BRAIN TUMOR IDENTIFICATION USING IMAGE PROCESSING TECHNIQUES

A thesis presented in partial fulfilment of the
requirements for the degree of

Bachelor of Science Honors
in
Information Technology and Management

at IT Unit 2, Faculty of Science, University of Colombo
Sri Lanka.

M.H.D. Maduraarachchi

2022 April

Declaration

I declare that this research thesis is my work and has not been submitted in any form for another degree or diploma at any university or other institution of tertiary education. The work is almost entirely my work; the collaborative contributions have been clearly acknowledged. Due references have been provided on all supporting literature and resources.

……………………………   …………………….

M.H.D. Maduraarachchi   Date

2017s16550

s14089

Supervised by:

Name of the Supervisor	Signature of the Supervisor
Dr U.P. Liyanage	

……………………………….

Date

Abstract

A brain tumor is a mass or proliferation of uncontrollable abnormal cells in the brain tissues. It is one of the most harmful and hazardous causes of cancer death in both men and women worldwide. Tumors do not all turn into malignancies; nevertheless, some do. Doctors use MRI or CT scan images to identify brain malignancies manually. As a result, analysis reports differ depending on the doctors' expertise, experience, and perspective. For these reasons, standardizing and automating the operation would be a superior option. This paper proposes a methodology that can identify brain tumors automatically using image processing techniques. The proposed solution consists of three basic steps. The first phase is the preprocessing stage, which will improve the image's quality by eliminating noise and increasing the contrast. Next, we propose creating a neural network to detect brain tumors and determine whether a patient has one. We also have a parallel approach, a model that uses image processing techniques to detect the tumor location. The proposed methodology will be described in depth in this study.

Acknowledgement

I would like to acknowledge and extend my heartfelt gratitude to the following people who have helped and guided me in numerous ways to make the project a success.

Firstly, my supervisor Dr U.P. Liyanage, for the immense assistance and guidance that have given and motivated me towards achieving the project's aim.

For the immense assistance and guidance, Dr Paduma Samarawickrama, course coordinator of Bachelor of Science Honors in Information Technology and Management, Faculty of Science, the University of Colombo.

All the academic staff at the IT Unit 2 and the Department of Mathematics, Faculty of Science, University of Colombo, have given assistance and coordination.

All the academic and non-academic staff of the Faculty of Science, the University of Colombo who has given all the knowledge and guidance throughout the undergraduate period, members of my family, and the batch mates who have supported in numerous ways.

Table of Contents

List of Figures

Chapter 01 – Introduction

1.1 Introduction

A brain tumor is a lump or growth formed by uncontrollable aberrant cells in the brain tissues. It is one of the most harmful and hazardous causes of cancer death in both men and women worldwide. Tumors do not all turn into malignancies; nevertheless, some do. Tumors are divided into two categories. Such as malignant and benign tumors. Cancerous tumors are malignant tumors, and non-cancerous tumors are benign tumors [1]. As a result, they are accurately identifying a tumor as malignant is critical for patients' subsequent treatments.

Headache, seizures, vision problems, vomiting, mental changes, difficulty walking, speak in sensation are some symptoms in brain tumors [2]. Inherited genetic disorders from parents are the most common cause of brain tumors, albeit they may not always be genetically inherited. As a result, clinicians cannot accurately forecast physical behaviors that may lead to the development of a brain tumor—for example, drinking alcohol, smoking, and so on. There are numerous advanced medical tests available to detect brain tumors, and these tests should be used as the first step when a patient presents with an irregular headache to determine if it is a brain tumor or not.

There are three main steps within this proposed solution. The first step is the preprocessing stage which will enhance the quality of the image by reducing noise and increasing the contrast level of the image. Next, a neural network is developed to identify brain tumor and output whether the patient has a tumor or not. And an image processing-based approach to identify the tumor region.

Since obtaining medical-related sensitive data from government hospitals is strict, pre-collected medical datasets can be obtained from Kaggle (www.kaggle.com).

1.2 Problem in Brief

Brain tumor occurrence is increasing rapidly all over the world. The American Cancer Society estimates that brain and spinal cord tumors in the United States for 2021 include adults and children. About 24,530 malignant tumors of the brain or spinal cord (13,840 in males and 10,690 in females) will be diagnosed. About 18,600 people (10,500 males and 8,100 females) will die from brain and spinal cord tumors [3]. The leading cause of brain tumor misdiagnosis is observer error on MRI and CT (Computed Tomography) scan images. The observer error causes decision-making errors, scan and recognition errors, and decreased search satisfaction. Pre-processing is required before making conclusions based on an MRI scan image because it aids in magnifying inaccuracies caused by changes in the position of the relevant tumor. Manual processing of CT scans and MRI scan images take a long time, and because the eyes are less sensitive, it leads to cancers being missed in scanned images.

1.3 Significance

When a brain tumor grows, other organs near it may become pushed, causing most of the symptoms, and the tumor may also expand to other neighboring organs in the brain. As a result, detecting a brain tumor at an early stage is more beneficial to the patient without causing additional health complications. The primary benefit is the prevention of exposure to drugs with harmful side effects and massive surgeries.

With the advancement of technology, CNNs and image processing techniques may now be utilized to detect brain tumors using MR images, which is a significant benefit. Furthermore, with less human engagement, these brain tumor detection devices reliably detect the tumor. As a result, it will assist in the reduction of errors caused by the human analytical process. On the other hand, because it uses a common standard approach to evaluate digital photos, this proposed solution produces an unbiased analysis result. As a result, the goal is to offer a systematic approach that uses new methodologies to improve brain tumor diagnosis in the future, with the goal of reducing brain tumor fatalities.

1.4 Scope and Objectives

Deep learning methods have accounted for the tremendous acceleration of artificial intelligence research in medical image analysis, interpretation, and segmentation, with several potential applications across various medical subdisciplines. However, only a few studies that investigate different application scenarios are used in the medical context to evaluate the actual need and the practical challenges of model deployment. With the advancement of technology, CNN (Convolutional Neural Network) and image processing techniques may now be utilized to detect brain tumors using MR images, which is a significant benefit. Furthermore, these brain tumor detection devices are more accurate and require less human engagement to detect the tumor. Gathering up different inputs, computing some of their weights, forward output to operate functions reply with the expected output are a few steps that follow in the CNN. According to the CNN classification, the important features of MR Image are line, edge, object etc. In addition, CNN can automatically recognize complex features with more accuracy.

The main aim of this research is to implement an application to optimize brain tumor detection using MR images.

Furthermore, the objectives of the research are as follows.

- Study about the brain, the impact of brain tumors.
- Study about the types of brain tumors and the traditional ways of detecting them.
- Learn about preprocessing techniques.
- Study about the features based on MR images
- Implement an optimized solution for brain tumor detection

Chapter 02 – Literature Review

2.1 Introduction

This chapter discusses the related parallel research work being conducted on similar and associated fields to evaluate, establish, compare, and contrast the necessary technologies, methodologies, strategies, and techniques which would be adopted in the research. Most of the systems consist of three main modules, and these modules are widely discussed in this section.

2.2 Brain Tumor Detection using Image Processing Techniques

The research paper published at IEEE International Conference proposes a novel system for Brain tumor detection through (MRI) Magnetic Resonance Imaging. In this proposed technique, it used Optimized Kernel Probabilistic C-Means Algorithm to pre-process MR Images. Then Adaptive DW-MTM Filter is used to enhance the MRI quality. Finally, the enhanced image is segmented by using Regression Neural Network. The segmentation method is used to separate the tumor area from the background. This segmented image is used to diagnose brain tumor in the prior stage [4].

Another study presents an automatic brain tumor identification technique that uses a convolutional neural network (CNN) to train the brain tumor detection model and Python to implement it. As 3D photos can be fed into this system, 3D images will be produced. The conventional brain tumor classification is performed by using Fuzzy C Means (FCM) based segmentation, texture, and shape feature extraction and SVM (Support Vector Machine) and DNN (Deep Neural Network) based classification are carried out [5].

Sourabh Hanwat and Chandra Jayaraman have proposed research work where three different classification algorithms are used for brain tumor classification as benign, malignant, and normal MRI images. The proposed method used Dilate and Bwareafilt approach for skull removal. The median filter is used to remove the noise of the image. Binary threshold with morphological segmentation helped to highlight the tumor in MRI images. The classification is performed with the help of CNN, RF (Random

Forest), and KNN (K-Nearest Neighbor) algorithms. According to this research paper, CNN is achieved maximum accuracy of 98%, with cross-entropy being 0.097 and validation accuracy of 71%. Random Forests achieved 80% accuracy, and K-Nearest Neighbors achieved 74% accuracy, which is lesser than CNN. The analysis of research work results proved that the Convolutional Neural Network image classification method is better compared to other machine learning classification methods [6].

There is another proposed method, which removes noise and sharp edges. Gaussian, median filters are used to remove noise in pre-processing phase. This process enhances the quality of the image. Sharping edges will help to segment MR images clearly. Extract the 18 critical features from the segmented image and train the model using that features during the process phase. The final stage of post-processing of this method is done using Threshold Segmentation, Watershed Segmentation and Morphological Operators [7].

2.2.1 Pre-processing

2.2.1.1 Noise Removal

In every pre-processing image phase, the main and the important part is to remove the unwanted noise of an image and enhance it to make sure that it is ready to extract the focused features.

A. Lakshmi M.E et al. proposed a pre-processing method for MRIs using the curvelet transform method, which reduces the noise and smooths the image for further processing. The curvelet transform is better than the wavelet transform perceptually. This technique provides more sharp edges as well as sharp images.

- Curvelet transform

  This is a technique that can enhance the curves of the images. The fundamental way of the method is based on dividing the whole picture into smaller parts of overlapping rectangles and applying the ridgelet transform on each rectangle.

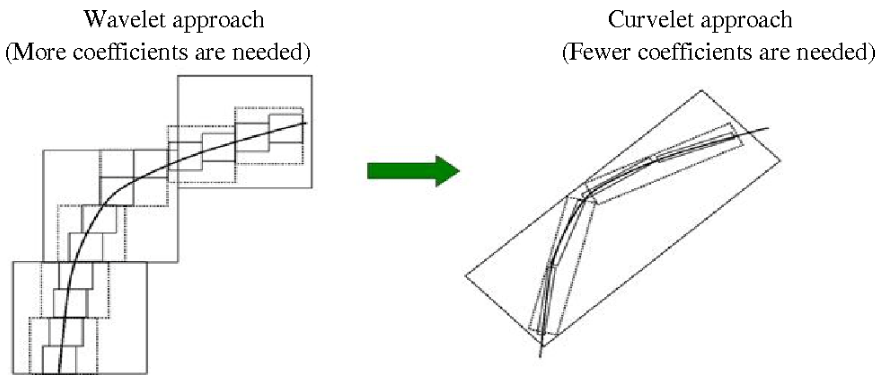

Wavelet approach
(More coefficients are needed)
Curvelet approach
(Fewer coefficients are needed)

Figure 1: Curved edges on wavelet vs curvelet

In the above image, you can see the curved edges of the curvelet is sharper than the wavelet transformation. And the problem of discontinuity of the edges are addressed by the curvelet transform rather than the wavelet transform [8].

2.1.1.2 Artifact Removal

In an MRI, an artifact is something that causes a disturbance but does not appear in the original image. Because of hardware issues, software issues, and the printed details of the patients, artifacts appear in an image.

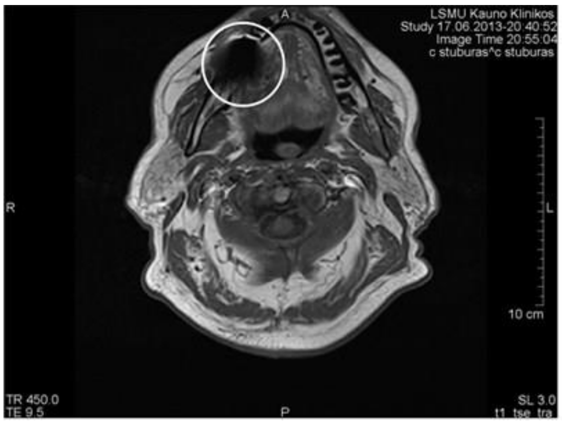


Figure 2: Image with artifacts

Sudipta Roy et al. has proposed a method to remove these artifacts. First, the image will be converted to a binary image. Then the standard deviation of the image will be calculated and decide the threshold value so that the background and the foreground of the image will be separated. So, the total intensity is calculated by,

$$T = \sum_{I} h\,[I]$$

I [m, n] – binary image

h- Intensity of each pixel of gray image

So, the average intensity is calculated by,

$$I_{avg} = \frac{1}{T} \sum_{(m,n)\sum I} I\,[\mathrm{m,n}]$$

And the standard deviation or the threshold value is also calculated.

Next each pixel will be categorized to 0 or 1 based on the calculated threshold value.

Then in the second stage, the different connected components will be identified and arranged based on their areas, 1st, and 2nd. Then the 2nd component will be identified as the artefacts [9].

2.3 Deep Learning Techniques

There are several Deep learning-based systems for automatically detecting brain tumors and their location without the need for human intervention. Deep learning techniques are being used to detect brain tumors and identify regions in the clinical process in this type of research. There are currently just handful of solutions in use as a solution for the issue.

Ali Ari and Davut Hanbay have proposed a method in the Turkish Journal of Electrical Engineering & Computer Sciences 2018 named "Deep learning-based brain tumor classification and detection system". DICOM (Digital Imagine and Communication in Medicine) images are used for this project. This dataset contains both benign and malignant tumor, images which belong to each axial, coronal and sagittal plane.

