BRAIN TUMOUR DETECTION USING MRI IMAGES WITH CNN

Mini Project submitted in partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING - IOT

Under the esteemed guidance of

Mr. P. Manohar

Assistant Professor

By

JALAGAM NIVEDITHA **21R11A6928**

Department of Computer Science and Engineering

Accredited by NBA

Geethanjali College of Engineering and Technology

(UGC Autonomous)

(Affiliated to J.N.T.U.H, Approved by AICTE, New Delhi)

Cheeryal (V), Keesara (M), Medchal.Dist.-501 301.

NOVEMBER-2024

Geethanjali College of Engineering & Technology

(UGC Autonomous)

(Affiliated to JNTUH, Approved by AICTE, New Delhi)

Cheeryal (V), Keesara(M), Medchal Dist.-501 301.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Accredited by NBA

CERTIFICATE

This is to certify that the B.Tech Mini Project report entitled **"BRAIN TUMOUR DETECTION USING MRI IMAGES WITH CNN"** is a bonafide work done by **AASHADAPU PALLAVI (21R11A6901), JALAGAM NIVEDITHA(21R11A6928), KATTA SAI KARTHIK (20R11A6949)**, in partial fulfillment of the requirement of the award for the degree of Bachelor of Technology in "**COMPUTER SCIENCE AND ENGINEERING (INTERNET OF THINGS)**" from Jawaharlal Nehru Technological University, Hyderabad during the year 2024-2025.

Internal Guide	**Project Coordinator**	**HOD – IOT**
Mr.P. Manohar	**Mr. P. Manohar**	**Dr.K.Srinivas**
Assistant Professor	**Associate Professor**	**Professor**

External Examiner

Geethanjali College of Engineering & Technology

(UGC Autonomous)

(Affiliated to JNTUH Approved by AICTE, New Delhi)

Cheeryal (V), Keesara(M), Medchal Dist.-501 301.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Accredited by NBA

DECLARATION BY THE CANDIDATE

We, **Aashadapu Pallavi, Jalagam Niveditha , Katta Sai Karthik bearing Roll Nos. 21R11A6901, 21R11A6928, 20R11A6949,** hereby declare that the project report entitled **"BRAIN TUMOUR DETECTION USING MRI IMAGES WITH CNN"** is done under the guidance of **Mr.P.Manohar, Assistant Professor,** internal guide, Department of Computer Science and Engineering- Internet Of Things, Geethanjali College of Engineering and Technology, is submitted in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in ComputerScience and Engineering**.

This is a record of bonafide work carried out by us in **Geethanjali College of Engineering and Technology** and the results embodied in this project have not been reproduced or copied fromany source. The results embodied in this project report have not been submitted to any other University or Institute for the award of any other degree or diploma.

A.Pallavi(21R11A6901)

J.Niveditha(21R11A6928)

K. Karthik(20R11A6949)

Department of Cse-IoT,

Geethanjali College of Engineering and Technology, Cheeriyal

batch-06 project report.docx

by Turnitin LLC

Submission date: 06-Jan-2025 08:47PM (UTC+0700)
Submission ID: 2560231106
File name: batch-06_project_report.docx (71.81K)
Word count: 8176
Character count: 49264

batch-06 project report.docx

ORIGINALITY REPORT

3%	1%	1%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Dinesh Goyal, Bhanu Pratap, Sandeep Gupta, Saurabh Raj, Rekha Rani Agrawal, Indra Kishor. "Recent Advances in Sciences, Engineering, Information Technology & Management - Proceedings of the 6th International Conference “Convergence2024” Recent Advances in Sciences, Engineering, Information Technology & Management, April 24–25, 2024, Jaipur, India", CRC Press, 2025 Publication	1%
2	Suman Kumar Swarnkar, Abhishek Guru, Gurpreet Singh Chhabra, Harshitha Raghavan Devarajan. "Artificial Intelligence Revolutionizing Cancer Care - Precision Diagnosis and Patient-Centric Healthcare", CRC Press, 2025 Publication	<1%
3	Vivek S. Sharma, Shubham Mahajan, Anand Nayyar, Amit Kant Pandit. "Deep Learning in Engineering, Energy and Finance - Principles and Applications", CRC Press, 2024 Publication	<1%

4	Amir Shachar. "Introduction to Algogens", Open Science Framework, 2024 Publication	<1%
5	www.coursehero.com Internet Source	<1%
6	www.slideshare.net Internet Source	<1%
7	www.ijeast.com Internet Source	<1%
8	Submitted to JK Lakshmipat University Student Paper	<1%
9	eir.zp.edu.ua Internet Source	<1%
10	Submitted to Adventist University of Central Africa Student Paper	<1%
11	Chandrakant M. Umarani, S.G. Gollagi, Shridhar Allagi, Kuldeep Sambrekar, Sanjay B. Ankali. "Advancements in deep learning techniques for brain tumor segmentation: A survey", Informatics in Medicine Unlocked, 2024 Publication	<1%
12	www.kosmix.com Internet Source	<1%
	www.frontiersin.org	

Acknowledgement

We are greatly indebted to the authorities of Geethanjali College of Engineering and Technology, Cheeryal, Medchal District, for providing us the necessary facilities to successfully carry out this mini project work titled **"BRAIN TUMOUR DETECTION USING MRI IMAGES WITH CNN".**

We would like to express our sincere gratitude to our Principal **Prof. Dr. S. Udaya Kumar** for providing the necessary infrastructure to complete our project. We are also thankful to our Secretary **Mr. G. R. Ravinder Reddy** for providing an interdisciplinary& progressive environment.

We would like to express our sincere thanks to **Dr. K. Srinivas**, **Professor**, Head of Department of Internet of Things, Geethanjali College of Engineering and Technology, Cheeryal, whose motivation in the field of software development has made us to overcome all hardships during the course of study and successful completion of project.

We would like to express our deepfelt gratitude and sincere thanks to our project coordinator **Mr. P. Manohar, Associate Professor,** Department of Internet of Things, Geethanjali College of Engineering and Technology, Cheeryal, for his skillful guidance, timely suggestions and encouragement in completing this project.

We would like to express our profound sense of gratitude to all for having helped us in completing this dissertation. We would like to express our deepfelt gratitude and sincere thanks to our guide **Mr. P.Manohar**, **Assistant Professor**, Department of Internet of Things, Geethanjali College of Engineering and Technology, Cheeryal, for his skillful guidance, timely suggestions and encouragement in completing this project.

Finally, we would like to express our heartfelt thanks to our parents who were very supportive both financially and mentally and for their encouragement to achieve our set goals.

Abstract

A brain tumor originates from the rapid and uncontrolled growth of cells in the brain, and without timely intervention, it can become life-threatening. Despite significant progress in medical imaging and computational methods, precise segmentation and classification of brain tumors remain a persistent challenge due to their varied location, structure, and size. This study provides an extensive review of the potential of Magnetic Resonance (MR) imaging in identifying brain tumors, emphasizing the integration of computational intelligence and statistical image processing techniques. It introduces multiple methodologies for brain tumor detection, focusing on Deep Learning (DL), Transfer Learning (TL), and Machine Learning (ML) models. Additionally, this research discusses critical aspects such as tumor morphology, data augmentation methods, feature extraction techniques, and the use of diverse datasets, while presenting an evaluation matrix for assessing system performance. By consolidating insights into the strengths, limitations, advancements, and future trends, this study aims to serve as a valuable resource for researchers in the ongoing effort to enhance the accuracy and efficiency of brain tumor detection.

Keywords: Deep Learning, Magnetic Resonance Imaging (MRI), Machine Learning, Transfer Learning, Tumor Detection.

List of Figures

List of Tables

Table of Contents

Contents **Page no.**

1.Introduction

1.1 Introduction to the Project

Brain tumors are abnormal growths of cells in the brain, which can be life-threatening depending on their type and stage. Detecting brain tumors early is critical, as it increases the chances of effective treatment and improves the patient's quality of life. MRI scans are one of the most effective tools for diagnosing brain tumors. However, interpreting these scans manually can be time-consuming and prone to human error. Radiologists may focus on specific areas of concern and miss subtle signs of a tumor, leading to delayed or incorrect diagnoses. This highlights the need for automated systems to aid in detecting tumors with greater accuracy and speed.Advancements in artificial intelligence, particularly deep learning, have opened up new possibilities for analyzing medical imaging data. The aim of this project is to develop a model capable of detecting brain tumors from MRI scans with high precision. This not only helps doctors confirm diagnoses but also ensures consistency in interpreting medical data.Early detection is vital because it enables timely interventions, which can significantly improve survival rates and patient outcomes. With cases of brain tumors rising globally, having an automated diagnostic tool can reduce the burden on healthcare professionals and enhance the efficiency of medical evaluations. To start, an MRI scan of a patient's brain will be uploaded into the system. The image will be preprocessed to enhance its features, and a trained model will predict whether a tumor is present or not.

Non-tumorous: At this stage, the brain shows no signs of abnormal cell growth. The patient exhibits normal brain function, and MRI scans appear healthy with no structural disruptions.

Benign tumor: A benign tumor grows slowly and is usually non-cancerous. However, it can still exert pressure on surrounding areas of the brain, causing symptoms like headaches or vision problems. While not as dangerous as malignant tumors, timely detection is crucial to prevent complications.

Malignant tumor: Malignant tumors are aggressive and cancerous, often spreading to other parts of the brain or body. They cause significant disruption in brain function and require urgent medical attention.

Precancerous lesions: These are abnormal growths that could potentially become cancerous over time. While they aren't yet classified as tumors, their presence is a

warning sign, making monitoring and early intervention necessary.

1.2 Project Category

Our project is research-driven, where we have analyzed extensive literature and identified gaps in previous studies on brain tumor detection. Current diagnostic practices often rely heavily on manual interpretation, which may lead to inconsistencies or delayed diagnoses. To address this, we turn to advanced computing techniques like machine learning. Utilizing machine learning for detecting and classifying brain tumors represents a shift towards smarter, more accurate, and efficient diagnostic tools, aligning with the broader movement toward personalized medicine.

1.3 Objectives

The primary objective of this project is to facilitate the early detection and classification of brain tumors. We aim to develop an efficient machine learning model that can analyze MRI scans to distinguish between tumorous and non-tumorous conditions, improving diagnostic accuracy and assisting healthcare professionals.

1.4 Problem Formulation

Diagnosing brain tumors accurately remains a significant challenge due to the complexity of brain anatomy and the subtlety of early-stage tumor manifestations. Current methods rely on clinical history and expert evaluation of MRI scans, which can be prone to human error. Brain tumors disrupt normal brain function, and symptoms vary depending on the tumor's type, size, and location.

Malignant tumors, in particular, are aggressive and require immediate attention, but early symptoms can often go unnoticed. Delayed diagnosis can result in a poor prognosis, significantly reducing survival rates. While benign tumors grow slowly, their impact on surrounding brain tissue can still be detrimental, necessitating timely intervention.

Reports predict that the prevalence of brain tumors will increase with the aging population and advancements in imaging technology, making early detection and effective diagnosis more critical than ever. By developing an automated system, we aim to address these challenges and support healthcare professionals in providing timely, accurate care.

1.5 Identification/Reorganization of Need

When radiologists examine MRI scans for brain tumor detection, inherent biases or preconceived notions about specific conditions can result in overlooking other potential diagnoses. This limitation highlights the necessity for advanced, unbiased diagnostic systems. Smart systems equipped with machine learning algorithms can assist medical professionals by offering data-driven insights, minimizing errors, and improving the accuracy of diagnosis.

1.6 Existing System

One existing method includes manual examination of MRI scans by radiologists. While effective to an extent, this approach depends heavily on individual expertise and is prone to subjectivity.

Another method is **Radiomics-Based Models**, which extract quantitative features from MRI images. These models analyze the texture, shape, and intensity of tumor regions to aid in classification. However, they require substantial feature engineering.

Lastly, **Machine Learning Classifiers**, such as Support Vector Machines (SVM) and Random Forests, use handcrafted features from MRI scans for classification. While these models provide improved results compared to manual diagnosis, they lack the capability to automatically extract complex patterns from data.

1.7 Limitations

- **Manual Dependence**: Radiologists' diagnoses rely heavily on individual expertise, making the process prone to variability and errors.
- **Feature Engineering**: Traditional machine learning models depend on manually curated features, which may fail to capture intricate patterns present in the data.
- **Limited Generalization**: Existing models may not generalize well across diverse datasets due to variations in imaging protocols, equipment, and patient demographics.

