Medical Image Analysis and Tumor Detection

using deep learning techniques

A CAPSTONE PROJECT REPORT

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

BACHELOR OF TECHNOLOGY
in
Computer Science Engineering

by

Deepika Mangamuri(19BCE7041)
Mounika Sadineni (19BCE7074)
Kommu Bala Eswar(19BCN7003)

Under the Guidance of

DR. Nandha Kumar R

SCHOOL OF COMPUTER SCIENCE ENGINEERING
VIT-AP UNIVERSITY
AMARAVATI- 522237

DECEMBER 2022

CERTIFICATE

This is to certify that the Capstone Project work titled "**Medical Image Analysis and Tumor Detection**" that is being submitted by **Deepika Mangamuri(19BCE7041)**
Mounika Sadineni (19BCE7074), Kommu Bala Eswar(19BCN7003) 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.

Dr. Nandha Kumar R
Guide

The thesis is satisfactory / unsatisfactory

Internal Examiner **External Examiner**

Approved by

PROGRAM CHAIR **DEAN**
B. Tech. CSE School of Computer Science Engineering

ACKNOWLEDGEMENTS

I would like to express my gratitude to **Dr. G. Viswanathan,** Chancellor, **Dr. Sekar Viswanathan,** Vice president, **Dr. S. V. Kota Reddy,** Vice Chancellor and Dean **Dr. Sudha S V**, School of Computer Science and Engineering, for providing a conducive environment to work during the tenure of the course.

In jubilant mood, I express ingeniously my whole-hearted thanks to **Dr. Saroj Kumar Panigraphy,** the Program Chair and Professor, School of Computer Science and Engineering and it's my pleasure to express deep sense of gratitude to **Dr. Nandha Kumar R**, Associate Professor, School of Computer Science and Engineering, VIT-AP, for his constant guidance, continual encouragement, understanding; more than all, he taught me patience in my endeavor.

It is indeed my pleasure to thank all faculty members working as limbs of our university for their selfless enthusiasm coupled with timely encouragement showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support.

At last, but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly towards the successful completion of this project.

Place: Amaravati
Date: 29-12-2022

Deepika Mangamuri
Mounika Sadineni
K. Bala Eswar

ABSTRACT

The most prevalent and deadly disease, brain tumors, have an extremely low life expectancy at their most advanced stage. Thus, planning the patient's therapy is a crucial step in enhancing their quality of life. The evaluation of tumors in the brain, lung, liver, breast, prostate, etc., typically uses a variety of imaging modalities, including computed tomography (CT), magnetic resonance imaging (MRI), and ultrasound images. MRI images are employed in particular in this study to identify brain malignancies. Manually classifying a tumor as opposed to a non-tumor at a specific time is impossible due to the enormous amount of data produced by MRI scans. However, it has drawbacks because only a few photos can be given good quantitative values. Therefore, reliable and automatic classification systems are crucial to lowering the human mortality rate. Automatic brain tumor classification is a complicated issue due to the significant spatial and structural diversity of the areas around brain tumors. This study suggests utilizing Convolutional Neural Networks (CNN) classification to detect brain tumors automatically. According to experimental findings, the CNN archives outperform all other state-of-the-art approaches with 95% accuracy and only moderate complexity. It also visualizes the tumor area using the Watershed algorithm.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

Medical imaging is the method and procedure used to provide visual depictions of a body's interior for medical analysis and clinical intervention, as well as to show how specific organs or tissues work. Medical imaging aims to identify and cure disease and disclose internal structures covered by the skin and bones. Medical imaging also creates a database of typical anatomy and physiology to detect anomalies.

The globe has been afflicted by and lost many lives to brain tumors, one of the most frequent and deadly brain illnesses. Cancer is when cancer cells grow in the brain's tissues. According to a recent cancer study, more than 1 lakh people worldwide receive a brain tumor diagnosis each year. Despite consistent efforts to deal with brain tumor consequences, statistics indicate unfavorable outcomes for tumor patients. To counter this, researchers are utilizing computer vision to understand better cancers in their early stages and how to treat them with cutting-edge therapies.

The two most common procedures to determine the presence of a tumor and pinpoint its location for different treatment choices are magnetic resonance imaging (MR imaging), and computed tomography (CT) scans of the brain. Due to their portability and greater capacity to provide high-definition images of diseased tissues, these two scans are still employed often. Several other treatments are currently available for tumors, including chemotherapy, radiation therapy, and surgery. The size, type, and tumor stage visible in the MR picture are just a few variables influencing the treatment choice. It is also responsible for determining whether cancer has spread to other body parts.

MRI is the most frequently used method to image brain tumors and pinpoint their location. In contrast to other approaches, the traditional method for classifying and identifying tumor cells in CT and MR images continues to be highly supported for human assessment. Because they are non-destructive and non-ionizing, MR pictures are mainly used. High-definition images provided by MR imaging are frequently used to find brain malignancies. MRI uses a variety of methods, including T1- and T2-weighted images. There are various methods for processing photographs,

including pre-processing, segmenting pictures, enhancing pictures, extracting features, and classifiers.

Automated brain tumor identification and categorization are the focus of our investigation. MRI and CT scan frequently examine the brain's anatomical structure. The project aims to identify tumors using a straightforward Graphical User Interface in brain MR images (GUI). The primary goal of brain tumor detection is to support clinical diagnosis. By integrating various techniques, the goal is to create an algorithm that ensures the presence of a tumor, hence creating a reliable way of tumor detection in MR brain pictures. Tumor identification, segmentation, thresholding, and image acquisition are used.

This research aims to remove tumors from MR brain images and show them in a way that is easy to understand for everyone. This effort aims to present helpful information to the users in a more digestible style, particularly for the medical team caring for the patient. This study is to find a method to use a convolution neural network to determine whether a tumor is present in the provided MR brain image. After that Watershed Segmentation technique is used to determine its stage and also to visualize the tumor's location.

1.1 OBJECTIVES

The following are the objectives of this project:

- To develop a platform for people that is
  - low cost
  - reliable
  - Accessible way of identifying tumors.
- To train the networks on a large set of pre-classified "labelled" images to achieve high accuracy of image classification on new unseen images.
- To make the real-time diagnosis possible using CNN algorithm for image processing.
- To enable a collaborative approach towards continually increasing the tumor database for improved CNN classification accuracy and tracking for outbreaks.

1.2 BACKGROUND AND LITERATURE SURVEY

Because the outcome is so critical for the treatment of patients, the resilience and predicting the accuracy of the algorithms are particularly important in medical diagnosis. For prediction, a variety of well-liked classification and clustering methods are used. A medical image's representation is intended to be condensed into a meaningful image through clustering, which also makes the image easier to evaluate. Algorithms for clustering and classification are being developed with the goal of improving the diagnosis process's ability to forecast anomalies.

We give a brief overview of the various clustering strategies that have been put forth from 2002 to 2018 in the literature survey. We have read a few publications, each of which takes a different tack when it comes to parameter segmentation. The following lists the summary of each paper.

- "Brain MR Image Segmentation for Tumor Detection Using Artificial Neural Networks," International Journal of Engineering and Technology (IJET), Vol. 5, No 2, April-May 2013. Monica Subashini.M. and Sarat Kumar Sahoo.

A method for spotting tumors in brain MR images has been proposed by Monica Subashini and Sarat Kumar Sahoo. They also worked on a variety of techniques, such as backpropagation networks for categorizing tumor cells from brain MRI pictures, pulse-coupled neural networks for strengthening the brain MRI images, and noise removal techniques. The backpropagation network aids in the prediction of a tumor in a brain MR image, and they noticed picture augmentation and segmentation while using their proposed technique.

- K. Sudharani, T. C. Sarma and K. Satya Rasad, "Intelligent Brain Tumor lesion classification and identification from MRI images using a K-NN technique," 2015 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), Kumaracoil, 2015, pp. 777-780. DOI: 10.1109/ICCICCT.2015.7475384

To locate and contain the fully hysterical section within the aberrant tissues, K. Sudharani et al. applied a K-nearest neighbour technique to the MR images. Although the process for the suggested work is slow, it provides beautiful results. The sample training step is what determines accuracy. Magnetic Resonance Imaging Techniques research on Brain Tumor Identification.

- S. Li, J.T. Kwok, I.W Tsang, and Y. Wang, ―Fusing Images with Different Focuses using Support Vector Machines, Proceedings of the IEEE Transaction on Neural Networks, China, November 2007.

According to Li et al., optical lenses with lengthy focal lengths make it difficult to perform edge detection, picture segmentation, and matching. Wavelet-based picture fusion is one of the various approaches that researchers have previously suggested for this mechanism. A support vector machine (SVM) and a discrete wavelet frame transform (DWFT) can be used to enhance the wavelet function (SVM). The authors of this research tested five sets of 256-level pictures. According to experimental findings, this technique is more effective and reliable because it is unaffected by consistency checks and activity level assessments. Dynamic ranges are not taken into account when computing, and the paper is only focused on one fusion-related task.

- Artificial Neural Networks for MRI Image-Based Tumor Detection

In many diagnostic and therapeutic applications, automatic flaws detection in MR images is crucial. To improve accuracy, yield, and speed up diagnosis, our work proposed an automated brain tumor detection technique. The tissues will be divided into two categories: normal and pathological. The MR images presented here come from both normal and diseased brain tissues. This technique performs this classification using a neural network. The goal of this research is to automatically categorize brain tissues into normal and abnormal groups, saving the radiologist time and improving diagnosis accuracy and yield.

1.3 ORGANIZATION OF THE REPORT

The remaining chapters of the project report are described as follows:

- Chapter 2 contains the proposed system, methodology, and software details.
- Chapter 3 discusses the results obtained after the project was implemented.
- Chapter 4 concludes the report.
- Chapter 5 consists of codes.
- Chapter 6 gives references.

CHAPTER 2

MEDICAL IMAGE ANALYSIS AND BRAIN TUMOR DETECTION

This Chapter describes the proposed system, working methodology, software and hardware details.

2.1 PROPOSED SYSTEM

The proposed block diagram shows the working architecture of the project.