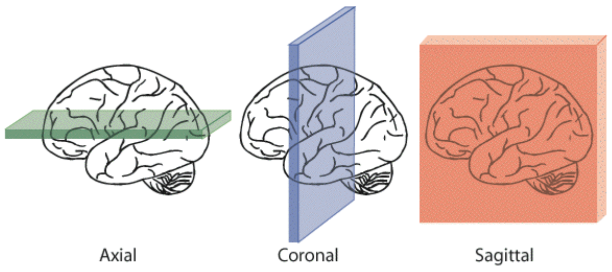


Figure 3: Different view of Brain MRI

There are three main stages: pre-processing stage, image classification stage and extraction of tumor region based on image processing techniques. Brain tumors were classified as benign or malignant using ELM-LRF (Extreme Learning Machines) within the classification stage. Applied convolution and pooling operation to the dataset in the input layer. Convolution filter size r, convolution filter number K, pooling size, and regulation coefficient C. Values were selected for r, K, and pooling size and determine the most suitable value for C with the minimum fault. This CNN was implemented with six layers. The input layer is the first one, the second one is the convolution layer. After the input layer, six convolution filters are used in this layer. The pooling layer is the third layer built after the first convolutional layer. There is another convolution layer with 12 convolution filters as the fourth layer. Again, there is a pooling layer after the second convolution layer as the fifth layer. The fully connected layer is the last. This CNN model used the sigmoid activation function as the activation function. The performance of the proposed ELM-LRF method is compared with the few other approaches. Such as CNN, Statistical features, and Gabor-wavelet features. Compared to the other method, this is quite simple, useful, and needs a concise time to train because of no iteration and randomly generated input weights [10].

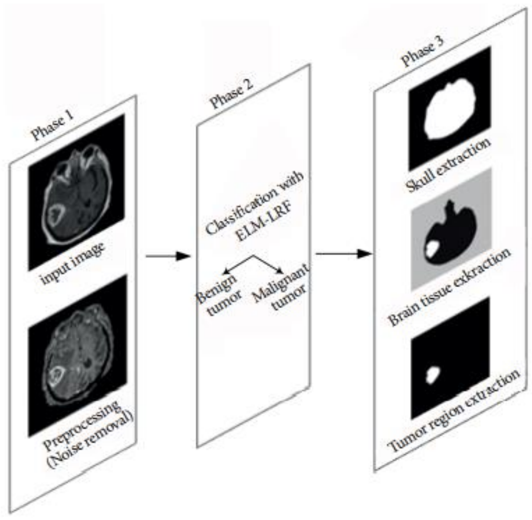


Figure 4: Proposed method by Ali Ari and Davut Hanbay

Another solution, called "Brain Tumor Segmentation Using Convolutional Neural Network with Tensor Flow," was proposed by M. Malathi and P. Sinthia. The dataset for this study includes MRI scans, and the segmentation approaches are tensor flow. The research was conducted in python because of the code's compatibility and clarity and the availability of more graphic packages. The proposed system's segmentation was built using CNN with small 3x3 kernels, which helps generate deep architecture with a limited number of weights in the network. The CNN that was used in this case is made of four layers. These are the input and convolution layers, respectively. The picture patches for the remaining half of the CNN are generated by the input layer. The purpose of picture patching is to save computing time and memory space by calculating linear and non-linear relationships between each voxel of a big 3D input image. It was also beneficial to look for a relationship inside a specific region rather than the entire picture. Multiple convolution layers were used for extracting low-level features like borders and corners. Feature maps are the output of the convolution layer. The activation function for this project was ReLU (Rectified Linear Units). Keras is used to build a training model [11].

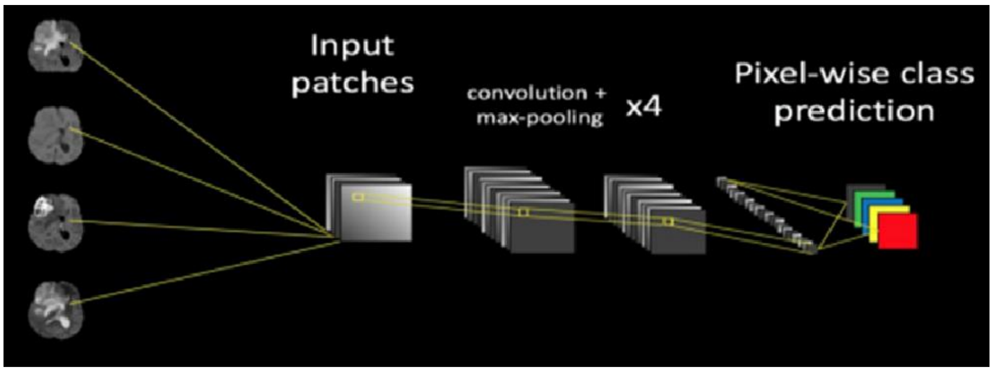


Figure 5: CNN architecture of the model by Malathi and Sinthia

2.4 Image Processing Techniques

Research done by Vipin Y. Borole, Sunil S. Nimbhore and Dr Seema S. Kawthekar has proposed a system that can identify the tumor region by the MR image. First, they have used the median filter for noise removal. It replaces the value of the centre cell with the median value of the intensity values of the neighbourhood cells. The image that removes salt and paper noise, also called impulse noise, is sent to the next phase. In the next phase, this system enhances the input image by improving the contrast of the MR image. Edge detection can be quickly done after this enhancement. This system has used various methods to find the edge. The canny edge detection method has given the most accurate output as they found. This technique finds the boundaries of objects within the image. It detects discontinuities in the brightness of each pixel.

In this system, the thresholding is done manually according to the input image. After thresholding, this system uses morphological operations to maximize the accuracy of the output [12].

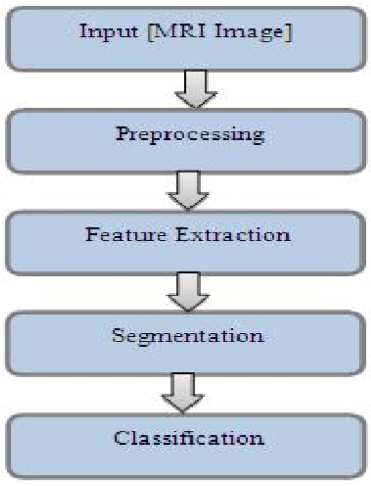


Figure 6: The proposed system by Vipin Y. Borole et al

N. Sravanthi et al. proposed a tumor identification system using Image processing techniques. The proposed method for detecting a brain tumor includes three diagnostic tasks: pre-processing, segmentation, and feature extraction. Later, they calculated and classified the area based on this. As mentioned above, the obtained CT image is pre-processed. The pre-processed image is segmented, and later they extract features from the segmented image [13]

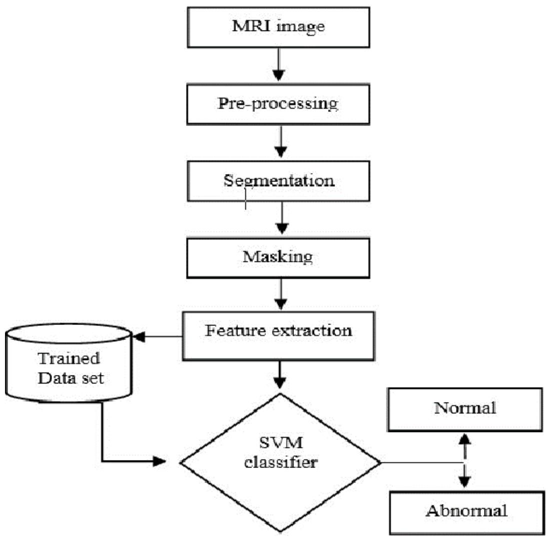


Figure 7: Proposed system by Sravanthi et al

P.D.Yadav and Y.M.Patil of Electronics Engineering, Department KIT (India) proposed a tumor identification system using k-means, fuzzy c means and watershed segmentation. Pre-processing and picture segmentation are the two steps of the proposed approach. A tracking method and a median filter are used for pre-processing. K-means and Fuzzy c methods are used to do classification.

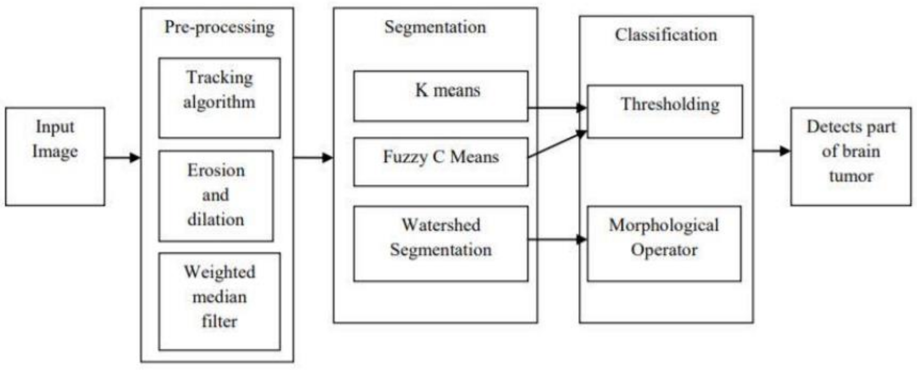


Figure 8: Proposed system by P.D.Yadav and Y.M.Patil

- Fuzzy c means method

  The goal of the Fuzzy c method is to find the cluster's centers, also called centroid to minimize the dissimilarity functions. As the beginning step, the algorithm selects the initial cluster centroid. After several iterations of the algorithm, the final output converges to the actual centroid. Therefore, a high-quality set of initial clusters should be achieved, and it is very important on the FCM algorithm. If the right set of initial cluster centroids is chosen, the algorithm does not need many iterations to give the optimal result. The Fuzzy C-Means algorithm is an algorithm that iterates multiple times to find clusters in data that uses the fuzzy membership concepts.

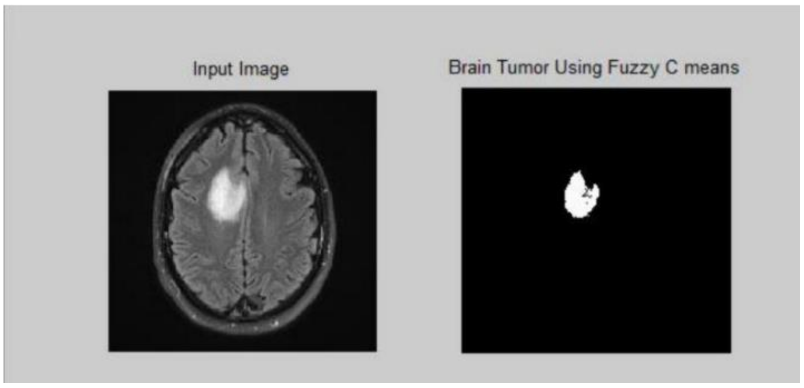


Figure 9: Output of the system by P.D.Yadav and Y.M.Patil

Chapter 03 – Adopted Technologies

3.1 Introduction

This chapter includes a brief description of the technologies used in the system's implementation. It also contains reasons for using such technologies, such as the technology's performance and applicability to the issue domain.

3.2 Programming Languages

Python will be used as the core programming language in implementing the system. Python's flexibility of use allows programmers to create reliable systems. It can input photos (as well as transformations, meshes, and point sets) and view and analyse them using a user-friendly graphical interface. It is well-known for its functionality and adaptability.

3.3 Libraries

3.3.1 OpenCV

This project is mainly focused on image processing techniques. So, the OpenCV library has been used for this. It is an open-source computer vision library that can be used on many different platforms. OpenCV aids the research in performing various processing activities. It has a good performance compared to the other libraries available.

3.3.2 Keras

Keras is a free, open-source Python framework for developing and evaluating deep learning models that are powerful and simple. It covers Theano and TensorFlow, two efficient numerical computation frameworks, and allows to create and train neural network models with just a few lines of code. It has features like activation functions, optimizers, layers, tools, and objectives to write deep neural network codes.

3.3.3 Tensorflow

Tensorflow is an open-source Machine Learning Framework made by Google that facilitates efficient performance in machine learning and complex computational tasks. In Tensorflow, nodes represent mathematical operations, and the edges represent the data arrays (tensors) used to communicate between them.

3.3.3 Scikit-learn

Scikit-learn is a key machine learning toolkit for the Python programming language. Scikit-learn is a set of machine learning tools that include mathematical, statistical, and general-purpose algorithms that serve as the foundation for various machine learning technologies.

3.3.4. Matplotlib

Matplotlib is a Python package that allows you to create static, animated, and interactive visualizations. It is a plotting library for the Python programming language. It provides an object-oriented API for embedding charts into applications utilizing GUI toolkits such as Tkinter, wxPython, Qt, or GTK.

Chapter 04 – Methodology

4.1 Introduction

This chapter discusses the system's design that is proposed for the problem addressed in the research. It contains the proposed system's top-level architecture and the conceptual designs for each system's modules. It explains what each module does and how the modules relate to one another.

4.2 Architecture of the overall system

This is a system that can automatically identify brain tumors using image processing techniques. There are three main steps in the proposed solution. The first step is the pre-processing stage which will enhance the quality of the image, cropping and obtaining the desired part of the brain. Next, convolutional neural network is proposed to develop to identify brain tumor and output whether the patient has a tumor or not. Along with that it is proposed to implement a module to identify the tumor region with image processing techniques.

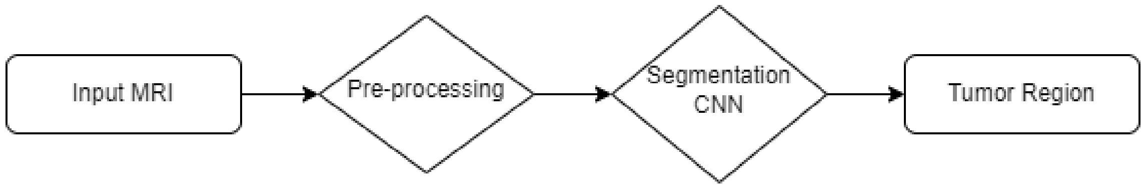


Figure 10: Process of the System

4.2.1 Pre-processing

To increase the accuracy of the CNN and the classification, pre-processing must be done before it. The pre-processing part is required because of mainly two reasons:

1. The presence of artifacts could impact on the result.
2. Enhance the quality of the image.

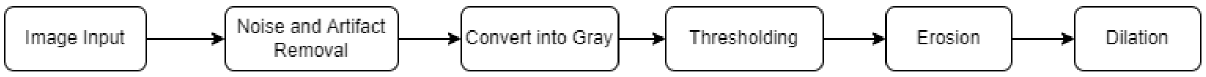


Figure 11: Pre-processing Architecture

4.2.1.2 Dataset Acquisition

Medical image dataset acquisition is much more difficult as those data are very confident. Obtaining a high-quality dataset preferred for image processing and model training is also a challenge.

For this research, the MRI scanned images were acquired from an online available website, www.kaggle.com.

