

BRAIN TUMOR DETECTION

An App Development Project Report Submitted
In partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

by

Muda Vennela

Mohammad Johnny Pasha

P.Ganesh

Roll No:20N31A05F6

Roll No:20N31A05F0

Roll No:20N31A05H2

Under the esteemed guidance of

Mrs.S. Archana
Assistant Professor



Department of Computer Science and Engineering

Malla Reddy College of Engineering & Technology

(Autonomous Institution- UGC, Govt. of India)

(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA & NAAC with 'A' Grade)

Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100

website: www.mrcet.ac.in

2020-2024



Malla Reddy College of Engineering & Technology

(Autonomous Institution- UGC, Govt. of India)

(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA & NAAC with 'A' Grade)

Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100

website: www.mrcet.ac.in

CERTIFICATE

This is to certify that this is the bonafide record of the project entitled “BRAIN TUMOR DETECTION”, submitted by M.Vennela(20N31A05F6), Md.Johnny Pasha(20N31A05F0) and P. anesh(20N31A05H2) of B. Tech in the partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Science and Engineering, Department of CSE during the year 2022-2023. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

Internal Guide

Mrs. S. Archana
Assistant Professor

Head of the Department

Dr. S. Shanthi
Professor

External Examiner

DECLARATION

We hereby declare that the project titled “**Brain Tumor Detection**” submitted to Malla Reddy College of Engineering and Technology (UGC Autonomous), affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a result of original research carried-out in this thesis. It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

M. Vennela - (20N31A05F6)

Md. Johnny Pasha - (20N31A05F0)

P. Ganesh - (20N31A05H2)

ACKNOWLEDGEMENT

We feel ourselves honored and privileged to place our warm salutation to our college Malla Reddy College of Engineering and Technology (UGC-Autonomous) and our principal

Dr. S. Srinivasa Rao who gave us the opportunity to have experience in engineering and profound technical knowledge.

We express our heartiest thanks to our Head of the Department **Dr. S. Shanthi** for encouraging us in every aspect of our project and helping us realize our full potential.

We would like to thank our internal guide **Mrs.S.Archana** Assistant Professor, Associate Professor for their regular guidance and constant encouragement. We are extremely grateful to her valuable suggestions and unflinching co- operation throughout project work.

We would like to thank our Class Incharge **Mrs.M.Agnisha** who in spite of being busy with her duties took time to guide and keep us on the correct path.

We would also like to thank all the supporting staff of the Department of CSE and all other departments who have been helpful directly or indirectly in making our project a success.

We are extremely grateful to our parents for their blessings and prayers for the completion of our project that gave us strength to do our project.

M. Vennela - (20N31A05F6)

Md. Johnny Pasha - (20N31A05F0)

P. Ganesh - (20N31A05H2)

ABSTRACT

Brain tumor is a mass of tissue containing abnormal cells in the brain. Tumors can be benign and malignant. Benign tumors are not cancerous while malignant are cancerous. Although some tumors are not cancerous, detecting them in the early stages is very important to start proper treatment. The traditional way of detecting tumors involve a radiologist analyzing MRI images and giving a report on whether there is a tumor or not. But this method was difficult as the radiologists are supposed look at MRI scans and find the abnormal area by themselves. This was also a time taking process when the radiologist needs to analyze too many images. To solve this problem, we can use many deep learning algorithms. One of these deep learning methods to detect tumors in human brain in Convolutional Neural Networks or CNN. The proposed system builds upon the existing system. The extraction of the tumor mask is in the same way that was done in the K-Means method. But the proposed model uses Convolutional Neural Networks or commonly known as ConvNet or CNN for better image segmentation as it is currently the best deep learning algorithm for image classification and segmentation.

CONTENTS

CHAPTER NO	TITLE	PAGE NO
1	INTRODUCTION	1-2
	1.1 Problem Definition	
	1.2 Objective of the project	
	1.3 Existing System	
	1.4 Proposed System	
	1.5 Limitations of the project	
	1.6 Modules	
2	ANALYSIS	
	2.1 Introduction	
	2.2 Software Requirement Specification	
	2.2.1 Software Requirement	
	2.2.2 Hardware Requirement	
	2.3 Architecture	3 - 4
3	DESIGN	
	3.1 Introduction	
	3.2 Flow Diagram	
	3.3 Data Set Descriptions	
	3.4 Data Pre-processing Techniques	5 - 8
	3.5 Methods and Algorithms	
	3.6 Building a Model	
	3.7 Evaluation	
4	DEPLOYMENT AND EVALUATION	
	4.1 Introduction	9 - 14
	4.2 Implementation	
	4.3 Final Results	
5	CONCLUSION & FUTURE SCOPE	15 - 16
	5.1 Project Conclusion	
	5.2 Future Scope	
	5.3 References	

CHAPTER 1

INTRODUCTION

1.1 Problem Definition

Detecting brain tumor in human beings using deep learning techniques such as Convolutional Neural Networks.

1.2 Objectives of the project

The objective of this project is to create a real time application which will take MRI of brain as an input, pre-process the image, perform segmentation and highlight the ROI or Region of Interest where the tumor is located.

1.3 Existing System

The existing system used a K-Means Clustering Algorithm for Image Segmentation which was able to give satisfactory results in some cases but failed to give very accurate results in most cases. K-Means Segmentation's tumor masking would apply to the entire brain giving a result in such a way that the tumor area is significantly high which was misleading.

1.4 Proposed System

The proposed system builds upon the existing system. The extraction of the tumor mask is in the same way that was done in the K-Means method. But the proposed model uses Convolutional Neural Networks or commonly known as ConvNet or CNN for better image segmentation as it is currently the best deep learning algorithm for image classification and segmentation.

