A Project Report on

Brain Tumor Detection Using Deep Learning

Submitted in partial fulfillment for award of

Bachelor of Technology

Degree

in

Computer Science and Engineering

By

Y. Mani Sudheer (Y20ACS589) **P. Karunakar (Y20ACS539)**

S. S. V. S. Eswar (Y20ACS554) **R. Pavan (Y20ACS544)**

Under the guidance of

Mr. S. Naga Chandra Sekhar, **M. Tech.**

Assistant Professor

Department of Computer Science and Engineering

Bapatla Engineering College

(Autonomous)

(Affiliated to Acharya Nagarjuna University)

BAPATLA – 522 102, Andhra Pradesh, INDIA

2023-2024

Department of
Computer Science and Engineering

CERTIFICATE

This is to certify that the project report entitled **Brain Tumor Detection Using Deep Learning** that is being submitted by Y. Mani Sudheer (Y20ACS589), P. Karunakar (Y20ACS539), S. S. V. S. Eswar (Y20ACS554), R. Pavan (Y20ACS544) in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering to the Acharya Nagarjuna University is a record of bonafide work carried out by them under our guidance and supervision.

Date:

Signature of the Guide
Mr. S. Naga Chandra Sekhar
Assistant Professor

Signature of the HOD
Dr. M. Rajesh Babu
Associate Professor

DECLARATION

We declare that this project work is composed by ourselves, that the work contained herein is our own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Y. Mani Sudheer (Y20ACS589)

P. Karunakar (Y20ACS539)

S. S. V. S. Eswar (Y20ACS554)

R. Pavan (Y20ACS544)

ACKNOWLEDGEMENT

We sincerely thank the following distinguished personalities who have given their advice and support for successful completion of the work.

We are deeply indebted to our most respected guide **Mr. S. Naga Chandra Sekhar**, Ass.Prof. Department of CSE, for his valuable and inspiring guidance, comments, suggestions and encouragement.

We extend our sincere thanks to **Dr. M. Rajesh Babu**, Assoc. Prof. & Head of the Dept. for extending his cooperation and providing the required resources.

We would like to thank our beloved Principal **Dr. Nazeer Shaik** for providing the online resources and other facilities to carry out this work.

We would like to express our sincere thanks to our project coordinator **Dr. N. Sudhakar,** Prof. Dept. of CSE for his helpful suggestions in presenting this document.

We extend our sincere thanks to all other teaching faculty and non-teaching staff of the department, who helped directly or indirectly for their cooperation and encouragement.

Y. Mani Sudheer (Y20ACS589)

P. Karunakar (Y20ACS539)

S. S. V. S. Eswar (Y20ACS554)

R. Pavan (Y20ACS544)

Table of Contents

List of Figures viii

List of Tables ix

Abstract x

1 Introduction 1

1.1 Risk Factors of Brain Tumor 2

1.2 Data Pre-Processing 2

1.2.1 Need of Pre-processing 2

1.3 Feature Extraction 3

1.4 Deep Learning 3

1.5 Convolutional Neural Networks 3

1.6 Transfer Learning 4

1.7 Activation Function 6

1.8 Full Convnet Architecture 6

1.9 Image Classification 7

1.9.1 Pre-Processing 7

1.9.2 Splitting Our Dataset 8

1.9.3 Building Model and Parameter Optimization 8

1.9.4 Testing And Deployment Phase 8

1.10 Underfitting and Overfitting 9

1.10.1 Underfitting 9

1.10.2 Overfitting 9

2 Literature Survey 10

3 Software and Hardware Requirements 12

3.1 Hardware Requirements 12

3.2 Software Requirements 12

3.2.1 Libraries Requirements: 13
4 Methodology 17
4.1 Problem Statement 17
4.2 Overview of Existing System 18
4.3 Proposed System 23
4.4 Data Pre-Processing 24
4.4.1 Cropping Data 24
4.4.2 Augmenting Data 25
4.5 Working of CNN model 26
4.6 Working of VGG16 model 27
5 Design 29
5.1 UML Diagrams 29
5.1.1 Use Case Diagram 30
5.1.2 Activity Diagram 31
5.1.3 Sequence Diagram 32
6 Implementation 33
6.1 Dataset 33
6.2 Importing Data 33
6.3 Pre-processing Data 33
6.4 Model Selection 34
6.4.1 Building CNN Model 34
6.4.2 Using VGG16 for Better Performance 35
6.5 Saving Model 37
6.6 Web Interface 37
7 Results 39
7.1 Pre-processing Results 39
7.1.1 Data Cropping Results 39

7.1.2 Data Augmentation 40
7.1.3 Model Training 41
7.1.4 Fitting Model 42
7.2 Metrics for Evaluation 43
7.2.1 Predicting Image 45
8 Conclusion 46
9 Limitations and Future Scope 47
10 Bibliography 48

List of Figures

Figure 1.1 Brain Magnetic Resonance Images 1

Figure 1.2 Convolution Neural Networks.....4

Figure 1.3 Full Convnet Architecture 7

Figure 1.4 Underfitting Vs Overfitting 9

Figure 4.1 Existing Workflow of Brain Tumor Detection. [8] 18

Figure 4.2 Proposed Workflow of Brain Tumor Detection 23

Figure 4.3 Cropping the Brain Image Counter Position 25

Figure 4.4 Augmentation Images.....25

Figure 4.5 Working of CNN Model for Brain Tumor Detection.....26

Figure 4.6 VGG16 Layered Architecture 27

Figure 4.7 Working of VGG16 Model for Brain Tumor Detection 28

Figure 5.1 Use case Diagram 30

Figure 5.2 Activity Diagram 31

Figure 5.3 Sequence Diagram.....32

Figure 6.1 Flask Framework.....38

Figure 7.1 Cropping the MRI Scan Images 39

Figure 7.2 Augmented Data Images 40

Figure 7.3 Train CNN image data.....42

Figure 7.4 Test CNN image data 42

Figure 7.5 Train VGG16 image data 42

Figure 7.6 Test VGG16 image data 43

Figure 7.7 Confusion Matrix for the Proposed CNN and VGG16 model 44

List of Tables

Table 1 Existing CNN Model Summary.....22

Table 2 Proposed CNN model Summary.....41

Table 3 Proposed VGG16 model Summary 41

Table 4 Comparison CNN Accuracy with VGG16 Accuracy 43

Abstract

A brain tumor is a disease caused due to the growth of abnormal cells in the brain. There are two main categories of brain tumor, they are non-cancerous (benign) brain tumor and cancerous(malignant**)** brain tumor. Survival rate of a tumor prone patient is difficult to predict because brain tumor is uncommon and are different types. As per the cancer research by United Kingdom, around 15 out of every 100 people with brain cancer will be able to survive for ten or more years after being diagnosed.

Treatment for brain tumor depends on various factors like: the type of tumor, how abnormal the cells are and where it is in the brain etc. With the growth of Artificial Intelligence, Deep learning models are used to diagnose the brain tumor by taking the images of magnetic resonance imaging. Magnetic Resonances Imaging (MRI) is a type of scanning method that uses strong magnetic fields and radio waves to produce detailed images of the inner body.

The research work carried out uses Deep learning models like convolutional neural network (CNN) model and VGG-16 architecture to detect the tumor region in the scanned brain images. We have considered Brain MRI images of 253 patients, out of which 155 MRI images are tumorous and 98 of them are non-tumorous. The paper presents a comparative study of the outcomes of CNN model and VGG-16 architecture used.

Keywords: Convolutional Neural Network, VGG-16 architecture, medical imaging, Tumor identification, Neural networks.

1 Introduction

Brain tumor is one of the most rigorous diseases in medical science. An effective and efficient analysis is always a key concern for the radiologist in the premature phase of tumor growth. Histological grading, based on a stereotactic biopsy test, is the gold standard and the convention for detecting the grade of a brain tumor. There are many risk factors involving the biopsy test, including bleeding from the tumor and brain causing infection, seizures, severe migraine, stroke, coma and even death. But the main concern with stereotactic biopsy is that it is not 100% accurate which may result in a serious diagnostic error followed by wrong clinical management of the disease.

A brain tumor occurs when abnormal cells form within the brain. There are two main types of tumors: cancerous (malignant) tumors and benign tumors. Cancerous tumors can be divided into primary tumors, which start within the brain, and secondary tumors, which have spread from elsewhere, known as brain metastasis tumors. All types of brain tumors may produce symptoms that vary depending on the part of the brain involved. These symptoms may include headaches, seizures, problems with vision, vomiting and mental changes. The headache is classically worse in the morning and goes away with vomiting. Other symptoms may include difficulty walking, speaking or sensations. As the disease progresses, unconsciousness may occur.

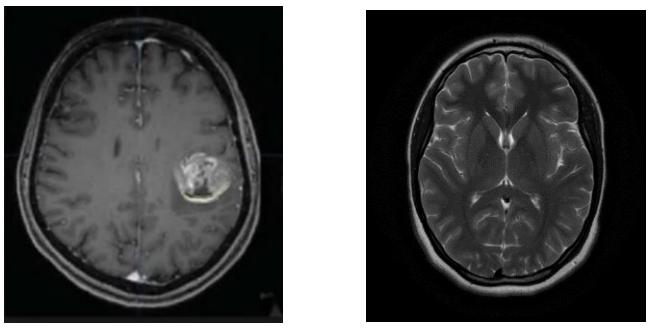


Figure 1.1 Brain Magnetic Resonance Images

1.1 Risk Factors of Brain Tumor

Risk factors of brain tumor are, Ionizing Radiation: Ionizing radiation from high dose x-rays (such as radiation therapy from a large machine aimed at the head) and other sources can cause cell damage that leads to a tumor. People exposed to ionizing radiation may have an increased risk of a brain tumor. such as meningioma or glioma. Family History: It is rare for brain tumors to run in a family. Only a very small number of families have several members with brain tumors.

1.2 Data Pre-Processing

Data pre-processing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behavior or trends, and likely to contain many errors. Data pre-processing uses data-driven data base application such as customer relationship management and rule-based applications.

1.2.1 Need of Pre-processing

a) Real-world data are generally.

b) Incomplete Lacking attribute values, lacking certain attributes of interest, or containing any aggregate values.

c) Noisy-Containing errors and outliers.

d) Inconsistent-Containing discrepancies in codes or names.

1.3 Feature Extraction

Feature extraction redefines a large set of redundant data into a set of reduced dimensions called features. Feature extraction helps to check the resulted values of the parameters with the standard values and differentiate between leukemic and healthy data. In feature extraction, the acquired data from the image is transformed and labelled to a particular set of features, which is going to be used for further classification. The binary equivalent images produced by the segmentation technique of blood cell and cell nucleus are used to extract those morphological features.

1.4 Deep Learning

Deep learning is a method in artificial intelligence (AI) that teaches computers to process data in a way that is inspired by the human brain. Deep learning models can recognize complex patterns in pictures, text, sounds, and other data to produce accurate insights and predictions.

1.5 Convolutional Neural Networks

In neural networks, Convolutional neural network (ConvNets or CNNs) is one of the main categories to do images recognition, images classifications. Object's detections, recognition faces etc., are some of the areas where CNNs are widely used. CNN image classifications take an input image, process it and classify it under certain categories. Computers see an input image as array of pixels, and it depends on the image resolution. Based on the image resolution, it will see h x w x d (h Height, w Width, d - Dimension).

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the

other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics. The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlaps to cover the entire visual area.

The advancements in Computer Vision with Deep Learning have been constructed and perfected with time, primarily over one particular algorithm Convolutional Neural Network.