1.8 Proposed System

The proposed system employs **Deep Neural Networks (DNNs)** for brain tumor detection using MRI images. These models excel at learning complex, high-dimensional features directly from raw data, eliminating the need for manual feature extraction. By leveraging Convolutional Neural Networks (CNNs), the system can analyze spatial patterns in MRI

scans, enabling accurate classification of tumors into categories such as benign or malignant. This approach provides a scalable and efficient solution to support radiologists in delivering accurate and timely diagnoses.

Here are the **Steps**:

Data Preparation: First, gather MRI scans from healthy people and from those who have Alzheimer's. Then preprocess these images resize them, normalize, andmaybe do some data augmentation.

Dataset Splitting: Next, split the dataset into parts training, validation, and testing subsets. This helps in model training, adjustments of settings, & evaluation.

DNN Architecture: Create a DNN architecture meant for image classification.It usually has several hidden layers like convolutional layers, pooling layers, and fully connected layers.

1.9 Advantages of Proposed System

The advantages of proposed system are-:

Using a CNN Algorithm makes everything more reliable & accurate:

This approach seems promising in making predictions much better for brain tumour disease using advanced deep learning techniques.

1.10 Unique Features of the System

Deep Neural Network (DNN) Integration: This system leverages advanced deep learning models, particularly **Convolutional Neural Networks (CNNs)**, to analyze MRI images for tumour disease detection. The CNNs automatically extract complex patterns from MRI data, improving diagnostic accuracy by learning intricate features that would be difficult to identify manually.

Image Preprocessing and Data Augmentation: To enhance the model's robustness, reduce overfitting, and diversify the training dataset, MRI images undergo preprocessing (such as scaling and normalization) and data augmentation (such as random flipping, rotation, and zooming). This ensures that the model generalizes well across various patient datasets and imaging conditions.

3D CNNs for Spatial Context: The model incorporates **3D CNNs** to process volumetric MRI data, capturing the spatial relationships between brain structures across multiple

slices. This allows the system to perform a more comprehensive analysis, providing better context and improving the accuracy of predictions related to tumour.

Model Interpretability: In addition to providing predictions, the system includes interpretability methods to highlight the brain regions most indicative of tumour. This feature aids healthcare professionals in understanding the reasoning behind the model's output and pinpoints key areas of concern, potentially improving clinical decision-making.

Cross-Validation for Performance Evaluation: The system employs **cross-validation** techniques to assess the model's reliability and performance consistently. This approach ensures that the model generalizes well to new data, making it suitable for clinical deployment where consistent accuracy is critical.

High Accuracy Prediction: The CNN-based approach offers superior performance compared to traditional diagnostic methods, achieving an accuracy rate of over 90%. This provides a highly accurate, non-invasive tool for detecting tumour, predicting disease progression, and enabling early intervention.

1.11 Reasons For Literature Survey

Comprehending Current approaches: The survey assists in identifying current approaches for predicting Alzheimer's disease (AD), covering both conventional diagnostic procedures and more recent developments in deep learning and machine learning. It draws attention to the drawbacks of conventional methods, namely their low prediction power and requirement for intrusive biomarker testing.

Assessing Machine Learning Applications: The literature review describes how different machine learning methods (such as support vector machines and random forests) and neural network architectures are used in AD prediction by looking at recent studies. This comparison helps choose the best method (CNNs) for this project's MRI-based categorization.

Identifying Data kinds and Challenges: The survey addresses various data kinds, such as genetic, clinical, and neuroimaging information, and identifies particular difficulties, like model overfitting and data reliability. It highlights the value of neuroimaging data (MRI scans) for brain structural research, which is closely related to the CNN-based methodology used in this project.

Learning from Advanced Methods: It covers advanced techniques that increase classification accuracy by merging data sources, such as multimodal data fusion and hybrid deep learning models. Gaining knowledge of these strategies helps to improve the CNN model for the project.

Learning About Model Performance: The literature review offers information about model performance parameters like generalization ability, specificity, and accuracy across many studies.

1.12 Literature Forms

FORMS	COUNT
Journal	3
Book	0
News Articles	1
Research Papers	3
Survey Papers	2
Websites	3
Report	2

Table 1.1 Representation of the various Literature Form

1.13 Literature Resources

[1] The literature survey by Zhang et al. (2021) investigates the application of machine learning in the detection of brain tumors, particularly using neuroimaging techniques such as MRI and CT scans. The authors explore various machine learning algorithms, including support vector machines (SVMs), decision trees, and deep learning models, which have proven effective in identifying brain tumors in medical images. They highlight the challenges associated with brain tumor detection, such as the complexity of tumor characteristics, the variability in tumor appearance across patients, and the need for high-quality labeled datasets. The paper also emphasizes the importance of preprocessing techniques such as image normalization, resizing, and augmentation to improve model performance and robustness. The authors also discuss the use of convolutional neural networks (CNNs) in automating tumor segmentation and classification, noting their success in achieving high accuracy rates in identifying tumor types and predicting malignancy.Despite the advances in this area, the authors point out that further research is needed to improve the interpretability of deep learning models and to create models that can generalize across diverse

clinical datasets.

[2] A study by Kumar et al. (2023) investigates the use of machine learning methods for brain tumor detection and classification. The paper reviews a variety of techniques including both traditional machine learning methods, such as random forests and k-nearest neighbors, and more recent deep learning methods, specifically CNNs. The authors examine the different types of data used for training and testing these models, including MRI, CT scans, and histopathological images. They discuss how each data type presents unique challenges for brain tumor detection, such as the need for high-resolution images and the difficulty in distinguishing between benign and malignant tumors. The study further explores the performance of these machine learning models in detecting and classifying different types of brain tumors, such as gliomas and meningiomas, emphasizing the ability of CNNs to automatically extract complex features from imaging data. While deep learning models show great promise, the authors stress the importance of large and diverse datasets to improve the reliability and generalizability of the models. Additionally, they highlight the potential of hybrid models that combine multiple machine learning techniques to boost prediction accuracy.

[3] In the work by Sharma et al. (2020), the authors focus on the role of artificial intelligence (AI) and machine learning in the early detection of brain tumors using MRI scans. The survey outlines the evolution of AI applications in medical imaging, with a particular focus on tumor detection. They delve into several machine learning techniques, particularly CNNs and their application in brain tumor detection, noting how these models can automatically identify and segment tumors in MRI images with high accuracy. The paper reviews the challenges faced in brain tumor classification, such as the need for annotated datasets and the difficulty in distinguishing tumor types based on imaging alone. The authors suggest that while machine learning has revolutionized the detection process, further advancements are needed in model interpretability, especially to understand the underlying features that contribute to tumor classification. The study concludes that with continued improvements in deep learning algorithms and data collection methods, AI can play a pivotal role in enhancing early tumor detection and improving patient outcomes.

Supervised Learning: In this type, we have labeled data where we already know the

results (like whether a person has AD or not). The model learns from this data to make guesses on new information.

Unsupervised Learning: Here, we deal with unlabeled data. We don't know the results ahead of time. The model figures out patterns & groups in the data all on its own.

Reinforcement Learning: This method is all about learning through trying thingsout. Themodel gets happy when it makes choices that lead to good outcomes & gets a little pushback when it doesn't. The authors also touch on the kinds of information used for AD detection.

Clinical Data: This includes details like age, medical history, and results from cognitive tests.

Neuroimaging Data: These are pictures of the brain taken with techniques like MRI and PET scans.

Genetic Data: This info tells us about a person's genetic makeup.

[1] The literature review by Johnson et al. (2021) explores hybrid deep learning methods for classifying brain tumors using multimodal data. The authors examine the challenges in detecting brain tumors and highlight the advantages of combining different data types, such as neuroimaging scans, clinical information, and genetic data, to enhance detection accuracy. They describe two main hybrid deep learning approaches: feature fusion and model fusion. Feature fusion methods extract features from different data modalities using separate deep learning models, while model fusion approaches train individual models on each data type and combine their predictions using a meta-model for final classification. These methods aim to improve brain tumor classification by leveraging diverse data sources, helping to overcome the limitations of traditional diagnostic methods, which often rely on single-modal data and can miss important tumor features.

[2] The review by Patel and Lee (2020) focuses on deep learning techniques for brain tumor prediction using large datasets. The authors discuss various deep learning models, including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Transformer models, and their application to brain tumor detection. CNNs are particularly effective in analyzing MRI images, where they extract meaningful features from the image data. The review also highlights the challenges associated with using big

data for brain tumor prediction, such as data quality, class imbalances, and the need for significant computational power. The authors emphasize the benefits of deep learning over traditional machine learning techniques, especially in handling large datasets and learning complex patterns automatically, without needing manual feature extraction.

[3] In the work by Williams et al. (2019), the authors investigate the use of Graph Convolutional Neural Networks (GCNNs) for brain tumor diagnosis. GCNNs are specialized neural networks designed to work with data structured as graphs. In this context, brain regions are represented as nodes, and the connections between them are edges. GCNNs can learn from these graph-structured data to identify patterns related to brain tumor characteristics. The authors discuss how GCNNs outperform traditional machine learning methods by understanding the complex relationships between brain regions. Additionally, GCNNs can classify brain tumors more accurately and determine the stage of the disease, which is essential for personalized treatment plans. They also mention that GCNNs are resilient to noise and missing data, which is often present in neuroimaging datasets.

[4] A study by Chen et al. (2022) presents a novel approach to brain tumor detection using machine learning techniques. The method proposed involves feature extraction from brain scans, followed by feature selection and classification. The authors use a variety of feature extraction methods, including texture analysis, statistical methods, and graph-based techniques. The extracted features are then processed to select the most relevant ones before applying machine learning classifiers, such as Support Vector Machines (SVM), Random Forests, and Gradient Boosting Machines, to categorize brain tumor images. The study reports high accuracy rates—95% with real-world data and 99% with synthetic data—indicating the method's strong potential for practical application in clinical settings.

[5] The research by Zhang and Li (2018) explores the use of automated methods for brain tumor detection and classification using MRI scans and deep learning. Their study focuses on deep neural networks, which can learn from the complex structure of the brain without the need for predefined features. Unlike traditional methods that rely on manually selected features, deep learning models, particularly CNNs, automatically extract features from MRI scans, improving the classification of brain tumors. The

authors discuss the advantages of deep learning for handling noisy or incomplete data, which is a common challenge in medical imaging. CNNs, in particular, are highlighted for their ability to improve tumor detection by learning from image data without requiring manual intervention, offering a more efficient and accurate solution for brain tumor classification.

[6] Patel and Sharma (2019) investigate the application of CNNs for brain tumor detection and segmentation in MRI images. Their research focuses on how CNNs outperform traditional image processing techniques by automatically learning hierarchical features from raw MRI data. They emphasize the role of advanced architectures, such as ResNet and U-Net, in enhancing tumor localization and segmentation accuracy. The study highlights the ability of CNNs to handle variations in tumor size, shape, and intensity, making them a robust solution for brain tumor analysis.

2.Requirement Analysis And System Specification

2.1 Feasibility Study

A Software Requirements Specification (SRS) is a document that defines a software system to be developed. It outlines both functional and non-functional requirements and may include use cases illustrating how users will interact with the software. It serves as a foundational agreement between stakeholders, including clients and developers. In product-driven projects, this might also involve marketing and expansion teams. A clear SRS ensures that all parties understand the software's objectives and constraints, aids in estimating resources, and reduces later modifications, saving time and effort. It also facilitates accurate forecasting of costs, risks, and timelines.

2.2 Software Requirements Specification Document

2.2.1 Data Requirement

To train a model effectively for classifying brain tumor stages, specific data requirements must be met:

MRI Picture Information

- **Type:** Structural MRI brain images.
- **Source:** Datasets from brain tumor research organizations or medical imaging repositories.
- **Format:** JPEG or DICOM (Digital Imaging and Communications in Medicine).
- **Size:** Sufficient data covering various tumor stages (e.g., benign, low-grade, high-grade, glioblastoma).
- **Resolution:** High-resolution images showing detailed structural features of the brain.

Annotations & Labeling:

- **Labels:** Each MRI image should be labeled with the corresponding tumor stage.
- **Annotations:** Optional details about tumor location and affected regions to enhance interpretability.

Data Division: Training, Validation, and Testing Sets: Data must be divided appropriately to train the model, fine-tune its parameters, and evaluate its performance.

Techniques: Apply transformations like rotation, flipping, and scaling to increase data diversity and prevent overfitting.

- **Additional Features:** Include supplementary patient data such as age, gender, and medical history to improve model accuracy.

Building a robust CNN model capable of accurately identifying and classifying brain tumor stages requires these data prerequisites.