1.5 Limitations of the Project

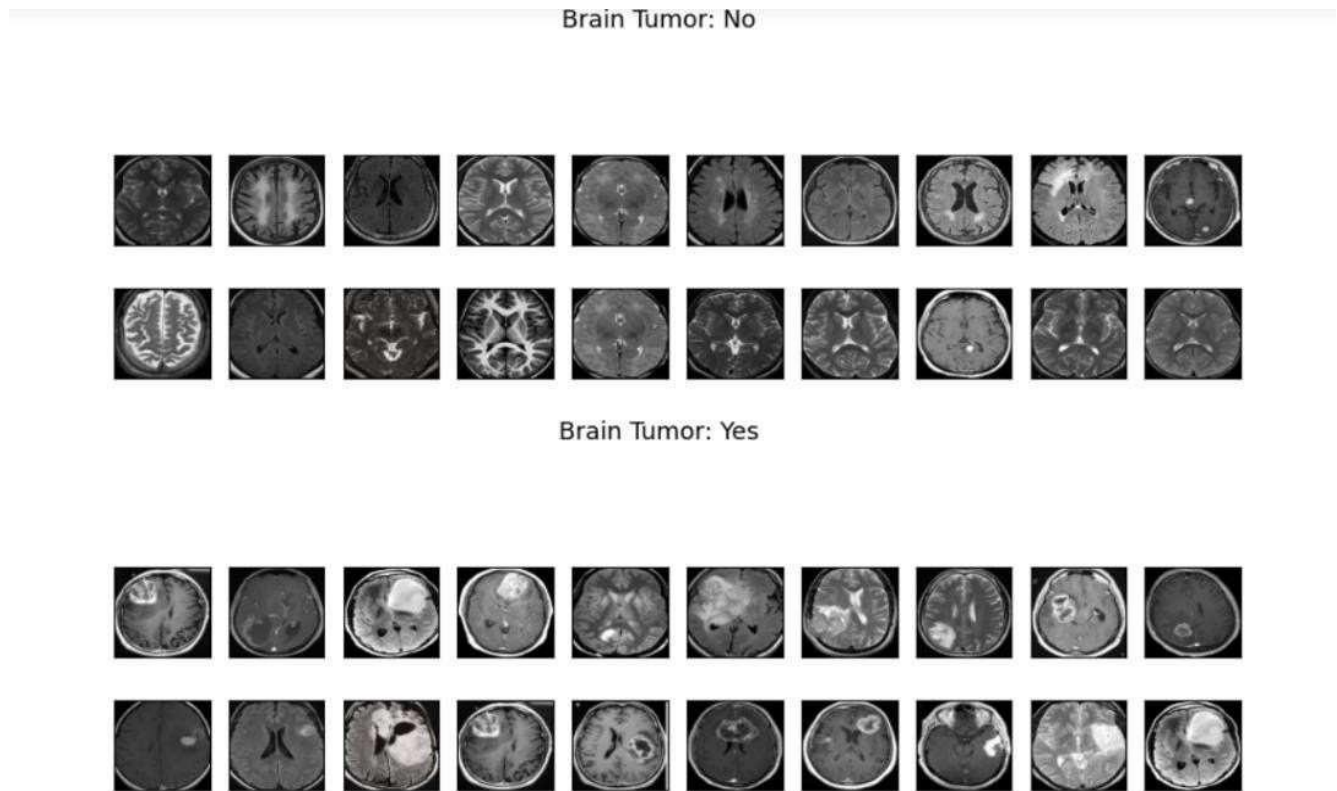
- As of now, the model only takes in black and white images to detect tumor.
- The model will not be able to perform segmentation if the image has the tumor in a significantly different shade when compared to the trained images.

1.6 Modules

- Data Collection and Pre-processing
- Building, training, and choosing the best model
- Model Evaluation

1.6.1 Data Collection and Pre-processing

The data set is collected from Kaggle and consists of 253 labelled images (98 labelled No, 155 labelled Yes). All the images in the dataset are resized to (240, 240) for better training and accurate results. Later, all the images were converted to 2 channel images (grey scale).



1.6.2 Building, Training, and choosing the best model

Out of 253 images 70% (69 No and 108 Yes) of the images from yes and no were used to train the model while 15% (15 No, 23 Yes) were used for validation and other 15% (14 No, 24 Yes) were used for testing. Then the training and validation data has been split into 21 batches (10 images each batch) and was trained through 30 epochs. The best model was obtained at the 23rd epoch with the training accuracy of 96% and validation accuracy of 93%.

1.6.3 Model Evaluation

The best model was evaluated with the testing set and an accuracy around 90% was obtained. This means the model was able to detect whether the tumor is present or not exactly from almost all the images.

CHAPTER 2

ANALYSIS

2.1 Introduction

This brain tumor detection algorithm is a simple yet very machine specification demanding model. Although the model runs without any problem, we might run into issues while training and building the model if we do not have a device with all the requirements. The issues could be longer training time or the algorithm not running at all because of missing software. Hence, we need to ensure that all the software requirements meet while running the application and have suitable hardware while trying to re-create the model.

The proposed model is a major step up from the existing model as we can clearly see in the chapters of this report that the proposed model was able to detect tumors very accurately in almost all cases while also displaying the exact tumor masking.

2.2 Software and Hardware Requirements

To run the detection algorithm, it is mandatory to have a computer with a good GPU and is also recommended to have all the software and devices drivers updated to the latest version. Detailed specifications are found below.

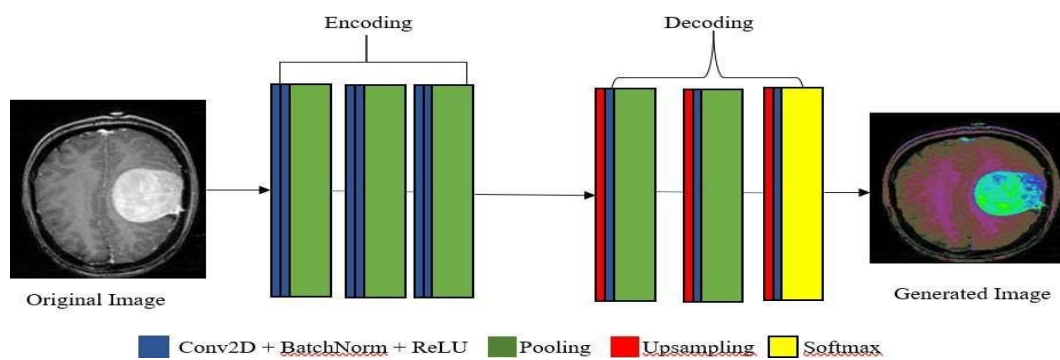
2.2.1 Software Requirements

- OS: Windows 10/11 or macOS 10.9 or above
- IDE/Text Editors: Jupyter Notebook, PyCharm, Visual Studio Code
- Python 3.8 or above
- OpenCV 4.2.5 or above, TensorFlow, Tkinter, Sci-kit Learn, Matplotlib

2.2.2 Hardware Requirements

- CPU: Intel Core i5 11th Gen or above or AMD Ryzen 5 4000 Series or above
- GPU: Nvidia GTX 1070 or RTX 2060 and above or an AMD Radeon Equivalent