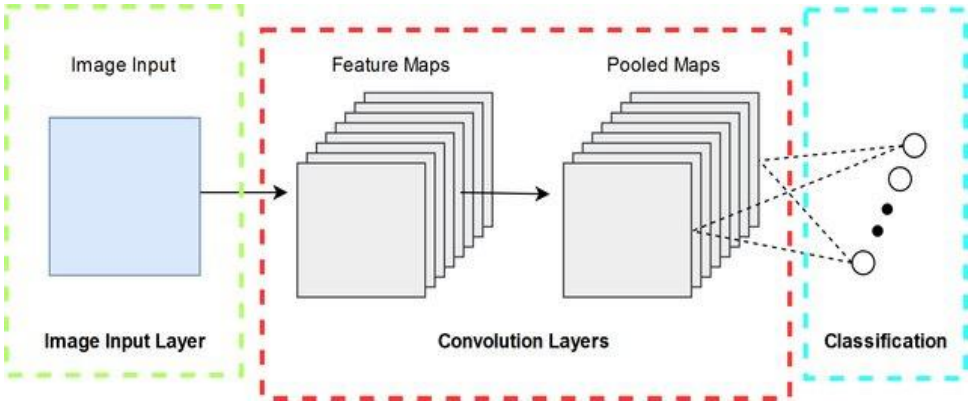


Figure 1.2 Convolution Neural Networks

1.6 Transfer Learning

A major assumption in many machine learning and data mining algorithms is that the training and future data must be in the same feature space and have the same distribution. However, in many real-world applications, this assumption may not hold.

For example, we sometimes have a classification task in one domain of interest, but we only have sufficient training data in another domain of interest, where the latter

data may be in a different feature space or follow a different data distribution. In such cases, knowledge transfer, if done successfully, would greatly improve the performance of learning by avoiding much expensive data labelling efforts. In recent years, transfer learning has emerged as a new learning framework to address this problem. Transfer learning allows neural networks to use significantly less data. With transfer learning, we are in effect transferring the 'knowledge' that a model has learned from a previous task, to our current one. The idea is that the two tasks are not totally disjoint, as such we can leverage whatever network parameters that model has learned through its extensive training, without having to do that training ourselves. Transfer learning has been consistently proven to boost model accuracy and reduce required training time, less data, less time, more accuracy.

Transfer learning is classified to three different settings: inductive transfer learning, transductive transfer learning and unsupervised transfer learning. Most previous works focused on the settings. Furthermore, each of the approaches to transfer learning can be classified into four contexts based on "what to transfer" in learning. They include the instance-transfer approach, the feature-representation-transfer approach, the parameter transfer approach, and the relational-knowledge-transfer approach, respectively.

The smaller networks converged & were then used as initializations for the larger, deeper networks- This process is called pre-training. While making logical sense, pre-training is a very time consuming, tedious task, requiring an entire network to be trained before it can serve as an initialization for a deeper network.

1.7 Activation Function

Sigmoid function ranges from 0 to 1 and is used to predict probability as an output in case of binary classification while Softmax function is used for multi-class classification. tanh function ranges from -1 to 1 and is considered better than sigmoid in binary classification using feed forward algorithm. ReLU (Rectified Linear Unit) ranges from 0 to infinity and Leaky ReLU (better version of ReLU) ranges- from -infinity to +infinity. ReLU stands for Rectified Linear Unit for a non-linear operation. The output is $f(x) = \max(0,x)$.ReLU‘s purpose is to introduce non-linearity in our ConvNet. Since, the real world data would want our ConvNet to learn would be non-negative linear values. There are other nonlinear functions such as tanh or sigmoid that can also be used instead of ReLU. Most of the data scientists use ReLU since performance wise ReLU is better than the other two. Stride is the number of pixels that would move over the input matrix one at a time.

Sometimes the filter does not fit perfectly fit the input image. We have two options: either pad the picture with zeros (zero-padding) so that it fits or drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid part of the image.

1.8 Full Convnet Architecture

A fully connected CNN (Full ConvNet) is built by stacking Convolutional layers (CONV), Pooling layers (POOL) and at the output a Fully-Connected layer (FC). The CONV layer consists of a set of learnable filters or kernels. Each filter slides across the input image and compute the discrete convolution between the filter and the receptive field of the input. If the input has multiple channels or depth (For example an RGB

image), then the convolution outputs are summed together, followed by the non-linearity.

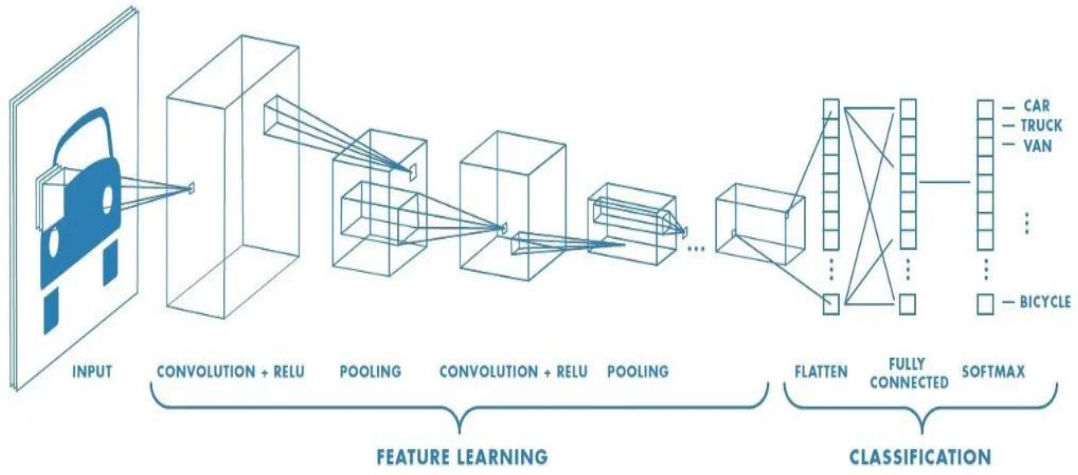


Figure 1.3 Full Convnet Architecture

1.9 Image Classification

Image classification is a critical task with widespread applications across various domains. It enables machines to interpret visual data, aiding in tasks such as medical diagnosis, autonomous driving, security surveillance, and e-commerce. The steps involved in image classification are:

1.9.1 Pre-Processing

To ensure the images are organized and free from noise, it's crucial to perform thorough cleaning. This step is pivotal, especially given the presence of noise and disorganized content in the provided data. Therefore, implementing pre-processing steps becomes essential. Basically, some pre-processing steps for the images are

a) Normalize the pixel values

b) Crop the images

c) Rotation of images

d) Adjust hue, contrast and saturation

1.9.2 Splitting Our Dataset

It takes a long time to calculate the gradient of the model using the entire large dataset. We therefore will use a small batch of images during iteration of the optimizer. The batch size is normally 32 or 64 or 128 since we have fairly many images.

1.9.3 Building Model and Parameter Optimization

Now that we're done pre-processing and splitting our dataset, we can start implementing our neural network. The parameters identification is a challenging task since we have so many parameters to be adjusted. There are lot of parameters to be trained. This is why training a model will take a lot of time. But with the help of the latest hardware such as GPU and TPU we can easily finish our training of neural networks. After training our model is ready for the testing phase.

1.9.4 Testing And Deployment Phase

Now that we have a trained neural network, we can use it for testing. After testing if our model can predict the unseen data successfully then we are ready to deploy the model. Otherwise, this problem can be reduced by optimizing the parameters or by giving more data to the model. Feature extraction is the main phase in machine learning but in deep learning it is a built-in process. The model itself can extract the features of the data in deep learning.

1.10 Underfitting and Overfitting

The problem of underfitting and overfitting finally appears when we talk about the polynomial degree. The degree represents how much flexibility is in the model, with a higher power allowing the model freedom to hit as many data points as possible. An underfit model will be less flexible and cannot account for the data. The best way to understand the issue is to look at models demonstrating both situations.

1.10.1 Underfitting

A statistical method or machine learning algorithm fails to adequately capture the underlying trend within the data, resulting in decreased accuracy. This commonly occurs when there is insufficient data or when attempting to fit linear models to nonlinear data. Underfitting leads to oversimplified rules, resulting in inaccurate predictions.

1.10.2 Overfitting

A statistical model or machine learning algorithm becomes overfitted when trained with excessive data, akin to trying to fit into oversized pants. This abundance of data causes the model to learn from noise and inaccuracies, resulting in incorrect categorization. To prevent overfitting, it's advisable to utilize linear algorithms for linear data or adjust parameters like maximal depth in decision trees.

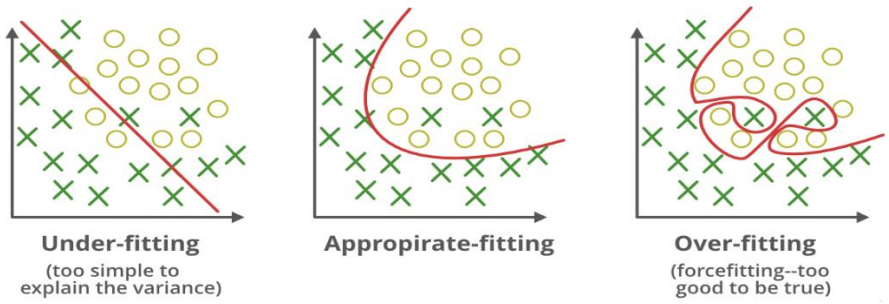


Figure 1.4 Underfitting Vs Overfitting

2 Literature Survey

A study conducted in 2022 [1] introduced a DL architecture modeled after LeNet to distinguish between normal and pathological MRI images, factoring in age and gender. Impressively, this approach yielded an 88% overall Accuracy, surpassing competing models such as CNN-DNN (80%), SVM (82%), and Alex Net (64%).

Researchers implemented an ensemble technique [2], leveraging a combination of convolutional neural networks (CNNs) such as ResNet50, VGG16, and InceptionV3, to enhance brain tumor detection. Each CNN model underwent individual training and was then merged using a voting mechanism, achieving an impressive 94% accuracy rate. This collaborative approach demonstrated superior performance and resilience compared to single-model methodologies, indicating promising avenues for diagnostic advancements in brain tumor detection and classification.

In the study [3], researchers focused on classifying brain tumors using deep learning methods combined with radiomic characteristics. The authors classified tumors using a mixture of a deep neural network (particularly ResNet50) and more typical machine learning methods. The quantitative radiomic characteristics were retrieved from brain MRI data and used. When it came to categorizing brain tumors into their respective subtypes, the suggested method was able to attain an accuracy rate of 88% overall. The use of deep learning and radiomic characteristics resulted in an increase in both the accuracy and discriminatory power of tumor classification, providing helpful insights for the development of personalized treatment plans.

In a significant study [4],researchers embarked on investigating transfer learning by utilizing pre-trained weights from ResNet50 to train a model on datasets of brain MRI images. Remarkably, the model achieved an impressive accuracy of 92% in

detecting brain tumors, owing to the integration of binary cross-entropy loss and gradient descent optimization techniques. This highlights the substantial potential of deep learning methodologies in clinical contexts, exhibiting superior performance compared to conventional methods.

A hybrid model for brain tumor segmentation [5] by integrating the U-Net and ResNet50 architectural frameworks. Initially, U-Net was utilized for coarse segmentation, followed by ResNet50 for fine segmentation. The model was trained on a large dataset of brain MRI images manually annotated by experienced radiologists. Demonstrating remarkable accuracy and precision, the hybrid model achieved a Dice similarity coefficient (DSC) of 0.92 in segmenting brain tumors.

In their comprehensive assessment [6], researchers explore various deep learning approaches applicable to medical image analysis. They delve into different architectures, including CNNs, RNNs, and GANs, and their applications in tasks like picture classification, segmentation, and detection. Encompassing a wide array of medical imaging modalities such as MRI, CT, and microscopy, the survey highlights the utility of deep learning in medical image processing

Researchers explore deep learning methods [7], for brain tumor segmentation, comparing CNNs, UNets, and attention-based models. Challenges like limited data and class imbalance are discussed, while recent advancements are highlighted. The paper emphasizes deep learning's effectiveness, especially with CNN-based architectures and U-Net variations, though challenges such as class imbalance and minor lesion segmentation persist, indicating the need for further development before clinical adoption.