The dataset acquired from kaggle.com contained 253 images with 155 images with "yes" or tumorous Brain MRIs and 98 images with "no" or non-tumorous brain MRIs. This is a clear and labelled dataset that can be directly used for pre-processing.

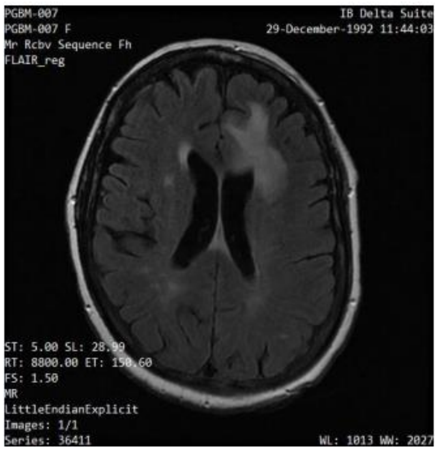


Figure 12: Acquired Image

4.2.1.2 Removing artifacts

Different artifacts, as well as the patient's name, id, and other information, can be found in MRI or CT scan images. Before moving on to the processing stage, these artifacts should be removed. To do this, a threshold mechanism can be used. The primary component, the skull, as well as the entire brain, will be extracted after this operation.

4.2.2 CNN Segmentation Model

Magnetic Resonance Images are fed into the Convolutional Neural network as test data, and those test data includes both tumors and non-tumor MR Images.

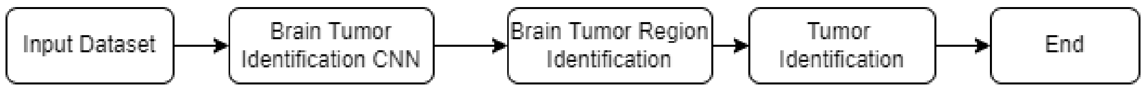

Input Dataset
Brain Tumor Identification CNN
Brain Tumor Region Identification
Tumor Identification
End

Figure 13: Architecture of CNN model

4.2.2.1 Convolutional Neural Network Classifier

The Convolutional Neural Network (CNN) is a deep neural network mainly used in image classification and computer vision applications. A simple neural network consists of an input layer, a hidden layer, and an output layer. Two or more hidden layers can be found in a deep neural network. Convolution layers are followed by a fully connected neural network in a convolutional neural network. According to the CNN classification, the essential features of MR Image are line, edge, object etc. In addition, CNN can automatically recognize complex features with more accuracy.

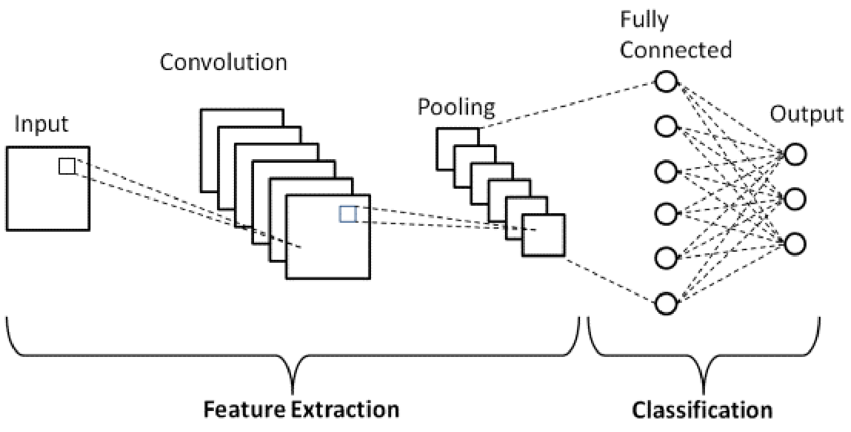


Figure 14: Structure of a Convolutional Neural Network

4.2.2.2 Activation Function

The Rectified Linear Unit (ReLU) activation function is used to overcome the vanishing, gradient problem letting models to learn faster and perform better in the neural network with more layers like CNN. There are two more activation functions: Sigmoid and Tanh Activation Functions. But these activation functions cannot overcome the vanishing gradient problem in the neural network consisting of more layers.

4.2.2.3 Pooling

It combines contiguous nearby features in the feature maps. This integration of possibly redundant features builds the representation more tightly packed and invariant to small image changes, such as insignificant details; it also reduces the computational load of the next stages. Max-pooling or average-pooling is commonly used approach to join features.

4.2.2.4 Regularization

Regularization is a technique that regulates the model complexity. Deep learning neural networks like CNN quickly overfit a training dataset with few outlined examples. This helps to make less risk of overfitting while enhancing the generalization of convolutional neural networks. Dropout will be used as regularization method in fully connected layer. Make large number of different network architectures with use of single model by indiscriminately dropping out the nodes within the training period and all nodes are involved to execute this process. This method called as dropout.

4.2.2.5 Loss Function

This will calculate the gap between predicated value and real value. The calculation is done during model optimization process. Simply, the loss is used to calculate the gradients.

4.2.2.6 Dataset

The dataset for this project consists of only 253 brain MRI images, and the model used to train these images is a Convolutional Neural Network (CNN), which is the best neural network for training image datasets. However, this amount of data is insufficient for those types of models, as the model's accuracy might be low. Hence, the size of the dataset proposed to increase, and there are tumorous images more than non-tumorous images. That means this is not a balanced dataset. Therefore, dataset have to be converted to the balanced dataset while increasing its size, before feed the Neural Network. Data augmentation is to be done to minimize that issue.

4.2.3 Image Processing Model

The goal of the model is to identify the tumor region accurately based on the results obtained by the CNN module. Morphological operations such as Erosion, Dilation and feature extraction technics will be used for this process.

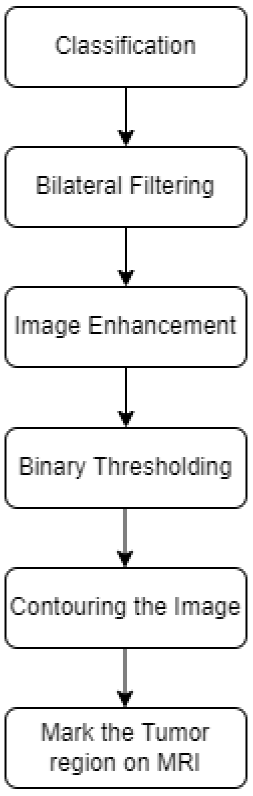


Figure 15: Image Processing Model

4.2.3.1 Smoothing

A bilateral filter is a non-linear image smoothing filter that preserves edges while reducing noise. It uses a weighted average of intensity data from surrounding pixels to replace the intensity of each pixel. A Gaussian distribution can be used to calculate this weight.

4.2.3.2 Enhancement

Image enhancement is the technique of highlighting key aspects of an image while weakening or deleting any extraneous information based on the demands of the user. Eliminating noise, uncovering blurred details, and altering levels to highlight parts of an image are just a few examples.

Point operators which called as pixel transformations and neighborhood operation are also called as area-based operators which are the prominent approaches that can use in image processing. Considering the intensity distribution of an average brain MR image, here used the pixel transformation approach to improve the correctness of brightness and contrast. Here ‘α’ and ‘β’ are the parameters that use with the transformation equation, and they are representing contrast and brightness respectively. Here used the value of alpha as 1.3 (gain factor) and the value of beta as .8 (bias factor).

$$g(i, j) = \alpha \cdot f(i, j) + \beta$$

Pixel Transformation Equation

4.2.3.3 Thresholding

After the enhancement binary thresholding method is used to identify separate components of the MR image. In binary thresholding the pixel values which are below the threshold level will be assigned 0 and the pixel values which are above the threshold will assigned 255. The enhanced image should be converted to the gray scale prior to the binarization to reduce the three RGB channels to a single channel.

A histogram-based algorithm has used here to find the threshold value from each image.

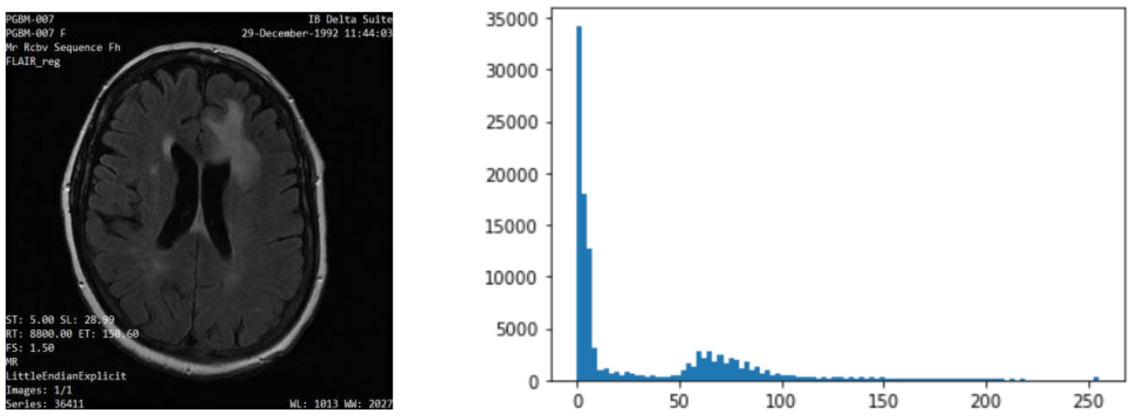


Figure 16: MRI and the relevant Image histogram

4.2.3.4 Morphological Operations

Morphological image processing is a set of non-linear processes that deal with the shape or morphology of image features. These operations are used to determine an object's shape and boundary. Morphological operations depend only on the relative ordering of pixel values, not on their absolute numerical value. There for these methods suit properly with binary images.

The purpose of using morphological operators here is to display the classified brain tumor area to the user. In this section mainly the Erosion and Dilation methods are used.

- Erosion

  Erosion separates objects in a binary image by increasing black pixels and decreasing white pixels according to the structuring element. This technic is used for shrinking process of an image. Objects with connected edges in an image can be separated by using this technic. In this system, we have used the 3x3 kernel to do erosion.

- Dilation

  Dilation works as a grow up process of the image. It increases the white pixels and decrease the black pixels according to the structural element.

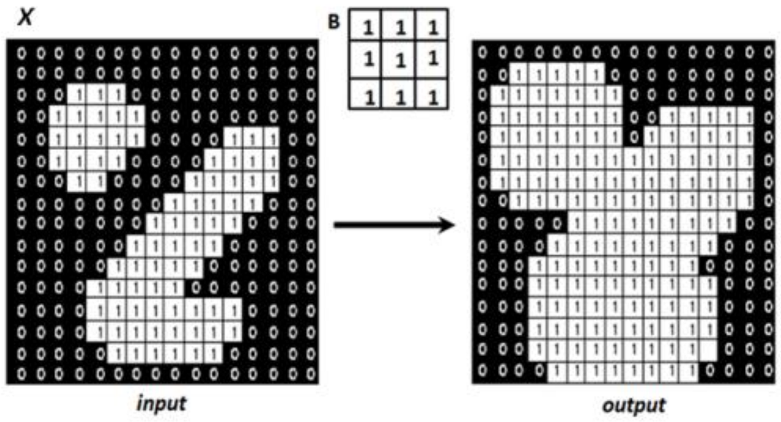


Figure 17: Visual representation of an input image and Dilated image

4.2.3.5 Contour

Contour is a hypothetical curve that connects all the continuous boundary points in each component of the binarized image. This method identifies pixels which have same intensity level. Therefor we can use this method to identify separate components of a binarized image. Contour finding methods mainly works according to the square tracing algorithm.

Chapter 05 – Implementation

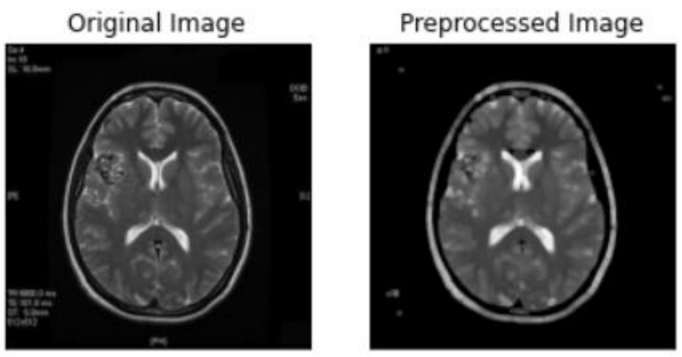
5.1 Introduction

The experiments carried out in implementing the system is illustrated in this chapter. It gives a description about the current status of the implementation process of the system.

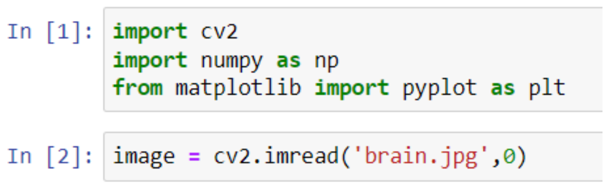
5.2 Preprocessing

5.2.1. Artifact Removal

In order to identify the tumor region of a MR image first the artifacts should be removed. It is important since it improve the accuracy of tumor identification. There can be many approaches to obtain artifact removal. But in this study, Image processing techniques were used.