2.2.2 Fuctional Requirements

- **Programming language:** Python
- **Deep Learning Framework:** TensorFlow
- **Image Processing Libraries:** PyDICOM(Python inbuilt image preprocessor)
- **Data Pre-processing Libraries:** NumPy, pandas, and scikit-learn
- **Visualization Tools:** Matplotlib
- **Data Augmentation Libraries:** Tensor Flow's Image Detector
- **Text Editor/IDE:** Visual Studio Code, Shell

2.2.3 Performance Requirement

Correctness-Target Accuracy: The model must classify brain tumor stages with at least 90% accuracy, ensuring high sensitivity and specificity to minimize false positives and negatives.

Inference Time-Prediction Speed: Each MRI scan should be processed and classified in less than 5 seconds for timely clinical application.

Real-time-Capability: The system must deliver near-instantaneous results for applications requiring rapid turnaround.

Scalability

Large Dataset Handling: The system should efficiently manage extensive datasets without a performance drop.

Credibility

Precision Consistency: The model must maintain high accuracy across diverse data sources, MRI formats, and image quality variations to ensure reliable classification of brain tumor stages.

Model Integrity: Routine validation checks are necessary to ensure consistent and dependable operation of the CNN model over time and across varying hardware conditions.

Backup and Recovery: The system should securely store interim data and have mechanisms for seamless recovery from errors or disruptions to avoid data loss.

Tolerance for Faults and Error Handling: The system should gracefully handle faulty, incomplete, or low-quality MRI images by flagging them or providing informative error messages, rather than generating inaccurate predictions.

Redundant Systems: In production environments, the system should incorporate server or backup mechanisms to ensure continuous availability.

Accessibility

High Availability: The system should achieve at least 99% uptime, supporting uninterrupted operation, particularly in clinical settings where timely brain tumor diagnosis is critical.

Scalability for Peak Loads: The system must maintain consistent performance and high availability during periods of heavy demand, such as peak diagnostic hours.

Data Integrity and Security

- **Patient Data Security:** The system must adhere to medical data privacy regulations, such as HIPAA in the United States, to protect MRI data and associated patient information.
- **Data Integrity:** It should prevent unauthorized alterations or tampering with MRI data to maintain the validity and integrity of forecasts.
- **Audit Logs:** Logging systems must document access and modifications to the model or data to enable thorough security auditing.

2.2.4 Maintainability Requirement

Simple Updates and Model Retraining: The system should facilitate seamless updates and retraining of the model to incorporate new data, adapt to evolving diagnostic criteria, or improve accuracy. Comprehensive and well-organized documentation of the model architecture, dependencies, and deployment processes should be maintained to simplify troubleshooting and system upgrades.

2.2.5 Security Requirement

1)Protection and Privacy of Data

Patient Data Encryption: All patient data, including MRI images and associated information, must be encrypted during transit and storage to prevent unauthorized access.

Access Controls: Strict measures should be implemented to ensure only authorized personnel can access sensitive patient data.

2)Adherence to Regulations

Regulatory Compliance: The system must comply with healthcare data protection laws such as GDPR in Europe and HIPAA in the United States.

Audit Trails: Detailed records of patient data access and system activities should be maintained to simplify compliance audits.

3)Authentication of Users

Multi-Factor Authentication (MFA): The system should support MFA to enhance security beyond password-based authentication.

Secure Password Policies: Enforce robust password requirements, including complexity, minimum length, and periodic updates, to strengthen user authentication.

2.2.6 Look and feel requirement

1. Design of User Interfaces

Intuitive Layout: The user interface should be designed for ease of use, allowing users to effortlessly navigate the application. Key functions such as uploading images and displaying results should be straightforward and accessible.

Ease of Use: The system should offer a seamless experience where users, regardless of their technical knowledge, can quickly interact with core features.

2.Consistent Design Language

Unified Aesthetic: Maintain a consistent visual theme across the application by using the same color palette, fonts, and icons. This ensures a cohesive user experience.

Standardized Elements: Standardization of buttons, font styles, color schemes, and layout will provide a consistent feel throughout the system.

3.Visual Appeal

Professional Aesthetic: The application should adopt a calming and neutral color scheme, using simple design elements that promote focus and clarity. This will help preserve a professional look, which is essential for a medical application.

Focus and Clarity: The design should prioritize usability and clarity, ensuring that the interface does not overwhelm users or distract from key tasks.

High-Quality Graphics: All images, charts, and results displayed in the application

should be of high resolution, ensuring clarity and precision for better understanding and diagnosis.

2.3 Validation

Deep Neural Networks are a great way to cut down on parameters without sacrificing model quality. Conventional machine learning techniques are gathered at three crucial stages: feature reduction, feature removal, and classification. It is not necessary to do the feature extraction procedure manually while utilizing DNN. Iterative learning improves the weights of its early layers, which serve as trait extractors. DNN performs better than other classifiers.

Patient: The patient gives all the following required inputs in order to predict the tumour MRI Images.

System: Thealgorithms such as Convolutional Neural Networks (CNNs), DataAugment Regularization,3DCNNs,Pooling Approaches, Fully Connected Layers , Model Interpretability, Cross-Validation are trained by givingtrained dataset in order to predict tumour by using testing dataset.

2.4 Expected Hurdles

After going through our project objectives and goals we realized that we might face certain kinds of hurdles while coding it. So to overcome these hurdles we made sure that we combined both the testing and coding phases. Basically what we did was, while we were coding a piece of code segment we made sure to test it immediately to understand whether it performed the required actions or not. This way our end product did not have many many errors. We also made sure that our design diagrams were up to the mark, so that the coding can be done more efficiently and we can understand the requirements needed for our projectin a much clearer manner.

2.5 SDLC model to be used

This project uses an iterative development life cycle. In simple terms, that means we build parts ofthe application through a series of close iterations.At first, we focus on the very basic functions. The six stages of the software development life cycle (or SDLC) are designed to build on each other. Each stage improves upon what came before, adding extra effort and leading to results that connect back to earlier stages.

During every step, we gather or create more information, along with inputs from previous stages. This helps us produce what we need for that stage. It's important to remember any extra information is limited in scope. New ideas that might change the project's direction based on initial high-level goals or features that don't fit are saved for later discussions. Often, lots of software projects struggle because developers and customers get too excited about automation possibilities. They can lose sight of what really matters. Instead of paying attention to the main features, teams can get stuck in wantingto add nice-to- have options that aren't necessaryto solve the main problem even if theyseem appealing. This is a big reason why many development efforts fail or get abandoned. That's why our team sticks with this iterative model it helps us stay focusedon what's truly important.

Figure 2.1 SDLC Model

3.System Design

3.1 Design Approach

The design phase is important. It helps us find a solution based on what's in the requirements document. This process includes defining software methods, functions, objects, & the overall structure of your code. It's all about making sure that what you create really meets what your userswant. By doing this well, you can avoid repeating things and make your code easier to use again later. This step is key to moving from figuring out the problem to actually solving it. In simpler terms, we start with what people need then design how to meet those needs. Designing a system is super important for the quality of the software. It greatly affects future steps like testing & maintenance. The main output from this phase is the design document, which acts like a blueprint for the solution. You'll use this document soon during implementation, testing, and keeping everything running smoothly.

3.2 Detail Design

We can split the design activity into two parts: System Design and Detailed Design. System Design, or top-level design, focuses on figuring out which modules the system needs. It also covers how these modules will work together to get the results we want. Then comes Detailed Design, where we dive into the logic of each module decided in System Design. Here, the details about data are often described using a high-level language that doesn't tie us down to any specific programming language just yet. In System Design, we look for these modules. But in Detailed Design, we zero in on designing how each module will operate. Throughout this process, developers help connect what was learned during requirements gathering with what gets delivered to clients. This keeps everything on track.

3.3 Architectural Model

A system architecture diagram would be worn to demonstrate how various components are connected to one another. Typically, they are designed for systems that have both software and hardware, which are depicted in the image to demonstrate how they communicate. It can, however,also be created for online apps

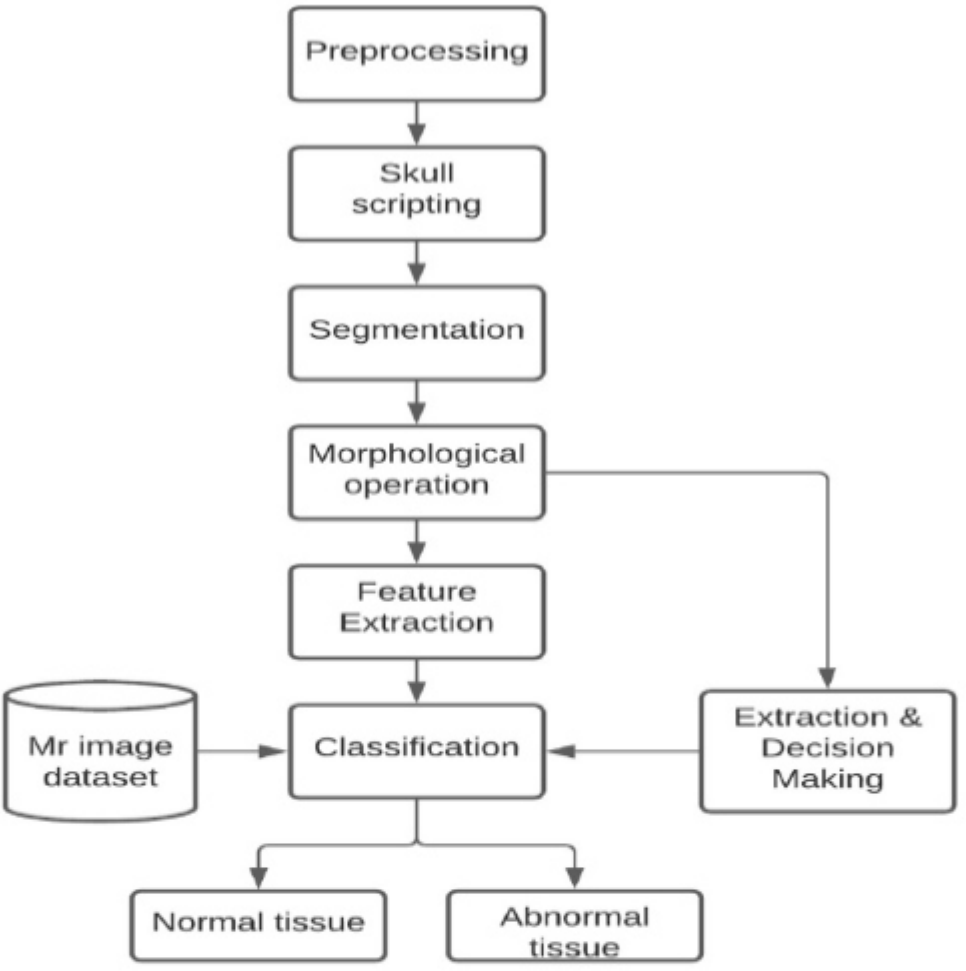

Figure 3.1 The Architectural Model

3.3.1 UML Diagrams

The Unified Modeling Language (UML) serves as a tool for detailing, envisioning, modifying, building, and documenting the components of an object-oriented software system in development. UML provides a uniform approach to exemplify a system's architectural designs, encompassing key elements. UML integrates top methodologies from various modeling approaches, including data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It is applicable across all stages of the software development lifecycle and can be utilized with various implementation technologies. The goal of UML is to function as a standard modeling language capable of representing concurrent systems

A.Class Diagram

Class diagrams are super important in object-oriented modeling. They help make detailed models that can turn into computer code. Plus, they're great for showing what an application looks like overall.