3 Software and Hardware Requirements

3.1 Hardware Requirements

a) Hard Disk : 128GB and Above

b) RAM : 8GB and Above

c) Processor : Intel Core i5 or Ryzen 5

3.2 Software Requirements

a) Operating System : Windows, MacOS

b) Software : Python

c) Tools : Anaconda (Jupiter Notebook IDE)

Python is a free, open-source programming language. Therefore, all you must do is install Python once, and you can start working with it. Not to mention that you can contribute your own code to the community. Python is also a cross-platform compatible language. So, what does this mean? Well, you can install and run Python on several operating systems.

Python is also a great visualization tool. It provides libraries such as Matplotlib, and seaborn to create stunning visualizations. Python packages like NumPy, Matplotlib, Pandas for mathematical computation and plotting graphs. Kaggle was used to obtain the online dataset. GitHub and Stack overflow was used for reference in case of programming syntax errors.

3.2.1 Libraries Requirements:

Library requirements is crucial for project development, as it determines available functionalities, ensures compatibility, streamlines development processes, and fosters reliability and scalability. Clear and well-defined library requirements are essential for building robust, efficient, and sustainable software solutions that meet project objectives effectively.

OpenCV

Open-source Computer Vision is a popular computer vision library started by Intel in 1999. OpenCV is a cross-platform library using which we can develop real-time computer vision applications. OpenCV mainly focuses on image processing, video capture and analysis including features like face detection and object detection. OpenCV uses machine learning algorithms to search for faces within a picture. Because faces are so complicated, there isn't one simple test that will tell you if it found a face or not. Instead, there are thousands of small patterns and features that must be matched.

The algorithms break the task of identifying the f ace into thousands of smaller, bite-sized tasks, each of which is easy to solve. OpenCV mainly deals with following fields:

NumPy

NumPy's main goal is large, efficient, multi-dimensional array representations. This project deals with a lot of images and each image is represented as a NumPy array. Representing images as NumPy arrays is not only computational and resource efficient, but many other image processing, and machine learning libraries use NumPy array representations as well.

Matplotlib

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open-source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications.

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like python, Qt, or GTK.... SciPy makes use of Matplotlib.

Scikit-learn

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Pillow

Python Imaging Library (abbreviated as PIL) (in newer versions known as Pillow) is a free library for the Python programming language that adds support for opening, manipulating, and saving many different images file formats. It is available for Windows, MacOS and Linux.

OS

The OS module in python provides functions for interacting with 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 filesystem.

Keras

Keras stands out as one of the leading high-level neural network APIs, written in Python and compatible with multiple back-end neural network computation engines. It was developed with the intention of being user-friendly, modular, and easy to extend, making it a popular choice among developers. Keras emphasizes simplicity and follows best practices to minimize cognitive load, ensuring a smooth user experience. Its modular design allows users to easily combine neural layers, cost functions, optimizers, initialization schemes, activation functions, and regularization schemes to create new models. Additionally, the flexibility of Keras allows for the straightforward addition of new modules as classes and functions. Importantly, models are defined directly in Python code, eliminating the need for separate model configuration files.

Features:

a) Keras is a high-level interface and uses Theano or TensorFlow for its backend.

b) It runs smoothly on both CPU and GPU.

c) Keras supports almost all the models of a neural network connected, convolutional, pooling, recurrent, embedding. fully etc. Furthermore, these models can be combined to build more complex models.

d) Keras, being modular in nature, is incredibly expressive, flexible, and apt for innovative research.

e) Keras is a completely Python-based framework, which makes it easy to debug and explore.

TensorFlow

TensorFlow is a Python-centric open-source library revolutionizing numerical computation in machine learning. With its intuitive interface, TensorFlow simplifies the training and execution of deep neural networks, powering tasks like handwritten digit classification and image recognition with exceptional efficiency. Moreover, TensorFlow's robust support for advanced models such as word embeddings, recurrent neural networks, and sequence-to-sequence models addresses the complexities of various applications, including machine translation, natural language processing.

One of TensorFlow's notable strengths lies in its seamless integration into production environments, enabling scalable prediction across a multitude of platforms. Whether deploying models on local machines, cloud clusters, or mobile devices like iOS and Android, TensorFlow ensures consistent performance across diverse environments. Additionally, TensorFlow's compatibility with Google Cloud's custom TensorFlow Processing Unit (TPU) provides accelerated processing capabilities, especially advantageous for users leveraging Google's cloud infrastructure. In essence, TensorFlow's versatility empowers users to deploy models across different devices with ease, ensuring accessibility and efficiency in serving predictions for a wide range of machine learning applications.

Imutils

Imutils serve as a collection of convenient functions designed to streamline fundamental image processing tasks, including translation, rotation, resizing, skeletonization, and displaying Matplotlib images. These functions are tailored to simplify image manipulation with OpenCV, ensuring compatibility with both Python 2.7 and Python 3 environments.

4 Methodology

The methodology aims to provide a comprehensive framework for developing an automated brain tumor detection system, leveraging advanced deep learning techniques to address the challenges associated with manual classification and improve diagnostic accuracy.

4.1 Problem Statement

The brain tumors, are the most common and aggressive disease, leading to a very short life expectancy in their highest grade. Thus, treatment planning is a key stage to improve the quality of life of patients. Generally, various image techniques such as Computed Tomography (CT), Magnetic Resonance Imaging (MRI) and ultrasound image are used to evaluate the tumor in a brain, lung, liver, breast, prostate...etc. Especially, in this work MRI images are used to diagnose tumor in the brain. However, the huge amount of data generated by MRI scan thwarts manual classification of tumor vs non-tumor in a particular time. But it has some limitation (i.e.) accurate quantitative measurements are provided for a limited number of images.

Hence trusted and automatic classification scheme are essential to prevent the death rate of human. The automatic brain tumor classification is a very challenging task in large spatial and structural variability of surrounding region of brain tumor. In this work, automatic brain tumor detection is proposed by using Convolutional Neural Networks (CNN) classification. The deeper architecture design is performed by using small kernels. The weight of the neuron is given as small. Experimental results show that the deep learning models archive high accuracy with low complexity and compared with the all-other state of arts methods.

4.2 Overview of Existing System

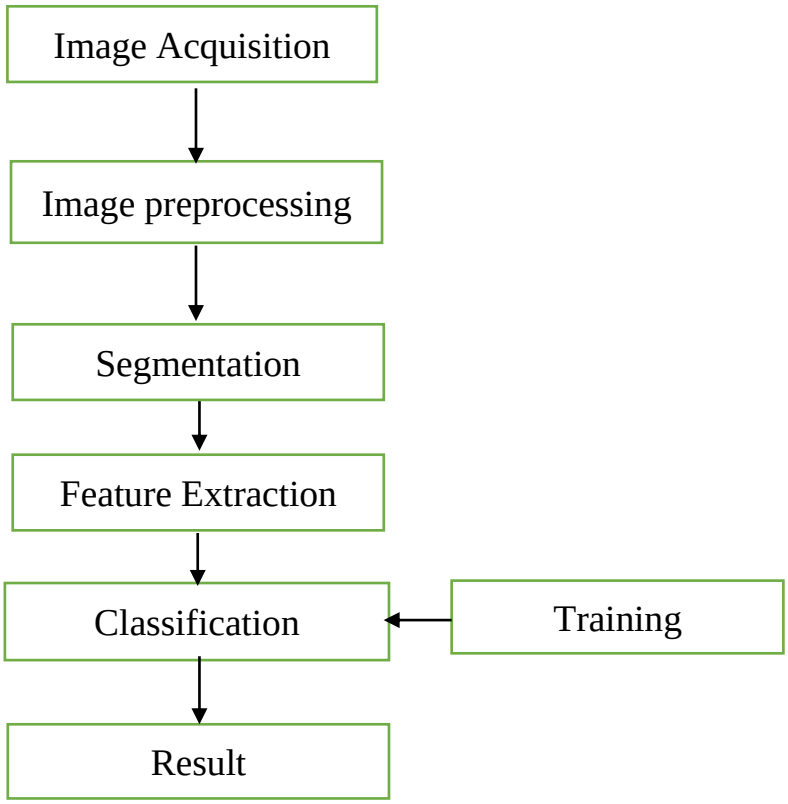


Figure 4.1 Existing Workflow of Brain Tumor Detection. [8]

Image Acquisition

Every digital image processing task begins with obtaining a unique input image from the source that provided it (where an image may be discovered). There are actually numerous methods to utilize image gathering. When thinking about this step, the only thing that comes to mind is capturing a photo of the actual surroundings, however it is also feasible to obtain an existing image file from any electronic source using any gathering approach. When viewed from the side of the intended image processing project, the acquired image remains completely unprocessed. Having a consistent starting point can be quite important in several industries. It was created using the same hardware that created the original image.

Different sources are utilized to retrieve raw photographs depending on the requirement or type of operation. Diagnostic imaging apparatus, such as CT and MRI machines, are devices that can be used to obtain images for diagnostic purposes, such as the detection of tumor and cancer. Satellites can also supply images for the administration of geographic data and related applications. Magnetic resonance imaging was used to collect the data for the investigation.

Image Preprocessing

A picture may be preprocessed using filters, formatting, removing artefacts, and shade correction, among other processes, to improve image data before computing processing. The purpose of filtering is to enhance the stability of the image's features by lowering a small amount noise in the surroundings, adjusting the intensity of each particle image, reducing reflections, and concealing portions of the images [9]. Images can be scaled to make a smaller, more compact version of the initial image for easier processing, but this is only advised if the resulting image retains all the original's meaning and appearance after scaling. Although it is physically conceivable, processing of digital images is the most used method. A few of the tasks involved in image preparation are described below.

Image conversion: This is a part of preprocessing, which may entail changing an image's format or color space to accelerate computation. For instance, because they are simpler to analyses than those with wider color gamut, such as RGB or other sorts, colorful picture graphs are routinely turned into colorless or black and white images. image transformation is not often essential, nevertheless; it could potentially be omitted in circumstances when the results might change how an image is interpreted. In this case, compared to graphs composed of monochrome pictures, colored graphics offer

more detailed information. Maintaining the color image in its original state or converting it back to color is essential for the aim of detecting brain cancer illnesses.

Image augmentation: A phase in the editing process where several approaches are used to give an image a further, more significant meaning. Contrast adjustment is the most widely used method for enhancing photographs. Improving the visual impact of an image is the major aim of the picture rehabilitation sub-step. In contrast to picture augmentation, which is subjective, picture restoration is objective. This is due to the fact that methods for photo restoration usually incorporate probabilistic or mathematical models of image deterioration.

Image enhancement: Image enhancement is the technique of enhancing an image's visual appeal. Before photos are fed into a neural network in deep learning, they are first preprocessed using image enhancing methods. By lowering noise, boosting contrast, and enhancing the visual quality of the input image, image enhancement in deep learning aims to increase the accuracy and robustness of the neural network [10].

Image Segmentation

In the context of digital image processing, image segmentation refers to any method that splits or arranges an image into distinct pertinent chunks or segments. A variety of data are used to construct the picture segmentation approach. This can be a portion of a picture, color data, or border data. The reliability of extraction of features and the accuracy of identification are closely related to the quality of the segmentation result. The fields of image processing and computer vision today use a range of image segmentation methods based on the difficulty or image that needs to be segmented. There are benefits and drawbacks to each strategy.

The two primary strategies for segmentation are region-based and edge-based. A picture is separated into portions for area-based approaches to segmentation based on how similar the pixel values are inside a region, for as in terms of color, texture, or intensity. Edge-based approaches divide a picture into portions based on pixel value shifts or discontinuities, such as edges or division lines.

