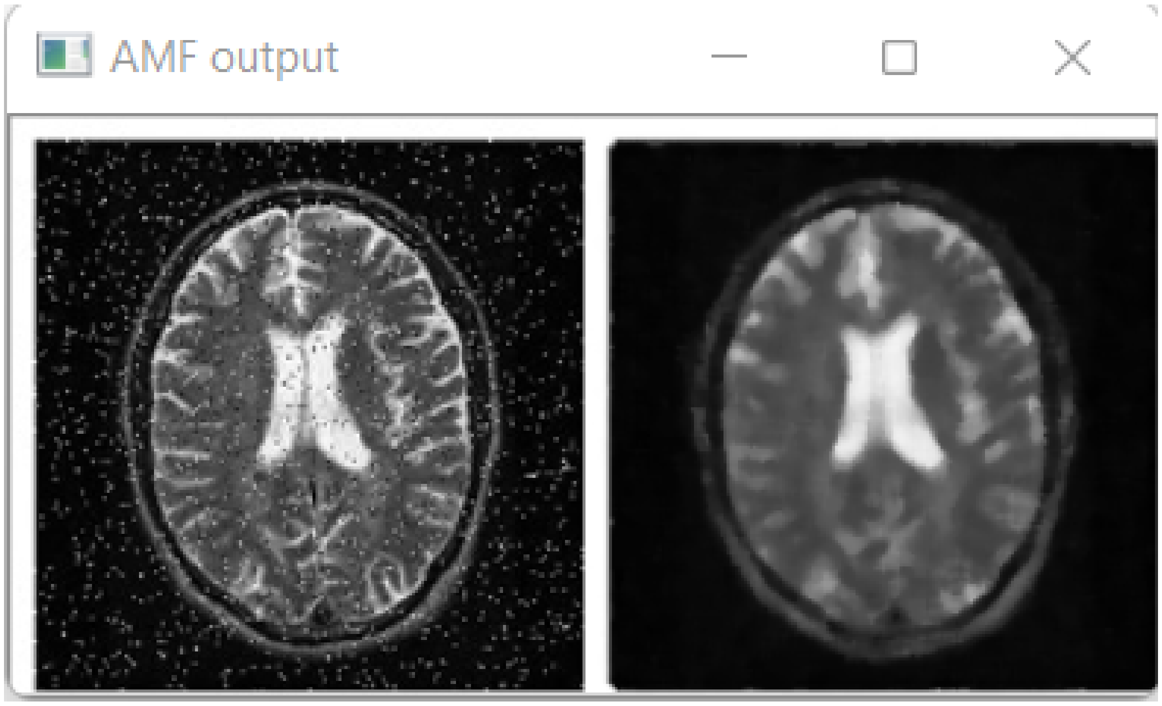
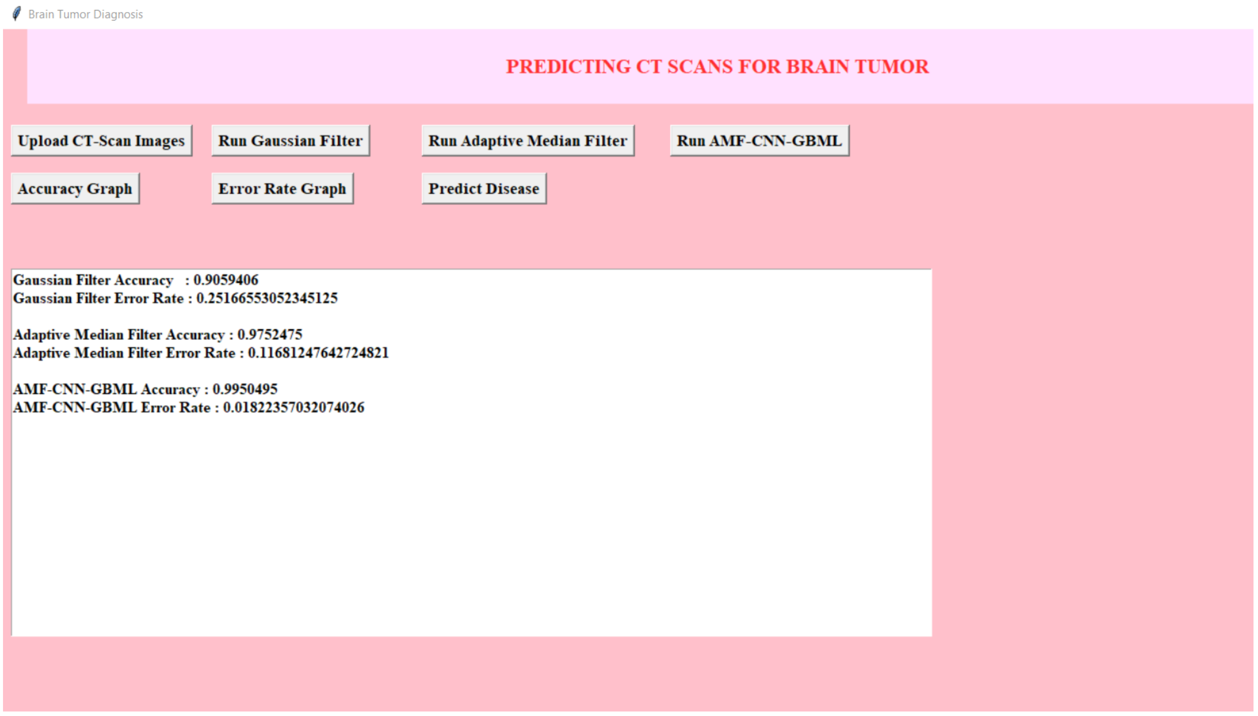


Figure 4.

- After the dataset is loaded with the Adaptive Median Filter algorithm, we make the images as grayscale and reduce the noise. The above image is the output.
- Left image contains noise (before AMF)
- Right image no noise (after AMF)



```
Gaussian Filter Accuracy   : 0.9059406
Gaussian Filter Error Rate : 0.25166553052345125

Adaptive Median Filter Accuracy : 0.9752475
Adaptive Median Filter Error Rate : 0.11681247642724821

AMF-CNN-GBML Accuracy : 0.9950495
AMF-CNN-GBML Error Rate : 0.01822357032074026
```

Figure 5.

- When we click the run Gaussian filter, a gaussian filter is applied on the dataset and its accuracy and error rate are obtained.
- When we run Adaptive Median Filter, Its accuracy and error rate are obtained.
- When we run AMF-CNN- GBML, its accuracy and error rate on the CT Scan dataset are displayed.
- The above images show all the algorithms accuracy and error rates.

