

# ENHANCING BRAIN TUMOR DIAGNOSIS:AI-DRIVEN SEGMENTATION AND CLASSIFICATION FOR ACCURATE DETECTION

*Minor project-II report submitted  
in partial fulfillment of the requirement for award of the degree of*

**Bachelor of Technology  
in  
Computer Science & Engineering**

**By**

<b>KATTA PALLAVI</b>	<b>(21UECS0285)</b>	<b>(19121)</b>
<b>GANGADASU BHARATH REDDY</b>	<b>(21UECS0351)</b>	<b>(19119)</b>
<b>JAMEDAR ASHFAQ AHMED</b>	<b>(21UECS0238)</b>	<b>(19080)</b>

*Under the guidance of  
DR. G. NALLASIVAN, ME, PhD.  
ASSISTANT PROFESSOR*



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING  
SCHOOL OF COMPUTING**

**VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF  
SCIENCE & TECHNOLOGY**

**(Deemed to be University Estd u/s 3 of UGC Act, 1956)**

**Accredited by NAAC with A++ Grade  
CHENNAI 600 062, TAMILNADU, INDIA**

# **ENHANCING BRAIN TUMOR DIAGNOSIS:AI-DRIVEN SEGMENTATION AND CLASSIFICATION FOR ACCURATE DETECTION**

*Minor project-II report submitted  
in partial fulfillment of the requirement for award of the degree of*

**Bachelor of Technology  
in  
Computer Science & Engineering  
By**

<b>KATTA PALLAVI</b>	<b>(21UECS0285)</b>	<b>(19121)</b>
<b>GANGADASU BHARATH REDDY</b>	<b>(21UECS0351)</b>	<b>(19119)</b>
<b>JAMEDAR ASHFAQ AHMED</b>	<b>(21UECS0238)</b>	<b>(19080)</b>

*Under the guidance of  
DR. G. NALLASIVAN, ME, PhD.  
ASSISTANT PROFESSOR*



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING  
SCHOOL OF COMPUTING**

**VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF  
SCIENCE & TECHNOLOGY**

**(Deemed to be University Estd u/s 3 of UGC Act, 1956)**

**Accredited by NAAC with A++ Grade  
CHENNAI 600 062, TAMILNADU, INDIA**

**May, 2024**

# CERTIFICATE

It is certified that the work contained in the project report titled ENHANCING BRAIN TUMOR DIAGNOSIS:AI-DRIVEN SEGMENTATION AND CLASSIFICATION FOR ACCURATE DETECTION by KATTA PALLAVI(21UECS0285) (19121), G BHARATH REDDY (21UECS0351)(19119), JAMEDAR ASHFAQ AHMED (21UECS0238) (19080) has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

**Signature of Supervisor**

**Computer Science & Engineering**

**School of Computing**

**Vel Tech Rangarajan Dr. Sagunthala R&D**

**Institute of Science & Technology**

**May, 2024**

**Signature of Professor In-charge**

**Computer Science & Engineering**

**School of Computing**

**Vel Tech Rangarajan Dr. Sagunthala R&D**

**Institute of Science & Technology**

**May, 2024**

# DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

KATTA PALLAVI

Date:        /        /

GANGADASU BHARATH REDDY

Date:        /        /

JAMEDAR ASHFAQ AHMED

Date:        /        /

# APPROVAL SHEET

This project report entitled “ENHANSING BRAIN TUMOR DIAGNOSIS : AI-DRIVEN SEGMENTATION AND CLASSIFICATION FOR ACCURATE DETECTION” by KATTA PALLAVI (21UECS0285), G BHARATH REDDY (21UECS0351), J ASHFAQ AHMED (21UECS0238) is approved for the degree of B.Tech in Computer Science & Engineering.

**Examiners**

**Supervisor**

Dr. G. NALLASIVAN,M.E,phD.

**Date:**        /        /

**Place:**

# ACKNOWLEDGEMENT

We express our deepest gratitude to our respected **Founder Chancellor and President Col. Prof. Dr. R. RANGARAJAN B.E. (EEE), B.E. (MECH), M.S (AUTO),D.Sc., Foundress President Dr. R. SAGUNTHALA RANGARAJAN M.B.B.S.** Chairperson Managing Trustee and Vice President.

We are very much grateful to our beloved **Vice Chancellor Prof. S. SALIVAHANAN**, for providing us with an environment to complete our project successfully.

We record indebtedness to our **Professor & Dean, Department of Computer Science & Engineering, School of Computing, Dr. V. SRINIVASA RAO, M.Tech., Ph.D.**, for immense care and encouragement towards us throughout the course of this project.

We are thankful to our **Head, Department of Computer Science & Engineering, Dr.M.S. MURALI DHAR, M.E., Ph.D.**, for providing immense support in all our endeavors.

We also take this opportunity to express a deep sense of gratitude to our **Internal Supervisor DR. G. NALLASIVAN, ME, PhD** for his cordial support, valuable information and guidance, he helped us in completing this project through various stages.

A special thanks to our **Project Coordinators Mr. V. ASHOK KUMAR, M.Tech., Ms. U.HEMAVATHI, M.E., Ms. C. SHYAMALA KUMARI, M.E.**, for their valuable guidance and support throughout the course of the project.

We thank our department faculty, supporting staff and friends for their help and guidance to complete this project.

**KATTA PALLAVI** (21UECS0285)

**GANGADASU BHARATH REDDY** (21UECS0351)

**JAMEDAR ASHFAQ AHMED** (21UECS0238)

## ABSTRACT

Enhancing Brain Tumor Segmentation and Classification Using Hybrid Architecture, represents a pivotal endeavor in the realm of medical imaging analysis. Leveraging a sophisticated hybrid architecture, the system integrates renowned deep learning models, AlexNet and DenseNet, for classification, and employs Random Forest for both classification and segmentation tasks. This innovative approach is driven by the necessity to address the limitations of existing methodologies, which often rely on manual interpretation and lack the efficiency required for the complex task of brain tumor analysis. AlexNet and DenseNet, chosen for their proven efficacy in handling intricate patterns within medical images, constitute the backbone of the system's classification capabilities. Their depth and hierarchical feature extraction mechanisms empower the system to discern subtle nuances indicative of brain tumor characteristics. The decision to employ ensemble learning, specifically Random Forest, for both classification and segmentation tasks is rooted in its versatility. Random Forest excels in aggregating the outputs of diverse decision trees, providing robust classification outcomes while simultaneously offering precise segmentation results. The adaptability of the system to diverse datasets is a fundamental consideration, given the inherent variability in medical imaging data. By integrating AlexNet and DenseNet, the system demonstrates flexibility in handling variations across imaging modalities and patient demographics. This adaptability is crucial for ensuring the system's applicability across a broad spectrum of neuroimaging scenarios, contributing to its reliability and generalizability in clinical settings. Reducing manual intervention is a paramount objective of the project. The hybrid architecture automates the brain tumor analysis process, mitigating subjectivity and potential errors associated with manual interpretation. The synergy between deep learning models and ensemble learning techniques enables the system to navigate complex data spaces, resulting in more accurate and consistent diagnoses.

### **Keywords:**

Brain Tumor, Brain MRI images, CNN, Deep Learning, Machine learning, Image segmentation, Accuracy, Image classification, Alexnet, Densenet, Transfer Learning, Random Forest

# LIST OF FIGURES

4.1	Architecture Diagram for Brain Tumor Diagnosis . . . . .	14
4.2	Data Flow Diagram for Brain Tumor Diagnosis . . . . .	15
4.3	Use Case Diagram for Brain Tumor Diagnosis . . . . .	16
4.4	Class Diagram for Brain Tumor Diagnosis . . . . .	17
4.5	Sequence Diagram for Brain Tumor Diagnosis . . . . .	18
4.6	Activity Diagram for Brain Tumor Diagnosis . . . . .	19
4.7	Managed Diagnosis Module . . . . .	22
4.8	Random Forest Algorithm Module . . . . .	23
4.9	Training and Testing Module . . . . .	24
5.1	Brain Tumor Diagnosis Pre-processing Program . . . . .	26
5.2	Knowing the Tumor Type . . . . .	27
5.3	Knowing the Tumor . . . . .	28
5.4	Tumor Type . . . . .	29
5.5	Knowing the Tumor Type . . . . .	30
5.6	Test Image . . . . .	31
5.7	Test Image . . . . .	32
6.1	Metrics of Algorithm . . . . .	37
8.1	Plagiarism Report . . . . .	40
9.1	Poster Presentation . . . . .	50



# LIST OF ACRONYMS AND ABBREVIATIONS

<b>ABBREVIATION</b>	<b>DEFINITION</b>
AI	Artificial Intelligence
CNN	Convolutional Neural Network
DL	Deep Learning
GPU	Graphics Processing Unit
NIC	Network Interface Card
RF	Random Forest

# TABLE OF CONTENTS

	Page.No
<b>ABSTRACT</b>	<b>v</b>
<b>LIST OF FIGURES</b>	<b>vi</b>
<b>LIST OF ACRONYMS AND ABBREVIATIONS</b>	<b>vii</b>
<b>1 INTRODUCTION</b>	<b>1</b>
1.1 Introduction . . . . .	1
1.2 Aim of the project . . . . .	1
1.3 Project Domain . . . . .	2
1.4 Scope of the Project . . . . .	2
<b>2 LITERATURE REVIEW</b>	<b>4</b>
<b>3 PROJECT DESCRIPTION</b>	<b>8</b>
3.1 Existing System . . . . .	8
3.2 Proposed System . . . . .	8
3.3 Feasibility Study . . . . .	9
3.3.1 Economic Feasibility . . . . .	9
3.3.2 Technical Feasibility . . . . .	9
3.3.3 Social Feasibility . . . . .	10
3.4 System Specification . . . . .	11
3.4.1 Hardware Specification . . . . .	11
3.4.2 Software Specification . . . . .	11
3.4.3 Standards and Policies . . . . .	12
<b>4 METHODOLOGY</b>	<b>14</b>
4.1 Architecture Diagram for Brain Tumor Diagnosis . . . . .	14
4.2 Design Phase . . . . .	15
4.2.1 Data Flow Diagram for Brain Tumor Diagnosis . . . . .	15
4.2.2 Use Case Diagram for Brain Tumor Diagnosis . . . . .	16
4.2.3 Class Diagram for Brain Tumor Diagnosis . . . . .	17

4.2.4	Sequence Diagram for Brain Tumor Diagnosis . . . . .	18
4.2.5	Activity Diagram for Brain Tumor Diagnosis . . . . .	19
4.3	Algorithm & Pseudo Code . . . . .	20
4.3.1	Brain Tumor Diagnosis Algorithm . . . . .	20
4.3.2	Pseudo Code . . . . .	20
4.4	Module Description . . . . .	22
4.4.1	Managed Diagnosis Module . . . . .	22
4.4.2	Random Forest Algorithm Module . . . . .	23
4.4.3	Training and Testing Module . . . . .	24
4.5	Steps to execute/run/implement the project . . . . .	25
4.5.1	Execute . . . . .	25
4.5.2	Run . . . . .	25
4.5.3	Implement the Project . . . . .	25
<b>5</b>	<b>IMPLEMENTATION AND TESTING</b>	<b>26</b>
5.1	Input and Output . . . . .	26
5.1.1	Brain Tumor Diagnosis Pre-processing Program . . . . .	26
5.1.2	Knowing the Tumor Type . . . . .	27
5.2	Testing . . . . .	27
5.3	Types of Testing . . . . .	28
5.3.1	Testing the Tumor . . . . .	28
5.3.2	Integration testing . . . . .	29
5.3.3	System testing . . . . .	30
5.3.4	Test Result . . . . .	32
<b>6</b>	<b>RESULTS AND DISCUSSIONS</b>	<b>33</b>
6.1	Efficiency of the Proposed System . . . . .	33
6.2	Comparison of Existing and Proposed System . . . . .	33
6.3	Proposed system: . . . . .	34
6.4	Sample Code . . . . .	35
<b>7</b>	<b>CONCLUSION AND FUTURE ENHANCEMENTS</b>	<b>38</b>
7.1	Conclusion . . . . .	38
7.2	Future Enhancements . . . . .	38
<b>8</b>	<b>PLAGIARISM REPORT</b>	<b>40</b>

<b>9</b>	<b>SOURCE CODE &amp; POSTER PRESENTATION</b>	<b>41</b>
9.1	Source Code . . . . .	41
9.2	Poster Presentation . . . . .	50
	<b>References</b>	<b>51</b>

# Chapter 1

## INTRODUCTION

### 1.1 Introduction

Embarking on the frontier of medical innovation, our project delves into the intricate world of Enhancing Brain Tumor Segmentation and Classification through the implementation of a sophisticated hybrid architecture. In this groundbreaking initiative, we seamlessly integrate the formidable capabilities of AlexNet and DenseNet, two state-of-the-art convolutional neural networks meticulously crafted for classification tasks. Elevating the sophistication, we introduce Random Forest, an ensemble learning algorithm not only adept at classification but also uniquely applied to segmentation tasks. This orchestrated fusion of cutting-edge technologies represents a quantum leap in the landscape of brain tumor analysis. Our endeavor aims to redefine the benchmarks of accuracy and efficiency in identifying and delineating brain tumors, offering a revolutionary tool for healthcare professionals. As we embark on this transformative journey, the overarching goal is to reshape diagnostic methodologies, pushing the boundaries of precision and ushering in a new era in the understanding and treatment of neurological disorders.

### 1.2 Aim of the project

The aim of our project, "Enhancing Brain Tumor Segmentation and Classification Using Hybrid Architecture," is to revolutionize the landscape of medical imaging by developing a sophisticated computational framework. Through the fusion of advanced neural network architectures AlexNet and DenseNet for classification and the versatile application of Random Forest for both classification and segmentation, we aspire to significantly elevate the accuracy and efficiency of brain tumor analysis. The primary goal is to create a robust tool that enhances the diagnostic capabilities of healthcare professionals, enabling more precise identification, classification, and delineation of brain tumors. By pushing the boundaries of current methodologies, we aim to contribute to the broader field of medical image analysis, fostering break-

throughs in neurological disorder diagnostics and treatment strategies. This project's overarching ambition extends beyond technological innovation. It seeks to bridge the gap between cutting-edge computational approaches and clinical practice, ultimately improving patient outcomes and solidifying its place at the forefront of advancements in medical diagnostics.

### **1.3 Project Domain**

Operating within the expansive domain of medical image analysis, our project concentrates on the intricate realm of brain tumor segmentation and classification. At the intersection of computer vision, machine learning, and healthcare, we employ advanced neural network architectures such as AlexNet and DenseNet, coupled with the versatile Random Forest algorithm. This amalgamation places our project on the cutting edge of computational methodologies in the medical field. The primary objective is to elevate the precision of analyzing medical imaging data related to brain tumors, striving for heightened diagnostic accuracy and more nuanced treatment strategies. As we navigate this interdisciplinary domain, our project aims to contribute substantively to the broader landscape of medical image analysis, leveraging technology to advance our comprehension and management of neurological disorders. In essence, our project operates as a bridge between innovative computational approaches and the evolving demands of healthcare, harnessing the potential to reshape diagnostic practices and treatment protocols within the complex and critical field of brain tumor analysis.