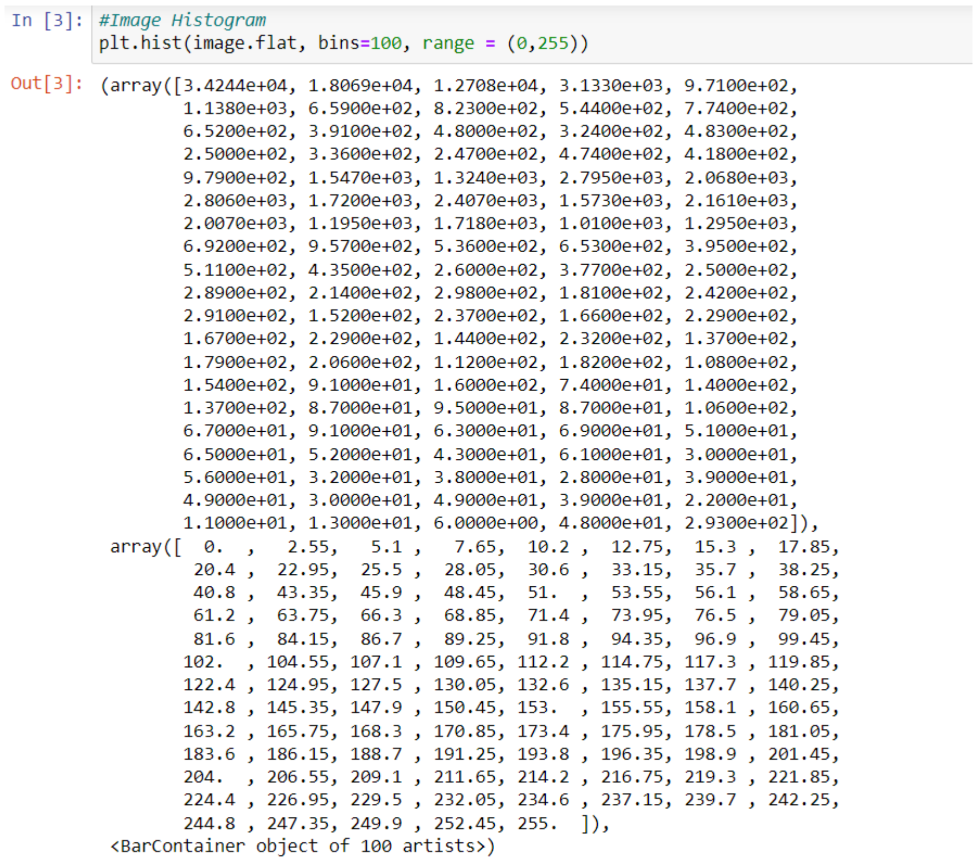
As the implementation, first the required libraries were imported. Then, the targeted image was loaded and checked the color levels and converted to a gray image.
It is easy to handle a gray image than any other color image since it has only 256 color levels. Even the dataset is MRI images, there can be differences in the color levels. Hence it is a good practice to convert all the MRI images to gray color format.



```
In [1]: import cv2
        import numpy as np
        from matplotlib import pyplot as plt

In [2]: image = cv2.imread('brain.jpg',0)
```

Figure 18: Python code 01

Next a histogram was plotted based on the intensities of the gray scale image. Image histogram is one of the main methods to enhance the images.



```
In [3]: #Image Histogram
        plt.hist(image.flat, bins=100, range = (0,255))

Out[3]: (array([3.4244e+04, 1.8069e+04, 1.2708e+04, 3.1330e+03, 9.7100e+02,
                1.1380e+03, 6.5900e+02, 8.2300e+02, 5.4400e+02, 7.7400e+02,
                6.5200e+02, 3.9100e+02, 4.8000e+02, 3.2400e+02, 4.8300e+02,
                2.5000e+02, 3.3600e+02, 2.4700e+02, 4.7400e+02, 4.1800e+02,
                9.7900e+02, 1.5470e+03, 1.3240e+03, 2.7950e+03, 2.0680e+03,
                2.8060e+03, 1.7200e+03, 2.4070e+03, 1.5730e+03, 2.1610e+03,
                2.0070e+03, 1.1950e+03, 1.7180e+03, 1.0100e+03, 1.2950e+03,
                6.9200e+02, 9.5700e+02, 5.3600e+02, 6.5300e+02, 3.9500e+02,
                5.1100e+02, 4.3500e+02, 2.6000e+02, 3.7700e+02, 2.5000e+02,
                2.8900e+02, 2.1400e+02, 2.9800e+02, 1.8100e+02, 2.4200e+02,
                2.9100e+02, 1.5200e+02, 2.3700e+02, 1.6600e+02, 2.2900e+02,
                1.6700e+02, 2.2900e+02, 1.4400e+02, 2.3200e+02, 1.3700e+02,
                1.7900e+02, 2.0600e+02, 1.1200e+02, 1.8200e+02, 1.0800e+02,
                1.5400e+02, 9.1000e+01, 1.6000e+02, 7.4000e+01, 1.4000e+02,
                1.3700e+02, 8.7000e+01, 9.5000e+01, 8.7000e+01, 1.0600e+02,
                6.7000e+01, 9.1000e+01, 6.3000e+01, 6.9000e+01, 5.1000e+01,
                6.5000e+01, 5.2000e+01, 4.3000e+01, 6.1000e+01, 3.0000e+01,
                5.6000e+01, 3.2000e+01, 3.8000e+01, 2.8000e+01, 3.9000e+01,
                4.9000e+01, 3.0000e+01, 4.9000e+01, 3.9000e+01, 2.2000e+01,
                1.1000e+01, 1.3000e+01, 6.0000e+00, 4.8000e+01, 2.9300e+02]),
         array([  0.  ,   2.55,   5.1 ,   7.65,  10.2 ,  12.75,  15.3 ,  17.85,
                 20.4 ,  22.95,  25.5 ,  28.05,  30.6 ,  33.15,  35.7 ,  38.25,
                 40.8 ,  43.35,  45.9 ,  48.45,  51.  ,  53.55,  56.1 ,  58.65,
                 61.2 ,  63.75,  66.3 ,  68.85,  71.4 ,  73.95,  76.5 ,  79.05,
                 81.6 ,  84.15,  86.7 ,  89.25,  91.8 ,  94.35,  96.9 ,  99.45,
                102.  , 104.55, 107.1 , 109.65, 112.2 , 114.75, 117.3 , 119.85,
                122.4 , 124.95, 127.5 , 130.05, 132.6 , 135.15, 137.7 , 140.25,
                142.8 , 145.35, 147.9 , 150.45, 153.  , 155.55, 158.1 , 160.65,
                163.2 , 165.75, 168.3 , 170.85, 173.4 , 175.95, 178.5 , 181.05,
                183.6 , 186.15, 188.7 , 191.25, 193.8 , 196.35, 198.9 , 201.45,
                204.  , 206.55, 209.1 , 211.65, 214.2 , 216.75, 219.3 , 221.85,
                224.4 , 226.95, 229.5 , 232.05, 234.6 , 237.15, 239.7 , 242.25,
                244.8 , 247.35, 249.9 , 252.45, 255.  ]),
         <BarContainer object of 100 artists>)
```

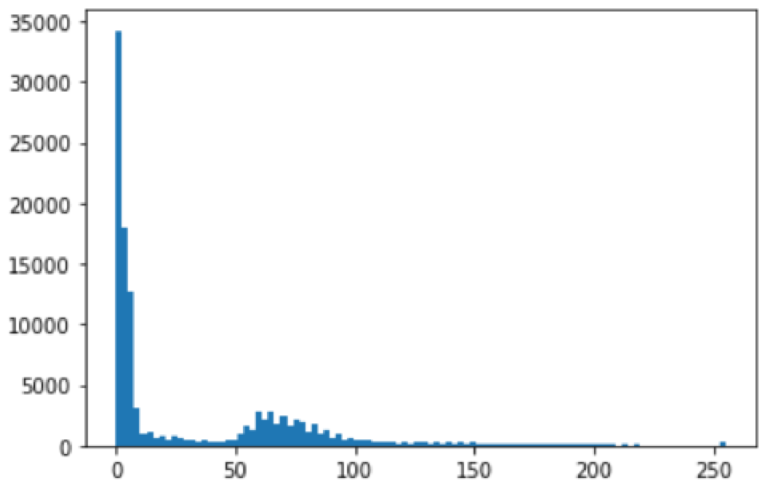


Figure 19: Python code 02

By looking at the histogram, we can roughly decide the nearest threshold value that can be used to make the digital image binary. There are many methods to threshold an image.

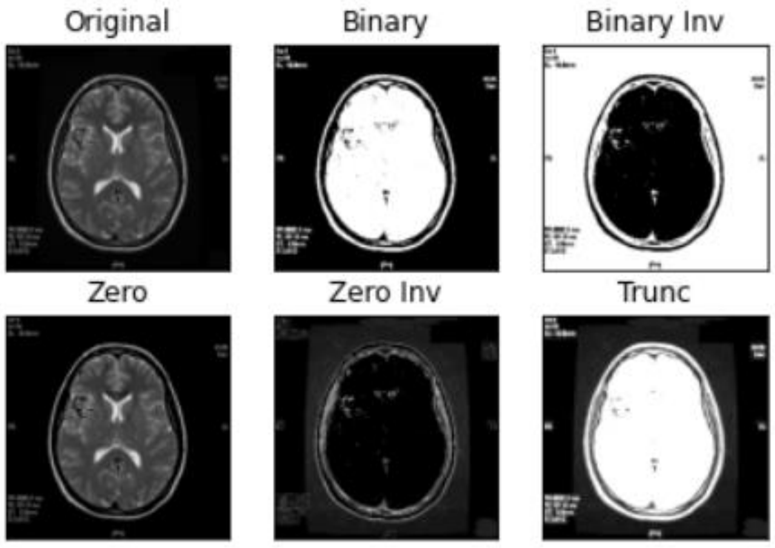


Figure 20: Different thresholding options

An optimum method of thresholding is Otsu's method. Comparing with the above methods, Otsu's method is identified as the best method to threshold the MRIs. So finally, the Otsu's method was used to make the image binary. By this the artifacts have been removed.

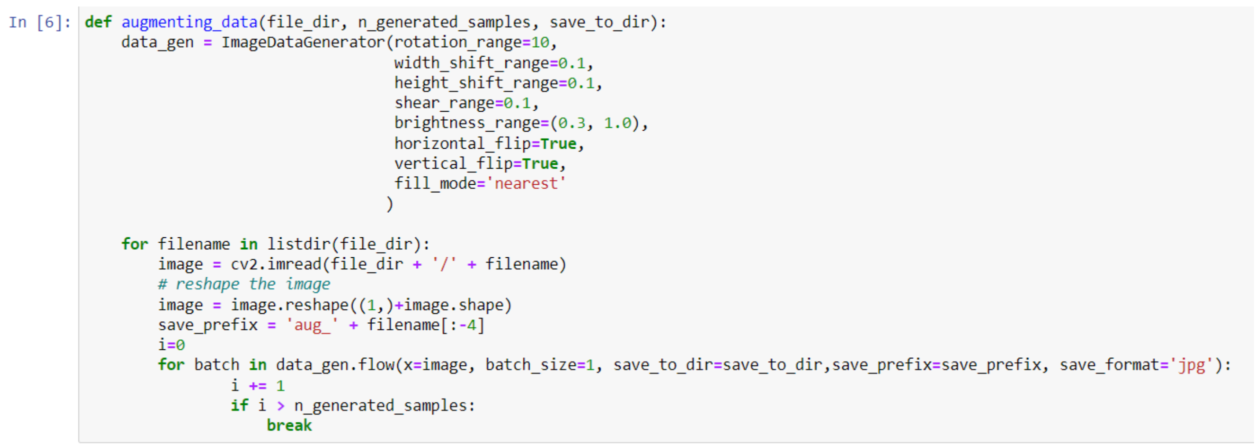
5.3 CNN Model

5.3.1 Data Augmentation

This will help to synthetically make grater in amount of training dataset and decrease the overfitting. Increase the size of training dataset by generating new versions of selected dataset. Large size of training dataset will increase the accuracy of the CNN. Augmentation technique create verity of images and this process improve the ability of the fit models to generalize what they have learned to new images.

The used dataset includes 253 brain MRI images. Those images were separated in to two categories such as "yes" and "no". "yes" folder contains 155 brain MRI images with the tumors and “no” folder Implementation contains 98 brain MRI images without tumor. This dataset has not enough data to train a convolutional neural network. Hence, increment of the dataset is done using data augmentation techniques.

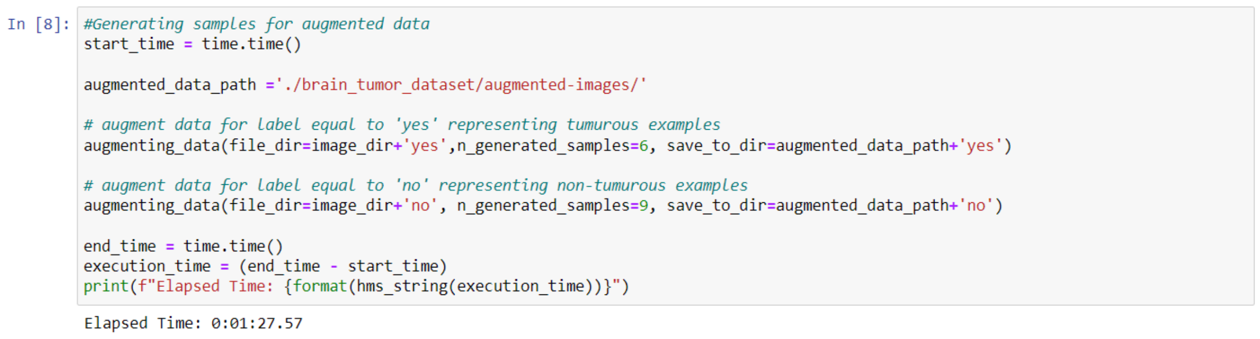
TensorFlow, OpenCV, Matplot are used for the data augmentation.



```
In [6]: def augmenting_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'
                                         )

            for filename in 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
```

Figure 21: Data augmentation function

From the above code segmentation, image loading from original location was done and additional images were generated by reshaping original images using techniques like rotate, shift, and flip. After, the generated images were stored.



```
In [8]: #Generating samples for augmented data
        start_time = time.time()

        augmented_data_path ='./brain_tumor_dataset/augmented-images/'

        # augment data for label equal to 'yes' representing tumurous examples
        augmenting_data(file_dir=image_dir+'yes',n_generated_samples=6, save_to_dir=augmented_data_path+'yes')

        # augment data for label equal to 'no' representing non-tumurous examples
        augmenting_data(file_dir=image_dir+'no', n_generated_samples=9, save_to_dir=augmented_data_path+'no')

        end_time = time.time()
        execution_time = (end_time - start_time)
        print(f"Elapsed Time: {format(hms_string(execution_time))}")

        Elapsed Time: 0:01:27.57
```