Table 1. Accuracy Comparison Table

ALGORITHM	ACCURACY OBTAINED
Gaussian Filter	0.9059406
Adaptive Median Filter	0.9752475
AMF-CNN-GBML	0.9950495

Table 2. Error Rate Comparison Table

ALGORITHM	ERROR RATE
Gaussian Filter	0.25166
Adaptive Median Filter	0.11681
AMF-CNN-GBML	0.01822

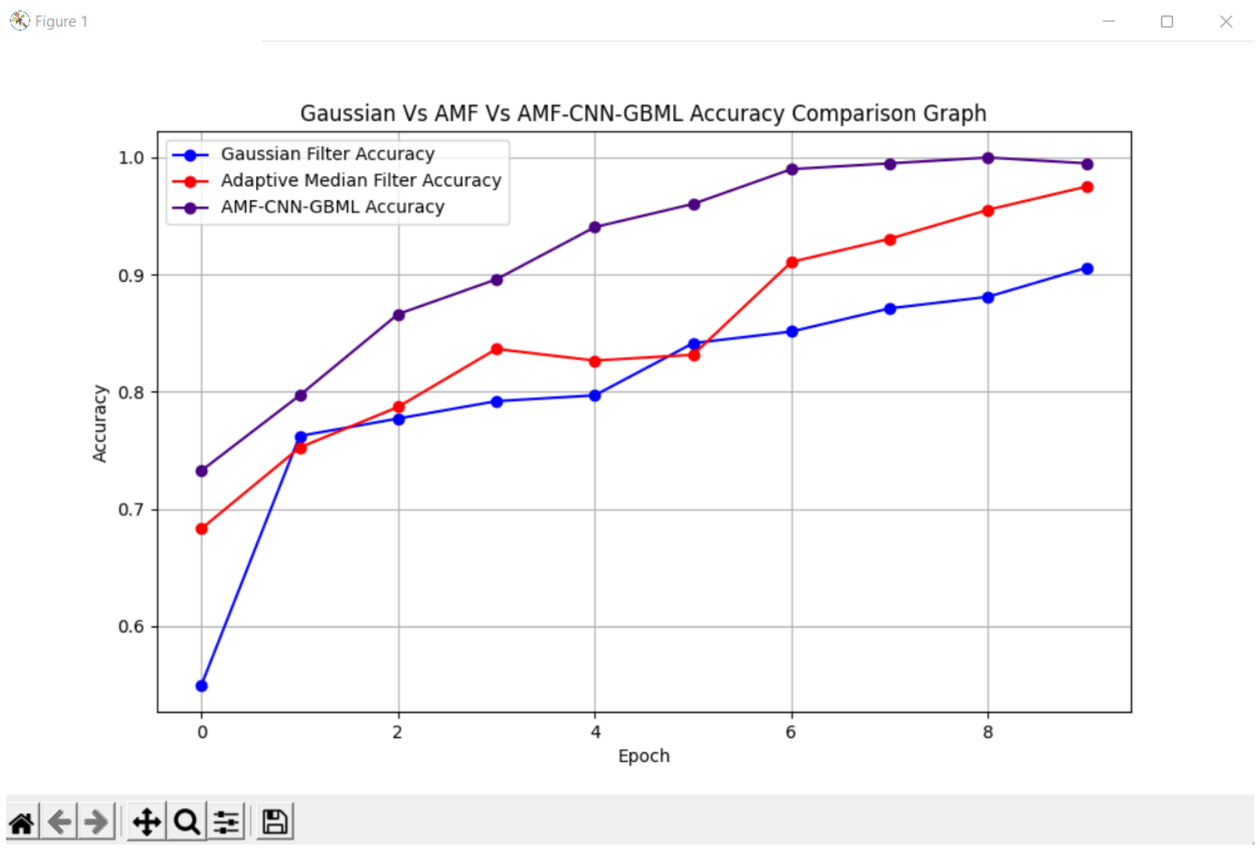


Figure 6.

- This figure shows us the accuracy comparison graph of the algorithms used in the project.
- As the number of epochs increases the accuracy of the model also increases.

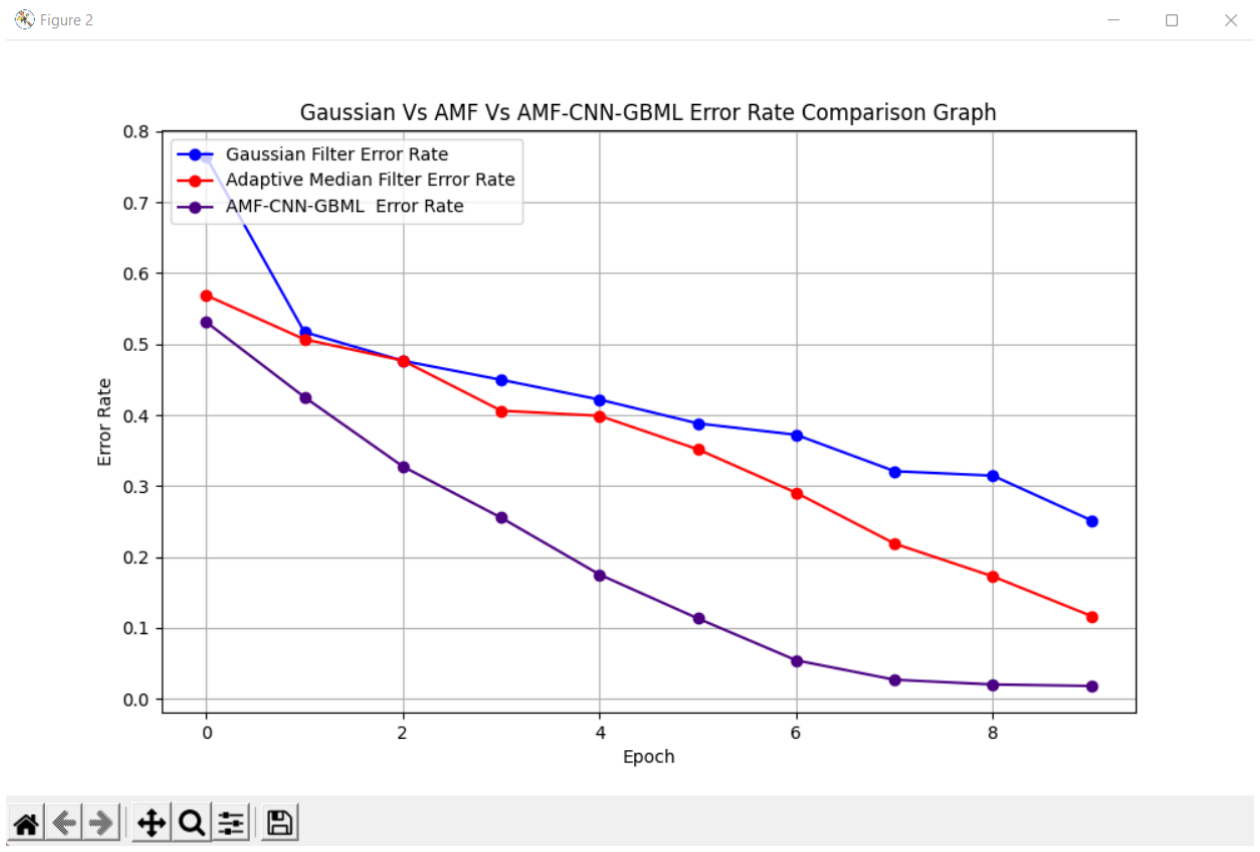


Figure 7.

- This figure shows us the error rate comparison of all three models I used in this project.
- As the number of epochs increases, the accuracy also increases so the rate is getting reduced.
- AMF-CNN-GBML has the highest accuracy so its error rate is less than other models.

