A Project Report on

Brain tumor detection using Canny Edge Technique and Convolutional Neural Network

Submitted in partial fulfillment for the award of the degree of

B.Tech (Computer Science Engineering with Specialization in Information Security)

By
Ishitva Verma(18BCI0155)
Saloni Uday Parekh(18BCI0212)

Course Code: CSE 3001
Course Title: Software Engineering

Under the guidance of

Prof. Deepikaa S
Assistant Professor (Junior),
VIT , Vellore.

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

October 2020

Table of Contents

Abstract

Brain tumor is a group of tissue that is pre arranged by a slow addition of irregular cells. It occurs when abnormal formation of cells takes place within the brain. Recently it is becoming a major cause of death for many people. The seriousness of brain tumor is very big among all the variety of cancers, so to save a life, immediate detection and proper treatment is required. Detection of these cells is a difficult problem, because of the formation of the tumor cells. It is very essential to compare brain tumors from the MRI treatment. Imaging technology in medicine made the doctors see the interior portions of the body for easy diagnosis. It also helps doctors to make keyhole surgeries for reaching the interior parts without really opening too much of the body. CT scanner, Ultrasound and Magnetic Resonance Imaging (MRI) took over X-Ray imaging by making the doctors look at the body's elusive third dimension. Image processing techniques along with convolutional neural networks developed for analyzing remote sensing data may be changed to predict the outputs of medical imaging systems to get best benefit to analyze symptoms of the patients with ease.

Through this project we look forward to building a software that combines the technologies of Image Processing along with Machine Learning through a trained Convolutional Neural Network Model that will help us detect brain tumor on a preprocessed MRI Scan Image of the brain with an accuracy of about 91%.

Acknowledgements

We are grateful to VIT University for encouraging and giving us the opportunity to perform this project. We would like to convey our special thanks of gratitude to **Prof. Deepika S.** for her continued support and advice behind ideating and implementing the project : **Brain tumor detection using Canny Edge Technique and Convolutional Neural Network.** Her guidance and words of help played a major role in the timely and successful completion of the project. We thank her for constantly encouraging us to pursue the best results. Last but not the least, we are thankful to all family and friends for their help. The completion of this project is attributed to everyone's combined effort.

Ishitva Verma

Saloni Uday Parekh

1. Introduction

1.1. Motivation

Recently brain tumor is becoming a major cause of death for many people. The seriousness of brain tumor is very big among all the variety of cancers, so to save a life, immediate detection and proper treatment is required. Detection of these cells is a difficult problem, because of the formation of the tumor cells. It is very essential to easily and efficiently detect brain tumors from MRI treatment, which can give results with maximum accuracy.

Statistical methods and simple thresholds are often not able to accurately segment the numerous elements of the GBM (Glioblastoma Multiforme) tumor, like local contrast enhancement, edema and necrosis which leads to inaccurate detection. Most voxel-based methods cannot achieve satisfactory results in larger data sets, and the methods based on generative or discriminative models have intrinsic limitations during application, such as small sample set learning and transfer. Therefore we thought of developing something to overcome these challenges and help detect tumors with as much accuracy as possible, with the help of our knowledge that we have gained in curriculum courses of Image Processing and Machine Learning.

1.2. Aim of the proposed Work

We aim to take a brain MRI scan as input and predict whether a tumor is present in the brain or not.The project shall firstly incorporate pre-processing of MRI scan which includes steps such as performing Watershed Segmentation to remove noise and Canny edge detection to perform erosion and dilation (morphological operations).After this, the Canny-edged pre-processed image will be given as an input to the Convolutional Neural Network Model which will predict whether the tumor is present.

1.3. Objective(s) of the proposed work

<u>Our Approach for understanding the Problem:</u>

Segmentation and detection of a brain tumor like glioblastoma multiforme (GBM) in magnetic resonance (MR) images is quite difficult due to its intrinsically heterogeneous signal characteristics. We intend to develop and test a robust segmentation method for brain tumor MRI scans.

<u>Basic Objective of Pre-Processing of MRI scan</u>

- *Performing Watershed Segmentation*

Any grayscale image can be observed as a topographic surface where high intensity denotes peaks and hills while low intensity denotes valleys. We start filling every isolated valley (local minima) with different coloured water (labels). With the water rises, depending on the peaks (gradients) nearby, water from different valleys, obviously with different colours will start to merge. To avoid that, we build barriers in the locations where water merges. Then we continue the work of filling water and building barriers by the time all the peaks are under water. Then the barriers give the segmentation result. [7] But this approach gives us over-segmented results due to noise or any other irregularities in the image. So OpenCV implemented a marker-based watershed algorithm where we can specify which all valley points are to be merged and which are not. It is an interactive image segmentation.

- *Otsu's binarization to threshold the image.*

Now we need to deduct any small white noises in the image. For that we can use morphological openings. We can implement morphological closing to deduct any small holes in the object. So, now we know for sure that regions near to the centre of objects are foreground and regions much away from the object are background. Only region we are not sure of is the boundary region of the tumor. So, we need to extract the area where we are sure that it is the tumor. Erosion removes the boundary pixels. So with whatever part that is remaining, we can be sure it is the tumor. That would work if objects (different parts in the brain tumor and surrounding) are not touching each other. But if they touch each other, another good option would be to find the distance transform and apply a proper threshold.

- *Dilation of the result.*

Next, we need to find the area where we are sure it isn't the tumor. For that, we shall dilate the result. Dilation increases the object boundary to background. This helps us make sure that whatever region in background in result is really a background, since the boundary region is removed. The remaining regions are the ones which we are not aware of, whether it is coins or background. Watershed algorithms should find it. We will mark unknown regions, defined by unknown, with 0.

- *Applying Watershed*

Now once our marker is ready, it is time for the final step, apply watershed. Then the marker image will be modified. The boundary region will be marked with -1. So, in this way, the brain MRI scan will be pre-processed. After this, the pre-processed image will be given as an input to the Convolutional Neural Network Model which shall predict whether the tumor is present.

1.4. Report Organization

The report has been compiled meticulously after the implementation of the project. Section 2 is about the literature survey conducted by our team. Section 3 specifies the analysis of the proposed system. Section 4 goes into the detailed design of the system. Section 5 contains the implementation code and proof of the testing modules conducted to debug our system. Section 6 is the conclusion for our report. It also contains the limitations of the system that we have identified and the different measures that we intend to work upon in the future. Section 7 contains the references.

2. Literature Survey

Brain Tumor Detection has always remained a topic for research and improvement in the field of Machine Learning. Giving the system capability to detect the tumor based on the input that is a brain MRI Scan, automatically, using different algorithms and ML models was the aim for these systems. We have come far in this research field and following are some research work done.

2.1 Survey of the Existing Models/Work

Pratima Purushottam Gumaste and Vinayak K. Bairagi [1] developed an algorithm that facilitates the study of feature extraction from the brain right and left hemispheres.The proposed study, also highlights a completely different advanced higher order statistical features extracted from the chosen region of brain slice. The tumor area is extracted from statistical features using Support Vector Machine. The proposed methodology can be used to locate tumor tissues based on a single-spectral structural Magnetic resonance image.

T. Kalaiselvi, P. Sriramakrishnan and P. Nagaraja [2] constructed a brain tumor boundary using bi-modal fuzzy histogram thresholding and edge indication map (EIM). The proposed work has two major steps. Initially it is aimed to enhance the contrast in order to make the sharp edges. An intensity transformation is used for contrast enhancement with automatic threshold value produced by bimodal fuzzy histogram thresholding technique. Next the EIM is generated by hybrid approach with the results of existing edge operators and maximum voting scheme. The 3D volume construction using edge indication maps is very useful to analyze the brain tumor location during the surgical planning process.

Dr. Sourabh Mukharjee and Minakshi Sharma [3] presented an image segmentation technique for locating brain tumor. The work that has been proposed has been divided in two phases- In the first phase MRI image database is collected and then preprocessing is done to improve quality of image. Second-phase includes three steps-Feature selection, Feature extraction, and Image segmentation. For feature extraction proposed work uses GLCM . In order to improve accuracy only a subset of features is chosen using hybrid Genetic algorithm consisting of Genetic Algorithm and fuzzy rough set, and based on these features fuzzy rules and membership functions are predicted for segmentation of brain tumor from MRI images.

Prince Ebenezer Adjei, Joyce Raissa Yaho Ndjanzoue, Richard Junior Amedzrovi Agbesi and Henry Nunoo-Mensah[4] implemented Simple Linear Iterative Clustering (SLIC) to segment brain images as per their color and spatial proximities. The ratio of the mean and variance of the image pixels are determined in order to obtain an optimum threshold value. Region merging after thresholding was carried out. The final output image was an image with circled out tumor sections. The segmentation sticks to boundaries and the procedure is very speedy and reproducible.

Prof. P.Senthil [5] encapsulated the information about the brain tumor and an Sushisen algorithm that uses BRATS 2016 for basic detection of the brain tumor location with the help of image mining. There are different scanning techniques like X-ray, MRI, CT scan that are used by the radiologist to a great extent to get information about human body anatomy to analyze the diseases. Modern image mining techniques are helpful for pre-mining scanned images in medical areas. Modern image mining techniques improve these scanned images to get more details about human anatomy (Relevance +Accuracy +Time) in earlier stages. Image mining technique can help physicians and practitioners diagnose diseases easily.

Sathya Narayana N, Vidya T and Sarthak U Kuchalli. [6] present an approach to detect whether an MRI scan of a brain contains a tumor or not using machine learning. Once detected, it will then classify the type of tumor as either benign or malignant. In this work, the system correctly classifies MRI images into images with tumor and images without tumor. This has to be done with no human intervention. In order to apply several types of classifiers, there is a need to pre-process several aspects of the images such as the color, image file extension, area of interest and contrast level.

2.2 Summary/Gaps identified in the Survey

From the above survey, it can be summarised that there have been numerous advancements and researches in the medical industry pertaining to brain tumor detection till date. Various types of machine learning algorithms, artificial intelligence techniques and image segmentation techniques have been involved in this research. Algorithms like Random Forest, Support Vector Machine and hybrid genetic algorithms have been involved. Image mining techniques, Simple Linear Iterative Clustering for segmentation and region merging methods have been followed as far as image processing is concerned.

However, in the above research, it has come to our notice that the accuracy with which the tumor is detected is usually only up to 85 or 86%. This often does not guarantee the efficiency and reliability of detecting the tumor. Therefore, what we, in our proposed work have taken care of is that the accuracy should be as much as possible, i.e 91% in our case. This is due to our implementation concerning the watershed segmentation technique applied on the MRI scan, and later processed by the trained CNN model that brings out this efficiency and accuracy.

3. Proposed System Requirements Analysis and Design

3.1. Introduction

Requirement Analysis and proposal of the complete detailed design of the software before starting its implementation is extremely crucial for successful results and long-lasting quality of the system [9] . These specifications help the developers as well as the users to deeply understand the fundamental needs of the project and helps the developers deliver the software as per the expectations of the clients and users. Requirements basically include noting down the hardware, software, system, user functional and non-functional requirements of the system. It is important to identify the stakeholders and the end-users as well to build the software keeping their expectations and needs in mind.Work-breakdown structure, including the verb-based WBS, noun-based WBS and time based WBS also plays a major role in building the system efficiently with all the features involved.

3.2. Requirement Analysis

3.2.1. Stakeholder Identification

A Stakeholder is a person or a legal entity in a development project that can influence the requirements of the system. The aim of this project is to research, design, develop and evaluate an intelligent system to detect cerebral auto-regulation and brain damages after traumatic head injury. The following stakeholders have been identified in the system.

- Doctors Patients
- Medical Students
- Neurologists
- Radiologists
- Hospital Board of Directors
- CEO/ Shareholder
- Government
- Ministry of Health
- Competitors (Existing products)
- Developer
- Project Supervisor

Stakeholders	**Point of View**
Doctors	This stakeholder wants the system to be functional
Medical Students	Wants to analyse MRI images to improve their academic and clinical skills
Neurologists and Radiologists	This stakeholder wants the system to be functional
Patients	This stakeholder wants to get medical treatments to get into better condition
CEO/Shareholder	Wants it to be good return on investment and to obtain profits
Ministry of Health and Government	Wants system to comply with rules and regulations / Want system to be run effectively and safely
Developer	Wants to market the developed product
Project Supervisor	This stakeholder wants to guide the development of the system

Table 3.2.1.1 : Stakeholders' point of view

3.2.2. Functional Requirements

This includes the services the system should provide or how the system should react to particular problems.

1. To be able to upload an image consisting of Brain MRI Scan
2. To get the final segmented image of the brain scan, depicting the tumor if any
3. To check and display whether the input image is detected with tumor or not i.e positive or negative result

3.2.3. Non Functional Requirements

It describes how the system should work. The Non-Functional Requirements include

- Usability requirement : The software should be very easy to use. The patient only has to give an input of the MRI scan, and if a tumour is present the software will provide a clear image of the tumour.

- Serviceability requirement : It will be made sure that the system provides the services that it is built for efficiently, and that the demands of the end-user to obtain the results from the software are fulfilled.

- Manageability requirement : The administrators of the system would be able to easily monitor and have the ability to debug and analyse the failures if any and make sure that the system is running up-to-the mark.

- Recoverability requirement : The system should be able to recover the images if lost, due to force shut down of the software or the device.

- Reliability requirement : Problem solving must be made sure that the system is reliable in its operations and for noting the sensitive details. Our project gives a clear image of the patient's current state whether they have a tumour or not. The patient will not have to run here and there for a number of tests for the Brain Tumour Detection. They can rely on our product.

- Maintainability requirement : The software shall be updated and maintained regularly, checked for the quality and improved functionality if needed to be added.

- Availability requirement : Our software will be easily available to every hospital where there is a huge problem related to Brain Scan. It will be uploaded online so that people can download and can be used in their houses too.

3.2.4. System Requirements

3.2.4.1. H/W Requirements

- Processor: i5-8250U CPU @ 1.60GHz Intel(R) Core(TM)
- RAM: 8.00 GB
- Operating System: Windows
- System Type: x64 based processor, 64-bit Operating System

3.2.4.2. S/W Requirements

- IDE
    - Google Colab
- OpenCV
- Python
- NumPy
- Flask
- Frontend Technologies :
    - HTML
    - CSS

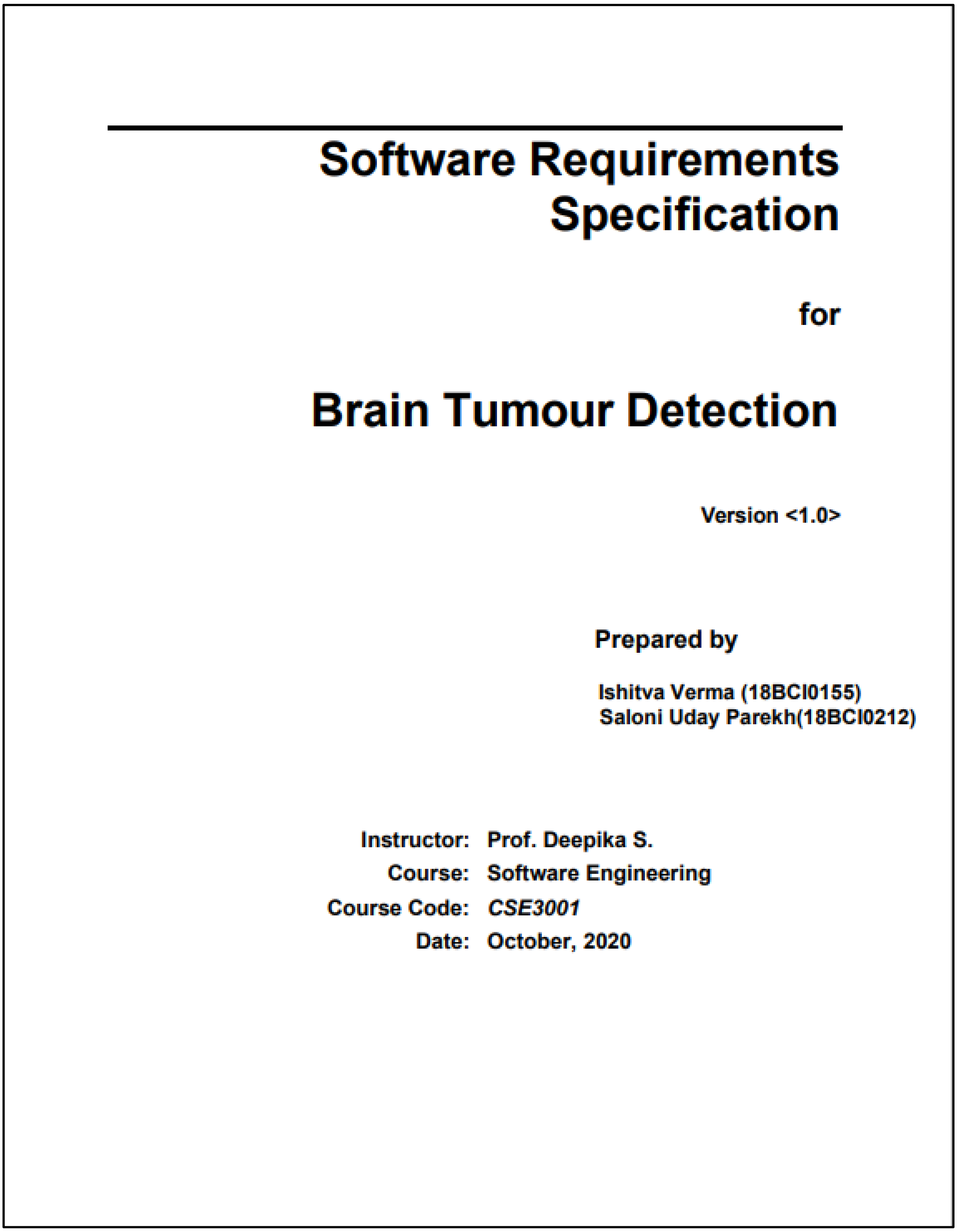
3.2.5. User Requirements

- Uploading an MRI Scan Image
- Getting the result shown as Tumor Detected - Positive or Negative

3.2.6. Major Functions of the System

- Pre-processing of the Brain MRI image
- Segmentation using K-means clustering algorithm
- Feature Extraction using GLCM
- Feature Reduction
- Image Recognition and compression – using PCA
- Canny Edge Detection of image to detect its edges
- Addition of this image on thresholded output
- Stage I Classification – Normal or Abnormal using PNN
- Stage II Classification – Benign or Malignant Tumour
- Morphological Operations – Erosion and Dilation
- Training of CNN Model
- Predictions from CNN Model

3.2.7. Software Requirement Specification document



Software Requirements Specification

for

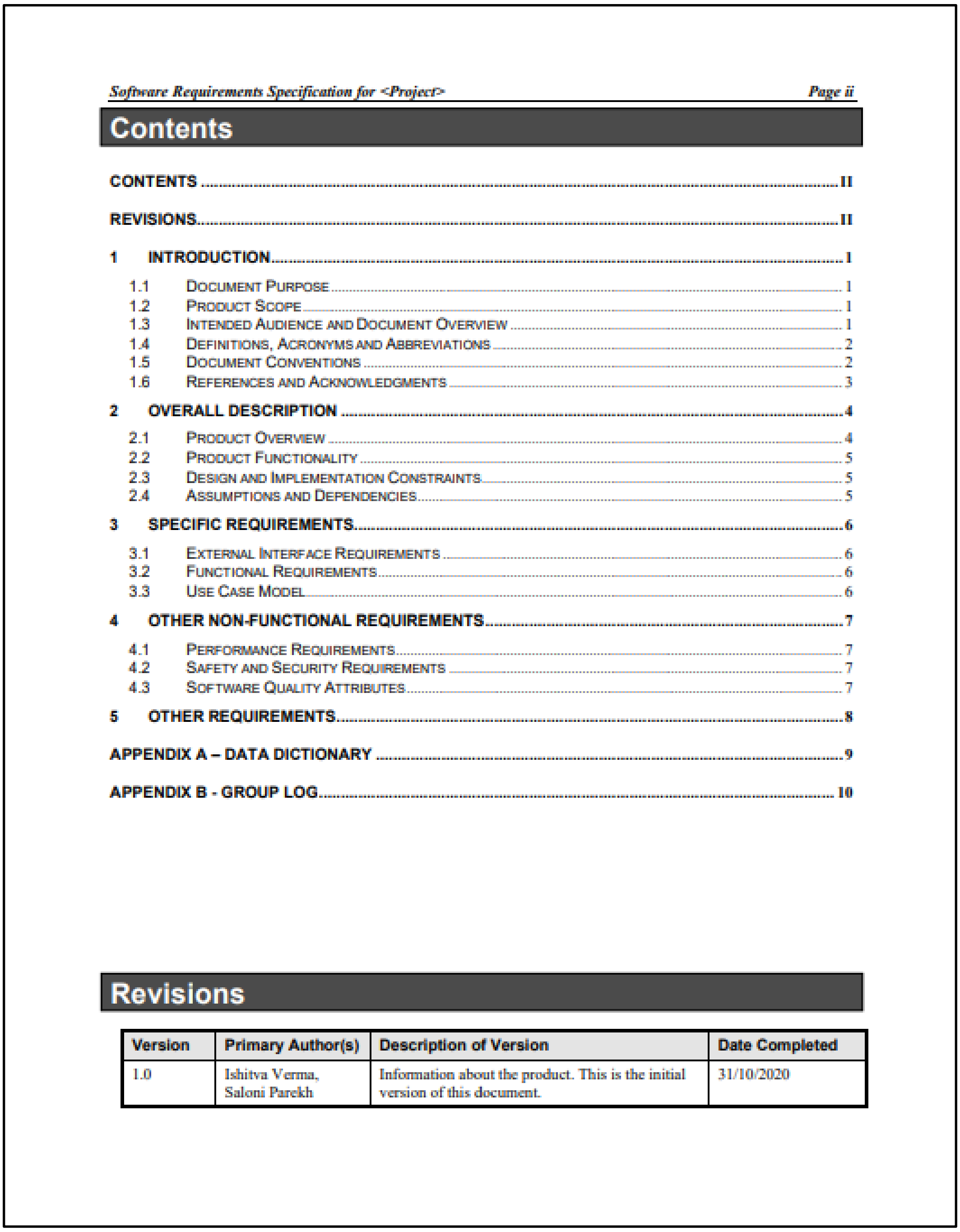
Brain Tumour Detection

Version <1.0>

Prepared by

Ishitva Verma (18BCI0155)
Saloni Uday Parekh(18BCI0212)