Figure 22: Determine the number of augment images

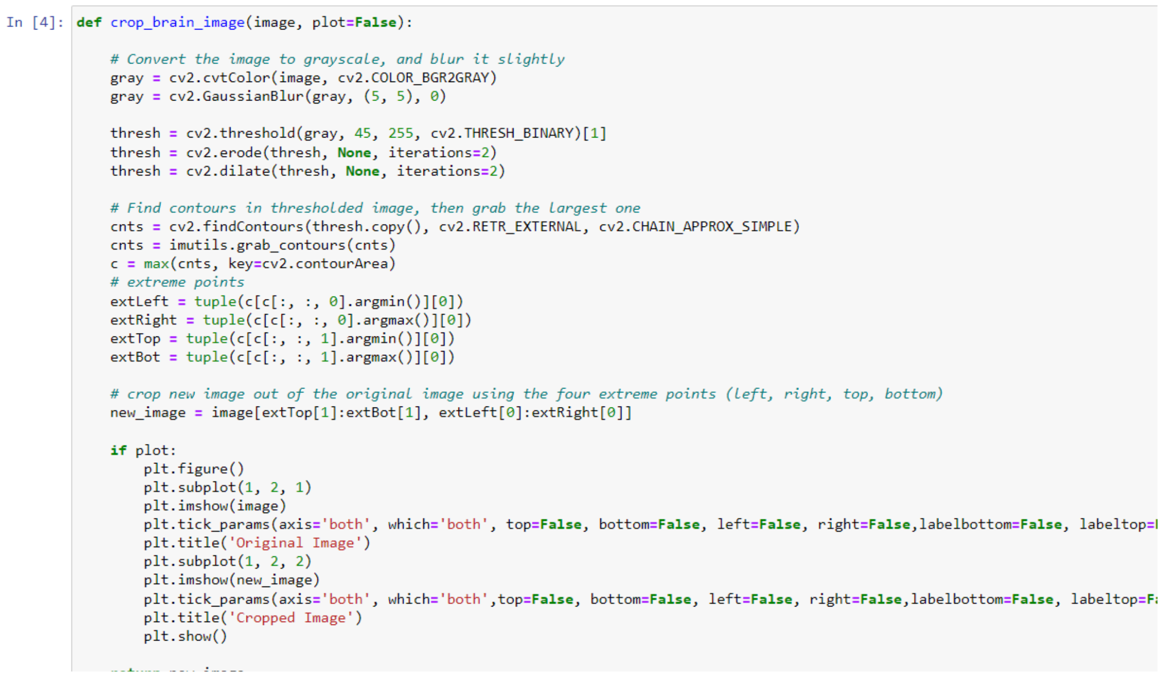
There are 61% of tumorous brain MRI images and 39% non-tumorous brain MRI images in the dataset which was used for the research project. Create nine new images for each image in “no” set and six images for each image in “yes” set, to generate balance dataset by solving this problem. The newly created augmented dataset contains

2065 total number of brain MRI images which has 1085 “yes” images (positive images) and 980 “no” images (negative images). As a percentage, dataset includes with 52.5% “yes” and 47.5% “no” images.

5.3.2 Model No - 1

5.3.2.1 Data Preparation

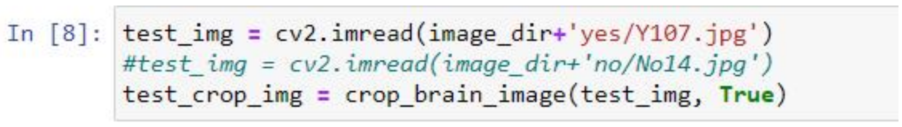
Unnecessary background of the images was removed with use of cropping techniques by finding extreme left, right, top, bottom points of the pre-processed input brain MRI. Code segment which used to crop images is given below.



```
In [4]: def crop_brain_image(image, plot=False):

            # Convert the image to grayscale, and blur it slightly
            gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            gray = cv2.GaussianBlur(gray, (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=
                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=
                plt.title('Cropped Image')
                plt.show()
```

Figure 23: Cropping of MRI

The image is slightly blurred and covered in grayscale in the above code snippet. Then, to reduce minor noise in the photos, run a process of erosion and dilation. Then, to grab the largest image, detect the contours of the thresholded images. Finally, locate the brain MRI's extreme data point and crop the original image to create a new brain MRI.



```
In [8]: test_img = cv2.imread(image_dir+'yes/Y107.jpg')
        #test_img = cv2.imread(image_dir+'no/No14.jpg')
        test_crop_img = crop_brain_image(test_img, True)
```

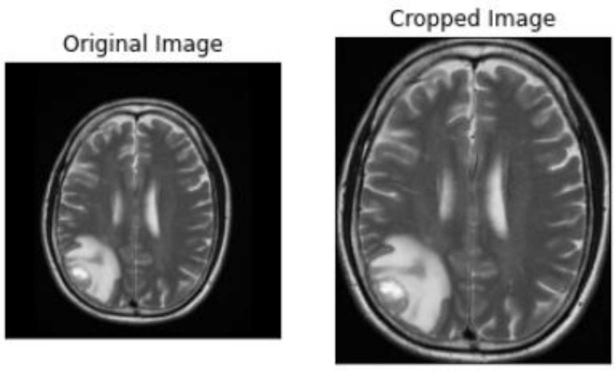
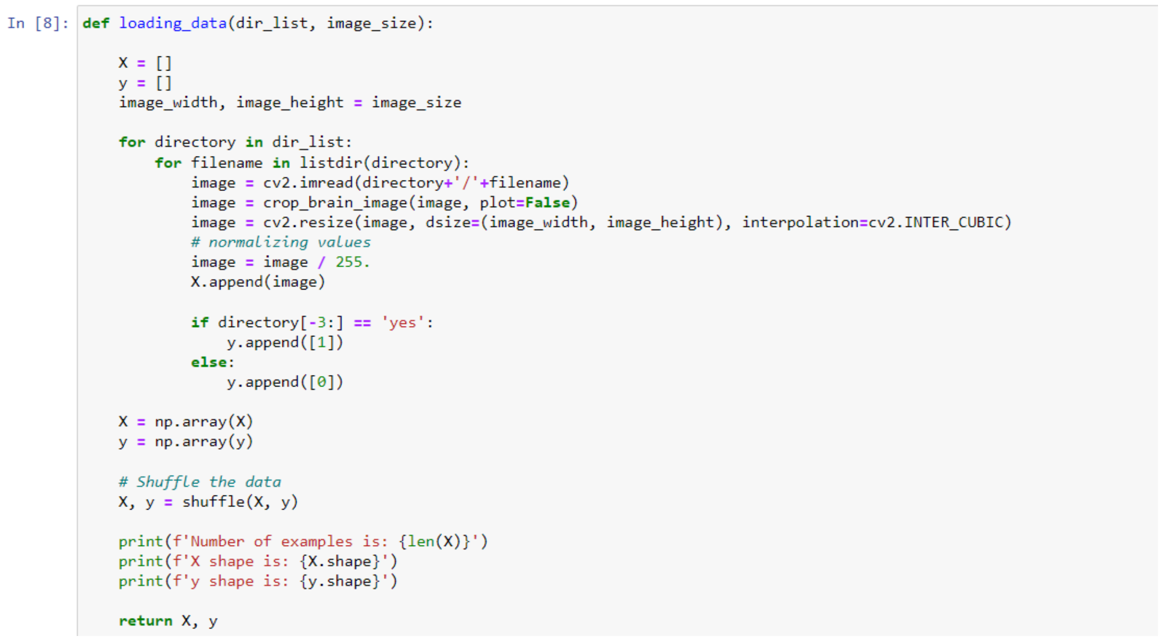


Figure 24: Brain MRI after cropping process

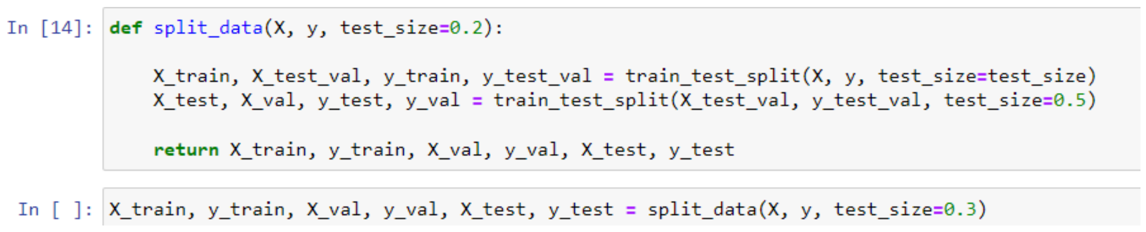
Resize all the images to same size. Because data set contains different size of images and need same size data to feed the convolutional neural network. After resizing, the image size is 240, 240, 5 (image width, image height and number of channels). Normalize the data because need to range scaled pixel value between 0-1.



```
In [8]: def loading_data(dir_list, image_size):

            X = []
            y = []
            image_width, image_height = image_size

            for directory in dir_list:
                for filename in listdir(directory):
                    image = cv2.imread(directory+'/'+filename)
                    image = crop_brain_image(image, plot=False)
                    image = cv2.resize(image, dsize=(image_width, image_height), interpolation=cv2.INTER_CUBIC)
                    # normalizing values
                    image = image / 255.
                    X.append(image)

                    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
```

Figure 21: Loading Data

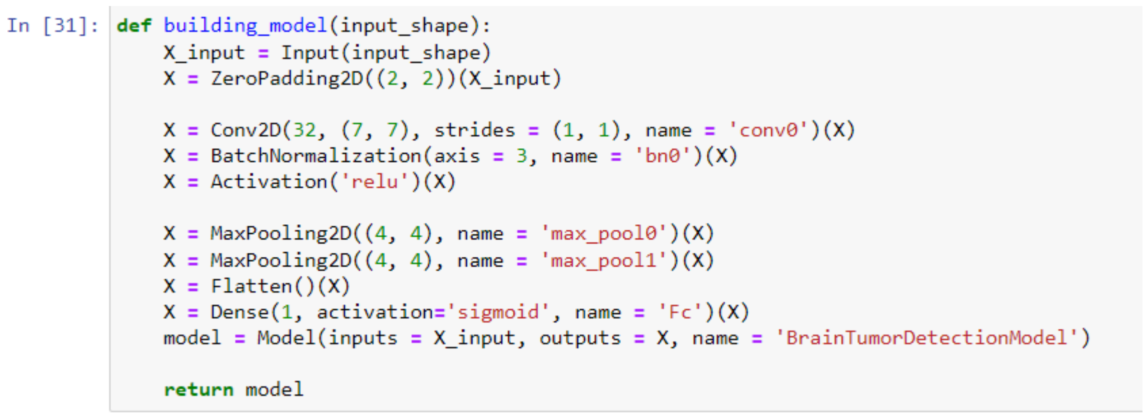
5.3.2.2 Split Dataset



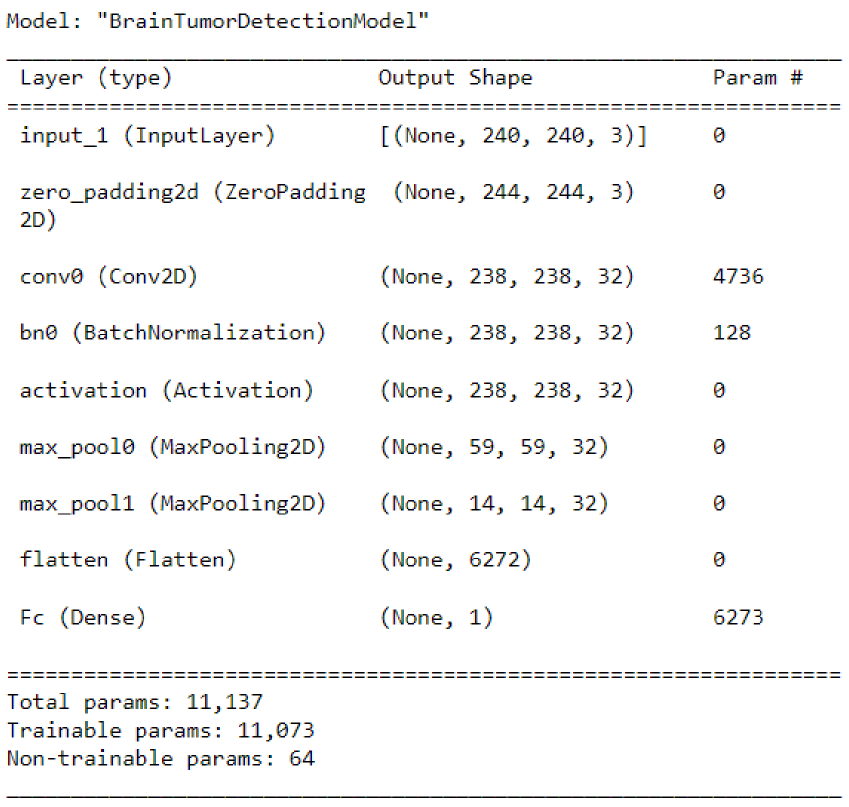
```
In [14]: def split_data(X, y, test_size=0.2):

             X_train, X_test_val, y_train, y_test_val = train_test_split(X, y, test_size=test_size)
             X_test, X_val, y_test, y_val = train_test_split(X_test_val, y_test_val, test_size=0.5)

             return X_train, y_train, X_val, y_val, X_test, y_test

In [ ]: X_train, y_train, X_val, y_val, X_test, y_test = split_data(X, y, test_size=0.3)
```

Figure 22: Splitting Dataset

5.3.2.3 Model Building



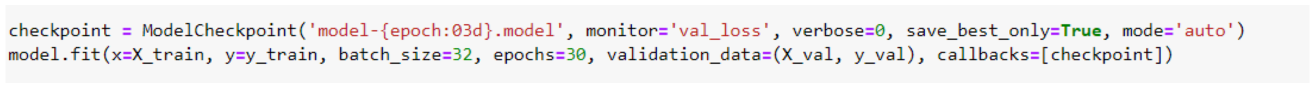
```
In [31]: def building_model(input_shape):
             X_input = Input(input_shape)
             X = ZeroPadding2D((2, 2))(X_input)

             X = Conv2D(32, (7, 7), strides = (1, 1), name = 'conv0')(X)
             X = BatchNormalization(axis = 3, name = 'bn0')(X)
             X = Activation('relu')(X)

             X = MaxPooling2D((4, 4), name = 'max_pool0')(X)
             X = MaxPooling2D((4, 4), name = 'max_pool1')(X)
             X = Flatten()(X)
             X = Dense(1, activation='sigmoid', name = 'Fc')(X)
             model = Model(inputs = X_input, outputs = X, name = 'BrainTumorDetectionModel')

             return model
```



```
Model: "BrainTumorDetectionModel"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 240, 240, 3)]     0

 zero_padding2d (ZeroPadding  (None, 244, 244, 3)      0
 2D)

 conv0 (Conv2D)              (None, 238, 238, 32)      4736

 bn0 (BatchNormalization)    (None, 238, 238, 32)      128

 activation (Activation)     (None, 238, 238, 32)      0

 max_pool0 (MaxPooling2D)    (None, 59, 59, 32)        0

 max_pool1 (MaxPooling2D)    (None, 14, 14, 32)        0

 flatten (Flatten)           (None, 6272)              0

 Fc (Dense)                  (None, 1)                 6273

=================================================================
Total params: 11,137
Trainable params: 11,073
Non-trainable params: 64
_________________________________________________________________
```