### **1.4 Scope of the Project**

The scope is expansive, encompassing the convergence of cutting-edge technologies to refine brain tumor analysis in medical imaging. By integrating advanced neural network architectures, including AlexNet and DenseNet, for classification and deploying the versatile Random Forest algorithm for both classification and segmentation, our project aims to broaden the horizons of diagnostic accuracy and efficiency. Beyond merely identifying and classifying brain tumors, the scope extends to enhancing the precision of tumor boundary delineation, crucial for formulating effective treatment strategies. The comprehensive scope of this project is not confined solely to technological advancements. It spans the translation of computational

methodologies into practical applications within clinical settings. By pushing the boundaries of current diagnostic practices, we aspire to contribute significantly to the evolving field of medical image analysis, fostering breakthroughs that hold the potential to reshape the landscape of neurological disorder diagnostics and treatment protocols. This project thus emerges as a pivotal force in bridging the gap between innovative computational approaches and tangible improvements in patient care

## Chapter 2

# LITERATURE REVIEW

[1] Abdulaziz Salamah Aljaloud et al.,(2022), conducted a comprehensive literature review on early diagnosis of brain tumors using deep learning and machine learning techniques. Focused on detecting Alzheimer's disease using deep learning algorithms. The author worked on image category recognition using completed local ternary patterns. Developed a system for breast cancer detection using deep learning. Analyzed Gabor-wavelet based features for brain tumor classification. He worked on MR brain tumor classification and segmentation using wavelets. These studies highlight the use of various techniques for medical image analysis and classification, contributing to the early detection and diagnosis of brain tumors, 2022.

[2] Angela Zhang et al., (2020), conducted a comprehensive literature review on brain tumor segmentation and overall survival prediction. The author conduct a literature review to provide a comprehensive overview of the existing research in this field. Also discuss the use of deep neural networks, suchas U-Net and DeepMedic, for brain tumor segmentation and highlight the importance of incorporating location information through brain parcellation atlases. The author also explore the use of tractographic features extracted from structural MR images to predict overall survival. It mention the significance of the BraTS 2018 dataset, which provides labeled MR images and ground-truth lesionsfor training and validation. The study aims to improve the accuracy of survival prediction and provide valuable insights for personalized treatment planning in brain tumor patients, 2020.

[3] Ayesha Jabbar et al., (2019), conducted a comprehensive literature review on a hybrid Caps-VGGNet model for brain tumor detection and classification. Several machine and deep learning-based classification and anomaly detection techniques have been applied to biological images. The model integrates the CapsNet model with the VGGNet model by adding the layers of VGGNet. The proposed model achieves high accuracy, specificity, and sensitivity on the Brats20 dataset compared



to other conventional and hybrid models. Previous research has also used CNN models for tumor detection and classification, but they require large datasets. The proposed model addresses this challenge by automatically extracting and classifying features. Future research should evaluate the model on different types of data and with a larger dataset to ensure its robustness and effectiveness, 2019.

[4] Dinthisrang Daimary et al., (2019), conducted a comprehensive literature review on a hybrid CapsVGGNet model for brain tumor detection and classification. Several machine and deep learningbased classification and anomaly detection techniques have been applied to biological images. The model integrates the CapsNet model with the VGGNet model by adding the layers of VGGNet. The proposed model achieves high accuracy, specificity, and sensitivity on the Brats20 dataset compared to other conventional and hybrid models. Previous research has also used CNN models for tumor detection and classification, but they require large datasets. The proposed model addresses this challenge by automatically extracting and classifying features. Future research should evaluate the model on different types of data and with a larger dataset to ensure its robustness and effectiveness, 2019.

[5] Feyza Altunbey Ozbay et al., (2023), conducted a comprehensive literature review on the detection of brain tumors using deep learning techniques. The authors discuss various studies that have been conducted in this field, highlighting the different methods and models proposed for brain tumor classification. The review includes studies that have utilized convolutional neural networks (CNNs) such as VGG16, CapsNet, and ResNet, as well as other techniques like segmentation and feature fusion. The accuracy rates achieved by these methods range from 84 to 98. The author also compare their proposed hybrid model, which combines features extracted from different CNN architectures using mRMR and Grad-CAM techniques. The results show that their model outperforms other state-of-the-art methods, achieving an accuracy rate of 99.1 without Grad-CAM and 99.6 with Grad-CAM. Overall, the literature review demonstrates the advancements in brain tumor detection using deep learning and highlights the effectiveness of the proposed hybrid model in accurately classifying brain tumors, 2023.

[6] Mohammed Rasool et al., (2022), conducted a comprehensive literature review on brain tumor classification using MRI images. The authors discuss various

methods and techniques used in previous studies for brain tumor diagnosis and classification. Compare their proposed hybrid deep learning model with other existing methods in terms of accuracy and computational analysis. The review highlights the importance of early detection of brain tumors and the role of machine learning algorithms in improving diagnosis and treatment outcomes. The authors mention the use of features such as Gray Level Co-occurrence Matrix (GLCM), Support Vector Machine (SVM), and Convolutional Neural Networks (CNN) in different studies. Also discuss the advantages of their proposed model, which combines a pre-trained Google-Net model with SVM for feature extraction and pattern classification. The model achieved an accuracy of 98.1 in classifying brain tumors, surpassing other methods in the comparison. Overall, the literature review provides an overview of the current state of research in brain tumor classification and highlights the potential of deep learning models in improving diagnosis accuracy, 2022.

[7] Nagwa M. Aboelenein et al.,(2020), conducted a comprehensive literature review on the topic of brain tumor segmentation using deep learning techniques. The paper highlights the importance of accurate tumor segmentation in medical diagnosis and treatment planning. It discusses the challenges associated with manual segmentation and the need for automatic or semi-automatic methods. The review covers various deep learning-based approaches proposed in the literature for brain tumor segmentation. It mentions the use of fully convolutional neural networks (FCN), conditional random fields (CRF), and multiscale CNNs as effective segmentation methods. The U-Net architecture is also discussed as a widely used framework for brain tumor segmentation. The paper introduces the proposed HTTU-Net architecture, which addresses the challenges of brain tumor segmentation, such as tumor variability and low contrast to surrounding tissue. The architecture consists of two tracks, focusing on tumor form and size, and capturing contextual information. The use of a hybrid loss function combining focal loss and generalized dice loss is also mentioned to mitigate class imbalance. Overall, the literature review provides an overview of the current state-of-the-art in brain tumor segmentation using deep learning techniques and sets the stage for the proposed HTTU-Net architecture, 2020.

[8] Shakir Khan et al.,(2023), conducted a comprehensive literature review on U-Net based Brain Tumor Segmentation: A Comprehensive Review” provides a comprehensive overview of the advancements and innovations in U-Net architecture

for brain tumor segmentation from MRIs. The authors highlight the ongoing potential of U-Net in improving the performance of brain tumor segmentation and discuss recent trends in this field. They also provide a quantitative comparison of different U-Net architectures and experiment with four U-Net models on the BraTS 2020 dataset to evaluate their performance. The paper concludes by analyzing the limitations and challenges of medical image analysis and emphasizing the importance of developing new architectures for optimization. Overall, the paper serves as a valuable resource for researchers and practitioners working in the field of medical image segmentation, 2023.

[9] T. Balamurugan et al.,(2022), conducted a comprehensive literature review on brain tumor segmentation and classification using deep learning techniques. The study explores the application of a hybrid deep convolutional neural network (CNN) with a LuNet classifier for accurate identification and classification of brain tumors. The authors discuss the use of various machine learning algorithms such as random forest, K-nearest neighbor, decision tree, and deep learning models like CNN and LuNet. They highlight the advantages and limitations of these techniques in terms of performance, computational complexity, and dataset size. The review also mentions the use of image preprocessing techniques, feature extraction methods, and performance evaluation metrics in brain tumor analysis. Overall, the literature review provides a comprehensive overview of the existing research in the field and sets the foundation for the proposed hybrid CNN-LuNet model, 2022.

[10] Zengxin Liu et al.,(2023), conducted a comprehensive literature review on TransMVU: Multi-view 2D U-Nets with transformer for brain tumour segmentation discusses the use of attention in feedforward CNN models for medical imaging tasks. The paper mentions the USE-Net, which incorporates Squeeze-and-Excitation (SE) blocks into the U-Net architecture and demonstrates strong performance on prostate MRI datasets. While CNN-based methods have shown good detail, the paper segmentation in MRI images. The author propose a multi-view approach that leverages the advantages of both the transformer and 2D U-Net. Experimental results on the BraTS20 dataset demonstrate that TransMVU outperforms state-of-the-art 2D models and classic 3D models. This research direction shows promise for increasing the accuracy of tumor segmentation while maintaining computational efficiency, 2023.

## Chapter 3

# PROJECT DESCRIPTION

### 3.1 Existing System

The existing system for brain tumor segmentation and classification faces challenges in achieving optimal precision and efficiency. Traditional methods often rely on manual interpretation of medical images, introducing subjectivity and time-consuming processes. Furthermore, these approaches may struggle to discern intricate patterns and subtle features crucial for accurate diagnosis. The limitations become more pronounced when dealing with complex structures like brain tumors that demand a sophisticated understanding of diverse image characteristics. In response to these challenges, our project proposes a transformative shift by introducing a hybrid architecture. This innovative approach integrates advanced neural network models, AlexNet and DenseNet, tailored for classification, and leverages the adaptability of Random Forest for both classification and segmentation tasks.

By doing so, we aim to overcome the shortcomings of the existing system, offering a more automated, accurate, and efficient solution for brain tumor analysis in medical imaging. This shift not only addresses current limitations but also signifies a substantial step toward enhancing the overall efficacy and reliability of diagnostic processes in neuroimaging.

### 3.2 Proposed System

The proposed system introduces a revolutionary approach to brain tumor segmentation and classification, addressing limitations inherent in current methodologies. By integrating advanced neural network models, namely AlexNet and DenseNet, specifically designed for classification tasks, and employing the versatile Random Forest algorithm for both segmentation and classification, our system aims to redefine the landscape of medical imaging analysis. This innovative fusion enhances the precision and efficiency of brain tumor analysis, surpassing the constraints of man-

ual interpretation and automating the process with sophisticated machine learning techniques. With this proposed hybrid architecture, we anticipate a transformative impact on diagnostic accuracy and overall efficacy in neuroimaging.

This system not only represents a technological leap but also signifies a crucial step towards providing healthcare professionals with a powerful, adaptive tool, poised to meet the evolving challenges of neurodiagnostics in modern healthcare.

### **3.3 Feasibility Study**

#### **3.3.1 Economic Feasibility**

From an economic perspective, the proposed system demonstrates promising feasibility. While initial implementation costs may include acquiring and configuring the necessary hardware and software infrastructure, the long-term benefits are substantial. The automation and efficiency gains achieved through the hybrid architecture, integrating AlexNet, DenseNet, and Random Forest, contribute to potential cost savings by streamlining the diagnostic process. Moreover, the enhanced accuracy in brain tumor analysis may result in earlier and more precise diagnoses, potentially reducing treatment costs and improving patient outcomes.

The proposed system's adaptability to evolving medical imaging standards positions it as a valuable long-term investment. Its impact on healthcare economics extends beyond immediate financial considerations, aligning with a broader trend toward leveraging advanced technologies for more effective and economical patient care. In conclusion, the economic feasibility of this project is underscored by its potential to not only optimize diagnostic processes but also contribute to more cost-effective and improved healthcare delivery.

#### **3.3.2 Technical Feasibility**

The technical feasibility of the proposed system is robust, considering the advancements in neural network architectures and ensemble learning algorithms. Integration of state-of-the-art models like AlexNet and DenseNet, along with the implementation of the versatile Random Forest algorithm, aligns with current trends in machine learning and computational capabilities. The project leverages widely adopted frameworks and tools in the field of deep learning, ensuring compatibility and ease of integration. The adaptability of the proposed hybrid

architecture to diverse medical imaging datasets enhances its technical viability, allowing for potential scalability and applicability across various healthcare settings. Furthermore, continuous advancements in hardware technology provide ample resources to support the computational requirements of the proposed system. With the technical landscape evolving rapidly, the proposed system is well-positioned to harness emerging technologies, ensuring its sustainability and relevance in the ever-changing field of medical image analysis.

Overall, the technical feasibility of the project is underpinned by the utilization of cutting-edge methodologies and the capacity to seamlessly integrate with existing technological infrastructures. The technical feasibility of our project, integrating advanced neural network models and ensemble learning, is underpinned by cutting-edge methodologies, ensuring adaptability to evolving technologies and seamless integration with existing infrastructures

### **3.3.3 Social Feasibility**

From a social perspective, the proposed brain tumor analysis system showcases compelling feasibility. Its potential to enhance diagnostic accuracy and streamline medical imaging processes aligns with societal expectations for advanced healthcare solutions. The system's automation features contribute to reducing the burden on healthcare professionals, allowing them to focus more on patient care. Moreover, the improved precision in brain tumor identification and classification directly translates to enhanced patient outcomes. Quicker and more accurate diagnoses facilitate timely interventions, potentially improving survival rates and quality of life for individuals facing neurological disorders.

The societal impact extends to increased accessibility to advanced diagnostic tools, promoting a more equitable distribution of healthcare resources. By addressing critical healthcare challenges, the proposed system aligns with societal expectations for innovative solutions that not only improve efficiency but also contribute to better overall healthcare outcomes and accessibility. This social feasibility positions the project as a valuable asset in advancing healthcare standards and promoting a more inclusive and effective healthcare system

### 3.4 System Specification

#### 3.4.1 Hardware Specification

- **Central Processing Unit (CPU):** A multicore processor with sufficient computational power to handle complex deep learning tasks.
- **Graphics Processing Unit (GPU):** A high-performance GPU, preferably with CUDA support, to accelerate the training and inference processes of neural network models like AlexNet and DenseNet.
- **Memory (RAM):** A substantial amount of RAM to accommodate the large datasets and facilitate efficient model training and processing.
- **Storage:** A high-capacity, fast storage solution (SSD) for storing large medical imaging datasets and model parameters.
- **Network Interface Card (NIC):** A high-speed NIC for seamless data transfer, especially when dealing with extensive medical image datasets.
- **Display:** A high-resolution monitor for visualizing and interpreting medical images during system development and testing.
- **Miscellaneous:** Additional components such as a power supply unit, cooling system, and other peripherals as required for system stability and functionality.