Feature Extraction

Feature extraction is a process of extracting meaningful features from raw input data, such as images, audio, or text that can be used as inputs to a machine learning model. In deep learning, feature extraction involves using a pre-trained neural network to extract high level features from the input data, which can then be used as inputs to a new neural network. For analysis utilizing a convolution tool, pooling, or stride, feature extraction is the process of extracting and noticing various elements of an image, such as lines, shapes, and margins.

According to the literature analysis and related studies, most brain MRI classification tasks that are carried out using deep learning approaches use GLCM and GLRLM algorithms to extract texture characteristics. Although CNN was used to divide the data into normal and pathological categories, transfer learning and CNN techniques were linked. The main argument in favor of utilizing CNN in deep learning is the employment of several extraction of features processes, which can produce a representation of data autonomously from input data. Due to variables including the availability of a lot of data and improvements in hardware technology, CNN's research has expanded. A variety of innovative CNN designs have also been described recently. These structures, among other things, made use of diverse activation and loss functions, parameter optimization, and new architectural techniques.

Classification

Many machine learning approaches are being implemented in disease detection from brain images. Artificial neural networks can be used here to classify if the features are extracted in an order. An ANN classifier assumes one feature that is not related to any other feature. Deep learning techniques will be effective here to classify tumor image without segmentation. A deep neural network can be created with a Convolutional neural network algorithm. General architecture of convolutional neural networks is shown in figure 6. In deep learning, the feature is extracted from the entire image automatically. Convolution in the CNN architecture performs this operation. The number of feature maps increases with the increase in CONV layer.

Reduction of dimension is required to initiate training. Pooling layer down samples the feature dimension. Fully connected layers manipulate the score of each label. SoftMax layers prepare the model with feature and class score. The CNN architecture is slightly modified in its dimension for training the brain tumor images.

Table 1 Existing CNN 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

4.3 Proposed System

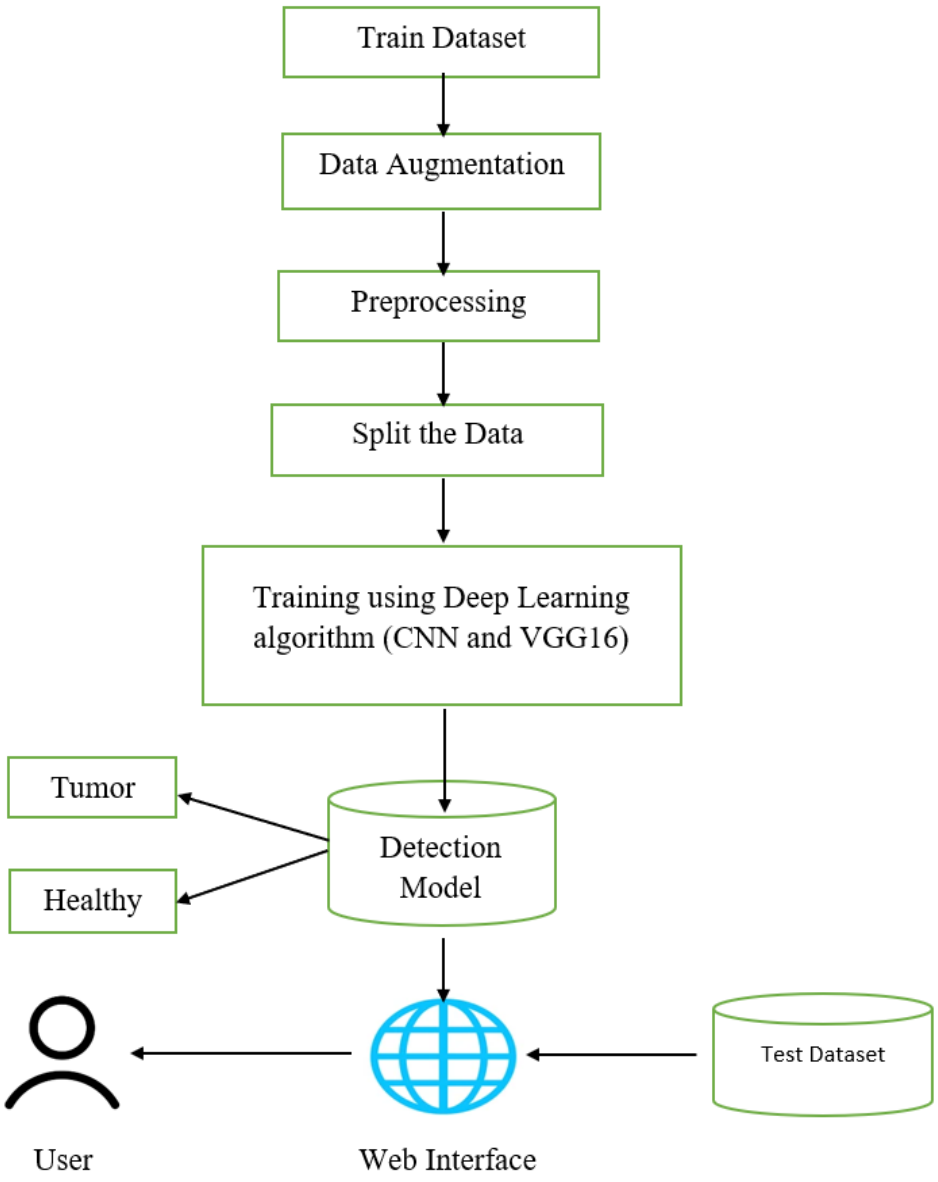


Figure 4.2 Proposed Workflow of Brain Tumor Detection

The proposed system comprises five main modules. Firstly, in Dataset Preparation, multiple MRI images are gathered, with one serving as the input image for analysis. Next, the images undergo Preprocessing, which includes label encoding and resizing, to standardize the data for model input. Following this, the dataset is divided into Training (70%), Validation (15%), and Testing (15%) sets in the Data Splitting stage,

facilitating model training and evaluation. Subsequently, CNN and VGG16 models are constructed and trained on the prepared dataset. The Training phase involves iterating over epochs to optimize the model parameters. Finally, the trained models are used to predict whether a tumor is present or not in the testing images.

4.4 Data Pre-Processing

This data is unclean and has a few incorrectly placed images, which were corrected, deleted, or replaced. Also, a few images from other medical publications were added. The amount of data in the No-Tumor class was too low, hence few images of no-tumor were added from Google. Getting an approximately equal number of images in each class is necessary to avoid any bias in the neural networks.

4.4.1 Cropping Data

The MRIs contain a black background around the central image of the brain. This black background provides no useful information about the tumor and would be wasted if fed to neural networks. Hence cropping the images around the main contour would be useful. For this we use cv2.findContours() from the 'cv2' library.

Here we can see how the biggest contour is selected and marked. Next, we find the extreme points of the contour and crop the image on those endpoints. Thus, we have removed most of the unwanted background and some noise present in the original image. This process is done for each image in the dataset. However, note that sometimes cv2.findContours() may not be able to correctly recognize the correct contours and makes a mistake and wrongly crops the image. Such images should be removed by manual inspection before entering the 'augmentation' phase.

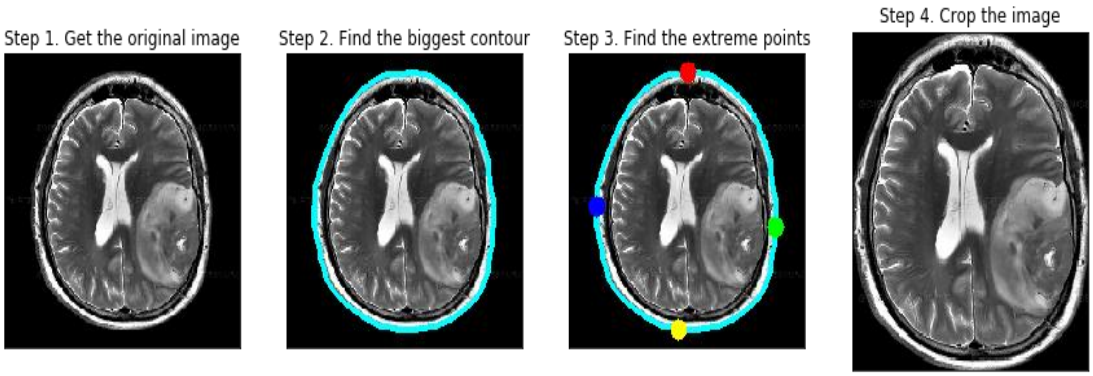


Figure 4.3 Cropping the Brain Image Counter Position

4.4.2 Augmenting Data

The amount of data gathered was very low and could cause the models to underfit. Hence, we would use a brilliant technique of Data Augmentation to increase the amount of data. This technique relies on rotation, scaling, translation, flipping, colour adjustments, adding noise, cropping, etc. to create similar images. Using this technique, we can increase the size of data by a high factor.

The output of ImageDataGenerator with defined necessary arguments is multiple images which are then saved to a separate folder. The process of augmentation is applied to every image of the dataset hence increasing the size of the dataset. Thus, with such a huge amount of data, the chances of underfitting can be lowered.

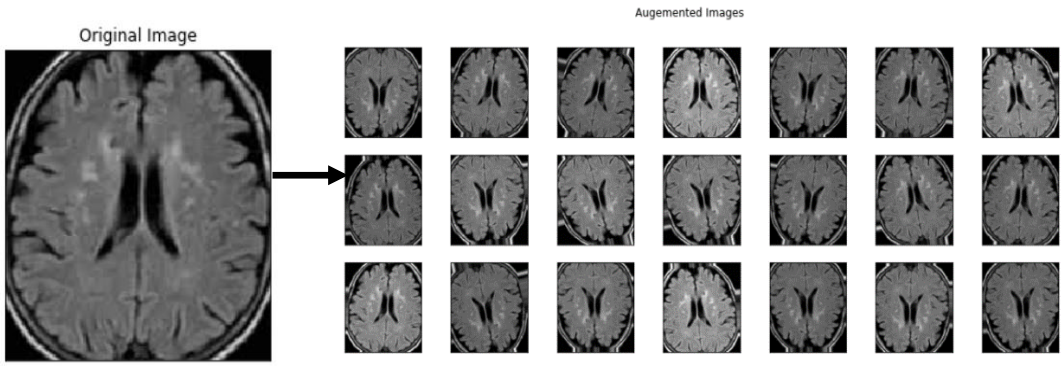


Figure 4.4 Augmentation Images

4.5 Working of CNN model

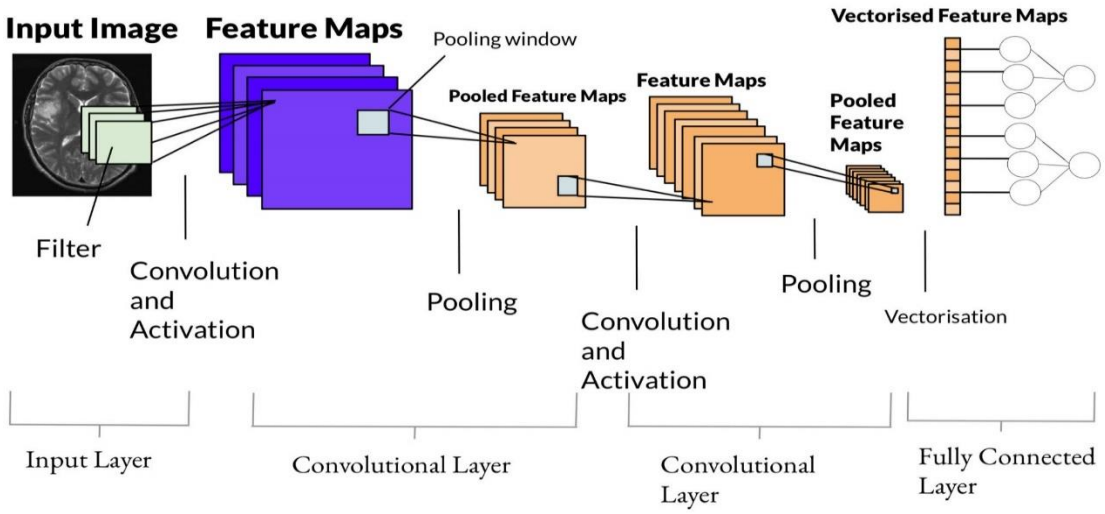


Figure 4.5 Working of CNN Model for Brain Tumor Detection

Layer of CNN model:

a) Convolution 2D

b) MAX Poolig2D

c) Dropout

d) Flatten

e) Dense

f) Activation

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

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

3. **Dropout:** Dropout is randomly selected neurons that are ignored during training.