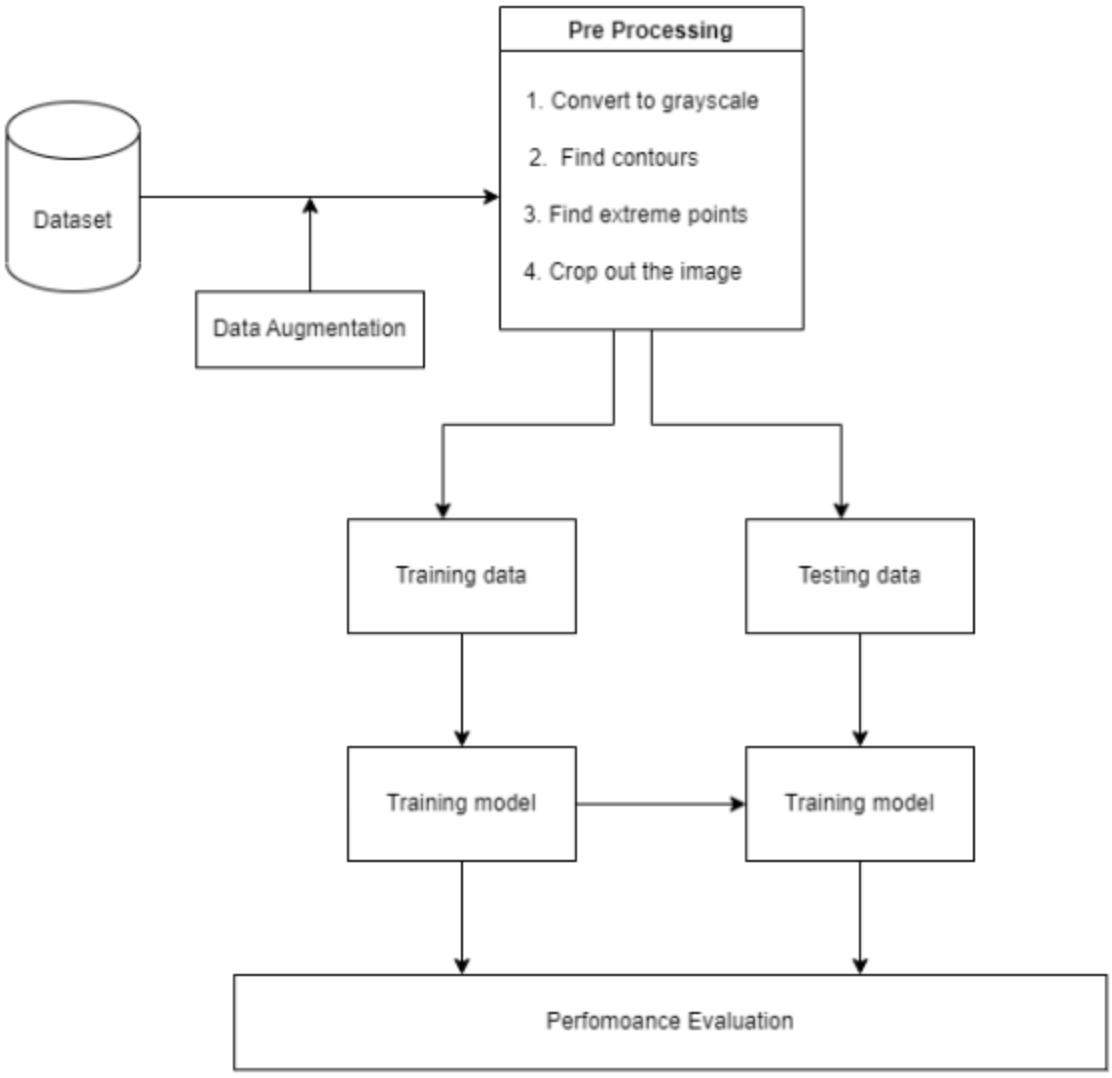


Figure1: System Block Diagram

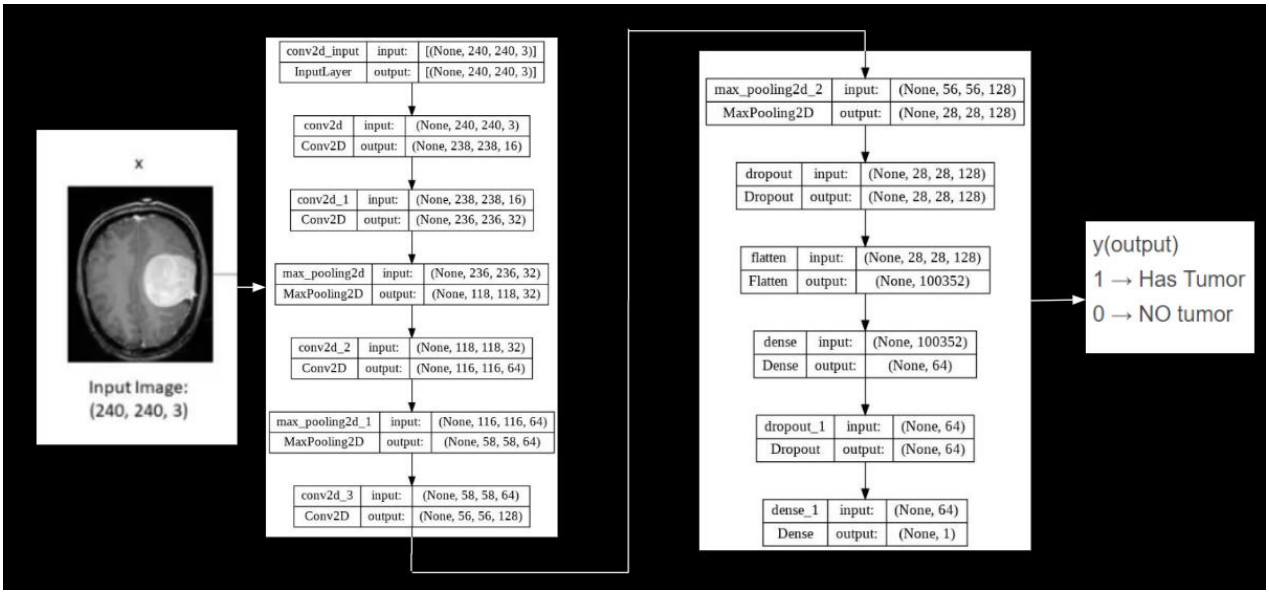


Figure2: CNN Architecture

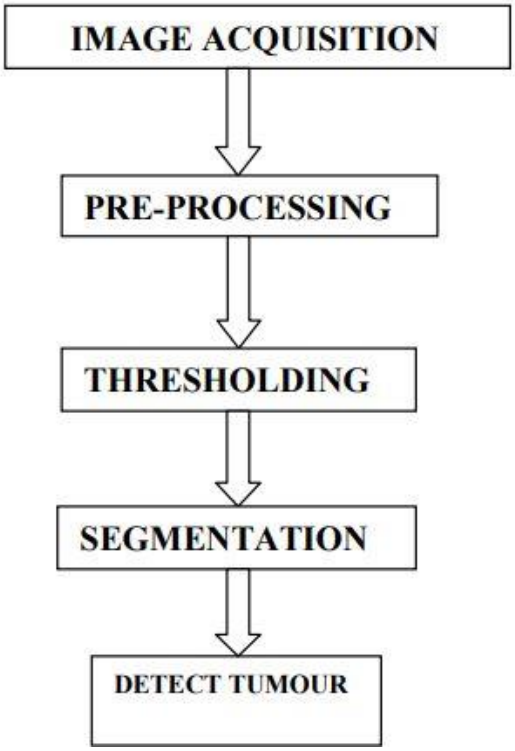


Figure2: View Tumor Architecture

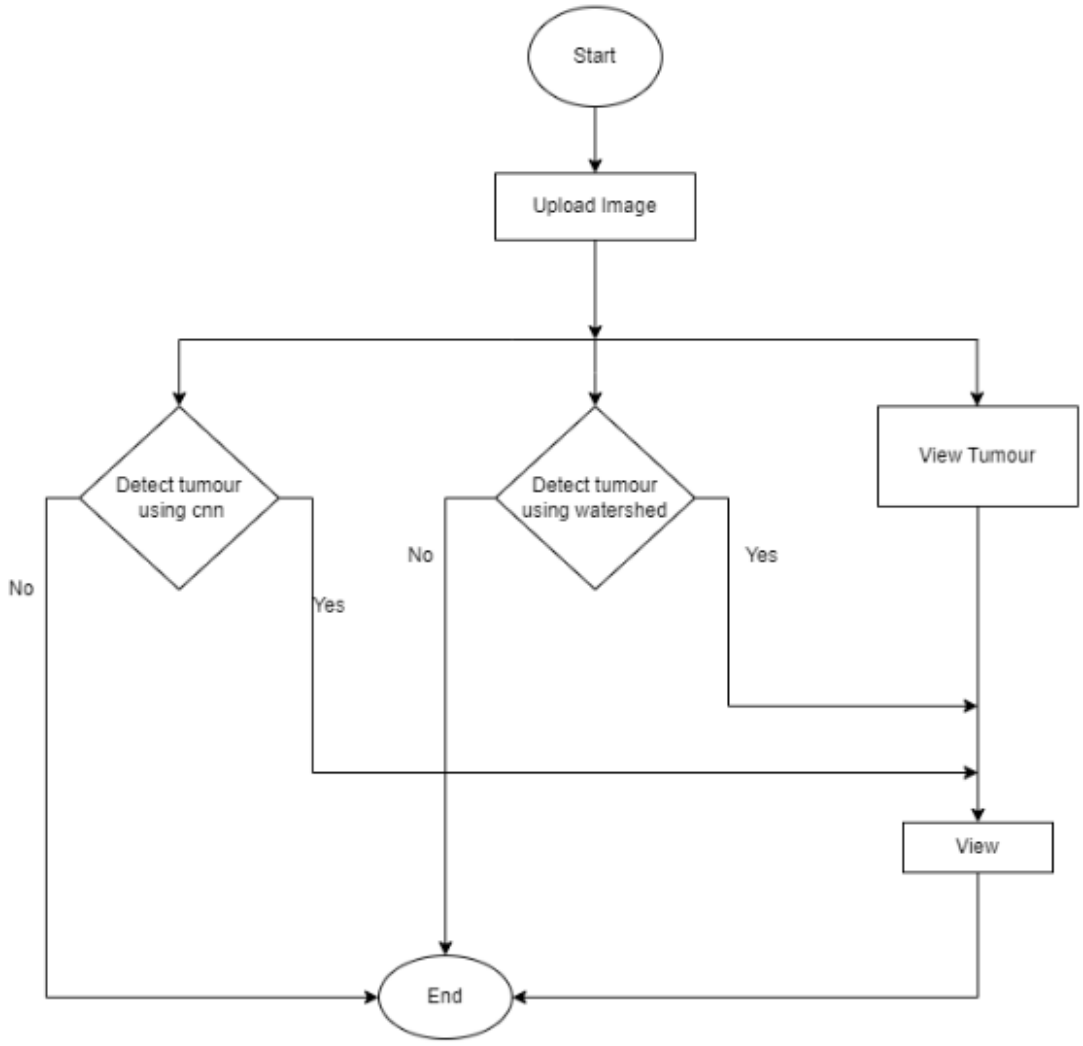


Figure2: GUI Architecture

2.2 WORKING METHODOLOGY

The proposed methodology consists of three major modules; Image Pre- Processing and Enhancement, Classification using CNN and Watershed, and Viewing tumors using the Watershed algorithm. Each module is unique in its way. The data used to test and train the system was downloaded from Kaggle. Data Augmentation is done on the dataset. Then The Bilateral Filter pre-processes the input image, while the Sobel Filter enhances the image appearance. The resultant image is then segmented using binary thresholding and subjected to morphological procedures. Following that, the four extreme points will be used to crop the MRI image. After that, CNN is used to classify the images and predict the tumor. Finally, Watershed Algorithm is used to visualize the tumor present in the MRI image. The Tkinter framework is used to create a GUI. The user can