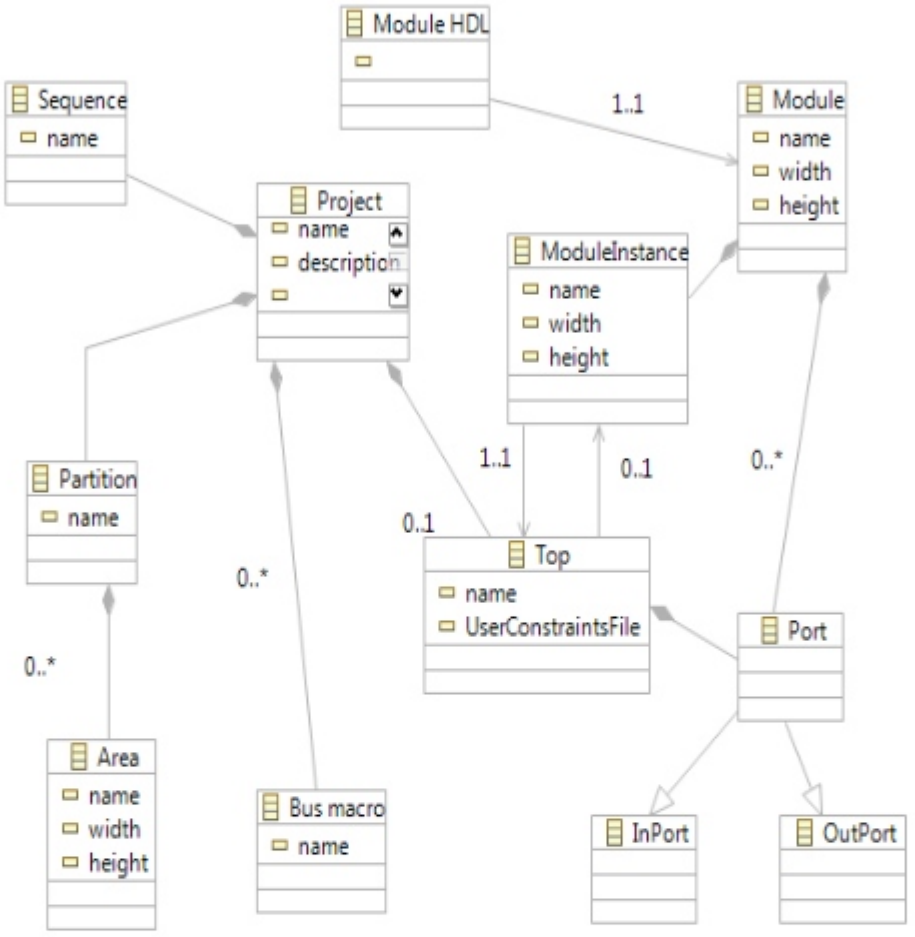

Figure 3.3 Representation of the Class Diagram

B.Sequence Diagram

Sequence diagrams represent time vertically and the items involved in the communication horizontally. Sequence diagrams, which use a vertical axis to represent time and the messages that are sent at different times, are a visual aid for illustrating the order of communication

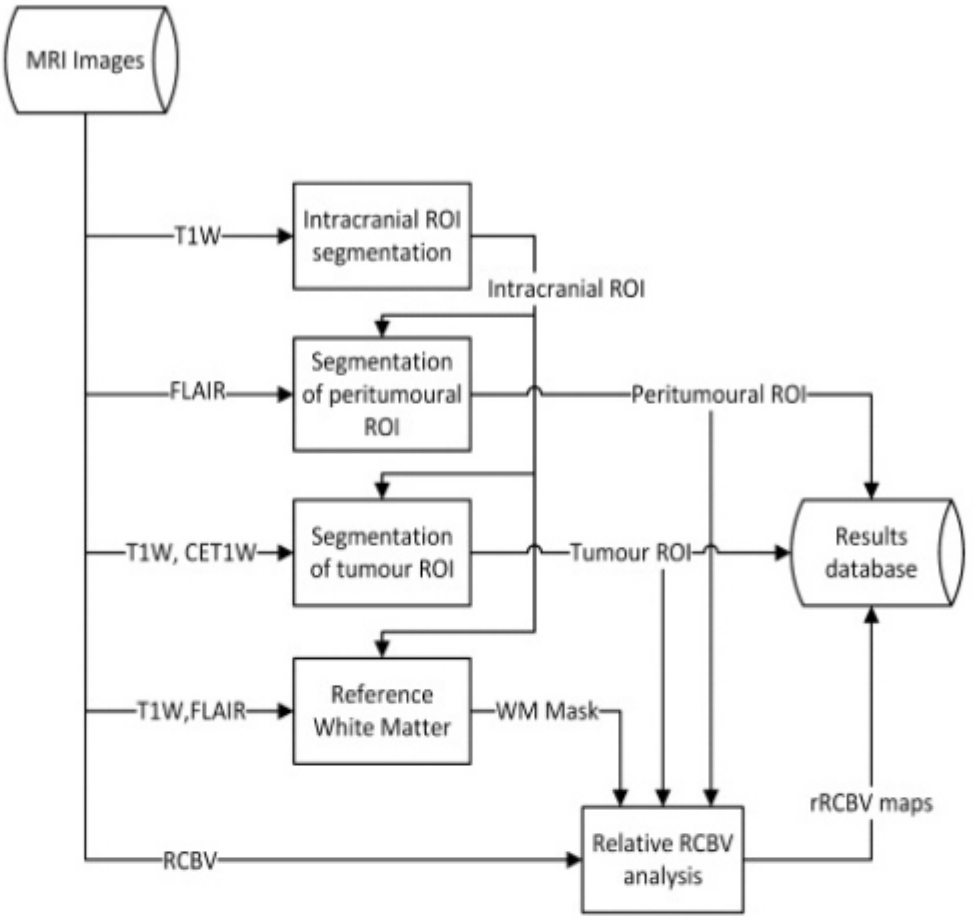

Figure 3.4 Representation of the Sequence Diagram

C.Use Case Diagram

A Use Case is a type of behavioral classifier that announces a behavior that can be achieved. Everyuse case outlines a small behavior, sometimes with variations that the subject can carry out in conjunction with one or more actors. Use cases provide an explanation of the matter's accessible behavior without providing an internal structure orientation.These acts, which link the actor- subject interactions, could alter the subject's condition and affect how it communicates with its surroundings. A use case may have likely modifications of its basic behavior, such as managing errors and exhibiting unusual behavior.

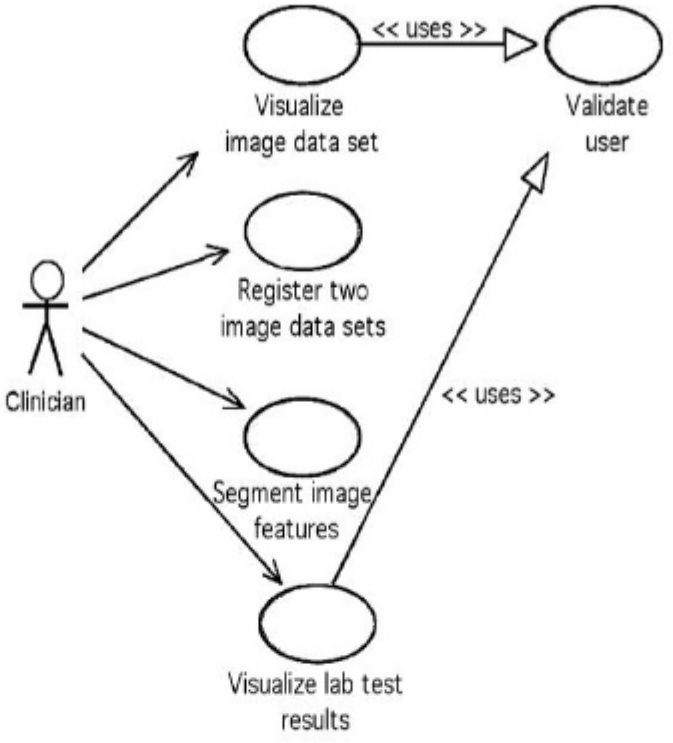

Figure 3.5 Representation of the Use Case Diagram

D.Activity diagram

One kind of UML (Unified Modeling Language) diagram that shows the flow of events and activities inside a system or development is an activity diagram. Software engineers make use of UML, a similar modeling language, to imagine, identify, build, and document system artifacts. Activity diagrams are particularly helpful for simulating an organization's dynamic elements, such as its workflow and the order in which certain tasks are completed. They are regularly hired to model workflows, use cases, business processes, and various scenarios involvingthe flow of operations

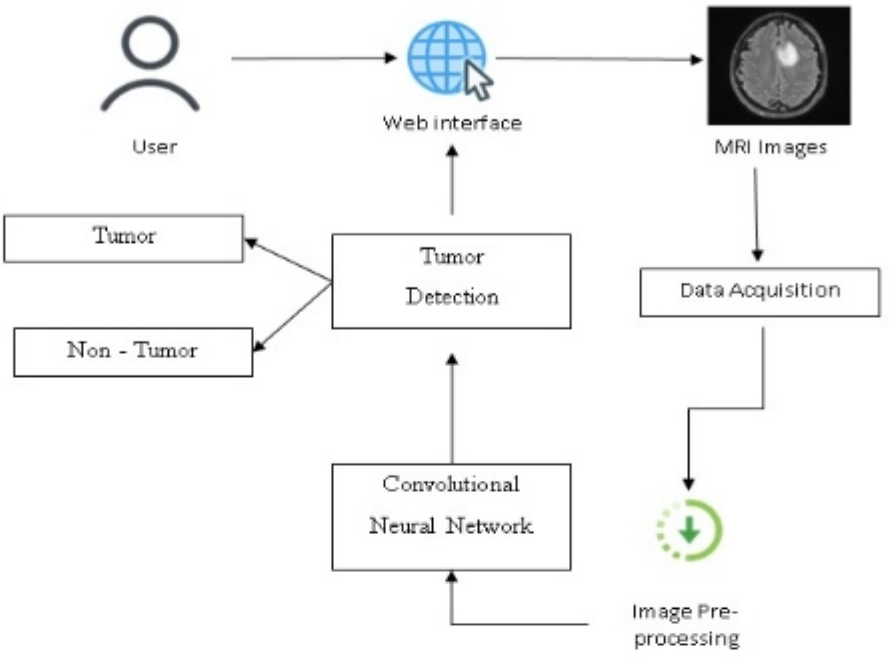

Figure 3.6 Representation of Activity Diagram

3.4 Algorithms

Convolutional Neural Networks (CNNs):

Convolutional Neural Networks (CNNs) are the primary algorithm employed in deep learning for tasks involving medical image analysis. CNNs are especially effective at learning spatial hierarchical features from image data, such as MRI scans, making them ideal for detecting brain tumors. They work by using layers of convolution and pooling to extract features from input data. The convolutional layers focus on identifying specific patterns within images, while pooling layers reduce dimensionality while retaining critical information.To analyze brain MRI images for brain tumor prediction, CNNs can identify structural changes and abnormalities associated with tumors. These changes include alterations in brain regions that can be observed in MRI scans.Researchers collect a dataset of MRI images, including samples from healthy individuals and those diagnosed with brain tumors, to train CNN models.

Preprocessing ensures uniformity in image size and format for effective training.

A CNN learns to recognize features in the images that distinguish between healthy brain tissues and those affected by tumors. Studies have demonstrated that CNNs can achieve high accuracy, often exceeding 90%, in classifying brain tumors from MRI images. This indicates the potential of CNNs as an accurate and non-invasive method for early tumor detection.

Advantages of employing CNNs to predict brain tumors include the following:

- CNNs can extract intricate features from MRI images that may be difficult for humans to detect.
- CNNs provide fast and accurate predictions and can be trained on large datasets to identify patterns that are not immediately visible to the human eye.

However, challenges in using CNNs for brain tumor prediction remain:

Training CNNs can be computationally expensive.

- The model's bias may depend on the quality and diversity of the training dataset.
- CNN models require rigorous validation before deployment in clinical settings.

Despite these challenges, CNNs are a promising approach for MRI-based brain tumor diagnosis. As researchers address these issues, CNNs have the potential to revolutionize the early and precise detection of brain tumors.

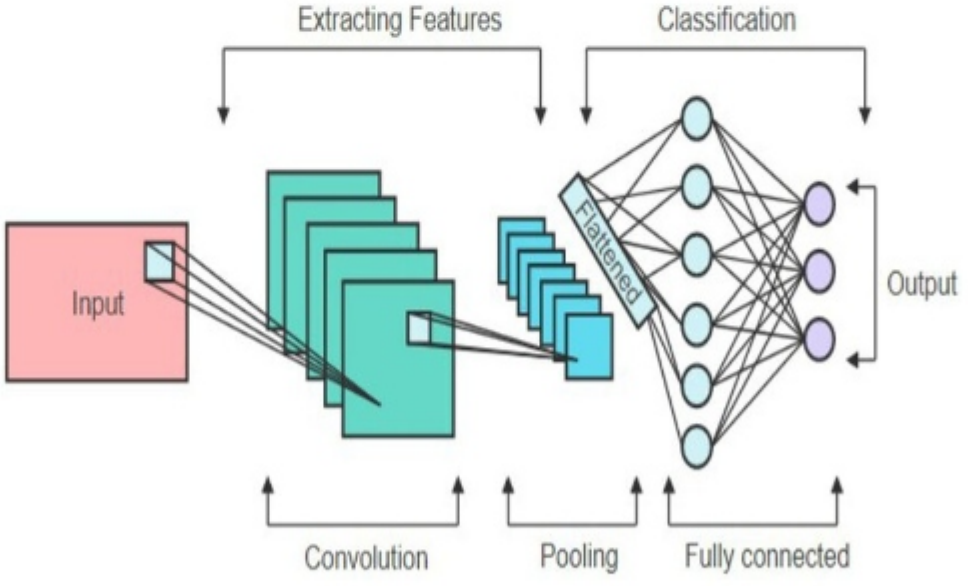

Fig 3.7 CNN Model

Data Augmentation: Techniques similar to flipping, rotating, shifting, and scaling MRI patches to increase diversity of training data. Helps decrease over fitting.