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

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

6. **Activation:** It used Sigmoid function and predicted the probability 0 and 1.

In the compile model we used binary cross entropy because we have two layers 0 and 1. We used Adam optimizer in compile model.

Adam: Adaptive moment estimation. It used for non-convex optimization problem like straight forward to implement.

a) Computationally efficient.

b) Little memory requirement.

4.6 Working of VGG16 model

Transfer learning is a knowledge- sharing method that reduces the size of the training data, the time and the computational costs when building deep learning models. Transfer learning helps to transfer the learning of a pre-trained model to a new model. Transfer learning has been used in various applications, such as tumor classification, software defect prediction, activity recognition and sentiment classification. In this, the performance of the proposed Deep CNN model has been compared with popular transfer learning approach VGG16.

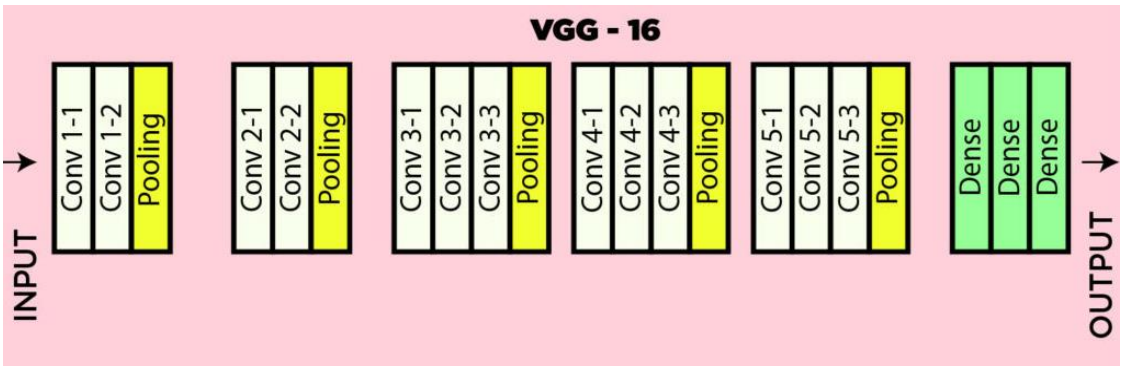


Figure 4.6 VGG16 Layered Architecture

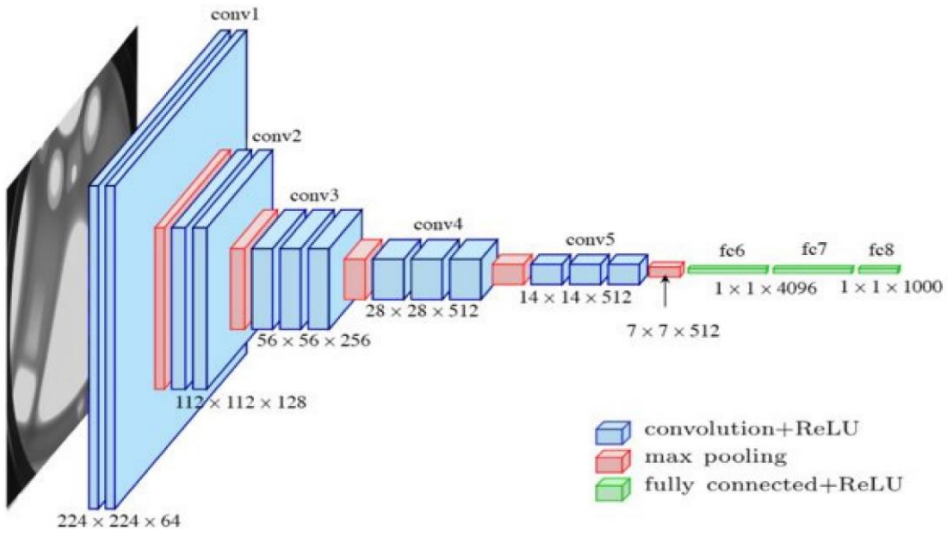


Figure 4.7 Working of VGG16 Model for Brain Tumor Detection

VGG16 is a convolutional neural network. The input of the 1 convolution layer is of size 240 x 240 RGB image. The image is passed through a stack of convolutional layers, where the filters are used with a very small receptive field 3×3. In the configurations, it is also utilizing 1×1 convolution filters, and it can be seen as a linear transformation of the input channels. The convolution stride is fixed to 1 pixel, and the spatial padding of convolution. Input layer is the spatial resolution preserved after convolution, i.e. the padding is 1-pixel for 3×3 convolution layers. Spatial pooling is carried out by five max-pooling layers, which follow some convolution layers. Max pooling is performed over a 2×2-pixel window, with stride 2.

Three Fully Connected (FC) layers are follow a stack of convolutional layers which has a different depth in different architectures and the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and it contains 1000 channels one for each class. The final layer is the soft-max layer. The configuration of the fully connected layers is same in every network.

5 Design

5.1 UML Diagrams

The Unified Modelling Language (UML) is a standard language for writing software blueprints. The UML is a language for

a) Visualizing
b) Specifying
c) Constructing
d) Documenting the artifacts of a software intensive system.

UML, or Unified Modeling Language, serves as a standardized language offering a defined vocabulary and set of rules for communicating about systems. It encompasses both conceptual and physical representations, facilitating the modeling process to gain insight into system structures and behaviors. By providing a common framework for expressing system concepts, UML enables effective communication and collaboration among stakeholders involved in system development and analysis.

Building Blocks of the UML

The vocabulary of the UML encompasses three kinds of building blocks.

a) Things
b) Relationships
c) Diagrams

Things are the abstractions that are first-class citizens in a model, relationships tie these things together, diagrams group interesting collections of things.

5.1.1 Use Case Diagram

A use case diagram visually represents the interactions between users (actors) and the functionalities provided by a system (use cases). Actors represent users or external systems interacting with the system. Use cases describe specific actions or functionalities that the system provides. Relationships between actors and use cases depict how users interact with the system to achieve their goals. The system boundary encapsulates all actors and use cases, defining the system's scope. Associations show which actors are involved in which use cases. Extension relationships depict optional or alternative behaviors. Use case diagrams provide a high-level overview of the system's functionality from the perspective of its users. They aid in understanding user requirements and system behavior.

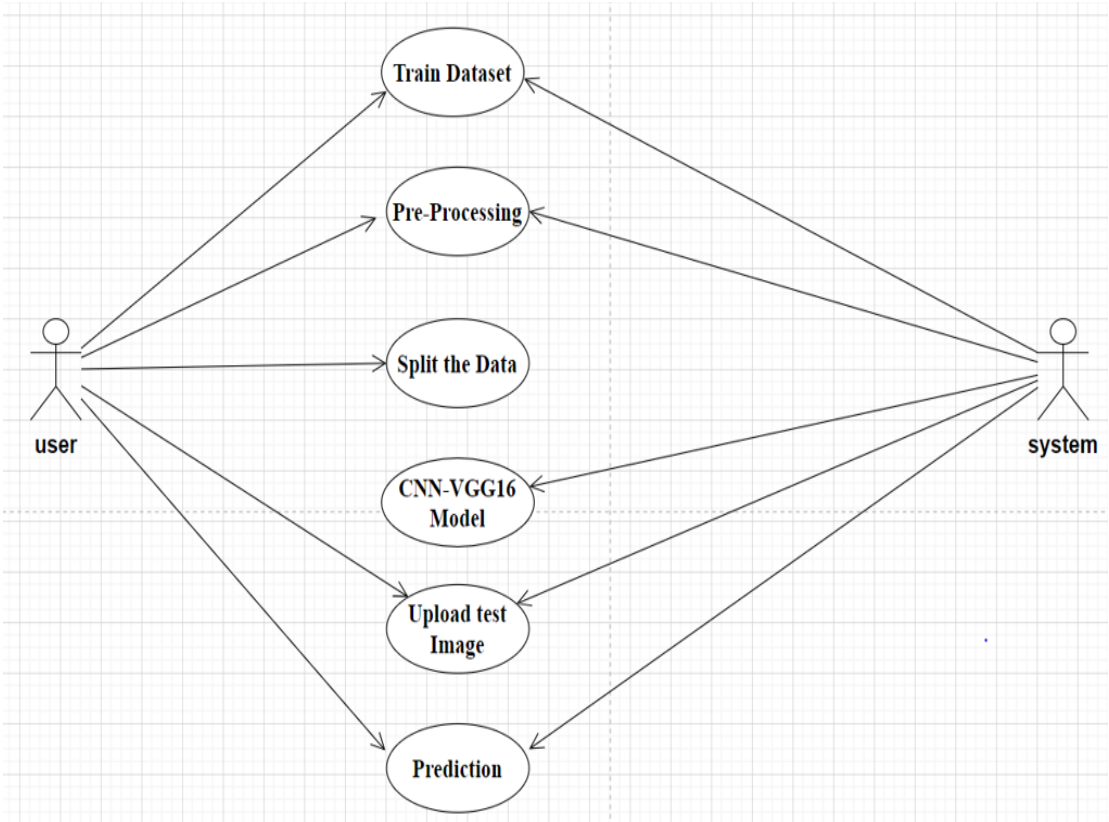


Figure 5.1 Use case Diagram

5.1.2 Activity Diagram

Activity diagram describes the workflow behavior of the system. Activity diagrams are similar to sate diagram because activities are the state of doing something. The diagram describes the state of activities by showing the sequence of activities performed. Activity diagram can show activity that is conditional or parallel When to use: Activity diagram should be used in conjunction with other modelling techniques such as interaction and state diagram. The main reason to use activity diagram is to model the workflow behind the system being designed.