2.3 Architecture



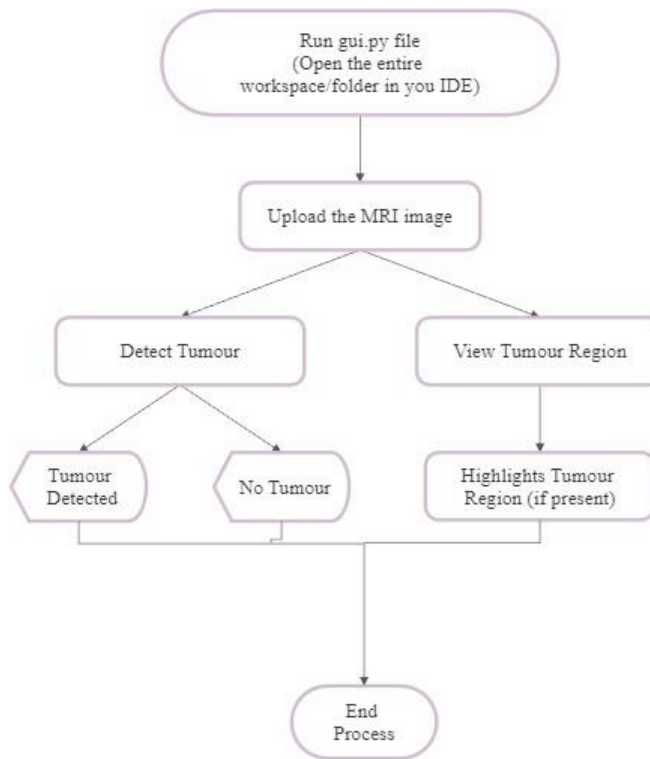
CHAPTER 3

DESIGN

3.1 Introduction

The application or detection model was designed in such a way that the user can upload any brain MRI scan and do two different operations, detect tumor and view tumor, both in couple of clicks. Detection of tumor is instantaneous where as the viewing the tumor region is a 2-step process which happens very quickly.

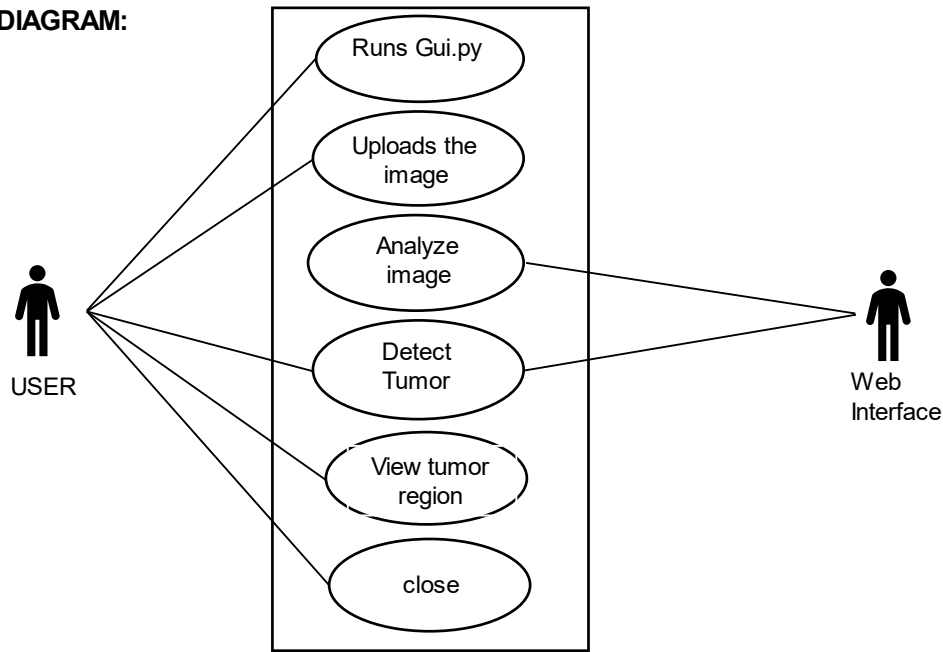
3.2 Flow Diagram



3.3 Dataset Descriptions

The dataset has been collected from Kaggle.com. Dataset consists of 253 MRI images of human brain with 155 being labelled YES for tumorous images and 98 are labelled as NO non-tumorous images. The images are greyscale and are of different dimensions/sizes. Noise is minimum to zero and there were no obscured images as well. Although it is a small dataset, it is the only dataset of brain MRI scans available on the internet for public and can be increased by using Data Augmentation method.

USE CASE DIAGRAM:



3.4 Data Pre-processing Techniques

Every image has been pre-processed in 3 steps:

1. Image has been cropped in way such that the borders were minimal while maintaining the quality of the image and making sure that part of the image where the brain is has not been cropped.
2. Resize all the images to (240, 240, 3) (width, height, channels) because the images come in different sizes, and we need all of them to be in equal sizes to be able train the neural network properly to obtain a good detection model.
3. Apply normalization to ensure that the input parameter has similar data distribution. In this case, we normalize the image to scale the pixel values in the range 0-1.

3.5 Methods and Algorithms

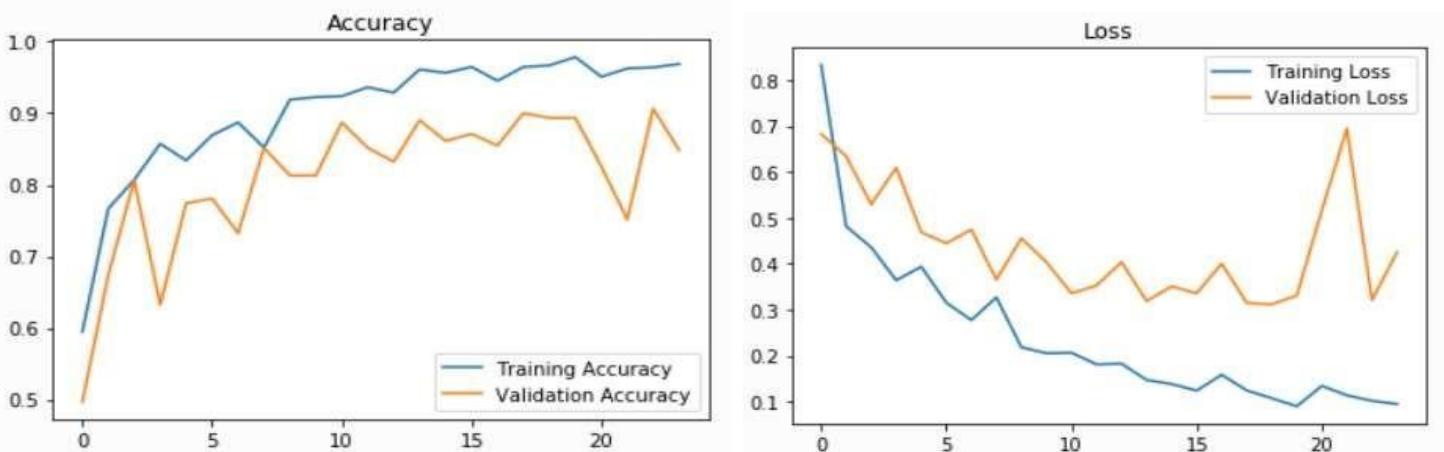
The method used to segment the tumor region from the MRI scan is Region Based Segmentation. In this method, the algorithm runs through the entire image and separates each region based on the pre-defined rules, highlighting the region of interest. This method has been built using CNN algorithm.

