A PRELIMINARY PROJECT REPORT ON

Brain Tumor Detection Using Deep Learning

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY,
PUNE IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE

BACHELOR OF ENGINEERING

In

COMPUTER ENGINEERING

Of

SAVITRIBAI PHULE PUNE UNIVERSITY

AMAN HANSRAJ YADAV	**(B190234444)**
ABHISHEK KUMAR	**(B190234202)**
VISHWESH SINGH	**(B190234437)**
ANAND JAISWAL	**(B190234292)**

Under the guidance of
Prof. R. A. VASMATKAR

DEPARTMENT OF COMPUTER ENGINEERING
SINHGAD COLLEGE OF ENGINEERING, PUNE-41
Accredited by NAAC

2022-2023

Sinhgad Technical Education Society,
Sinhgad College of Engineering, Pune-41
Department of Computer Engineering

Date:

CERTIFICATE

This is to certify that the preliminary project report entitled

BRAIN TUMOR DETECTION USING DEEP LEARNING

Submitted by

AMAN HANSRAJ YADAV	**(405B095)**
ABHISHEK KUMAR	**(405B002)**
VISHWESH SINGH	**(405B092)**
ANAND JAISWAL	**(405B035)**

is a bonafide work carried out by him/her under the supervision of Prof. R. A. Vasmatkar guide and it is approved for the partial fulfillment of the requirements of Savitribai Phule Pune University, Pune for the award of the degree of Bachelor of Engineering (Computer Engineering) during the year 2021-22.

Prof. R. A. Vasmatkar
Guide
Department of Computer Engineering

Dr. M. P. Wankhade
Head
Department of Computer Engineering

Dr. S. D. Lokhande
Principal
Sinhgad College of Engineering

ACKNOWLEDGEMENT

We are very thankful to all the teachers who have provided us valuable guidance towards the completion of this project on “Brain Tumor Detection Using Deep Learning” as part of syllabus of bachelor’s course of BE. The help and guidance given by all the teachers from time to time shall carry us a long way.

We would like to take this opportunity to express our deep sense of gratitude to our project guide **Prof. R. A Vasmatkar** for guiding us in the right manner, correcting our doubts as well as giving us adviceabout design and solutions for the problems which we faced by giving her valuable time wheneverwe required while developing this project. We are especially grateful to our Head of Department **Dr. M. P Wankhade** for his valuable guidance, encouragement and for allowing us to use college resources and facilities. The acknowledgement will be incomplete if we do not thank our Principal **Dr. S. D Lokhande,** who gave us their blessings and motivation which has been highly instrumental in making our project.

ABSTRACT

The anomalous development of cells in brain causes brain tumor that may lead to death. The rate of deaths can be reduced by early detection of tumor. Most common method to detect the tumor in brain is the use of Magnetic Resonance Imaging (MRI). MR images are considered because it gives a clear structure of the tumor. In this paper we proposed a novel mechanism for detecting tumor from MR image by applying machine learning algorithms especially with CNN model.

The motivation behind this study is to detect brain tumor and provide better treatment for the sufferings. The abnormal growths of cells in the brain are called tumors and cancer is a term used to represent malignant tumors. Usually, MRI scans are used for the detection of cancer regions in the brain. Positron Emission Tomography, Cerebral Arteriogram, Lumbar Puncture, Molecular testing are also used for brain tumor detection. In this study, MRI scan images are taken to analyze the disease condition. Objectives of this research work are:

i) identify the abnormal image

ii) segment tumor region.

Density of the tumor can be estimated from the segmented mask and it will help in therapy. Deep learning technique is employed to detect abnormality from MRI images. Multilevel thresholding is applied to segment the tumor region. Number of malignant pixels gives the density of the affected region.

List of Figures

ABBREVIATIONS

1. CNN	Convolutional Neural Network
2. MRI	Magnetic Resonance Imaging
3. OS	Operating System
4. PIL	Python Imaging Library
5. SVM	Support Vector Machine

Contents

Chapter 1

INTRODUCTION

1.1 Deep Learning

Deep learning is an extremely fast-moving field, and the huge number of research papers and ideas can be overwhelming. Even seasoned researchers have a hard time telling company PR from real breakthroughs. The goal of this post is to review those ideas that have stood the test of time, which is perhaps the only significance test one should rely on. These ideas, or improvements of them, have been used repeatedly. They are known to work.

If you were to start in deep learning today, understanding and implementing each of these techniques would give you an excellent foundation for understanding recent research and working on your own projects. Working through papers in historical order is also a useful exercise to understand where the current techniques come from and why they were invented in the first place.

An interesting factor of deep learning is that its application domains (e.g., vision, natural language, speech) share most techniques. For example, someone who has worked in deep learning for computer vision his whole career could quickly be productive in NLP research. The specific network architectures may differ, but the concepts, approaches, and code are mostly the same. The goal here is not to give in-depth explanations or code examples for these techniques. It's not easily possible to summarize long complex papers into a single paragraph. Instead, here is provided a brief overview of each technique, its historical context, and links to papers and implementations.

The list sticks to what most people would consider the popular mainstream domains of vision, natural language, speech, and reinforcement learning/games.

1.2 Brain Tumor Detection System

The human body is made up of many organs and brain is the most critical and vital organ of them all. One of the common reasons for dysfunction of brain is brain tumor. A tumor is nothing but excess cells growing in an uncontrolled manner. Brain tumor cells grow in a way that they eventually take up all the nutrients meant for the healthy cells and tissues, which results in brain failure. Currently, doctors locate the position and the area of brain tumor by looking at the MR Images of the brain of the patient manually. This results in inaccurate detection of the tumor and is considered very time consuming.

A Brain Cancer is very critical disease which causes deaths of many individuals. The brain tumor detection and classification system is available so that it can be diagnosed at an early stage. Cancer classification is the most challenging tasks in clinical diagnosis. This project deals with such a system, which uses computer, based procedures to detect tumor blocks and classify the type of tumor using Convolution Neural Network Algorithm for MRI images of different patients. Different types of image processing techniques like image segmentation, image enhancement and feature extraction are used for the brain tumor detection in the MRI images of the cancer-affected patients. Detecting Brain tumor using Image Processing techniques it involves the four stages- Image Pre-Processing, Image segmentation, Feature Extraction, and Classification. Image processing and neural network techniques are used for improve the performance of detecting and classifying brain tumor in MRI images.

1.3 Objectives

- To provide doctors good software to identify tumor and their causes.
- Save patient’s time.
- Provide a solution appropriately at early stages.
- Get timely consultation.