#### 3.4.2 Software Specification

- **Operating System:** A compatible operating system, such as Linux or Windows, providing a stable environment for system deployment and development.
- **Deep Learning Frameworks:** Installation of popular deep learning frameworks like TensorFlow or PyTorch to facilitate the implementation and training of neural network models (AlexNet, DenseNet).
- **Programming Language:** Use of a programming language like Python for system development, leveraging its extensive libraries and frameworks for machine learning.
- **Development Environment:** An integrated development environment (IDE) such as Jupyter Notebooks or PyCharm for coding, testing, and debugging.
- **Ensemble Learning Libraries:** Integration of ensemble learning libraries, such as Scikit-Learn, for implementing the Random Forest algorithm for both classification and segmentation.

- **Medical Imaging Libraries:** Utilization of medical imaging libraries like DICOM or OpenCV for handling and processing medical image data.
- **Version Control:** Implementation of version control tools like Git for collaborative development and tracking changes in the codebase.
- **Documentation Tools:** Adoption of documentation tools like Sphinx or Doxygen to ensure comprehensive and accessible system documentation

### 3.4.3 Standards and Policies

#### **Standards:**

**HIPAA (Health Insurance Portability and Accountability Act):** The system upholds HIPAA standards to safeguard patient data confidentiality and privacy during the segmentation and classification processes. **DICOM (Digital Imaging and Communications in Medicine):** Compliance with DICOM standards ensures the seamless integration and compatibility of medical imaging data, promoting interoperability across different healthcare systems. **Data Security Standards:** The system follows industry-standard encryption protocols, access controls, and secure data transmission practices to protect sensitive medical information from unauthorized access or breaches. **Ethical Guidelines:** Adherence to ethical guidelines governing medical research and artificial intelligence ensures responsible and transparent use of technology in the analysis of brain tumor data. **Informed Consent Policies:** The project complies with informed consent policies, obtaining explicit permission from patients for the utilization of their medical data in the brain tumor segmentation and classification processes.

#### **Policies:**

**Data Privacy Policy:** The system strictly adheres to a comprehensive data privacy policy, outlining measures to protect patient confidentiality and ensure responsible handling of medical imaging data during segmentation and classification. **Security Policy:** A robust security policy is implemented, encompassing encryption protocols, access controls, and secure data transmission to safeguard against unauthorized access and maintain the integrity of medical information. **Ethical Use Policy:** The system operates under a strict ethical use policy, delineating guidelines for the responsible and transparent application of artificial intelligence in brain tumor analysis, prioritizing patient well-being and fairness. **Informed Consent Policy:** In accordance with ethical standards, the project follows a stringent informed consent policy, en-



sureing that patients provide explicit consent for the utilization of their medical data in the segmentation and classification processes. Quality Assurance Policy: A dedicated quality assurance policy is in place to uphold the accuracy and reliability of the brain tumor analysis results, encompassing regular testing, validation, and adherence to performance benchmarks. Anaconda Prompt Anaconda prompt is a type of command line interface which explicitly deals with the ML( MachineLearning) modules.And navigator is available in all the Windows,Linux and MacOS.The anaconda prompt has many number of IDE's which make the coding easier. The UI can also be implemented in 11 python.

**Standard Used: ISO/IEC 27001**

### **Jupyter**

It's like an open source web application that allows us to share and create the documents which contains the live code, equations, visualizations and narrative text. It can be used for data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning.

**Standard Used: ISO/IEC 27001**

## Chapter 4

# METHODOLOGY

### 4.1 Architecture Diagram for Brain Tumor Diagnosis

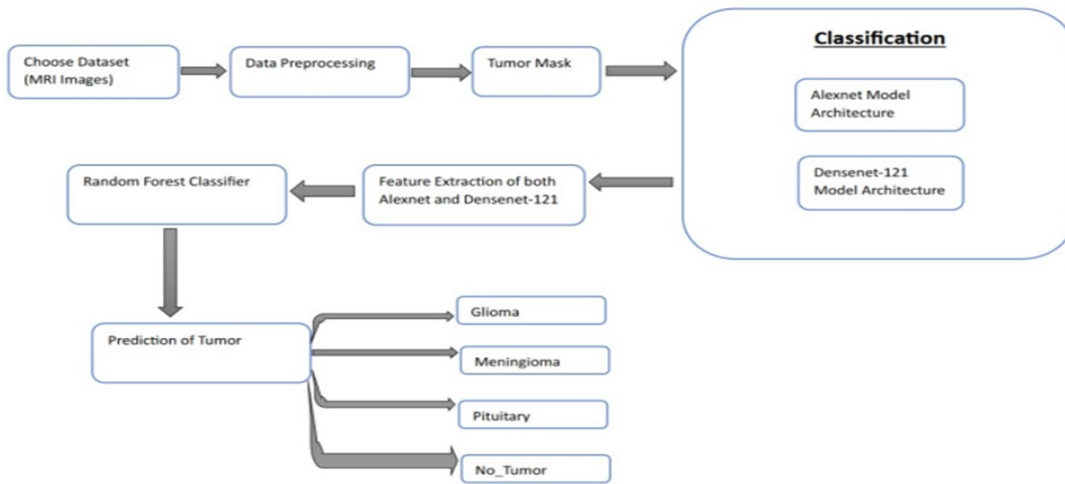


Figure 4.1: Architecture Diagram for Brain Tumor Diagnosis

The above figure 4.1 describes about the flowchart outlines of a system that utilizes machine learning and MRI images to classify brain tumors. This process progresses through several key steps:

The first step where the dataset containing brain MRI images is selected. This dataset should be diverse and representative of the different types of brain tumors that the model will be trained to classify. It involves artificially increasing the dataset size by creating variations of the existing images. This can help to improve the model's generalizability and prevent overfitting. Some common data augmentation techniques include rotation, flipping, and adding noise. In this step, a mask of the tumor region is created for each image in the dataset. This can be done manually by a medical expert or automatically using image segmentation techniques. It involves extracting features from the preprocessed images. These features are mathematical representations of the image data that can be used by the machine learning models to classify the tumors.

## 4.2 Design Phase

### 4.2.1 Data Flow Diagram for Brain Tumor Diagnosis

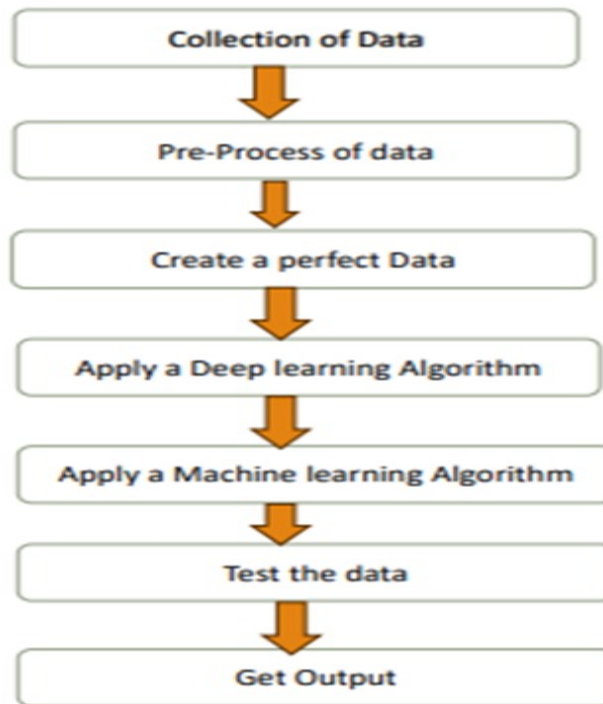


Figure 4.2: Data Flow Diagram for Brain Tumor Diagnosis

The above figure 4.2 describes the flow of process which involve Gather MRI scans of healthy brains and brains with different tumors (glioma, meningioma, pituitary adenoma). Prepare the scans for analysis by resizing, rotating, and formatting them consistently. Preprocess again (may differ slightly from step 1 preprocessing). Use AlexNet and DenseNet to extract high-level features like tumor presence and type from the scans. Preprocess differently to suit Random Forests (e.g., intensity normalization). Engineer specific features for brain tumors and segmentation, like texture measures or intensity gradients. Merge the extracted features from both paths (or feed them into separate branches). Use a Random Forest classifier to analyze the combined features and predict the tumor type. Train a separate Random Forest model on labeled data (with tumor boundaries marked) for segmentation. This model would generate a mask outlining the exact tumor region within the brain. Train and test each model (Random Forest classifiers and segmenter) on separate data sets. Evaluate their performance using metrics like accuracy, sensitivity, and specificity.

#### 4.2.2 Use Case Diagram for Brain Tumor Diagnosis

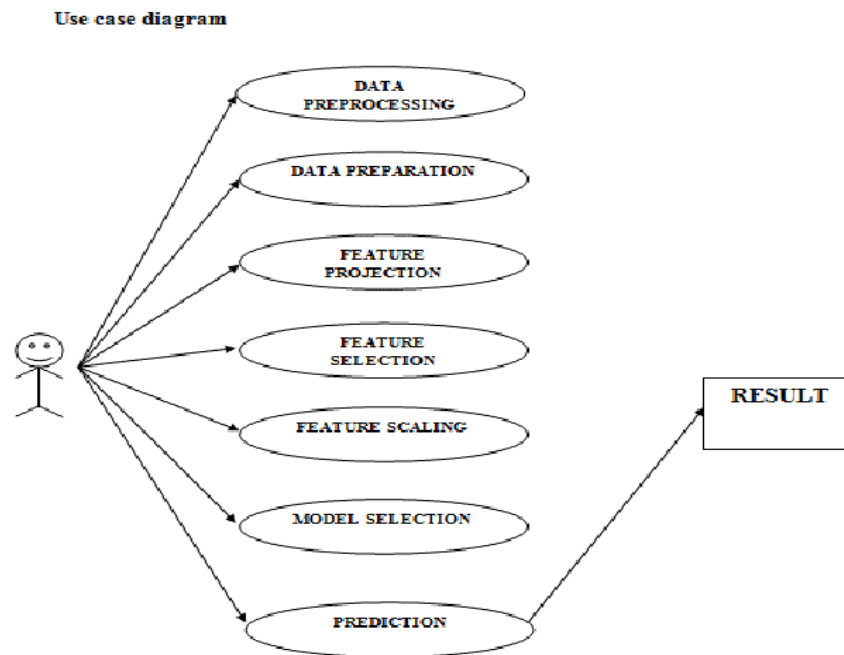


Figure 4.3: Use Case Diagram for Brain Tumor Diagnosis

The above figure 4.3 depicts a typical machine learning workflow, outlining the steps involved in training and deploying an ML model. It starts with data being imported, either directly or by pasting it. This data undergoes preprocessing, which may involve cleaning, transforming, and formatting it to ensure its suitability for the chosen ML algorithm. Next, the preprocessed data is divided into two sections: training data and test data. The training data, the larger portion, is used to train the ML algorithm, while the test data, the smaller portion, is used to evaluate the model's performance after training. The ML algorithm is then applied to the training data. This involves selecting an appropriate algorithm, setting its parameters, and training it on the data. As the algorithm processes the training data, it learns patterns and relationships within the data. Once trained, the model is applied to the test data to assess its performance. Metrics like accuracy, precision, and recall are used to evaluate how well the model performs. If the model's performance meets expectations, it can be deployed. This involves integrating it into a real-world application where it can make predictions on new, unseen data. Overall, the figure summarizes the essential steps in building and deploying an ML model, highlighting the importance of data preparation, training, evaluation, and deployment.

### 4.2.3 Class Diagram for Brain Tumor Diagnosis

is a MR slice of patient's head.

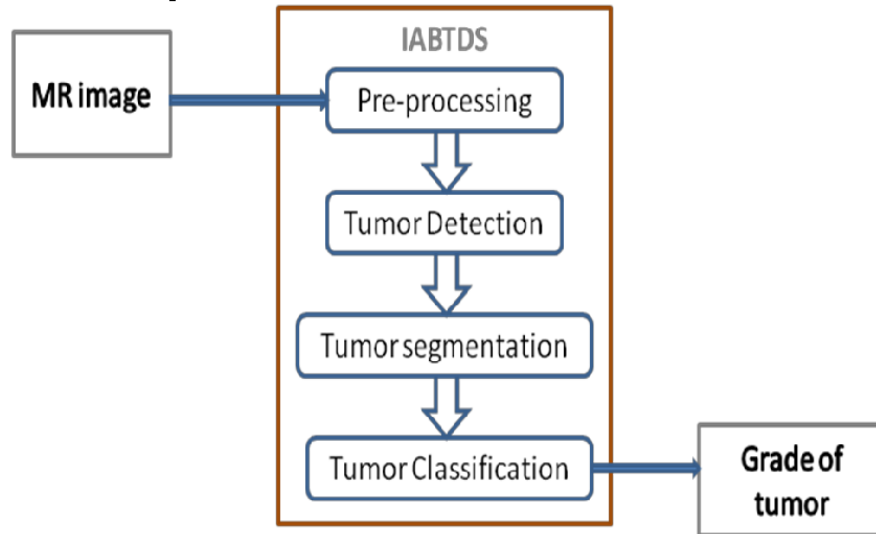


Fig. 1. Block diagram of proposed method

Figure 4.4: Class Diagram for Brain Tumor Diagnosis

The above figure 4.4 shows a flowchart of the process of creating and evaluating a mask. The process starts with loading a dataset of images and their associated labels. The dataset is then augmented to increase its size and diversity. Next, a tumor mask is generated for each image in the dataset. The tumor mask is a binary image that indicates the presence or absence of a tumor in the corresponding image. The next step is to extract features from the images and masks. Features are characteristics of the images and masks that can be used to distinguish between healthy and tumor-bearing tissue. The final step is to deploy the trained model. The model can be deployed in a variety of ways, such as a web service, a mobile app, or a medical imaging device. The figure shows the key steps involved in the process of creating and evaluating a mask. The process is iterative and can be repeated to improve the performance of the model.