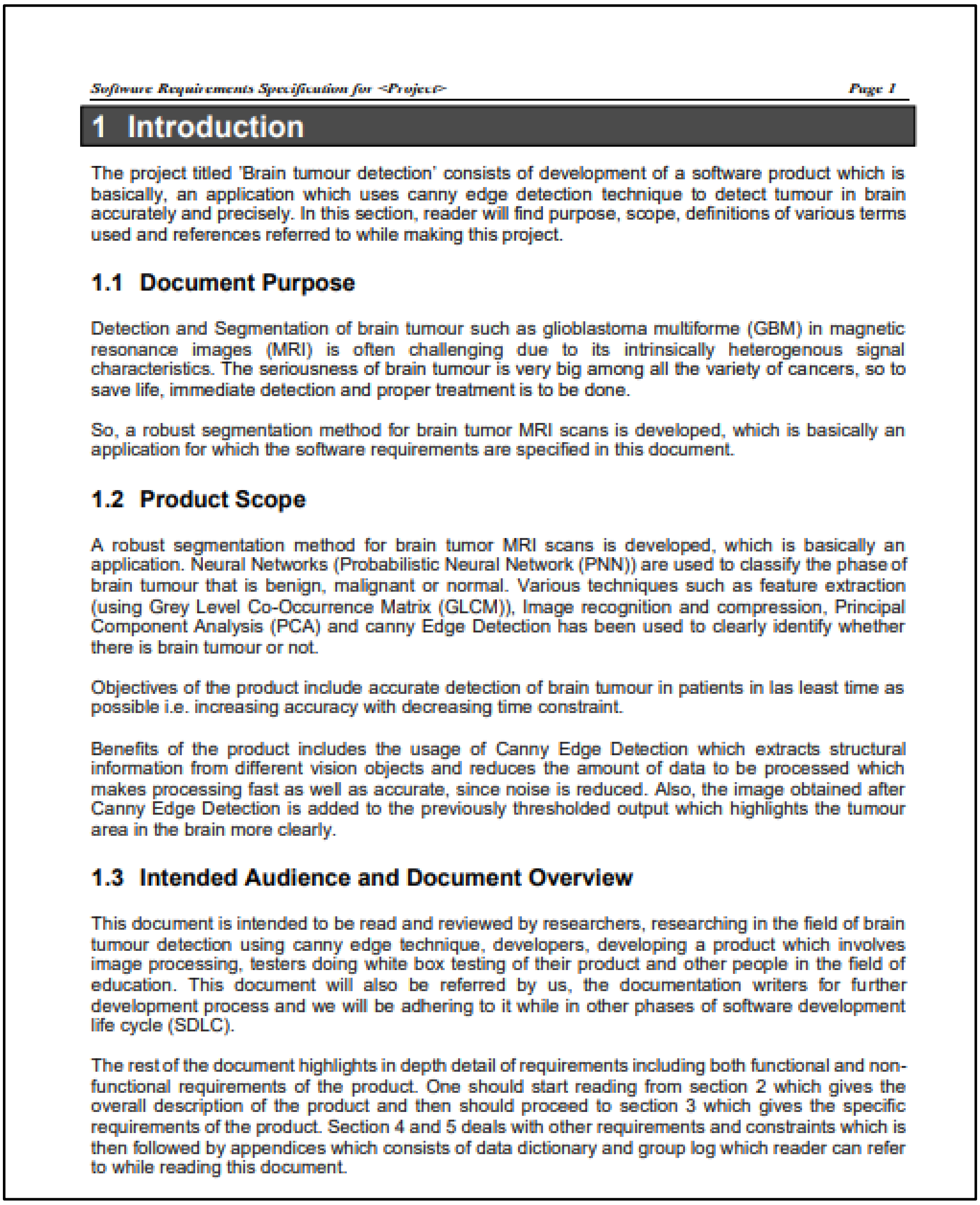
Instructor:	Prof. Deepika S.
Course:	Software Engineering
Course Code:	*CSE3001*
Date:	October, 2020



Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Ishitva Verma, Saloni Parekh	Information about the product. This is the initial version of this document.	31/10/2020



1 Introduction

The project titled 'Brain tumour detection' consists of development of a software product which is basically, an application which uses canny edge detection technique to detect tumour in brain accurately and precisely. In this section, reader will find purpose, scope, definitions of various terms used and references referred to while making this project.

1.1 Document Purpose

Detection and Segmentation of brain tumour such as glioblastoma multiforme (GBM) in magnetic resonance images (MRI) is often challenging due to its intrinsically heterogenous signal characteristics. The seriousness of brain tumour is very big among all the variety of cancers, so to save life, immediate detection and proper treatment is to be done.

So, a robust segmentation method for brain tumor MRI scans is developed, which is basically an application for which the software requirements are specified in this document.

1.2 Product Scope

A robust segmentation method for brain tumor MRI scans is developed, which is basically an application. Neural Networks (Probabilistic Neural Network (PNN)) are used to classify the phase of brain tumour that is benign, malignant or normal. Various techniques such as feature extraction (using Grey Level Co-Occurrence Matrix (GLCM)), Image recognition and compression, Principal Component Analysis (PCA) and canny Edge Detection has been used to clearly identify whether there is brain tumour or not.

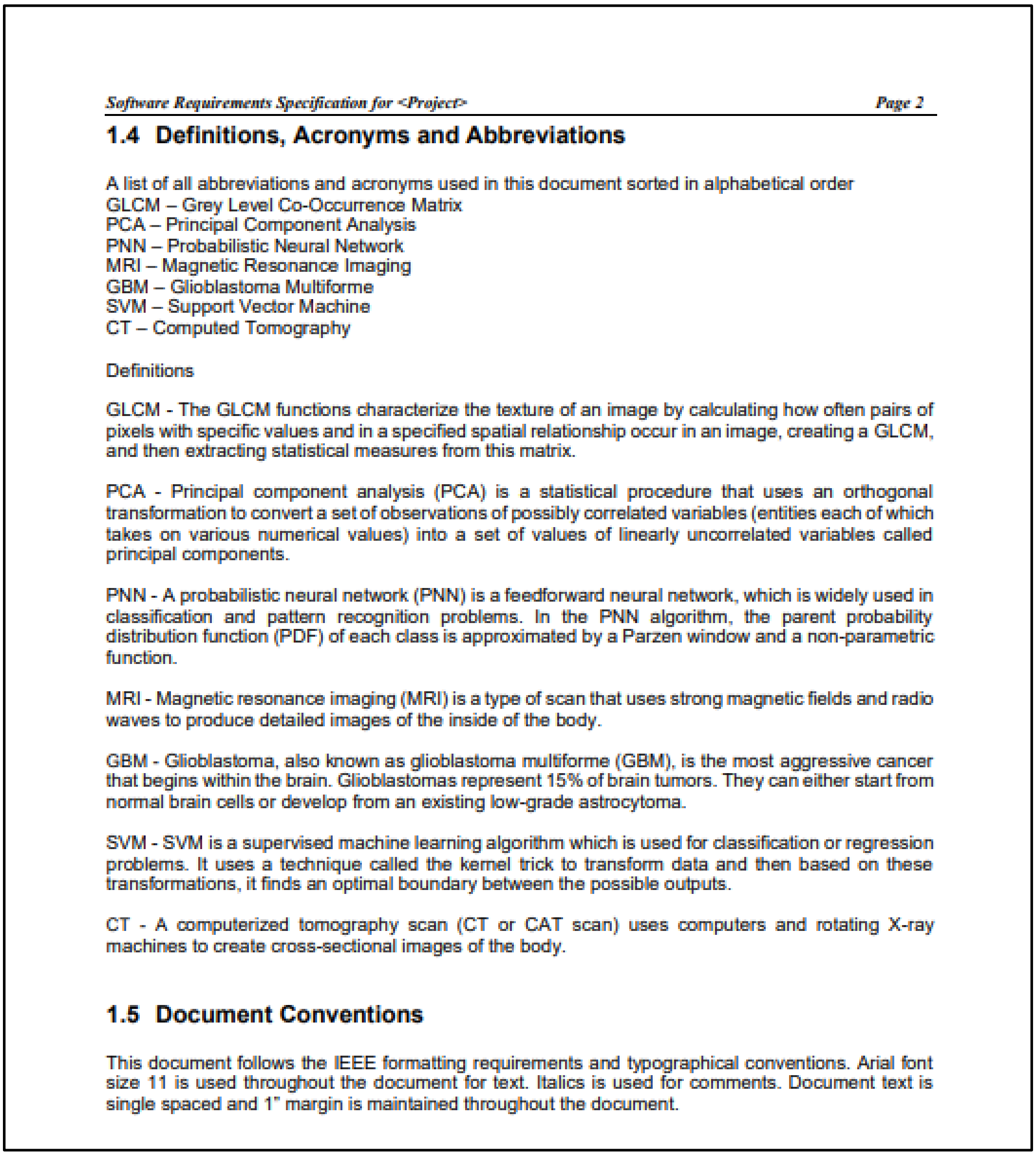
Objectives of the product include accurate detection of brain tumour in patients in las least time as possible i.e. increasing accuracy with decreasing time constraint.

Benefits of the product includes the usage of Canny Edge Detection which extracts structural information from different vision objects and reduces the amount of data to be processed which makes processing fast as well as accurate, since noise is reduced. Also, the image obtained after Canny Edge Detection is added to the previously thresholded output which highlights the tumour area in the brain more clearly.

1.3 Intended Audience and Document Overview

This document is intended to be read and reviewed by researchers, researching in the field of brain tumour detection using canny edge technique, developers, developing a product which involves image processing, testers doing white box testing of their product and other people in the field of education. This document will also be referred by us, the documentation writers for further development process and we will be adhering to it while in other phases of software development life cycle (SDLC).

The rest of the document highlights in depth detail of requirements including both functional and non-functional requirements of the product. One should start reading from section 2 which gives the overall description of the product and then should proceed to section 3 which gives the specific requirements of the product. Section 4 and 5 deals with other requirements and constraints which is then followed by appendices which consists of data dictionary and group log which reader can refer to while reading this document.



1.4 Definitions, Acronyms and Abbreviations

A list of all abbreviations and acronyms used in this document sorted in alphabetical order
GLCM – Grey Level Co-Occurrence Matrix
PCA – Principal Component Analysis
PNN – Probabilistic Neural Network
MRI – Magnetic Resonance Imaging
GBM – Glioblastoma Multiforme
SVM – Support Vector Machine
CT – Computed Tomography

Definitions

GLCM - The GLCM functions characterize the texture of an image by calculating how often pairs of pixels with specific values and in a specified spatial relationship occur in an image, creating a GLCM, and then extracting statistical measures from this matrix.

PCA - Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called principal components.

PNN - A probabilistic neural network (PNN) is a feedforward neural network, which is widely used in classification and pattern recognition problems. In the PNN algorithm, the parent probability distribution function (PDF) of each class is approximated by a Parzen window and a non-parametric function.

MRI - Magnetic resonance imaging (MRI) is a type of scan that uses strong magnetic fields and radio waves to produce detailed images of the inside of the body.

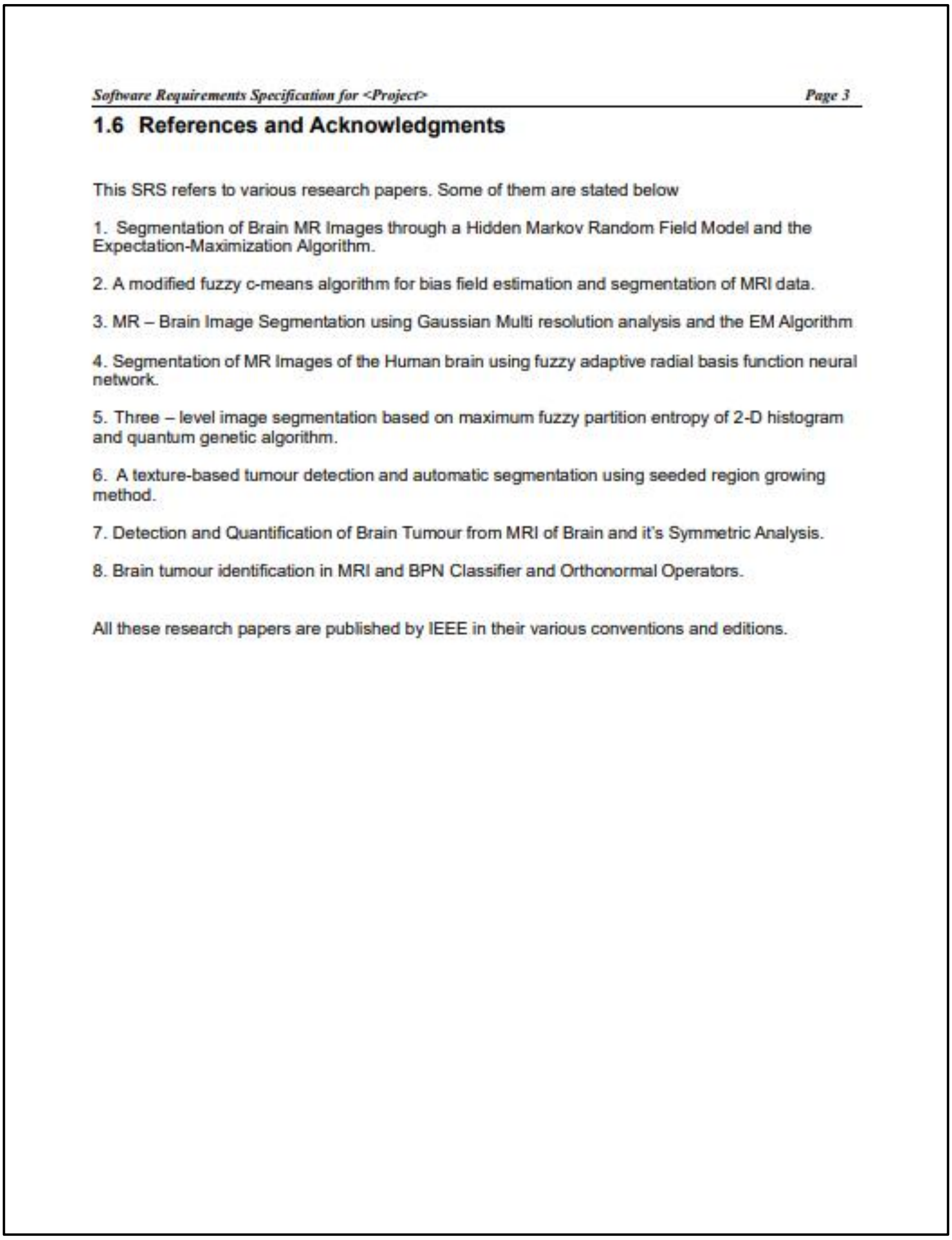
GBM - Glioblastoma, also known as glioblastoma multiforme (GBM), is the most aggressive cancer that begins within the brain. Glioblastomas represent 15% of brain tumors. They can either start from normal brain cells or develop from an existing low-grade astrocytoma.

SVM - SVM is a supervised machine learning algorithm which is used for classification or regression problems. It uses a technique called the kernel trick to transform data and then based on these transformations, it finds an optimal boundary between the possible outputs.

CT - A computerized tomography scan (CT or CAT scan) uses computers and rotating X-ray machines to create cross-sectional images of the body.

1.5 Document Conventions

This document follows the IEEE formatting requirements and typographical conventions. Arial font size 11 is used throughout the document for text. Italics is used for comments. Document text is single spaced and 1" margin is maintained throughout the document.



1.6 References and Acknowledgments

This SRS refers to various research papers. Some of them are stated below

1. Segmentation of Brain MR Images through a Hidden Markov Random Field Model and the Expectation-Maximization Algorithm.

2. A modified fuzzy c-means algorithm for bias field estimation and segmentation of MRI data.

3. MR – Brain Image Segmentation using Gaussian Multi resolution analysis and the EM Algorithm

4. Segmentation of MR Images of the Human brain using fuzzy adaptive radial basis function neural network.

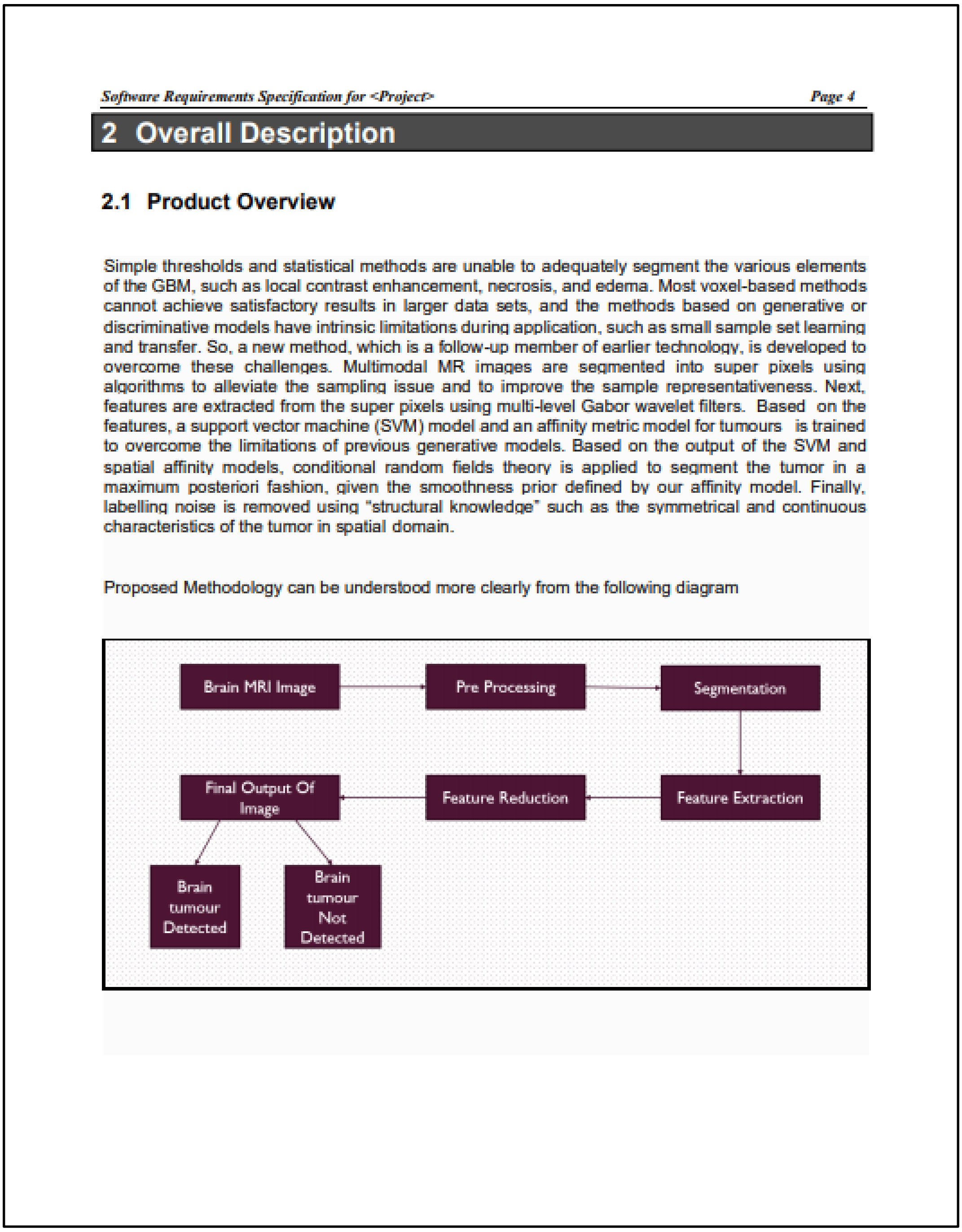
5. Three – level image segmentation based on maximum fuzzy partition entropy of 2-D histogram and quantum genetic algorithm.

6. A texture-based tumour detection and automatic segmentation using seeded region growing method.

7. Detection and Quantification of Brain Tumour from MRI of Brain and it's Symmetric Analysis.

8. Brain tumour identification in MRI and BPN Classifier and Orthonormal Operators.

All these research papers are published by IEEE in their various conventions and editions.