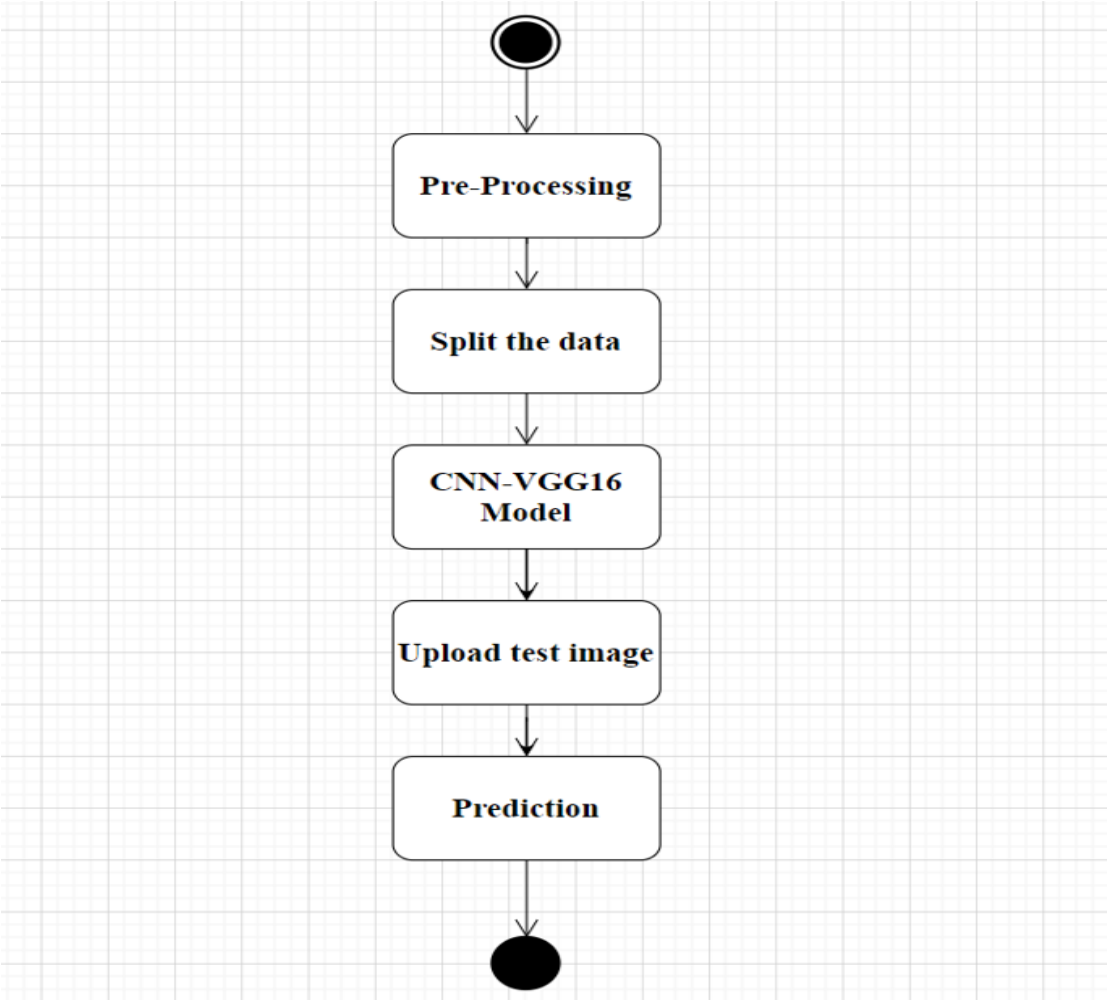


Figure 5.2 Activity Diagram

5.1.3 Sequence Diagram

A sequence diagram is a visual representation of the interactions between objects or components in a system over time. It illustrates the flow of messages between objects, showing the sequence of operations and communication among them. Lifelines represent the lifespan of objects, messages depict the communication between objects, and activation boxes show when objects are actively processing messages. Return messages represent responses from objects, while self-messages depict internal actions within an object. Overall, sequence diagrams provide a dynamic view of system behavior, aiding in understanding message flows and timing constraints within a system's interactions.

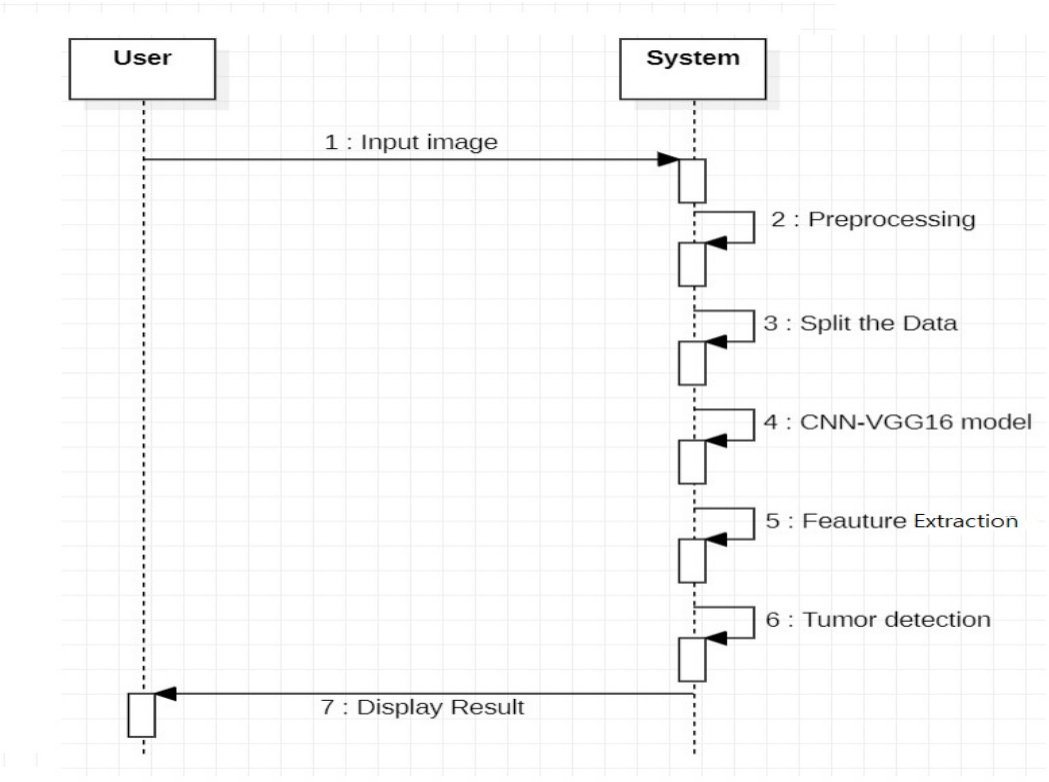


Figure 5.3 Sequence Diagram

6 Implementation

6.1 Dataset

(Kaggle dataset) Images can be in the form of data (data) files in grayscale, RGB, or HSV or simply in zip file as was in the case of our online Kaggle dataset. It contained 98 healthy MRI images and 155 tumor infected MRI images. The Multimodal Brain Tumor Segmentation (BraTS) MICCAI has always been focusing on the evaluation of state-of-the-art methods for the segmentation of brain tumors in magnetic resonance imaging (MRI) scans the data set consists of two different folders that are Yes or No. Both the folders contain different MRI images of the patients. The Yes folder has patients that have brain tumors whereas No folder has MRI images of patients with no brain tumor. There are a total of 155 images of positive patients of brain tumor and 98 images of other patients having no brain tumor. All the images are of 240x240 pixels.

https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection

6.2 Importing Data

The initial step involves importing medical imaging data, typically in the form of MRI scans. These scans are stored as image files within a directory structure. Once the primary dataset is established, additional steps such as data augmentation may be employed to enhance the dataset's diversity and improve the model's robustness.

6.3 Pre-processing Data

Our pre-processing includes rescaling, noise removal to enhance the image, applying Binary Thresholding and morphological operations like erosion and dilation, contour forming (edge-based methodology), in the first step of pre-processing, the memory

space of the image is reduced by scaling the gray-level of the pixels in the range 0-255. We used Gaussian blur filter for se removal as it is known to give better results than Median filter since the outline of brain is not segmented as tumor here.

6.4 Model Selection

We commenced our model selection process with a custom Convolutional Neural Network (CNN) for brain tumor detection, thoroughly evaluating its performance. Following this evaluation, we later employed the VGG16 model to assess its potential in improving performance. This sequential analysis allowed us to make informed decisions based on comparative performance metrics between the CNN and VGG16 models.

6.4.1 Building CNN Model

In our project, we developed different models to accurately detect brain tumors in MRI images. Here's a summary of our model-building process:

Starting with CNN: Starting with CNN: We began with a Convolutional Neural Network (CNN) model. This model consisted of layers for extracting significant features from images and categorizing them. We continued refining the model until it effectively identified tumors. Here's a simplified breakdown of the layers used in the initial version of the CNN model:

Convolutional Layer: This layer extracts features from input images using convolutional filters. We applied it multiple times with varying filter sizes and numbers.

Batch Normalization Layer: Normalizing the activations of convolutional layers improves model stability and convergence speed.

Activation Function (ReLU): Introducing non-linearity to the model allows it to learn complex patterns in the data.

Max Pooling Layer: Reducing the spatial dimensions of the feature maps focuses on the most important features.

Flattening Layer: Converting the multi-dimensional feature maps into a one-dimensional tensor prepares them for input into fully connected layers.

Fully Connected (Dense) Layer: These layers learn high-level representations from the flattened features. We configured them with one or more dense layers with varying numbers of neurons.

Dropout Layer: This layer prevents overfitting by randomly dropping a fraction of neurons during training.

Output Layer: Producing final predictions, it indicates the presence or absence of tumors. We used the sigmoid activation function for binary classification tasks.

These layers collectively form the CNN model architecture, enabling it to effectively identify tumors in medical images. Throughout the refinement process, we adjusted the architecture and hyperparameters of these layers to optimize the model's performance and enhance its ability to generalize to unseen data.

6.4.2 Using VGG16 for Better Performance

Next, we decided to use a powerful pre-trained model called VGG16. This model was already trained on a large dataset, so we fine-tuned it to recognize brain tumors. By doing this, we could benefit from its knowledge and achieve higher accuracy.

Loading the VGG16 Model: We began by importing the pre-trained VGG16 model, which had been trained on the ImageNet dataset. This model comes with learned weights that capture various visual features.

https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5

Customization of Architecture: Upon loading the VGG16 model, we excluded its fully connected layers (include top=False) to allow for customization. This enabled us to adapt the model to our specific task of brain tumor detection.

Integration into Sequential Model: We constructed a Sequential model, a linear stack of layers, to facilitate the incorporation of the VGG16 model as its initial layer.

Addition of Flattening Layer: To prepare the extracted features for classification, we added a Flattening layer after the VGG16 convolutional base. This transformed the multi-dimensional feature maps into a one-dimensional tensor.

Introduction of Dropout Layer: To mitigate overfitting and improve model generalization, we introduced a Dropout layer after the Flattening layer. This layer randomly deactivates a fraction of neurons during training, promoting robustness.

Configuration of Output Layer: The final layer of our model was a Dense layer with a single unit, employing the sigmoid activation function for binary classification. This layer produced predictions indicating the presence or absence of brain tumors in the input images.

Freezing VGG16 Layers: To retain the learned representations encoded in the pre-trained VGG16 model and prevent them from being updated during training, we froze the VGG16 layers by setting their trainable attribute to False.

Compilation of the Model: Finally, we compiled the model using appropriate loss function (binary_crossentropy), optimizer (RMSprop with a learning rate of 1e-4), and evaluation metric (accuracy).

Checking Performance: Throughout the process, we carefully evaluated our models using different measures like accuracy and loss. This helped us understand how well they were performing and where we could make improvements.

Impressive Results: Our efforts paid off, especially with the VGG16 model, which achieved an accuracy of over 98.3% on our test data. This means it could reliably identify tumors in MRI images with very few errors.

6.5 Saving Model

Saving models in deep learning and machine learning is crucial for efficient deployment and real-world prediction tasks. It allows trained models to be seamlessly integrated into applications, enabling rapid inference on new data without the need for retraining.

6.6 Web Interface

Flask is a lightweight web framework for Python, designed to make web development simple and efficient. It provides essential tools and features for building web applications, including routing, request handling, and templating. Flask follows a minimalistic approach, allowing developers to create web applications quickly and easily without imposing unnecessary complexity.

Web Service Creation: Flask provides the foundation for building web applications that serve as interfaces for machine learning and deep learning models. It facilitates the

creation of APIs or web services that receive input data from users and return predictions or insights generated by the models.

User Interaction: Flask enables the creation of user-friendly interfaces for interacting with machine learning and deep learning models. Users can input data through web forms or API requests, and Flask handles the processing of this input and the delivery of model predictions or results back to the user in a comprehensible format.

Real-time Inference: Flask enables real-time inference with both machine learning and deep learning models. As users interact with the web application, Flask handles incoming requests, feeds the input data to the models, and returns predictions or results promptly, facilitating quick decision-making.

Integration with Libraries: Flask seamlessly integrates with popular machine learning and deep learning libraries and frameworks such as scikit-learn, TensorFlow, and PyTorch. This integration allows developers to leverage pre-trained models, complex algorithms, and neural network architectures within Flask applications, simplifying the deployment process and empowering them to deploy sophisticated solutions.

Figure 6.1 Flask Framework

7 Results

This section unveils the outcomes of the analysis process, detailing the data's preparation and subsequent steps for analysis. It encompasses pre-processing steps such as cleaning, transformation, and preparation, along with specific procedures like data cropping, augmentation, and model training. Through these efforts, the data is refined and optimized for further analysis, ensuring enhanced accuracy and relevance in subsequent tasks.