upload a medical image into the system. Then the system will detect the tumor using CNN and locate the tumor if present. Based on the width of the tumor, it will display the stage in which the tumor is.

MODULE 1: IMAGE PREPROCESSING

1. *IMAGE PREPROCESSING:*

From Kaggle, the Brain MRI image dataset has been downloaded. These MRI scans are used as the first step's input. A crucial first step in raising the caliber of the brain MRI image is pre-processing. The brain MRI image is first transformed into a similar gray-scale image in the initial stage. The adaptive bilateral filtering approach is used to eliminate unwanted noise and the distorted noises that are present in the brain image. This raises the accuracy rate of classification and diagnosis.

2. *IMAGE ACQUISITION FROM DATASET:*

Taking an image from a dataset and processing it is how image acquisition is done in image processing. The reason it comes first in the workflow sequence is that processing cannot take place without an image. The captured image is entirely raw. Here, the image is processed using the native device's directory structure.

3. *CONVERT THE IMAGE FROM ONE COLOR SPACE TO ANOTHER:*

The program cv2.cvtColor(input image, flag) is used to convert an image's colour, and flag specifies the conversion methodology. In our approach, we transform the original image into a grayscale version.

4. *FILTERS:*

Filters are often implemented in image processing to reduce the image's high frequencies.
Bilateral filter: It also functions as an image noise-reduction and smoothing non-linear filter. It swaps out each pixel's intensity for a weighted sum of pixel intensities. The Gaussian distribution provides the foundation for this weight. Bilateral filtering smooth images while preserving edges. It creates a degree of grey based on how similar they are and how close they are symmetrically, then prefers closer values over farthest values.

5. *IMAGE ENHANCEMENT:*

Brain MRI visuals are more susceptible to noise than any other medical imaging, we need to optimize MRI image quality while limiting noise for better segmentation. In our research, we reduced the Gaussian noise in the brain MRI, which enhanced segmentation ability. Because we just require the malfunctioning tissues from the brain MRI, we may ignore the subtle details. There may be a very small area of abnormal region or tumor tissues in an MRI, but we need the largest one. Additionally, compared to other noise types, gaussian noise may appear more frequently in MRI scans. The Gaussian filter was used as a result. The add-Weighted technique was then used to optimize the image. The image is first blurred because we are aware that the majority of the high-frequency components can be suppressed by applying a smoothing filter to an image. We then take this smoothed image and dividing it from the original image; this difference is referred to as a mask. The majority of the high-frequency components that the smoothing filter has blocked will therefore be present in the output image. The high-frequency components of the original will be improved by reapplying this mask.

6. *IMAGE SEGMENTATION USING BINARY THRESHOLD*

The technique of segmenting a picture into various portions is called image segmentation. The main goal of this division is to keep the quality of the pictures while making it simple to examine and comprehend them. The edges of the objects in the photos are also tracked using this approach. The pixels are classified using this method based on their properties and intensity. These pieces take on the qualities of the complete actual picture, including intensity and resemblance. The body's contours are generated using the image segmentation method for therapeutic purpose.

The convenient method for segmenting images is thresholding. The process of turning a greyscale image into a binary image, where the two levels are allocated to pixels with values above or below the set threshold value. We employ the cv2.threshold(src, thresh, maxval, type[dst]) method in openCV

A single-channel array is subjected to fixed-level thresholding using this operation. The function is often used to extract a binary (binary) picture from a grayscale image in order to remove noise, which entails deleting pixels with values that are either too tiny or too high.

1. src - input image that should be gray scale

2. thresh - threshold value

3. maxval - It represents the value—in this case, 25, that will be provided if the pixel value exceeds (or occasionally falls short of) the setpoint.

4. type - thresholding type

We used cv2.THRESH_BINARY here

7. MORPHOLOGICAL OPERATIONS:

A structural element is added by morphological processes to an input image to produce an output image of the same size. When performing a morphological operation, each output pixel's value is determined by comparing it to it neighbors in the input image. In our project two morphological operations are used; erosion and dilation. In a picture, dilation involves adding pixels to object borders, and erosion involves taking pixels away from edges.

Thereafter, find contours in thresholded image, then grab the largest one. Simply said, a contour is a curve that connects all continuous points (along the perimeter) that have the same hue or luminosity.

8. EXTREME POINTS

Thereafter find the extreme points in the top bottom left and right sides of the MRI image using python tuples and crop new image out of the original image using the four extreme points (left, right, top, bottom)

MODULE 2: CLASSIFICATION MODELS

For identifying images, including any type of medical imaging, classification is the best strategy. We must import Keras and the additional packages we'll use to create the CNN for this stage.

- To filter the data and create a feature map, convolution is applied to the input information. The filter iteratively multiplies each element of the input image's matrix. The feature map records the outcome for each receptive field where convolution occurs. It is in charge of identifying features in pixels. For CNNs, we used 240 feature detectors. The input shape is the following parameter. For a colorful image, the number of channels is 3, and the

dimensions of the 2D array in each channel are provided along in the input shape tuple. The activation function is the last argument. The rectifier function is utilized.

- The pooling layer, which summarizes the features in a section of the feature map produced by a convolution layer, is the next layer. Here, the feature map region covered by the filter is utilized to select the maximum element from the pooling procedure, which is known as Max pooling. The result following the max-pooling layer would be a feature map that included the highest noticeable features from the prior feature map. We shrink the feature map's size in this stage. Typically, we establish pools that are 2x2 in size for maximum pooling.
- Then, in dropout, we arbitrarily disable a portion of a layer's neurons at each training phase by dropping out the neuron values. The dropout layer uses the rate as support for its claim. It stands for the percentage of input units that will be lost. Here, the rate is set to 0.2, which indicates that every epoch, 20% of the neurons in this layer will be deleted at arbitrarily.
- The Flattening layer is then used to combine all of the 2d arrays produced by the pooled feature maps into a single, long, continuous linear vector that is provided as an input to the classification process.
- Every neuron in the fully connected layer is assigned to a particular feature that might be present in a picture. This is accomplished by applying a fully connected layer. The likelihood that the feature is present in the image is represented by the value that the neuron passes on to the following layer.
- The vector we collected earlier will now be used as the input for the neural network by utilizing Keras' Dense function. The no of nodes in the hidden layer is the first parameter, output. You will require more computational power to fit the model as the number of dimensions increases. Selecting nodes in powers of two is a typical approach.

  classifier.add (Dense (output = 64))
- The output layer is the next layer that has to be added. Since we anticipate a binary result in this situation, we will employ the sigmoid activation function. Since we just anticipate the classes' expected probability, the output in this case is 1.

  model. add (Dense (units=1, activation=’sigmoid’))

In case of Watershed Algorithm depending upon the width of the tumor present we will classify it and determine the stage.

MODULE 3: VIEWING TUMOR USING WATERSHED ALGORITHM

This stage of MRI image processing involves scaling the picture input by image conversion and removing noise from the image using a filtering technique. After thresholding, or turning the image into binary, or black and white, segmentation of this image was carried out. Thresholding is used first, and then the Watershed segmentation approach is used to segment and isolate items that touch one another. The marked MRI imaging area is where the tumor is located. There was exactly one label in the image that was defined as having a maximum frequency value because the indicated area takes up the largest amount of space in the image. An edge-strength extraction operation called the morphological gradient produces symmetric edges between foreground and background regions. A binary edge image is then created by thresholding the resulting image.

1. IMAGE ACQUISITION:

The MRI images were obtained from freely accessible sources. The used pictures were.jpg files. Using the 'imread' command, this image might be read from either a computer or the internet.

2. PRE-PROCESSING

The MRI Images which are obtained from publicly available sources cannot be fed directly for the processing of these images as they contain noises such as external noise or noise due to the movement of the patient during the MRI scan etc. These noises needed to be removed and the images needed to be enhanced for the detection of efficient brain tumors.

3. THRESHOLDING

Thresholding is the simplest method of image segmentation. It is a non-linear operation that converts a grey-scale image into a binary image where the two levels are assigned to pixels that are below or above the specified threshold value. In Open CV, we use cv2.threshold() function:

cv2.threshold(src, thresh, maxval, type[dst])

This function applies fixed-level thresholding to a single-channel array. The function is typically used to get a bi-level (binary) image out of a grayscale image for removing noise, that is, filtering out pixels with too small or too large values. “maxval” is the set threshold value which compares with input values, when the input is greater than the set threshold value it gives output is set maxval value and it is shown with white colour in gray images. when the input pixel intensity values are less than the set threshold, its output is black colour. It converts or transforms the grayscale image, says I, to a bilateral or binary image. BW image output replaces each pixel of the input image with the greatest luminance level of value of 1 (white) and replaces all other pixels' value with 0 (black).