3.6 Building A Model

The pre-processed image is fed to the CNN model which has an input layer, convolution layers and a fully connected layer which activates a specific neuron to give specific output or decision. The input image forms the input layer. The image is represented as a 240x240 pixel matrix. Each pixel reveals certain features.

In the first convolution layer 8 filters of 3x3 size kernels each are applied over the input image by sliding through the position one by one and in total 8 feature maps are produced this process is called feature extraction. These features are then fed to ReLU activation function which performs a threshold operation to each input element where values less than zero are set to zero. A max pooling layer of 2x2 window size is applied to the output of ReLU layer which results into down sampling the feature maps into 128x128 pixel size.

The output of previous convolution layer serves as input to second convolution layer. Second convolution layer consists of 12 filters of 3x3 size kernels which are applied to each of the 8 features maps obtained from previous layer. Similar ReLU and max pooling operations are performed to produce down-sampled data of 64x64 pixel. Same operations are continued for the third convolution layer where 24 filters of 3x3 size kernels are used. Again, ReLU operation is applied and fed to the max pooling layer which produces 32x32 pixel data. The operations performed throughout the three layers extracts prominent and important features necessary for accurate classification. The output of the third convolutional layer is 24 features maps of 32x32 pixels each. These are then flattened to a single vector of length $32 \times 32 \times 24 = 24576$, which is used as the input to a fully connected layer with 106 neurons (or elements). This feeds into another fully connected layer with 2 neurons, one for each of the classes, which is used to determine the class of the image, that is, tumorous or non-tumorous. Unsupervised learning is used for tumor segmentation as the dataset used for the proposed model does not contain ground truth segmentation results required to train the model. Variety of techniques are available for segmentation using unsupervised learning, we are using CNN for our proposed model. The decoded data regenerated from the encoded data has close resemblance with input image size of the output image is same as that of the input image. The



decoded image looks just like the input

image but carries only the important features in the image and hides the irrelevant noisy data. Thus, by the process of encoding and decoding unwanted noise can be reduced which improves the accuracy in locating region of interest.

This model is then built and compiled. To obtain better results, we use Adam optimizer. After compilation, we train the model by using pre-processed images that were split for training and validation. The images are split into 21 batches (10 per each batch) and will be ran through 30 epochs or more if needed. The epoch where we get the best validation accuracies will be evaluated for choosing the best model.

3.7 Evaluation

The models have been evaluated by testing them with the test image dataset and re-testing them with train and validation images. We have used F1 Score, Accuracy to evaluate our models. The best model that was chosen was giving an accuracy of 94% on training set, 91% on validation and 90% on testing set. Upon reinterpretation, the positive example used to test the model were about 50% resulting that our model was able to classify the images as tumorous or non-tumorous most of the time. Finally, the F1 Score obtained on validation and training sets were 0.91 and 0.89 respectively.