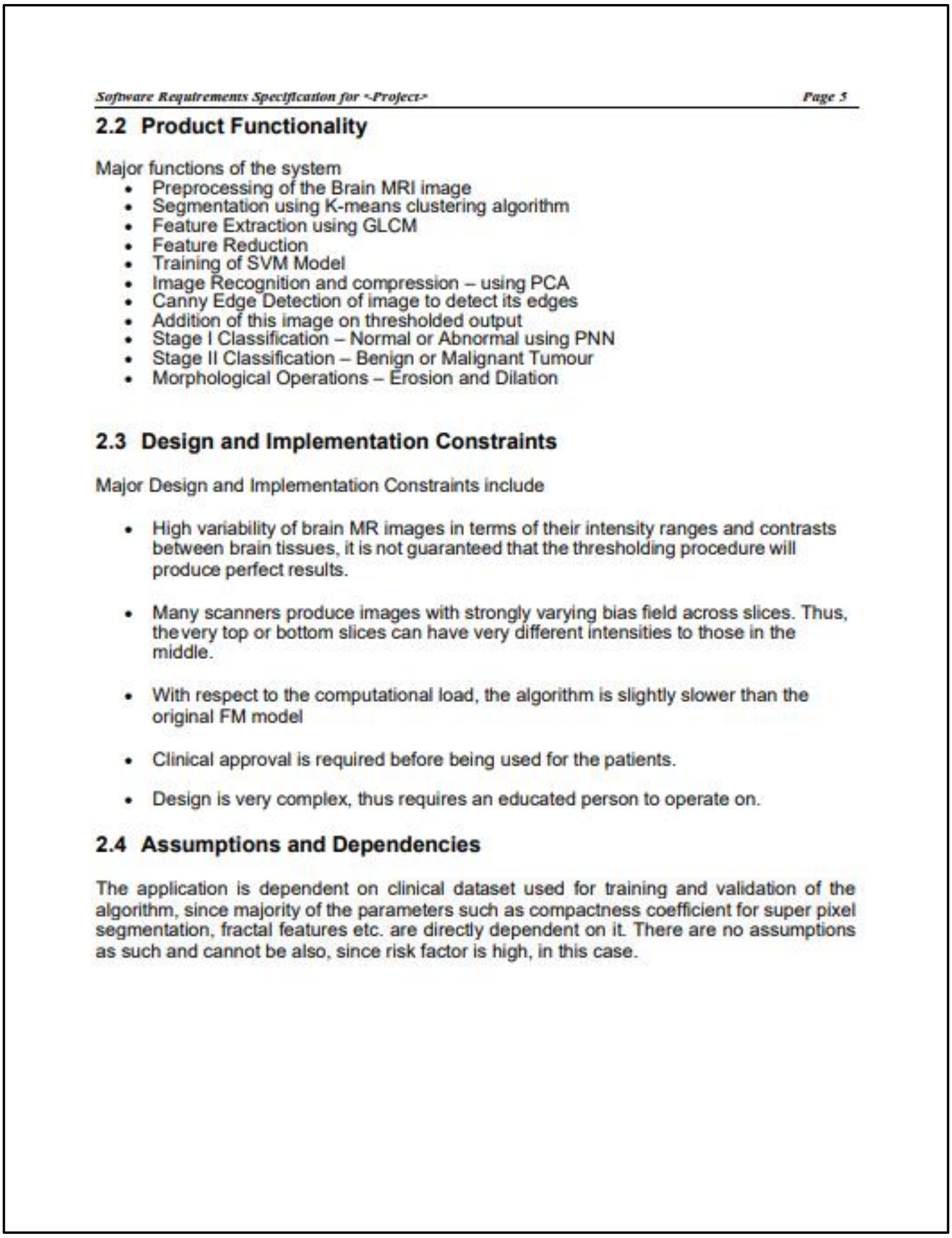


2 Overall Description

2.1 Product Overview

Simple thresholds and statistical methods are unable to adequately segment the various elements of the GBM, such as local contrast enhancement, necrosis, and edema. Most voxel-based methods cannot achieve satisfactory results in larger data sets, and the methods based on generative or discriminative models have intrinsic limitations during application, such as small sample set learning and transfer. So, a new method, which is a follow-up member of earlier technology, is developed to overcome these challenges. Multimodal MR images are segmented into super pixels using algorithms to alleviate the sampling issue and to improve the sample representativeness. Next, features are extracted from the super pixels using multi-level Gabor wavelet filters. Based on the features, a support vector machine (SVM) model and an affinity metric model for tumours is trained to overcome the limitations of previous generative models. Based on the output of the SVM and spatial affinity models, conditional random fields theory is applied to segment the tumor in a maximum posteriori fashion, given the smoothness prior defined by our affinity model. Finally, labelling noise is removed using "structural knowledge" such as the symmetrical and continuous characteristics of the tumor in spatial domain.

Proposed Methodology can be understood more clearly from the following diagram



2.2 Product Functionality

Major functions of the system

- Preprocessing of the Brain MRI image
- Segmentation using K-means clustering algorithm
- Feature Extraction using GLCM
- Feature Reduction
- Training of SVM Model
- Image Recognition and compression – using PCA
- Canny Edge Detection of image to detect its edges
- Addition of this image on thresholded output
- Stage I Classification – Normal or Abnormal using PNN
- Stage II Classification – Benign or Malignant Tumour
- Morphological Operations – Erosion and Dilation

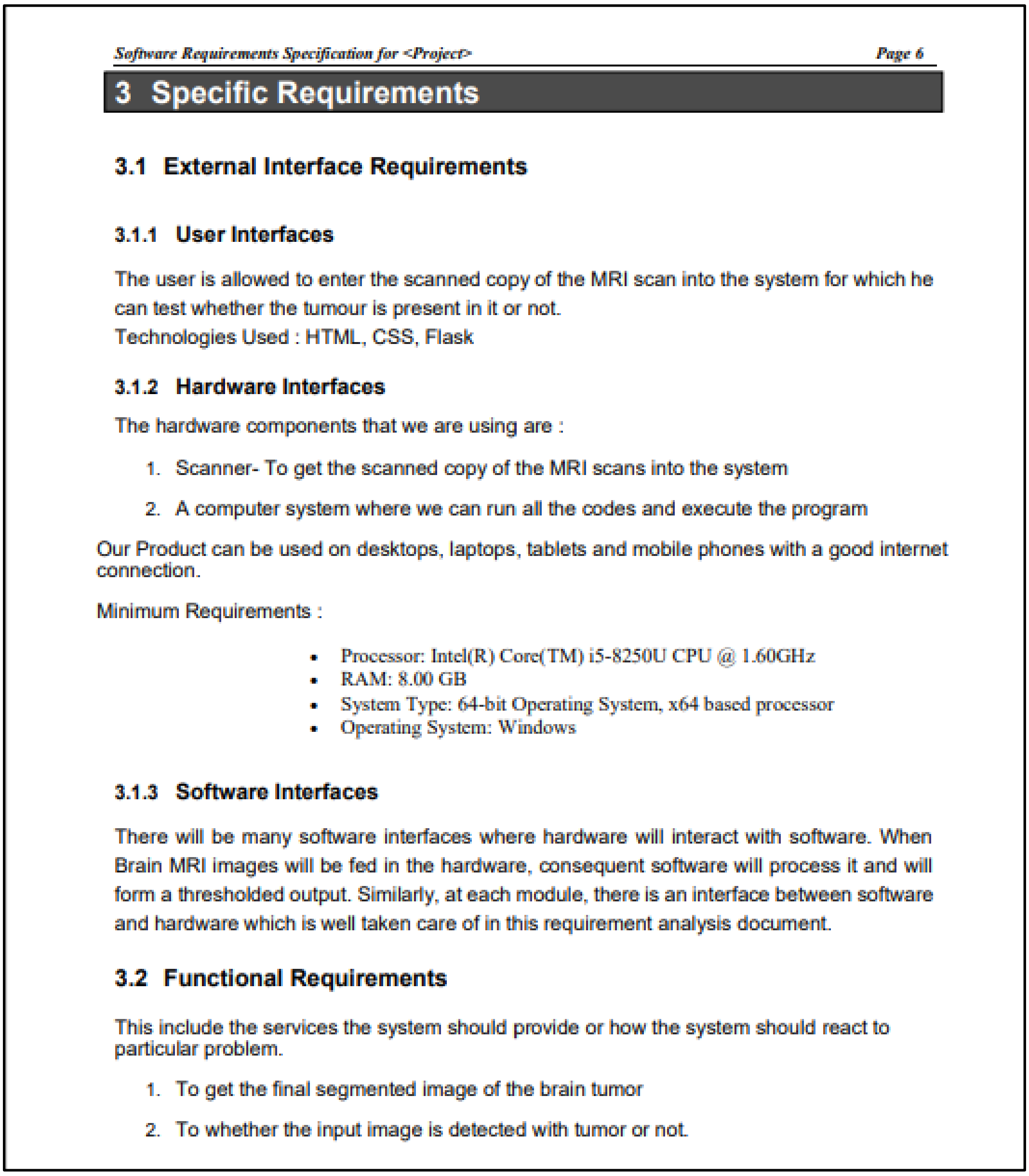
2.3 Design and Implementation Constraints

Major Design and Implementation Constraints include

- High variability of brain MR images in terms of their intensity ranges and contrasts between brain tissues, it is not guaranteed that the thresholding procedure will produce perfect results.
- Many scanners produce images with strongly varying bias field across slices. Thus, thevery top or bottom slices can have very different intensities to those in the middle.
- With respect to the computational load, the algorithm is slightly slower than the original FM model
- Clinical approval is required before being used for the patients.
- Design is very complex, thus requires an educated person to operate on.

2.4 Assumptions and Dependencies

The application is dependent on clinical dataset used for training and validation of the algorithm, since majority of the parameters such as compactness coefficient for super pixel segmentation, fractal features etc. are directly dependent on it. There are no assumptions as such and cannot be also, since risk factor is high, in this case.



3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The user is allowed to enter the scanned copy of the MRI scan into the system for which he can test whether the tumour is present in it or not.
Technologies Used : HTML, CSS, Flask

3.1.2 Hardware Interfaces

The hardware components that we are using are :

1. Scanner- To get the scanned copy of the MRI scans into the system
2. A computer system where we can run all the codes and execute the program

Our Product can be used on desktops, laptops, tablets and mobile phones with a good internet connection.

Minimum Requirements :

- Processor: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
- RAM: 8.00 GB
- System Type: 64-bit Operating System, x64 based processor
- Operating System: Windows

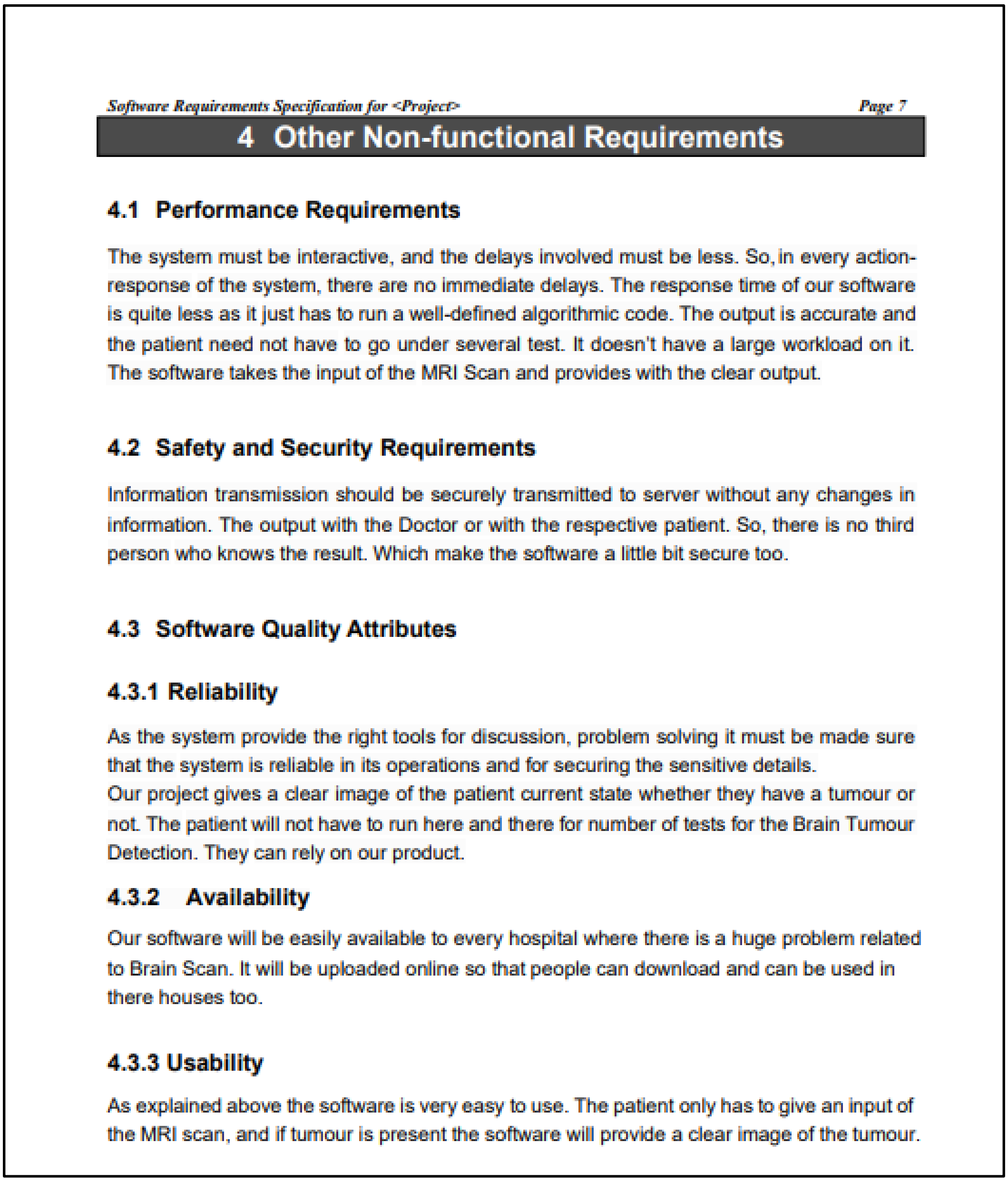
3.1.3 Software Interfaces

There will be many software interfaces where hardware will interact with software. When Brain MRI images will be fed in the hardware, consequent software will process it and will form a thresholded output. Similarly, at each module, there is an interface between software and hardware which is well taken care of in this requirement analysis document.

3.2 Functional Requirements

This include the services the system should provide or how the system should react to particular problem.

1. To get the final segmented image of the brain tumor
2. To whether the input image is detected with tumor or not.



4 Other Non-functional Requirements

4.1 Performance Requirements

The system must be interactive, and the delays involved must be less. So, in every action-response of the system, there are no immediate delays. The response time of our software is quite less as it just has to run a well-defined algorithmic code. The output is accurate and the patient need not have to go under several test. It doesn't have a large workload on it. The software takes the input of the MRI Scan and provides with the clear output.

4.2 Safety and Security Requirements

Information transmission should be securely transmitted to server without any changes in information. The output with the Doctor or with the respective patient. So, there is no third person who knows the result. Which make the software a little bit secure too.

4.3 Software Quality Attributes

4.3.1 Reliability

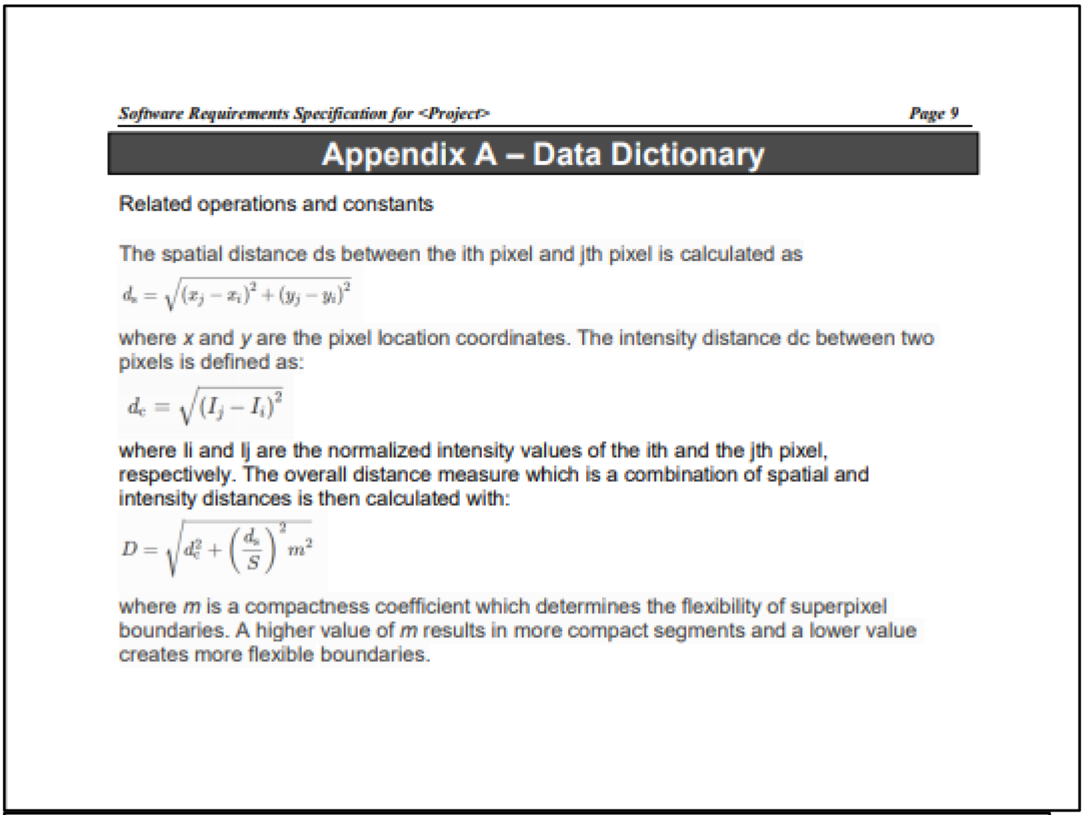
As the system provide the right tools for discussion, problem solving it must be made sure that the system is reliable in its operations and for securing the sensitive details.
Our project gives a clear image of the patient current state whether they have a tumour or not. The patient will not have to run here and there for number of tests for the Brain Tumour Detection. They can rely on our product.

4.3.2 Availability

Our software will be easily available to every hospital where there is a huge problem related to Brain Scan. It will be uploaded online so that people can download and can be used in there houses too.

4.3.3 Usability

As explained above the software is very easy to use. The patient only has to give an input of the MRI scan, and if tumour is present the software will provide a clear image of the tumour.



Software Requirements Specification for <Project> Page 9

Appendix A – Data Dictionary

Related operations and constants

The spatial distance ds between the ith pixel and jth pixel is calculated as

$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$

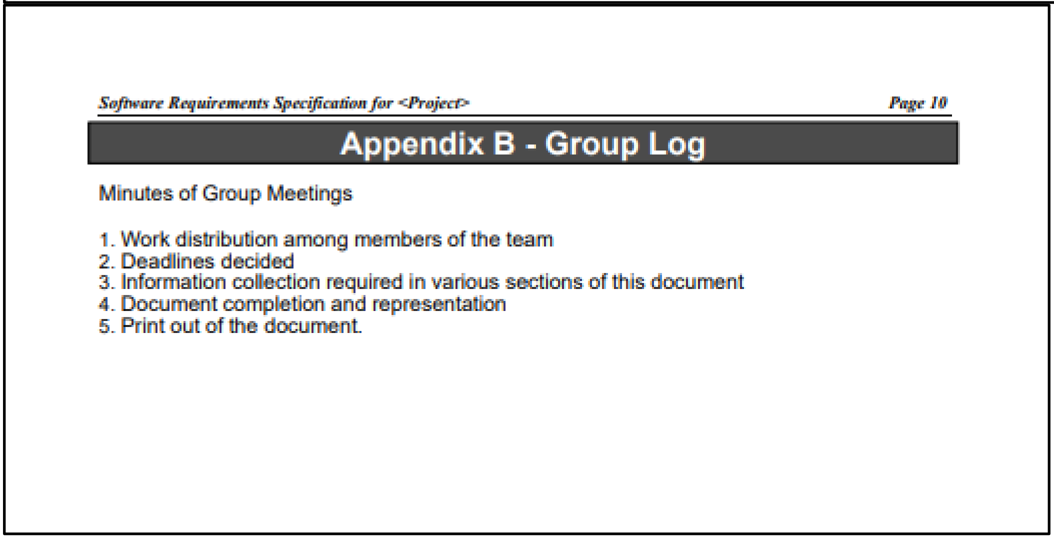
where *x* and *y* are the pixel location coordinates. The intensity distance dc between two pixels is defined as:

$$d_c = \sqrt{(I_j - I_i)^2}$$

where Ii and Ij are the normalized intensity values of the ith and the jth pixel, respectively. The overall distance measure which is a combination of spatial and intensity distances is then calculated with:

$$D = \sqrt{d_c^2 + \left(\frac{d_s}{S}\right)^2 m^2}$$

where *m* is a compactness coefficient which determines the flexibility of superpixel boundaries. A higher value of *m* results in more compact segments and a lower value creates more flexible boundaries.



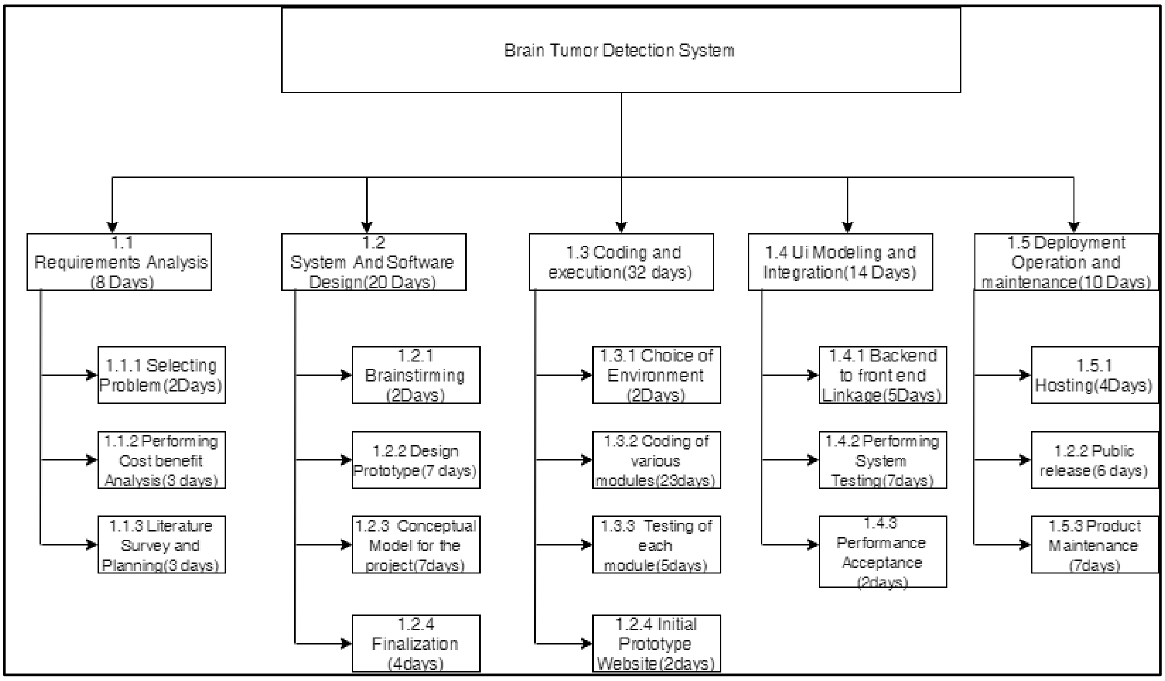
Software Requirements Specification for <Project> Page 10

Appendix B - Group Log

Minutes of Group Meetings

1. Work distribution among members of the team
2. Deadlines decided
3. Information collection required in various sections of this document
4. Document completion and representation
5. Print out of the document.