Figure 23: Model Summary

5.3.2.4 Model Training

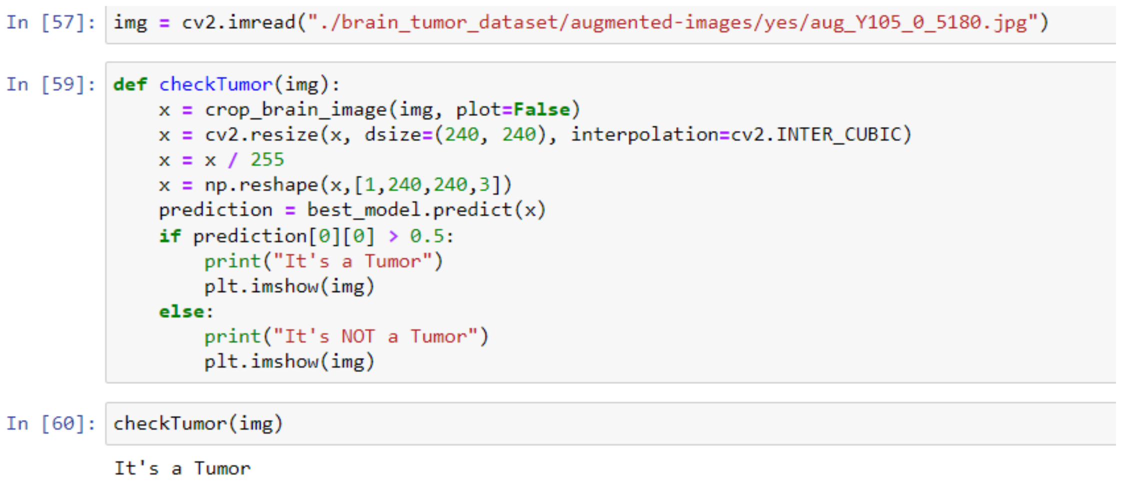


```
checkpoint = ModelCheckpoint('model-{epoch:03d}.model', monitor='val_loss', verbose=0, save_best_only=True, mode='auto')
model.fit(x=X_train, y=y_train, batch_size=32, epochs=30, validation_data=(X_val, y_val), callbacks=[checkpoint])
```

Figure 24: Fitting the model

5.3.2.5 Model Checking

After selecting the model with the highest accuracy as the best model among the saved models, it was used to predict new MRI image of the brain is tumorous or not. Following figures show the result for images.



```
In [57]: img = cv2.imread("./brain_tumor_dataset/augmented-images/yes/aug_Y105_0_5180.jpg")

In [59]: def checkTumor(img):
             x = crop_brain_image(img, plot=False)
             x = cv2.resize(x, dsize=(240, 240), interpolation=cv2.INTER_CUBIC)
             x = x / 255
             x = np.reshape(x,[1,240,240,3])
             prediction = best_model.predict(x)
             if prediction[0][0] > 0.5:
                 print("It's a Tumor")
                 plt.imshow(img)
             else:
                 print("It's NOT a Tumor")
                 plt.imshow(img)

In [60]: checkTumor(img)

         It's a Tumor
```

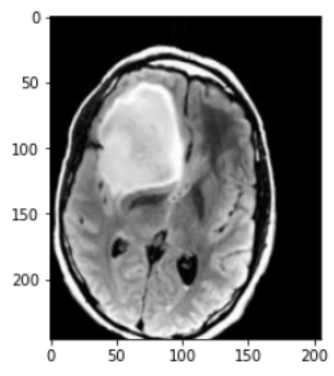


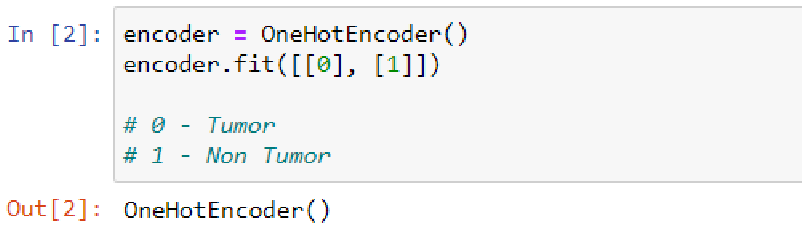
Figure 25: Model Checking

5.3.3 Model No - 2

5.3.3.1 Data Preparation

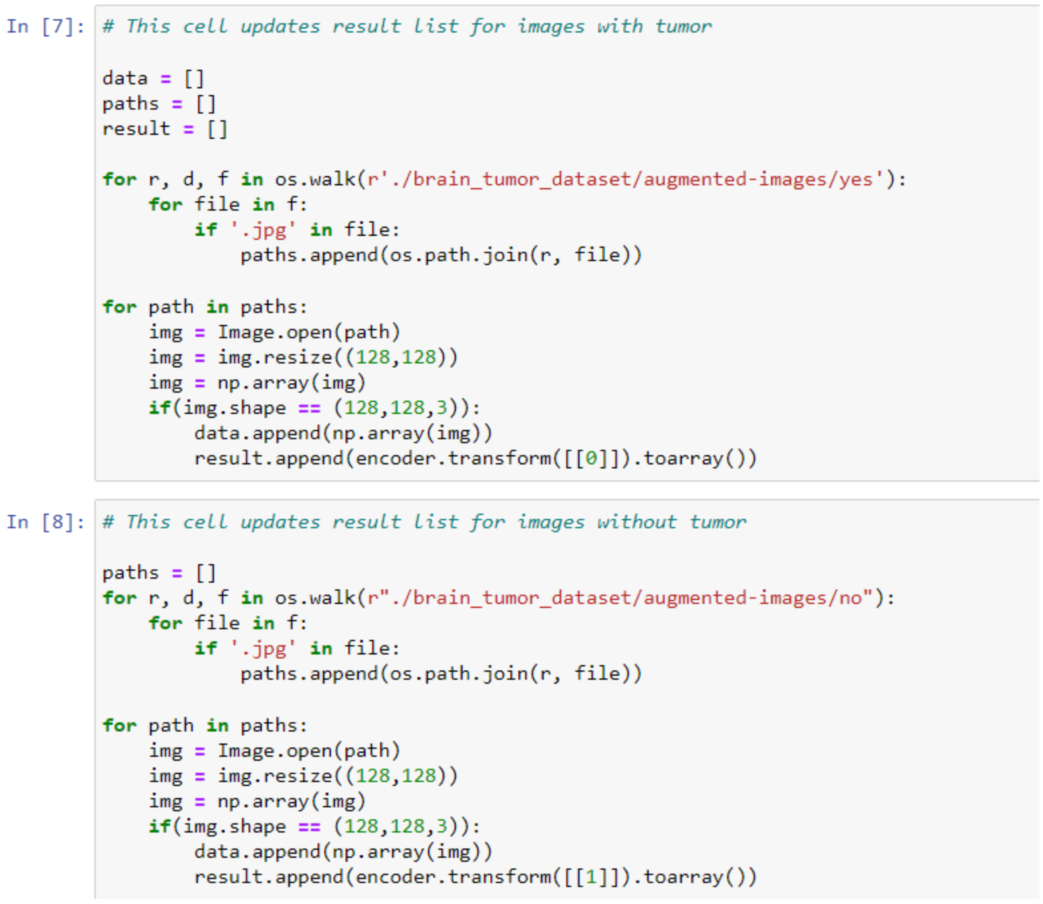
After importing the necessary libraries, the categorical features were encoded using one hot encoding function. Since there are just two variables as ‘Tumor’ and ‘Non-Tumor’, one hot encoder is enough to use.

Then the tumorous and non-tumorous data are inserted and loaded into lists separately.



```
In [2]: encoder = OneHotEncoder()
        encoder.fit([[0], [1]])

        # 0 - Tumor
        # 1 - Non Tumor

Out[2]: OneHotEncoder()
```

Figure 26: Implementing One Hot Encoder

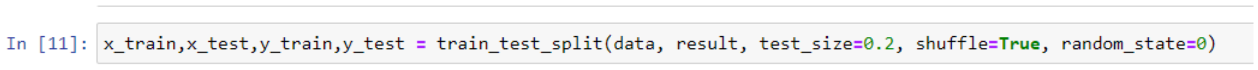


```
In [7]: # This cell updates result list for images with tumor

        data = []
        paths = []
        result = []

        for r, d, f in os.walk(r'./brain_tumor_dataset/augmented-images/yes'):
            for file in f:
                if '.jpg' in file:
                    paths.append(os.path.join(r, file))

        for path in paths:
            img = Image.open(path)
            img = img.resize((128,128))
            img = np.array(img)
            if(img.shape == (128,128,3)):
                data.append(np.array(img))
                result.append(encoder.transform([[0]]).toarray())
```

```
In [8]: # This cell updates result list for images without tumor

        paths = []
        for r, d, f in os.walk(r"./brain_tumor_dataset/augmented-images/no"):
            for file in f:
                if '.jpg' in file:
                    paths.append(os.path.join(r, file))

        for path in paths:
            img = Image.open(path)
            img = img.resize((128,128))
            img = np.array(img)
            if(img.shape == (128,128,3)):
                data.append(np.array(img))
                result.append(encoder.transform([[1]]).toarray())
```

Figure 27: Loading Data

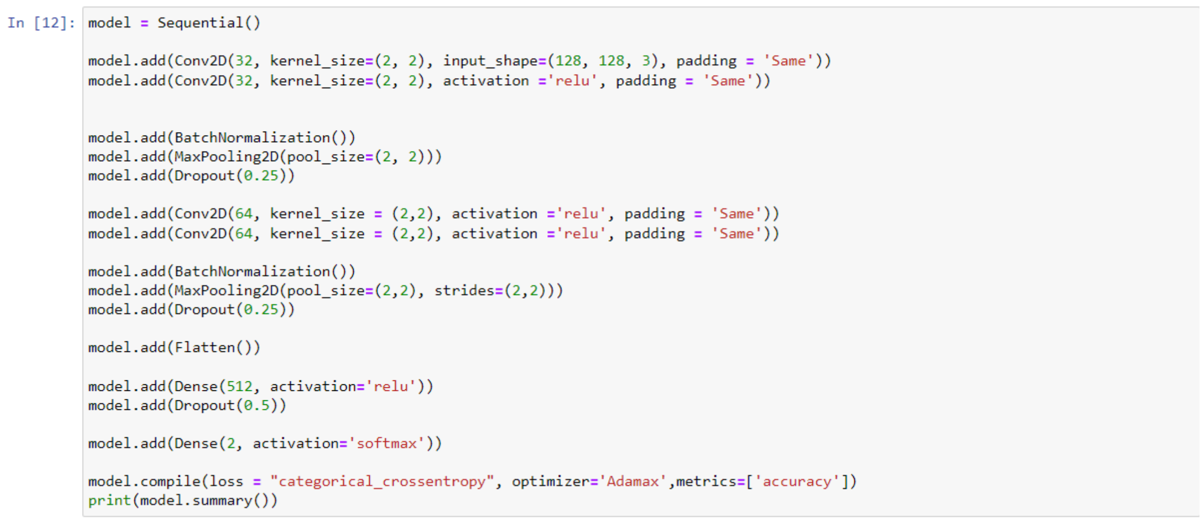
5.3.3.2 Split Dataset



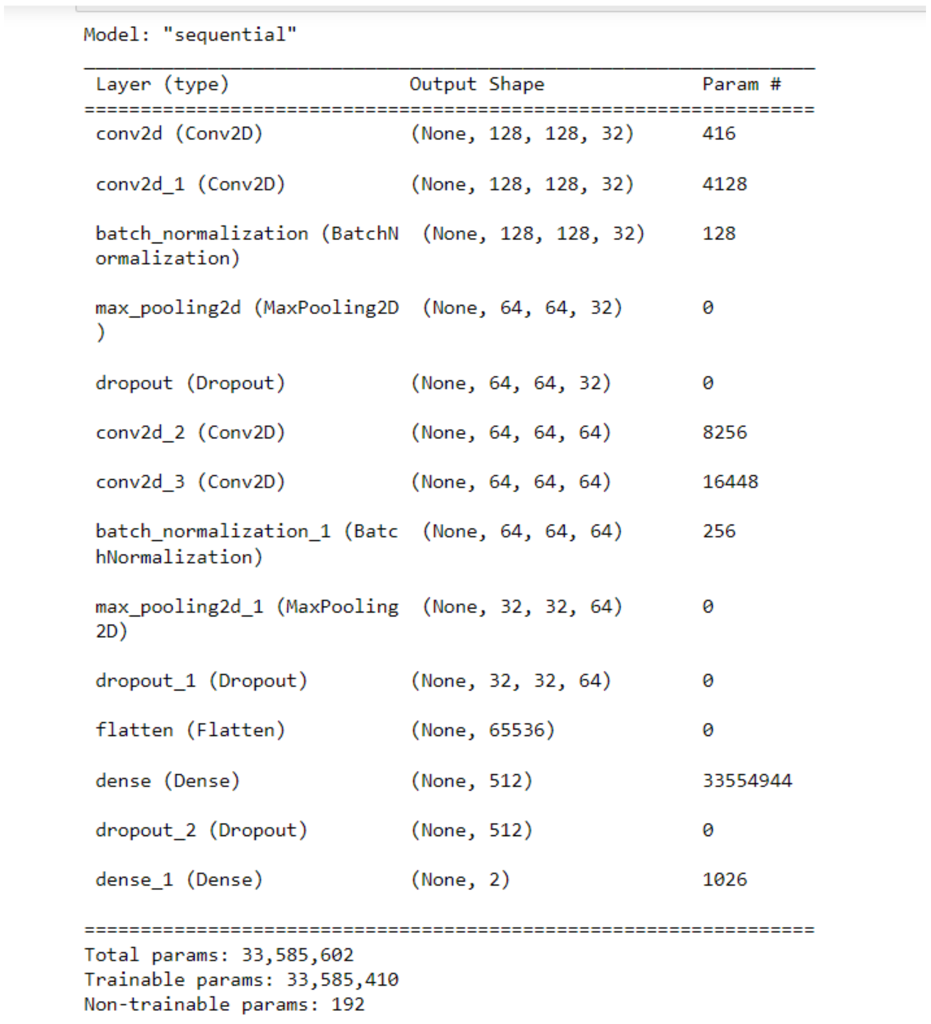
```
In [11]: x_train,x_test,y_train,y_test = train_test_split(data, result, test_size=0.2, shuffle=True, random_state=0)
```