1.4 Motivation

The main motivation behind Brain tumor detection is to detect brain tumors at very early stages. Magnetic resonance imaging (MRI) is a noninvasive method for producing three-dimensional (3D) tomographic images of the human body. MRI is most often used for the detection of tumors, lesions, and other abnormalities in soft tissues, such as the brain. Clinically, radiologists qualitatively analyze films produced by MRI scanners.

Computer-aided techniques for analyzing and visualizing magnetic resonance (MR) images have been investigated. Many researchers have focused on detecting and quantifying abnormalities in the brain. Automatically identifying the brain in MR images of the head is an important step in this process. Another important step for computer-aided analysis is data quality assurance. MR images contain unwanted intensity variations due to imperfections in MRI scanners. Removing or reducing these variations can improve the accuracy of automated analysis.

This project presents a novel, fully automatic method for intracranial boundary detection and tumor detection MR images of the head. The intracranial boundary is the boundary between the brain and the intracranial cavity. It accurately segments the tumor from other features in the head.

Chapter 2
LITERATURE SURVEY

Paper-1: A New Convolutional Neural Network Architecture for Automatic Detection of Brain Tumors in Magnetic Resonance Imaging Images

- **Publication Year:** 2022
- **Author:** AHMED S. MUSALLAM , AHMED S. SHERIF 2 , AND MOHAMED K. HUSSEIN
- **Journal Name:** IEEE Access: The Multidisciplinary Open Access Journal.
- **Summary:** This paper provides a new method for detecting brain tumor by deep learning method. Brain diseases are mainly caused by abnormal growth of brain cells that may damage the brain structure, and eventually will lead to malignant brain cancer. An early diagnosis to enable decisive treatment using a Computer-Aided Diagnosis (CAD) system has major challenges, especially accurate detection of different diseases in the magnetic resonance imaging (MRI) images. In this paper, a three-step preprocessing is proposed to enhance the quality of MRI images, along with a new Deep Convolutional Neural Network (DCNN) architecture for effective diagnosis of glioma, meningioma, and pituitary. The architecture uses batch normalization for fast training with a higher learning rate and ease initialization of the layer weights. The proposed architecture is a computationally lightweight model with a small number of convolutional, max-pooling layers and training iterations. A demonstrative comparison between the proposed architecture and other discussed models in this paper is conducted. An outstanding competitive accuracy is achieved of 98.22% overall, 99% in detecting glioma, 99.13% in detecting meningioma, 97.3% in detecting pituitary and 97.14% in detecting normal images when tested on a dataset with 3394 MRI images. Experimental results prove the robustness of the proposed architecture which has increased the detection accuracy of a variety of brain diseases in a short time.

Paper-2: Learning Methods of Convolutional Neural Network Combined With Image Feature Extraction in Brain Tumor Detection

- **Publication Year:** 2020
- **Author:** Weiguang Wang, Fanlong Bu , Ziyi Lin , And Shuangqing Zhai
- **Journal Name:** IEEE Access: The Multidisciplinary Open Access Journal.
- **Summary:** Computer-aided detection technology is less applied in brain tumor detection terminals, and it is difficult to eliminate the influence of various interference factors on the diagnosis results. In order to promote the application of computer-aided detection technology in brain tumor detection, this study based on convolutional neural network, combined with MRI detection technology to construct a model adapted to brain tumor feature detection. The main function of this research model is to segment and recognize MRI brain tumors and use convolutional layer to perform convolution operation to improve recognition efficiency and rate and combine artificially selected features with machine learning features. In addition, this article uses feature fusion to further improve the diagnostic results. Finally, this article designs experiments to perform performance analysis. The research shows that the model algorithm designed in this article has certain practical effects and can provide theoretical reference for subsequent related research.

Paper-3: Design and Implementing Brain Tumor Detection Using Machine Learning Approach

- **Publication Year:** 2019
- **Author:** G.Hemanth, M.Janardhan, L.Sujihelen
- **Journal Name:** Third International Conference on Trends in Electronics and Informatics
- **Summary:** Referring the earlier section, it's revealed that output generated is quiet precise and clear. Accuracy achieved at the end relies upon processing of every step. There are lot of exiting methods for every step, hence the methods that offer better results are selected. At the last, brain tumor classification takes place. To detect brain tumor detection, there exist different classical approaches but the present work utilizes the traditional neural network approach for detecting brain tumor, since the brain tumor detection images relies upon the neighborhood

pixels. The CNN approach provides powerful brain tumor detection. The proposed algorithm is implemented on multiple images and the output retrieved is best and effective.

Chapter 3

PROJECT PLANNING AND MANAGEMENT

3.1 INTRODUCTION

This chapter covers the project planning and management details. It also covers system requirements Specifications. SRS is considered as the base for the effort estimations and project scheduling.

3.2 SYSTEM REQUIREMENTS SPECIFICATIONS

This section gives the detailed description of all types of requirements to be satisfied by the system to be developed.

3.2.1 System Overview

The system that has to be developed is a web application that allows users to submit an MRI image and determine whether or not it contains a tumor.

The system will display both the uploaded image and a preprocessed image once the user uploads data in an image format, such as a jpeg or png file. The user can then decide if a tumor is present or not. The system will then display results that are predicted by a convolutional neural network model.

3.2.2 Functional requirements

- User should be redirected to log in.
- On successful login user should be able to upload image.
- Uploaded image should be analyzed for tumor detection

3.2.3 Main Flow

- Upload the image
- Pre-process the image according to the requirement with the available features.
- Visualize the result.

Chapter 4

TOOLS AND TECHNOLOGIES USED

This project is developed in Jupyter Notebook environment in python programming language.

4.1 Jupyter Notebook:

The Jupyter Notebook is an open-source web application that enables you to make and share documents that contain live code, equations, visualizations, and narrative text.

Uses include: data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning, and much more.

4.2 Python:

Python was the language of selection for this project. This was a straightforward call for many reasons.

1. Python as a language has a vast community behind it. Any problems which may be faced is simply resolved with a visit to Stack Overflow. Python is among the foremost standard language on the positioning that makes it very likely there will be straight answer to any question.

2. Python has an abundance of powerful tools prepared for scientific computing Packages like NumPy, Pandas and SciPy area unit freely available and well documented. Packages like these will dramatically scale back, and change the code required to write a given program. This makes iteration fast.

3. Python as a language is forgiving and permits for program that appear as if pseudo code. This can be helpful once pseudo code given in tutorial papers must be enforced and tested. Using python this step is sometimes fairly trivial. However, Python is not without its errors. The language is dynamically written and packages are area unit infamous for Duck writing. This may be frustrating once a package technique returns one thing that, for instance, looks like an array instead of being an actual array. Plus, the actual fact that standard Python documentation does not clearly state the return type of a method, this can lead to a lot of trials and error testing that will not otherwise happen in a powerfully written language. This is a problem that produces learning to use a replacement Python package or library more difficult than it otherwise may be.