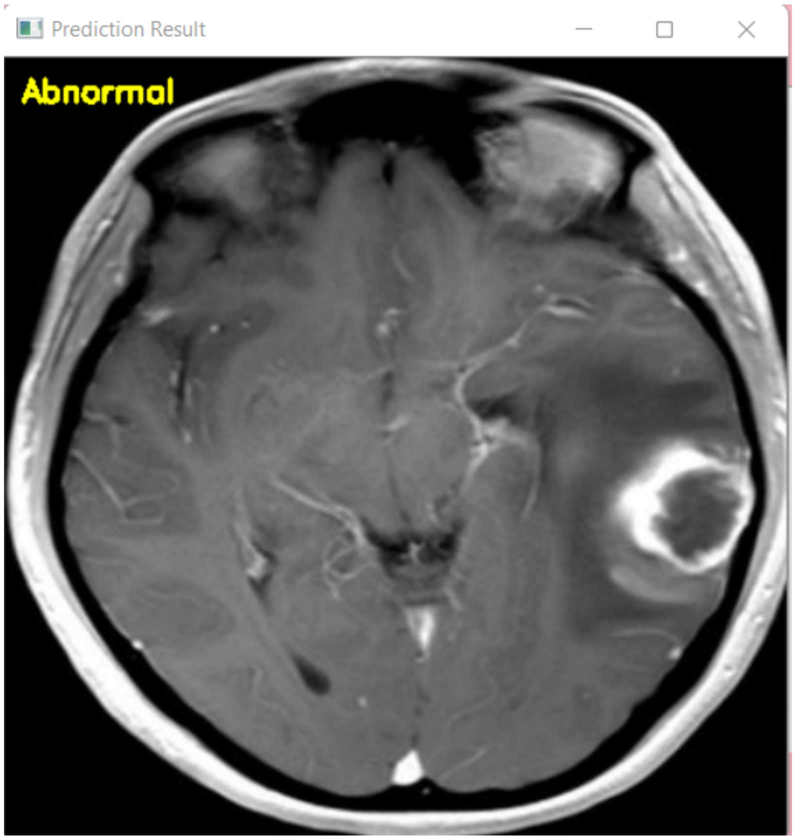


Figure 8.

- When we click on Predict disease we will be prompted to upload a CT Scan image.
- If the model detects the tumor in the brain, it gives the result as Abnormal.

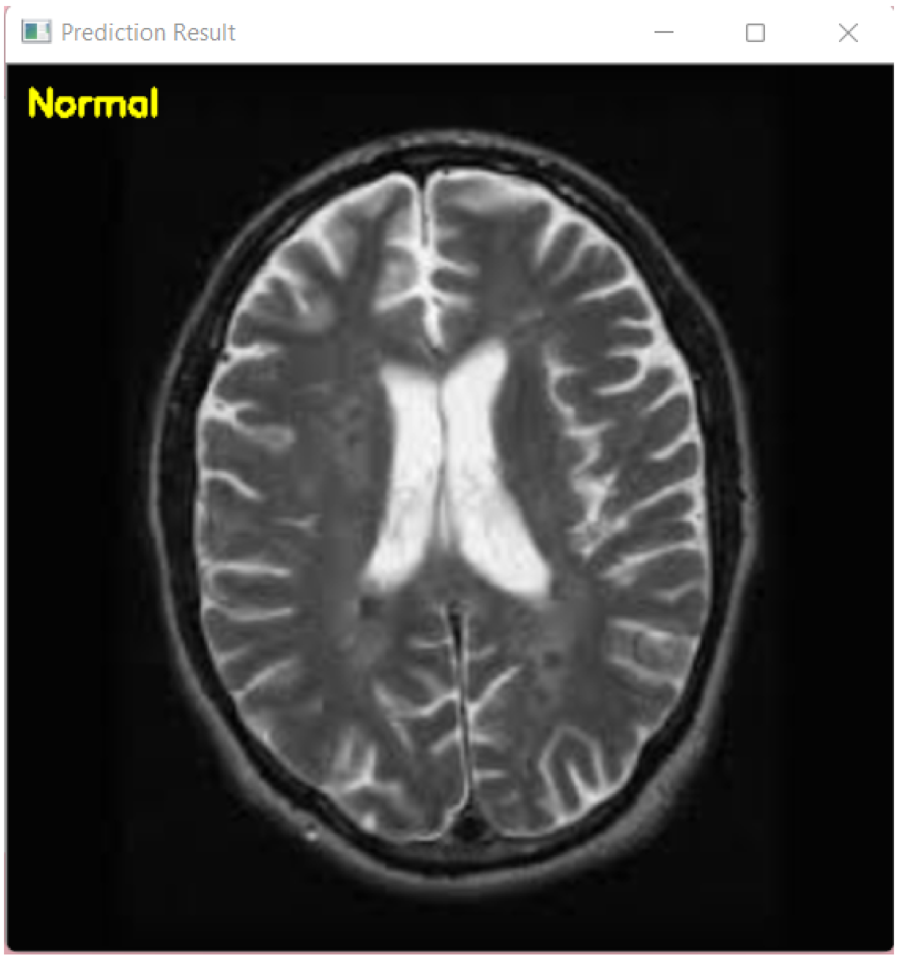


Figure 9.

- When we click on Predict disease we will be prompted to upload a CT Scan image.
- If the model detects the tumor in the brain, it gives the result as Abnormal, if it does not detect the tumor it gives *normal* as result.

CHAPTER 4

CONCLUSION AND FUTURE WORK

I proposed a system which detects brain tumors in CT Scans with Convolutional neural networks and Machine Learning Techniques. I have compared the accuracies and error rates of three algorithms namely Gaussian filter, Adaptive Median Filter, and AMF-CNN-GBML. Dataset images are converted into grayscale images and noise is removed from the images. I conclude that this system can efficiently detect brain tumors in CT Scans with an accuracy of 99% (obtained in AMF-CNN-GBML).

The trained model needs a robust and large dataset for proper accurate results. Medical data gathering is a hard task, so we need to make sure that the proposed algorithms will be robust enough to obtain accurate results in predicting brain tumors.

CHAPTER 5

APPENDIX

ALGORITHM CODES:

```
from tkinter import *
import tkinter
from tkinter import filedialog
import numpy as np
from tkinter.filedialog import askdirectory
from tkinter import simpledialog
import cv2
from keras.utils.np_utils import to_categorical
from keras.layers import Input
from keras.models import Model
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D
from keras.models import Sequential
import keras
import pickle
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from math import log10, sqrt
from PIL import Image, ImageFilter
import matplotlib.pyplot as plt
import webbrowser

main = tkinter.Tk()
```

```
main.title("Brain Tumor Diagnosis")
main.geometry("1500x1500")

global filename
global gaussian,amf,cnngb
global model
global Y,Y1
global X_gaussian, X_amf, X_cnngb

def rgb2gray(rgb):
    if(len(rgb.shape) == 3):
        return np.uint8(np.dot(rgb[...,:3], [0.2989, 0.5870, 0.1140]))
    else:#already a grayscale
        return rgb

def calculate_median(array):
    """Return the median of 1-d array"""
    sorted_array = np.sort(array) #timsort (O(nlogn))
    median = sorted_array[len(array)//2]
    return median

def level_A(z_min, z_med, z_max, z_xy, S_xy, S_max):
    if(z_min < z_med < z_max):
        return level_B(z_min, z_med, z_max, z_xy, S_xy, S_max)
    else:
        S_xy += 2 #increase the size of S_xy to the next odd value.
        if(S_xy <= S_max): #repeat process
            return level_A(z_min, z_med, z_max, z_xy, S_xy, S_max)
        else:
            return z_med
```

```
def level_B(z_min, z_med, z_max, z_xy, S_xy, S_max):
    if(z_min < z_xy < z_max):
        return z_xy
    else:
        return z_med

def adaptivemf(image, initial_window, max_window):
    """runs the Adaptive Median Filter proess on an image"""
    xlength, ylength = image.shape #get the shape of the image.

    z_min, z_med, z_max, z_xy = 0, 0, 0, 0
    S_max = max_window
    S_xy = initial_window #dynamically to grow

    output_image = image.copy()

    for row in range(S_xy, xlength-S_xy-1):
        for col in range(S_xy, ylength-S_xy-1):
            filter_window = image[row - S_xy : row + S_xy + 1, col - S_xy : col + S_xy + 1] #filter
window
            target = filter_window.reshape(-1) #make 1-dimensional
            z_min = np.min(target) #min of intensity values
            z_max = np.max(target) #max of intensity values
            z_med = calculate_median(target) #median of intensity values
            z_xy = image[row, col] #current intensity

            #Level A & B
            new_intensity = level_A(z_min, z_med, z_max, z_xy, S_xy, S_max)
            output_image[row, col] = new_intensity
    return output_image
```