3.2.8. Work breakdown structure



Brain Tumor Detection System
1.1 Requirements Analysis (8 Days)
1.2 System And Software Design(20 Days)
1.3 Coding and execution(32 days)
1.4 Ui Modeling and Integration(14 Days)
1.5 Deployment Operation and maintenance(10 Days)
1.1.1 Selecting Problem (2Days)
1.1.2 Performing Cost benefit Analysis(3 days)
1.1.3 Literature Survey and Planning(3 days)
1.2.1 Brainstirming (2Days)
1.2.2 Design Prototype(7 days)
1.2.3 Conceptual Model for the project(7days)
1.2.4 Finalization (4days)
1.3.1 Choice of Environment (2Days)
1.3.2 Coding of various modules(23days)
1.3.3 Testing of each module(5days)
1.2.4 Initial Prototype Website(2days)
1.4.1 Backend to front end Linkage(5Days)
1.4.2 Performing System Testing(7days)
1.4.3 Performance Acceptance (2days)
1.5.1 Hosting(4Days)
1.2.2 Public release(6 days)
1.5.3 Product Maintenance (7days)

Fig. 3.2.8.1 Time based work breakdown structure

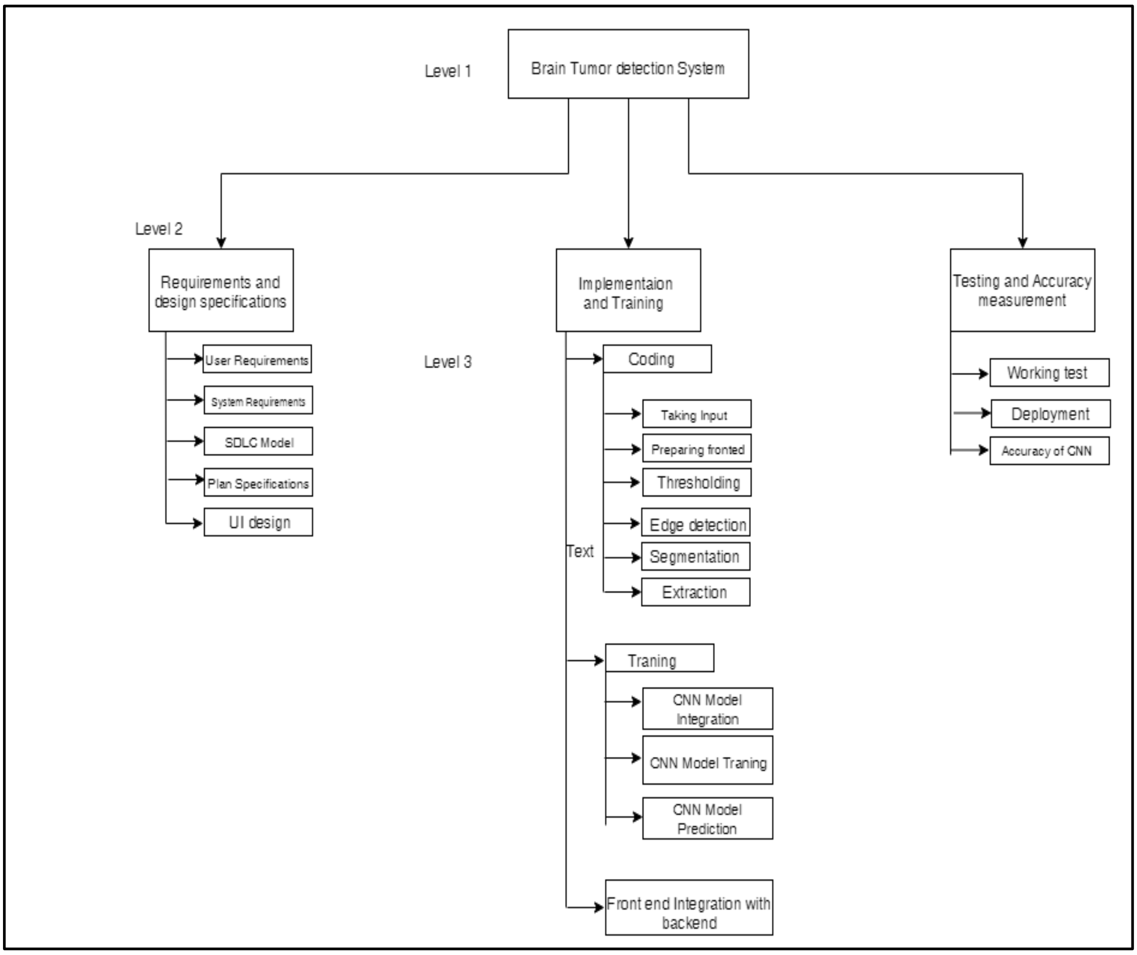


Fig. 3.2.8.2 Verb based work breakdown structure

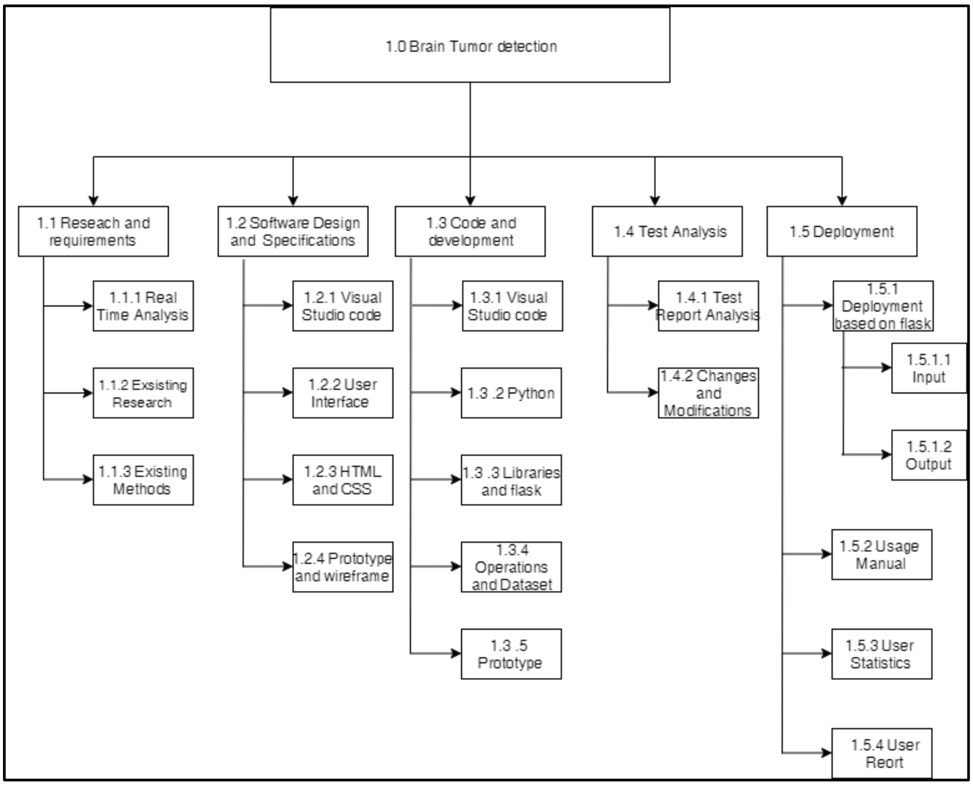


Fig. 3.2.8.3. Noun based work breakdown structure

3.2.9. Pert chart (with task table)

A PERT chart is a tool specially for project management for organizing, coordinating and scheduling tasks within a project. It is a way to identify the minimum time needed to complete the total project, to observe the tasks involved in completing a project and the time needed to complete each task.

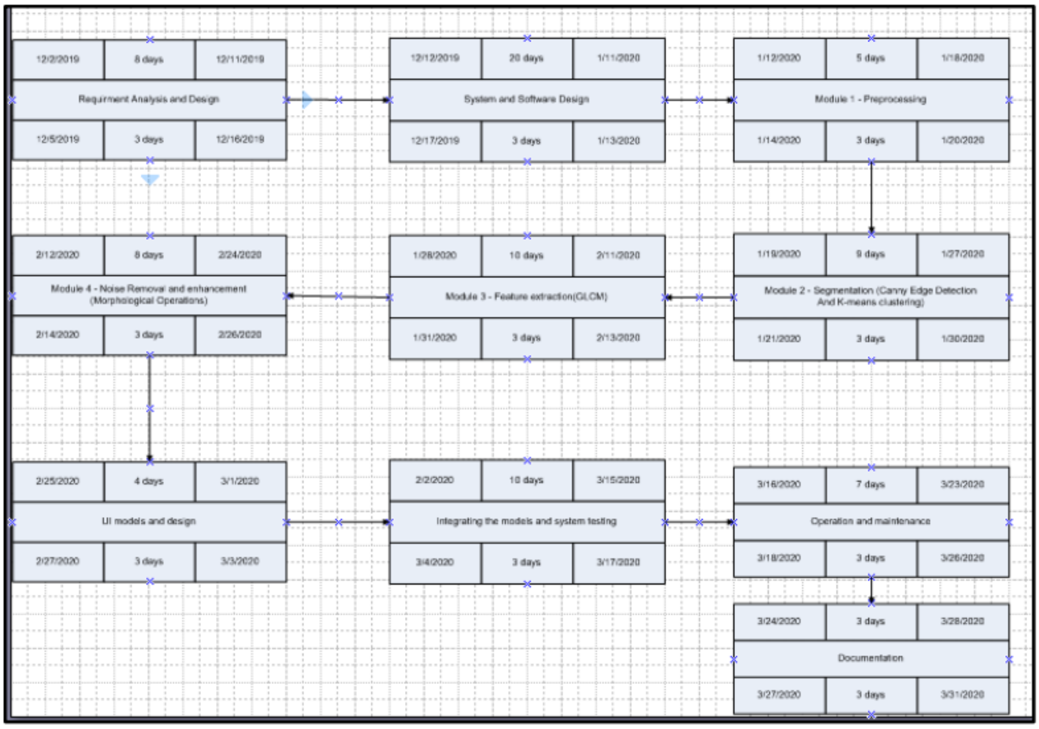
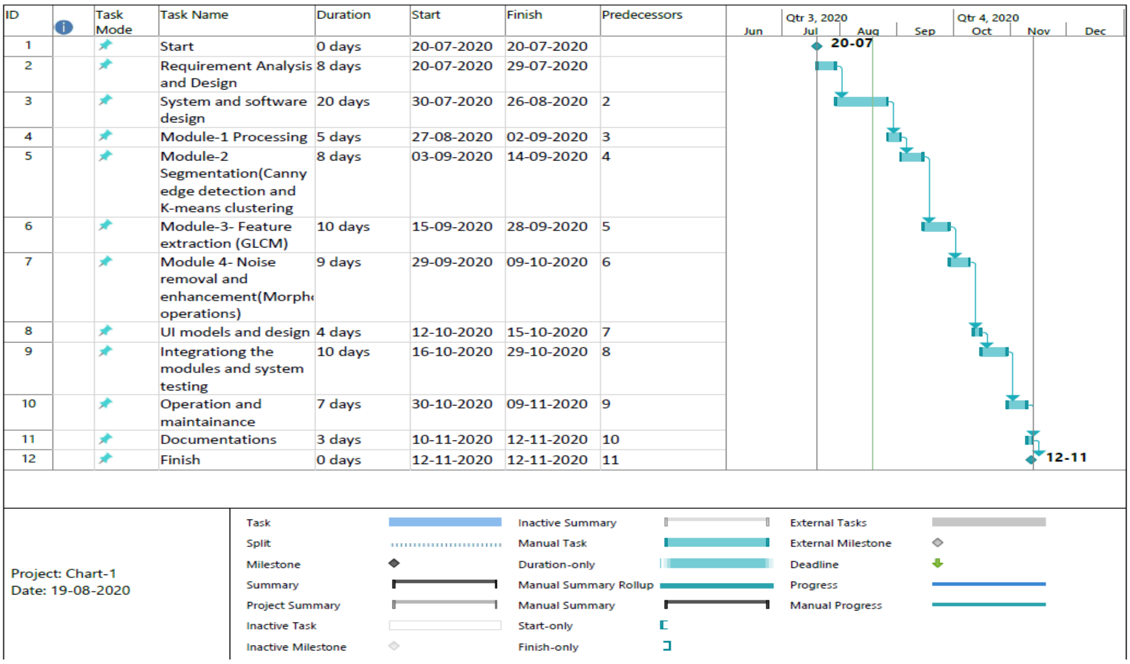


Fig. 3.2.9.1 Pert chart

3.2.10. Gantt Chart

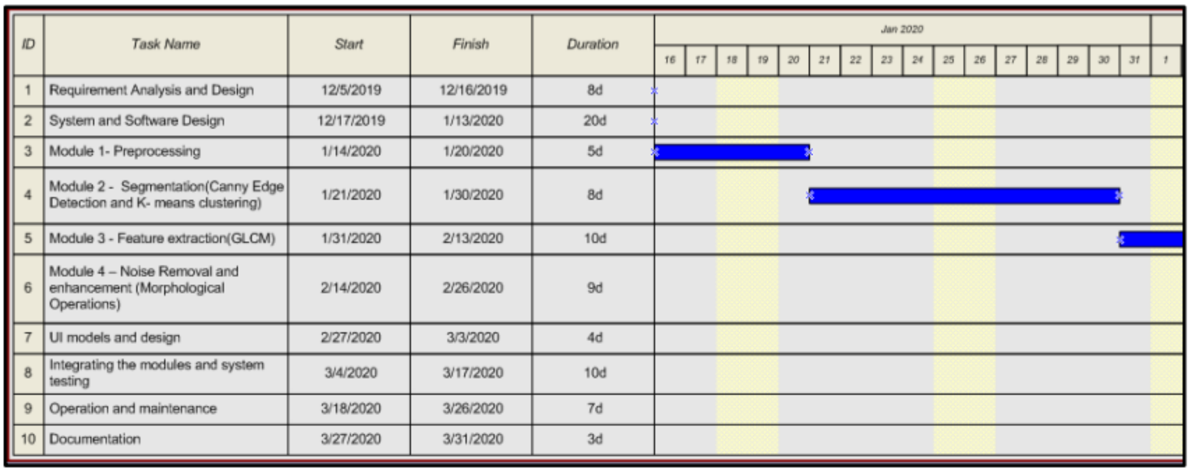
A Gantt chart is a way to view the tasks visually that are scheduled over time. Gantt charts are used for strategizing projects of various sizes and they are a helpful way of denoting what work is scheduled to be done on a specific day. They also enable you to observe the start and end dates of a project in one common view.



ID	ⓘ	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		📌	Start	0 days	20-07-2020	20-07-2020	
2		📌	Requirement Analysis and Design	8 days	20-07-2020	29-07-2020	
3		📌	System and software design	20 days	30-07-2020	26-08-2020	2
4		📌	Module-1 Processing	5 days	27-08-2020	02-09-2020	3
5		📌	Module-2 Segmentation(Canny edge detection and K-means clustering	8 days	03-09-2020	14-09-2020	4
6		📌	Module-3- Feature extraction (GLCM)	10 days	15-09-2020	28-09-2020	5
7		📌	Module 4- Noise removal and enhancement(Morph operations)	9 days	29-09-2020	09-10-2020	6
8		📌	UI models and design	4 days	12-10-2020	15-10-2020	7
9		📌	Integrationg the modules and system testing	10 days	16-10-2020	29-10-2020	8
10		📌	Operation and maintainance	7 days	30-10-2020	09-11-2020	9
11		📌	Documentations	3 days	10-11-2020	12-11-2020	10
12		📌	Finish	0 days	12-11-2020	12-11-2020	11

Project: Chart-1
Date: 19-08-2020

Fig. 3.2.10.1 Gantt chart



ID	Task Name	Start	Finish	Duration
1	Requirement Analysis and Design	12/5/2019	12/16/2019	8d
2	System and Software Design	12/17/2019	1/13/2020	20d
3	Module 1- Preprocessing	1/14/2020	1/20/2020	5d
4	Module 2 - Segmentation(Canny Edge Detection and K- means clustering)	1/21/2020	1/30/2020	8d
5	Module 3 - Feature extraction(GLCM)	1/31/2020	2/13/2020	10d
6	Module 4 – Noise Removal and enhancement (Morphological Operations)	2/14/2020	2/26/2020	9d
7	UI models and design	2/27/2020	3/3/2020	4d
8	Integrating the modules and system testing	3/4/2020	3/17/2020	10d
9	Operation and maintenance	3/18/2020	3/26/2020	7d
10	Documentation	3/27/2020	3/31/2020	3d

Fig. 3.2.10.2 Gantt chart (time based)

4. Design of the Proposed System

4.1. Introduction

Proposal of the complete detailed design of the software plays an important role behind the looks and feel of the system. Design, if decided and discussed beforehand, helps the designers make sure that the overview of the system that they are designing is as per the demands of the customers. The product is known to be a hit when the stakeholders and the end-users are satisfied with the UI and UX of the system and it is user-friendly in all aspects. It is not only concerned with the look and color-scheme of the software but also how well the database is designed as per the requirements. In short, the design requirements involve the user-interface designs demanded by the customer for the frontend and detailed Entity-Relationship and UML diagrams for better understanding of the work behind the scenes.

4.2. High level Design

4.2.1. Architecture design

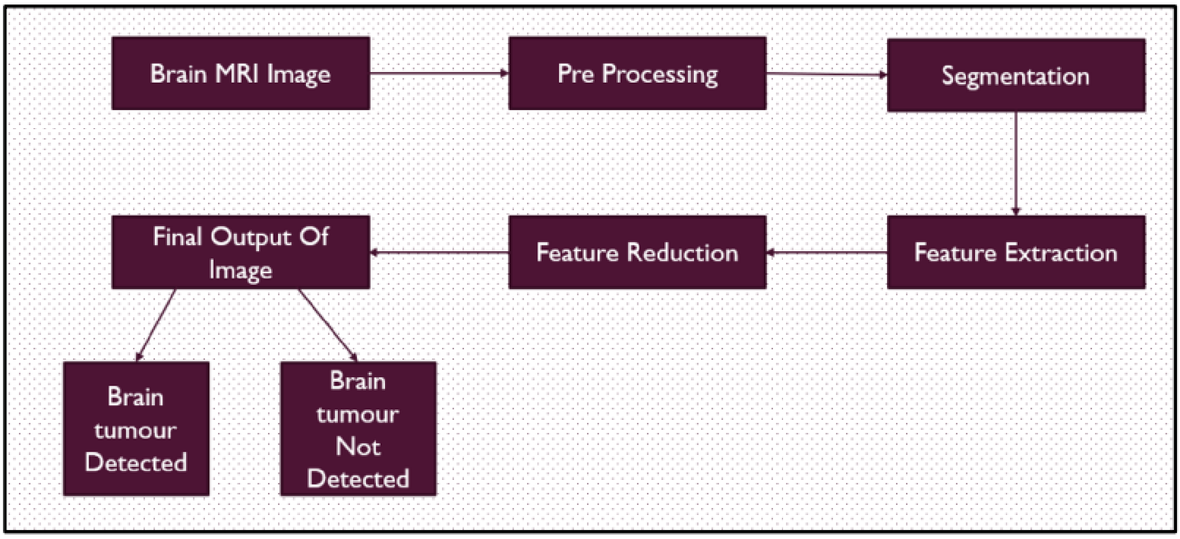


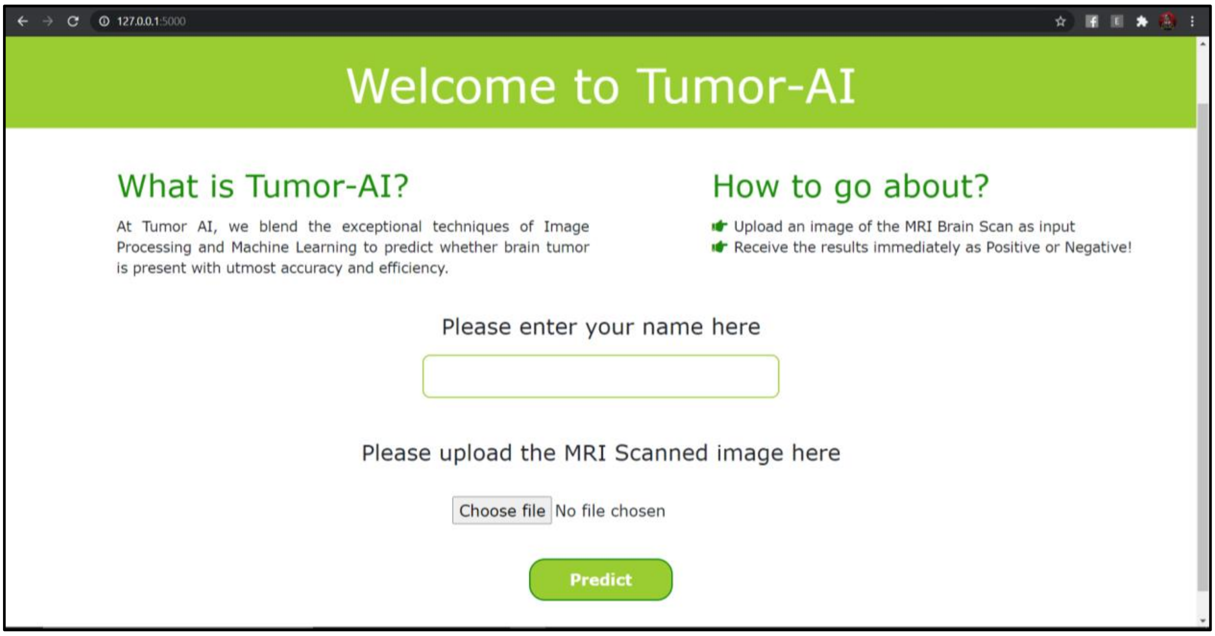
Fig. 4.2.1.1 Architecture design

4.2.2. Explanation of the Architecture diagram

The complete architecture of the software involves five separate and independent modules. After giving a brain MRI image Scan as the input to the software, firstly we pre-process the image. The pre-processing happens with the help of a series of Image processing techniques explained in the documentation. Image watershed segmentation is thereby performed on the pre-processed image for more clarity and distinct prediction.