	Validation Set	Training Set
F1 Score	0.91	0.89
Accuracy	91%	90%

Table 1: F1 scores and accuracies of Validation and Training Sets

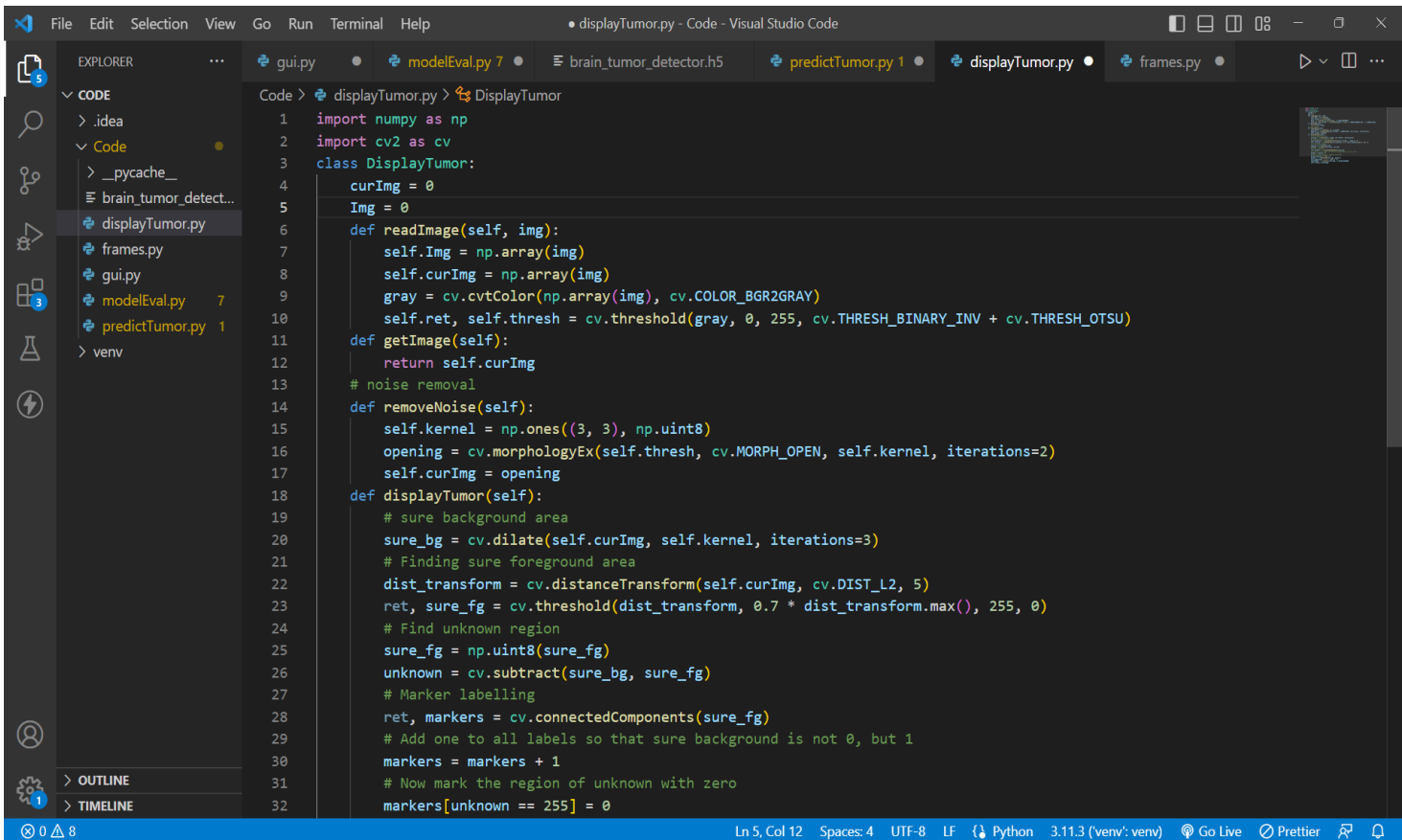
CHAPTER 4

DEPLOYMENT AND RESULTS

4.1 Introduction

The following code was written in Python. The libraries used were OpenCV, TensorFlow, TKinter Sci-Kit Learn, Matplotlib and other basic essential libraries. OpenCV was used for image pre-processing. TensorFlow was used to build the CNN model. Different layers such as Conv2d, MaxPool- ing2D, BatchNormalization, Flatten and call-backs such as tensorboard and ModelCheckpoints were used to build the model. Sci-Kit Learn was used to find the accuracy and F1 score of the model. It was also used to create the training and testing sets. Below is the entire source code.

4.2 Implementation



```
File Edit Selection View Go Run Terminal Help
• displayTumor.py - Code - Visual Studio Code
EXPLORER
CODE
> .idea
> Code
  > __pycache__
    brain_tumor_detect...
  displayTumor.py
  frames.py
  gui.py
  modelEval.py 7
  predictTumor.py 1
  venv
Code > displayTumor.py > DisplayTumor
1 import numpy as np
2 import cv2 as cv
3 class DisplayTumor:
4     curImg = 0
5     Img = 0
6     def readImage(self, img):
7         self.Img = np.array(img)
8         self.curImg = np.array(img)
9         gray = cv.cvtColor(np.array(img), cv.COLOR_BGR2GRAY)
10        self.ret, self.thresh = cv.threshold(gray, 0, 255, cv.THRESH_BINARY_INV + cv.THRESH_OTSU)
11    def getImage(self):
12        return self.curImg
13    # noise removal
14    def removeNoise(self):
15        self.kernel = np.ones((3, 3), np.uint8)
16        opening = cv.morphologyEx(self.thresh, cv.MORPH_OPEN, self.kernel, iterations=2)
17        self.curImg = opening
18    def displayTumor(self):
19        # sure background area
20        sure_bg = cv.dilate(self.curImg, self.kernel, iterations=3)
21        # Finding sure foreground area
22        dist_transform = cv.distanceTransform(self.curImg, cv.DIST_L2, 5)
23        ret, sure_fg = cv.threshold(dist_transform, 0.7 * dist_transform.max(), 255, 0)
24        # Find unknown region
25        sure_fg = np.uint8(sure_fg)
26        unknown = cv.subtract(sure_bg, sure_fg)
27        # Marker labelling
28        ret, markers = cv.connectedComponents(sure_fg)
29        # Add one to all labels so that sure background is not 0, but 1
30        markers = markers + 1
31        # Now mark the region of unknown with zero
32        markers[unknown == 255] = 0
Ln 5, Col 12 Spaces: 4 UTF-8 LF Python 3.11.3 (venv: venv) Go Live Prettier
```

The image shows a PyCharm IDE interface with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The breadcrumb at the top indicates the current file is frames.py in a Code - Visual Studio Code workspace. The Explorer sidebar on the left shows a project structure with folders .idea and Code, and files __pycache__, brain_tumor_detect..., displayTumor.py, frames.py (selected), gui.py, modelEval.py (with a '7' next to it), predictTumor.py (with a '1' next to it), and venv. The Code editor displays the contents of frames.py, showing a Python script with line numbers 66 through 87. The script defines a listWF, uses tkinter for window management, and includes methods for displaying and reading images. The status bar at the bottom shows 'Ln 3, Col 22', 'Spaces: 4', 'UTF-8', 'LF', 'Python', '3.11.3 (venv: venv)', 'Go Live', 'Prettier', and a file icon. The bottom right corner of the status bar shows '0 8'.


```

File Edit Selection View Go Run Terminal Help
gui.py - Code - Visual Studio Code

EXPLORER
CODE
> .idea
> Code
> __pycache__
> brain_tumor_detect...
> displayTumor.py
> frames.py
> gui.py
> modelEval.py 7
> predictTumor.py 1
> venv

Code > gui.py > Gui > check
3 from tkinter import filedialog
4 import cv2 as cv
5 from frames import *
6 from displayTumor import *
7 from predictTumor import *
8 class Gui:
9     MainWindow = 0
10    listOfWinFrame = list()
11    FirstFrame = object()
12    val = 0
13    fileName = 0
14    DT = object()
15    wHeight = 700
16    wWidth = 1180
17    def __init__(self):
18        global MainWindow
19        MainWindow = tkinter.Tk()
20        MainWindow.geometry('1200x720')
21        MainWindow.resizable(width=False, height=False)
22        self.DT = DisplayTumor()
23        self.fileName = tkinter.StringVar()
24        self.FirstFrame = Frames(self, MainWindow, self.wWidth, self.wHeight, 0, 0)
25        self.FirstFrame.btnView['state'] = 'disable'
26        self.listOfWinFrame.append(self.FirstFrame)
27        WindowLabel = tkinter.Label(self.FirstFrame.getFrames(), text="Brain Tumor Detection", height=1, width=40)
28        WindowLabel.place(x=320, y=30)
29        WindowLabel.configure(background="White", font=("Comic Sans MS", 16, "bold"))
30        self.val = tkinter.IntVar()
31        RB1 = tkinter.Radiobutton(self.FirstFrame.getFrames(), text="Detect Tumor", variable=self.val, value=1, comma
32        RB1.place(x=250, y=200)
33        RB2 = tkinter.Radiobutton(self.FirstFrame.getFrames(), text="View Tumor Region", variable=self.val, value=2, c
34        RB2.place(x=250, y=250)