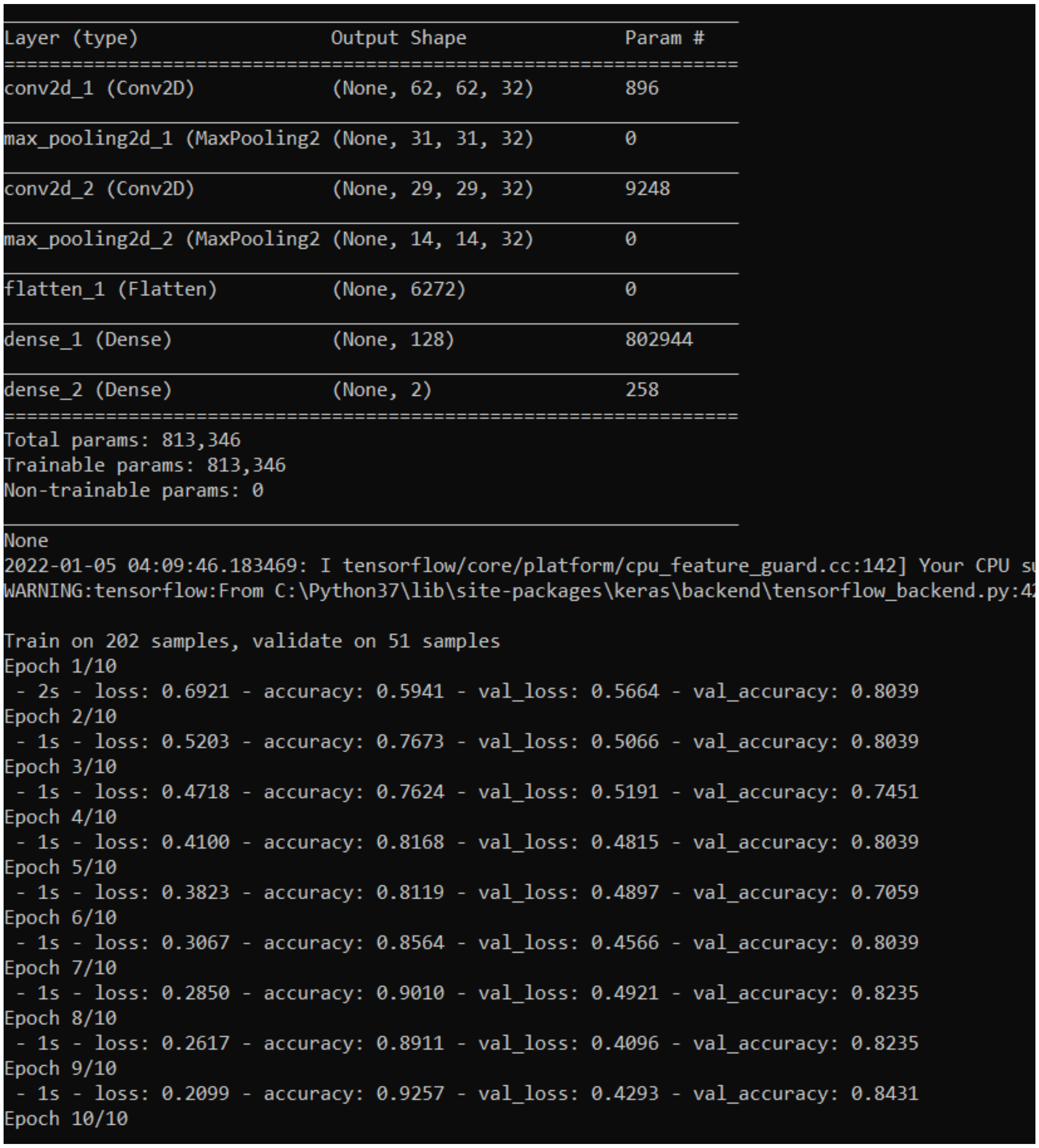
```
def get_feature_layer(model, data):
    total_layers = len(model.layers)
    fl_index = total_layers-1
    feature_layer_model =
Model(inputs=model.input,outputs=model.get_layer(index=fl_index).output)
    feature_layer_output = feature_layer_model.predict(data)
    return feature_layer_output

def upload():
  global filename
  global Y,Y1
  global X_gaussian, X_amf, X_cnngb
  filename = filedialog.askdirectory(initialdir = ".")

  X_gaussian = np.load("npy/gaussian.txt.npy")
  X_amf = np.load("npy/amf.txt.npy")
  X_cnngb = np.load("npy/amf.txt.npy")
  Y = np.load("npy/labels.txt.npy")
  Y1 = np.load("npy/labels.txt.npy")
  Y = to_categorical(Y)

  X_gaussian = X_gaussian.astype('float32')
  X_gaussian = X_gaussian/255
  X_amf = X_amf.astype('float32')
  X_amf = X_amf/255
  X_cnngb = X_cnngb.astype('float32')
  X_cnngb = X_cnngb/255

  indices = np.arange(X_gaussian.shape[0])
  np.random.shuffle(indices)
  X_gaussian = X_gaussian[indices]
  X_amf = X_amf[indices]
  X_cnngb = X_cnngb[indices]
  Y = Y[indices]
  Y1 = Y1[indices]

  text.delete('1.0', END)
  text.insert(END,filename+' Loaded')

  img = cv2.imread('test.jpeg')
  median = cv2.medianBlur(img, 5)
  compare = np.concatenate((img, median), axis=1) #side by side comparison
  cv2.imshow('AMF output', compare)
  cv2.waitKey(0)
  cv2.destroyAllWindows()

def runGaussian():
  global gaussian
  classifier = Sequential()
  classifier.add(Convolution2D(32, 3, 3, input_shape = (64, 64, 3), activation = 'relu'))
  classifier.add(MaxPooling2D(pool_size = (2, 2)))
  classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))
  classifier.add(MaxPooling2D(pool_size = (2, 2)))
  classifier.add(Flatten())
  classifier.add(Dense(output_dim = 128, activation = 'relu'))
  classifier.add(Dense(output_dim = 2, activation = 'softmax'))
  classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
  print(classifier.summary())
  gaussian = classifier.fit(X_gaussian, Y, batch_size=32, epochs=10, validation_split=0.2,
shuffle=True, verbose=2)
```

```
  gaussian = gaussian.history
  acc = gaussian['accuracy']
  loss = gaussian['loss']
  text.delete('1.0', END)
  text.insert(END,'Gaussian Filter Accuracy   : '+str(acc[9])+"\n")
  text.insert(END,'Gaussian Filter Error Rate : '+str(loss[9])+"\n")
  img = cv2.imread('test.jpeg')
  gau = cv2.GaussianBlur(img,(5,5),0)




def AMF():
  global amf
  image_org = Image.open("test.jpeg")
  image = np.array(image_org)
  grayscale_image = rgb2gray(image)
  output = adaptivemf(grayscale_image, 3, 11)
  output = cv2.medianBlur(output, 5)
  cv2.imwrite("clean.jpg",output)
  classifier = Sequential()
  classifier.add(Convolution2D(32, 3, 3, input_shape = (64, 64, 3), activation = 'relu'))
  classifier.add(MaxPooling2D(pool_size = (2, 2)))
  classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))
  classifier.add(MaxPooling2D(pool_size = (2, 2)))
  classifier.add(Flatten())
  classifier.add(Dense(output_dim = 128, activation = 'relu'))
  classifier.add(Dense(output_dim = 2, activation = 'softmax'))
  classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
  print(classifier.summary())
  amf = classifier.fit(X_amf, Y, batch_size=24, epochs=10, validation_split=0.2, shuffle=True, 
verbose=2)
  amf = amf.history
  acc = amf['accuracy']
  loss = amf['loss']
  text.insert(END,'\nAdaptive Median Filter Accuracy : '+str(acc[9])+"\n")
  text.insert(END,'Adaptive Median Filter Error Rate : '+str(loss[9])+"\n")
  first = cv2.imread("test.jpeg",0)
  second = cv2.imread("clean.jpg",0)



def AMFCNN():
  global cnngb
  global model
  image_org = Image.open("test.jpeg")
  image = np.array(image_org)
  grayscale_image = rgb2gray(image)
  output = adaptivemf(grayscale_image, 3, 11)
  cv2.imwrite("clean.jpg",output)
  classifier = Sequential()
  classifier.add(Convolution2D(32, 3, 3, input_shape = (64, 64, 3), activation = 'relu'))
  classifier.add(MaxPooling2D(pool_size = (2, 2)))
  classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))
  classifier.add(MaxPooling2D(pool_size = (2, 2)))
  classifier.add(Flatten())