#### 4.2.4 Sequence Diagram for Brain Tumor Diagnosis

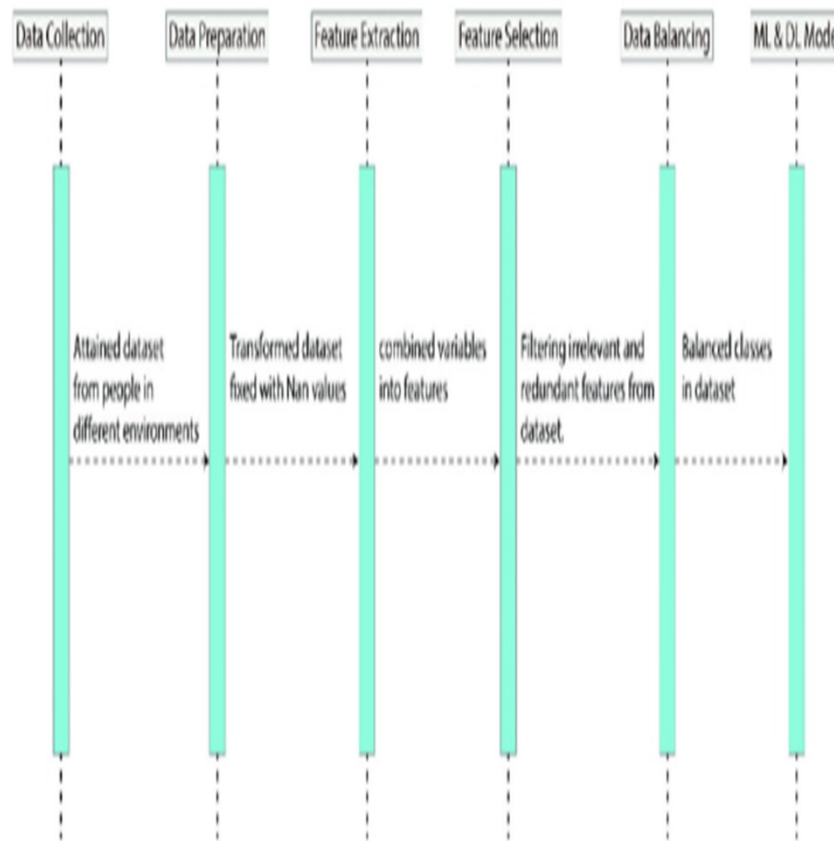


Figure 4.5: Sequence Diagram for Brain Tumor Diagnosis

The above figure 4.5 describes the process of working in a systematic way. It involves data is collected from people in different environments. This could include things like their age, gender, health history, and lifestyle factors. Once the data is collected, it needs to be prepared for analysis. This could involve cleaning the data, removing errors, and formatting it correctly. After the data is prepared, it's transformed into numerical values. This is done so that the computer can understand the data and use it to make predictions. Once the data is transformed, it's ready for feature extraction. This step involves identifying the most important features in the data that are relevant to the task at hand. After the features are extracted, they're then selected. This means that some features may be removed if they're not relevant or if they're redundant. Finally, the data is balanced. This means that the classes in the data are made to have roughly the same number of observations. This is important so that the machine learning model doesn't become biased towards one class or another.

#### 4.2.5 Activity Diagram for Brain Tumor Diagnosis

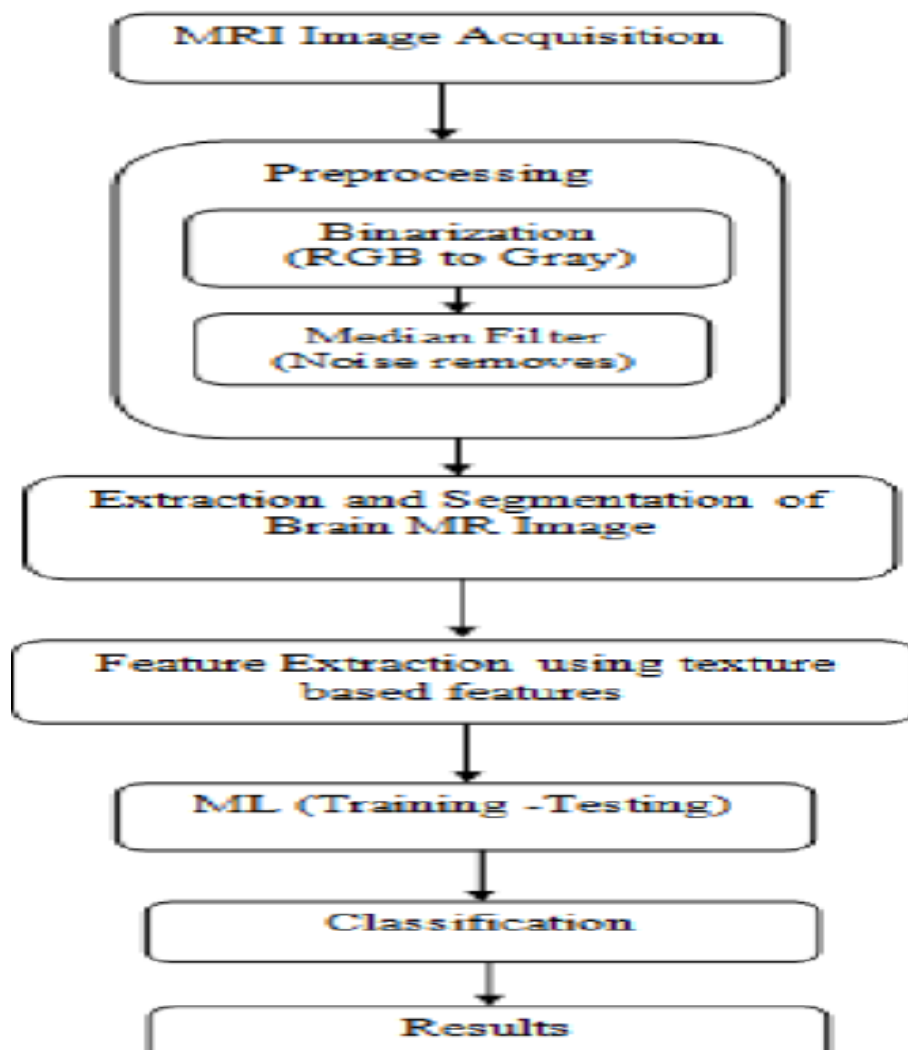


Figure 4.6: Activity Diagram for Brain Tumor Diagnosis

The above figure 4.6 describes the process of extracting and segmenting brain MR images. It's likely part of a medical image analysis application, like one used in tumor detection or anatomical studies. The process starts with MRI image acquisition, where the images are obtained using a magnetic resonance imaging scanner. Next, preprocessing takes place, which involves steps like converting the image from RGB to grayscale and applying a median filter to remove noise. This helps prepare the image for further analysis. The diagram indicates training and testing phases, suggesting the use of a supervised learning approach. Here, a machine learning model is trained on labeled data where each image has its regions identified. During training, the model learns the relationship between image features and the corresponding tissue labels.

## 4.3 Algorithm & Pseudo Code

### 4.3.1 Brain Tumor Diagnosis Algorithm

Step 1: Mount Google Drive using the Google Colab library.

Step 2: Load and resize images using an image processing library (e.g., OpenCV or PIL), and save the resized images to a new directory.

Step 3: Implement tumor mask segmentation using image processing techniques.

Step 4: Refine segmented tumor masks through post-processing techniques.

Step 5: Extract relevant features from the tumor masks (e.g., perimeter, area, circularity, GLCM features).

Step 6: Set up a Random Forest Classifier and a Multi-Layer Perceptron (Neural Network).

Step 7: Split the data into training and testing sets, and train both the Random Forest and Neural Network models.

Step 8: Evaluate the accuracy of both models using the testing set.

Step 9: Optionally, perform hyperparameter tuning for the Random Forest model.

Step 10: Load and preprocess new images, extract features, and use the trained models to predict tumor types.

Step 11: Visualize segmentation and classification results and perform validation if needed. Step 12: Optionally, predict the tumor type on a single new image using both the Random Forest and Neural Network models.

Step 13: The algorithm is complete.

### 4.3.2 Pseudo Code

```
1  import numpy as np
2  import tensorflow as tf
3  from sklearn.model_selection import train_test_split
4  from sklearn.preprocessing import StandardScaler
5  from sklearn.svm import SVC
6  from sklearn.ensemble import RandomForestClassifier
7  from skimage import io, color, filters
8  from skimage.transform import resize
9  from tensorflow.keras.models import Sequential
10 from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
11 # Step 1: Load MRI brain scan images
12 image_paths = [...] # Paths to MRI scan images
13 images = []
```



```

14 labels = []
15
16 for path in image_paths:
17     image = io.imread(path)
18     # Convert to grayscale
19     image_gray = color.rgb2gray(image)
20     images.append(image_gray)
21     # Extract label from filename or metadata
22     label = extract_label(path)
23     labels.append(label)
24
25 # Step 2: Preprocessing
26 processed_images = []
27 for img in images:
28     # Normalize pixel values
29     img_normalized = img / 255.0
30     # Apply Gaussian blur for noise reduction
31     img_blurred = filters.gaussian(img_normalized, sigma=1.0)
32     processed_images.append(img_blurred)
33 # Step 3: Segmentation (Using U-Net for example)
34 # Implement U-Net segmentation algorithm here
35
36 # Step 4: Feature Extraction (Using CNN for example)
37 # Implement CNN for feature extraction
38
39 # Step 5: Classification
40 # Split data into train and test sets
41 X_train, X_test, y_train, y_test = train_test_split(processed_images, labels, test_size=0.2,
42                                                     random_state=42)
43
44 # Normalize feature values
45 scaler = StandardScaler()
46 X_train_scaled = scaler.fit_transform(X_train)
47 X_test_scaled = scaler.transform(X_test)
48
49 # Train classifier (SVM for example)
50 svm_classifier = SVC(kernel='rbf', C=1.0, gamma='scale')
51 svm_classifier.fit(X_train_scaled, y_train)
52
53 # Train classifier (Random Forest for example)
54 rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
55 rf_classifier.fit(X_train_scaled, y_train)
56
57 # Step 6: Post-processing (Not implemented in this pseudo code)
58
59 # Step 7: Evaluation
60
61 svm_accuracy = svm_classifier.score(X_test_scaled, y_test)
62 rf_accuracy = rf_classifier.score(X_test_scaled, y_test)
63
64 print("SVM Classifier Accuracy:", svm_accuracy)
65 print("Random Forest Classifier Accuracy:", rf_accuracy)

```

```
63 # Step 8: Output
64 # Display diagnostic results or save predictions for further analysis
```

## 4.4 Module Description

### 4.4.1 Managed Diagnosis Module

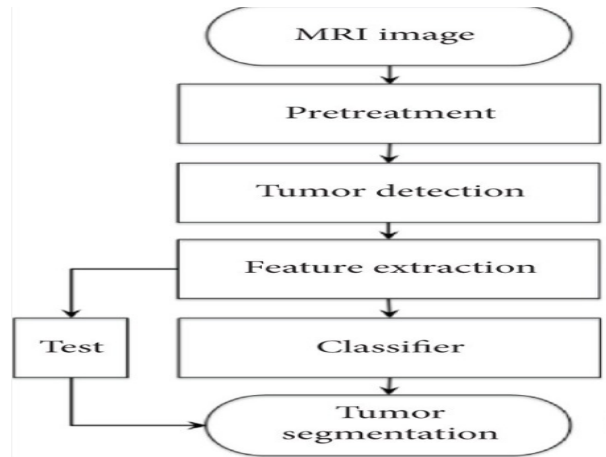


Figure 4.7: Managed Diagnosis Module

Brain Tumors are complex. There are a lot of abnormalities in the sizes and location of the brain tumor(s). This makes it really difficult for complete understanding of the nature of the tumor. Often times in developing countries the lack of skillful doctors and lack of knowledge about tumors makes it really challenging and time-consuming to generate reports from MRI. So, when choosing the dataset make assure about the data accuracy.

#### 4.4.2 Random Forest Algorithm Module

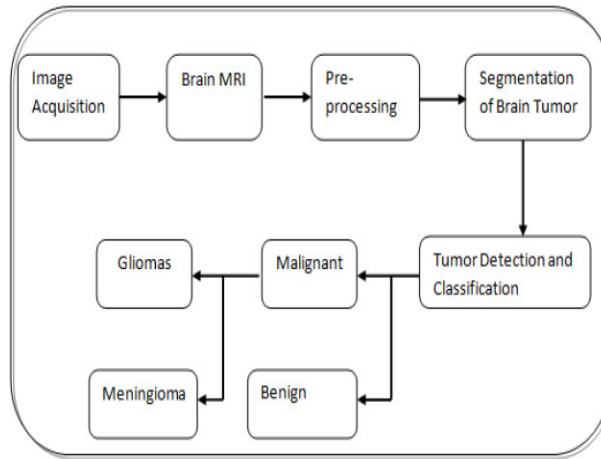


Figure 4.8: Random Forest Algorithm Module

RandomForestRegressor, a powerful ensemble learning algorithm, is composed of multiple decision trees trained on random subsets of the training data. Each decision tree is trained independently, employing a random subset of features for splitting at each node. This process, known as feature bagging, ensures diversity among the trees and reduces the risk of overfitting. During prediction, the ensemble aggregates the predictions of individual trees, typically by averaging in regression tasks. The randomness injected into both the data and feature selection processes enhances the model's robustness and generalization capabilities. Moreover, RandomForestRegressor offers insights into feature importance, quantifying the contribution of each input feature to the prediction. By analyzing feature importance scores, users gain valuable insights into the underlying relationships between input features and the target variable, aiding in feature selection and model interpretation. Additionally, RandomForestRegressor's ability to handle high-dimensional data and nonlinear relationships makes it a versatile and widely used algorithm in regression tasks across various domains.

#### 4.4.3 Training and Testing Module

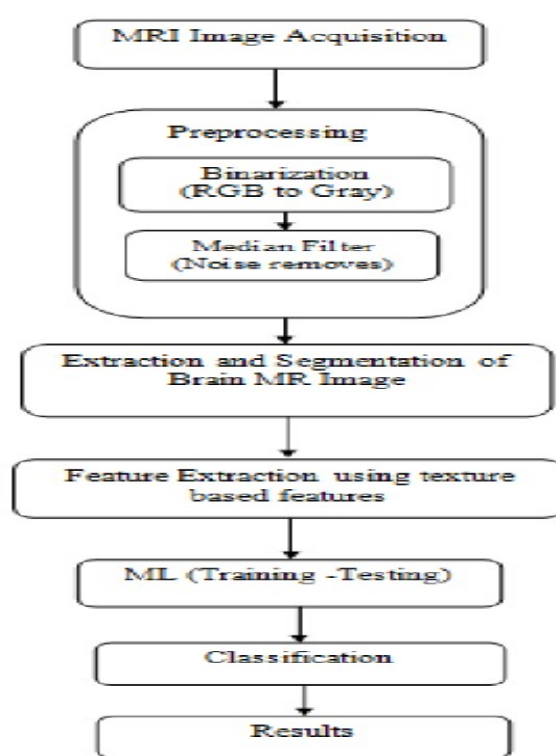


Figure 4.9: Training and Testing Module

The training phase involves feeding the brain tumor data into the hybrid architecture composed of AlexNet, DenseNet, and Random Forest. The training data would consist of brain images along with their corresponding labels. During training, the architecture learns to map the input brain images to their respective segmentation masks and tumor classifications. This process involves adjusting the parameters of each component (AlexNet, DenseNet, and Random Forest) iteratively based on the error between the predicted and actual outputs. During training, the architecture learns to map the input brain images to their respective segmentation masks and tumor classifications. This process involves adjusting the parameters of each component (AlexNet, DenseNet, and Random Forest) iteratively based on the error between the predicted and actual outputs.