4.3 Python Libraries

Python has prebuilt libraries to make model building an easy job. Python libraries used in this project:

OS:

The OS module in Python provides functions for interacting with the operating system. Python OS module provides the facility to establish the interaction between the user and the operating system. OS comes under Python's standard utility modules.

This module provides a portable way of using operating system-dependent functionality. The *os* and *os.path* modules include many functions to interact with the file system. It offers many useful OS functions that are used to perform OS-based tasks and get related information about operating system. This module offers a portable way of using operating system dependent functionality.

The Python OS module lets us work with the files and directories.

CV2:

OpenCV is a Python library that is used to solve computer vision problems. Computer vision include understanding and analyzing digital images by the computer and process the images or provide relevant data after analyzing the image.

OpenCV is an open-source library used in machine learning and image processing.It performs tasks such as recognizing handwritten digits, human faces and objects.

PIL:

The Python Imaging Library adds image processing capabilities to your Python interpreter. This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.

It incorporates lightweight image processing tools that aids in editing, creating and saving images. Support for Python Imaging Library got discontinued in 2011, but a project named pillow forked the original PIL project and added Python3.x support to it. Pillow was announced as a replacement for PIL for future usage. Pillow supports a large number of image file formats including BMP, PNG, JPEG, and TIFF. The library encourages adding support for newer formats in the library by creating new file decoders.

The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.

Numpy:

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It is open-source software. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Sci-kit learn:

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Important features of scikit-learn:

- Simple and efficient tools for data mining and data analysis. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means, etc.
- Accessible to everybody and reusable in various contexts.
- Built on the top of NumPy, SciPy, and matplotlib.
- Open source, commercially usable – BSD license.

Tensorflow:

TensorFlow is an open-source machine learning framework for all developers. It is used for implementing machine learning and deep learning applications. To develop and research on fascinating ideas on artificial intelligence, Google team created TensorFlow. TensorFlow is designed in Python programming language, hence it is considered an easy-to-understand framework.

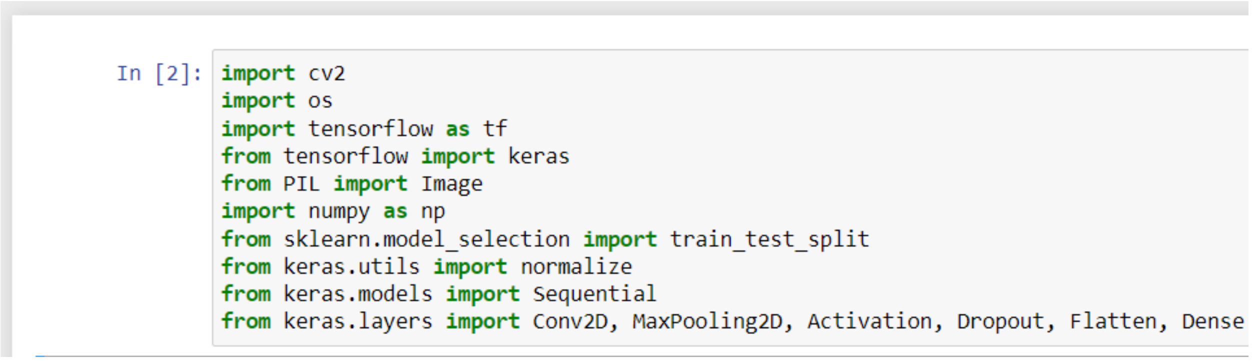
Keras is compact, easy to learn, high-level Python library run on top of TensorFlow framework. It is made with focus of understanding deep learning techniques, such as creating layers for neural networks maintaining the concepts of shapes and mathematical details. The creation of freamework can be of the following two types −

- Sequential API
- Functional API

Consider the following eight steps to create deep learning model in Keras −

- Loading the data
- Preprocess the loaded data
- Definition of model
- Compiling the model
- Fit the specified model
- Evaluate it
- Make the required predictions
- Save the model

This brain tumor detection model is built using the sequential model of Keras library.



```
In [2]: import cv2
        import os
        import tensorflow as tf
        from tensorflow import keras
        from PIL import Image
        import numpy as np
        from sklearn.model_selection import train_test_split
        from keras.utils import normalize
        from keras.models import Sequential
        from keras.layers import Conv2D, MaxPooling2D, Activation, Dropout, Flatten, Dense
```

4.4 USER APPLICATION

HTML:

The most used markup language for making web pages and web applications is HTML (Hypertext Markup Language). It offers a framework for arranging and formatting a webpage's text, graphics, links, and other elements.

To specify various elements and their attributes within a document, HTML uses tags. Angle brackets (>) are used to enclose tags, which normally occur in pairs (an opening tag and a closing tag), though some tags can close on their own. Between the opening and closing tags is where the element's content is located.

CSS :

A style sheet language called CSS (Cascading Style Sheets) is used to describe the display and layout of an HTML or XML document. By defining how elements should be styled, such as their colours, fonts, spacing, and positioning, it offers a mechanism to govern how web pages look.

CSS operates by connecting style directives to HTML components. The characteristics and values that should be applied to the chosen elements are specified by these style guidelines. A selector and a declaration block make up the CSS rules.

JAVASCRIPT:

Programming language JavaScript enables you to dynamically add behaviour and interactivity to web pages. Creating interactive features, validating forms, manipulating PHP updating material on the page, managing events, and communicating with servers are the main uses of it in web development.

Web browsers run JavaScript code, enabling the development of interactive components and improving user experience. It can be included as distinct JavaScript files and linked to HTML using the script src="file.js">/script> tag, or it can be directly embedded into HTML documents using script> tags

Bootstrap :

Bootstrap is a popular open-source front-end framework used for designing and developing responsive and mobile-first websites. It provides a set of pre-built CSS and JavaScript components, such as navigation bars, buttons, forms, grids, and more, that can be easily integrated into web projects.

Chapter 5
ANALYSIS & DESIGN

5.1 Introduction

This chapter covers the analysis and design of the considered system.

5.1.1 CNN Algorithm

A Convolutional Neural Network, or CNN, is a deep learning neural network sketched for processing structured arrays of data such as portrayals. CNN are very satisfactory at picking up on design in the input image, such as lines, gradients, circles, or even eyes and faces. This characteristics that make CNN so robust for computer vision. CNN can run directly on an underdone image and do not need any preprocessing.