  classifier.add(Dense(output_dim = 128, activation = 'relu'))
  classifier.add(Dense(output_dim = 2, activation = 'softmax'))
  classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
  print(classifier.summary())
  cnngb = classifier.fit(X_cnngb, Y, batch_size=8, epochs=10, validation_split=0.2, shuffle=True, 
verbose=2)
  cnngb = cnngb.history
  acc = cnngb['accuracy']
  loss = cnngb['loss']
  text.insert(END,'\nAMF-CNN-GBML Accuracy : '+str(acc[9])+"\n")
  text.insert(END,'AMF-CNN-GBML Error Rate : '+str(loss[9])+"\n")
  first = cv2.imread("test.jpeg",0)
  second = cv2.imread("clean.jpg",0)
  cnn_data = get_feature_layer(classifier,X_cnngb)#getting features from CNN
  gb = GradientBoostingClassifier()
  gb = gb.fit(cnn_data, Y1) #passing CNN deep features to gradient boosting algorithm for better 
prediction or classification
  prediction = gb.predict(cnn_data);
  cnn_gb_acc =  accuracy_score(prediction,Y1)
  model = classifier


def accuracyGraph():
  g_acc = gaussian['accuracy']
  a_acc = amf['accuracy']
  c_acc = cnngb['accuracy']

  plt.figure(figsize=(10,6))
  plt.grid(True)
  plt.xlabel('Epoch')
  plt.ylabel('Accuracy')
  plt.plot(g_acc, 'ro-', color = 'blue')
  plt.plot(a_acc, 'ro-', color = 'red')
  plt.plot(c_acc, 'ro-', color = 'indigo')
  plt.legend(['Gaussian Filter Accuracy', 'Adaptive Median Filter Accuracy','AMF-CNN-GBML
Accuracy'], loc='upper left')
  plt.title('Gaussian Vs AMF Vs AMF-CNN-GBML Accuracy Comparison Graph')
  plt.show()



def errorGraph():
  g_loss = gaussian['loss']
  a_loss = amf['loss']
  c_loss = cnngb['loss']
  plt.figure(figsize=(10,6))
  plt.grid(True)
  plt.xlabel('Epoch')
  plt.ylabel('Error Rate')
  plt.plot(g_loss, 'ro-', color = 'blue')
  plt.plot(a_loss, 'ro-', color = 'red')
  plt.plot(c_loss, 'ro-', color = 'indigo')
  plt.legend(['Gaussian Filter Error Rate', 'Adaptive Median Filter Error Rate','AMF-CNN-GBML
Error Rate'], loc='upper left')
  plt.title('Gaussian Vs AMF Vs AMF-CNN-GBML Error Rate Comparison Graph')
  plt.show()

def predictDisease():
  name = filedialog.askopenfilename(initialdir="testImages")
  img = cv2.imread(name)
  img = cv2.resize(img, (64,64))
  im2arr = np.array(img)
  im2arr = im2arr.reshape(1,64,64,3)
  XX = np.asarray(im2arr)
  XX = XX.astype('float32')
  XX = XX/255
  preds = model.predict(XX)
  print(str(preds)+" "+str(np.argmax(preds)))
  predict = np.argmax(preds)
  print(predict)
  img = cv2.imread(name)
  img = cv2.resize(img,(450,450))
  if predict == 0:
    cv2.putText(img, 'Normal', (10, 25),  cv2.FONT_HERSHEY_SIMPLEX,0.6, (0, 255, 255), 2)
  else:
    cv2.putText(img, 'Abnormal', (10, 25),  cv2.FONT_HERSHEY_SIMPLEX,0.6, (0, 255, 255),
2)
  cv2.imshow("Prediction Result",img)
  cv2.waitKey(0)


font = ('times', 16, 'bold')
title = Label(main, text='PREDICTING CT SCANS FOR BRAIN TUMOR', justify=LEFT)
title.config(bg='thistle1', fg='firebrick1')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=100,y=5)
title.pack()

font1 = ('times', 13, 'bold')
uploadButton = Button(main, text="Upload CT-Scan Images", command=upload)
uploadButton.place(x=10,y=100)
uploadButton.config(font=font1)

gaussianButton = Button(main, text="Run Gaussian Filter", command=runGaussian)
gaussianButton.place(x=220,y=100)
gaussianButton.config(font=font1)

amfButton = Button(main, text="Run Adaptive Median Filter", command=AMF)
amfButton.place(x=440,y=100)
amfButton.config(font=font1)

cnnButton = Button(main, text="Run AMF-CNN-GBML", command=AMFCNN)
cnnButton.place(x=700,y=100)
cnnButton.config(font=font1)

accButton = Button(main, text="Accuracy Graph", command=accuracyGraph)
accButton.place(x=10,y=150)
accButton.config(font=font1)

errorButton = Button(main, text="Error Rate Graph", command=errorGraph)
errorButton.place(x=220,y=150)
errorButton.config(font=font1)

predictButton = Button(main, text="Predict Disease", command=predictDisease)
predictButton.place(x=440,y=150)
predictButton.config(font=font1)



font1 = ('times', 12, 'bold')
text=Text(main,height=20,width=120)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=10,y=250)
text.config(font=font1)

main.config(bg='pink')
main.mainloop()
```