This image is then taken to the trained CNN model, where the essential and unique features are extracted to perform the desired functionality. The redundant and repeated features are discarded through the feature reduction process. The model predicts the tumor with an accuracy of about 91%. Finally, the watershed segmented image is then displayed as the output which shows the tumor if any. The results are determined and displayed as positive or negative to the end-users.

4.2.3. UI design



Welcome to Tumor-AI

What is Tumor-AI?

At Tumor AI, we blend the exceptional techniques of Image Processing and Machine Learning to predict whether brain tumor is present with utmost accuracy and efficiency.

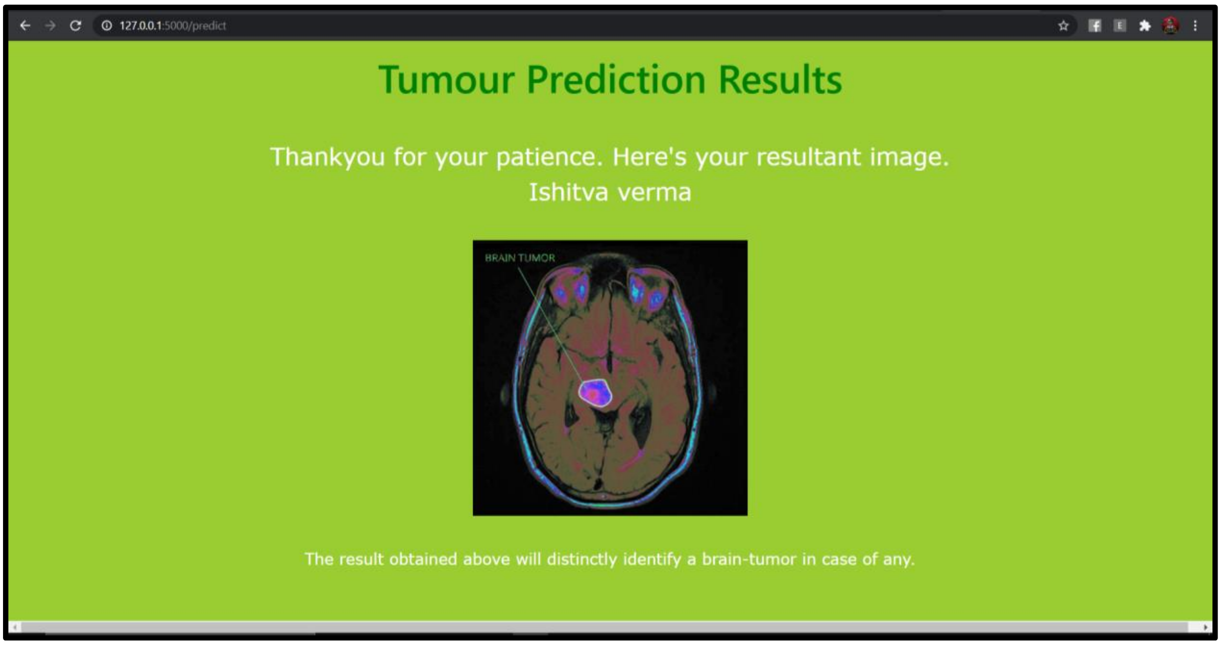
How to go about?

- Upload an image of the MRI Brain Scan as input
- Receive the results immediately as Positive or Negative!

Please enter your name here

Please upload the MRI Scanned image here

Fig. 4.2.3.1 Home Page



Tumour Prediction Results

Thankyou for your patience. Here's your resultant image.
Ishitva verma

The result obtained above will distinctly identify a brain-tumor in case of any.

Fig. 4.2.3.2 Result Page

4.3. Detailed Design (ER Diagram/UML Diagram/Mathematical Modeling)

4.3.1. ER Diagram

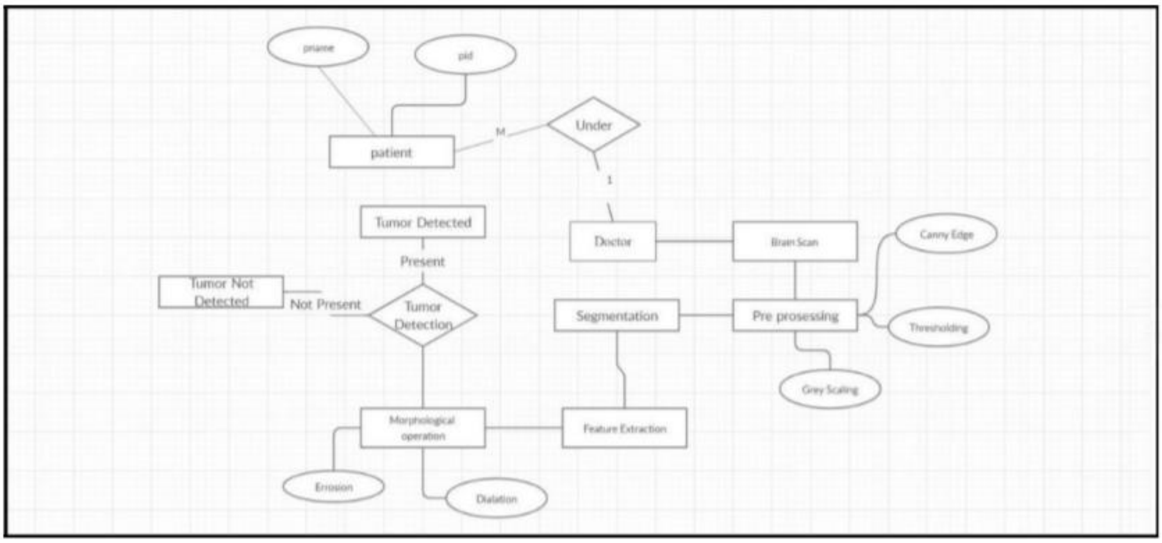


Fig. 4.3.1 ER Diagram

4.3.2. UML diagrams

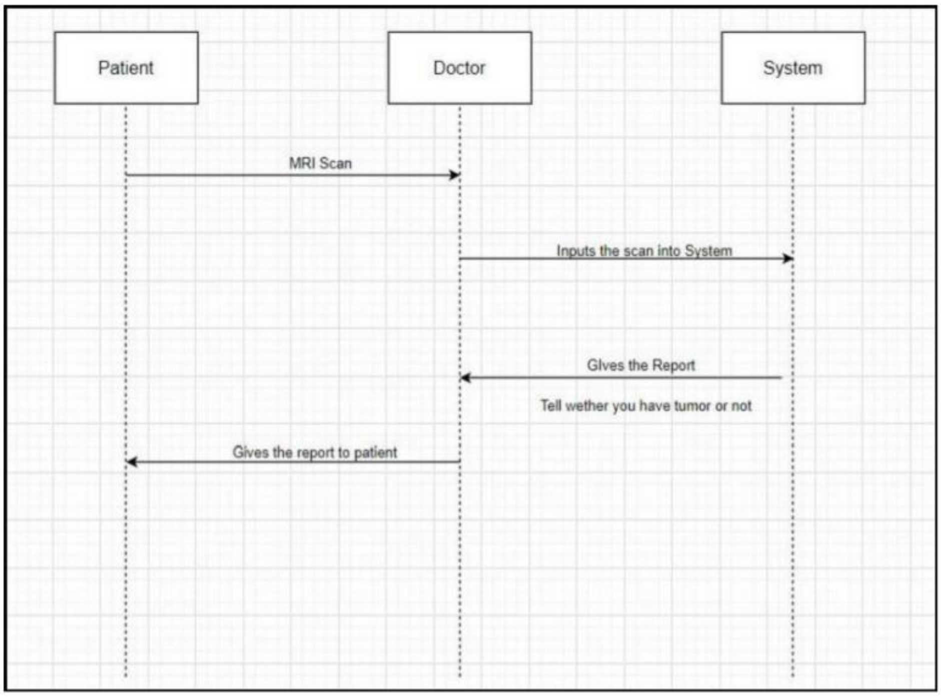

Patient
Doctor
System
MRI Scan
Inputs the scan into System
GIves the Report
Tell wether you have tumor or not
Gives the report to patient

Fig. 4.3.2.1 Sequence Diagram

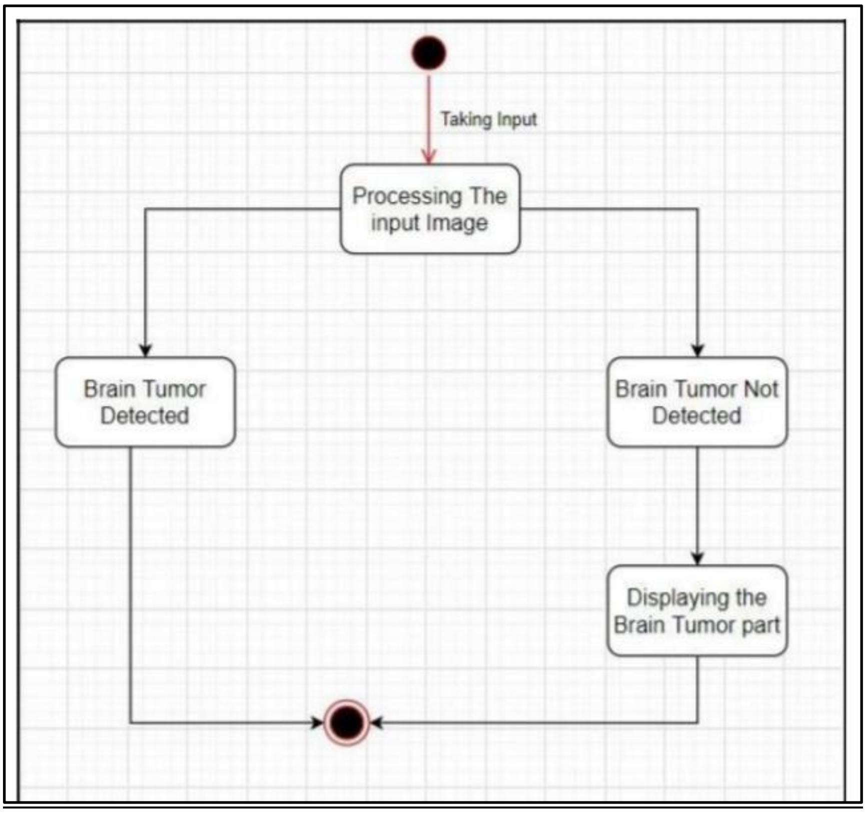


Fig. 4.3.2.2 State Diagram

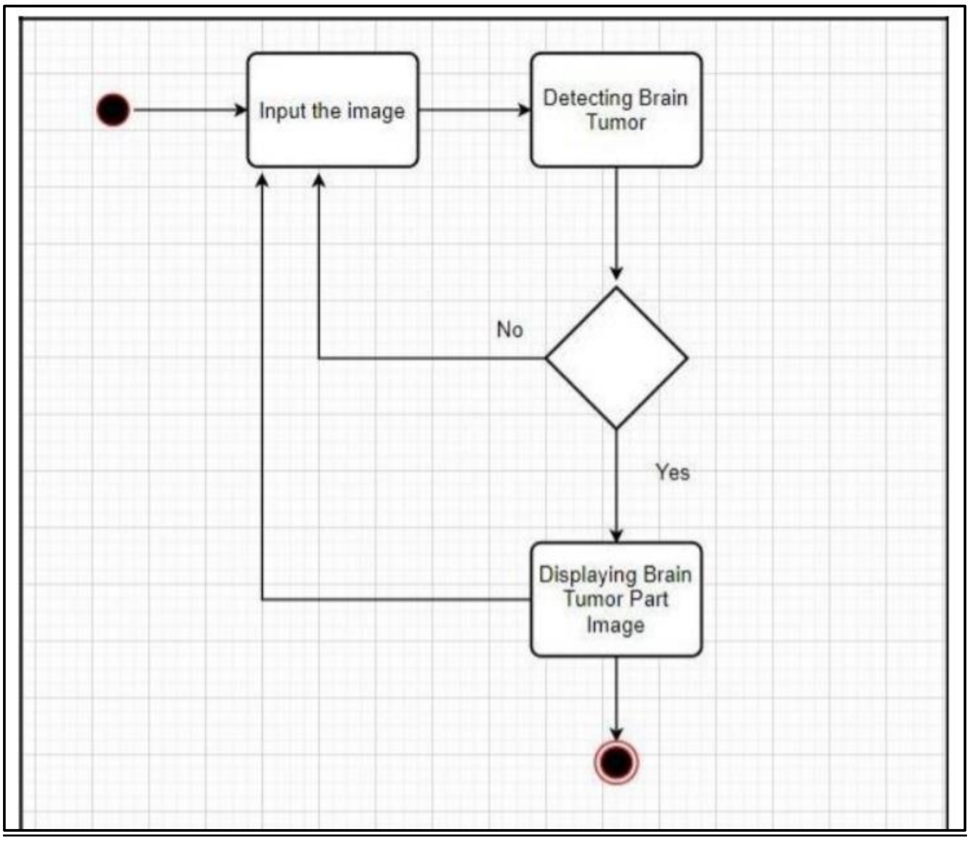


Fig. 4.3.2.3 Activity Diagram

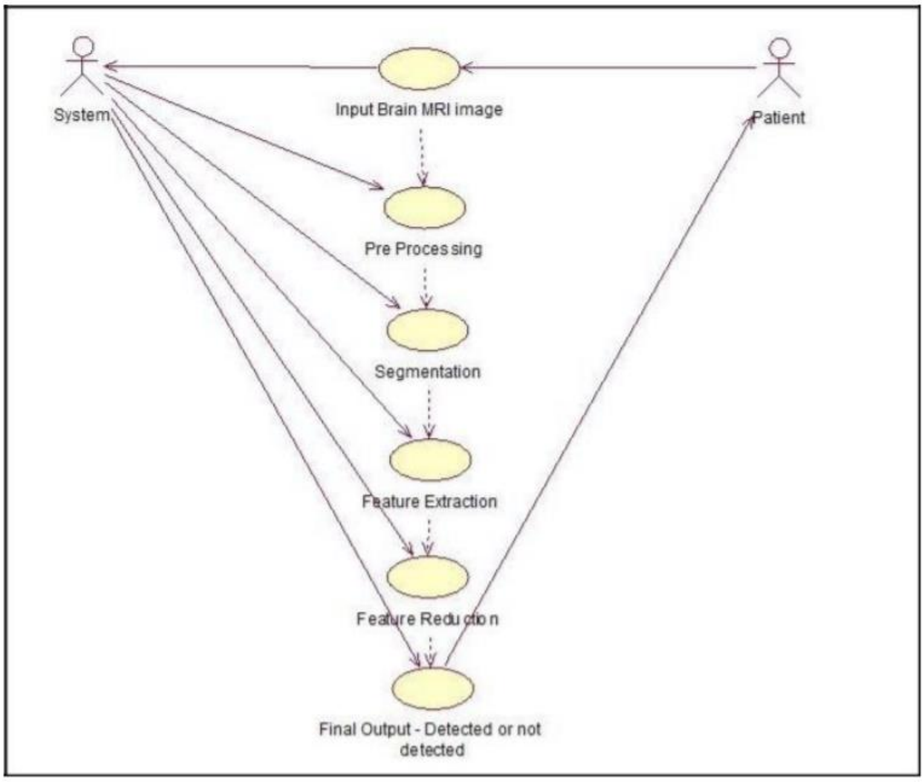


Fig. 4.3.2.4 Use case Diagram

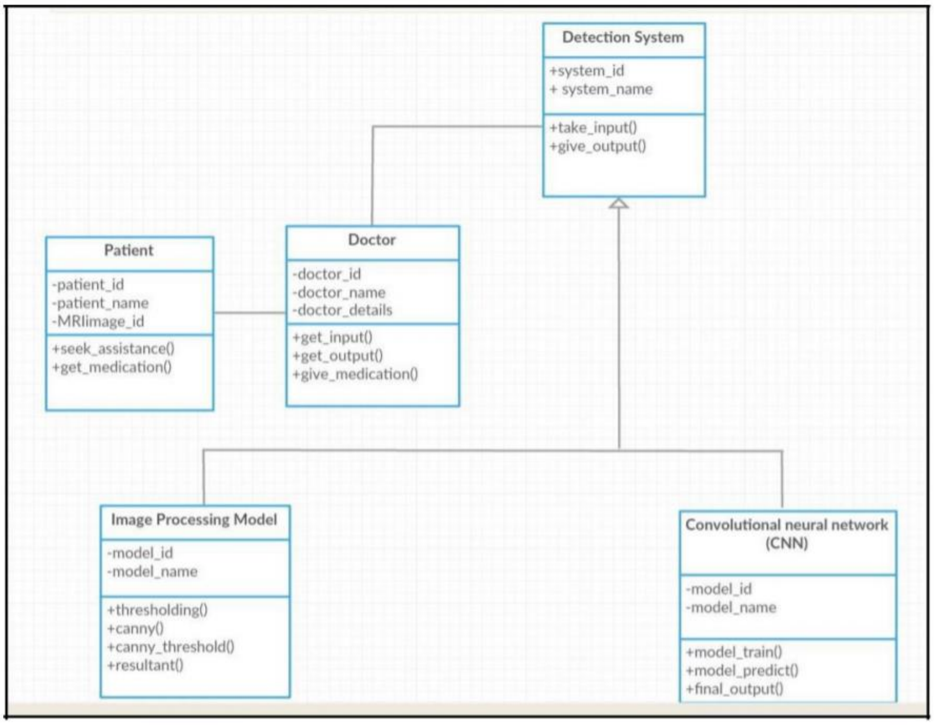
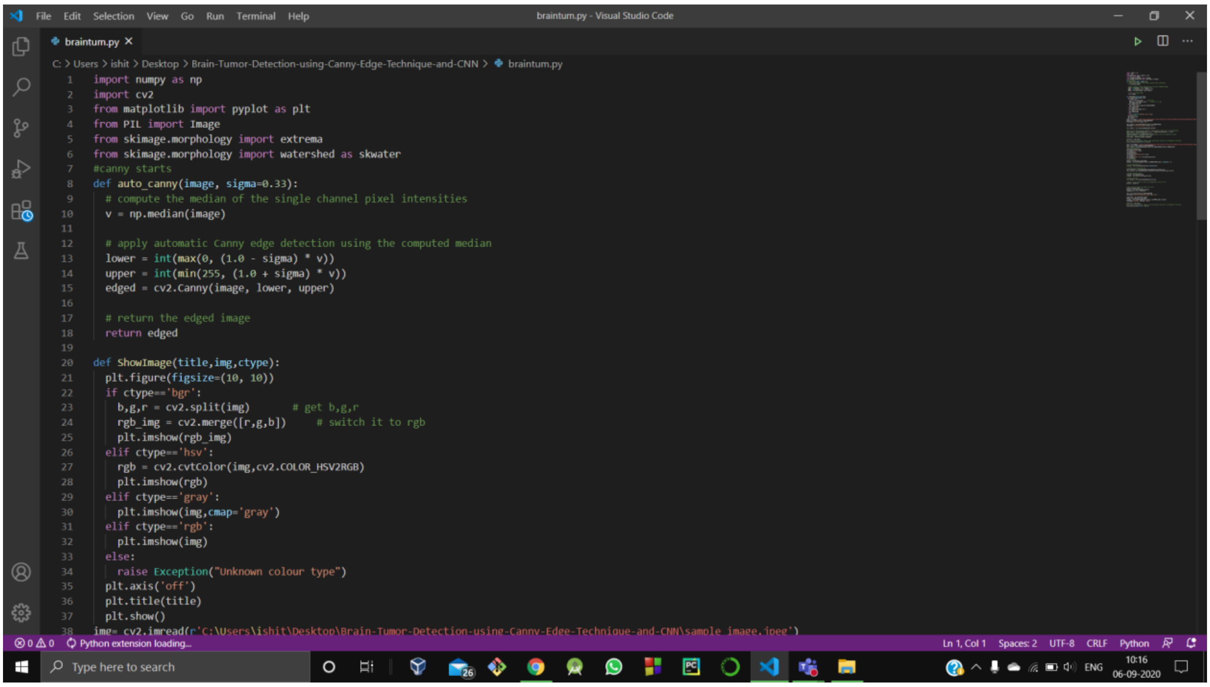


Fig. 4.3.2.5 Class Diagram

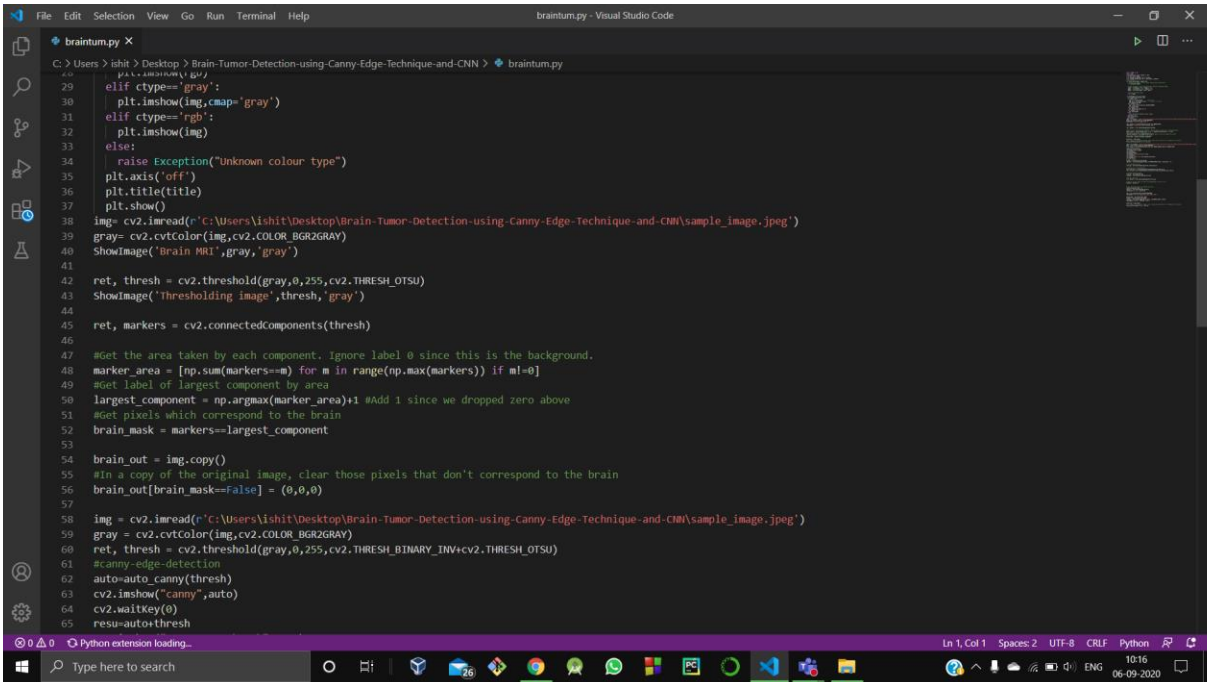
5. Implementation and Testing

5.1. Implementation details (snapshots)


File Edit Selection View Go Run Terminal Help
braintum.py - Visual Studio Code
braintum.py
C: > Users > ishit > Desktop > Brain-Tumor-Detection-using-Canny-Edge-Technique-and-CNN > braintum.py
⊗ 0 ⚠ 0 Python extension loading...
Ln 1, Col 1 Spaces: 2 UTF-8 CRLF Python
Type here to search
ENG 10:16 06-09-2020