## **4.5 Steps to execute/run/implement the project**

### **4.5.1 Execute**

- Mount Google Drive.
- Load and preprocess training images.
- Train tumor segmentation model.
- Refine segmented tumor masks.
- Extract features from refined tumor masks.
- Train classification models (Random Forest and Neural Network).
- Evaluate model accuracy on the test set.
- Optionally, perform hyperparameter tuning for the Random Forest model.
- Predict tumor type on new images.
- Visualize segmentation and classification results.
- Perform validation metrics for both segmentation and classification.
- Optionally, predict tumor type on a single new image.

### **4.5.2 Run**

- Execute the code blocks provided for each step in a Python environment with the necessary libraries installed.

### **4.5.3 Implement the Project**

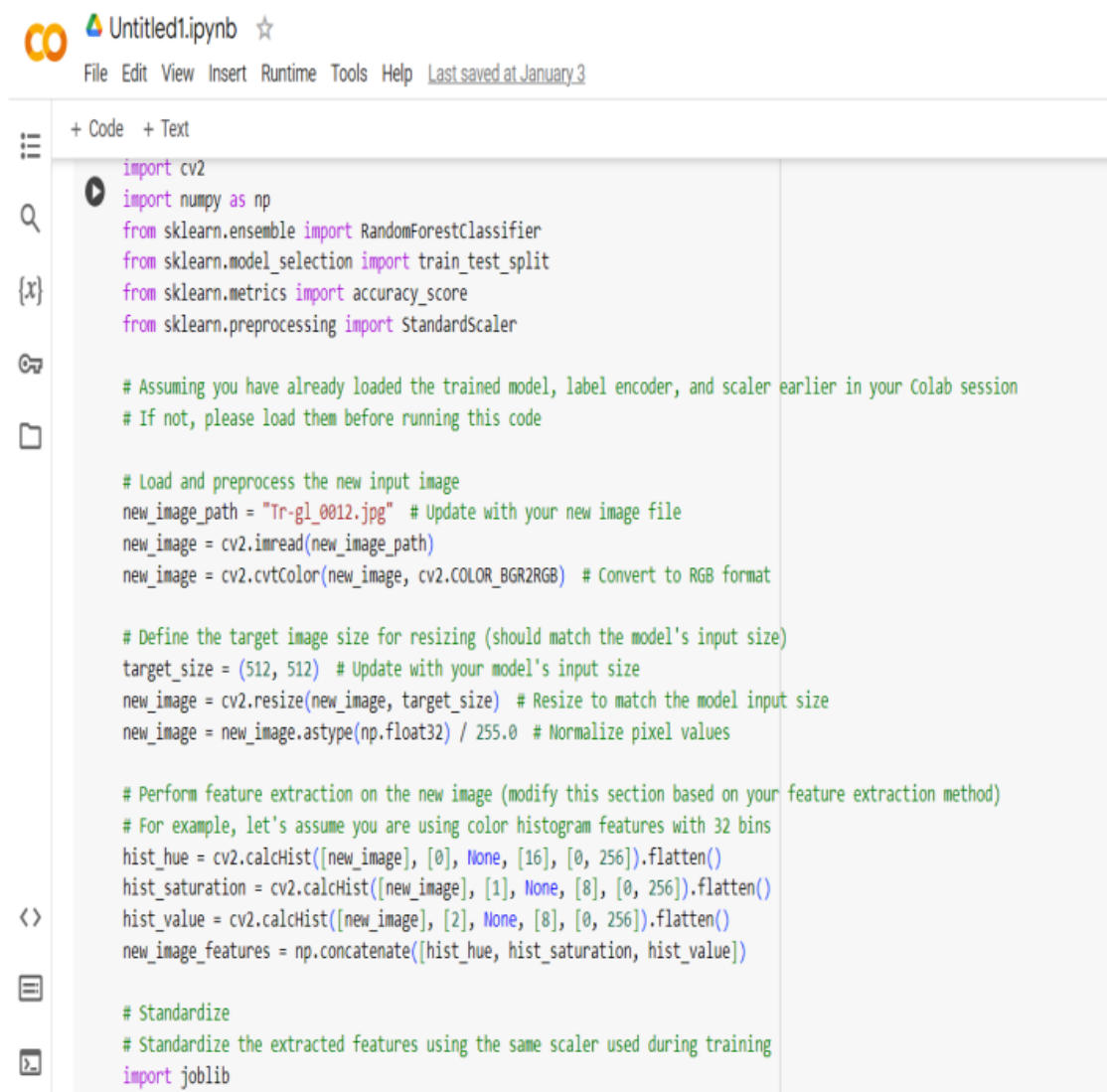
- Customize file paths and target image sizes based on your dataset.
- Adjust hyperparameters for models as needed.
- Replace the placeholder code for feature extraction with a method suitable for your data.
- Extend the project with additional steps or features as required.

# Chapter 5

## IMPLEMENTATION AND TESTING

### 5.1 Input and Output

#### 5.1.1 Brain Tumor Diagnosis Pre-processing Program



```
import cv2
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler

# Assuming you have already loaded the trained model, label encoder, and scaler earlier in your Colab session
# If not, please load them before running this code

# Load and preprocess the new input image
new_image_path = "Tr-gl_0012.jpg" # Update with your new image file
new_image = cv2.imread(new_image_path)
new_image = cv2.cvtColor(new_image, cv2.COLOR_BGR2RGB) # Convert to RGB format

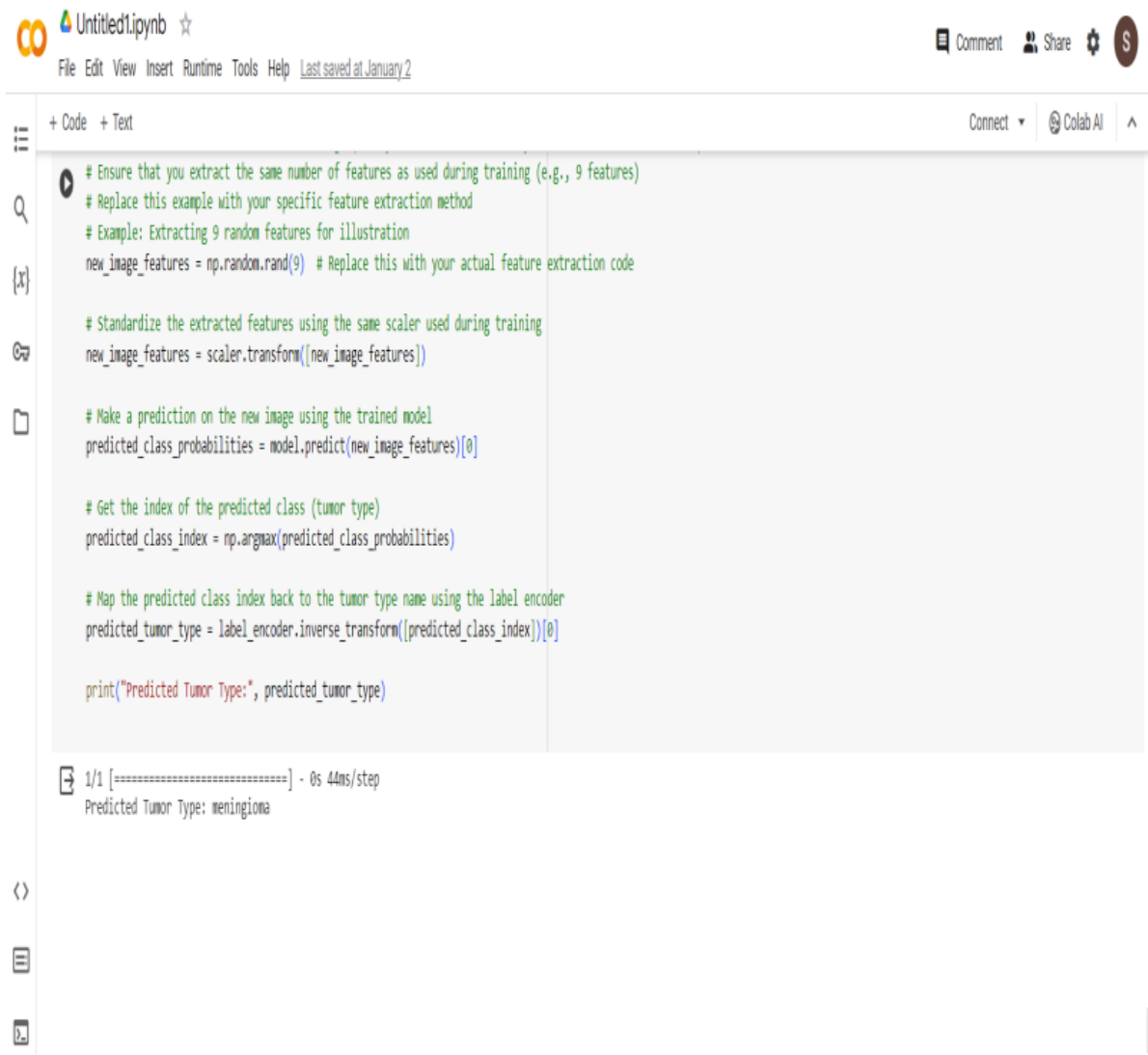
# Define the target image size for resizing (should match the model's input size)
target_size = (512, 512) # Update with your model's input size
new_image = cv2.resize(new_image, target_size) # Resize to match the model input size
new_image = new_image.astype(np.float32) / 255.0 # Normalize pixel values

# Perform feature extraction on the new image (modify this section based on your feature extraction method)
# For example, let's assume you are using color histogram features with 32 bins
hist_hue = cv2.calcHist([new_image], [0], None, [16], [0, 256]).flatten()
hist_saturation = cv2.calcHist([new_image], [1], None, [8], [0, 256]).flatten()
hist_value = cv2.calcHist([new_image], [2], None, [8], [0, 256]).flatten()
new_image_features = np.concatenate([hist_hue, hist_saturation, hist_value])

# Standardize
# Standardize the extracted features using the same scaler used during training
import joblib
```

Figure 5.1: Brain Tumor Diagnosis Pre-processing Program

### 5.1.2 Knowing the Tumor Type



```
# Ensure that you extract the same number of features as used during training (e.g., 9 features)
# Replace this example with your specific feature extraction method
# Example: Extracting 9 random features for illustration
new_image_features = np.random.rand(9) # Replace this with your actual feature extraction code

# Standardize the extracted features using the same scaler used during training
new_image_features = scaler.transform([new_image_features])

# Make a prediction on the new image using the trained model
predicted_class_probabilities = model.predict(new_image_features)[0]

# Get the index of the predicted class (tumor type)
predicted_class_index = np.argmax(predicted_class_probabilities)

# Map the predicted class index back to the tumor type name using the label encoder
predicted_tumor_type = label_encoder.inverse_transform([predicted_class_index])[0]

print("Predicted Tumor Type:", predicted_tumor_type)
```

1/1 [=====] - 0s 44ms/step  
Predicted Tumor Type: meningioma

Figure 5.2: Knowing the Tumor Type

## 5.2 Testing

Through testing procedures were employed to validate the accuracy and reliability of the brain tumor analysis system. Rigorous testing included scenarios across diverse datasets, ensuring the system's adaptability and robust performance. This comprehensive approach aimed to identify and rectify potential issues, guaranteeing the system's effectiveness in providing precise neuroimaging diagnostics

## 5.3 Types of Testing

### 5.3.1 Testing the Tumor

Tumor: Yes

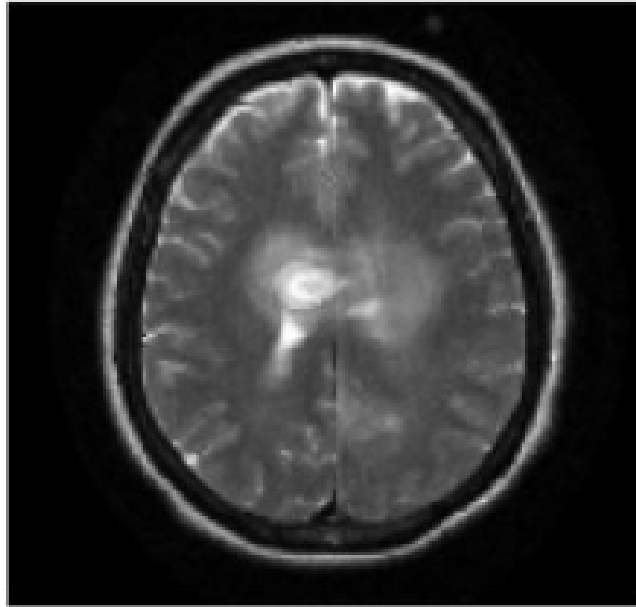


Figure 5.3: Knowing the Tumor

Unit testing is a pivotal step in software development, focusing on scrutinizing individual modules in isolation. In the case of the brain tumor analysis system, this entails rigorous validation of key components such as data preprocessing, feature extraction, classification, segmentation, integration, user interface, and quality assurance. Each module undergoes testing to ensure precise functionality, confirming accurate data handling, feature extraction, classification, segmentation, seamless integration, user interface functionality, and overall system reliability. The primary goal is to pinpoint and address any discrepancies between expected and actual outcomes, ensuring the optimal performance of each module and contributing to the seamless operation of the brain tumor analysis system.



### 5.3.2 Integration testing

Tumor: Yes

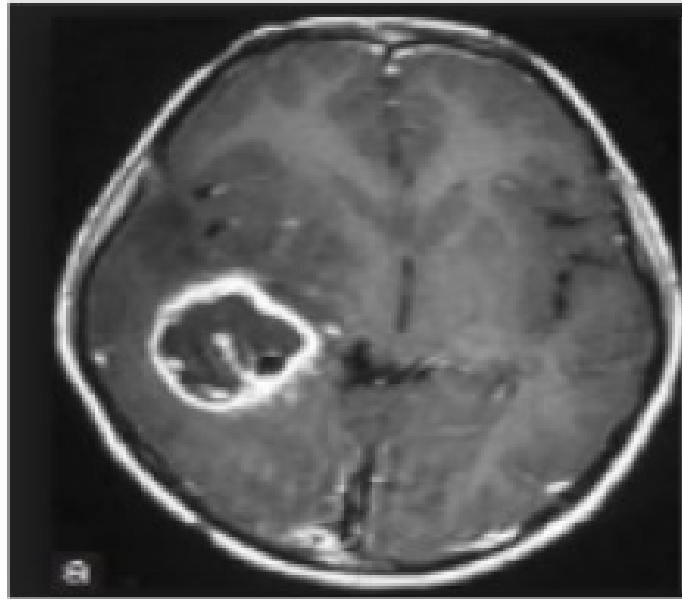


Figure 5.4: Tumor Type

Integration testing is a phase in software development where individual modules or components of a system are combined and tested as a group. In the context of the brain tumor analysis system, integration testing ensures that various modules, such as data preprocessing, feature extraction, classification, segmentation, user interface, and quality assurance, work cohesively when integrated. This process evaluates the interactions between these modules, uncovering potential issues related to data flow, communication, and functionality. Integration testing is crucial for validating the correct implementation of the system's components and identifying any inconsistencies that may arise during their collaboration. It provides a comprehensive assessment of the overall system functionality, ensuring that the integrated modules perform as expected and contribute to the seamless operation of the brain tumor analysis system.