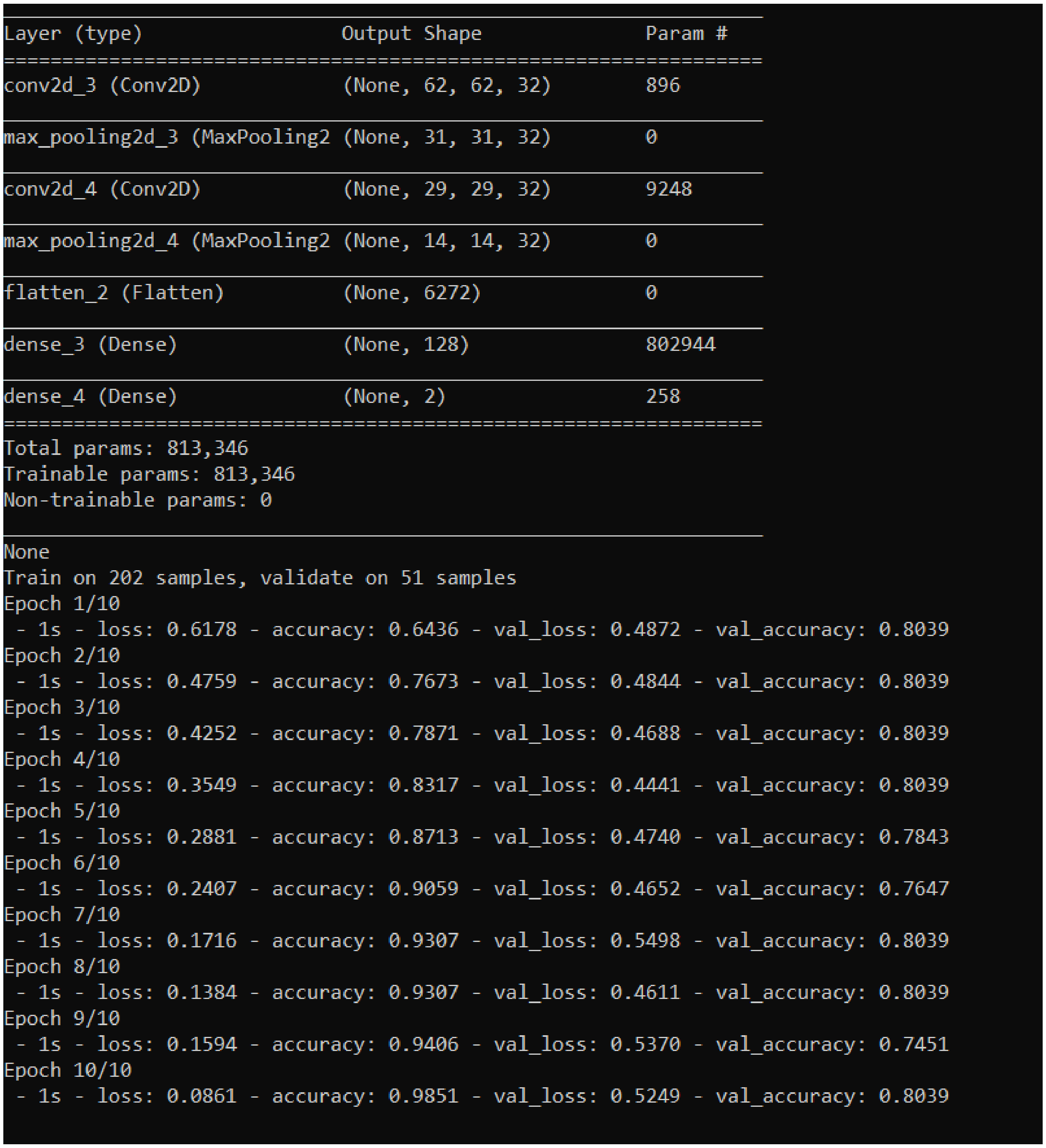
BATCH FILE CODE

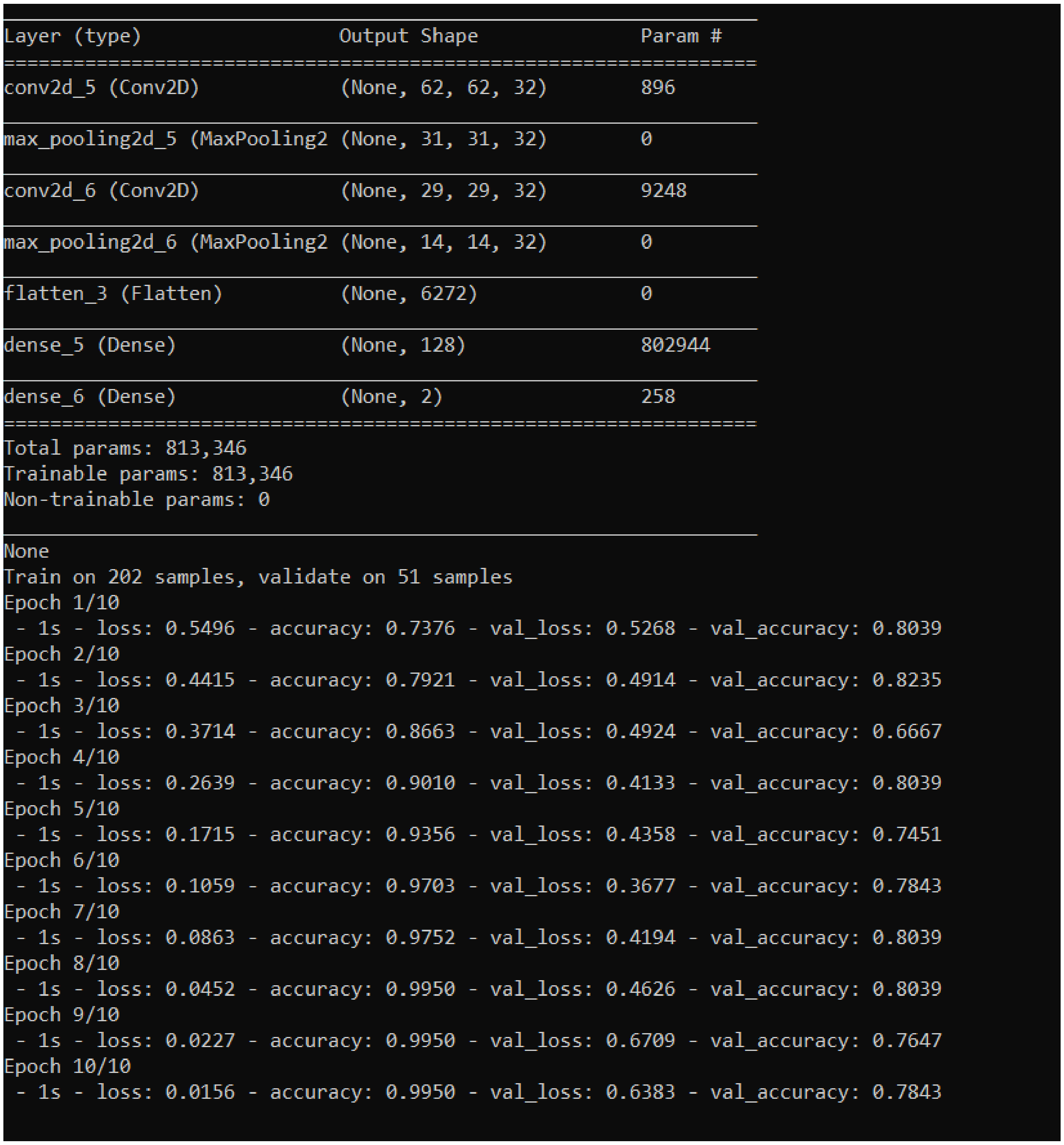
```
python final_code.py

pause
```

IMPLEMENTATION

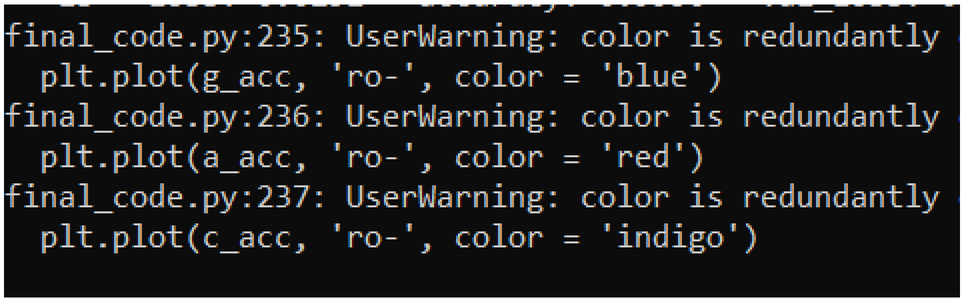


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 62, 62, 32)        896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 29, 29, 32)        9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               802944
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 258
=================================================================
Total params: 813,346
Trainable params: 813,346
Non-trainable params: 0
_________________________________________________________________
None
2022-01-05 04:09:46.183469: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU su
WARNING:tensorflow:From C:\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:42