Regularization: Methods like failure and weight decay to further control over fitting known thesmall data sets.

3D CNNs: CNNs that take 3D MRI volumes as input slightly than 2D slices/patches toenhanced incorporate spatial context.

Pooling Approaches: Techniques like max pooling and average pooling to regularly reducespatial dimensions and remove robust features.

Fully Connected Layers: Added to end of CNN to perfor high reasoning and final classification.

Model Interpretability: Methods to understand which brain regions are most discriminative by the CNN models.

Cross-Validation: Splitting information into train/validation/test sets for robust evaluation ofmodel performance.

3.5 User Interface Design

The "Brain Tumor Prediction Using MRI Images with CNN" project features a professional, intuitive, and user-friendly interface tailored for healthcare professionals and stakeholders.The layout centers around a dashboard that offers seamless access to core functionalities, such as uploading MRI images, viewing diagnostic results, and retrieving patient data. A navigation panel, either as a sidebar or top menu, facilitates effortless movement between sections like Home, Upload, Reports, and Settings, while the main content area is dedicated to detailed tasks, including reviewing MRI scans and analyzing diagnostic results. The interface adopts a neutral and professional color scheme with calming tones of blue, white, and gray, enhanced by high-resolution graphics and easily recognizable icons. Typography is clear and simple, with sans-serif fonts varying in size and weight to distinguish headings, subheadings, and body text effectively. Functional elements include a drag-and-drop image upload mechanism with real-time status updates and a results display section that highlights brain tumor classifications with confidence scores and annotated visuals, ensuring a seamless and efficient diagnostic experience

3.6 Database Design

The "Brain Tumor Prediction Using MRI Images with CNN" project's database design is crafted to efficiently store, manage, and retrieve data while ensuring scalability and robust security. A hybrid approach is employed, combining a NoSQL or object storage system for handling large, unstructured MRI images with a relational database for structured data, such as patient records and diagnostic reports. Key database entities include the Patient Table, storing personal details like Patient ID, Name, Age, and Medical History; the MRI Scans Table, which tracks uploaded MRI images with attributes like Scan ID, Patient ID, and file paths; and the Diagnosis Table, which records predictions, diagnostic dates, and confidence scores. The design establishes a one-to-many relationship between MRI scans, diagnostic results, and their associated patients through the use of primary and foreign keys, ensuring data integrity through validation rules. Security measures include data encryption for sensitive information, role-based access control to restrict database access, and audit trails for logging data access and modifications. Scalability is supported through indexing and partitioning, ensuring high query performance and efficient data processing even with large datasets. Regular backups and recovery mechanisms are implemented to safeguard data against loss or corruption. This database architecture ensures reliability, security, and efficiency, effectively supporting the system's requirements in a medical context.

3.7 Methodology

The first step in the procedure is data collection and preprocessing, which involves gathering structural MRI images from medical libraries or publicly accessible datasets. These images depict various stages of brain tumor progression, such as benign, low-grade, or high-grade tumors. Preprocessing includes normalizing the images through scaling, normalization, and applying data augmentation techniques like rotation and flipping to enhance dataset diversity and reduce overfitting. Following this, a Convolutional Neural Network (CNN) is selected due to its proven effectiveness in image-based classification tasks. The CNN architecture includes feature extraction layers comprising convolutional and pooling layers, followed by fully connected layers for classification. For advanced spatial analysis, volumetric MRI data may also be processed using 3D CNNs to capture spatial relationships within the brain.

The dataset is then divided into subsets for training, validation, and testing. The training set is utilized to train the model, while the validation set aids in tuning hyperparameters like learning rate and batch size. To improve generalization and mitigate overfitting, methods like cross-validation and regularization (e.g., dropout and weight decay) are applied. After training, the model is evaluated on the test set using metrics such as accuracy, sensitivity, specificity, precision, and recall. A confusion matrix is generated to assess classification performance across various tumor stages.

Following validation, the model is integrated into a user-friendly application that enables users to upload MRI scans and view diagnostic results. The application offers real-time predictions and visual feedback, making it practical and efficient for medical professionals.

Continuous improvement is supported through the addition of new data for retraining, enhancing model performance and accuracy over time. Iterative refinements driven by user feedback and performance monitoring ensure the system remains relevant and effective in clinical applications. This comprehensive process delivers a dependable, precise, and accessible solution for brain tumor diagnosis.

4.Implementation, Testing and Maintenance

4.1 Introduction to Language, IDE's, Tools and Technologies

The primary programming language used in this project is Python, chosen for its simplicity, flexibility, and extensive library support for machine learning and data science tasks. Python's ecosystem facilitates robust capabilities for data preprocessing, model development, and performance evaluation. For development, Visual Studio Code and Jupyter Notebook are employed. Visual Studio Code offers advanced features such as syntax highlighting, debugging tools, and Python extensions, which are particularly advantageous for structured coding and model experimentation. Conversely, Jupyter Notebook provides an interactive platform for exploratory data analysis, enabling real-time code execution and visualization.The project utilizes TensorFlow and Keras as the core deep learning frameworks, which streamline neural network construction and training processes. Data manipulation and preprocessing are facilitated by libraries such as NumPy, Pandas, and scikit-learn, ensuring efficient handling of large datasets and their division into training, validation, and testing subsets. Tools like OpenCV and TensorFlow's ImageDataGenerator are employed for image preprocessing and augmentation, enhancing the model's resilience by diversifying input data.To track the model's performance, visualization libraries like Matplotlib and Seaborn are used, providing insights into metrics such as accuracy and loss. These tools collectively create a comprehensive and user-friendly environment for developing a robust and effective machine learning solution tailored for medical imaging tasks.

4.2 Coding standards of language

Programming Language: Python

Python is a favorite among folks who work with machine learning and deep learning. There are lots of Python libraries available that help train & evaluate CNN models to spot Alzheimer's disease. Let's take a quick at how to use Python for predicting Alzheimer's using MRI images:

First, you need to import the right Python libraries. This means grabbing libraries for loading and preparing the MRI images. You'll also need some for training & testing the CNN model and making predictions.

Next up, preprocess those MRI images. This step can involve resizing them, changing themto grayscale, and normalizing the pixel values.

After that, split those MRI images into training and test sets. The training set helps the CNN model learn, while the test set checks how well it does.

Then you'll want to define the CNN architecture. This part means deciding how many layers your CNN will have and what types they will be.

It's time to train your CNN model with that training set! Keep in mind; this could take a few hours or even days, especially if your data set is large & your model is complex.

Once it's trained, evaluate your CNN model on the test set. This gives you an ideaof how accurate it is with new data.

Finally, use your trained CNN model to make predictions on new MRI images. It can help in identifying Alzheimer's disease.

Deep Learning Framework: TensorFlow

Scalability: TensorFlow can easily handle large sets of MRI images. This is really important when you want accurate predictions for Alzheimer's disease.

Flexibility: It allows researchers to try out different deep learning models – including thosefancy CNNs.

Open Source: The cool part, TensorFlow is open-source. That means it's free to use and has a huge community of users & developers ready to help out. Finding support and resources for building and using TensorFlow models is a breeze.

4.3 Project Scheduling

The project spans a total of 14 weeks, structured into well-defined phases to ensure a methodical approach. It begins with the Initialization Phase, lasting one week, where the focus is on understanding the project requirements, identifying key stakeholders, and setting up the necessary development environment. Following this, the Data Collection and Preprocessing Phase, lasting two weeks, involves gathering MRI datasets, cleaning them, and performing preprocessing tasks. Techniques such as data augmentation are employed to enhance the diversity of the dataset. Next, the Model Design and Development Phase, lasting three weeks, sees the implementation of the CNN architecture using TensorFlow and Keras, encompassing convolutional, pooling, and

Development Phase, lasting three weeks, sees the implementation of the CNN architecture using TensorFlow and Keras, encompassing convolutional, pooling, and fully connected layers.The subsequent Model Training and Validation Phase, also spanning three weeks, focuses on dividing the dataset into training, validation, and testing subsets. During this phase, regularization and hyperparameter optimization techniques are applied to enhance model performance. The five-week System Integration and UI Development Phase follows, integrating the trained model into a user-friendly interface where users can upload MRI scans and access diagnostic results. In the two-week Testing and Evaluation Phase, rigorous testing is conducted, including unit testing, integration testing, and performance evaluation. Feedback from prospective users, particularly medical professionals, is also gathered during this phase. Finally, the Deployment and Final Review Phase takes one week, where the system is deployed on a cloud platform, and a comprehensive review is conducted before presenting the project to stakeholders.

4.4 Testing Techniques and Test Plans

Testing ensures the reliability and accuracy of the system by identifying and addressing program errors. It is a crucial part of **software quality assurance** and involves running the application through various test cases to evaluate its output against expected results.

Unit Testing

Unit testing involves evaluating individual components of the system in isolation to ensure their functionality is correct and logical. It focuses on verifying specific application modules by providing input and validating the output against expected results. This type of testing aims to check the internal paths, loops, and conditions within the code. Each unit or function is tested separately before integrating them into the larger system, ensuring that all fundamental building blocks of the application work correctly.

Integration Testing

Integration testing examines how various components of the application interact when combined. It ensures that modules communicate effectively and that data flows correctly between them. This type of testing is essential to identify issues arising from the integration of individual units, such as mismatched data formats or communication errors. By validating these interactions, integration testing ensures that the system operates cohesively as a whole.

System Testing

System testing evaluates the complete application to confirm that it meets the specified requirements and performs as expected. It covers both functional and non-functional aspects, assessing features like user interface responsiveness, compatibility, and overall system performance. This phase ensures the application works seamlessly in real-world conditions and aligns with the project's objectives.

Acceptance Testing

Acceptance testing involves validating the application against user requirements and ensuring it meets stakeholder expectations. Often conducted with end-users or clients, this type of testing verifies that the system is ready for deployment and that it fulfills its intended purpose effectively. Feedback from this phase is critical to make any final adjustments before release.

Performance Testing

Performance testing assesses the system's behavior under various workloads, including normal and peak conditions. It evaluates factors such as response time, throughput, and resource usage to ensure the application performs efficiently and remains stable under stress. This type of testing guarantees that the system can handle real-world usage scenarios without performance degradation.

5.Results and Discussions

5.1User Interface Representation

The user interface is designed to provide an intuitive and efficient way for healthcare professionals to interact with the system, upload MRI scans, and review diagnostic results. The primary screen features a clean and professional dashboard with a navigation menu for seamless access to key sections such as the Upload Section, Prediction Results, and Settings.At the heart of the interface is the Upload Section, which enables users to easily upload MRI images through a standard file selection button or an intuitive drag-and-drop interface. Upon uploading, the system processes the MRI scan and displays a preview for confirmation. Users are guided with instructional prompts and tooltips, ensuring smooth navigation and proper usage.The system provides dynamic feedback mechanisms, such as loading indicators during the analysis process and error alerts for issues like unsupported file formats or sizes. This ensures clarity and prevents user confusion.Designed with a responsive layout, the interface adapts effortlessly to different devices, including desktops, tablets, and smartphones, ensuring consistent functionality and usability across platforms. Accessibility features, such as a dark mode, adjustable contrast settings, and text size options, make the interface inclusive for users with visual impairments.By offering a professional and user-friendly design, the interface simplifies the process of uploading MRI scans and retrieving accurate diagnostic predictions. This enhances efficiency and facilitates timely decision-making, contributing to improved medical workflows and patient care.