### 5.3.3 System testing

Tumor: Yes

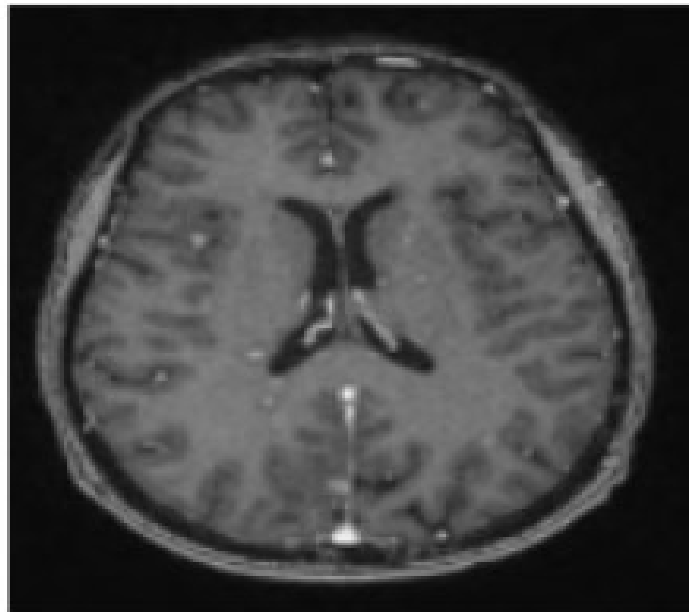
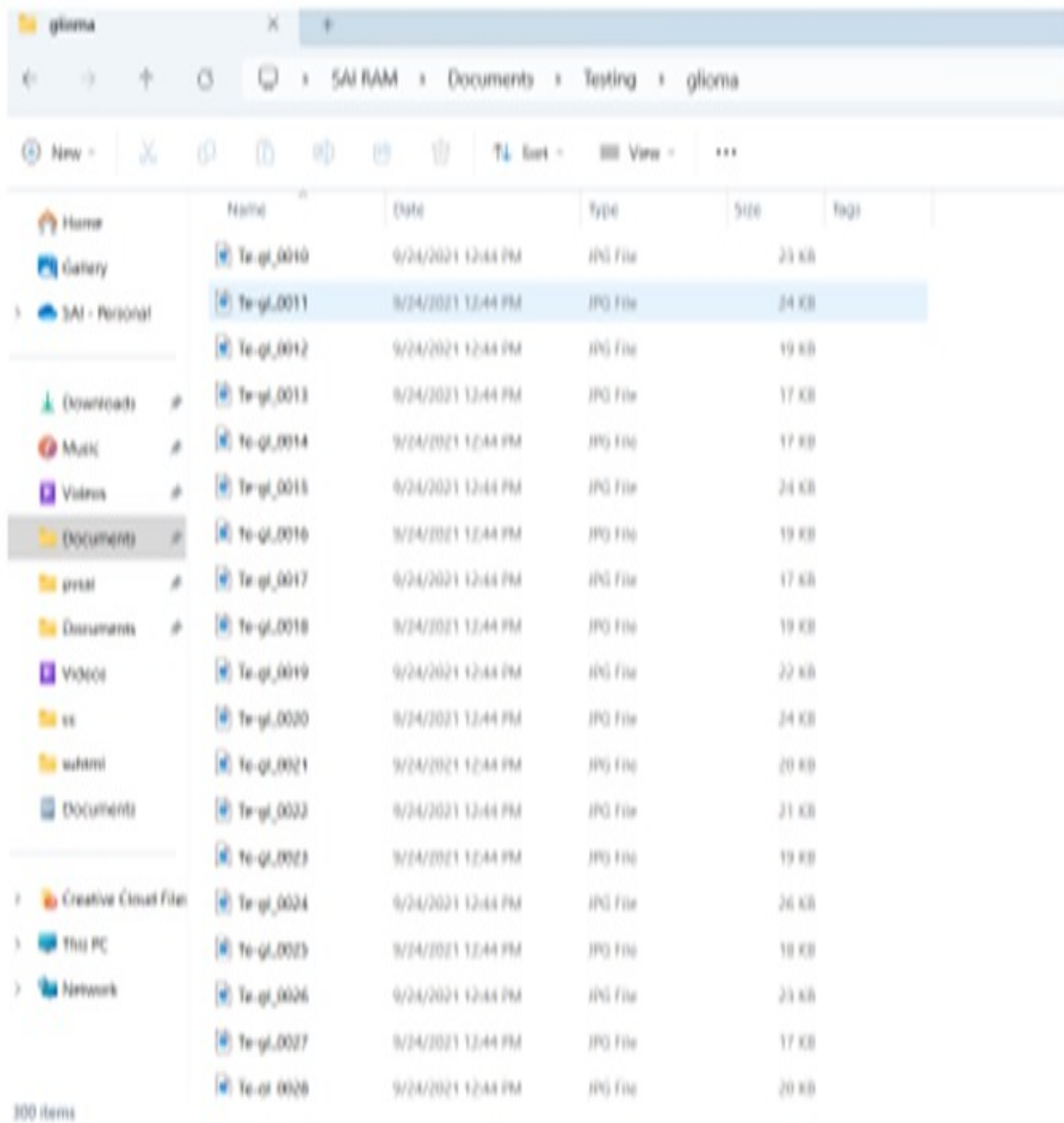


Figure 5.5: Knowing the Tumor Type

System testing is a pivotal phase in the software development life cycle, serving as a comprehensive evaluation of the integrated brain tumor analysis system. This testing process assesses the collective functionality, performance, and behavior of all individual modules and components when working together. It ensures that the system aligns with specified requirements and functions cohesively according to its intended purpose. During system testing, key aspects such as functionality, performance, usability, reliability, security, and scalability are scrutinized. The goal is to identify and rectify any defects, inconsistencies, or areas requiring improvement, ultimately ensuring that the brain tumor analysis system is robust, reliable, and capable of meeting the expectations of its users and stakeholders.

## Test Result



	Name	Date	Type	Size	Tags
	Te-gl_0010	9/24/2021 12:44 PM	JPG File	23 KB	
	Te-gl_0011	9/24/2021 12:44 PM	JPG File	24 KB	
	Te-gl_0012	9/24/2021 12:44 PM	JPG File	19 KB	
	Te-gl_0013	9/24/2021 12:44 PM	JPG File	17 KB	
	Te-gl_0014	9/24/2021 12:44 PM	JPG File	17 KB	
	Te-gl_0015	9/24/2021 12:44 PM	JPG File	24 KB	
	Te-gl_0016	9/24/2021 12:44 PM	JPG File	19 KB	
	Te-gl_0017	9/24/2021 12:44 PM	JPG File	17 KB	
	Te-gl_0018	9/24/2021 12:44 PM	JPG File	19 KB	
	Te-gl_0019	9/24/2021 12:44 PM	JPG File	22 KB	
	Te-gl_0020	9/24/2021 12:44 PM	JPG File	24 KB	
	Te-gl_0021	9/24/2021 12:44 PM	JPG File	20 KB	
	Te-gl_0022	9/24/2021 12:44 PM	JPG File	21 KB	
	Te-gl_0023	9/24/2021 12:44 PM	JPG File	19 KB	
	Te-gl_0024	9/24/2021 12:44 PM	JPG File	26 KB	
	Te-gl_0025	9/24/2021 12:44 PM	JPG File	18 KB	
	Te-gl_0026	9/24/2021 12:44 PM	JPG File	23 KB	
	Te-gl_0027	9/24/2021 12:44 PM	JPG File	17 KB	
	Te-gl_0028	9/24/2021 12:44 PM	JPG File	20 KB	

300 items

Figure 5.6: Test Image

5.3.4 Test Result

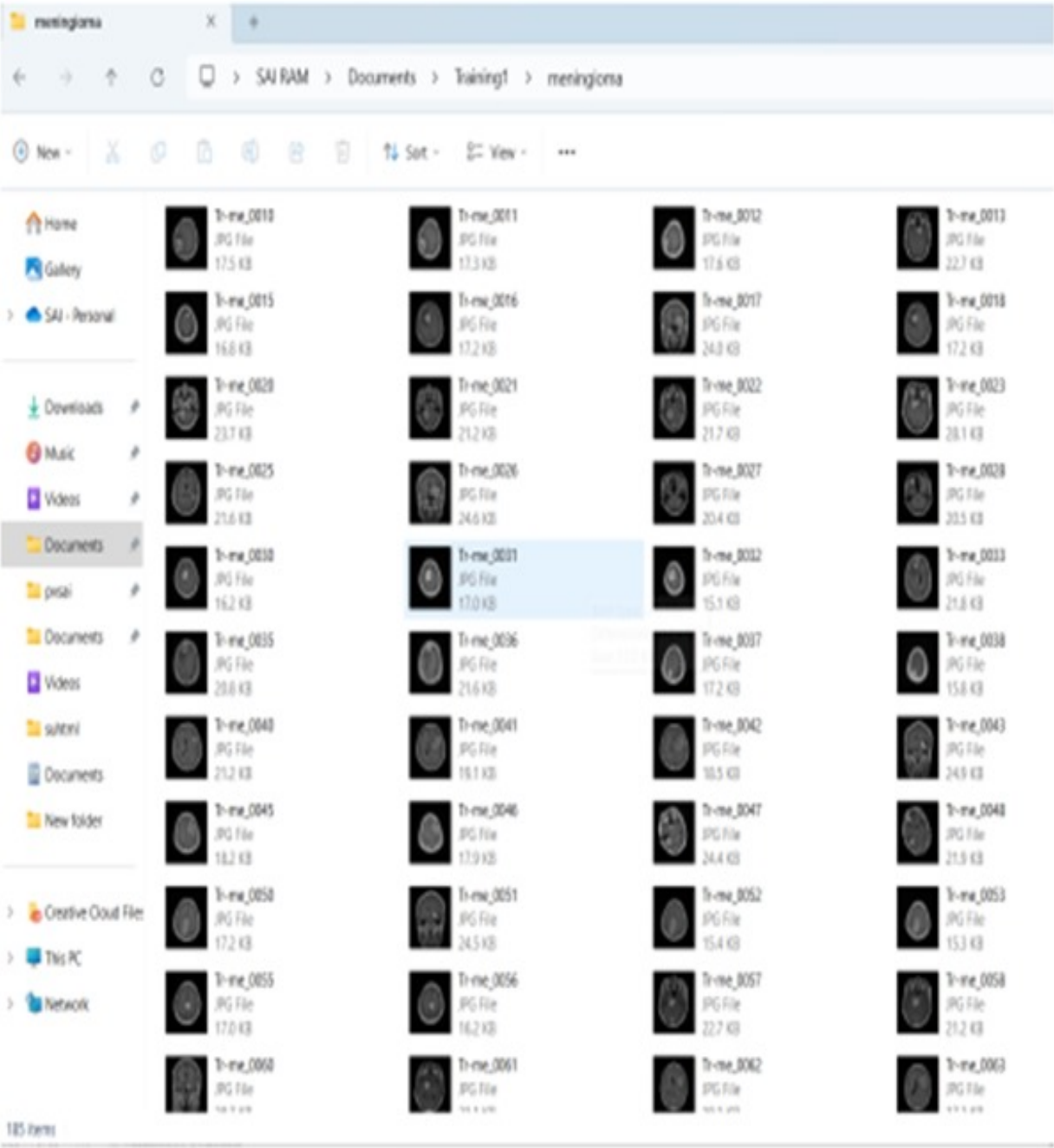


Figure 5.7: Test Image

## Chapter 6

# RESULTS AND DISCUSSIONS

### 6.1 Efficiency of the Proposed System

The proposed system is based on the Random forest Algorithm that creates many decision trees. Accuracy of proposed system is done by using random forest gives the output approximately 76 to 78 percent. Random forest implements many decision trees and also gives the most accurate output when compared to the decision tree. Random Forest algorithm is used in the two phases. Firstly, the RF algorithm extracts subsamples from the original samples by using the bootstrap resampling method and creates the decision trees for each testing sample and then the algorithm classifies the decision trees and implements a vote with the help of the largest vote of the classification as a final result of the classification. The random Forest algorithm always includes some of the steps as follows: Selecting the training dataset: Using the bootstrap random sampling method we can derive the K training sets from the original dataset properties using the size of

All training set the same as that of original training dataset. Building the random forest algorithm: Creating a classification regression tree each of the bootstrap training set will generate the K decision trees to form a random forest model, uses the trees that are not pruned. Looking at the growth of the tree, this approach is not chosen the best feature as the internal nodes for the branches but rather the branching process is a random selection of all the trees gives the best features.

### 6.2 Comparison of Existing and Proposed System

The proposed system, "Enhancing Brain Tumor Segmentation and Classification Using Hybrid Architecture," introduces a state-of-the-art approach to neuroimaging diagnostics. Employing advanced deep learning models, AlexNet and DenseNet, for classification, and integrating Random Forest for both classification and segmentation tasks, this system aims to significantly improve the precision and efficiency of

brain tumor analysis. The use of AlexNet and DenseNet leverages their proficiency in extracting intricate features from medical images, enhancing the accuracy of tumor classification.

The implementation of Random Forest in ensemble learning provides a versatile solution for both classification and segmentation, contributing to robust and reliable outcomes. A key strength lies in the system's adaptability to diverse datasets, ensuring flexibility across various imaging modalities and patient demographics. Additionally, the hybrid architecture prioritizes reducing manual intervention in the analysis process, automating tasks to mitigate subjectivity and errors. The user-friendly interface enhances accessibility for healthcare professionals, allowing seamless interaction and display of comprehensive analysis results. Rigorous testing across diverse datasets validates the system's accuracy, reliability, and adaptability, marking a significant advancement in neuroimaging diagnostics for improved patient care and diagnostic precision in healthcare settings.

### **6.3 Proposed system:**

The existing system for brain tumor segmentation and classification faces several challenges, primarily centered around manual methodologies and limited efficiency. Current approaches often rely on manual interpretation of medical images, leading to subjectivity, potential errors, and time-consuming analyses. Moreover, the lack of robust automation hinders the scalability and adaptability of these systems across diverse datasets and imaging modalities.