A CNN is a forward neural network, seldom with up to 20. The strength CNN comes from a particular kind of layer called the convolutional layer. CNN contain many convolutional layers assembled on top of each other each one competent of recognizing more sophisticated shapes. With three or four convolutional layer it is viable to recognize handwritten digit and with 25 layer it is possible to differentiate human faces.

The agenda for this sphere is to activate machine to view the world as humans do, perceive it in a alike fashion and even use the knowledge for a multitude of duty such as image and video recognition, image inspection and classification, media recreation, recommendation systems, natural language processing, etc.

Layer of CNN model:

1. Convolution 2D
2. MAX Poolig2D
3. Dropout

4. Flatten
5. Dense
6. Activation

Convolution 2D: In the Convolution 2D extract the featured from input image. It given the output in matrix form.

MAX Poolig2D: In the MAX polling 2D it takes the largest element from rectified feature map.

Dropout: Dropout is randomly selected neurons are ignored during training.

Flatten: Flatten feed output into fully connected layer. It gives data in list form.

Dense: A Linear operation in which every input is connected to every output by weight. It followed by nonlinear activation function.

Activation: It used Sigmoid function and predict the probability 0 and 1.

5.2 Representation

- Run the app
- Open the home page
- There are two options available after opening the home page: log in and sign up.
- You do not need to sign up if the email already registered.
- If the email is not register you need to sign up
- Then upload the image
- After uploading the image then the image will display the result
- Check the result
- Then the result will display

5.3 System Architecture

Convolutional Neural Network Model:

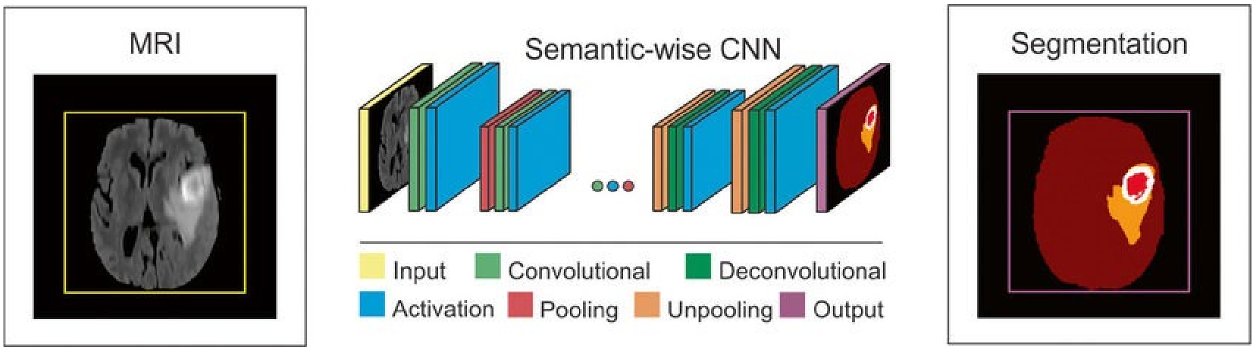


Fig 5.1 System Architecture

The steps to create this project are as follows:

1. Image Acquisition
2. Image Pre-Processing
3. Segmentation
4. Feature Extraction
5. Sampling Data into Training and Testing sets
6. Model Building
7. Model Evaluation

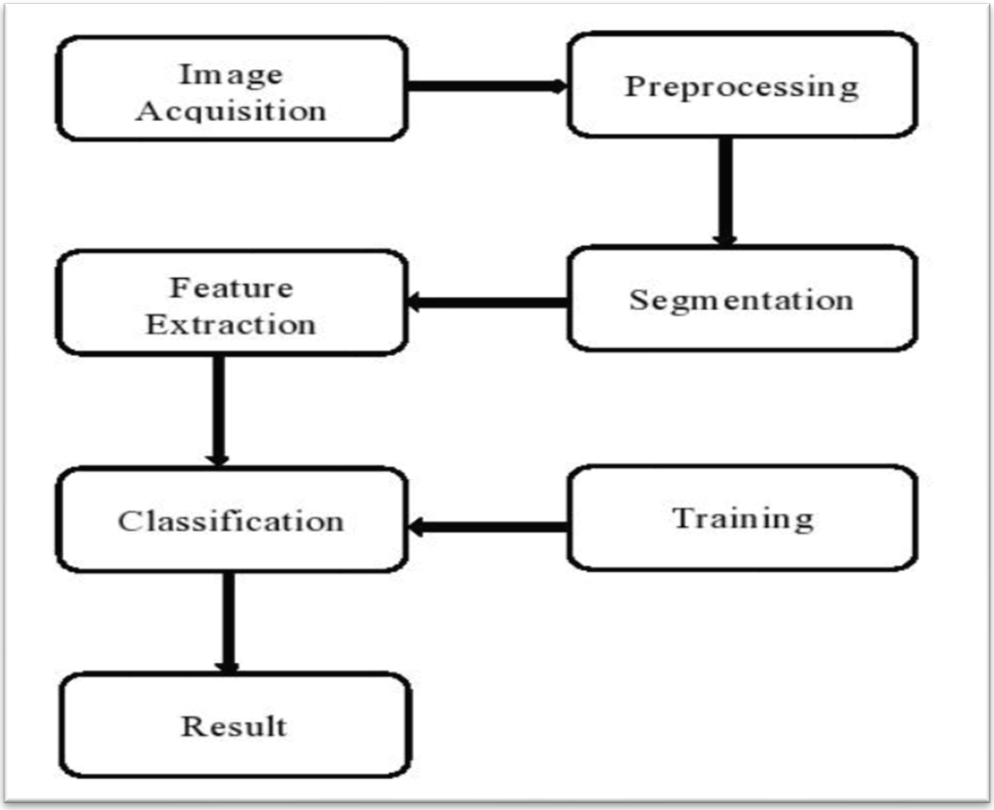


Fig 5.2 System Flow Chart

5.3.1 Use Case

A use case diagram, which is frequently complemented by other types of diagrams, displays the numerous use cases and user types the system has.