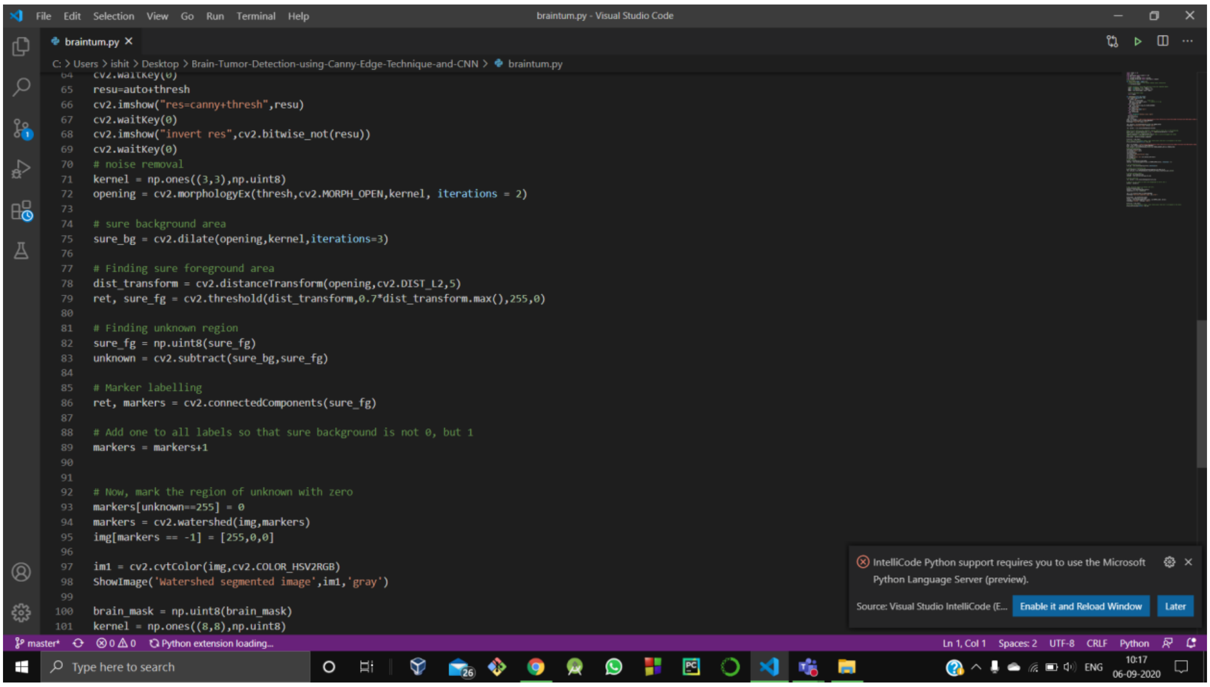
```python
import numpy as np
import cv2
from matplotlib import pyplot as plt
from PIL import Image
from skimage.morphology import extrema
from skimage.morphology import watershed as skwater
#canny starts
def auto_canny(image, sigma=0.33):
  # compute the median of the single channel pixel intensities
  v = np.median(image)

  # apply automatic Canny edge detection using the computed median
  lower = int(max(0, (1.0 - sigma) * v))
  upper = int(min(255, (1.0 + sigma) * v))
  edged = cv2.Canny(image, lower, upper)

  # return the edged image
  return edged

def ShowImage(title,img,ctype):
  plt.figure(figsize=(10, 10))
  if ctype=='bgr':
    b,g,r = cv2.split(img)       # get b,g,r
    rgb_img = cv2.merge([r,g,b])     # switch it to rgb
    plt.imshow(rgb_img)
  elif ctype=='hsv':
    rgb = cv2.cvtColor(img,cv2.COLOR_HSV2RGB)
    plt.imshow(rgb)
  elif ctype=='gray':
    plt.imshow(img,cmap='gray')
  elif ctype=='rgb':
    plt.imshow(img)
  else:
    raise Exception("Unknown colour type")
  plt.axis('off')
  plt.title(title)
  plt.show()
img= cv2.imread(r'C:\Users\ishit\Desktop\Brain-Tumor-Detection-using-Canny-Edge-Technique-and-CNN\sample_image.jpeg')
```

Fig. 5.1.1 : Code for Image Pre-processing


File Edit Selection View Go Run Terminal Help
braintum.py - Visual Studio Code
braintum.py
C: > Users > ishit > Desktop > Brain-Tumor-Detection-using-Canny-Edge-Technique-and-CNN > braintum.py
⊗ 0 ⚠ 0 Python extension loading...
Ln 1, Col 1 Spaces: 2 UTF-8 CRLF Python
Type here to search
ENG 10:16 06-09-2020

```python
  elif ctype=='gray':
    plt.imshow(img,cmap='gray')
  elif ctype=='rgb':
    plt.imshow(img)
  else:
    raise Exception("Unknown colour type")
  plt.axis('off')
  plt.title(title)
  plt.show()
img= cv2.imread(r'C:\Users\ishit\Desktop\Brain-Tumor-Detection-using-Canny-Edge-Technique-and-CNN\sample_image.jpeg')
gray= cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
ShowImage('Brain MRI',gray,'gray')

ret, thresh = cv2.threshold(gray,0,255,cv2.THRESH_OTSU)
ShowImage('Thresholding image',thresh,'gray')

ret, markers = cv2.connectedComponents(thresh)

#Get the area taken by each component. Ignore label 0 since this is the background.
marker_area = [np.sum(markers==m) for m in range(np.max(markers)) if m!=0]
#Get label of largest component by area
largest_component = np.argmax(marker_area)+1 #Add 1 since we dropped zero above
#Get pixels which correspond to the brain
brain_mask = markers==largest_component

brain_out = img.copy()
#In a copy of the original image, clear those pixels that don't correspond to the brain
brain_out[brain_mask==False] = (0,0,0)

img = cv2.imread(r'C:\Users\ishit\Desktop\Brain-Tumor-Detection-using-Canny-Edge-Technique-and-CNN\sample_image.jpeg')
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(gray,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
#canny-edge-detection
auto=auto_canny(thresh)
cv2.imshow("canny",auto)
cv2.waitKey(0)
resu=auto+thresh
```

Fig. 5.1.2 : Code for Image Pre-processing (contd.)

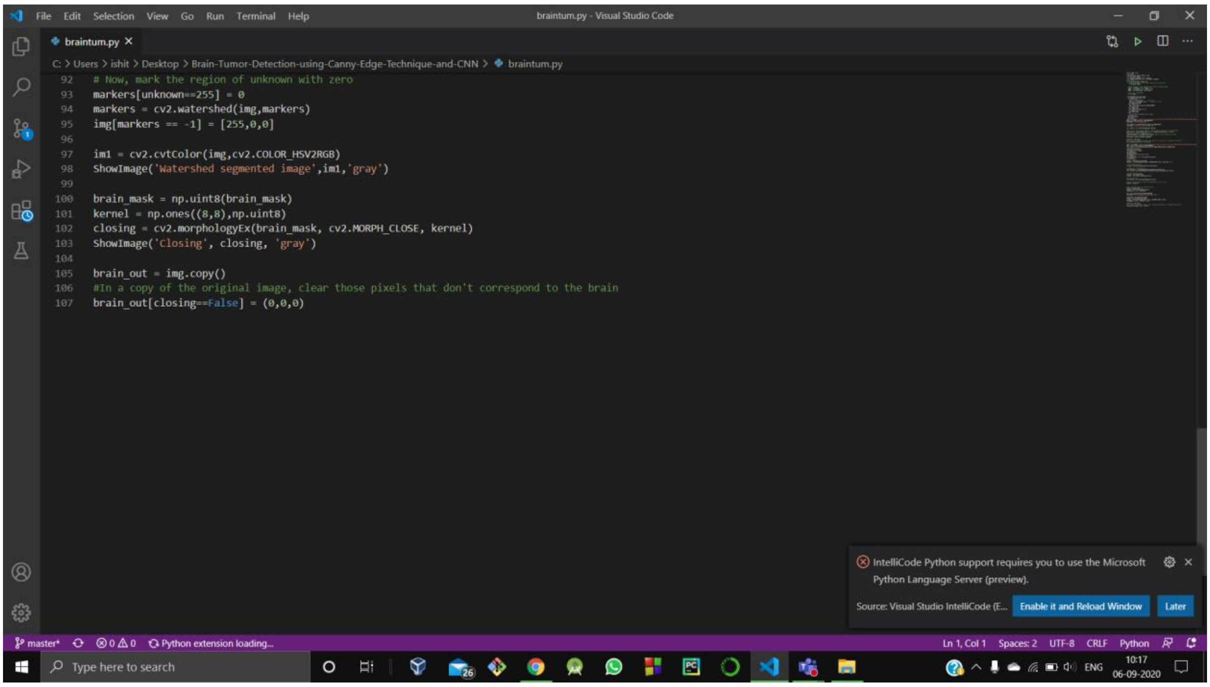

File
Edit
Selection
View
Go
Run
Terminal
Help
braintum.py - Visual Studio Code
braintum.py
C: > Users > ishit > Desktop > Brain-Tumor-Detection-using-Canny-Edge-Technique-and-CNN > braintum.py
master*
⊗ 0 ⚠ 0
Python extension loading...
Ln 1, Col 1
Spaces: 2
UTF-8
CRLF
Python
Type here to search
ENG
10:17
06-09-2020

```python
cv2.waitKey(0)
resu=auto+thresh
cv2.imshow("res=canny+thresh",resu)
cv2.waitKey(0)
cv2.imshow("invert res",cv2.bitwise_not(resu))
cv2.waitKey(0)
# noise removal
kernel = np.ones((3,3),np.uint8)
opening = cv2.morphologyEx(thresh,cv2.MORPH_OPEN,kernel, iterations = 2)

# sure background area
sure_bg = cv2.dilate(opening,kernel,iterations=3)

# Finding sure foreground area
dist_transform = cv2.distanceTransform(opening,cv2.DIST_L2,5)
ret, sure_fg = cv2.threshold(dist_transform,0.7*dist_transform.max(),255,0)

# Finding unknown region
sure_fg = np.uint8(sure_fg)
unknown = cv2.subtract(sure_bg,sure_fg)

# Marker labelling
ret, markers = cv2.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 = cv2.watershed(img,markers)
img[markers == -1] = [255,0,0]

im1 = cv2.cvtColor(img,cv2.COLOR_HSV2RGB)
ShowImage('Watershed segmented image',im1,'gray')

brain_mask = np.uint8(brain_mask)
kernel = np.ones((8,8),np.uint8)
```

IntelliCode Python support requires you to use the Microsoft Python Language Server (preview).

Source: Visual Studio IntelliCode (E...

Fig. 5.1.3 : Code for Image Pre-processing (contd.)


File
Edit
Selection
View
Go
Run
Terminal
Help
braintum.py - Visual Studio Code
braintum.py
C: > Users > ishit > Desktop > Brain-Tumor-Detection-using-Canny-Edge-Technique-and-CNN > braintum.py
master*
⊗ 0 ⚠ 0
Python extension loading...
Ln 1, Col 1
Spaces: 2
UTF-8
CRLF
Python
Type here to search
ENG
10:17
06-09-2020

```python
# Now, mark the region of unknown with zero
markers[unknown==255] = 0
markers = cv2.watershed(img,markers)
img[markers == -1] = [255,0,0]

im1 = cv2.cvtColor(img,cv2.COLOR_HSV2RGB)
ShowImage('Watershed segmented image',im1,'gray')

brain_mask = np.uint8(brain_mask)
kernel = np.ones((8,8),np.uint8)
closing = cv2.morphologyEx(brain_mask, cv2.MORPH_CLOSE, kernel)
ShowImage('Closing', closing, 'gray')

brain_out = img.copy()
#In a copy of the original image, clear those pixels that don't correspond to the brain
brain_out[closing==False] = (0,0,0)
```

IntelliCode Python support requires you to use the Microsoft Python Language Server (preview).

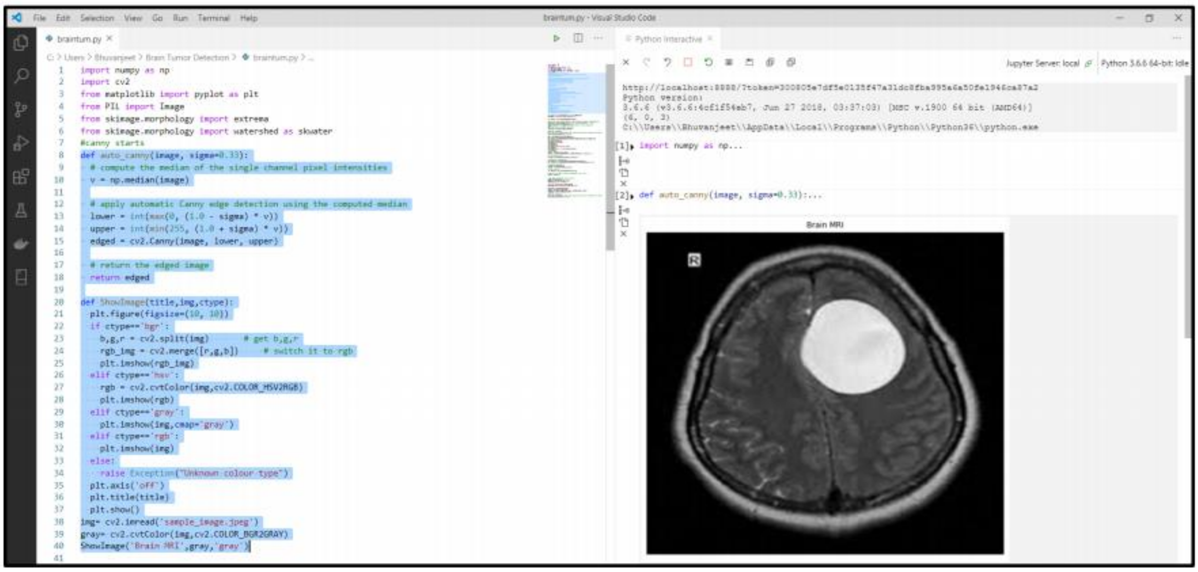
Source: Visual Studio IntelliCode (E...

Fig. 5.1.4 : Code for Image Pre-processing (contd.)

The images below show the various steps involved in brain tumor detection:

PHASE 1 - Data – Preprocessing :

We pre-process the MRI scan input image by performing image processing techniques one by one.


File Edit Selection View Go Run Terminal Help
braintum.py - Visual Studio Code
braintum.py
C: > Users > Bhuvanjeet > Brain Tumor Detection > braintum.py > ...
Python Interactive
Jupyter Server: local
Python 3.6.6 64-bit: Idle
Brain MRI
R

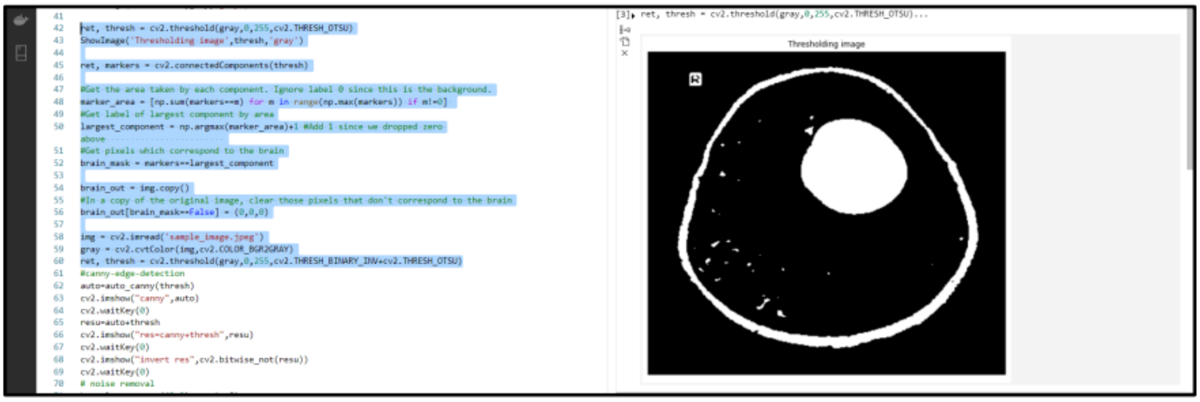
```
import numpy as np
import cv2
from matplotlib import pyplot as plt
from PIL import Image
from skimage.morphology import extrema
from skimage.morphology import watershed as skwater
#canny starts
def auto_canny(image, sigma=0.33):
    # compute the median of the single channel pixel intensities
    v = np.median(image)

    # apply automatic Canny edge detection using the computed median
    lower = int(max(0, (1.0 - sigma) * v))
    upper = int(min(255, (1.0 + sigma) * v))
    edged = cv2.Canny(image, lower, upper)

    # return the edged image
    return edged

def ShowImage(title,img,ctype):
    plt.figure(figsize=(10, 10))
    if ctype=='bgr':
        b,g,r = cv2.split(img)       # get b,g,r
        rgb_img = cv2.merge([r,g,b])     # switch it to rgb
        plt.imshow(rgb_img)
    elif ctype=='hsv':
        rgb = cv2.cvtColor(img,cv2.COLOR_HSV2RGB)
        plt.imshow(rgb)
    elif ctype=='gray':
        plt.imshow(img,cmap='gray')
    elif ctype=='rgb':
        plt.imshow(img)
    else:
        raise Exception("Unknown colour type")
    plt.axis('off')
    plt.title(title)
    plt.show()
img= cv2.imread('sample_image.jpeg')
gray= cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
ShowImage('Brain MRI',gray,'gray')
```

```
http://localhost:8888/?token=300805e7df5e0135f47a31dc8fba995a6a50fe1946ca87a2
Python version:
3.6.6 (v3.6.6:4cf1f54eb7, Jun 27 2018, 03:37:03) [MSC v.1900 64 bit (AMD64)]
(6, 0, 3)
C:\\Users\\Bhuvanjeet\\AppData\\Local\\Programs\\Python\\Python36\\python.exe
[1] import numpy as np...
[2] def auto_canny(image, sigma=0.33):...
```

Fig. 5.1.5 Input Brain MRI image

We can see clearly that Tumor is present in the image. Let's see how our model detects and predicts it.

Step 1 - Thresholded Image : Thresholding is used to create binary images from a grayscale image. It is like the most basic method of image segmentation. This makes image analysing easier.

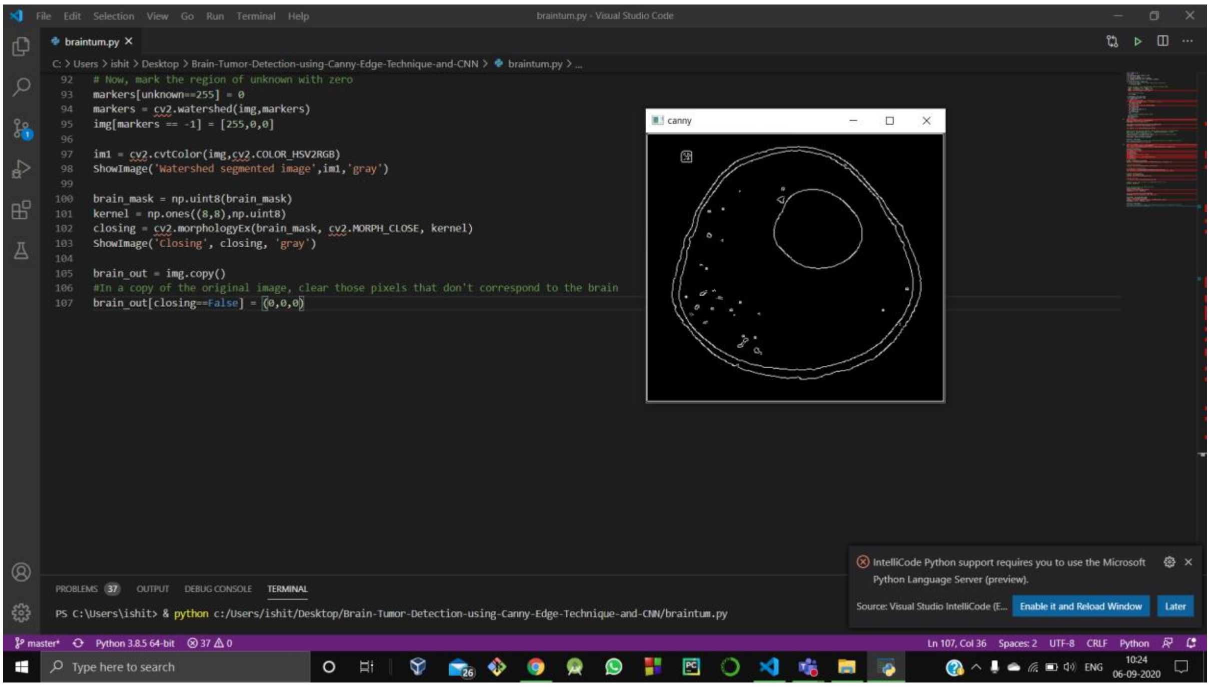


```

ret, thresh = cv2.threshold(gray,0,255,cv2.THRESH_OTSU)
ShowImage('Thresholding image',thresh,'gray')

ret, markers = cv2.connectedComponents(thresh)

#Get the area taken by each component. Ignore label 0 since this is the background.
marker_area = [np.sum(markers==m) for m in range(np.max(markers)) if m!=0]
#Get label of largest component by area
largest_component = np.argmax(marker_area)+1 #Add 1 since we dropped zero above
#Get pixels which correspond to the brain
brain_mask = markers==largest_component

brain_out = img.copy()
#In a copy of the original image, clear those pixels that don't correspond to the brain
brain_out[brain_mask==False] = (0,0,0)

img = cv2.imread('sample_image.jpeg')
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(gray,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
#canny-edge-detection
auto=auto_canny(thresh)
cv2.imshow("canny",auto)
cv2.waitKey(0)
resu=auto+thresh
cv2.imshow("res=canny+thresh",resu)
cv2.waitKey(0)
cv2.imshow("invert res",cv2.bitwise_not(resu))
cv2.waitKey(0)
# noise removal
```

```
[3] ret, thresh = cv2.threshold(gray,0,255,cv2.THRESH_OTSU)...
```

Fig. 5.1.6 Image Thresholding

Step 2 - Image after canny edge detection : Canny edge detection uses a multi-stage algorithm to figure out a vast range of edges in the images along with suppression of noise at the same time.