```

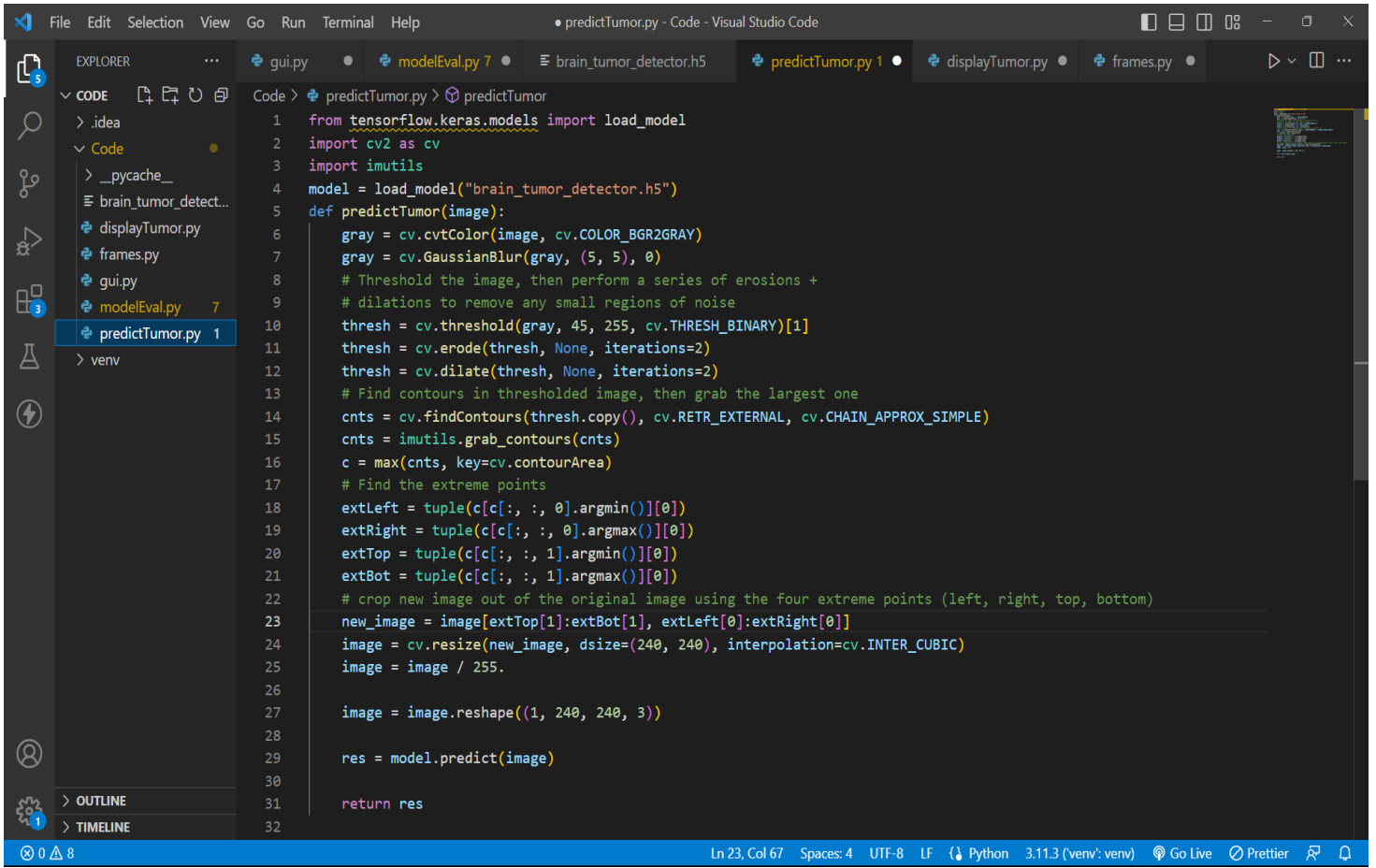
```

File Edit Selection View Go Run Terminal Help
gui.py - Code - Visual Studio Code

EXPLORER
CODE
> .idea
> Code
> __pycache__
> brain_tumor_detect...
> displayTumor.py
> frames.py
> gui.py
> modelEval.py 7
> predictTumor.py 1
> venv

Code > gui.py > Gui > check
51 global mriImage
52 #print(mriImage)
53 if (self.val.get() == 1):
54     self.listOfWinFrame = 0
55     self.listOfWinFrame = list()
56     self.listOfWinFrame.append(self.FirstFrame)
57     self.listOfWinFrame[0].setCallObject(self.DT)
58     res = predictTumor(mriImage)
59     if res > 0.5:
60         resLabel = tkinter.Label(self.FirstFrame.getFrames(), text="Tumor Detected", height=1, width=20)
61         resLabel.configure(background="White", font=("Comic Sans MS", 16, "bold"), fg="red")
62     else:
63         resLabel = tkinter.Label(self.FirstFrame.getFrames(), text="No Tumor", height=1, width=20)
64         resLabel.configure(background="White", font=("Comic Sans MS", 16, "bold"), fg="green")
65         resLabel.place(x=700, y=450)
66 elif (self.val.get() == 2):
67     self.listOfWinFrame = 0
68     self.listOfWinFrame = list()
69     self.listOfWinFrame.append(self.FirstFrame)
70     self.listOfWinFrame[0].setCallObject(self.DT)
71     self.listOfWinFrame[0].setMethod(self.DT.removeNoise)
72     secFrame = Frames(self, MainWindow, self.wWidth, self.wHeight, self.DT.displayTumor, self.DT)
73     self.listOfWinFrame.append(secFrame)
74     for i in range(len(self.listOfWinFrame)):
75         if (i != 0):
76             self.listOfWinFrame[i].hide()
77             self.listOfWinFrame[0].unhide()
78             if (len(self.listOfWinFrame) > 1):
79                 self.listOfWinFrame[0].btnView['state'] = 'active'
80     else:
81         print("Not Working")
82 mainObj = Gui()

```



4.1 Final Results

The results are as shown below. We can say that the proposed model is able to highlight the region of interest with minimum to no flaws. It is also accurate when compared to the existing model. The GUI is also very easy to use and can be understood very easily.