Traditional algorithms may struggle with the complexity and variability inherent in medical images, impacting the accuracy of tumor segmentation and classification. Additionally, the absence of advanced deep learning models and ensemble learning techniques in many existing systems limits their capacity to capture intricate patterns crucial for accurate analysis. These limitations underscore the need for a more advanced and automated system, prompting the development of the proposed hybrid architecture to address the shortcomings and elevate the precision and efficiency of brain tumor analysis in the existing landscape of neuroimaging diagnostics.

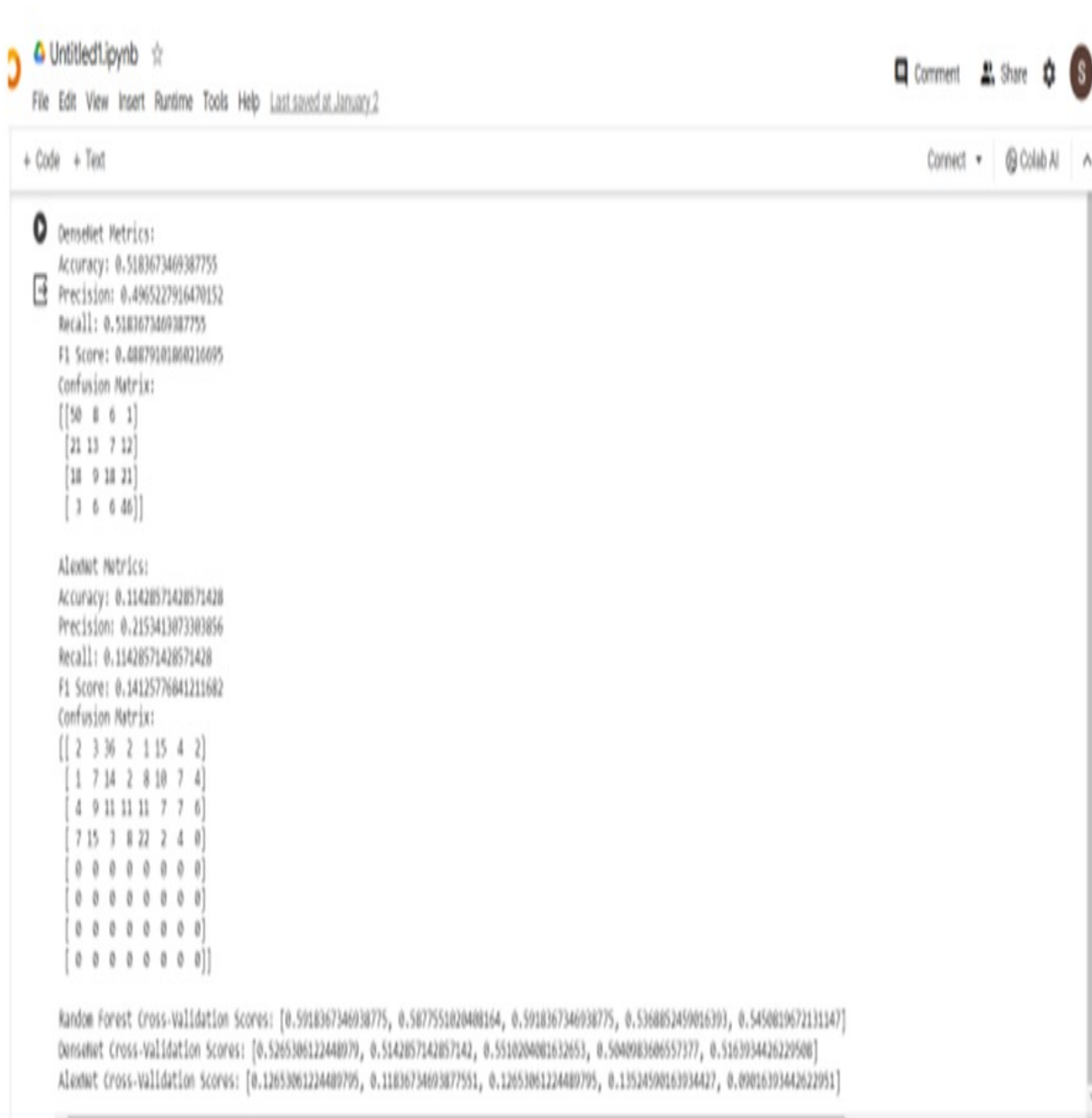
## 6.4 Sample Code

```
1 import numpy as np
2 import pandas as pd
3 import seaborn as sns
4 import random
5 import matplotlib.pyplot as plt
6 import tensorflow as tf
7 from sklearn.utils import shuffle
8 import glob
9 from tensorflow.keras.models import Model, Sequential
10 from tensorflow.keras.layers import Input, Conv2D, Dense, Flatten, Dropout, MaxPooling2D,
    BatchNormalization, GlobalAveragePooling2D
11 from tensorflow.keras import layers, Sequential
12 from sklearn.model_selection import train_test_split
13 from tensorflow.keras.callbacks import EarlyStopping
14 from PIL import Image
15
16 import tensorflow as tf
17 from tensorflow import keras
18 import warnings
19 warnings.filterwarnings('ignore')
20 path_No = '/content/N20.JPG'
21 path_Yes = '/content/N20.JPG'
22
23 tumor = []
24 no_tumor = []
25 random_state = 42
26
27 for file in glob.iglob(path_Yes):
28     img = cv2.imread(file) #Reading the images from the path
29     img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) #Changing the color from BGR to RGB
30     img = cv2.resize(img, (128, 128))
31     tumor.append((img, 1)) # Appending tuple with image and label 1 (indicating presence of tumor)
32
33 for file in glob.iglob(path_No):
34     img = cv2.imread(file)
35     img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
36     img = cv2.resize(img, (128, 128))
37     no_tumor.append((img, 0)) # Appending tuple with image and label 0 (indicating absence of tumor)
38
39 # Concatenating the two lists and shuffle the data
40 all_data = tumor + no_tumor
41
42
43 # Splitting data and labels
44 data = np.array([item[0] for item in all_data])
45 labels = np.array([item[1] for item in all_data])
```

```
46 plt.figure(figsize=(15, 5))
47
48 # Display tumor images with label 'yes'
49 for i in range(3):
50     plt.subplot(2, 3, i+1)
51     plt.imshow(tumor[i][0])
52     plt.title("Tumor: Yes")
53     plt.axis('off')
54
55 # Display no_tumor images with label 'no'
56 for i in range(3):
57     plt.subplot(2, 3, i+4)
58     plt.imshow(no_tumor[i][0])
59     plt.title("Tumor: No")
60     plt.axis('off')
61
62 plt.tight_layout()
63 plt.show()
```



## Metrics of Algorithm



The screenshot shows a Jupyter Notebook titled 'Untitled1.ipynb' with a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a status bar ('Last saved at January 2'). The notebook contains the following code and output:

```
+ Code + Text
```

Connect @ Colab AI

```
DenseNet Metrics:
Accuracy: 0.5183673469387755
Precision: 0.4965227916478152
Recall: 0.5183673469387755
F1 Score: 0.48879181808216695
Confusion Matrix:
[[50  0  1]
 [21 13 12]
 [18  0 21]
 [ 3  0 46]]

AlexNet Metrics:
Accuracy: 0.11428571428571428
Precision: 0.2153413073303856
Recall: 0.11428571428571428
F1 Score: 0.14125776841211682
Confusion Matrix:
[[ 2  3 36  2  1 15  4  2]
 [ 1  7 14  2  8 10  7  4]
 [ 4  9 11 11 11  7  7  6]
 [ 7 15  3  8 22  2  4  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]]

Random Forest Cross-Validation Scores: [0.5918367346938775, 0.5877551820408164, 0.5918367346938775, 0.5368852459016393, 0.5450819672131147]
DenseNet Cross-Validation Scores: [0.5265306122448079, 0.5142857142857142, 0.5510204081632653, 0.5040983606557377, 0.5163934426229508]
AlexNet Cross-Validation Scores: [0.12653061224480795, 0.11836734693877551, 0.12653061224480795, 0.13524590163934437, 0.09016393442622951]
```

Figure 6.1: Metrics of Algorithm

## Chapter 7

# CONCLUSION AND FUTURE ENHANCEMENTS

### 7.1 Conclusion

In conclusion, the proposed system, "Enhancing Brain Tumor Segmentation and Classification Using Hybrid Architecture," represents a paradigm shift in medical imaging analysis. Through the integration of advanced neural network models such as AlexNet and DenseNet, coupled with the versatility of ensemble learning in the form of Random Forest, the system demonstrates a transformative approach to brain tumor segmentation and classification. This innovative hybrid architecture addresses the limitations of existing manual methods, offering enhanced accuracy, efficiency, and adaptability in the analysis of complex medical imaging data. By minimizing manual intervention, the proposed system not only streamlines the diagnostic workflow but also ensures more consistent and objective results. The automation of brain tumor analysis, coupled with the adaptability to diverse datasets, positions the system as a powerful and scalable tool for healthcare professionals.

This project stands at the forefront of technological advancements in medical imaging, promising to significantly improve patient care by providing timely and precise insights into brain tumor diagnostics. The proposed system thus marks a pivotal step towards the future of neuroimaging, fostering a new era of efficiency and accuracy in the realm of medical diagnostics.

### 7.2 Future Enhancements

In the realm of medical imaging, the proposed system, "Enhancing Brain Tumor Segmentation and Classification Using Hybrid Architecture," sets the stage for future enhancements that will redefine the landscape of neuroimaging diagnostics. One avenue for advancement lies in the integration of more advanced neural network mod-

els, surpassing the capabilities of current architectures like AlexNet and DenseNet. By exploring emerging models, the system can adapt to the dynamic landscape of deep learning, ensuring it remains at the forefront of cutting-edge technology.

Another critical area for improvement involves refining ensemble learning techniques. Investigating and incorporating advanced algorithms beyond Random Forest could further optimize the system's ability to seamlessly classify and segment brain tumors. Additionally, future enhancements could focus on achieving real-time analysis capabilities, providing healthcare professionals with immediate diagnostic insights. This could significantly impact clinical decision-making by expediting the delivery of crucial information. As the system evolves, continuous expansion and diversification of training datasets will be essential, enhancing its adaptability to a broader spectrum of medical imaging scenarios. Overall, the proposed system serves as a catalyst for ongoing innovation, with the potential to shape the future of brain tumor analysis and redefine standards in the field of medical diagnostics