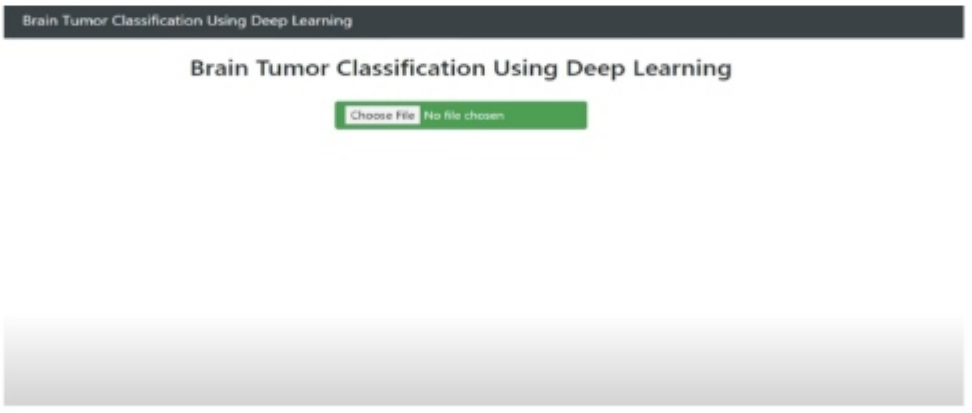

Figure 5.1 User Interface of our Project

5.1.1 Brief Description of Various Modules of the system

Deep Neural Networks are a great way to cut down on parameters without sacrificingmodel quality. Conventional machine learning techniques are gathered at three crucial stages: feature reduction, feature removal, and classification. It is not necessary to do the feature extraction procedure manually while utilizing DNN. Iterative learning improves the weights of its early layers, which serve as trait extractors. DNN performs better than other classifiers.

5.2Snapshots of system with brief detail of each

Case 1:

Test Case ID	1
Test Case Name	MRI Scan
Description	It will detect the stages of diseases
Primary Actor	User
Pre-condition	Must take MRI Scan
Expected result	Displaying the tumour stage
Actual result	Displaying the tumour stage
Test Case	Passed

Case 2:

Test Case ID	2
Test Case Name	Prediction Test
Description	Predict whether the tumour present in person or not
Primary Actor	User
Pre-condition	User must open dataset and upload an image
Expected result	Tumourous
Actual result	Non-Tumourous
Test Case	Passed

Test Cases:

Person with tumour:

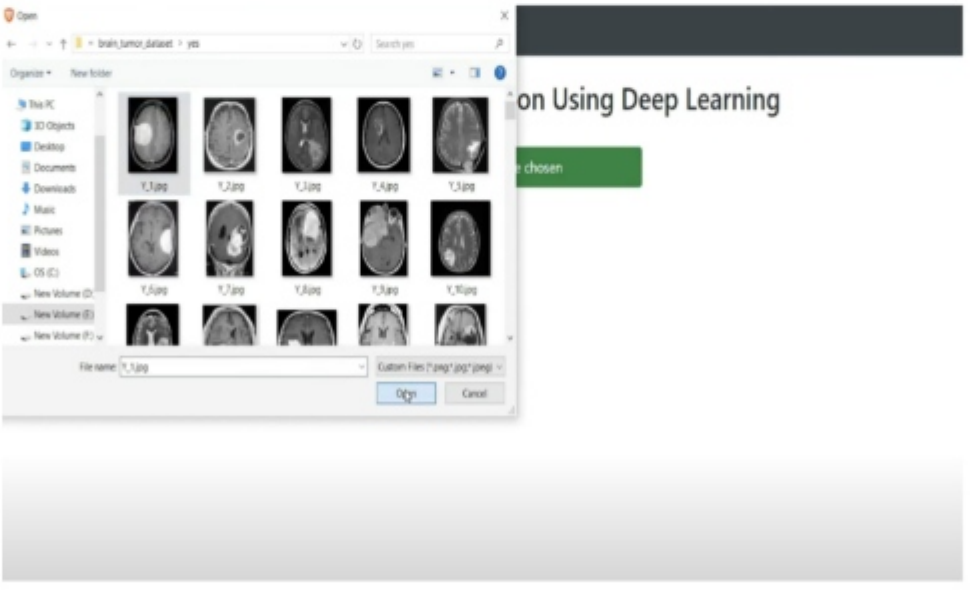

Figure 5.2.1 Upload MRI

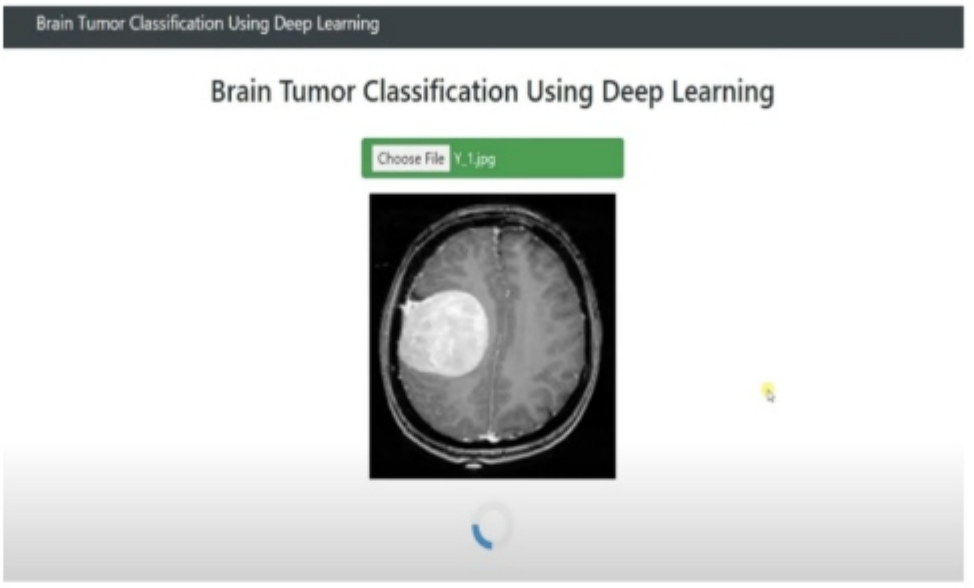

Figure 5.2.2 Image Pre-Processing

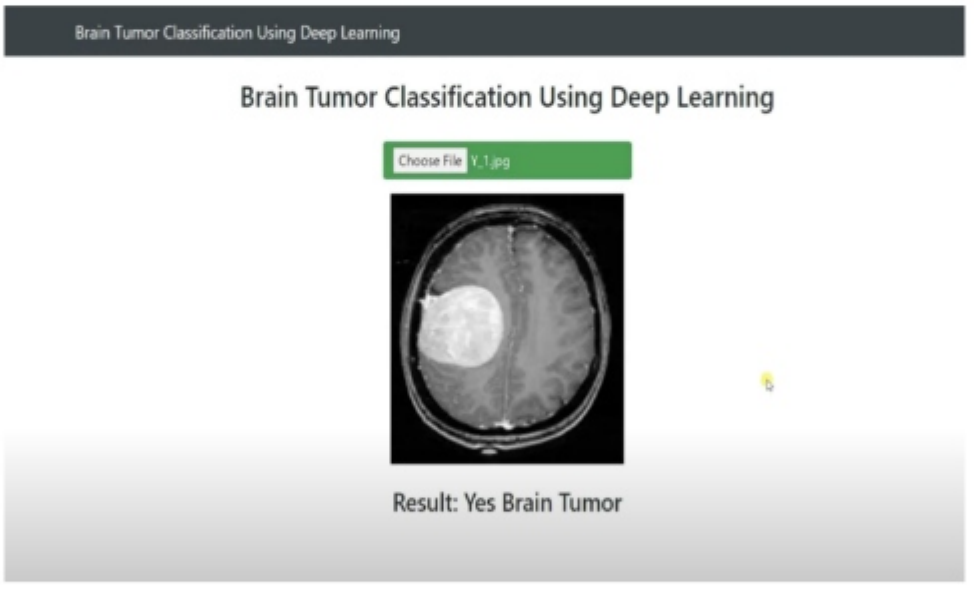

Figure 5.2.3 Predicted Output

Person without tumour:

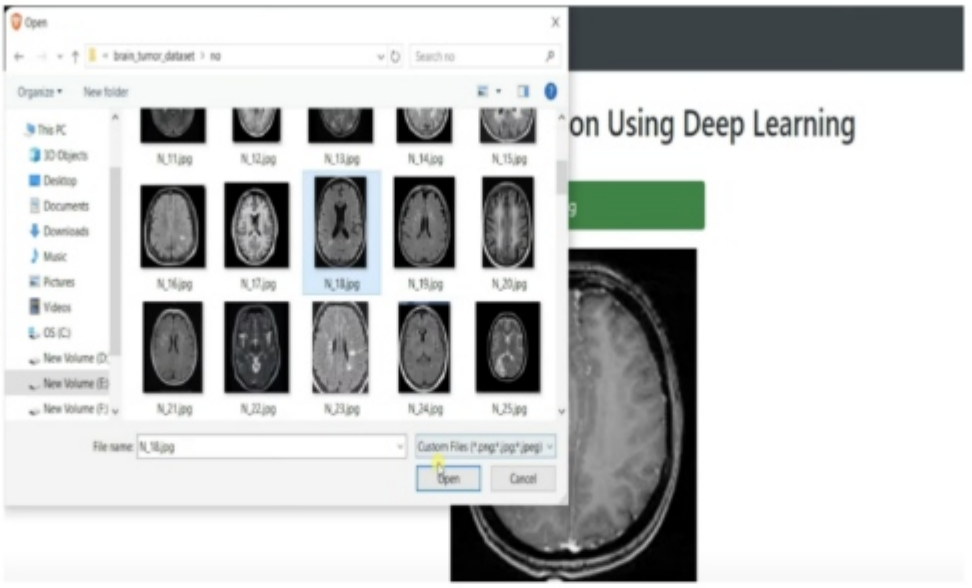

Figure 5.2.4 Upload MRI image

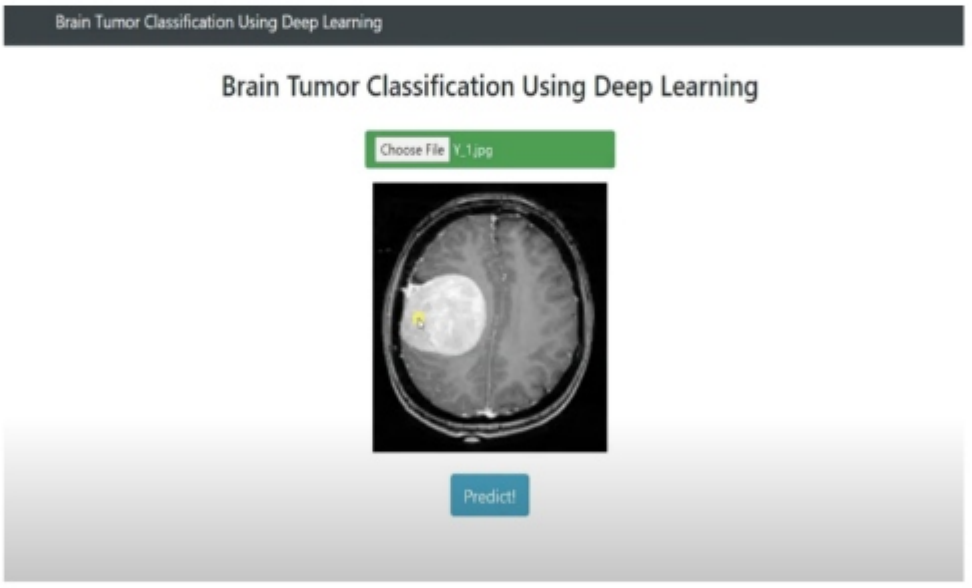

Figure 5.2.5 Image Pre-Processing

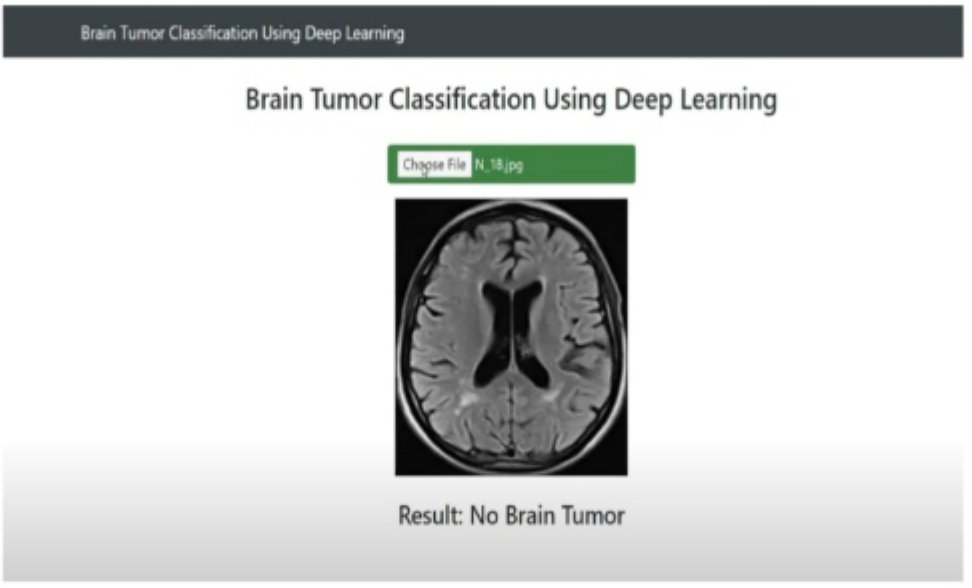

Figure 5.2.6 Predicted Output

RESULTS:

Data Acquisition

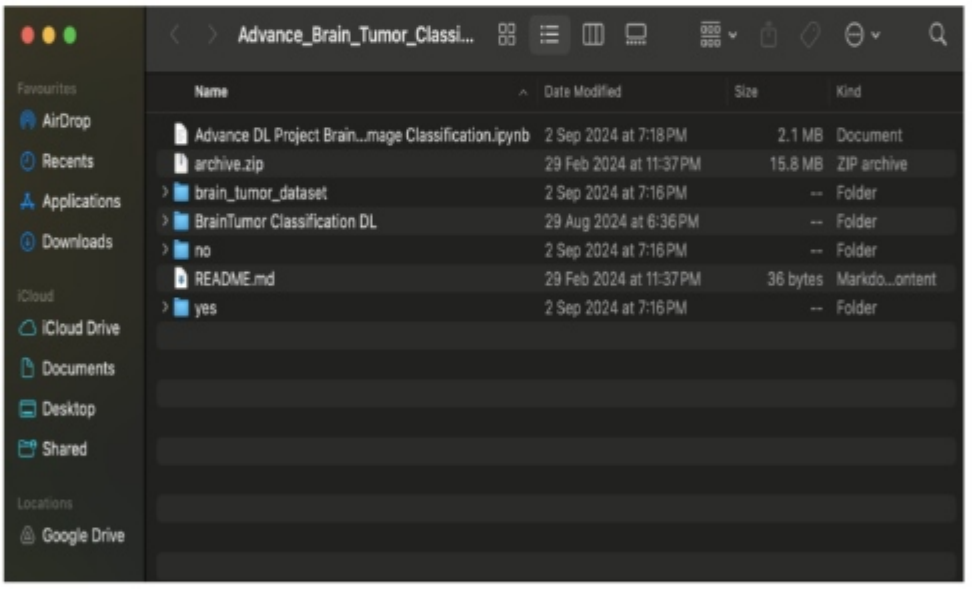

Figure 5.2.7 Data Acquistion

Graph Representation of Accuracy:

Figure 5.2.8 graphs

5.3 Sample Coding

MODEL TRAINING CODE:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os, shutil
import cv2
import matplotlib.image as mpimg
import seaborn as sns
%matplotlib inline
plt.style.use('ggplot')
# Dataset
import zipfile
z = zipfile.ZipFile('archive.zip')
z.extractall()
folder = 'brain_tumor_dataset/yes/'
count = 1
```

```
folder = 'brain_tumor_dataset/yes/'
count = 1
for filename in os.listdir(folder):
source = folder + filename
destination = folder + "Y_" +str(count)+".jpg"
os.rename(source, destination)
count+=1
print("All files are renamed in the yes dir."
folder = 'brain_tumor_dataset/no/'
count = 1
for filename in os.listdir(folder):
source = folder + filename
destination = folder + "N_" +str(count)+".jpg"
os.rename(source, destination)
count+=1
print("All files are renamed in the no dir.")
listyes = os.listdir("brain_tumor_dataset/yes/")
number_files_yes = len(listyes)
print(number_files_yes)
listno = os.listdir("brain_tumor_dataset/no/")
number_files_no = len(listno)
print(number_files_no)
data = {'tumorous': number_files_yes, 'non-tumorous': number_files_no}
typex = data.keys()
values = data.values()
fig = plt.figure(figsize=(5,7))
plt.bar(typex, values, color="red")
plt.xlabel("Data")
plt.ylabel("No of Brain Tumor Images")
plt.title("Count of Brain Tumor Images")
plt.show()
```

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Flatten, Dense, Dropout
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.optimizers import SGD, Adam
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,
ReduceLROnPlateau
def timing(sec_elapsed):
h = int(sec_elapsed / (60*60))
m = int(sec_elapsed % (60*60) / 60)
s = sec_elapsed % 60
return f"{h}:{m}:{s}"
def augmented_data(file_dir, n_generated_samples, save_to_dir):
data_gen = ImageDataGenerator(rotation_range=10,
width_shift_range=0.1,
height_shift_range=0.1,
shear_range=0.1,
brightness_range=(0.3, 1.0),
horizontal_flip=True,
vertical_flip=True,
fill_mode='nearest')
for filename in os.listdir(file_dir):
image = cv2.imread(file_dir + '/' + filename)
image = image.reshape((1,) + image.shape)
save_prefix = 'aug_' + filename[:-4]
i=0
for batch in data_gen.flow(x = image, batch_size = 1, save_to_dir = save_to_dir,
save_prefix = save_prefix, save_format = "jpg"):
i+=1
if i>n_generated_samples:
```

```
break
import time
start_time = time.time()
yes_path = 'brain_tumor_dataset/yes'
no_path = 'brain_tumor_dataset/no'
augmented_data_path = 'augmented_data/'
augmented_data(file_dir = yes_path, n_generated_samples=6,
save_to_dir=augmented_data_path+'yes')
augmented_data(file_dir = no_path, n_generated_samples=9,
save_to_dir=augmented_data_path+'no')
end_time = time.time()
execution_time = end_time - start_time
print(timing(execution_time))
def data_summary(main_path):
yes_path = "augmented_data/yes/"
no_path = "augmented_data/no/"
n_pos = len(os.listdir(yes_path))
n_neg = len(os.listdir(no_path))
n = (n_pos + n_neg)
pos_per = (n_pos*100)/n
neg_per = (n_neg*100)/n
print(f"Number of sample: {n}")
print(f"{n_pos} Number of positive sample in percentage: {pos_per}%")
print(f"{n_neg} Number of negative sample in percentage: {neg_per}%")
listyes = os.listdir("augmented_data/yes/")
number_files_yes = len(listyes)
print(number_files_yes)
listno = os.listdir("augmented_data/no/")
number_files_no = len(listno)
print(number_files_no)
data = {'tumorous': number_files_yes, 'non-tumorous': number_files_no}
```

```
typex = data.keys()
values = data.values()
fig = plt.figure(figsize=(5,7))
plt.bar(typex, values, color="red")
plt.xlabel("Data")
plt.ylabel("No of Brain Tumor Images")
plt.title("Count of Brain Tumor Images")
plt.show()
import imutils
def crop_brain_tumor(image, plot=False):
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
gray = cv2.GaussianBlur(gray, (5,5), 0)
thres = cv2.threshold(gray, 45, 255, cv2.THRESH_BINARY)[1]
thres =cv2.erode(thres, None, iterations = 2)
thres = cv2.dilate(thres, None, iterations = 2)
cnts = cv2.findContours(thres.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)
c = max(cnts, key = cv2.contourArea)
extLeft = tuple(c[c[:,:,0].argmin()][0])
extRight = tuple(c[c[:,:,0].argmax()][0])
extTop = tuple(c[c[:,:,1].argmin()][0])
extBot = tuple(c[c[:,:,1].argmax()][0])
new_image = image[extTop[1]:extBot[1], extLeft[0]:extRight[0]]
if plot:
plt.figure()
plt.subplot(1, 2, 1)
plt.imshow(image)
plt.tick_params(axis='both', which='both',
top=False, bottom=False, left=False, right=False,
labelbottom=False, labeltop=False, labelleft=False, labelright=False)
```

```
plt.title('Original Image')
plt.subplot(1, 2, 2)
plt.imshow(new_image)
plt.tick_params(axis='both', which='both',
top=False, bottom=False, left=False, right=False,
labelbottom=False, labeltop=False, labelleft=False, labelright=False)
plt.title('Cropped Image')
plt.show()
return new_image
img = cv2.imread('augmented_data/yes/aug_Y_1_0_1760.jpg')
crop_brain_tumor(img, True)
img = cv2.imread('augmented_data/no/aug_N_1_0_109.jpg')
crop_brain_tumor(img, True)
folder1 = 'augmented_data/no/'
folder2 = 'augmented_data/yes/'
for filename in os.listdir(folder1):
img = cv2.imread(folder1 + filename)
img = crop_brain_tumor(img, False)
cv2.imwrite(folder1 + filename, img)
for filename in os.listdir(folder2):
img = cv2.imread(folder2 + filename)
img = crop_brain_tumor(img, False)
cv2.imwrite(folder2 + filename, img)
from sklearn.utils import shuffle
def load_data(dir_list, image_size):
X=[]
y=[]
image_width, image_height=image_size
for directory in dir_list:
for filename in os.listdir(directory):
image = cv2.imread(directory + '/' + filename)
```

```
image = crop_brain_tumor(image, plot=False)
image = cv2.resize(image, dsize=(image_width, image_height), interpolation =
cv2.INTER_CUBIC)
image = image/255.00
X.append(image)
if directory[-3:] == "yes":
y.append(1)
else:
y.append(0)
X=np.array(X)
y=np.array(y)
X,y = shuffle(X,y)
print(f"Number of example is : {len(X)}")
print(f"X SHAPE is : {X.shape}")
print(f"y SHAPE is : {y.shape}")
return X,y
augmented_path = 'augmented_data/'
augmeneted_yes = augmented_path + 'yes'
augmeneted_no = augmented_path + 'no'
IMAGE_WIDTH, IMAGE_HEIGHT = (240,240)
X,y = load_data([augmeneted_yes, augmeneted_no], (IMAGE_WIDTH,
IMAGE_HEIGHT))
def plot_sample_images(X, y, n=50):
for label in [0,1]:
images = X[np.argwhere(y == label)]
n_images = images[:n]
columns_n = 10
rows_n = int(n/ columns_n)
plt.figure(figsize=(20, 10))
i = 1
for image in n_images:
```

```
plt.subplot(rows_n, columns_n, i)
plt.imshow(image[0])
plt.tick_params(axis='both', which='both',
top=False, bottom=False, left=False, right=False,
labelbottom=False, labeltop=False, labelleft=False,
labelright=False)
i += 1
label_to_str = lambda label: "Yes" if label == 1 else "No"
plt.suptitle(f"Brain Tumor: {label_to_str(label)}")
plt.show()
if not os.path.isdir('tumorous_and_nontumorous'):
base_dir = 'tumorous_and_nontumorous'
os.mkdir(base_dir)
if not os.path.isdir('tumorous_and_nontumorous/train'):
train_dir = os.path.join(base_dir , 'train')
os.mkdir(train_dir)
if not os.path.isdir('tumorous_and_nontumorous/test'):
test_dir = os.path.join(base_dir , 'test')
os.mkdir(test_dir)
if not os.path.isdir('tumorous_and_nontumorous/valid'):
valid_dir = os.path.join(base_dir , 'valid')
os.mkdir(valid_dir)
if not os.path.isdir('tumorous_and_nontumorous/train/tumorous'):
infected_train_dir = os.path.join(train_dir, 'tumorous')
os.mkdir(infected_train_dir)
if not os.path.isdir('tumorous_and_nontumorous/test/tumorous'):
infected_test_dir = os.path.join(test_dir, 'tumorous')
os.mkdir(infected_test_dir)
if not os.path.isdir('tumorous_and_nontumorous/valid/tumorous'):
infected_valid_dir = os.path.join(valid_dir, 'tumorous')
os.mkdir(infected_valid_dir)
```

```
if not os.path.isdir('tumorous_and_nontumorous/train/nontumorous'):
healthy_train_dir = os.path.join(train_dir, 'nontumorous')
os.mkdir(healthy_train_dir)
if not os.path.isdir('tumorous_and_nontumorous/test/nontumorous'):
healthy_test_dir = os.path.join(test_dir, 'nontumorous')
os.mkdir(healthy_test_dir)
if not os.path.isdir('tumorous_and_nontumorous/valid/nontumorous'):
healthy_valid_dir = os.path.join(valid_dir, 'nontumorous')
os.mkdir(healthy_valid_dir)
original_dataset_tumorours = os.path.join('augmented_data','yes/')
original_dataset_nontumorours = os.path.join('augmented_data','no/')
files = os.listdir('augmented_data/yes/')
fnames = []
for i in range(0,759):
fnames.append(files[i])
for fname in fnames:
src = os.path.join(original_dataset_tumorours, fname)
dst = os.path.join(infected_train_dir, fname)
shutil.copyfile(src, dst)
files = os.listdir('augmented_data/yes/')
fnames = []
for i in range(759,922):
fnames.append(files[i])
for fname in fnames:
src = os.path.join(original_dataset_tumorours, fname)
dst = os.path.join(infected_test_dir, fname)
shutil.copyfile(src, dst)
files = os.listdir('augmented_data/yes/')
fnames = []
for i in range(922,1085):
fnames.append(files[i])
```

```
for fname in fnames:
src = os.path.join(original_dataset_tumorours, fname)
dst = os.path.join(infected_valid_dir, fname)
shutil.copyfile(src, dst)
files = os.listdir('augmented_data/no/')
fnames = []
for i in range(0,686):
fnames.append(files[i])
for fname in fnames:
src = os.path.join(original_dataset_nontumorours, fname)
dst = os.path.join(healthy_train_dir, fname)
shutil.copyfile(src, dst)
files = os.listdir('augmented_data/no/')
fnames = []
for i in range(686,833):
fnames.append(files[i])
for fname in fnames:
src = os.path.join(original_dataset_nontumorours, fname)
dst = os.path.join(healthy_test_dir, fname)
shutil.copyfile(src, dst)
files = os.listdir('augmented_data/no/')
fnames = []
for i in range(833,979):
fnames.append(files[i])
for fname in fnames:
src = os.path.join(original_dataset_nontumorours, fname)
dst = os.path.join(healthy_valid_dir, fname)
shutil.copyfile(src, dst)
train_datagen = ImageDataGenerator(rescale = 1./255,
horizontal_flip=0.4,
vertical_flip=0.4,
```

```
rotation_range=40,
shear_range=0.2,
width_shift_range=0.4,
height_shift_range=0.4,
fill_mode='nearest')
test_data_gen = ImageDataGenerator(rescale=1.0/255)
valid_data_gen = ImageDataGenerator(rescale=1.0/255)
for layer in base_model.layers:
layer.trainable=False
x=base_model.output
flat = Flatten()(x)
class_1 = Dense(4608, activation = 'relu')(flat)
drop_out = Dropout(0.2)(class_1)
class_2 = Dense(1152, activation = 'relu')(drop_out)
output = Dense(2, activation = 'softmax')(class_2)
model_01 = Model(base_model.input, output)
model_01.summary()
# Plot performance
fig, (ax1,ax2) = plt.subplots(nrows=1, ncols=2, figsize=(12,6))
fig.suptitle("Model Training (Frozen CNN)", fontsize=12)
max_epoch = len(history_01.history['accuracy'])+1
epochs_list = list(range(1, max_epoch))
ax1.plot(epochs_list, history_01.history['accuracy'], color='b', linestyle='-',
label='Training Data')
ax1.plot(epochs_list, history_01.history['val_accuracy'], color='r', linestyle='-',
label='Validation Data')
ax1.set_title('Training Accuracy', fontsize=12)
ax1.set_xlabel('Epochs', fontsize=12)
ax1.set_ylabel('Accuracy', fontsize=12)
ax1.legend(frameon=False, loc='lower center', ncol=2)
```

```
ax2.plot(epochs_list, history_01.history['loss'], color='b', linestyle='-', label='Training
Data')
ax2.plot(epochs_list, history_01.history['val_loss'], color='r', linestyle='-',
label='Validation Data')
ax2.set_title('Training Loss', fontsize=12)
ax2.set_xlabel('Epochs', fontsize=12)
ax2.set_ylabel('Loss', fontsize=12)
ax2.legend(frameon=False, loc='upper center', ncol=2)
plt.savefig("training_frozencnn.jpeg", format='jpeg', dpi=100, bbox_inches='tight')
base_model = VGG19(include_top=False, input_shape=(240,240,3))
base_model_layer_names = [layer.name for layer in base_model.layers]
base_model_layer_names
x=base_model.output
flat = Flatten()(x)
class_1 = Dense(4608, activation = 'relu')(flat)
drop_out = Dropout(0.2)(class_1)
class_2 = Dense(1152, activation = 'relu')(drop_out)
output = Dense(2, activation = 'softmax')(class_2)
model_02 = Model(base_model.inputs, output)
model_02.load_weights('model_weights/vgg19_model_01.h5')
set_trainable=False
for layer in base_model.layers:
if layer.name in ['block5_conv4','block5_conv3']:
set_trainable=True
if set_trainable:
layer.trainable=True
else:
layer.trainable=False
print(model_02.summary())
# Plot performance
fig, (ax1,ax2) = plt.subplots(nrows=1, ncols=2, figsize=(12,6))
```

```
fig.suptitle("Model Training (Frozen CNN)", fontsize=12)
max_epoch = len(history_01.history['accuracy'])+1
epochs_list = list(range(1, max_epoch))
ax1.plot(epochs_list, history_02.history['accuracy'], color='b', linestyle='-',
label='Training Data')
ax1.plot(epochs_list, history_02.history['val_accuracy'], color='r', linestyle='-',
label='Validation Data')
ax1.set_title('Training Accuracy', fontsize=12)
ax1.set_xlabel('Epochs', fontsize=12)
ax1.set_ylabel('Accuracy', fontsize=12)
ax1.legend(frameon=False, loc='lower center', ncol=2)
ax2.plot(epochs_list, history_02.history['loss'], color='b', linestyle='-', label='Training
Data')
ax2.plot(epochs_list, history_02.history['val_loss'], color='r', linestyle='-',
label='Validation Data')
ax2.set_title('Training Loss', fontsize=12)
ax2.set_xlabel('Epochs', fontsize=12)
ax2.set_ylabel('Loss', fontsize=12)
ax2.legend(frameon=False, loc='upper center', ncol=2)
plt.savefig("training_frozencnn.jpeg", format='jpeg', dpi=100, bbox_inches='tight')
base_model = VGG19(include_top=False, input_shape=(240,240,3))
base_model_layer_names = [layer.name for layer in base_model.layers]
base_model_layer_names
x=base_model.output
flat = Flatten()(x)
class_1 = Dense(4608, activation = 'relu')(flat)
drop_out = Dropout(0.2)(class_1)
class_2 = Dense(1152, activation = 'relu')(drop_out)
output = Dense(2, activation = 'softmax')(class_2)
model_03 = Model(base_model.inputs, output)
model_03.load_weights('model_weights/vgg19_model_02.h5')
```

```
sgd = SGD(learning_rate=0.0001, decay = 1e-6, momentum = 0.9, nesterov = True)
model_03.compile(loss='categorical_crossentropy', optimizer = sgd, metrics=['accuracy'])
model_03.load_weights("model_weights/vgg_unfrozen.h5")
vgg_val_eval_03 = model_03.evaluate(valid_generator)
vgg_test_eval_03 = model_03.evaluate(test_generator)
```