Train on 202 samples, validate on 51 samples
Epoch 1/10
 - 2s - loss: 0.6921 - accuracy: 0.5941 - val_loss: 0.5664 - val_accuracy: 0.8039
Epoch 2/10
 - 1s - loss: 0.5203 - accuracy: 0.7673 - val_loss: 0.5066 - val_accuracy: 0.8039
Epoch 3/10
 - 1s - loss: 0.4718 - accuracy: 0.7624 - val_loss: 0.5191 - val_accuracy: 0.7451
Epoch 4/10
 - 1s - loss: 0.4100 - accuracy: 0.8168 - val_loss: 0.4815 - val_accuracy: 0.8039
Epoch 5/10
 - 1s - loss: 0.3823 - accuracy: 0.8119 - val_loss: 0.4897 - val_accuracy: 0.7059
Epoch 6/10
 - 1s - loss: 0.3067 - accuracy: 0.8564 - val_loss: 0.4566 - val_accuracy: 0.8039
Epoch 7/10
 - 1s - loss: 0.2850 - accuracy: 0.9010 - val_loss: 0.4921 - val_accuracy: 0.8235
Epoch 8/10
 - 1s - loss: 0.2617 - accuracy: 0.8911 - val_loss: 0.4096 - val_accuracy: 0.8235
Epoch 9/10
 - 1s - loss: 0.2099 - accuracy: 0.9257 - val_loss: 0.4293 - val_accuracy: 0.8431
Epoch 10/10
```

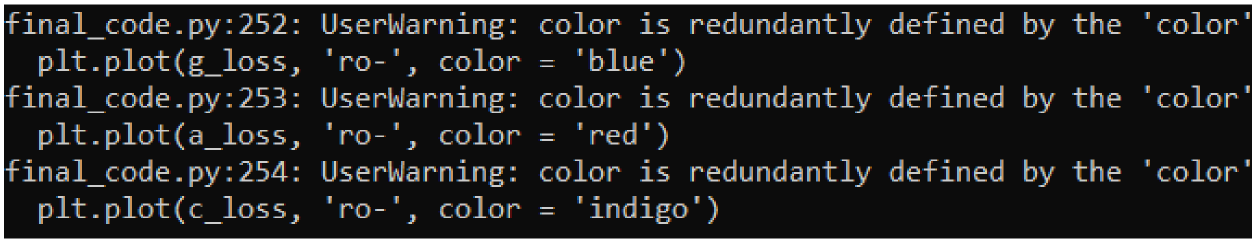

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 62, 62, 32)        896
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 29, 29, 32)        9248
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               802944
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 258
=================================================================
Total params: 813,346
Trainable params: 813,346
Non-trainable params: 0
_________________________________________________________________
None
Train on 202 samples, validate on 51 samples
Epoch 1/10
 - 1s - loss: 0.6178 - accuracy: 0.6436 - val_loss: 0.4872 - val_accuracy: 0.8039
Epoch 2/10
 - 1s - loss: 0.4759 - accuracy: 0.7673 - val_loss: 0.4844 - val_accuracy: 0.8039
Epoch 3/10
 - 1s - loss: 0.4252 - accuracy: 0.7871 - val_loss: 0.4688 - val_accuracy: 0.8039
Epoch 4/10
 - 1s - loss: 0.3549 - accuracy: 0.8317 - val_loss: 0.4441 - val_accuracy: 0.8039
Epoch 5/10
 - 1s - loss: 0.2881 - accuracy: 0.8713 - val_loss: 0.4740 - val_accuracy: 0.7843
Epoch 6/10
 - 1s - loss: 0.2407 - accuracy: 0.9059 - val_loss: 0.4652 - val_accuracy: 0.7647
Epoch 7/10
 - 1s - loss: 0.1716 - accuracy: 0.9307 - val_loss: 0.5498 - val_accuracy: 0.8039
Epoch 8/10
 - 1s - loss: 0.1384 - accuracy: 0.9307 - val_loss: 0.4611 - val_accuracy: 0.8039
Epoch 9/10
 - 1s - loss: 0.1594 - accuracy: 0.9406 - val_loss: 0.5370 - val_accuracy: 0.7451
Epoch 10/10
 - 1s - loss: 0.0861 - accuracy: 0.9851 - val_loss: 0.5249 - val_accuracy: 0.8039
```