Figure 28: Splitting Dataset

5.3.3.3 Model Building



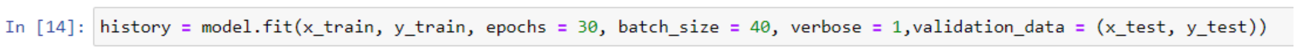
```
In [12]: model = Sequential()

         model.add(Conv2D(32, kernel_size=(2, 2), input_shape=(128, 128, 3), padding = 'Same'))
         model.add(Conv2D(32, kernel_size=(2, 2), activation ='relu', padding = 'Same'))


         model.add(BatchNormalization())
         model.add(MaxPooling2D(pool_size=(2, 2)))
         model.add(Dropout(0.25))

         model.add(Conv2D(64, kernel_size = (2,2), activation ='relu', padding = 'Same'))
         model.add(Conv2D(64, kernel_size = (2,2), activation ='relu', padding = 'Same'))

         model.add(BatchNormalization())
         model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
         model.add(Dropout(0.25))

         model.add(Flatten())

         model.add(Dense(512, activation='relu'))
         model.add(Dropout(0.5))

         model.add(Dense(2, activation='softmax'))

         model.compile(loss = "categorical_crossentropy", optimizer='Adamax',metrics=['accuracy'])
         print(model.summary())
```



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 128, 128, 32)      416

 conv2d_1 (Conv2D)           (None, 128, 128, 32)      4128

 batch_normalization (BatchN  (None, 128, 128, 32)     128
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 64, 64, 32)       0
 )

 dropout (Dropout)           (None, 64, 64, 32)        0

 conv2d_2 (Conv2D)           (None, 64, 64, 64)        8256

 conv2d_3 (Conv2D)           (None, 64, 64, 64)        16448

 batch_normalization_1 (Batc  (None, 64, 64, 64)       256
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 32, 32, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 32, 32, 64)        0

 flatten (Flatten)           (None, 65536)             0

 dense (Dense)               (None, 512)               33554944

 dropout_2 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 2)                 1026

=================================================================
Total params: 33,585,602
Trainable params: 33,585,410
Non-trainable params: 192
```

Figure 29: Model Summary

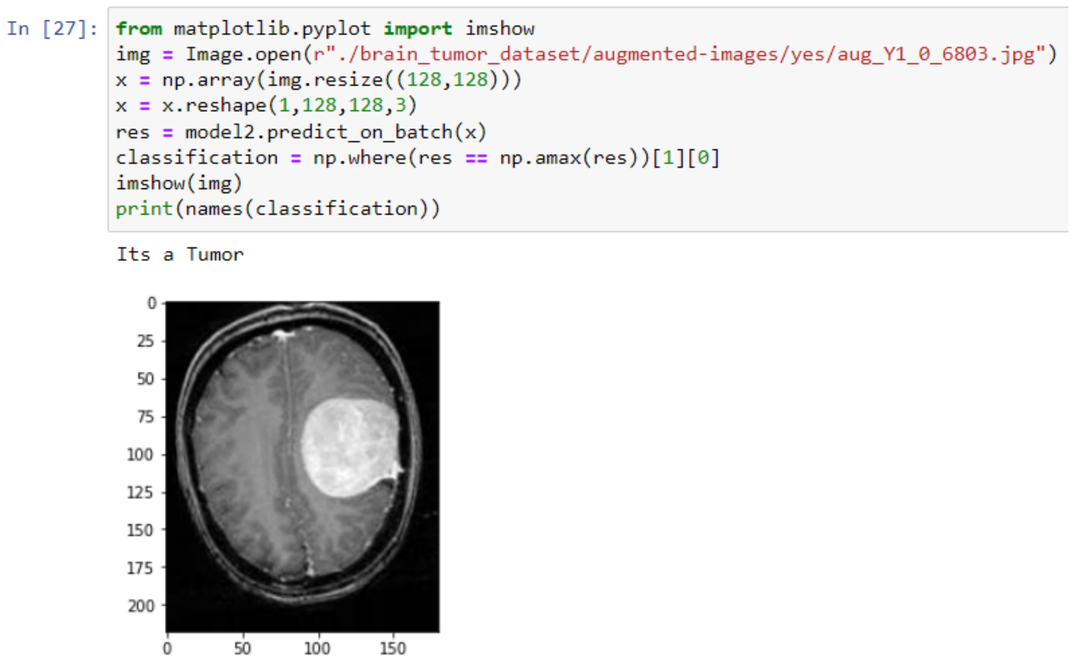
5.3.3.4 Model Training



```
In [14]: history = model.fit(x_train, y_train, epochs = 30, batch_size = 40, verbose = 1,validation_data = (x_test, y_test))
```

Figure 30: Model Fitting

5.3.3.5 Model Checking



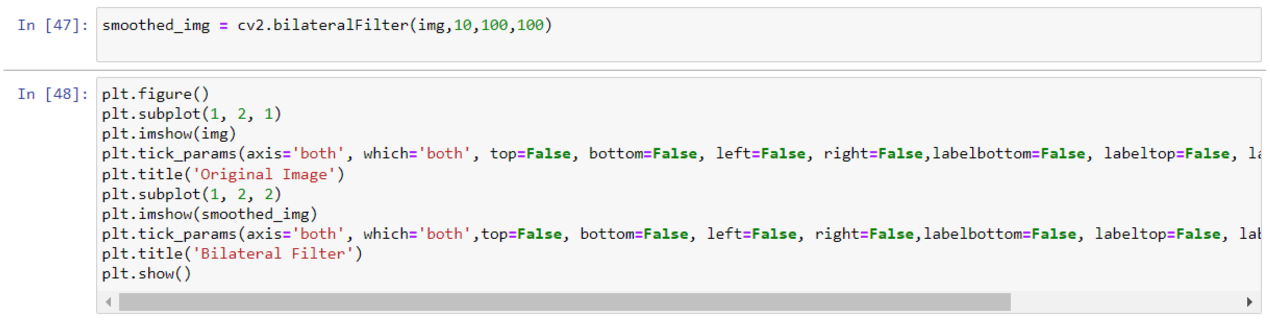
```
In [27]: from matplotlib.pyplot import imshow
         img = Image.open(r"./brain_tumor_dataset/augmented-images/yes/aug_Y1_0_6803.jpg")
         x = np.array(img.resize((128,128)))
         x = x.reshape(1,128,128,3)
         res = model2.predict_on_batch(x)
         classification = np.where(res == np.amax(res))[1][0]
         imshow(img)
         print(names(classification))

         Its a Tumor
```

Figure 31: Model Checking

5.4 Image Processing

5.4.1 Bilateral Filtering

Here Bilateral Filtering method has used to refine edges. The OpenCV library provides Bilateral Filter method which comprises of four arguments such that the input file, the diameter of each pixel neighbourhood, Sigma Colour and Sigma Space respectively.



```
In [47]: smoothed_img = cv2.bilateralFilter(img,10,100,100)
```

```
In [48]: plt.figure()
         plt.subplot(1, 2, 1)
         plt.imshow(img)
         plt.tick_params(axis='both', which='both', top=False, bottom=False, left=False, right=False,labelbottom=False, labeltop=False, la
         plt.title('Original Image')
         plt.subplot(1, 2, 2)
         plt.imshow(smoothed_img)
         plt.tick_params(axis='both', which='both',top=False, bottom=False, left=False, right=False,labelbottom=False, labeltop=False, lat
         plt.title('Bilateral Filter')
         plt.show()
```

Figure 32: Code snippet to apply bilateral filtering

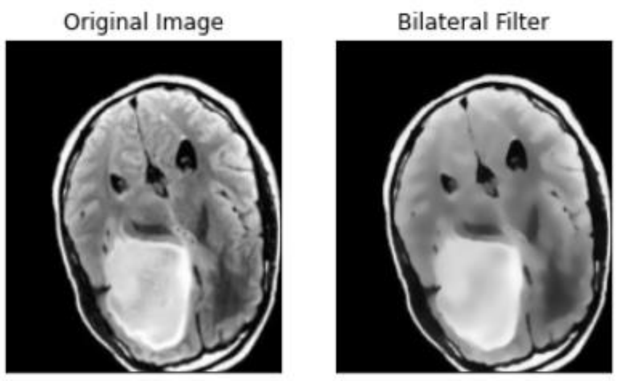
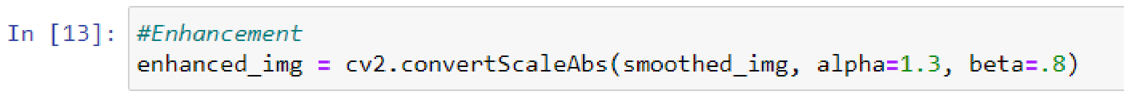


Figure 33: After applying bilateral filtering

5.4.2 Enhancement

For the easiness of contour finding, the smoothed image is enhanced by using pixel transformation function.



```
In [13]: #Enhancement
         enhanced_img = cv2.convertScaleAbs(smoothed_img, alpha=1.3, beta=.8)
```

Figure 34: Applying pixel transformation

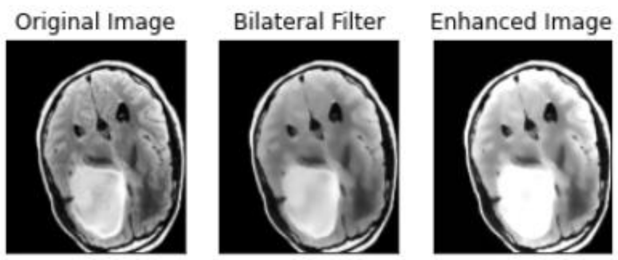
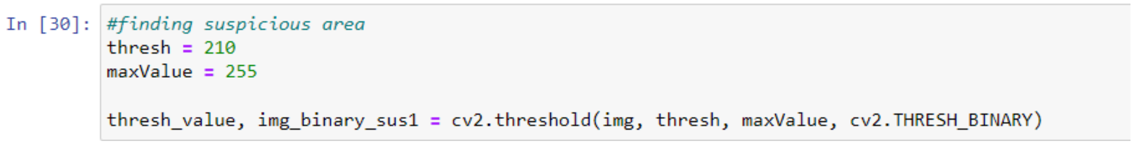


Figure 35: After the enhancement

5.4.3 Thresholding

Binary thresholding has used to find the separated objects by considering the intensity of the input image.



```
In [30]: #finding suspicious area
         thresh = 210
         maxValue = 255

         thresh_value, img_binary_sus1 = cv2.threshold(img, thresh, maxValue, cv2.THRESH_BINARY)
```

Figure 36: Code snippet for thresholding

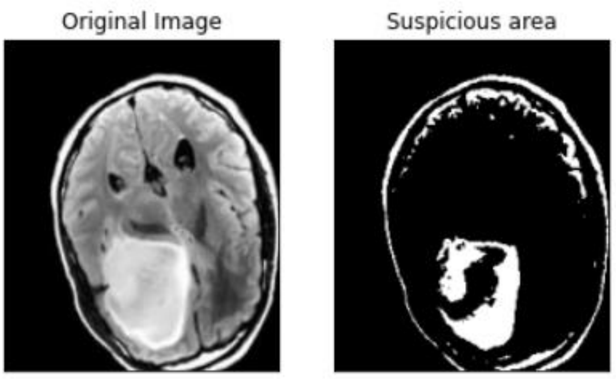


Figure 37: After applying thresholding

5.4.4 Morphological operations

Erosion

Erosion is used to shrinking process of an image. Objects with connected edges in an image can be separated by using this technique.

```
In [36]: #Morphological Operations
         #Erosion
         kernel = np.ones((5,5),np.uint8)
         erosion = cv2.erode(img,kernel,iterations=1)
         plt.imshow(erosion)

Out[36]: <matplotlib.image.AxesImage at 0x21bf75f41f0>
```

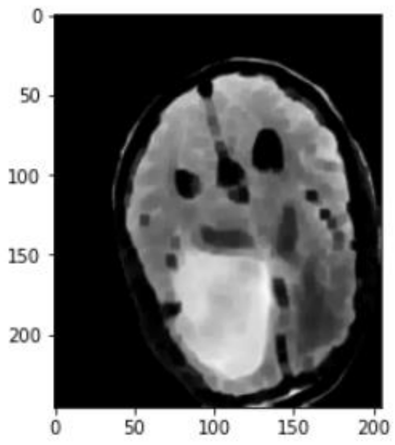


Figure 38: Applying erosion

Dilation

Pixels in a picture are enlarged via dilation. It increases the white pixels and decrease the black pixels according to the structural element.

```
In [38]: #Dilation
         kernel = np.ones((3,3), np.int8)
         erosion = cv2.dilate(img_binary_sus1,kernel, iterations=1)
         plt.imshow(erosion)

Out[38]: <matplotlib.image.AxesImage at 0x21bf57779a0>
```

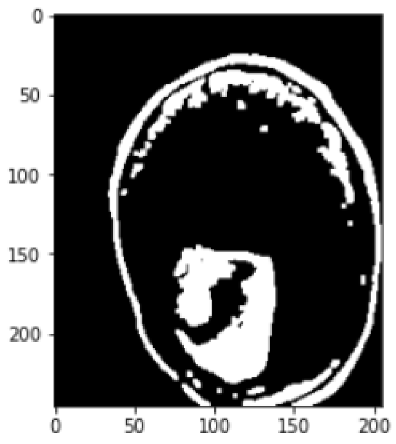


Figure 39: Applying dilation

5.4.5 Contour Identification

Contours are the separate objects where the number of pixels connected each other. This method returns an array of contours with its area. Hence, the maximum value gives the contour we need which is the tumor.