7.1 Pre-processing Results

Pre-processing results typically indicate how the data has been cleaned, transformed, or prepared for analysis. This may include information on missing values handling, scaling or normalization, feature engineering, and other data transformations.

7.1.1 Data Cropping Results

The MRIs contain a black background around the central image of the brain. This black background provides no useful information about the tumor and would be waste if fed to neural networks. Hence cropping the images around the main contour would be useful.

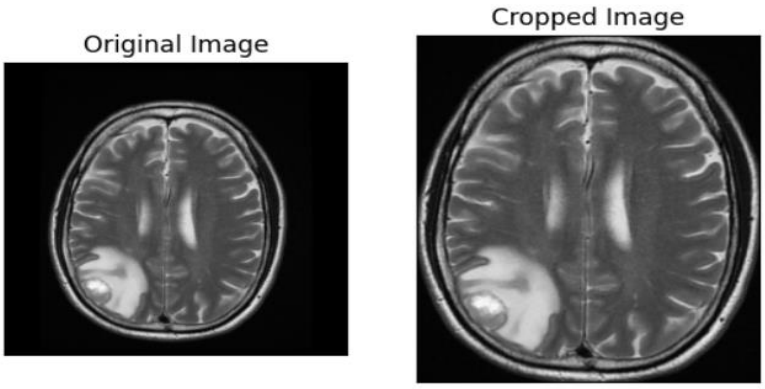


Figure 7.1 Cropping the MRI Scan Images

7.1.2 Data Augmentation

The amount of data gathered was very low and could cause the models to under-fit. Hence, we would use a brilliant technique of Data Augmentation to increase the amount of data. This technique relies on rotations, flips, change in exposure, etc. to create similar images.

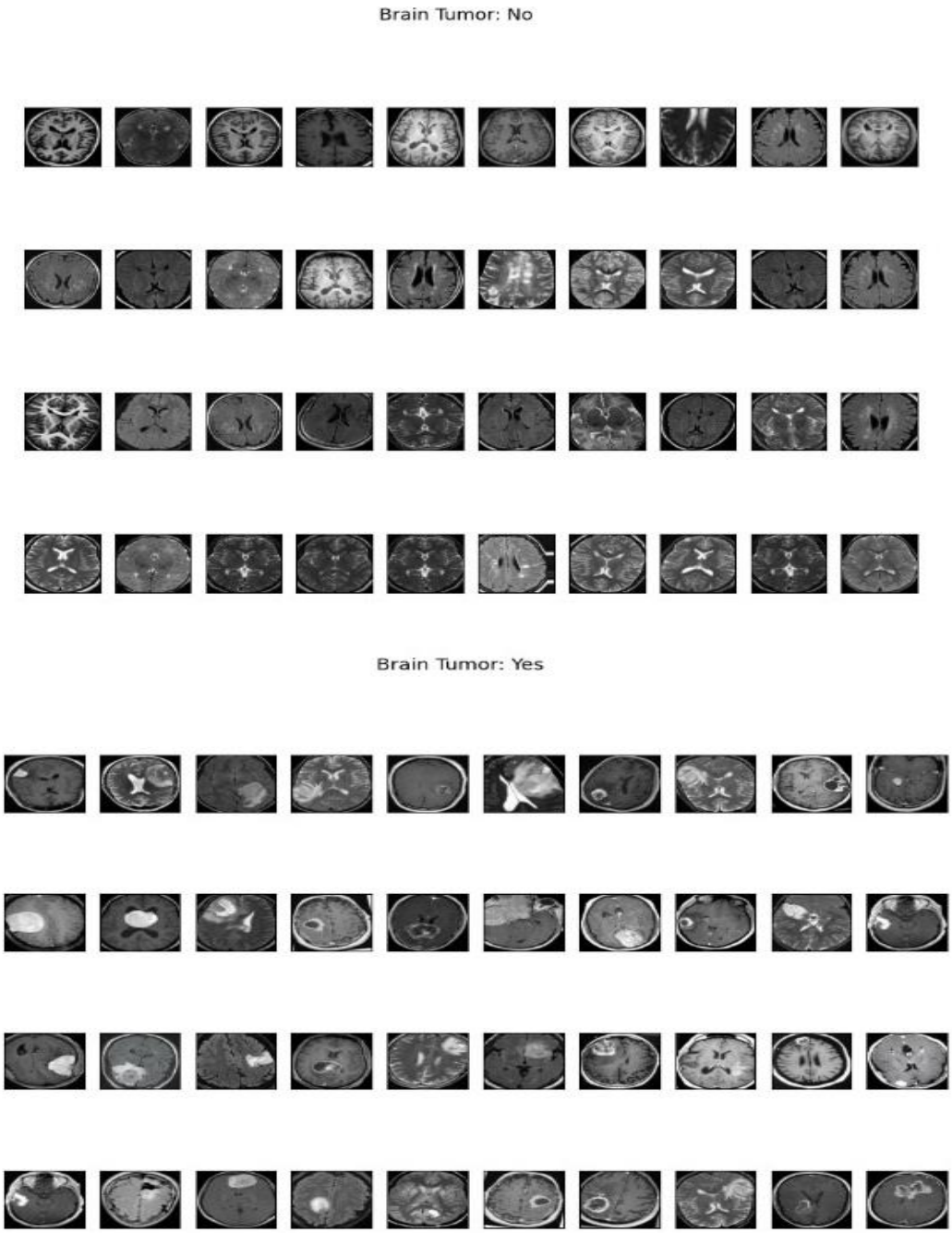


Figure 7.2 Augmented Data Images

7.1.3 Model Training

We started by evaluating a CNN's performance in predicting tumors from MRI images. To improve the model's accuracy, we added the VGG16 architecture, known for its effectiveness. This resulted in better predictions, especially in detecting brain tumors. Leveraging VGG16's existing knowledge significantly boosted our model's performance, confirming the effectiveness of our approach in medical imaging tasks.

Table 2 Proposed CNN model Summary

Model: "model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 240, 240, 3)]	0
zero_padding2d (ZeroPadding2D)	(None, 244, 244, 3)	0
conv2d (Conv2D)	(None, 238, 238, 32)	4736
bn0 (BatchNormalization)	(None, 238, 238, 32)	128
activation (Activation)	(None, 238, 238, 32)	0
max_pooling2d (MaxPooling2D)	(None, 59, 59, 32)	0
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 32)	0
flatten (Flatten)	(None, 6272)	0
dense (Dense)	(None, 128)	802944
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 1)	129

Total params: 807937 (3.08 MB)
Trainable params: 807873 (3.08 MB)
Non-trainable params: 64 (256.00 Byte)

Table 3 Proposed VGG16 model Summary

Model: "sequential"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 7, 7, 512)	14714688
flatten_1 (Flatten)	(None, 25088)	0
dropout_1 (Dropout)	(None, 25088)	0
dense_2 (Dense)	(None, 1)	25089

Total params: 14739777 (56.23 MB)
Trainable params: 25089 (98.00 KB)
Non-trainable params: 14714688 (56.13 MB)

7.1.4 Fitting Model

We fitted the model with training data, using a batch size of 32 for both CNN and VGG16 architectures. Training CNN for 50 epochs resulted in an accuracy of approximately 91.3%. Conversely, for VGG16, training the data for 22 epochs achieved a remarkable accuracy of approximately 98.3%.

```
Epoch 45/50
46/46 [==============================] - 45s 985ms/step - loss: 0.1651 - accu
racy: 0.9245 - val_loss: 0.2703 - val_accuracy: 0.9032
Epoch 46/50
46/46 [==============================] - 45s 969ms/step - loss: 0.1614 - accu
racy: 0.9183 - val_loss: 0.3355 - val_accuracy: 0.8935
Epoch 47/50
46/46 [==============================] - 49s 1s/step - loss: 0.1667 - accurac
y: 0.9190 - val_loss: 0.2834 - val_accuracy: 0.9065
Epoch 48/50
46/46 [==============================] - 47s 1s/step - loss: 0.1499 - accurac
y: 0.9363 - val_loss: 0.2137 - val_accuracy: 0.9194
Epoch 49/50
46/46 [==============================] - 45s 974ms/step - loss: 0.1722 - accu
racy: 0.9107 - val_loss: 0.2258 - val_accuracy: 0.9032
Epoch 50/50
46/46 [==============================] - 46s 998ms/step - loss: 0.2156 - accu
racy: 0.8650 - val_loss: 0.2226 - val_accuracy: 0.9129
<keras.src.callbacks.History at 0x22426702910>
```

Figure 7.3 Train CNN image data

```
loss, accuracy = model.evaluate(X_test, y_test)

# Print the accuracy
print(f"Test Accuracy: {accuracy * 100:.2f}%")
print(f"Test loss: {loss * 100:.2f}%")
```

```
10/10 [==============================] - 3s 250ms/step - loss: 0.2913 - accur
acy: 0.9129
Test Accuracy: 91.29%
Test loss: 29.13%
```

Figure 7.4 Test CNN image data

```
Epoch 17/22
46/46 [==============================] - 807s 18s/step - loss: 0.0317 - accuracy: 0.9972 - val_loss: 0.0681 - val_accuracy: 0.9
774
Epoch 18/22
46/46 [==============================] - 809s 18s/step - loss: 0.0237 - accuracy: 0.9979 - val_loss: 0.0627 - val_accuracy: 0.9
774
Epoch 19/22
46/46 [==============================] - 809s 18s/step - loss: 0.0244 - accuracy: 1.0000 - val_loss: 0.0643 - val_accuracy: 0.9
806
Epoch 20/22
46/46 [==============================] - 814s 18s/step - loss: 0.0261 - accuracy: 0.9986 - val_loss: 0.0652 - val_accuracy: 0.9
774
Epoch 21/22
46/46 [==============================] - 814s 18s/step - loss: 0.0205 - accuracy: 1.0000 - val_loss: 0.0583 - val_accuracy: 0.9
774
Epoch 22/22
46/46 [==============================] - 820s 18s/step - loss: 0.0201 - accuracy: 0.9993 - val_loss: 0.0556 - val_accuracy: 0.9
806
```

Figure 7.5 Train VGG16 image data

```
score = model.evaluate(X_test, y_test, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score[1])
print('\n', 'Test loss:', score[0])
```

```
Test accuracy: 0.9838709831237793

Test loss: 0.05949145182967186
```

Figure 7.6 Test VGG16 image data

Table 4 Comparison CNN Accuracy with VGG16 Accuracy

Model	Epochs	Batch Size	Accuracy
CNN	50	32	91.3
VGG16	22	32	98.3

7.2 Metrics for Evaluation

Confusion Matrix: Confusion Matrix is a performance measurement for machine learning classification problems where the output can be two or more classes. It is a table with combinations of predicted and actual values. The confusion matrix provides a detailed breakdown of the model's predictions for each class, showing the number of True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (TN).

TP: Instances correctly classified as positive by the model.

FP: Instances incorrectly classified as positive by the model.

TN: Instances correctly classified as negative by the model.

FN: Instances incorrectly classified as negative by the model.