File Edit Selection View Go Run Terminal Help
braintum.py - Visual Studio Code
braintum.py
C: > Users > ishit > Desktop > Brain-Tumor-Detection-using-Canny-Edge-Technique-and-CNN > braintum.py > ...
canny
PROBLEMS 37 OUTPUT DEBUG CONSOLE TERMINAL
master* Python 3.8.5 64-bit ⊗ 37 ⚠ 0
Ln 107, Col 36 Spaces: 2 UTF-8 CRLF Python
Type here to search
ENG 10:24 06-09-2020

```python
# Now, mark the region of unknown with zero
markers[unknown==255] = 0
markers = cv2.watershed(img,markers)
img[markers == -1] = [255,0,0]

im1 = cv2.cvtColor(img,cv2.COLOR_HSV2RGB)
ShowImage('Watershed segmented image',im1,'gray')

brain_mask = np.uint8(brain_mask)
kernel = np.ones((8,8),np.uint8)
closing = cv2.morphologyEx(brain_mask, cv2.MORPH_CLOSE, kernel)
ShowImage('Closing', closing, 'gray')

brain_out = img.copy()
#In a copy of the original image, clear those pixels that don't correspond to the brain
brain_out[closing==False] = (0,0,0)
```

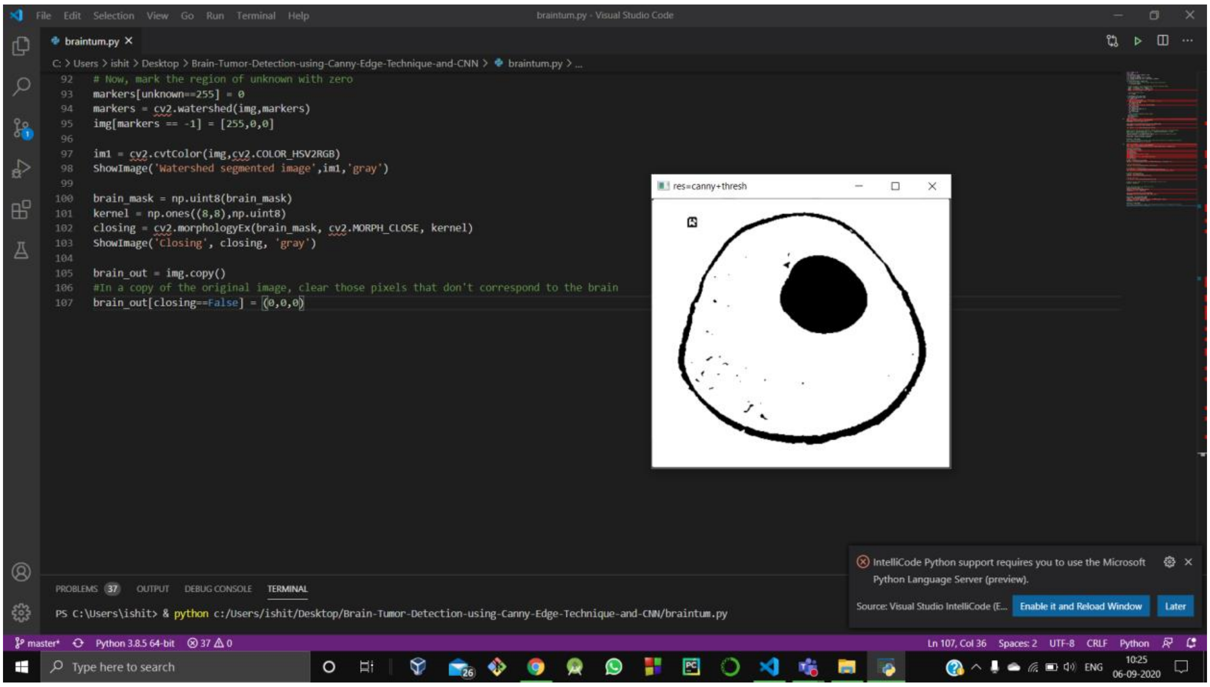
```
PS C:\Users\ishit> & python c:/Users/ishit/Desktop/Brain-Tumor-Detection-using-Canny-Edge-Technique-and-CNN/braintum.py
```

IntelliCode Python support requires you to use the Microsoft Python Language Server (preview).

Source: Visual Studio IntelliCode (E... Enable it and Reload Window Later

Fig. 5.1.6 Canny edge detection

Step 3- Overlapping of previously thresholded image on canny edge detected image : We overlap both the obtained images as one


File Edit Selection View Go Run Terminal Help
braintum.py - Visual Studio Code
braintum.py
C: > Users > ishit > Desktop > Brain-Tumor-Detection-using-Canny-Edge-Technique-and-CNN > braintum.py > ...
res=canny+thresh
PROBLEMS 37 OUTPUT DEBUG CONSOLE TERMINAL
master* Python 3.8.5 64-bit ⊗ 37 ⚠ 0
Ln 107, Col 36 Spaces: 2 UTF-8 CRLF Python
Type here to search
ENG 10:25 06-09-2020

```python
# Now, mark the region of unknown with zero
markers[unknown==255] = 0
markers = cv2.watershed(img,markers)
img[markers == -1] = [255,0,0]

im1 = cv2.cvtColor(img,cv2.COLOR_HSV2RGB)
ShowImage('Watershed segmented image',im1,'gray')

brain_mask = np.uint8(brain_mask)
kernel = np.ones((8,8),np.uint8)
closing = cv2.morphologyEx(brain_mask, cv2.MORPH_CLOSE, kernel)
ShowImage('Closing', closing, 'gray')

brain_out = img.copy()
#In a copy of the original image, clear those pixels that don't correspond to the brain
brain_out[closing==False] = (0,0,0)
```

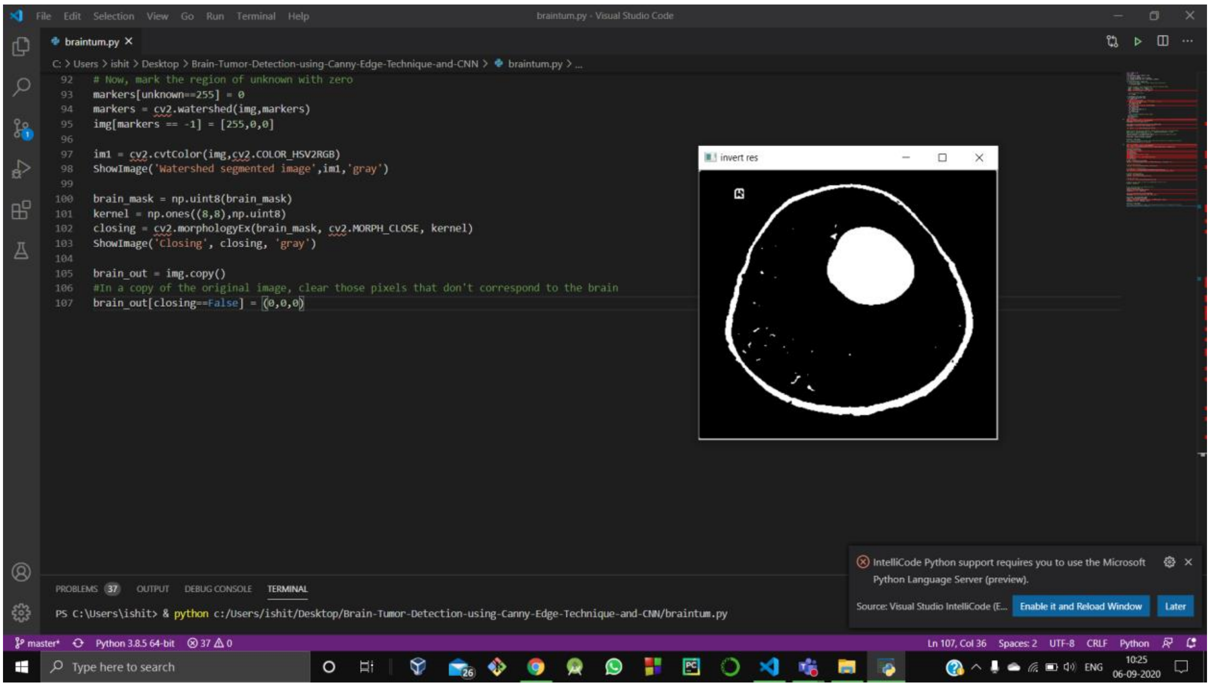
```
PS C:\Users\ishit> & python c:/Users/ishit/Desktop/Brain-Tumor-Detection-using-Canny-Edge-Technique-and-CNN/braintum.py
```

IntelliCode Python support requires you to use the Microsoft Python Language Server (preview).

Source: Visual Studio IntelliCode (E... Enable it and Reload Window Later

Fig. 5.1.7 Image overlapping

Step 4 - Inverting the above image i.e. taking the photographic negative of the above image, we get


braintum.py - Visual Studio Code
braintum.py
C: > Users > ishit > Desktop > Brain-Tumor-Detection-using-Canny-Edge-Technique-and-CNN > braintum.py > ...
invert res
PROBLEMS 37
OUTPUT
DEBUG CONSOLE
TERMINAL
Enable it and Reload Window
Later
master*
Python 3.8.5 64-bit
Ln 107, Col 36
Spaces: 2
UTF-8
CRLF
Python
Type here to search
10:25
06-09-2020

```python
# Now, mark the region of unknown with zero
markers[unknown==255] = 0
markers = cv2.watershed(img,markers)
img[markers == -1] = [255,0,0]

im1 = cv2.cvtColor(img,cv2.COLOR_HSV2RGB)
ShowImage('Watershed segmented image',im1,'gray')

brain_mask = np.uint8(brain_mask)
kernel = np.ones((8,8),np.uint8)
closing = cv2.morphologyEx(brain_mask, cv2.MORPH_CLOSE, kernel)
ShowImage('Closing', closing, 'gray')

brain_out = img.copy()
#In a copy of the original image, clear those pixels that don't correspond to the brain
brain_out[closing==False] = (0,0,0)
```

```
PS C:\Users\ishit> & python c:/Users/ishit/Desktop/Brain-Tumor-Detection-using-Canny-Edge-Technique-and-CNN/braintum.py
```

IntelliCode Python support requires you to use the Microsoft Python Language Server (preview).

Source: Visual Studio IntelliCode (E...

Fig. 5.1.8 Image Negative

Step 5 - Now performing Watershed Segmentation :

To identify the tumor more clearly, we perform the watershed segmentation. Such an algorithm is specifically useful for objects touching one another, as in this case we have the brain and the tumor. Here we view the grayscale image as a topological surface and our aim is to separate the two objects into different catchment basins divided by the watershed ridge lines. In simpler words it gives us more clarity to detect the tumor in this case.

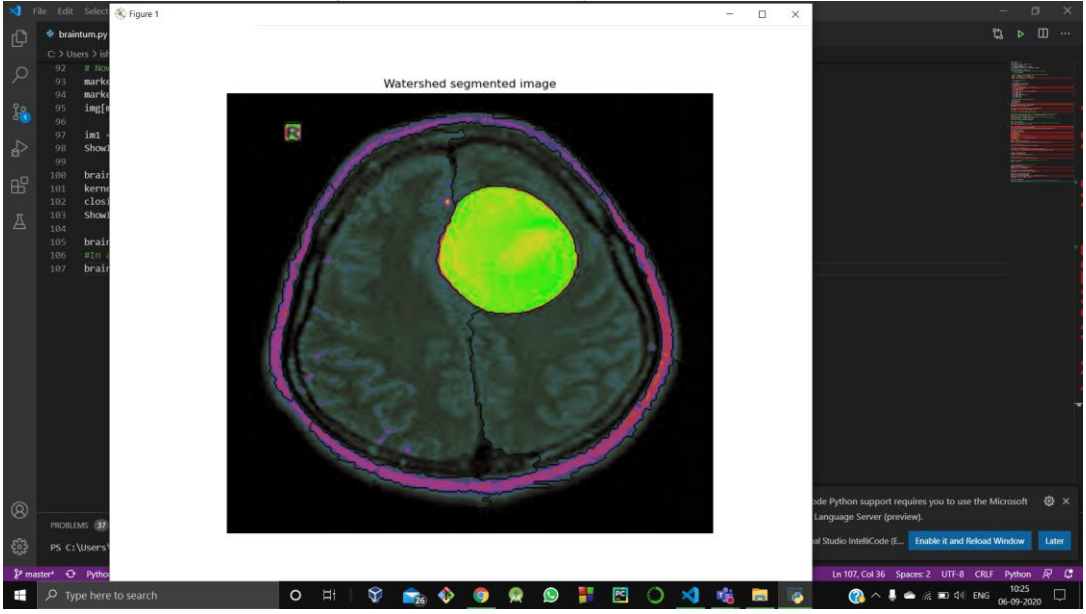
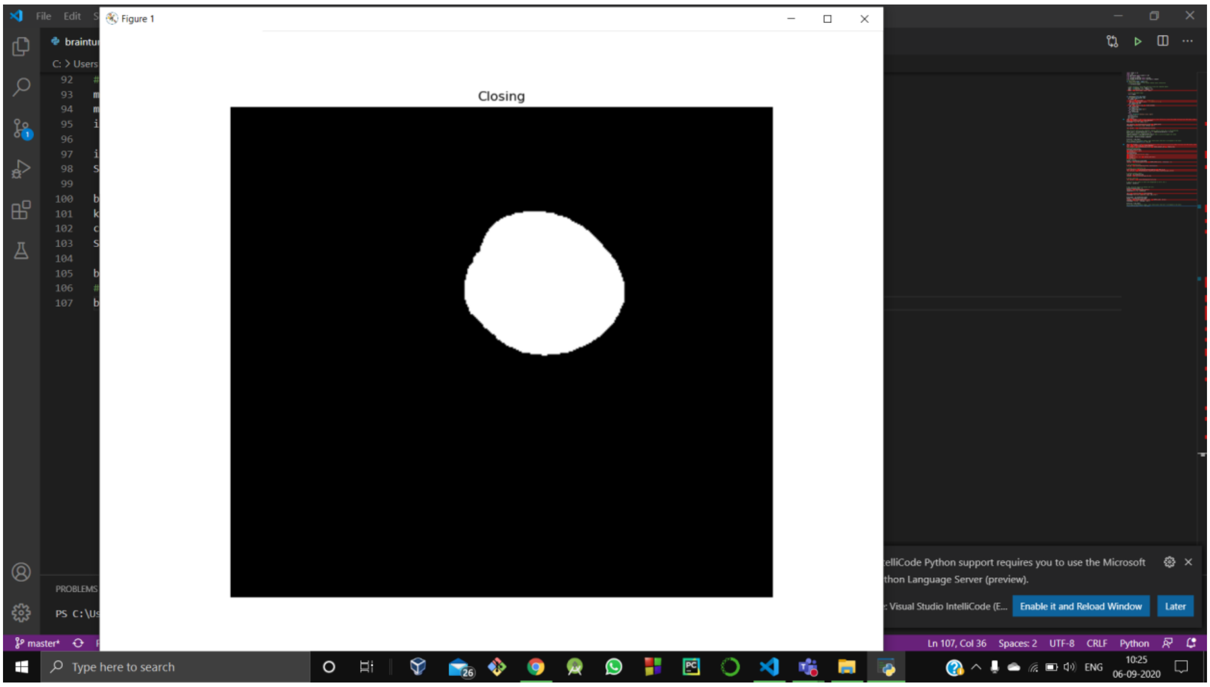


Fig. 5.1.9 Watershed Segmented Image

Step 6 - Finally, on closing this image, the tumor gets extracted.



ɪlliCode Python support requires you to use the Microsoft
thon Language Server (preview).
: Visual Studio IntelliCode (E... Enable it and Reload Window Later

Fig. 5.1.10 Closing of the image

PHASE 2 : Data Augmentation

After this, we will perform Data Augmentation on this pre-processed obtained image.
1- To create more images (since there are only 253 brain MRI images in the dataset)
2- To solve data imbalance issue (since 61% images are tumorous)
3- To do regularization (it prevents overfitting to some extent)

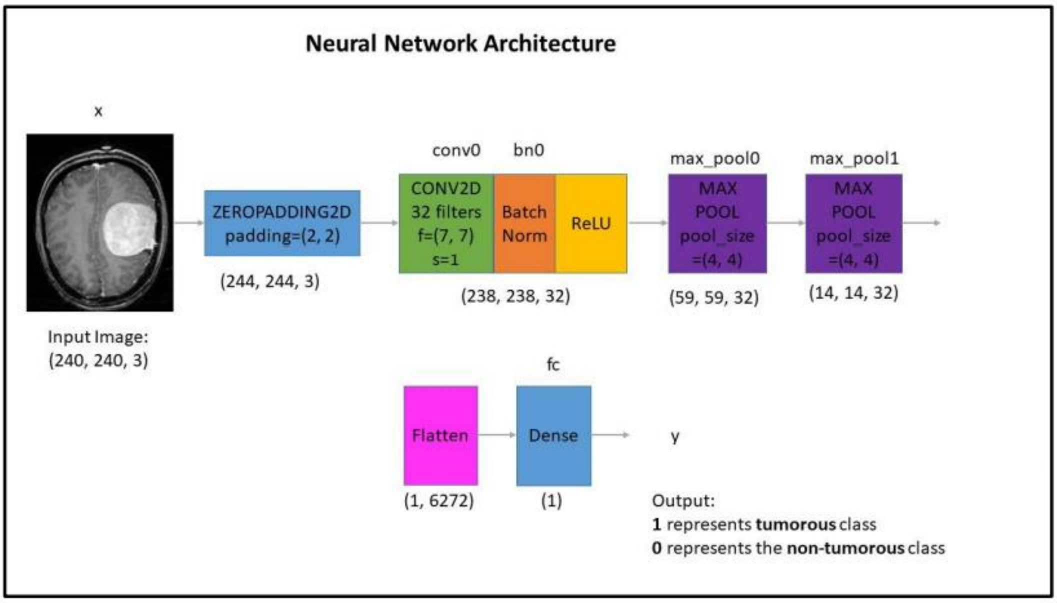
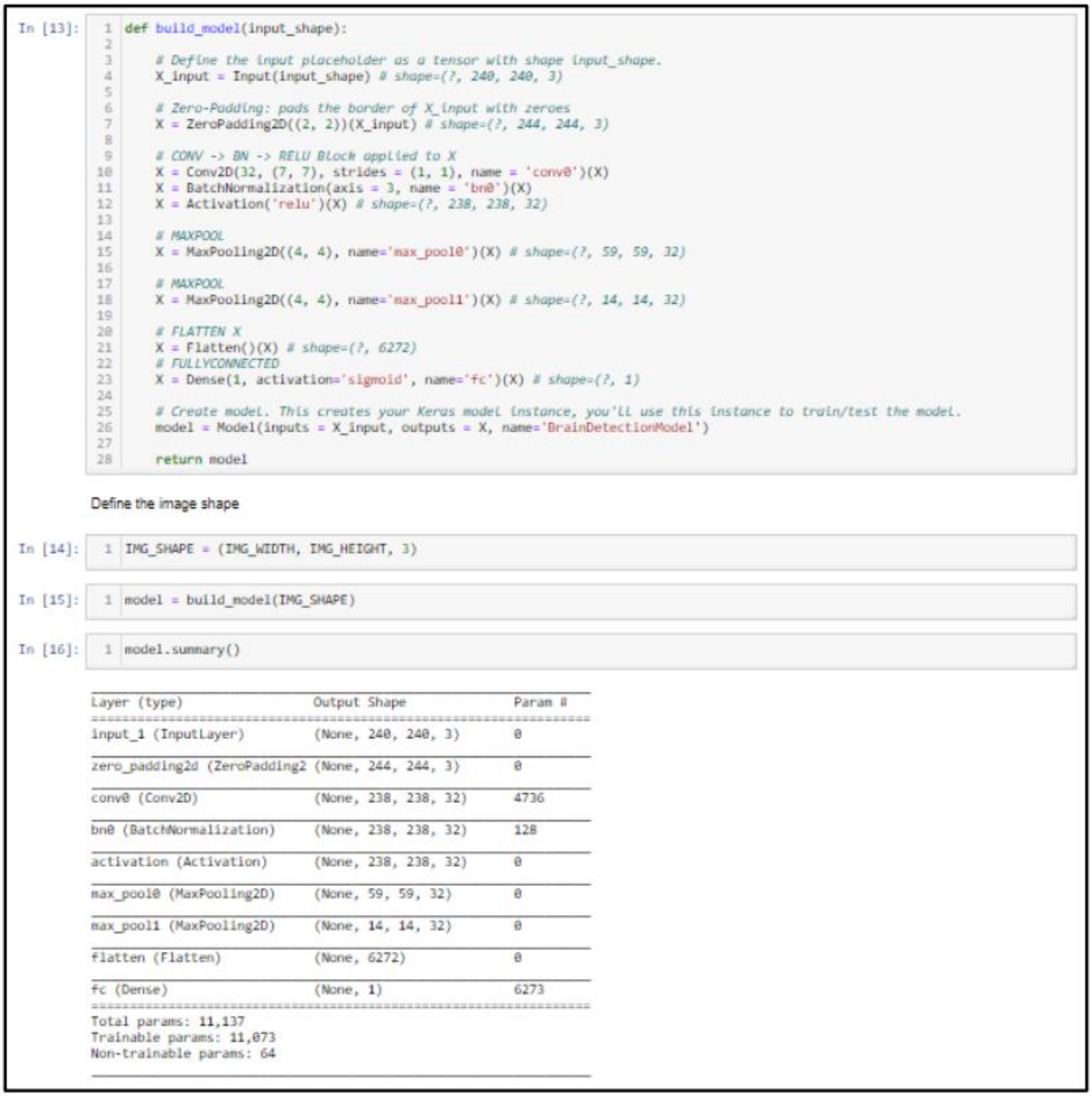


Fig. 5.1.11 Data Augmentation

PHASE 3 : Image passed on to the trained CNN Model

In neural networks, Convolutional neural networks algorithm is one of the main types of ML algorithms to perform recognition and classification of images [8]. Facial recognition, objects detections, etc., are some of the fields where CNNs are in demand.