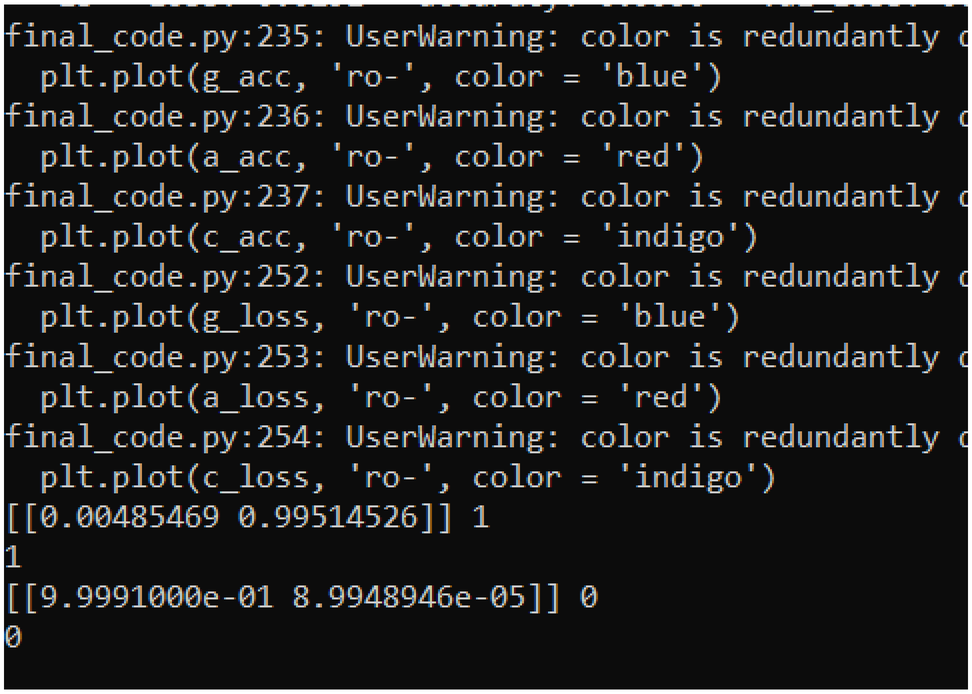
```
________________________________________________________________
Layer (type)                 Output Shape              Param #
================================================================
conv2d_5 (Conv2D)            (None, 62, 62, 32)        896
________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 31, 31, 32)        0
________________________________________________________________
conv2d_6 (Conv2D)            (None, 29, 29, 32)        9248
________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 14, 14, 32)        0
________________________________________________________________
flatten_3 (Flatten)          (None, 6272)              0
________________________________________________________________
dense_5 (Dense)              (None, 128)               802944
________________________________________________________________
dense_6 (Dense)              (None, 2)                 258
================================================================
Total params: 813,346
Trainable params: 813,346
Non-trainable params: 0
________________________________________________________________
None
Train on 202 samples, validate on 51 samples
Epoch 1/10
 - 1s - loss: 0.5496 - accuracy: 0.7376 - val_loss: 0.5268 - val_accuracy: 0.8039
Epoch 2/10
 - 1s - loss: 0.4415 - accuracy: 0.7921 - val_loss: 0.4914 - val_accuracy: 0.8235
Epoch 3/10
 - 1s - loss: 0.3714 - accuracy: 0.8663 - val_loss: 0.4924 - val_accuracy: 0.6667
Epoch 4/10
 - 1s - loss: 0.2639 - accuracy: 0.9010 - val_loss: 0.4133 - val_accuracy: 0.8039
Epoch 5/10
 - 1s - loss: 0.1715 - accuracy: 0.9356 - val_loss: 0.4358 - val_accuracy: 0.7451
Epoch 6/10
 - 1s - loss: 0.1059 - accuracy: 0.9703 - val_loss: 0.3677 - val_accuracy: 0.7843
Epoch 7/10
 - 1s - loss: 0.0863 - accuracy: 0.9752 - val_loss: 0.4194 - val_accuracy: 0.8039
Epoch 8/10
 - 1s - loss: 0.0452 - accuracy: 0.9950 - val_loss: 0.4626 - val_accuracy: 0.8039
Epoch 9/10
 - 1s - loss: 0.0227 - accuracy: 0.9950 - val_loss: 0.6709 - val_accuracy: 0.7647
Epoch 10/10
 - 1s - loss: 0.0156 - accuracy: 0.9950 - val_loss: 0.6383 - val_accuracy: 0.7843
```



```
final_code.py:235: UserWarning: color is redundantly
  plt.plot(g_acc, 'ro-', color = 'blue')
final_code.py:236: UserWarning: color is redundantly
  plt.plot(a_acc, 'ro-', color = 'red')
final_code.py:237: UserWarning: color is redundantly
  plt.plot(c_acc, 'ro-', color = 'indigo')
```



```
final_code.py:252: UserWarning: color is redundantly defined by the 'color'
  plt.plot(g_loss, 'ro-', color = 'blue')
final_code.py:253: UserWarning: color is redundantly defined by the 'color'
  plt.plot(a_loss, 'ro-', color = 'red')
final_code.py:254: UserWarning: color is redundantly defined by the 'color'
  plt.plot(c_loss, 'ro-', color = 'indigo')
```



```
final_code.py:235: UserWarning: color is redundantly
  plt.plot(g_acc, 'ro-', color = 'blue')
final_code.py:236: UserWarning: color is redundantly
  plt.plot(a_acc, 'ro-', color = 'red')
final_code.py:237: UserWarning: color is redundantly
  plt.plot(c_acc, 'ro-', color = 'indigo')
final_code.py:252: UserWarning: color is redundantly
  plt.plot(g_loss, 'ro-', color = 'blue')
final_code.py:253: UserWarning: color is redundantly
  plt.plot(a_loss, 'ro-', color = 'red')
final_code.py:254: UserWarning: color is redundantly
  plt.plot(c_loss, 'ro-', color = 'indigo')
[[0.00485469 0.99514526]] 1
1
[[9.9991000e-01 8.9948946e-05]] 0
0
```

CHAPTER 6

REFERENCES

[1] Mahmoud, Dalia and Mohamed, Eltaher 2012/12/01"Brain Tumor Detection Using Artificial Neural Networks"Journal of Science and Technology- Vol. 13, No. 2 ISSN 1605 – 427X Engineering and Computer Sciences (ECS)

[2] A.Sivaramakrishnan And Dr.M.Karnan "A Novel Based Approach For Extraction Of Brain Tumor In Mri Images Using Soft Computing Techniques," International Journal Of Advanced Research In Computer And Communication Engineering, Vol. 2, Issue 4, April 2013.

[3] Astina Minz and Chandrakant Mahobiya "MR Image Classification Using Adaboost for Brain Tumor Type." 2017 IEEE 7th International Advance Computing Conference (IACC) (2017)

[4] T.Logeswari and M.Karnan, "An Improved Implementation of Brain Tumor Detection Using Segmentation Based on Hierarchical Self Organizing Map", International Journal of Computer Theory and
Engineering, Vol. 2, No. 4, August, 2010

[5] S. Roy And S. K. Bandyopadhyay, "Detection and Qualification Of Brain Tumor From MRI Of Brain And Symmetric Analysis," International
Journal Of Information And Communication Technology Research,
Volume 2 No.6, June 2012

[6] Dou, W., Ruan, S., Chen, Y., Bloyet, D., and Constans, J. M. (2007), "A framework of fuzzy information fusion for segmentation of brain tumor tissues on MR images",
Image and Vision Computing

[7] Dipali M. Joshi, Rana. N. K, Misra. V. M, 2010. "Classification of Brain Cancer Using Artificial Neural Network", IEEE, 112-116.

[8] Letteboer, M. M. J, Olsen, O. F and Dam, E. B. 2004. Segmentation of tumors in magnetic resonance brain images using an interactive multiscale watershed algorithm. Acad Radiol

[9] Jayaram Udupa,Punam Saha,"Fuzzy Connectedness and Image Segmentation" , Proceedings of the IEEE,vol.91

[10] Samir Kumar Bandyopadhyay, Tuhin Utsab Paul "Segmentation of Brain Tumor from MRI image – Analysis of K Means and DBSCAN Clustering" International Journal of Research in

Engineering and Science (IJRES) ISSN (Online): 2320-9364, ISSN (Print): 2320-9356 www.ijres.org Volume 1 Issue 1 May. 2013 PP.48-57

[11] Han X, Fischl B "Atlas renormalization for improved brain MR image segmentation across scanner platforms" IEEE Trans Med Imaging 26(4)

[12] Monica Subashini.M, Sarat Kumar Sahoo, "Brain MR Image Segmentation for TumorDetection using Artificial Neural Networks" International Journal of Engineering and Technology (IJET), Vol.5

[13] Evangelia I. Zacharaki,Sumei Wang,Sanjeev Chawla,Dong Soo Yoo,Ronald Wolf,Elias R. Melhem,Christos Davatzikos "Classification of brain tumor type and grade using MRI texture and shape in a machine learning scheme"

[14] https://www.kaggle.com/navoneel/brain-mri-images-for-brain-tumor-detection

[15] https://www.geeksforgeeks.org/python-image-classification-using-keras/

[16] https://www.tutorialsteacher.com/python/create-gui-using-tkinter-python

[17] https://data-flair.training/blogs/brain-tumor-classification-machine-learning/

[18]https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/

BIODATA

Name: Gandrapu Sri Sai Lahari

Mobile No: +918374559491

Email: lahari.18bcn7120@vitap.ac.in

Permanent Address: Vijayawada, Andhra Pradesh, India