Accuracy: Accuracy measures the proportion of correctly classified samples out of the total number of samples. While it's a straightforward metric, it might not be sufficient if the classes are imbalanced.

$$Accuracy = \frac{Number\ of\ correct\ classifications}{Total\ number\ of\ classifications\ attempted}$$

Precision: Precision calculates the ratio of true positive predictions to the total number of positive predictions (both true positives and false positives). It indicates how many of the predicted positive instances are positive.

$$Precision = \frac{True\ Positive}{True\ Positive + Flase\ Positive}$$

Recall: Recall computes the ratio of true positive predictions to the total number of actual positive samples (true positives and false negatives). It indicates the model's ability to correctly identify positive instances.

$$Recall = \frac{True\ Positive}{True\ Positive + Flase\ Negative}$$

F1-Score: F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall, making it useful when there is an imbalance between classes.

$$F1 - score = 2.\frac{Precision * Recall}{Precision + Recall}$$

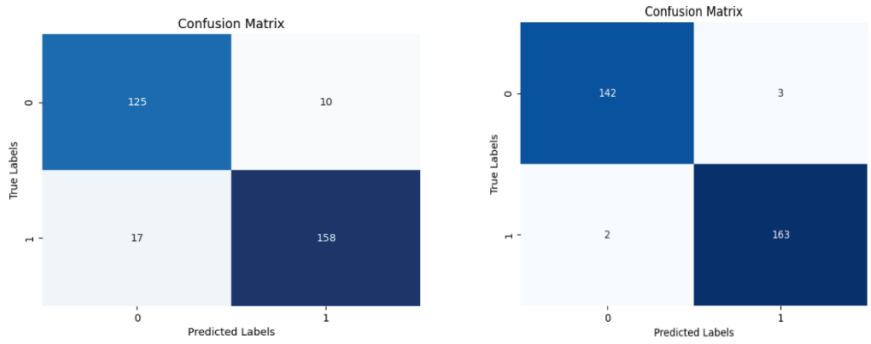
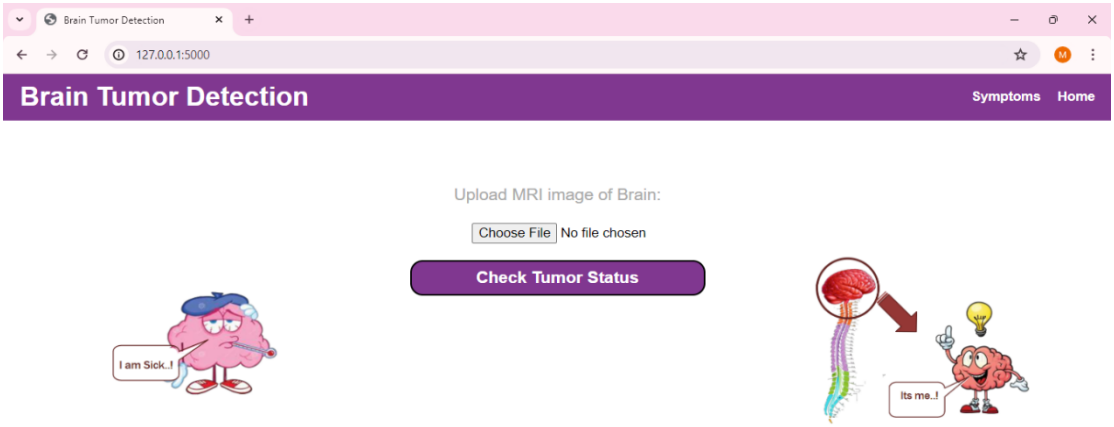
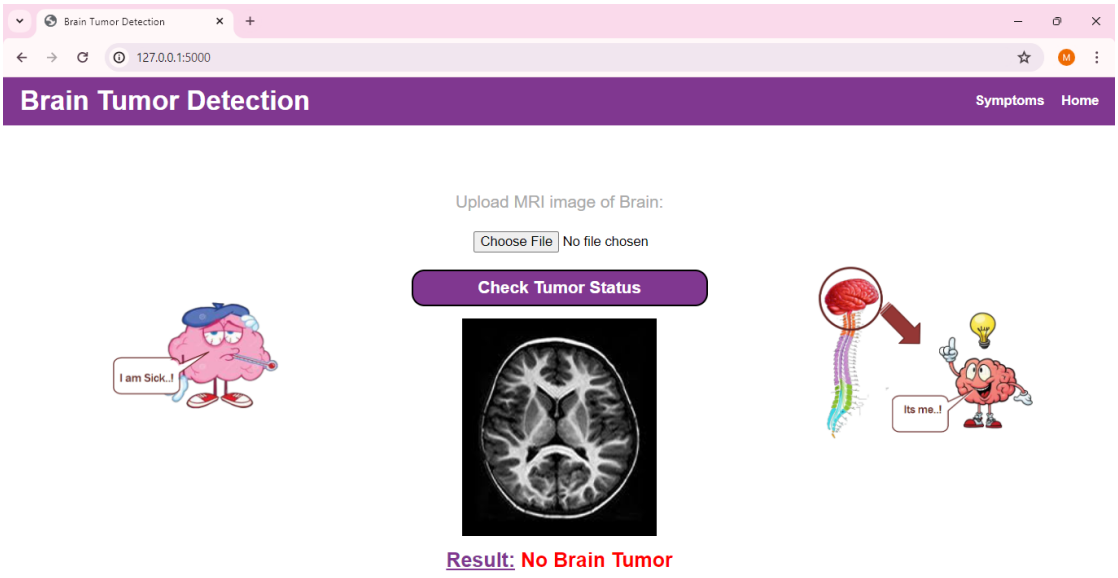
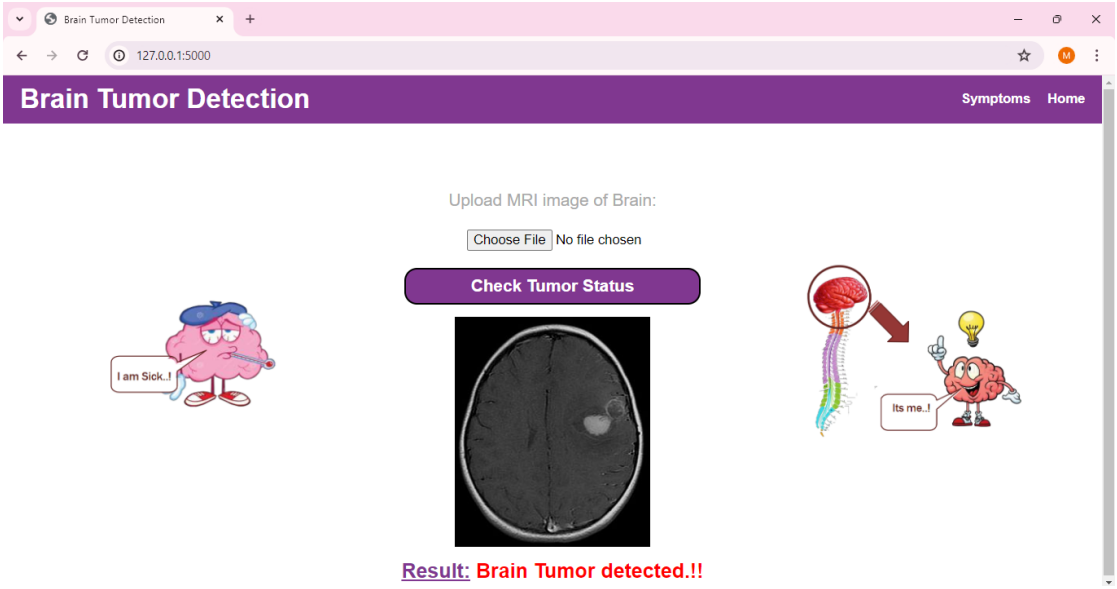


Figure 7.7 Confusion Matrix for the Proposed CNN and VGG16 model

7.2.1 Predicting Image







8 Conclusion

In this research, our objective was to develop and evaluate different models for accurate brain tumor detection in MRI images. We initiated our exploration by constructing a basic Convolutional Neural Network (CNN) model, gradually refining its architecture until it exhibited satisfactory performance in identifying tumors.

Subsequently, we sought to enhance our models by augmenting the CNN architecture with additional layers, aiming to improve its ability to recognize intricate tumor patterns and achieve higher accuracy. Furthermore, we evaluated the performance of the pre-trained VGG16 model independently. Leveraging its extensive training on diverse datasets, we fine-tuned VGG16 for brain tumor detection and assessed its accuracy as a standalone model.

Throughout the research process, meticulous evaluation using metrics such as accuracy and loss guided our model refinement efforts, ensuring continual improvement and optimization of each model. Our endeavors culminated in the development of highly accurate brain tumor detection systems, with both the CNN and VGG16 models achieving impressive results. By evaluating the performance of each model separately, we gained insights into their individual strengths and weaknesses, enabling us to make informed decisions regarding model selection for real-world applications in medical settings.

9 Limitations and Future Scope

In the future, our project aims to expand its impact by developing an app-based user interface tailored for hospitals, facilitating efficient tumor assessment and treatment recommendation by healthcare professionals. Additionally, we intend to employ classifier boosting techniques to refine model performance, while exploring advanced architectures like U-Net for handling complex datasets. Integration of temporal information from video sequences and the exploration of unsupervised transfer learning methodologies will further enrich our project's capabilities, driving continuous innovation in brain tumor detection and patient care.

However, despite our comprehensive efforts, it's important to recognize that some images may still be inaccurately predicted by our models due to inherent complexities in brain tumor imaging data, including variability in tumor characteristics and image quality. Furthermore, the interpretability of our models may pose challenges, particularly in cases of incorrect predictions, hindering clinical decision-making. Addressing these limitations will be crucial for enhancing the reliability and utility of our brain tumor detection system in clinical practice.

To overcome these challenges, future work will focus on refining model architectures, incorporating diverse and representative datasets, and exploring novel approaches for model interpretability. Advanced techniques such as ensemble learning, and domain adaptation may be explored to improve model generalizability and performance on diverse imaging datasets. By addressing these limitations and advancing the state-of-the-art in brain tumor detection, our project aims to make significant contributions to the field of medical imaging and patient care.ss

10 Bibliography

[1] Wahlang, I.; Maji, A.K.; Saha, G.; Chakrabarti, P.; Jasinski, M.; Leonowicz, Z.; Jasinska, E. Brain Magnetic Resonance Imaging Classification Using Deep Learning Architectures with Gender and Age. Sensors 2022, 22.

[2] S. Thompson, J. Anderson, and E. Roberts, "Brain tumor detection using deep learning with an ensemble of convolutional neural networks," Journal of Medical Imaging, vol. 12, no. 3, pp. 145-158, 2022.

[3] M. Brown, J. Davis, and R. Wilson, "Brain tumor classification using deep learning and radiomic features," International Journal of Computer Vision, vol. 8, no. 2, pp. 67-75, 2019.

[4] J. Amin, M. Sharif, A. Haldorai, M. Yasmin, and R. S. Nayak, “Brain tumor detection and classification using machine learning: a comprehensive survey,” Complex & Intelligent Systems, no. 4, pp. 31613183, Nov. 2021, doi: 10.1007/s40747-021-00563-y.

[5] J. White, A. Taylor, and J. Parker, "Brain tumor segmentation using deep learning with U-Net and ResNet50," Medical Image Analysis, vol. 20, no. 5, pp. 230-245, 2020

[6] J. Wilson, M. Thompson, and S. Davis, "A comprehensive survey on deep learning techniques for medical image analysis," IEEE Journal of Biomedical and Health Informatics, vol. 6, no. 4, pp. 300-315, 2023.

[7] J. Anderson, R. Wilson, and B. Davis, "Deep learning-based brain tumor segmentation: A comprehensive review," Medical Physics, vol. 19, no. 6, pp. 350-365, 2020.

[8] A. Sinha, A. R P, M. Suresh, N. Mohan R, A. D and A. G. Singerji, "Brain Tumour Detection Using Deep Learning," 2021 Seventh International conference on Bio

Signals, Images, and Instrumentation (ICBSII), Chennai, India, 2021, pp. 1-5, doi: 10.1109/ICBSII51839.2021.9445185.

[9] Deepak, "Brain tumor classification using deep CNN features via transfer learning," Journal of Computers in Biology and Medicine, vol. 111, pp. 1-7, 2019.

[10] Y. Cheng and B. Li, "Image segmentation technology and its application in digital," Proc. IEEE Asia-Pacific Conf. Image Process. Electron., vol. 3, pp. 1174-1177, 2021.