CNN is known for taking an input image and classifying it into different types/categories. In this case, it shall classify the MRI brain scan into tumor detected (positive) or not detected (negative).



In [13]:
```
def build_model(input_shape):

    # Define the input placeholder as a tensor with shape input_shape.
    X_input = Input(input_shape) # shape=(?, 240, 240, 3)

    # Zero-Padding: pads the border of X_input with zeroes
    X = ZeroPadding2D((2, 2))(X_input) # shape=(?, 244, 244, 3)

    # CONV -> BN -> RELU Block applied to X
    X = Conv2D(32, (7, 7), strides = (1, 1), name = 'conv0')(X)
    X = BatchNormalization(axis = 3, name = 'bn0')(X)
    X = Activation('relu')(X) # shape=(?, 238, 238, 32)

    # MAXPOOL
    X = MaxPooling2D((4, 4), name='max_pool0')(X) # shape=(?, 59, 59, 32)

    # MAXPOOL
    X = MaxPooling2D((4, 4), name='max_pool1')(X) # shape=(?, 14, 14, 32)

    # FLATTEN X
    X = Flatten()(X) # shape=(?, 6272)
    # FULLYCONNECTED
    X = Dense(1, activation='sigmoid', name='fc')(X) # shape=(?, 1)

    # Create model. This creates your Keras model instance, you'll use this instance to train/test the model.
    model = Model(inputs = X_input, outputs = X, name='BrainDetectionModel')

    return model
```

Define the image shape

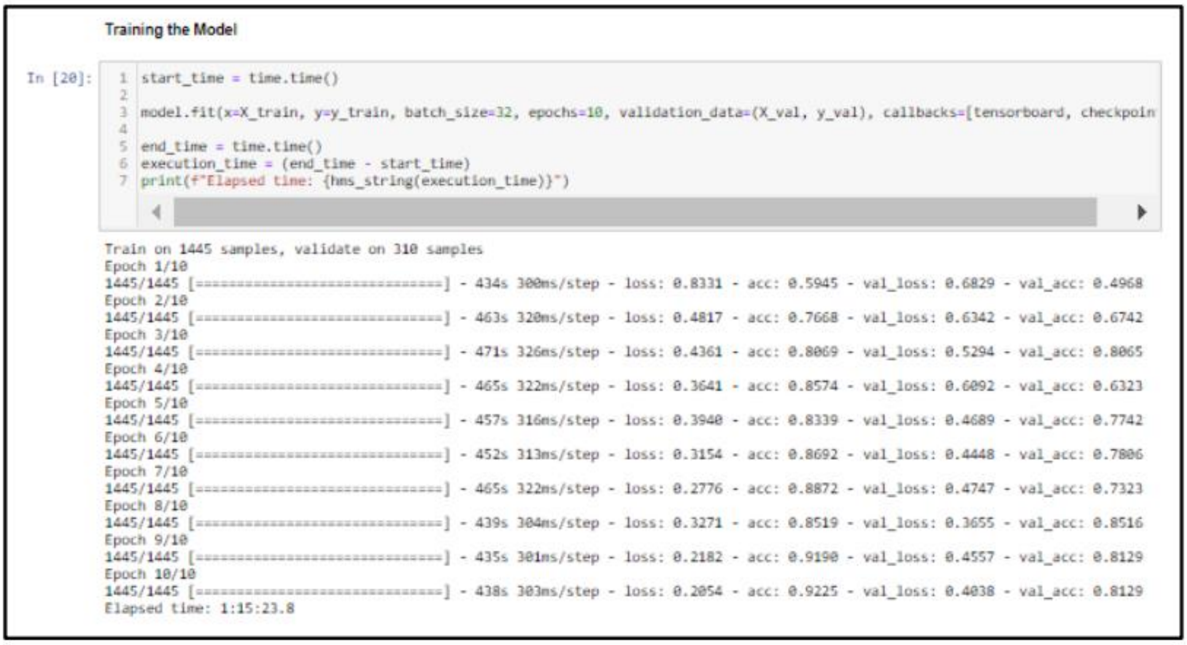
In [14]:
```
IMG_SHAPE = (IMG_WIDTH, IMG_HEIGHT, 3)
```

In [15]:
```
model = build_model(IMG_SHAPE)
```

In [16]:
```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 240, 240, 3)       0
_________________________________________________________________
zero_padding2d (ZeroPadding2 (None, 244, 244, 3)       0
_________________________________________________________________
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
_________________________________________________________________
```

Fig.5.1.12 CNN Model



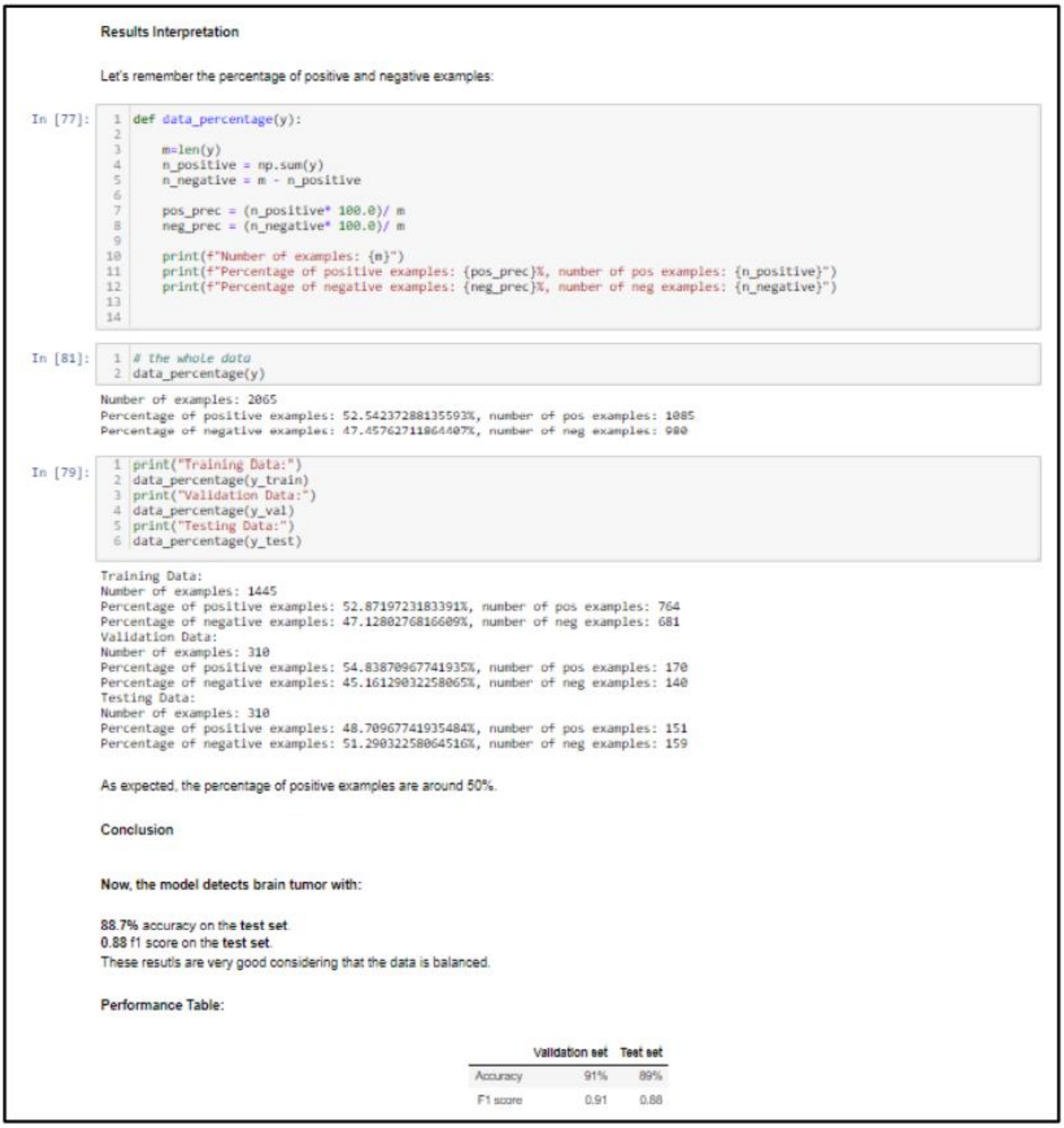
Training the Model

In [20]:

```
start_time = time.time()

model.fit(x=X_train, y=y_train, batch_size=32, epochs=10, validation_data=(X_val, y_val), callbacks=[tensorboard, checkpoin

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

```
Train on 1445 samples, validate on 310 samples
Epoch 1/10
1445/1445 [==============================] - 434s 300ms/step - loss: 0.8331 - acc: 0.5945 - val_loss: 0.6829 - val_acc: 0.4968
Epoch 2/10
1445/1445 [==============================] - 463s 320ms/step - loss: 0.4817 - acc: 0.7668 - val_loss: 0.6342 - val_acc: 0.6742
Epoch 3/10
1445/1445 [==============================] - 471s 326ms/step - loss: 0.4361 - acc: 0.8069 - val_loss: 0.5294 - val_acc: 0.8065
Epoch 4/10
1445/1445 [==============================] - 465s 322ms/step - loss: 0.3641 - acc: 0.8574 - val_loss: 0.6092 - val_acc: 0.6323
Epoch 5/10
1445/1445 [==============================] - 457s 316ms/step - loss: 0.3940 - acc: 0.8339 - val_loss: 0.4689 - val_acc: 0.7742
Epoch 6/10
1445/1445 [==============================] - 452s 313ms/step - loss: 0.3154 - acc: 0.8692 - val_loss: 0.4448 - val_acc: 0.7806
Epoch 7/10
1445/1445 [==============================] - 465s 322ms/step - loss: 0.2776 - acc: 0.8872 - val_loss: 0.4747 - val_acc: 0.7323
Epoch 8/10
1445/1445 [==============================] - 439s 304ms/step - loss: 0.3271 - acc: 0.8519 - val_loss: 0.3655 - val_acc: 0.8516
Epoch 9/10
1445/1445 [==============================] - 435s 301ms/step - loss: 0.2182 - acc: 0.9190 - val_loss: 0.4557 - val_acc: 0.8129
Epoch 10/10
1445/1445 [==============================] - 438s 303ms/step - loss: 0.2054 - acc: 0.9225 - val_loss: 0.4038 - val_acc: 0.8129
Elapsed time: 1:15:23.8
```

Fig. 5.1.13 Training the model



Results Interpretation

Let's remember the percentage of positive and negative examples:

```
In [77]:
def data_percentage(y):

    m=len(y)
    n_positive = np.sum(y)
    n_negative = m - n_positive

    pos_prec = (n_positive* 100.0)/ m
    neg_prec = (n_negative* 100.0)/ m

    print(f"Number of examples: {m}")
    print(f"Percentage of positive examples: {pos_prec}%, number of pos examples: {n_positive}")
    print(f"Percentage of negative examples: {neg_prec}%, number of neg examples: {n_negative}")

```

```
In [81]:
# the whole data
data_percentage(y)
```

```
Number of examples: 2065
Percentage of positive examples: 52.54237288135593%, number of pos examples: 1085
Percentage of negative examples: 47.45762711864407%, number of neg examples: 980
```

```
In [79]:
print("Training Data:")
data_percentage(y_train)
print("Validation Data:")
data_percentage(y_val)
print("Testing Data:")
data_percentage(y_test)
```

```
Training Data:
Number of examples: 1445
Percentage of positive examples: 52.8719723183391%, number of pos examples: 764
Percentage of negative examples: 47.1280276816609%, number of neg examples: 681
Validation Data:
Number of examples: 310
Percentage of positive examples: 54.83870967741935%, number of pos examples: 170
Percentage of negative examples: 45.16129032258065%, number of neg examples: 140
Testing Data:
Number of examples: 310
Percentage of positive examples: 48.70967741935484%, number of pos examples: 151
Percentage of negative examples: 51.29032258064516%, number of neg examples: 159
```

As expected, the percentage of positive examples are around 50%.

Conclusion

Now, the model detects brain tumor with:

88.7% accuracy on the **test set**.
0.88 f1 score on the **test set**.
These resutls are very good considering that the data is balanced.

Performance Table:

	Validation set	Test set
Accuracy	91%	89%
F1 score	0.91	0.88

Fig. 5.1.14 Result Interpretation and Analysis

	Validation Set	***Test Set***
Accuracy	***91%***	***89%***
F1 -score	***0.91***	***0.88***

Table 5.1.1 Tabular Results

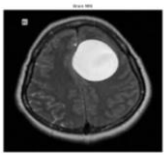
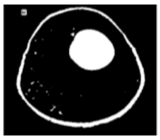
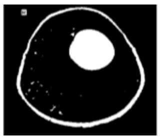
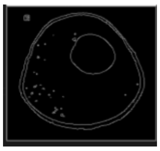
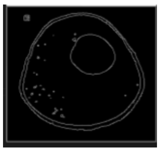
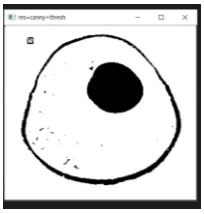
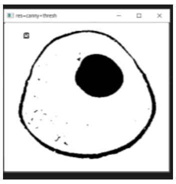
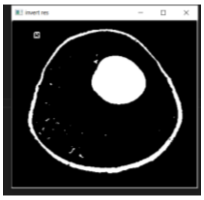
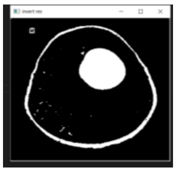
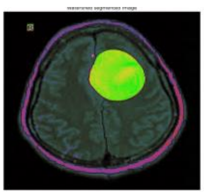
5.2. Testing

The project is based on brain tumor detection from the image that is produced after MRI. So the test cases for the project will be based on the different phases and modules in which the image is undergoing some changes and enhancing the quality of the image so as to produce a clear result whether the person has a tumor or not.

5.2.1. Type of Testing - Boundary Value Analysis

Boundary value analysis is a very efficient method of software testing where the tests are created to involve range-wise representatives of boundary values. Below we have included two major test cases involved in this project : image with and without a tumor, representing the boundary value edge-cases for the software, covering the main functionality aspect of the system.

5.2.2. Test cases

Test case number	Expected Input	Operation	Input Image	Expected Output	Output Image
T1 (Image with Tumor)	Initial MRI Image Input	Thresholding		Image containing only area of Interest	
	Thresholded image	Edge detection		Image containing only boundary	
	Edge detected Image	Negative and Thresholding		Negative of the input image and after noise reduction	
	Negative image after edge detection	Reverse for just thresholding		Image just containing the reversal of thresholding	
	Negative of thresholded image	Watershed segmentation		Segmented Image	

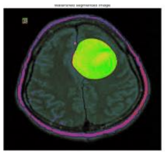
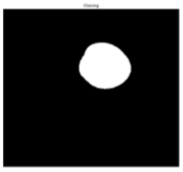
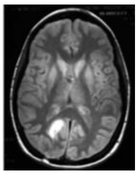
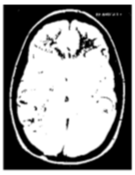
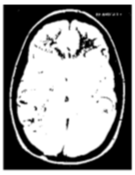
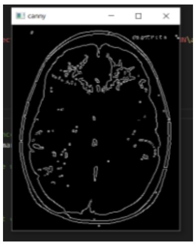
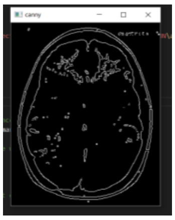
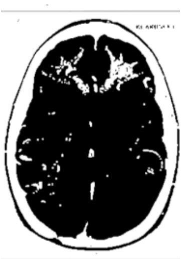
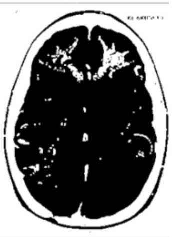
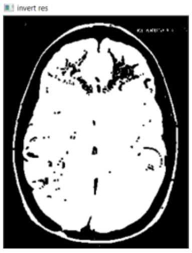
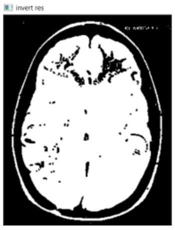
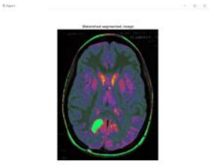
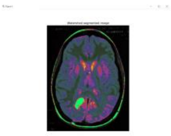
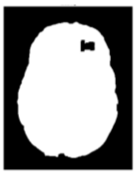
	Segmented Image	Closing		Tumor position	
T2 (Image without a tumor)	Initial MRI Image Input	Thresholding		Image containing only area of Interest	
	Thresholded image	Edge detection		Image containing only boundary	
	Edge detected Image	Negative and Thresholding		Negative of the input image and after noise reduction	
	Negative image after edge detection	Reverse for just thresholding		Image just containing the reversal of thresholding	
	Negative of thresholded image	Watershed segmentation		Segmented Image	
	Segmented Image	Closing		Tumor position	

Table 5.2.2.1 : Test Cases

6. Conclusion, Limitations and Scope for future Work

6.1 Conclusions

This project was an effort from our side to blend the concepts of Image Processing and Machine Learning to help the medical field detect tumors with as much accuracy as possible with the help of an efficient and trained Machine Learning Model. We successfully managed to build a software that is capable of detecting whether a brain tumor is present in the brain or not by feeding an MRI Scan brain image to the system and getting the resultant image depicting the brain tumor if any. This was achieved by pre-processing this input image with the help of a series of Image Processing Techniques such as Image Thresholding, Canny Edge Detection, Overlapping, Image Negative and finally, the Watershed Segmentation technique.

This pre-processed image was then given to a trained Convolutional Neural Network Model, that helped in detecting the tumor accurately. Before passing the image to the model, data augmentation was performed on the image, to generate an efficient dataset of similar images. Later on, after training the model, we could detect the tumor with about 91% accuracy which is a very rare case as seen in the literature papers researched above.

The UI was created with HTML and CSS and integrated to the backend with the help of Flask. The training of the model, and detection of tumor was done using Python on Collab with libraries such as NumPy and OpenCV. The software enables the user to upload the brain MRI scan and he can finally see the obtained image which easily identifies whether a tumor is present or not, and the result as positive or negative.

6.2 Limitations

1) High variability of brain MR images in terms of their intensity ranges and contrasts between brain tissues, it is not guaranteed that the thresholding procedure will produce perfect results.
2) Many scanners produce images with strongly varying bias fields across slices. Thus, the very top or bottom slices can have very different intensities to those in the middle.
3) With respect to the computational load, the algorithm is slightly slower than the original FM model
4) Clinical approval is required before being used for the patients.
5) Design is very complex, thus requires an educated person to operate on.

6.3 Scope for future work

1) This project, only deals with only the detection of tumor i.e present or not present. In the future, we aim to include this feature of classifying the tumor-type for the users as well i.e. Normal, Benign and Malignant.
2) We intend to generate a full-fledged positive/negative tumor detection report for the patients so that the doctors can simply hand over the report results without having to manually construct the report by themselves.
3) In future, we shall also include a treatment recommendation feature for the concerned tumor, which will include which surgery, along with the suitable medications involved and available for that specific type of tumor, by our AI virtual assistant.

7. References

[1] Pratima Purushottam Gumaste* and Vinayak K. Bairagi (2020). A Hybrid Method for Brain Tumor Detection using Advanced Textural Feature Extraction.Biomedical & Pharmacology Journal, March 2020. Vol. 13(1), p. 145-157.

[2] T. Kalaiselvi, P. Sriramakrishnan and P. Nagaraja (2016). Brain Tumor Boundary Detection by Edge Indication Map Using Bi-Modal Fuzzy Histogram Thresholding Technique from MRI T2-Weighted Scans. I.J. Image, Graphics and Signal Processing, 2016, 9, 51-59.

[3] Minakshi Sharma and Dr. Sourabh Mukharjee (2012). Brain Tumor Segmentation using hybrid Genetic Algorithm and Artificial Neural Network Fuzzy Inference System (ANFIS). International Journal of Fuzzy Logic Systems (IJFLS) Vol.2, No.4, October 2012

[4] Prince Ebenezer Adjei, Henry Nunoo-Mensah, Richard Junior Amedzrovi Agbesi and Joyce Raissa Yaho Ndjanzoue (2018). Brain Tumor Segmentation using SLIC Superpixels and Optimized Thresholding Algorithm, International Journal of Computer Applications (0975 - 8887), Volume 181 - No.20, October 2018.

[5] Prof. P.Senthil, Image Mining Brain Tumor Detection using Tad Plane Volume Rendering from MRI (IBITA), 2016, Journal of Computer - JoC, Vol.1 Issue. 1, June- 2016, pg. 1-13

[6] Sathya Narayana N, Vidya T and Sarthak U Kuchalli (June 2019). Tumor Detection and classification using random forest algorithm in brain MRI, IRJCS, AM Publications, *Vol.06, Issue06, JNCS10133.*

[7] Intuitive Image processing - watershed segmentation

[8] Understanding of Convolutional Neural Network (CNN) — Deep Learning

[9] Software Requirements for a quality software

[10] Colab codebase of project for training ML model and finding accuracy