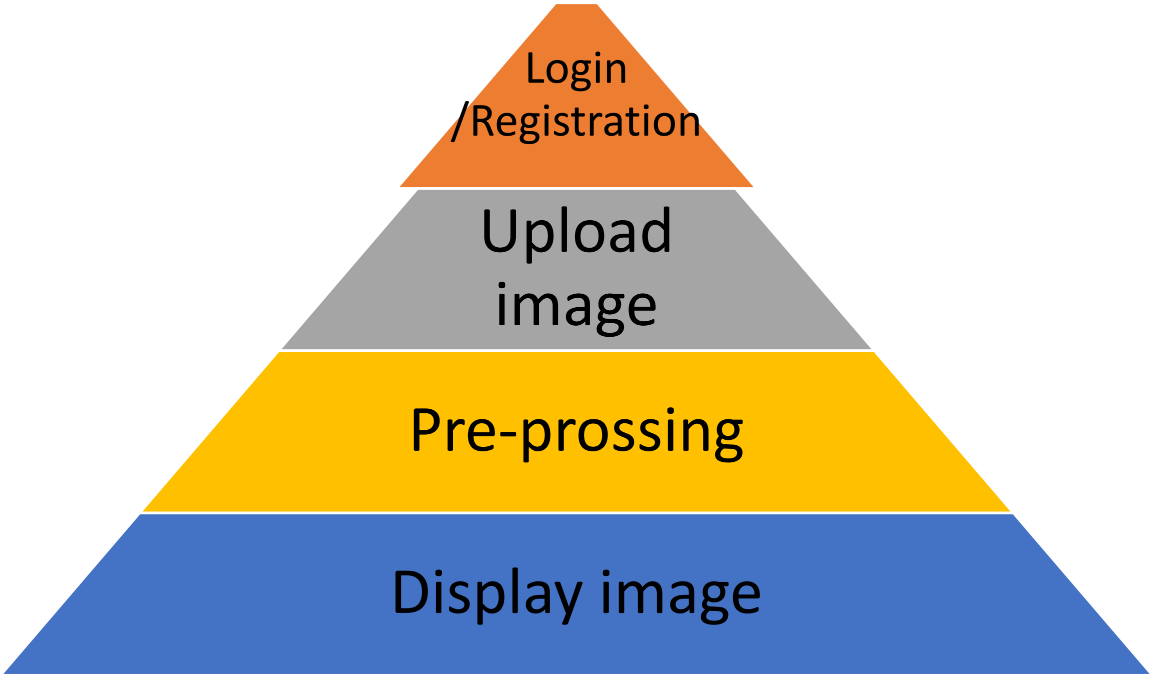


Fig 5.3 Use case

5.3.2 Activity Diagram

Activity diagrams are visual depictions of workflows with choice, iteration, and concurrency supported by activities and actions.

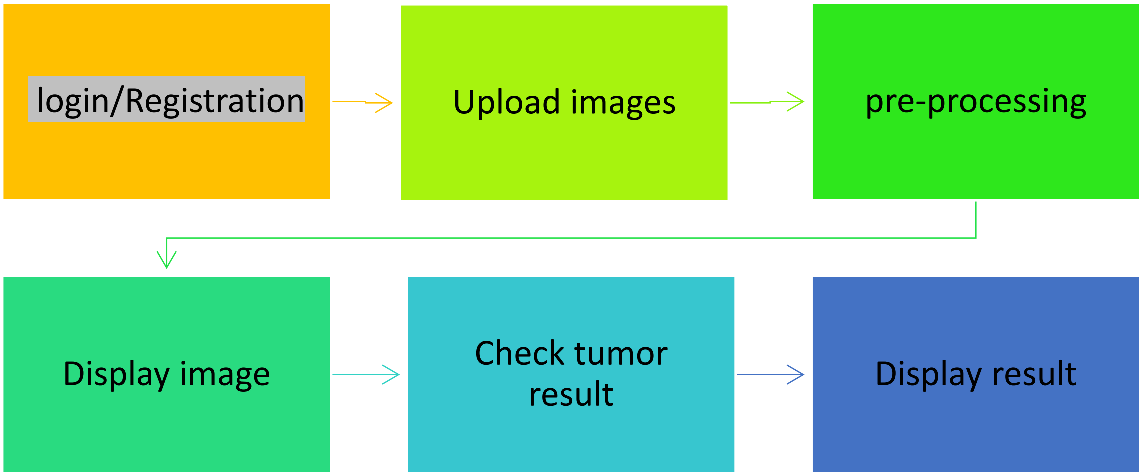


Fig 5.4 Activity Diagram

5.4.1 Image Acquisition

In this project a dataset from www.kaggle .com is used. The name of the data set is "**Br35H :: Brain Tumor Detection 2020**". This dataset contains two sets of MRI images. Each set has 1500 images. One set contains brain MRI reports with brain tumor present and the other one has health brain scans.

As this data contains a total of 3000 images out of which exactly 50% are tumor less, this is balanced data and best suited for training the model.