	Validation Set	Training Set
F1 Score	0.91	0.89
Accuracy	91%	90%

Table 2: F1 scores and accuracies of Validation and Training Sets

Output Screens:

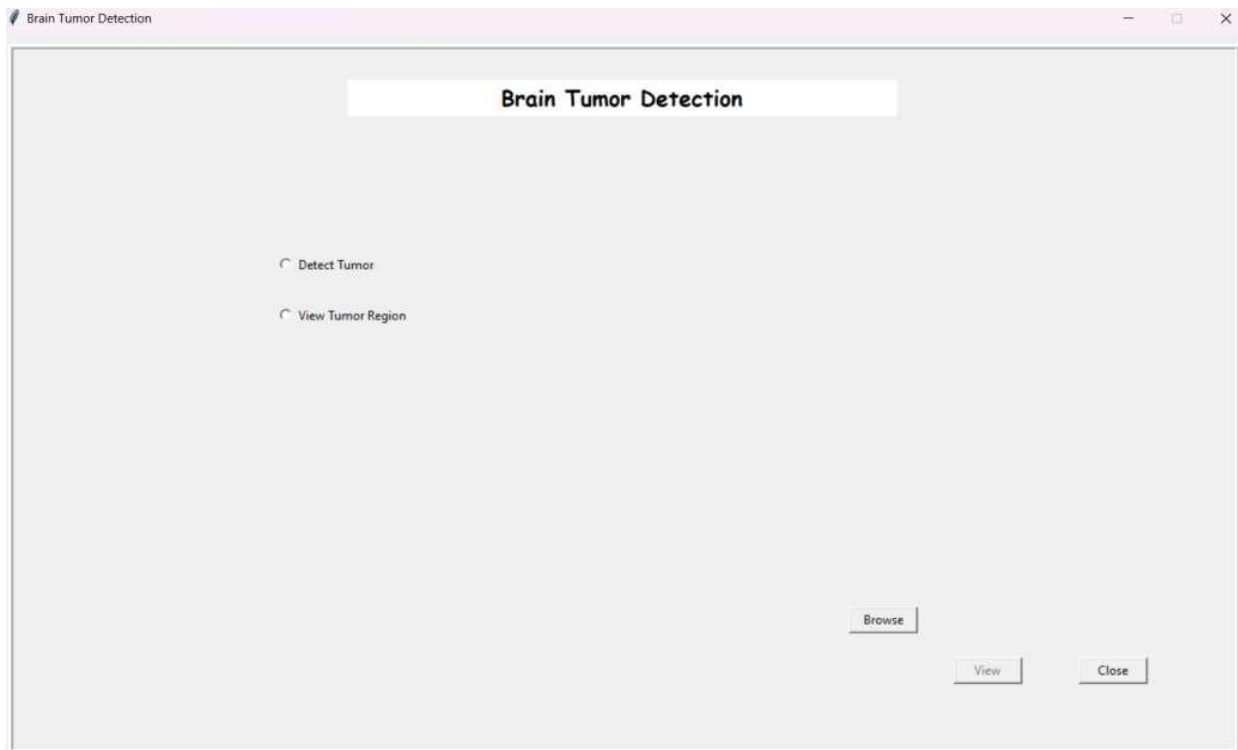


Fig 3. GUI window made using Tkinter

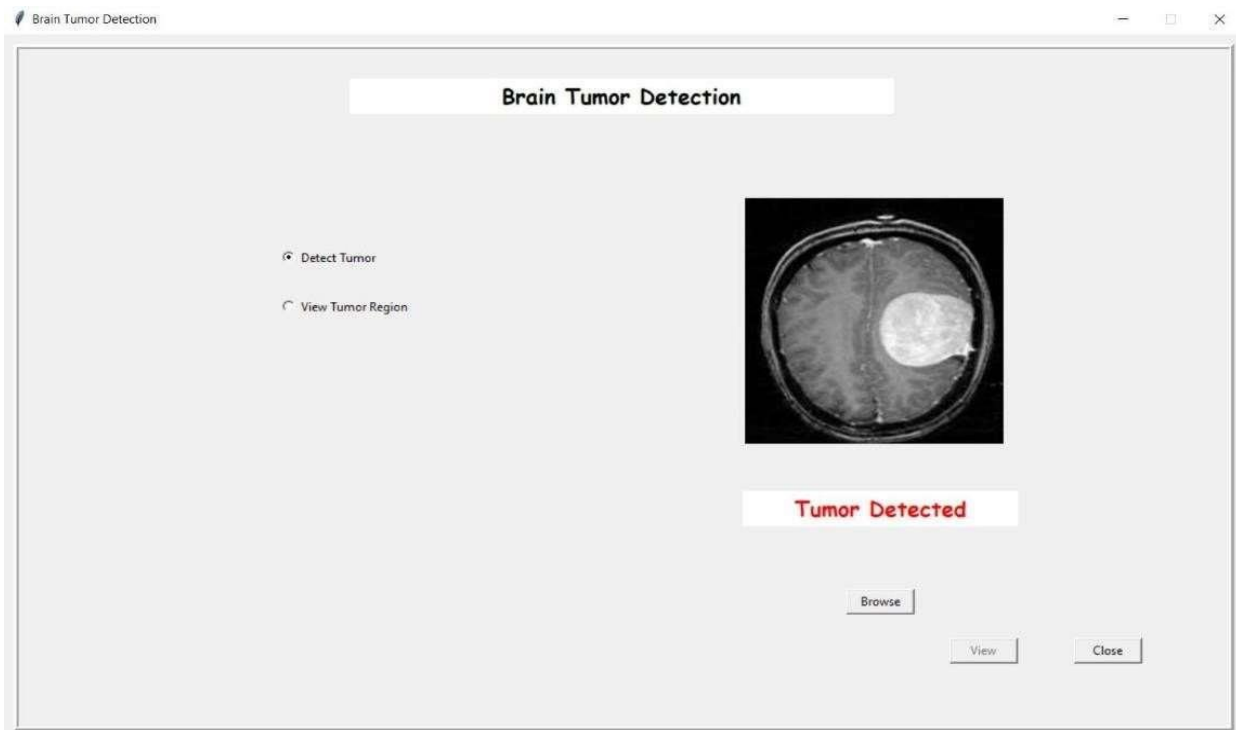


Fig 4. Tumour Detection

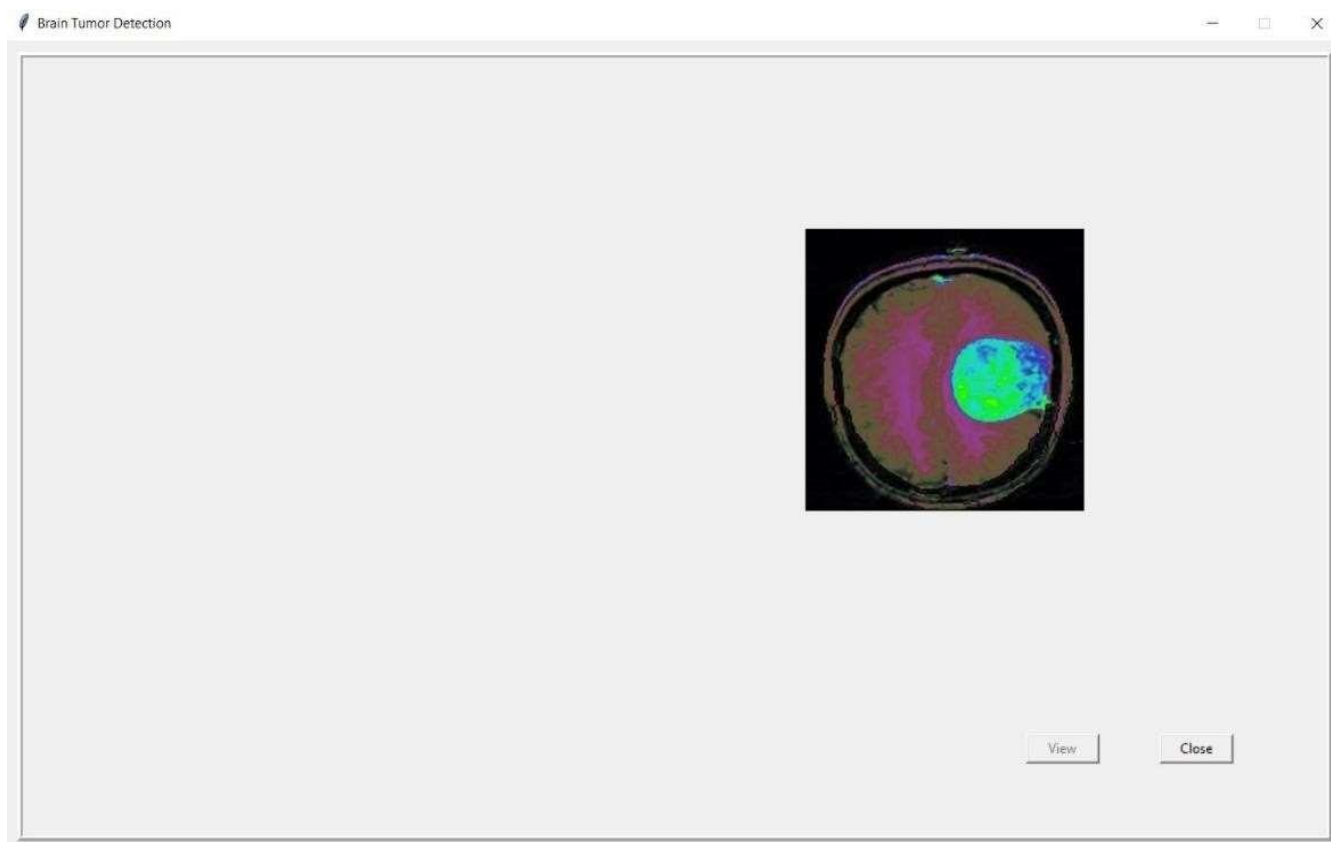


Fig 5. Viewing Tumour Region

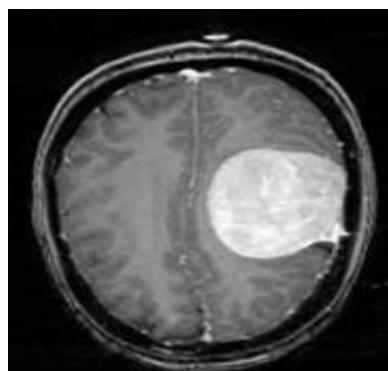


Fig 6. Original Image

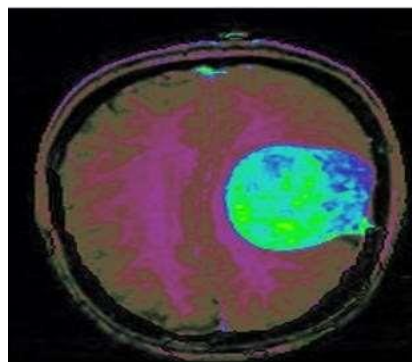


Fig 7. Region of Interest

CHAPTER 5

CONCLUSION & FUTURE SCOPE

5.1 Project Conclusion

The time-consuming process of brain tumor detection is thus simplified by automation. An accuracy of 90% on testing data is achieved by the proposed model for detecting brain tumor. After detecting the tumor with convolutional neural networks segmentation techniques like region-based Segmentation are applied over the tumorous image to locate the region of tumor in the image. As shown in Fig 7, the model was able to highlight the tumorous region almost exactly where the tumor is present and as shown in Fig 3, 4, 5 the GUI usable without any complications. Thus, an efficient model for detection and segmentation of brain tumor is build which saves human efforts and time.

5.2 FUTURE SCOPE

- This model can be improved in a way that it can be used to detect tumors from other MRI scans where the tumor area is not as white as shown in most of our dataset images.
- Right now, the ROI is generated in a 2-step process through the GUI which can be a bit confusion for the user. This can be altered to be obtained in a single step which is easier and less confusing. On top of that, the entire GUI can be made more user-friendly.

5.3 REFERENCES

- Dataset: <https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection?resource=download>
- Paper: Bal, Abhishek, Minakshi Banerjee, Punit Sharma, and Mausumi Maitra. "Brain tumor segmentation on MR image using K-Means and fuzzy-possibilistic clustering." In *2018 2nd international conference on electronics, materials engineering & nanotechnology (IEMENTech)*, pp. 1-8. IEEE, 2018.
- URL: <https://ieeexplore.ieee.org/document/8465390>