4. SEGMENTATION

Image segmentation is a technique of segregating the image into many parts. The basic aim of this segregation is to make the images easy to analyze and interpret with preserving the quality. This technique is also used to trace the objects’ borders within the images. This technique labels the pixels according to their intensity and characteristics. Those parts represent the entire original image and acquire its characteristics such as intensity and similarity. The image segmentation technique is used to create contours of the body for clinical purposes.

Segmentation methods have the ability to detect or identify the abnormal portion of the image which is useful for analyzing the size, volume, location, texture, and shape of the extracted image. MR image segmentation with the aid of preserving the threshold information, which is convenient to identify the broken regions extra precisely. It was a trendy surmise that the objects that are placed in close propinquity might be sharing similar houses and characteristics.

5. TUMOR DETECTION.

In the final step, the tumor is retrieved from the marked MRI image area. The marked area occupies the maximum space in the picture, so there was exactly one label in the picture which is designated as having a maximum frequency value. The Morphological gradient is an edge-strength extraction operator that gives symmetric edges between foreground and background regions. The resulting image is then thresholded to obtain a binary edge image.

2.3 SYSTEM DETAILS

2.3.1 SOFTWARE DETAILS

- Windows: Python 3.6.2 or above, PIP and NumPy 1.13.1

Python:

The general-purpose, interactive, object-oriented, and high-level programming language Python is particularly well-liked. It is a garbage-collected and dynamically typed programming language. Between 1985 and 1990, Guido van Rossum designed it. Python source code is also accessible under the GNU General Public License, just like Perl (GPL).

Programming languages such as procedural, object-oriented, and functional are supported by Python. Python's design philosophy places a strong emphasis on code readability

- Python is straightforward and really simple to learn. Python is adaptable and can be used to construct a wide variety of things.
- Python includes strong development libraries, such as those for AI and ML.
- Python is in high demand and pays well.

PIP:

Pip is a widely used package manager for Python that is used to instal and maintain packages. There is a selection of built-in functions and built-in packages included with the Python standard library. The Python standard library does not include data science libraries like scikit-learn and statsmodel. They can be set up using the command line and pip, the default package management for Python.

NumPy:

The Python package NumPy is used to manipulate arrays. Additionally, it provides functions for working with matrices, the Fourier transform, and the linear algebra domain. In the year 2005, Travis Oliphant developed NumPy. You can use it for free because it is an open source project. Numerical Python is referred to as NumPy.

- Lists can function like arrays in Python, but they take a long time to execute.
- NumPy seeks to offer array objects that are up to 50 times faster than Python lists.
- The NumPy array object is referred to as ndarray, and it has a number of accompanying methods that make using ndarray very simple.
- Data science, where efficiency and availability of resources are crucial, commonly uses arrays.

Pandas:

The most widely used Python data analysis library is called Pandas. With back-end source code that is solely written in C or Python, it offers highly optimised performance. Data analysis in Pandas is possible with Data analysis in Pandas is possible with Series and DataFrames.

Anaconda:

With the goal of streamlining package management and deployment, Anaconda is a free and open-source version of the Python and R computer languages for scientific computing. Conda, a package management system, controls package versioning. For Windows, Linux, and macOS, the Anaconda distribution provides data-science packages. The conda package manager and virtual environment manager are included in the anaconda distribution along with 1,500 programmes that were chosen from PyPI.

Jupyter Notebook:

The conda package manager and virtual environment manager are included in the anaconda distribution along with 1,500 programmes that were chosen from PyPI. Additionally, Anaconda Navigator, a graphical user interface, is provided as an alternative to the command line interface (CLI). A Jupyter Notebook document is a JSON file that has a versioned schema, an ordered list of input/output cells, and typically ends with the ". ipynb" suffix. These cells can contain code, text mathematics, graphs, and rich media.

Tensor Flow:

The software library Tensor Flow is free and open-source, and it may be used for differentiable programming and dataflow across a variety of tasks. It is a library for symbolic math, and neural networks and other machine learning techniques use it.

Keras:

An open-source neural network library created in Python is called Keras. It can be used with TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or Plaid ML as a foundation. Its user-friendliness, modularity, and extensibility are its main design goals as it aims to facilitate quick experimentation with deep neural networks. As well as a variety of tools to make working with image and text data easier, Keras includes numerous implementations of widely used neural-network building blocks like layers, objectives, activation functions, and optimizers. This helps to simplify the coding required to create deep neural networks.

2.3.2 HARDWARE CONFIGURATION

- Processor: Intel core i5 or above.
- 64-bit, quad-core, 2.5 GHz minimum per core
- Ram: 4 GB or more
- Hard disk: 10 GB of available space or more.
- Display: Dual XGA (1024 x 768) or higher resolution monitors
- Operating system: Windows.

CHAPTER 3

RESULTS AND DISCUSSIONS

ACCURACY OF CNN:

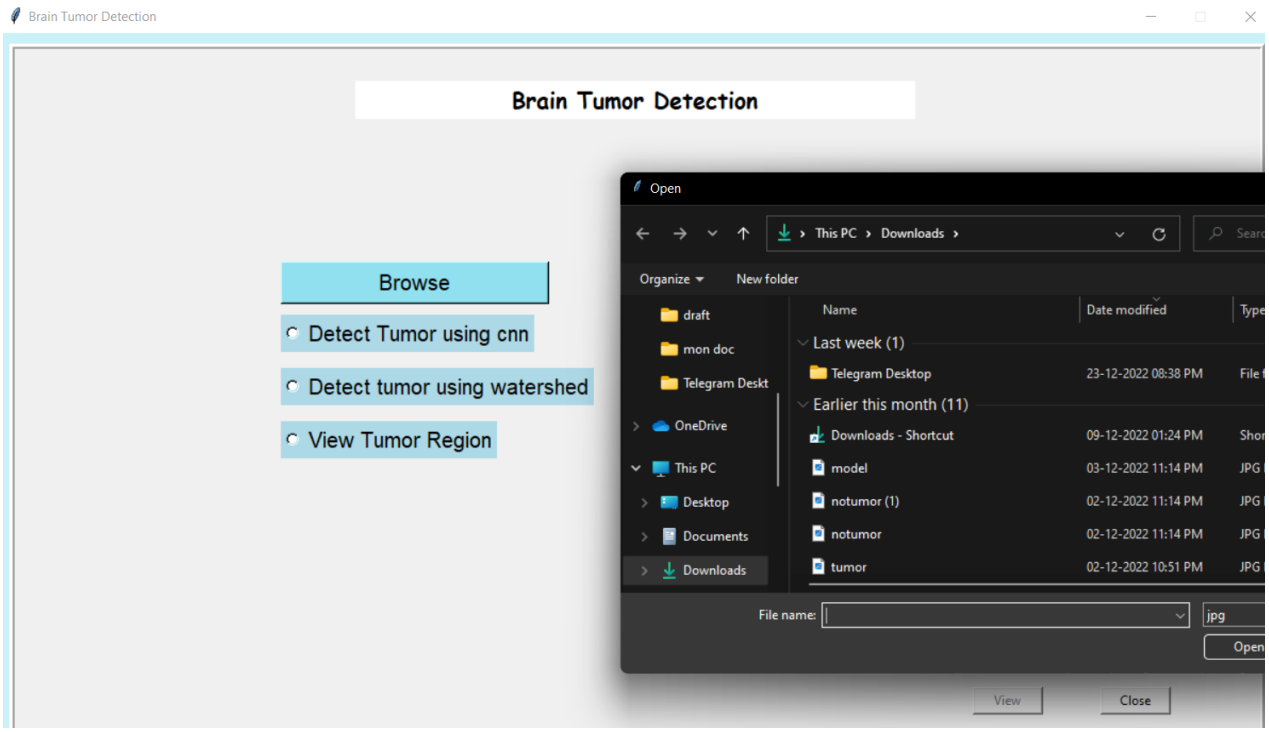
```
#Test our model on the test set
from keras.models import load_model
model = load_model('bestmodel.h5')
acc = model.evaluate(X_test, y_test)[1]
print(f'The accuracy of our model is {acc}')
```

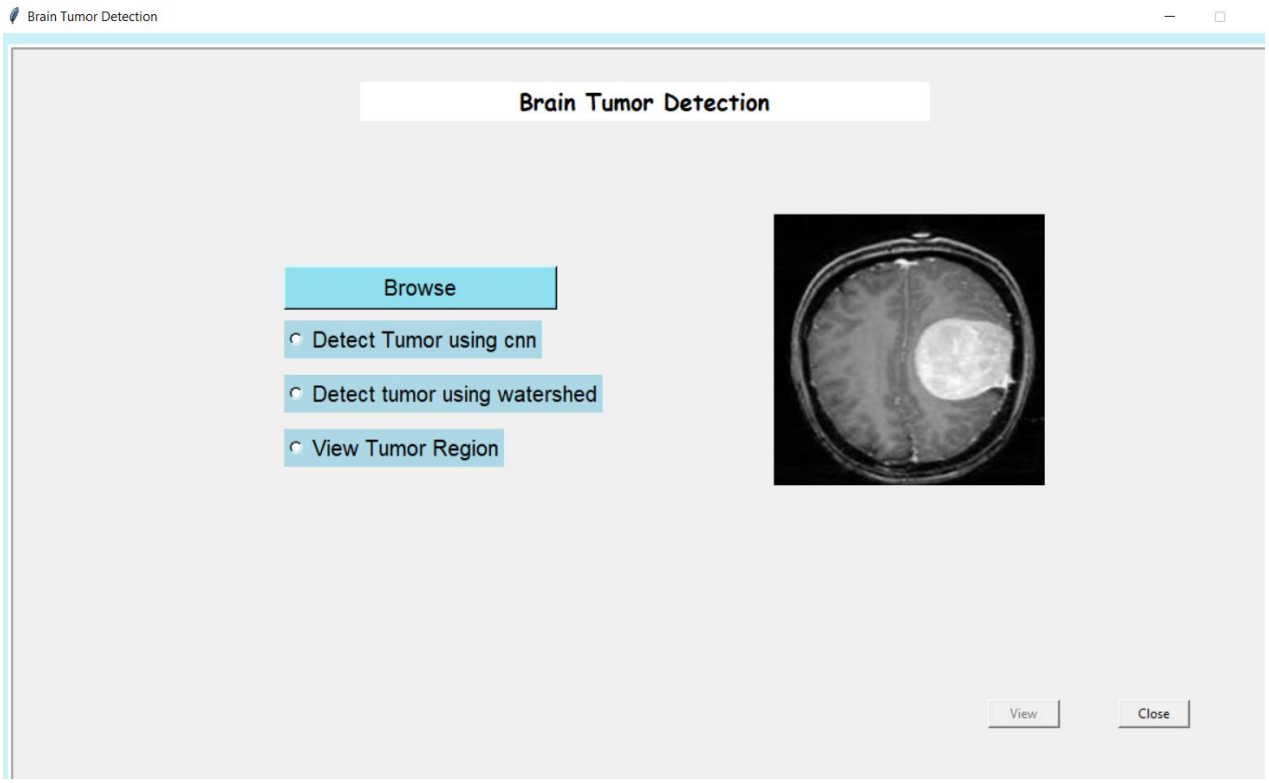
```
25/25 [==============================] - 13s 473ms/step - loss: 0.2134 - accuracy: 0.9524
The accuracy of our model is 0.9524405598640442
```