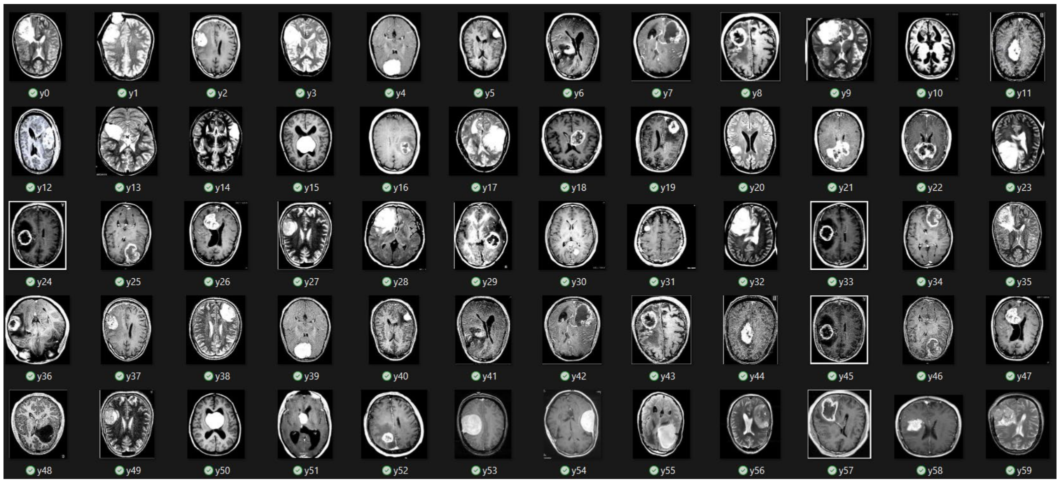


Fig 5.5: Dataset

5.4.2 Image Pre-Processing

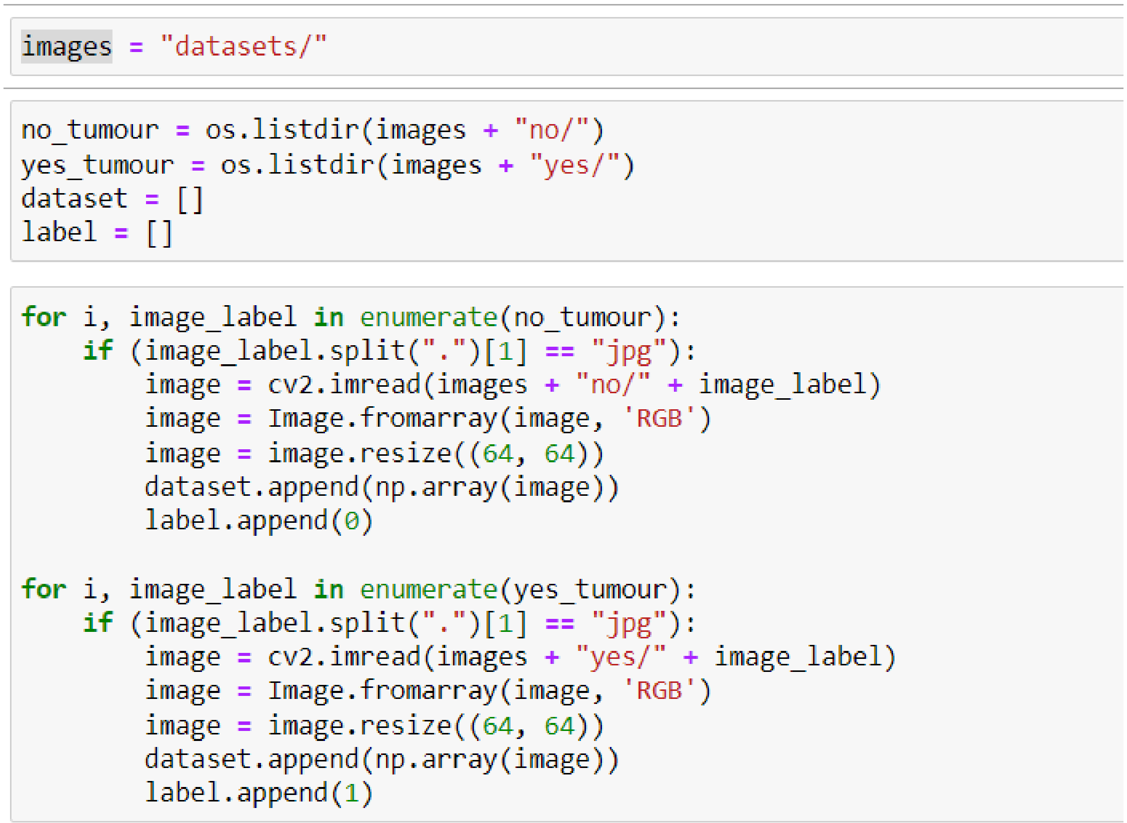
First the images need to be loaded into the python environment. For fetching the file names of all the images, we have used the os library. The imread(method of cv2 library is used to read the image where each pixel is assigned a value based on its weight.

The aim of the pre-processing step is preparing the brain images for further processing.

This process mainly depends on the data acquisition device which has its own intrinsic parameters. Gray scale or 2D conversion is needed, if the raw data is in 3D. Median filtering is best suited for biomedical images to avoid noise. The dataset contains images in different resolutions. As part of the augmentation process, each image is rotated and scaled to a standard format.

So, we have converted each image into 64 by 64 pixels, and also converted these raw images to RGB format using the PIL library for better segmentation.

Once are images are scaled and converted to an array format, we have assigned 0 to images with no tumor and 1 to images with brain tumor. The following code snippet shows how preprocessing needs to be done.



```
images = "datasets/"
```

```
no_tumour = os.listdir(images + "no/")
yes_tumour = os.listdir(images + "yes/")
dataset = []
label = []
```

```
for i, image_label in enumerate(no_tumour):
    if (image_label.split(".")[1] == "jpg"):
        image = cv2.imread(images + "no/" + image_label)
        image = Image.fromarray(image, 'RGB')
        image = image.resize((64, 64))
        dataset.append(np.array(image))
        label.append(0)

for i, image_label in enumerate(yes_tumour):
    if (image_label.split(".")[1] == "jpg"):
        image = cv2.imread(images + "yes/" + image_label)
        image = Image.fromarray(image, 'RGB')
        image = image.resize((64, 64))
        dataset.append(np.array(image))
        label.append(1)
```

Fig 5.6 Pre-processing implementation

5.4.3 Segmentation

In this step a digital image is partitioned into multiple segments. A particular region of the image is being separated from the background This step is very important for preprocessing Classification using CNN Segmentation. Thresholding and morphological operations (erosion, dilation, opening) are the simple steps to segment disease. But in the brain tumor images, the segmentation process at this level will not give the details of tumor regions.

The healthy images also have a similar intensity that resembles the tumor region. So, the segmentation process can be used to separate the skull of the brain. This Region of Interest (ROI) contains the tumor. OTSU based thresholding algorithm gives a segmented mask of the skull. Active contour method draws the boundary of the enclosed region. Second stage of segmentation can also be applied to the ROI to prepare the mask of tumor region. This method may not give good results in healthy images. This segmented image can be used to study the features of tumor region, which will help in the density estimation.

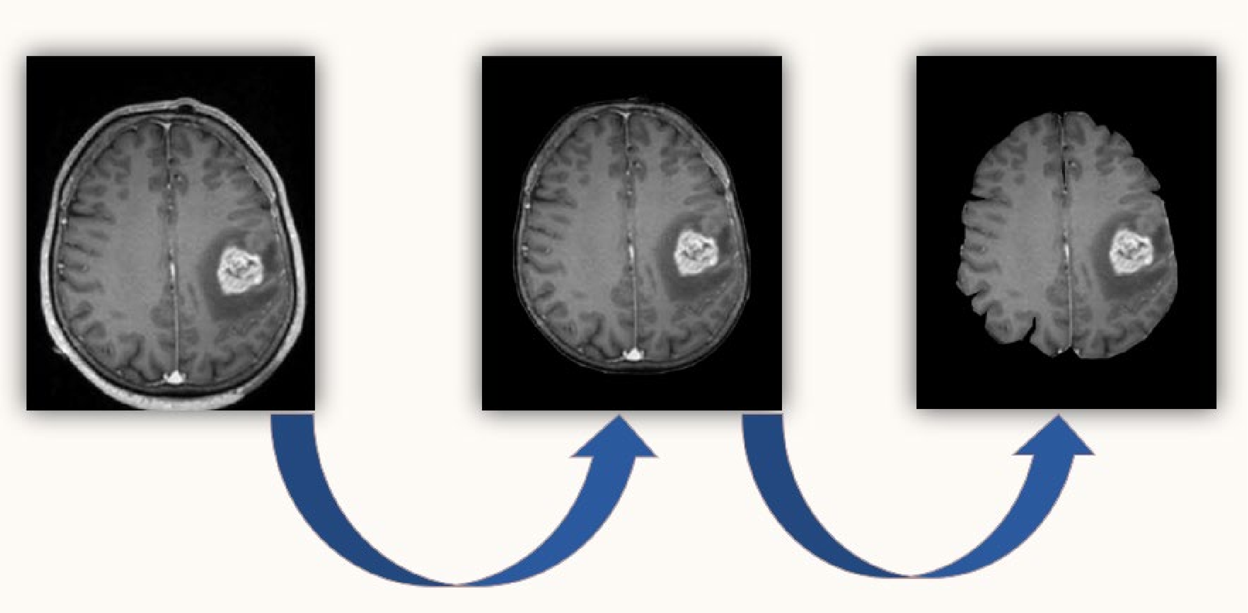


Fig 5.7 Image Segmentation

5.4.4 Feature Extraction

Computing the actual features can be analyzed to illustrate the behavior or symptom of the disease. The classification is mainly influenced with the feature selection. Common features are asymmetry, diameter, and border irregularity.