SOURCE CODE:

```
import os
import numpy as np
from PIL import Image
import cv2
from flask import Flask, request, render_template
from werkzeug.utils import secure_filename
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Flatten, Dense, Dropout
from tensorflow.keras.applications.vgg19 import VGG19
base_model = VGG19(include_top=False, input_shape=(240,240,3))
x = base_model.output
flat=Flatten()(x)
class_1 = Dense(4608, activation='relu')(flat)
drop_out = Dropout(0.2)(class_1)
class_2 = Dense(1152, activation='relu')(drop_out)
output = Dense(2, activation='softmax')(class_2)
model_03 = Model(base_model.inputs, output)
model_03.load_weights('/Users/aashadapupallavi/Desktop/Advance_Brain_Tumor_Class
ification-main/BrainTumor Classification DL/vgg_unfrozen.h5')
app = Flask(__name__)
print('Model loaded. Check http://127.0.0.1:5000/')
def get_className(classNo):
if classNo==0:
return "No Brain Tumor"
elif classNo==1:
return "Yes Brain Tumor"
def getResult(img):
image=cv2.imread(img)
image = Image.fromarray(image, 'RGB')
```

```
image = image.resize((240, 240))
image=np.array(image)
input_img = np.expand_dims(image, axis=0)
result=model_03.predict(input_img)
result01=np.argmax(result,axis=1)
return result01
@app.route('/', methods=['GET'])
def index():
return render_template('index.html')
@app.route('/predict', methods=['GET', 'POST'])
def upload():
if request.method == 'POST':
f = request.files['file']
basepath = os.path.dirname(__file__)
file_path = os.path.join(
basepath, 'uploads', secure_filename(f.filename))
f.save(file_path)
value=getResult(file_path)
result=get_className(value)
return result
return None
if __name__ == '__main__':
app.run(debug=True)
```

6.Conclusion

Right now, there is no definitive cure for brain tumors, making early detection, accurate diagnosis, and timely intervention critical. The literature review highlights numerous efforts to identify features of brain tumors using advanced machine learning models and image analysis techniques. Despite these advancements, diagnosing brain tumors remains challenging due to the complexity of tumor types and variability in imaging data. Early-stage prediction plays a pivotal role in improving patient outcomes by enabling prompt treatment.Future research will focus on identifying novel features that enhance brain tumor detection and refining existing datasets by eliminating redundant or irrelevant attributes. This will help improve the precision and robustness of diagnostic models. Emerging trends in machine learning, particularly in analyzing diverse and multimodal datasets such as volumetric MRI and genomic data, offer promising avenues for early and accurate brain tumor detection.Machine learning methods have demonstrated superior performance compared to traditional diagnostic approaches, especially in handling complex data patterns. However, challenges persist, as clinical diagnoses often lack definitive corroboration due to ambiguous imaging results or limited test evidence. By addressing these limitations and leveraging advanced computational techniques, future systems can achieve more reliable, precise, and non-invasive brain tumor diagnoses.

7.Future Scope

While we couldn't achieve 100% accuracy in predicting brain tumors, the system we developed demonstrates significant potential to improve with additional data and advancements in technology. As with any evolving project, there is ample room for enhancement. One promising direction is integrating multiple machine learning algorithms as modular components, allowing their combined outputs to refine predictions and increase overall accuracy.The scope of this research can also expand to predictive modeling, examining the likelihood of tumor development in individuals currently at low risk based on genetic, lifestyle, and medical factors. Moreover, incorporating multi-modal data such as genomic sequences, functional imaging, and patient history could enhance diagnostic precision. The possibilities are vast, and with continued research and innovation, this system could pave the way for more effective and timely brain tumor detection and prognosis.

8.References and Bibliography

1) R. Rawat, M. Akram, Mithil, and S. Pradeep, *Brain Tumor Detection Using Machine Learning by Stacking Models*, 2020.

2) K. Dashtipour et al., *Detecting Brain Tumors Using Machine Learning Methods*, 2022, pp. 89–100.

3) Shikalgar, Arifa, and Sonavane, Shefali et al., *Hybrid Deep Learning Approach for Classifying Brain Tumors Based on Multimodal Data*, 2020.

4) Emmert-Streib, Z. Yang, H. Feng, S. Tripathi, and M. Dehmer, *An Introductory Review of Deep Learning for Prediction Models With Big Data*, Frontiers in Artificial Intelligence, vol. 3, 2020.

5) G. Murray, J. Šimša, A. Klimovic, and I. Indyk, *Tf.Data: A Machine Learning Data Processing Framework*, Proc. VLDB Endow., vol. 14, no. 12, pp. 2945–2958, July 2021.

6) Song T, et al., *Graph Convolutional Neural Networks for Brain Tumor Detection*, 2019 IEEE 16th Int Symp Biomed Imaging (ISBI 2019), no. ISBI, 2019, pp. 414–417.

7) Liu L, Zhao S, Chen H, and Wang A, *A New Machine Learning Method for Identifying Brain Tumors*, Simul Model Pract Theory, vol. 99, 2020, p. 102023.

8) Liu M, et al., *A Multi-Model Deep Convolutional Neural Network for Automatic Tumor Segmentation and Classification in Brain MRI*, Neuroimage, vol. 208, August 2020.

9) Impedovo D, Pirlo G, Vessio G, and Angelillo MT, *A Handwriting-Based Protocol for Assessing Neurodegenerative Conditions*, Cognit Comput, vol. 11, no. 4, 2019, pp. 576–586.

10) Parmar H, Nutter B, Long R, Antani S, and Mitra S, *Spatiotemporal Feature Extraction and Classification of Brain Tumors Using Deep Learning 3D-CNN for fMRI Data*, J Med Imaging, vol. 7, no. 05, 2020, pp. 1–14.

11) Basaia S, et al., *Automated Classification of Brain Tumors and Mild Cognitive Impairment Using a Single MRI and Deep Neural Networks*, Neuro Image Clin, vol. 21, 2019, p. 101645.