We can see that our model works well on the test set, so it's not overfitted

a. IMAGE UPLOAD

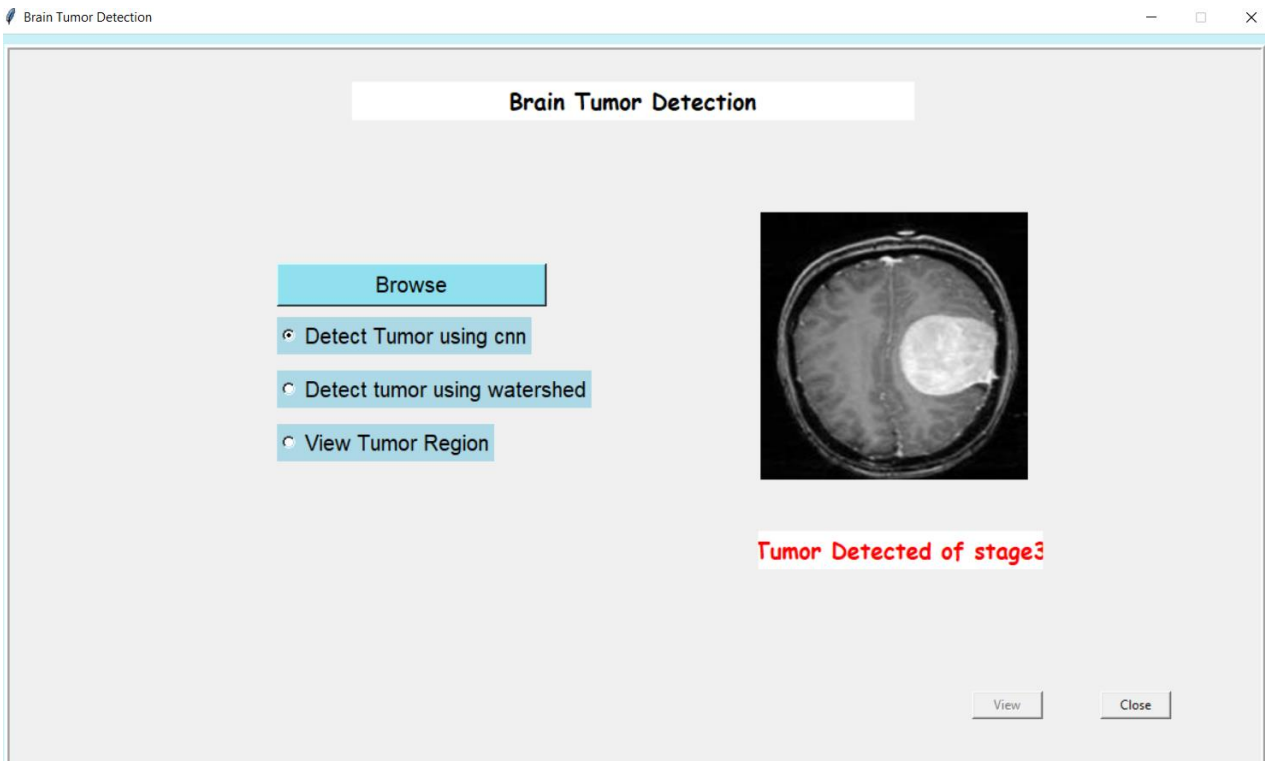
Here user can upload the brain MRI image to find out if there is any tumor.

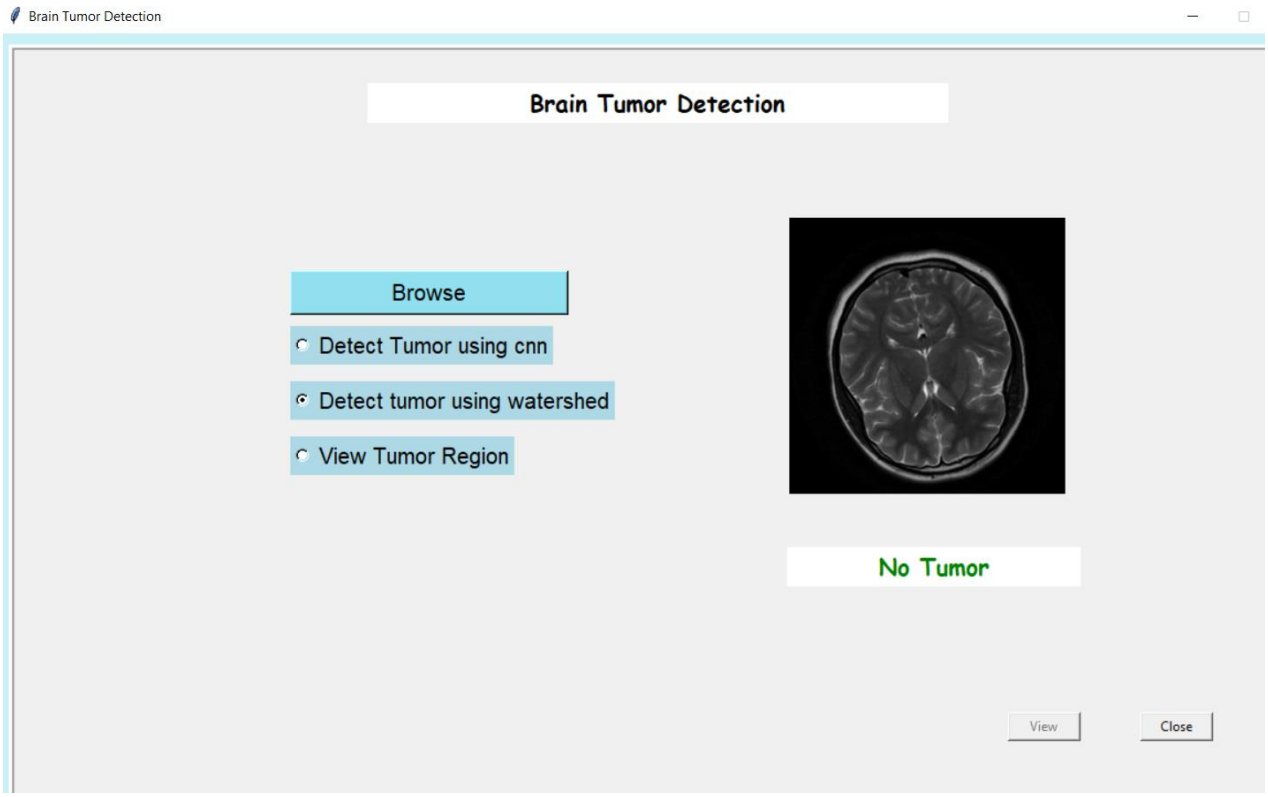




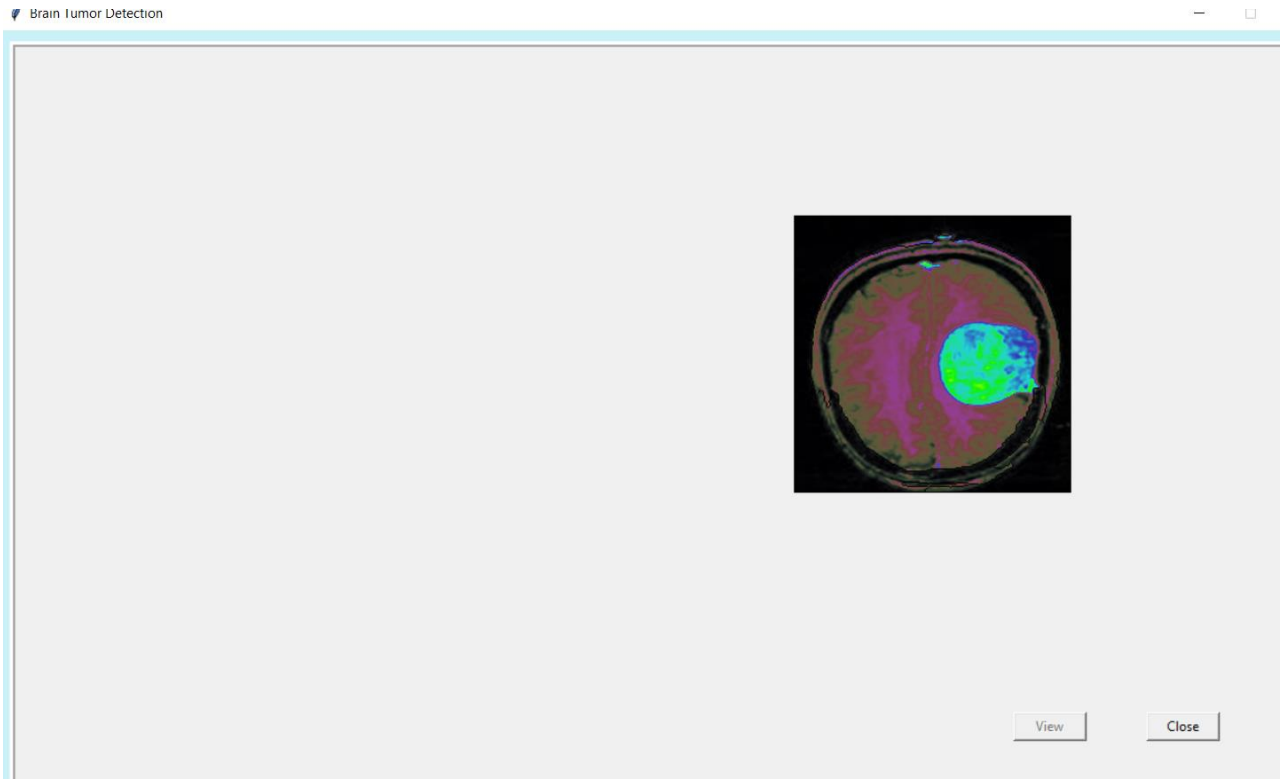
b. IDENTIFY Tumor

The models identify the tumor





c. VIEWING TUMOR



CHAPTER 4

CONCLUSION AND FUTURE WORK

This project proposes an automated, low-cost, and simple-to-use end-to-end User Interface to one of the most difficult challenges in the medical diagnosis domain for doctors and patients: precise, instant, and early diagnosis of brain tumor and knowledge of disease outbreaks, which would aid in quick decision making for tumor detection control measures. With the use of deep Convolutional Neural Networks (CNNs) for tumor categorization, this approach improves on past art. The high-performance deep CNN model offers real-time illness categorization through a Graphical User interface. By dynamically increasing the training dataset with user-added photos for retraining the CNN model, the collaborative model enables continual improvement in tumor classification accuracy. Overall, the results of our project show that the proposal has significant practical deployment potential due to different stages of tumor: the infrastructure is highly scalable, and the underlying algorithm works accurately even with a different stages of tumor categories, performs better with high fidelity real-life training data, improves accuracy with increase in the training dataset, is capable of detecting tumor in early stages, and can succumb to failure.