After applying the necessary convolution layers to the image and after segmentation is completed, only the main features are left in the image. This is shown in the following image.

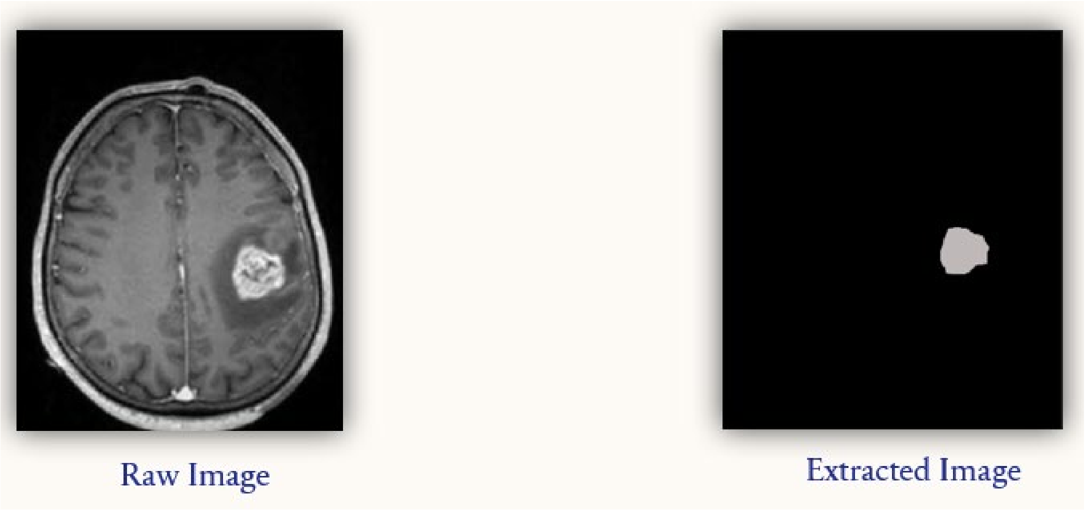
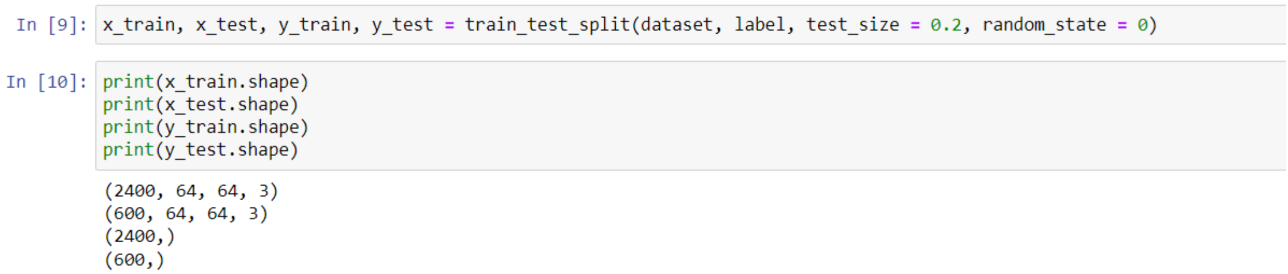


Fig 5.8 Feature extraction

5.4.5 Sampling Data into Training and Testing sets

To divide the data into training and testing sets efficiently, we have used the train_test_split module from sklearn library. The ratio of training to testing set is 80 : 20.



```
In [9]: x_train, x_test, y_train, y_test = train_test_split(dataset, label, test_size = 0.2, random_state = 0)

In [10]: print(x_train.shape)
         print(x_test.shape)
         print(y_train.shape)
         print(y_test.shape)

         (2400, 64, 64, 3)
         (600, 64, 64, 3)
         (2400,)
         (600,)
```

Fig 5.9 Splitting the dataset

5.4.6 Model Building

In this step we have built a CNN based model using the keras library of Tensorflow framework.

This model adds three convolutional layers using the Rectified Linear Unit (ReLU) activation function. The rectified linear activation function or ReLU is a non-linear function or piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It is the most commonly used activation function in neural networks, especially in Convolutional Neural Networks (CNNs) & Multilayer perceptrons.

It is simple yet it is more effective than its predecessors like sigmoid or tanh.

Graphically, it is represented as;