```
In [51]: #Contours for suspicious area

         cnts_for_sus_1 = cv2.findContours(img_binary_sus2.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
         cnts_for_sus_1 = imutils.grab_contours(cnts_for_sus_1)

         c1_max = max(cnts_for_sus_1, key=cv2.contourArea)
         contour_detect=cv2.drawContours(img, [c1_max], -1, (0,255,0), 1)
```

```
In [58]: plt.figure()
         plt.subplot(1, 2, 1)
         plt.imshow(contour_detect)
         plt.tick_params(axis='both', which='both', top=False, bottom=False, left=False, right=False,labelbottom=False, labeltop=False, la
         plt.title('Tumor Identified Image')

         plt.show()
```

Tumor Identified Image

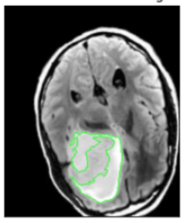


Figure 40: Contour Identification and Final output

Chapter 06 – Results and Discussion

6.1 CNN Model

Confusion Matrix

A confusion matrix is a method of summarizing a classification algorithm's performance. Calculating a confusion matrix can help to figure out what the classification model is getting right and where it's going wrong.

In this research two CNN sequential models were developed to classification of brain tumor. Among these two, the best model is chosen for further use. The respective confusion matrices for each model are depicted below.

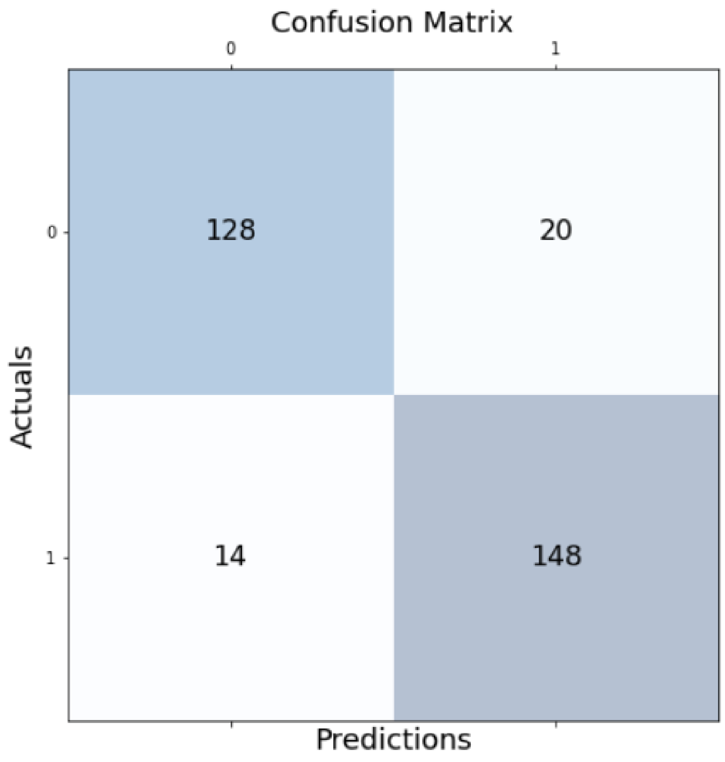


Figure 41: Confusion Matrix for model 01

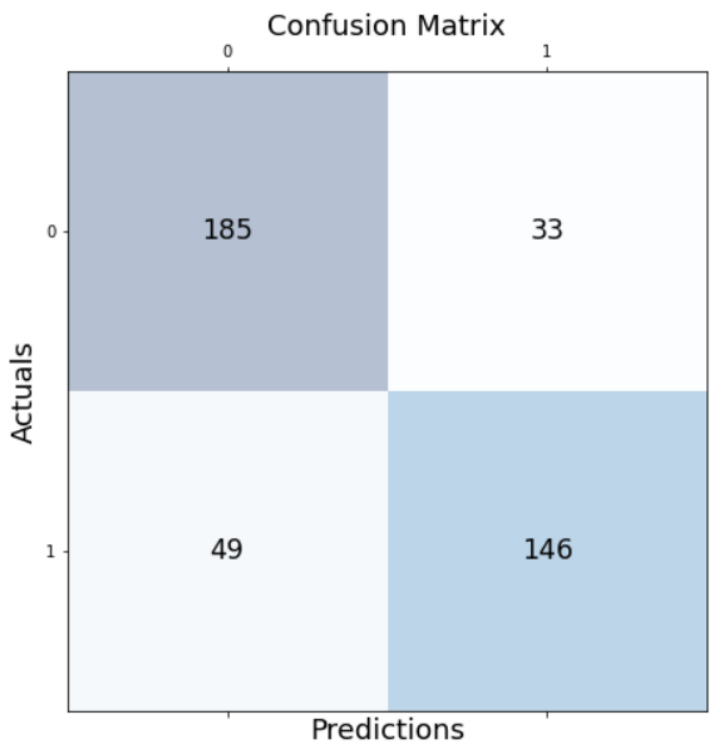


Figure 42: Confusion matrix for model 02

Classification Report

It's one of the metrics used to evaluate the success of a classification-based machine learning model. It shows the precision, recall, F1 score, and support of the model. It allows to gain a better understanding of trained model's overall performance.

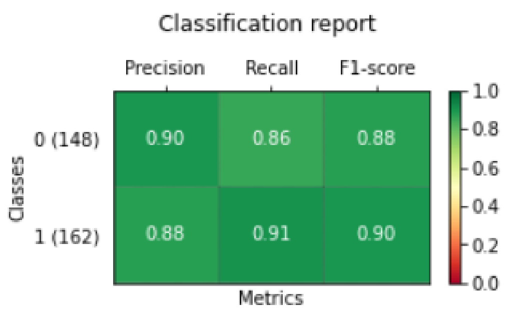


Figure 43: Model 01

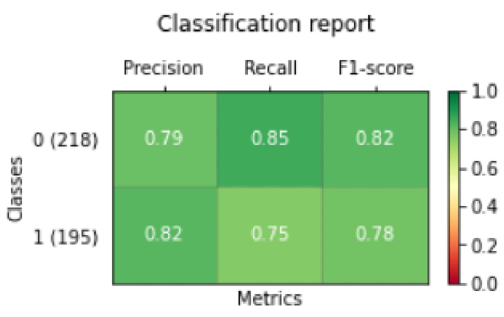


Figure 44: Model 02

Accuracy and the Loss of the Model

In machine learning, learning curves are a common diagnostic tool for algorithms that learn progressively from a training dataset. After each update during training, the model can be tested on the training dataset and a holdout validation dataset, and graphs of the measured performance can be produced to display learning curves.

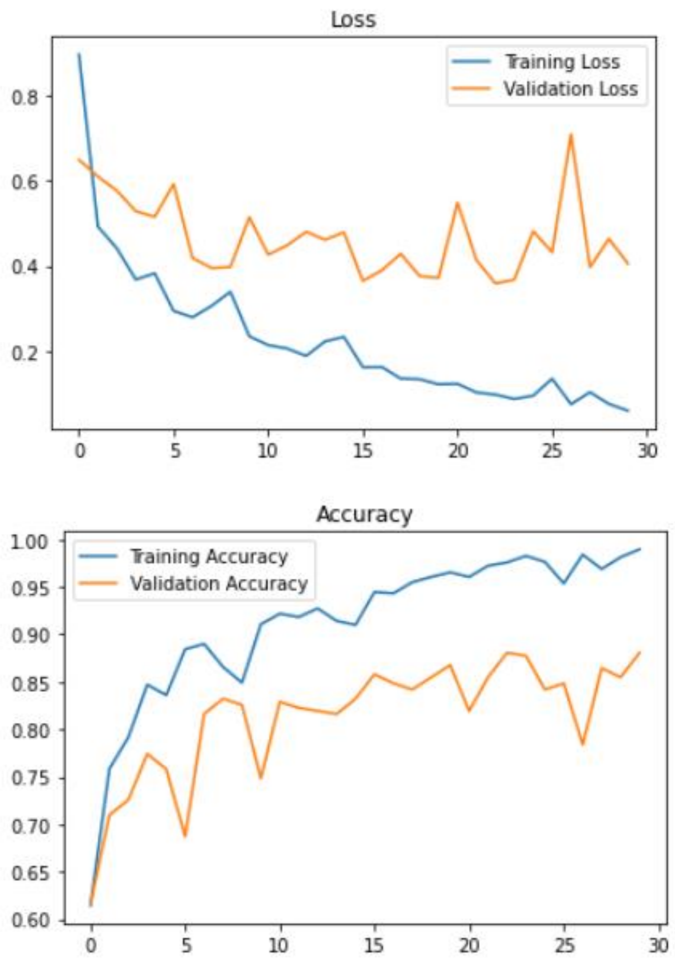
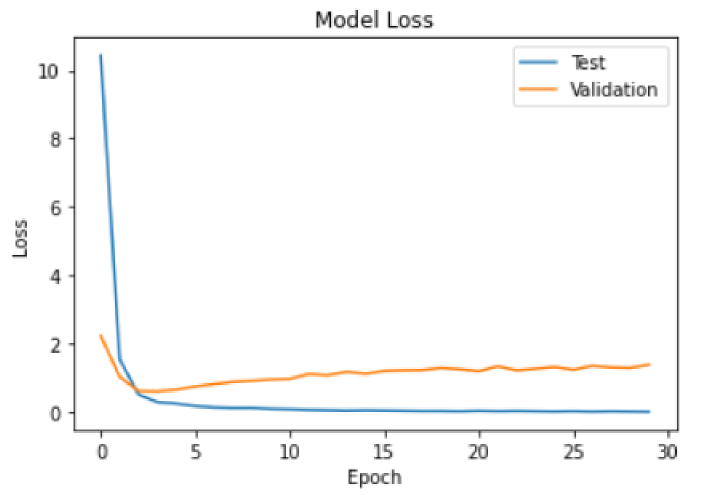


Figure 45: Learning curves of Model 01



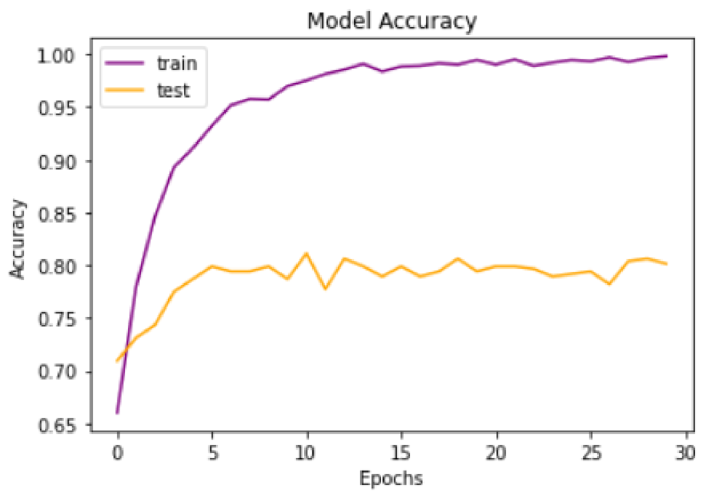


Figure 46: Learning curves of Model 02

Considering the above curves and reports, model 01 has high accuracy, high f1 score as well as precise confusion matrix than the model 02. Hence, model 01 is used for classification of brain tumors.

6.2 Discussion

The aim of this project is to implement a methodology to optimize the brain tumor detection using MR images. The proposed methodology is based on a core assumption that even if the doctors' experience varies, the manual brain tumor identification technique will remain the same. When investigating the manual procedure of brain tumor detection, it is convinced that final judgment differed slightly depending on the doctors and their experience. But this methodology could be feed into mechanism parallel to MRI scanning to give predictions to radiologists as well as doctors to aid their decisions.

There were several limitations faced within the project. It is precise that if this kind of CNN could train using at least 20,000 MRI images. It is difficult to find such an amount as those are confidential data and the PCs are not that enough for training a large model.

Other than the proposed methodology, this can be more optimized using following enhancements.

- Develop a parallel model to identify tumor severity.

References

[1] Shivakumarswamy G.M., Akshay Patil.V., Chethan T.A., Prajwal B.H., Sagar.V.Hande, "Brain tumour detection using Image processing and sending tumour information over GSM," International Journal of Advanced Research in Computer and Communication Engineering, vol. 5, no. 5, 2016.

[2] American Society of Clinical Oncology (ASCO), https://www.cancer.net/

[3] American Cancer Society, https://www.cancer.org/cancer/brain-spinal-cord-tumors-adults/about/key-statistics.html

[4] D. M. K. N. Nandha Gopal, Diagnose Brain Tumor Through MRI Using Image Processing Clustering Algorithms Such as Fuzzy C Means Along with Intelligent Optimization Techniques, 2010

[5] P. Gamage, "Identification of Brain Tumor using Image Processing Techniques," ResearchGate, 2017

[6] Sourabh Hanwat, Chandra Jayaraman, "Convolutional Neural Network for Brain Tumor Analysis Using MRI Images," International Journal of Engineering and Technology, vol. 11, 2019

[7] J. Seetha, S. Selvakumar Raja, "Brain Tumor Classification Using Convolutional Neural Networks," Biomedical & Pharmacology Journal, vol. 11, p. 4, 2018.

[8] T. A. R. V. A.Lakshmi, "Noise and skull removal of brain magnetic resonance image using curvelet transform and mathematical morphology" in 2014. International Conference on Electronics and Communication Systems (ICECS), 2014

[9] K. C. I. K. M. P. S. K. B. Sudipta Roy, "Artefact Removal from MRI of Brain Image," in International Refereed Journal of Engineering and Science (IRJES), March 2013

[10] Ali Ari, Davut Hanbay, "Deep learning-based brain tumor classification and detection system," Turkish Journal of Electrical Engineering & Computer Sciences, p. 12, 2018

[11] M Malathi, P Sinthia, "Brain Tumour Segmentation Using Convolutional Neural Network with Tesnsor Flow," Asian Pacific Journal of Cancer Prevention

[12] S. S. N. S. S. K. Vipin Y. Borole, "Image Processing Techniques for Brain Tumor Detection: A Review," International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), vol. Volume 4, no. Issue 5(2), October 2015

[13] N. Sravanthi et al “Brain Tumor detection using Image Processing”, International Journal of Scientific Research in Computer Science, Engineering and Information Technology, May 2021