Experimentation reveals that the suggested method requires a sizable training dataset for more accurate results; in the field of medical image processing, collecting medical data is a laborious task, and in certain rare instances, the datasets could not be available. In each of these scenarios, the suggested algorithm must be trustworthy enough to reliably identify tumour locations from MR images. The suggested method can be further improved by working with weakly trained algorithms that can detect irregularities with little or no training data. Self-learning algorithms would also help to improve algorithm accuracy and speed up processing.

CHAPTER 5

APPENDIX

CODES

PredictTumor.python code

```
from tensorflow.keras.models import load_model
from tensorflow.keras.utils import load_img, img_to_array
import tensorflow as tf
import cv2 as cv
import imutils
import numpy as np
from PIL import Image, ImageOps
import matplotlib.pyplot as plt
import tensorflow_hub as hub

model2 = load_model('brain_tumor_detector.h5')

model1 = load_model('bestmodel.h5')

def predictTumor(image):
     gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
     gray = cv.GaussianBlur(gray, (5, 5), 0)

    # Threshold the image, then perform a series of erosions +
    # dilations to remove any small regions of noise
     thresh = cv.threshold(gray, 45, 255, cv.THRESH_BINARY)[1]
     thresh = cv.erode(thresh, None, iterations=2)
     thresh = cv.dilate(thresh, None, iterations=2)

    # Find contours in thresholded image, then grab the largest one
     cnts = cv.findContours(thresh.copy(), cv.RETR_EXTERNAL,
cv.CHAIN_APPROX_SIMPLE)
     cnts = imutils.grab_contours(cnts)
     c = max(cnts, key=cv.contourArea)

    # Find the extreme points
     extLeft = tuple(c[c[:, :, 0].argmin()][0])
     extRight = tuple(c[c[:, :, 0].argmax()][0])
     extTop = tuple(c[c[:, :, 1].argmin()][0])
     extBot = tuple(c[c[:, :, 1].argmax()][0])

    # crop new image out of the original image using the four extreme points
(left, right, top, bottom)
```

```
     new_image = image[extTop[1]:extBot[1], extLeft[0]:extRight[0]]

     image = cv.resize(new_image, dsize=(240, 240), interpolation=cv.INTER_CUBIC)
     image = image / 255.

     image = image.reshape((1, 240, 240, 3))

     res = model2.predict(image)

     return res



def predict2Tumor(image):
    img_array_yes = img_to_array(image)/255
    img_array_yes = np.expand_dims(img_array_yes, axis=0)
    prediction1 = model1.predict(img_array_yes)
    class1 = np.round(prediction1).astype(int)
    if class1 == 0:
        res=0
    else:
        res=1

    return res
```

ViewTumor.python code

```
import numpy as np
import cv2 as cv


class DisplayTumor:
    curImg = 0
    Img = 0

    def readImage(self, img):
        self.Img = np.array(img)
        self.curImg = np.array(img)
        gray = cv.cvtColor(np.array(img), cv.COLOR_BGR2GRAY)
        self.ret, self.thresh = cv.threshold(gray, 0, 255, cv.THRESH_BINARY_INV +
cv.THRESH_OTSU)

    def getImage(self):
        return self.curImg
```

```
    # noise removal
    def removeNoise(self):
        self.kernel = np.ones((3, 3), np.uint8)
        opening = cv.morphologyEx(self.thresh, cv.MORPH_OPEN, self.kernel,
iterations=2)
        self.curImg = opening

    def displayTumor(self):
        # sure background area
        sure_bg = cv.dilate(self.curImg, self.kernel, iterations=3)

        # Finding sure foreground area
        dist_transform = cv.distanceTransform(self.curImg, cv.DIST_L2, 5)
        ret, sure_fg = cv.threshold(dist_transform, 0.7 * dist_transform.max(),
255, 0)

        # Find unknown region
        sure_fg = np.uint8(sure_fg)
        unknown = cv.subtract(sure_bg, sure_fg)

        # Marker labelling
        ret, markers = cv.connectedComponents(sure_fg)

        # Add one to all labels so that sure background is not 0, but 1
        markers = markers + 1

        # Now mark the region of unknown with zero
        markers[unknown == 255] = 0
        markers = cv.watershed(self.Img, markers)
        self.Img[markers == -1] = [255, 0, 0]

        tumorImage = cv.cvtColor(self.Img, cv.COLOR_HSV2BGR)
        self.curImg = tumorImage
```

GUI.python code

```
import tkinter
from PIL import Image
from tkinter import filedialog
import cv2 as cv
from frames import *
from displayTumor import *
from predictTumor import *
```

```
class Gui:
    MainWindow = 0
    listOfWinFrame = list()
    FirstFrame = object()
    val = 0
    fileName = 0
    DT = object()

    wHeight = 700
    wWidth = 1180

    def __init__(self):
        global MainWindow
        MainWindow = tkinter.Tk()
        MainWindow.geometry('1200x720')
        MainWindow['bg']='#CAF0F8'
        MainWindow.resizable(width=False, height=False)

        self.DT = DisplayTumor()

        self.fileName = tkinter.StringVar()

        self.FirstFrame = Frames(self, MainWindow, self.wWidth, self.wHeight, 0,
0)
        self.FirstFrame.btnView['state'] = 'disable'

        self.listOfWinFrame.append(self.FirstFrame)

        WindowLabel = tkinter.Label(self.FirstFrame.getFrames(), text="Brain
Tumor Detection", height=1, width=40)
        WindowLabel.place(x=320, y=30)
        WindowLabel.configure(background="White", font=("Comic Sans MS", 16,
"bold"))

        self.val = tkinter.IntVar()
        RB1 = tkinter.Radiobutton(self.FirstFrame.getFrames(), text="Detect Tumor
using cnn", variable=self.val,
                                  value=1, background = "light blue",
font=12,activeforeground="green", command=self.check)
        RB1.place(x=250, y=250)

        RB2 = tkinter.Radiobutton(self.FirstFrame.getFrames(), text="Detect tumor
using watershed",
                                  variable=self.val, background = "light blue",
font=12,activeforeground="green",value=3, command=self.check)
        RB2.place(x=250, y=300)
```

```
        RB3 = tkinter.Radiobutton(self.FirstFrame.getFrames(), text="View Tumor
Region",
                                  variable=self.val, value=2, background = "light
blue", font=12,activeforeground="green",command=self.check)
        RB3.place(x=250, y=350)



        browseBtn = tkinter.Button(self.FirstFrame.getFrames(),
fg="black",text="Browse", width=22, background = "#90E0EF",
font=15,activeforeground="green",command=self.browseWindow)
        browseBtn.place(x=250, y=200)

        MainWindow.mainloop()

    def getListOfWinFrame(self):
        return self.listOfWinFrame

    def browseWindow(self):
        global mriImage
        FILEOPENOPTIONS = dict(defaultextension='*.*',
                               filetypes=[('jpg', '*.jpg'), ('png', '*.png'),
('jpeg', '*.jpeg'), ('All Files', '*.*')])
        self.fileName = filedialog.askopenfilename(**FILEOPENOPTIONS)
        image = Image.open(self.fileName)
        imageName = str(self.fileName)
        mriImage = cv.imread(imageName, 1)
        self.listOfWinFrame[0].readImage(image)
        self.listOfWinFrame[0].displayImage()
        self.DT.readImage(image)

    def check(self):
        global mriImage
        #print(mriImage)
        if (self.val.get() == 3):
            self.listOfWinFrame = 0
            self.listOfWinFrame = list()
            self.listOfWinFrame.append(self.FirstFrame)

            self.listOfWinFrame[0].setCallObject(self.DT)

            res = predictTumor(mriImage)
            if res>0.5:
                d="1"
                if res>0.6:
                    d="2"
                if res>0.8:
```

```
                    d="3"
            if res>0.5:
                resLabel = tkinter.Label(self.FirstFrame.getFrames(), text="Tumor
Detected,Stage"+d, height=1, width=20)
                resLabel.configure(background="White", font=("Comic Sans MS", 16,
"bold"), fg="red")
            else:
                resLabel = tkinter.Label(self.FirstFrame.getFrames(), text="No
Tumor", height=1, width=20)
                resLabel.configure(background="White", font=("Comic Sans MS", 16,
"bold"), fg="green")

            resLabel.place(x=700, y=450)

        elif (self.val.get() == 1):
            self.listOfWinFrame = 0
            self.listOfWinFrame = list()
            self.listOfWinFrame.append(self.FirstFrame)

            self.listOfWinFrame[0].setCallObject(self.DT)

            res = predict2Tumor(mriImage)
            res2=predictTumor(mriImage)
            if res2>0.5:
                d="1"
                if res2>0.6:
                    d="2"
                if res2>0.8:
                    d="3"
            if res==1:
                resLabel = tkinter.Label(self.FirstFrame.getFrames(), text="Tumor
Detected,Stage"+d, height=1, width=20)
                resLabel.configure(background="White", font=("Comic Sans MS", 16,
"bold"), fg="red")
            else:
                resLabel = tkinter.Label(self.FirstFrame.getFrames(), text="No
Tumor", height=1, width=20)
                resLabel.configure(background="White", font=("Comic Sans MS", 16,
"bold"), fg="green")

            resLabel.place(x=700, y=450)

        elif (self.val.get() == 2):
            self.listOfWinFrame = 0
            self.listOfWinFrame = list()
            self.listOfWinFrame.append(self.FirstFrame)
```

```
            self.listOfWinFrame[0].setCallObject(self.DT)
            self.listOfWinFrame[0].setMethod(self.DT.removeNoise)
            secFrame = Frames(self, MainWindow, self.wWidth, self.wHeight,
self.DT.displayTumor, self.DT)

            self.listOfWinFrame.append(secFrame)


            for i in range(len(self.listOfWinFrame)):
                if (i != 0):
                    self.listOfWinFrame[i].hide()
            self.listOfWinFrame[0].unhide()

            if (len(self.listOfWinFrame) > 1):
                self.listOfWinFrame[0].btnView['state'] = 'active'

        else:
            print("Not Working")

mainObj = Gui()
```