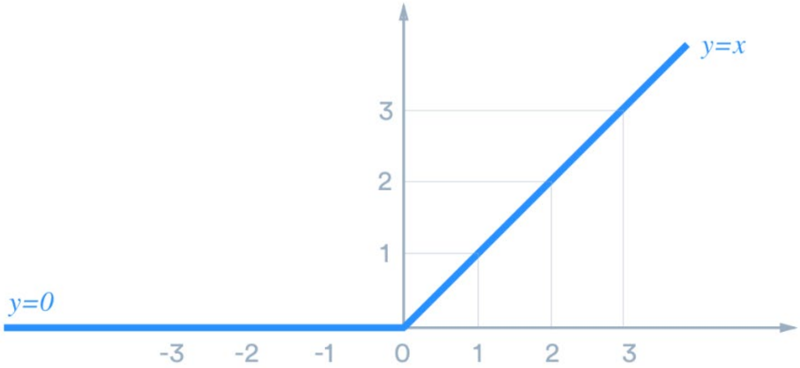
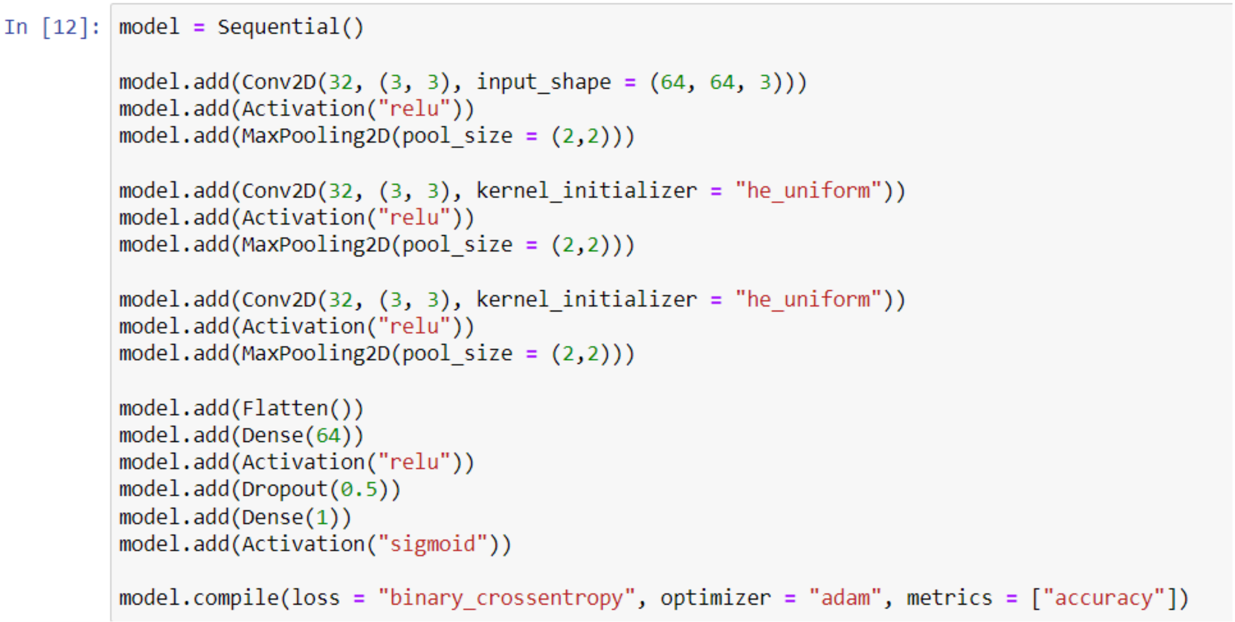


Fig 5.10 ReLU Activation Function



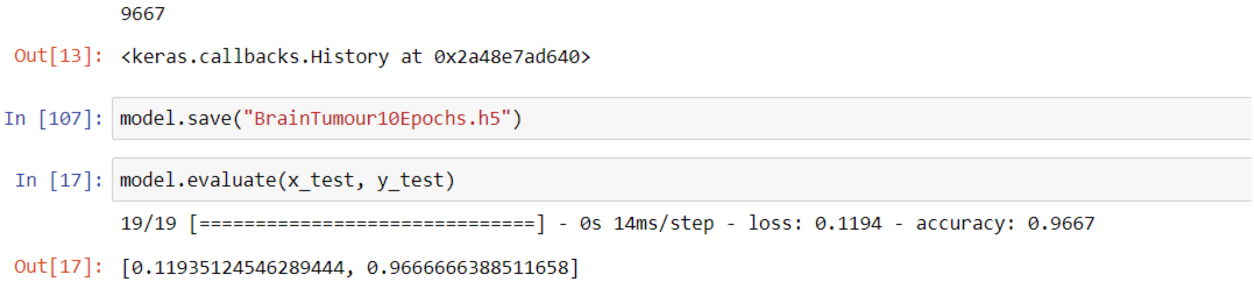
```
In [12]: model = Sequential()

         model.add(Conv2D(32, (3, 3), input_shape = (64, 64, 3)))
         model.add(Activation("relu"))
         model.add(MaxPooling2D(pool_size = (2,2)))

         model.add(Conv2D(32, (3, 3), kernel_initializer = "he_uniform"))
         model.add(Activation("relu"))
         model.add(MaxPooling2D(pool_size = (2,2)))

         model.add(Conv2D(32, (3, 3), kernel_initializer = "he_uniform"))
         model.add(Activation("relu"))
         model.add(MaxPooling2D(pool_size = (2,2)))

         model.add(Flatten())
         model.add(Dense(64))
         model.add(Activation("relu"))
         model.add(Dropout(0.5))
         model.add(Dense(1))
         model.add(Activation("sigmoid"))

         model.compile(loss = "binary_crossentropy", optimizer = "adam", metrics = ["accuracy"])
```

Fig 5.11 CNN Model

5.4.6 Model Evaluation

There are many evaluation metrics to evaluate the model. We make use of accuracy to evaluate the model. Accuracy: Accuracy shows how well the model predicts the class. Mathematically accuracy is defined as the truly predicted instances divided by the total number of instances.

The test accuracy of this model is 96.66%



```
            9667
Out[13]:    <keras.callbacks.History at 0x2a48e7ad640>

In [107]:   model.save("BrainTumour10Epochs.h5")

In [17]:    model.evaluate(x_test, y_test)
            19/19 [==============================] - 0s 14ms/step - loss: 0.1194 - accuracy: 0.9667
Out[17]:    [0.11935124546289444, 0.9666666388511658]
```

Fig 5.12 Model Evaluation

CONCLUSION

In brain tumor detection we have studied about image processing techniques likes image pre-processing, image segmentation, features extraction, classification and also deep learning technique- CNN.

In recent years, magnetic resonance imaging (MRI) has proven to be an excellent technique for clinical research, with applications such as brain tumor detection. When applied to these MRI pictures, deep learning algorithms aid in the detection of the tumor. In this system we have detected that a tumor is present or not. If the tumor is present then model return's yes otherwise it return's no.

This system can be improved to support with a web interface. Detection of different diseases can be also identified from the MRI images. In the healthcare field, the system will be quite beneficial. The proposed approach can determine whether or not a tumor exists. However, in the future, the system could be improved to recognize a specific type of tumor and provide therapy accordingly.

REFERENCE

1. Avigyan Sinha, Aneesh R P, Malavika Suresh, Nitha Mohan R, Abinaya D, Ashwin G Singerji- "Brain tumor Detection Using Deep Learning" 2021 Seventh International conference on Bio Signals, Images, and Instrumentation (ICBSII)

2. Dr. Chinta Someswararao, R Shiva Shankar, Sangapu Venkata Appaji, VMNSSVKR Gupta- "Brain Tumor Detection Model from MR Images using Convolutional Neural Network" IEEE ICSCAN 2020

3. G.Hemanth1, M.Janardhan2, L.Sujihelen- "DESIGN AND IMPLEMENTING BRAIN TUMOR DETECTION USING MACHINE LEARNING APPROACH" Proceedings of the Third International Conference on Trends in Electronics and Informatics (ICOEI 2019

4. www.kaggle.com