Frames.py code

```
import tkinter
from PIL import ImageTk
from PIL import Image

class Frames:
    xAxis = 0
    yAxis = 0
    MainWindow = 0
    MainObj = 0
    winFrame = object()
    btnClose = object()
    btnView = object()
    image = object()
    method = object()
    callingObj = object()
    labelImg = 0

    def __init__(self, mainObj, MainWin, wWidth, wHeight, function, Object,
xAxis=10, yAxis=10):
        self.xAxis = xAxis
        self.yAxis = yAxis
        self.MainWindow = MainWin
```

```
        self.MainObj = mainObj
        self.MainWindow.title("Brain Tumor Detection")
        if (self.callingObj != 0):
            self.callingObj = Object

        if (function != 0):
            self.method = function

        global winFrame
        self.winFrame = tkinter.Frame(self.MainWindow, width=wWidth,
height=wHeight)
        self.winFrame['borderwidth'] = 5
        self.winFrame['relief'] = 'ridge'
        self.winFrame.place(x=xAxis, y=yAxis)

        self.btnClose = tkinter.Button(self.winFrame, text="Close", width=8,
                                       command=lambda:
self.quitProgram(self.MainWindow))
        self.btnClose.place(x=1020, y=600)
        self.btnView = tkinter.Button(self.winFrame, text="View", width=8,
command=lambda: self.NextWindow(self.method))
        self.btnView.place(x=900, y=600)


    def setCallObject(self, obj):
        self.callingObj = obj


    def setMethod(self, function):
        self.method = function


    def quitProgram(self, window):
        global MainWindow
        self.MainWindow.destroy()


    def getFrames(self):
        global winFrame
        return self.winFrame


    def unhide(self):
        self.winFrame.place(x=self.xAxis, y=self.yAxis)


    def hide(self):
        self.winFrame.place_forget()
```

```
    def NextWindow(self, methodToExecute):
        listWF = list(self.MainObj.listOfWinFrame)

        if (self.method == 0 or self.callingObj == 0):
            print("Calling Method or the Object from which Method is called is
0")
            return

        if (self.method != 1):
            methodToExecute()
        if (self.callingObj == self.MainObj.DT):
            img = self.MainObj.DT.getImage()
        else:
            print("Error: No specified object for getImage() function")

        jpgImg = Image.fromarray(img)
        current = 0

        for i in range(len(listWF)):
            listWF[i].hide()
            if (listWF[i] == self):
                current = i

        if (current == len(listWF) - 1):
            listWF[current].unhide()
            listWF[current].readImage(jpgImg)
            listWF[current].displayImage()
            self.btnView['state'] = 'disable'
        else:
            listWF[current + 1].unhide()
            listWF[current + 1].readImage(jpgImg)
            listWF[current + 1].displayImage()

        print("Step " + str(current) + " Extraction complete!")


    def removeComponent(self):
        self.btnClose.destroy()
        self.btnView.destroy()


    def readImage(self, img):
        self.image = img


    def displayImage(self):
```

```
        imgTk = self.image.resize((250, 250), Image.ANTIALIAS)
        imgTk = ImageTk.PhotoImage(image=imgTk)
        self.image = imgTk
        self.labelImg = tkinter.Label(self.winFrame, image=self.image)
        self.labelImg.place(x=700, y=150)
```

CNN Classification IPYNB code

```
!pip install opencv-python
!pip install imutils
!pip install keras
!pip install tensorflow
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import matplotlib.pyplot as plt
import os
import math
import shutil
import glob
import cv2
import imutils
import seaborn as sns
from sklearn.utils import shuffle
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Conv2D, MaxPool2D, Dropout, Flatten, Dense, BatchNormalization,
GlobalAveragePooling2D
from keras.models import Model, Sequential
import keras
from tensorflow.keras.utils import load_img
## **Data Loading**
#Count images in the two folders: yes (Tumor) and no (Healthey)
root =  'brain_tumor_dataset'
dict_img = {}
for dir in os.listdir(root):
  dict_img[dir] = len(os.listdir(os.path.join(root, dir)))
dict_img
So we have 155 Brain MRI images with a tumor and 98 healthey ones.
#Plot some MRI images
plt.figure(figsize=(10, 10))
plt.subplot(2, 2, 1)
plt.imshow(load_img(os.path.join('yes', os.listdir('yes')[0])))
plt.title('yes')
plt.subplot(2, 2, 2)
plt.imshow(load_img(os.path.join('yes', os.listdir('yes')[1])))
plt.title('yes')
```

```
plt.subplot(2, 2, 3)
plt.imshow(load_img(os.path.join('no', os.listdir('no')[0])))
plt.title('no')
plt.subplot(2, 2, 4)
plt.imshow(load_img(os.path.join('no', os.listdir('no')[1])))
plt.title('no')
## **Image Augmentation**
#Define a function for image augmentation
def augment_data(file_dir, n_generated_samples, save_to_dir):
    data_gen = ImageDataGenerator(rotation_range=10,
                                  width_shift_range=0.1,
                                  height_shift_range=0.1,
                                  shear_range=0.1,
                                  brightness_range=(0.3, 1.0),
                                  horizontal_flip=True,
                                  vertical_flip=True,
                                  fill_mode='nearest',
                                  rescale= 1/255
                                 )

    for filename in os.listdir(file_dir):
        image = cv2.imread(file_dir + '/' + filename)
        # reshape the image
        image = image.reshape((1,)+image.shape)
        save_prefix = 'aug_' + filename[:-4]
        i=0
        for batch in data_gen.flow(x=image, batch_size=1,
save_to_dir=save_to_dir,save_prefix=save_prefix, save_format='jpg'):
            i += 1
            if i > n_generated_samples:
                break
#Create augmented images folders
#L = ['aug_train', 'aug_test', 'aug_val']
name = 'aug_data'
if not os.path.exists("./"+name):
  os.mkdir("./"+name)
  for dir in os.listdir(root):
    os.makedirs("./"+name+"/"+dir)
else:
  print(f"{name} Already exists")
#Augment data for the examples with the label 'yes' in the training set
augment_data(file_dir='./brain_tumor_dataset/yes',n_generated_samples=8,
save_to_dir='./aug_data/yes')
#Augment data for the examples with the label 'no' in the training set
augment_data(file_dir='./brain_tumor_dataset/no', n_generated_samples=12,
save_to_dir='./aug_data/no')
## **Split data into train, validation and test sets**
#Count images in the two folders: yes (Tumor) and no (Healthey) in the folder of augmented
images
```

```
root =  'aug_data'
dict_img = {}
for dir in os.listdir(root):
  dict_img[dir] = len(os.listdir(os.path.join(root, dir)))
dict_img
#Define a function that creates new folders for the train, test and val sets and append random
pictures to them based on the split percentage
def create_folders(name, perc):
  if not os.path.exists("./"+name):
    os.mkdir("./"+name)
    for dir in os.listdir(root):
      os.makedirs("./"+name+"/"+dir)
      for img in np.random.choice(a=os.listdir(os.path.join(root, dir)),
size=(math.floor(perc*dict_img[dir])), replace=False):
        Src = os.path.join(root, dir, img)
        Dest = os.path.join("./"+name, dir)
        shutil.copy(Src, Dest)
        os.remove(Src)
  else:
    print(f"{name} Already exists")
#Create the training set
create_folders('train', 0.7)
#Create the test set
create_folders('test', 0.15)
#Create the validation set
create_folders('val', 0.15)
#Define a function that counts images in the folders: yes (Tumor) and no (Healthey)
def count_img(folder):
  dict_img = {}
  for dir in os.listdir(folder):
    dict_img[dir] = len(os.listdir(os.path.join(folder, dir)))
  return dict_img
#Count images in the training set
count_img('train')
#Count images in the test set
count_img('test')
#Count images in the validation set
count_img('val')
# #Define a function for image augmentation
# def augment_data2(file_dir, n_generated_samples, save_to_dir):
#     data_gen = ImageDataGenerator(rescale = 1/255)

#     for filename in os.listdir(file_dir):
#         image = cv2.imread(file_dir + '/' + filename)
#         # reshape the image
#         image = image.reshape((1,)+image.shape)
#         save_prefix = 'aug_' + filename[:-4]
#         i=0
```

```
#         for batch in data_gen.flow(x=image, batch_size=1,
save_to_dir=save_to_dir,save_prefix=save_prefix, save_format='jpg'):
#             i += 1
#             if i > n_generated_samples:
#                 break
# #Augment data for the examples with the label 'yes' in the test set
# augment_data2(file_dir='./test/yes',n_generated_samples=8, save_to_dir='./aug_test/yes')
# #Augment data for the examples with the label 'no' in the training set
# augment_data2(file_dir='./test/no', n_generated_samples=12, save_to_dir='./aug_test/no')
# #Augment data for the examples with the label 'yes' in the validation set
# augment_data2(file_dir='./val/yes',n_generated_samples=8, save_to_dir='./aug_val/yes')
# #Augment data for the examples with the label 'no' in the training set
# augment_data2(file_dir='./val/no', n_generated_samples=12, save_to_dir='./aug_val/no')
# #Count images in the training set
# count_img('aug_train')
# #Count images in the training set
# count_img('aug_test')
# #Count images in the training set
# count_img('aug_val')
## **Image Preprocessing**
#Define a function that crop tha brain contour
def crop_brain_contour(image, plot=False):

    #Convert the image to grayscale, and blur it slightly
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    filtered = cv2.bilateralFilter(gray, 5,10,2.5)
    gray = cv2.GaussianBlur(filtered, (5, 5), 0)

    thresh = cv2.threshold(gray, 45, 255, cv2.THRESH_BINARY)[1]
    thresh = cv2.erode(thresh, None, iterations=2)
    thresh = cv2.dilate(thresh, None, iterations=2)

    #Find contours in thresholded image, then grab the largest one
    cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    cnts = imutils.grab_contours(cnts)
    c = max(cnts, key=cv2.contourArea)

    #Extreme points
    extLeft = tuple(c[c[:, :, 0].argmin()][0])
    extRight = tuple(c[c[:, :, 0].argmax()][0])
    extTop = tuple(c[c[:, :, 1].argmin()][0])
    extBot = tuple(c[c[:, :, 1].argmax()][0])

    #Crop new image out of the original image using the four extreme points (left, right, top,
bottom)
    new_image = image[extTop[1]:extBot[1], extLeft[0]:extRight[0]]

    if plot:
```

```
        plt.figure()
        plt.subplot(1, 2, 1)
        plt.imshow(image)
        plt.tick_params(axis='both', which='both', top=False, bottom=False, left=False,
right=False,labelbottom=False, labeltop=False, labelleft=False, labelright=False)
        plt.title('Original Image')
        plt.subplot(1, 2, 2)
        plt.imshow(new_image)
        plt.tick_params(axis='both', which='both',top=False, bottom=False, left=False,
right=False,labelbottom=False, labeltop=False, labelleft=False, labelright=False)
        plt.title('Cropped Image')
        plt.show()

    return new_image
#Example
ex_img = cv2.imread('no/1 no.jpeg')
img = crop_brain_contour(ex_img, plot=True)
#Define a function that load data
def load_data(dir_list, image_size):

    # load all images in a directory
    X = []
    y = []
    image_width, image_height = image_size

    for directory in dir_list:
        for filename in os.listdir(directory):
            image = cv2.imread(directory+'/'+filename)
            image = crop_brain_contour(image, plot=False)
            image = cv2.resize(image, dsize=(image_width, image_height),
interpolation=cv2.INTER_CUBIC)
            # normalize values
            image = image / 255.
            # convert image to numpy array and append it to X
            X.append(image)
            # append a value of 1 to the target array if the image
            # is in the folder named 'yes', otherwise append 0.
            if directory[-3:] == 'yes':
                y.append([1])
            else:
                y.append([0])

    X = np.array(X)
    y = np.array(y)

    # Shuffle the data
    X, y = shuffle(X, y)

    print(f'Number of examples is: {len(X)}')
```

```
    print(f'X shape is: {X.shape}')
    print(f'y shape is: {y.shape}')

    return X, y
#Load trainig data
IMG_WIDTH, IMG_HEIGHT = (240, 240)
X_train, y_train = load_data(['train/yes', 'train/no'], (IMG_WIDTH, IMG_HEIGHT))
#Define a function that plot images
def plot_sample_images(X, y, n=40):
    for label in [0,1]:
        # grab the first n images with the corresponding y values equal to label
        images = X[np.argwhere(y == label)]
        n_images = images[:n]

        columns_n = 10
        rows_n = int(n/ columns_n)

        plt.figure(figsize=(10, 8))

        i = 1 # current plot
        for image in n_images:
            plt.subplot(rows_n, columns_n, i)
            plt.imshow(image[0])

            # remove ticks
            plt.tick_params(axis='both', which='both',
                            top=False, bottom=False, left=False, right=False,
                            labelbottom=False, labeltop=False, labelleft=False, labelright=False)

            i += 1

        label_to_str = lambda label: "Yes" if label == 1 else "No"
        plt.suptitle(f"Brain Tumor: {label_to_str(label)}")
        plt.show()
#Plot samples from the training set
plot_sample_images(X_train, y_train)
```

We can notice that the noise has been deleted and some images look alike with some slight changes resulted from the data augmentation

```
#Load test data
IMG_WIDTH, IMG_HEIGHT = (240, 240)
X_test, y_test = load_data(['test/yes', 'test/no'], (IMG_WIDTH, IMG_HEIGHT))
#Load validation data
IMG_WIDTH, IMG_HEIGHT = (240, 240)
X_val, y_val = load_data(['val/yes', 'val/no'], (IMG_WIDTH, IMG_HEIGHT))
#Plot samples from the validation set
plot_sample_images(X_val, y_val)
## **CNN Model**
#Build our model
model = Sequential()
```

```
model.add(Conv2D(filters = 16, kernel_size = (3,3), activation = 'relu', input_shape = (240, 240, 3)))

model.add(Conv2D(filters = 32, kernel_size = (3,3), activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2)))

model.add(Conv2D(filters = 64, kernel_size = (3,3), activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2)))

model.add(Conv2D(filters = 128, kernel_size = (3,3), activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2)))

model.add(Dropout(rate = 0.25))

model.add(Flatten())
model.add(Dense(units = 64, activation = 'relu'))
model.add(Dropout(rate = 0.25))
model.add(Dense(units = 1, activation = 'sigmoid'))

model.summary()
!pip install pydot

!pip install pydotplus
!pip install graphviz
from tensorflow.keras.utils import plot_model
tf.keras.utils.plot_model(
model,
to_file="model.png",
show_shapes=True,
show_dtype=False,
show_layer_names=True,
rankdir="TB",
expand_nested=True,
dpi=96,
layer_range=None,
show_layer_activations=True,
)
#Compile our model
model.compile(optimizer = 'adam', loss = keras.losses.binary_crossentropy, metrics = ['accuracy'])
#Early stopping and model checkpoint
from keras.callbacks import ModelCheckpoint, EarlyStopping

es = EarlyStopping(monitor = 'val_accuracy', min_delta = 0.01, patience = 5, verbose = 1, mode = 'auto')
mc = ModelCheckpoint(monitor ='val_accuracy', filepath = './bestmodel.h5', verbose = 1, save_best_only = True, mode = 'auto')
```

```
cd = [es, mc]
#Train our model
hist = model.fit(x = X_tr

                ain, y = y_train, batch_size = 32, epochs = 30, validation_data = (X_val, y_val),
callbacks = cd, verbose = 1)
#Plot the graphical interpretation
h = hist.history
plt.plot(h['accuracy'], label = 'accuracy')
plt.plot(h['val_accuracy'], label = 'val-accuracy')
plt.title('Accuracy vs Val Accuracy')
plt.legend()
plt.show()
#Plot the graphical interpretation
h = hist.history
plt.plot(h['loss'], label = 'loss')
plt.plot(h['val_loss'], label = 'val-loss')
plt.title('Loss vs Val Loss')
plt.legend()
plt.show()
#Test our model on the test set
from keras.models import load_model
model = load_model('bestmodel.h5')
acc = model.evaluate(X_test, y_test)[1]
print(f'The accuracy of our model is {acc}')
```

CHAPTER 6

REFERENCES

[1] 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.

[2] Asra Aslam, Ekram Khan, M.M. Sufyan Beg, Improved Edge Detection Algorithm for Brain Tumor Segmentation, Procedia Computer Science, Volume 58,2015, Pp 430-437, ISSN 1877-0509.

[3] B.Sathya and R.Manavalan, Image Segmentation by Clustering Methods: Performance Analysis, International Journal of Computer Applications (0975 – 8887) Volume 29– No.11, September 2011.

[4] Devkota, B. & Alsadoon, Abeer & Prasad, P.W.C. & Singh, A.K. & Elchouemi, A.. (2018). Image Segmentation for Early Stage Brain Tumor Detection using Mathematical Morphological Reconstruction. Procedia Computer Science. 125. 115- 123. 10.1016/j.procs.2017.12.017.

[5] K. Sudharani, T. C. Sarma and K. Satya Rasad, "Intelligent Brain Tumor lesion classification and identification from MRI images using k-NN technique," 2015 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), Kumaracoil, 2015, pp. 777-780. DOI: 10.1109/ICCICCT.2015.7475384

[6] Kaur, Jaskirat & Agrawal, Sunil & Renu, Vig. (2012). A Comparative Analysis of Thresholding and Edge Detection Segmentation Techniques. International Journal of Computer Applications.vol. 39.pp. 29-34. 10.5120/4898-7432.

[7] Li, Shutao, JT-Y. Kwok, IW-H. Tsang and Yaonan Wang. "Fusing images with different focuses using support vector machines." IEEE Transactions on neural networks 15, no. 6 (2004): 1555-1561.

[8] M. Kumar and K. K. Mehta, "A Texture based Tumor detection and automatic Segmentation using Seeded Region Growing Method," International Journal of 49 Computer Technology and Applications, ISSN: 2229-6093, Vol. 2, Issue 4, PP. 855- 859 August 2011.

[9] Mahmoud, Dalia & Mohamed, Eltaher. (2012). Brain Tumor Detection Using Artificial Neural Networks. Journal of Science and Technology. 13. 31-39.

[10] Marroquin J.L., Vemuri B.C., Botello S., Calderon F. (2002) An Accurate and Efficient Bayesian Method for Automatic Segmentation of Brain MRI. In: Heyden A., Sparr G., Nielsen M., Johansen P. (eds) Computer Vision — ECCV 2002. ECCV 2002. Lecture Notes in Computer Science, vol 2353. Springer, Berlin, Heidelberg.

[11] Minz, Astina, and Chandrakant Mahobiya. “MR Image Classification Using Adaboost for Brain Tumor Type.” 2017 IEEE 7th International Advance Computing Conference (IACC) (2017): 701-705.

[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, No 2, Apr-May 2013.

[13] S. Li, J.T. Kwok, I.W Tsang, and Y. Wang, ―Fusing Images with Different Focuses using Support Vector Machines, Proceedings of the IEEE transaction on Neural Networks, China, November 2007.

[14] H. Yu and J.L. Fan, ―Three-level Image Segmentation Based on Maximum Fuzzy Partition Entropy of 2-D Histogram and Quantum Genetic Algorithm, Advanced Intelligent Computing Theories, and Applications. With Aspects of Artificial Intelligence. Lecture Notes in Computer Science, Berlin, Heidelberg 2008.