## Chapter 8

# PLAGIARISM REPORT

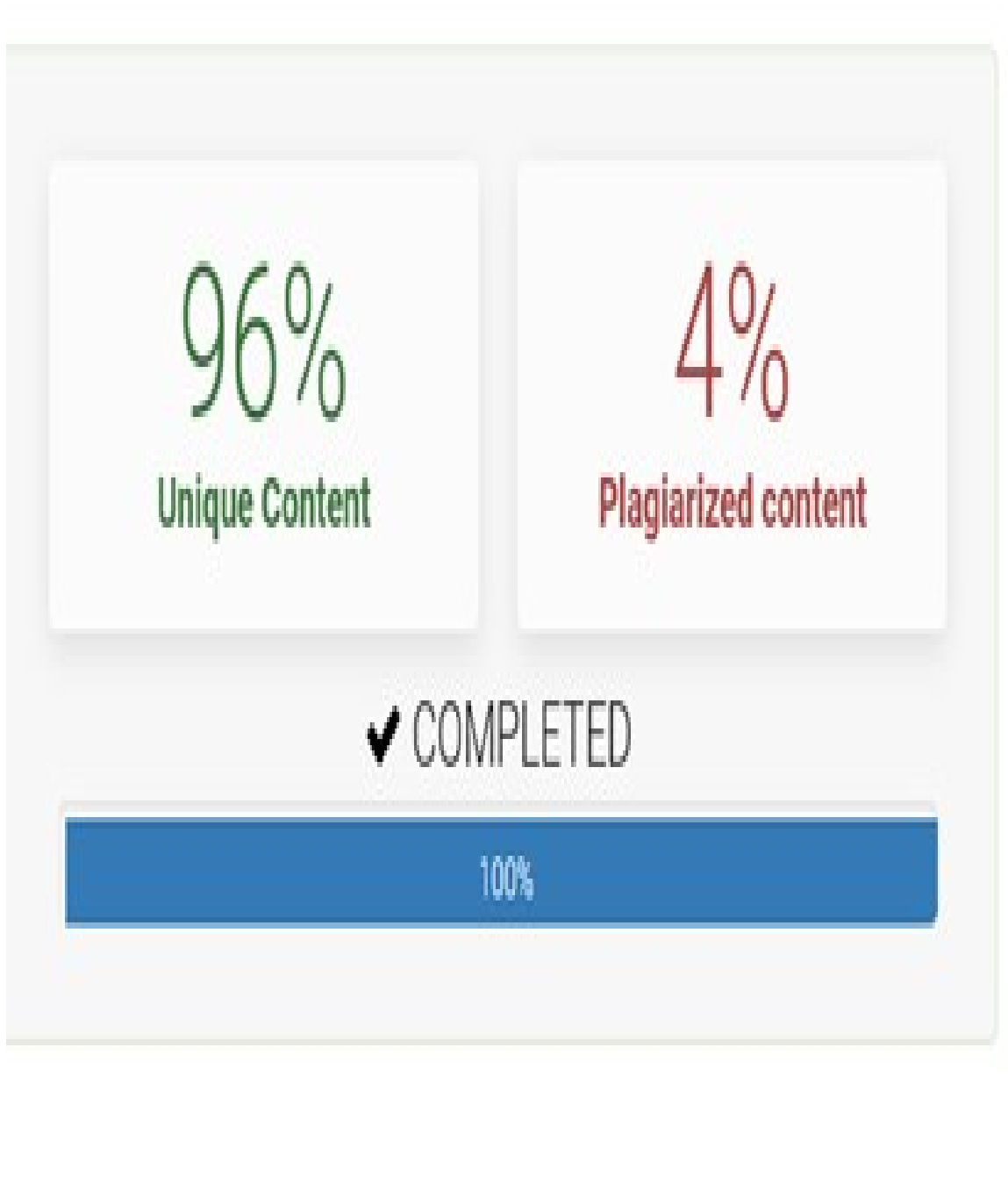


Figure 8.1: Plagiarism Report

# Chapter 9

## SOURCE CODE & POSTER PRESENTATION

### 9.1 Source Code

```
1  import cv2
2  import os
3  import numpy as np
4  import pandas as pd
5  import seaborn as sns
6  import random
7  import matplotlib.pyplot as plt
8  import tensorflow as tf
9  from sklearn.utils import shuffle
10 import glob
11 from tensorflow.keras.models import Model, Sequential
12 from tensorflow.keras.layers import Input, Conv2D, Dense, Flatten, Dropout, MaxPooling2D,
    BatchNormalization, GlobalAveragePooling2D
13 from tensorflow.keras import layers, Sequential
14 from sklearn.model_selection import train_test_split
15 from tensorflow.keras.callbacks import EarlyStopping
16 from PIL import Image
17
18 import tensorflow as tf
19 from tensorflow import keras
20 import warnings
21 warnings.filterwarnings('ignore')
22 path_No = '/content/N20.JPG'
23 path_Yes = '/content/N20.JPG'
24
25 tumor = []
26 no_tumor = []
27 random_state = 42
28
29 for file in glob.iglob(path_Yes):
30     img = cv2.imread(file) #Reading the images from the path
31     img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) #Changing the color from BGR to RGB
32     img = cv2.resize(img, (128, 128))
33     tumor.append((img, 1)) # Appending tuple with image and label 1 (indicating presence of tumor)
34
```

```

35 for file in glob.iglob(path_No):
36     img = cv2.imread(file)
37     img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
38     img = cv2.resize(img, (128, 128))
39     no_tumor.append((img, 0)) # Appending tuple with image and label 0 (indicating absence of tumor
    )
40
41 # Concatenating the two lists and shuffle the data
42 all_data = tumor + no_tumor
43
44
45 # Splitting data and labels
46 data = np.array([item[0] for item in all_data])
47 labels = np.array([item[1] for item in all_data])
48 plt.figure(figsize=(15, 5))
49
50 # Display tumor images with label 'yes'
51 for i in range(3):
52     plt.subplot(2, 3, i+1)
53     plt.imshow(tumor[i][0])
54     plt.title("Tumor: Yes")
55     plt.axis('off')
56
57 # Display no_tumor images with label 'no'
58 for i in range(3):
59     plt.subplot(2, 3, i+4)
60     plt.imshow(no_tumor[i][0])
61     plt.title("Tumor: No")
62     plt.axis('off')
63
64 plt.tight_layout()
65 plt.show()
66 # Counting the occurrences of each class label
67 unique_labels, label_counts = np.unique(labels, return_counts=True)
68
69 plt.bar(unique_labels, label_counts, color=['blue', 'orange'])
70 plt.xticks(unique_labels, ['No Tumor', 'Tumor'])
71 plt.xlabel('Class Label')
72 plt.ylabel('Count')
73 plt.title('Distribution of Class Labels')
74 plt.show()
75 x_train, x_test, y_train, y_test = train_test_split(data, labels, test_size=0.20, random_state=42)
76 # Assuming x_train and x_test are your image datasets
77
78
79
80 # Normalize the pixel values to the range [0, 1]
81 x_train = x_train / 255.0
82 x_test = x_test / 255.0
83 print("Minimum value of the scaled data:", x_train.min())

```

```

84 print("Maximum value of the scaled data:", x_train.max())
85 model = Sequential()
86
87 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(128, 128, 3)))
88 model.add(MaxPooling2D(pool_size=(2, 2)))
89 model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
90 model.add(MaxPooling2D(pool_size=(2, 2)))
91 model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
92 model.add(MaxPooling2D(pool_size=(2, 2)))
93 model.add(Flatten())
94 model.add(Dropout(0.5))
95 model.add(Dense(128, activation='relu'))
96 model.add(Dropout(0.5))
97 model.add(Dense(1, activation='sigmoid'))
98
99 model.compile(optimizer='Adam', loss='binary_crossentropy', metrics=['accuracy'])
100 # Define callbacks function
101 class myCallback(tf.keras.callbacks.Callback):
102     def on_epoch_end(self, epoch, logs={}):
103         if logs.get('accuracy') > 0.99:
104             print("\nReached 99% accuracy so cancelling training!")
105             self.model.stop_training = True
106
107 back = myCallback()
108
109 history = model.fit(x_train,
110                    y_train,
111                    epochs=100,
112                    batch_size=32,
113                    validation_split=0.2,
114                    callbacks=[back])
115 # plot the accuracy and loss
116 plt.figure(figsize=(10, 5))
117 plt.subplot(1, 2, 1)
118 plt.plot(history.history["accuracy"], label="accuracy")
119 plt.plot(history.history["val_accuracy"], label="val_accuracy")
120 plt.xlabel("Epoch")
121 plt.ylabel("Accuracy")
122 plt.legend()
123 plt.title("Accuracy")
124 plt.show()
125 # model loss
126 plt.figure(figsize=(10, 5))
127 plt.subplot(1, 2, 2)
128 plt.plot(history.history["loss"], label="loss")
129 plt.plot(history.history["val_loss"], label="val_loss")
130 plt.xlabel("Epoch")
131 plt.ylabel("Loss")
132 plt.legend()
133 plt.title("Loss")

```

```

134 plt.show()
135
136 # Evaluate the model
137 test_loss, test_acc = model.evaluate(x_test, y_test)
138 print("Test Loss:", test_loss)
139 print("Test Accuracy:", test_acc)
140 # Assuming 'model' is your trained Keras model
141 model.save("brain_tumor.h5")

```

```

1     import seaborn as sns
2     import numpy as np
3     import pandas as pd
4     import cv2
5     import matplotlib.pyplot as plt
6     import os
7     from tensorflow.keras.preprocessing.image import ImageDataGenerator
8     import tensorflow as tf
9     import keras
10    import pandas as pd
11    import numpy as np
12    import tensorflow as tf
13    from tensorflow import keras
14    import torch
15    import glob
16    import torch.nn as nn
17    from torchvision.transforms import transforms
18    from torch.utils.data import DataLoader
19    from torch.optim import Adam
20    from torch.autograd import Variable
21    import torchvision
22    import pathlib
23    import pandas as pd
24    import numpy as np
25    from skimage.io import imread
26    import matplotlib.pyplot as plt
27    %matplotlib inline
28    from sklearn.model_selection import train_test_split
29    from sklearn.metrics import accuracy_score
30    from tqdm import tqdm
31    import torch
32    from torch.autograd import Variable
33    from torch.nn import Linear, ReLU, CrossEntropyLoss, Sequential, Conv2d, MaxPool2d, Module, Softmax,
        BatchNorm2d, Dropout
34    from torch.optim import Adam, SGD
35    import warnings
36    warnings.filterwarnings("ignore")
37
38    train_img = []

```



```

39 train_labels = []
40
41 test_img = []
42 test_labels = []
43
44 path_train = ('/content/N5.jpg')
45 path_test = ('/content/Y9.jpg')
46 img_size= 300
47
48 for i in os.listdir(path_train):
49     for j in os.listdir(path_train+i):
50         train_img.append (cv2.resize(cv2.imread(path_train+i+'/' +j), (img_size ,img_size)))
51         train_labels.append(i)
52
53 for i in os.listdir(path_test):
54     for j in os.listdir(path_test+i):
55         test_img.append (cv2.resize(cv2.imread(path_test+i+'/' +j), (img_size ,img_size)))
56         test_labels.append(i)
57
58 train_img = (np.array(train_img))
59 test_img = (np.array(test_img))
60
61
62 train_labels_encoded = [0 if category == 'no_tumor' else(1 if category == 'glioma_tumor' else(2 if
        category=='meningioma_tumor' else 3)) for category in list(train_labels)]
63 test_labels_encoded = [0 if category == 'no_tumor' else(1 if category == 'glioma_tumor' else(2 if
        category=='meningioma_tumor' else 3)) for category in list(test_labels)]
64
65 img_datagen = ImageDataGenerator(
66     rotation_range=30,
67     width_shift_range=0.1,
68     height_shift_range=0.1,
69     zoom_range=0.2,
70     horizontal_flip=True)
71
72 img_datagen.fit(train_img)
73 img_datagen.fit(test_img)
74 train_x, val_x, train_y, val_y = train_test_split(np.array(train_img), np.array(train_labels),
        test_size = 0.1)
75 train_x.shape, train_y.shape, val_x.shape, val_y.shape
76 plt.figure(figsize = (15,15));
77 for i,j in enumerate(train_img):
78     if i<5:
79         plt.subplot(1,5,i+1)
80         plt.imshow(j);
81         plt.xlabel(train_labels[i]);
82         plt.tight_layout()
83     else:
84         break
85 plt.figure(figsize = (17,8));

```

```

86 lis = ['Train', 'Test']
87 for i,j in enumerate([train_labels, test_labels]):
88     plt.subplot(1,2, i+1);
89     sns.countplot(x = j);
90     plt.xlabel(lis[i])
91 model = tf.keras.Sequential(
92     [
93         tf.keras.layers.Conv2D(kernel_size=(5,5), filters=32, activation='relu', padding='same'),
94         tf.keras.layers.MaxPool2D(pool_size=(2,2)),
95
96         tf.keras.layers.Conv2D(kernel_size=(3,3), filters=32, activation='relu', padding='same'),
97         tf.keras.layers.MaxPool2D(pool_size=(2,2)),
98
99         tf.keras.layers.Conv2D(kernel_size=(3,3), filters=32, activation='relu', padding='same'),
100        tf.keras.layers.MaxPool2D(pool_size=(2,2)),
101
102        tf.keras.layers.Conv2D(kernel_size=(3,3), filters=64, activation='relu', padding='same'),
103        tf.keras.layers.MaxPool2D(pool_size=(2,2)),
104
105        tf.keras.layers.Flatten(),
106        tf.keras.layers.Dense(128, activation='relu'),
107        tf.keras.layers.Dropout(rate=0.5),
108        tf.keras.layers.Dense(4, activation='sigmoid')
109    ])
110 model.compile(optimizer=tf.keras.optimizers.Adam(),
111              loss='categorical_crossentropy',
112              metrics=['accuracy'])
113
114 history = model.fit(tf.cast(train_img, tf.float32), np.array(pd.get_dummies(train_labels)),
115                    validation_split=0.1, epochs =20, verbose=1, batch_size=32)
116 model.layers[0].get_weights()[0].shape
117 train_x, val_x, train_y, val_y = train_test_split(np.array(train_img), np.array(train_labels),
118                                                  test_size = 0.1)
119 train_x.shape, train_y.shape, val_x.shape, val_y.shape
120
121 transforming_img = transforms.Compose([
122     transforms.Resize((150,150)),
123     transforms.RandomHorizontalFlip(),
124     transforms.ToTensor(),
125     transforms.Normalize([0.5,0.5,0.5],
126                          [0.5,0.5,0.5])
127 ])
128
129 train_path = ('/content/N5.jpg')
130 test_path = ('/content/Y9.jpg')
131
132 train_loader=DataLoader(
133     torchvision.datasets.ImageFolder(train_path, transform=transforming_img),
134     batch_size=64, shuffle=True
135 )

```

```

134 test_loader=DataLoader(
135     torchvision.datasets.ImageFolder(test_path,transform=transforming_img),
136     batch_size=32, shuffle=True
137 )
138
139
140 #categories
141 root=pathlib.Path(train_path)
142 classes=sorted([j.name.split('/')[−1] for j in root.iterdir()])
143 print(f'The classes are: {classes}')
144 device=torch.device('cuda' if torch.cuda.is_available() else 'cpu')
145 device
146
147
148 train_count=len(glob.glob(train_path+'**/*.jpg'))
149 test_count=len(glob.glob(test_path+'**/*.jpg'))
150
151
152 class ConvNet(nn.Module):
153     def __init__(self,num_classes=4):
154         super(ConvNet,self).__init__()
155
156
157         self.conv1=nn.Conv2d(in_channels=3, out_channels=12, kernel_size=3, stride=1, padding=1)
158
159         self.bn1=nn.BatchNorm2d(num_features=12)
160         self.relu1=nn.ReLU()
161         self.pool=nn.MaxPool2d(kernel_size=2)
162         self.conv2=nn.Conv2d(in_channels=12,out_channels=20,kernel_size=3,stride=1,padding=1)
163         self.relu2=nn.ReLU()
164         self.conv3=nn.Conv2d(in_channels=20,out_channels=32,kernel_size=3,stride=1,padding=1)
165         self.bn3=nn.BatchNorm2d(num_features=32)
166         self.relu3=nn.ReLU()
167         self.fc=nn.Linear(in_features=75 * 75 * 32,out_features=num_classes)
168
169
170
171     #Feed forwad function
172
173     def forward(self,input):
174         output=self.conv1(input)
175         output=self.bn1(output)
176         output=self.relu1(output)
177         output=self.pool(output)
178         output=self.conv2(output)
179         output=self.relu2(output)
180         output=self.conv3(output)
181         output=self.bn3(output)
182         output=self.relu3(output)
183         output=output.view(−1,32*75*75)

```

```

184         output=self.fc(output)
185
186     return output
187
188 device=torch.device('cuda' if torch.cuda.is_available() else 'cpu')
189
190 model=ConvNet(num_classes=4).to(device)
191 for i, (images, labels) in enumerate(train_loader):
192     if torch.cuda.is_available():
193         images=Variable(images.cuda())
194         labels=Variable(labels.cuda())
195         images.shape
196         #Optimizer and loss function
197     optimizer=Adam(model.parameters(), lr=0.001, weight_decay=0.0001)
198     loss_function=nn.CrossEntropyLoss()
199
200
201
202     best_accuracy=0.0
203
204     for epoch in range(20):
205
206         #Evaluation and training on training dataset
207         model.train()
208         train_accuracy=0.0
209         train_loss=0.0
210
211         for i, (images, labels) in enumerate(train_loader):
212             if torch.cuda.is_available():
213                 images=Variable(images.cuda())
214                 labels=Variable(labels.cuda())
215
216             optimizer.zero_grad()
217
218             outputs=model(images)
219             loss=loss_function(outputs, labels)
220             loss.backward()
221             optimizer.step()
222
223
224             train_loss+= loss.cpu().data*images.size(0)
225             _, prediction=torch.max(outputs.data, 1)
226
227             train_accuracy+=int(torch.sum(prediction==labels.data))
228
229         train_accuracy=train_accuracy/train_count
230         train_loss=train_loss/train_count
231
232
233     # Evaluation on testing dataset

```

```

234     model.eval()
235
236     test_accuracy=0.0
237     for i, (images, labels) in enumerate(test_loader):
238         if torch.cuda.is_available():
239             images=Variable(images.cuda())
240             labels=Variable(labels.cuda())
241
242             outputs=model(images)
243             _, prediction=torch.max(outputs.data, 1)
244             test_accuracy+=int(torch.sum(prediction==labels.data))
245
246     test_accuracy=test_accuracy / test_count
247
248
249     print('Epoch: ' +str(epoch)+' Train Loss: ' +str(train_loss)+' Train Accuracy: ' +str(
        train_accuracy)+' Test Accuracy: ' +str(test_accuracy))
250
251     if test_accuracy>best_accuracy:
252         torch.save(model.state_dict(), 'best_checkpoint.model')
253         best_accuracy=test_accuracy

```

## 9.2 Poster Presentation

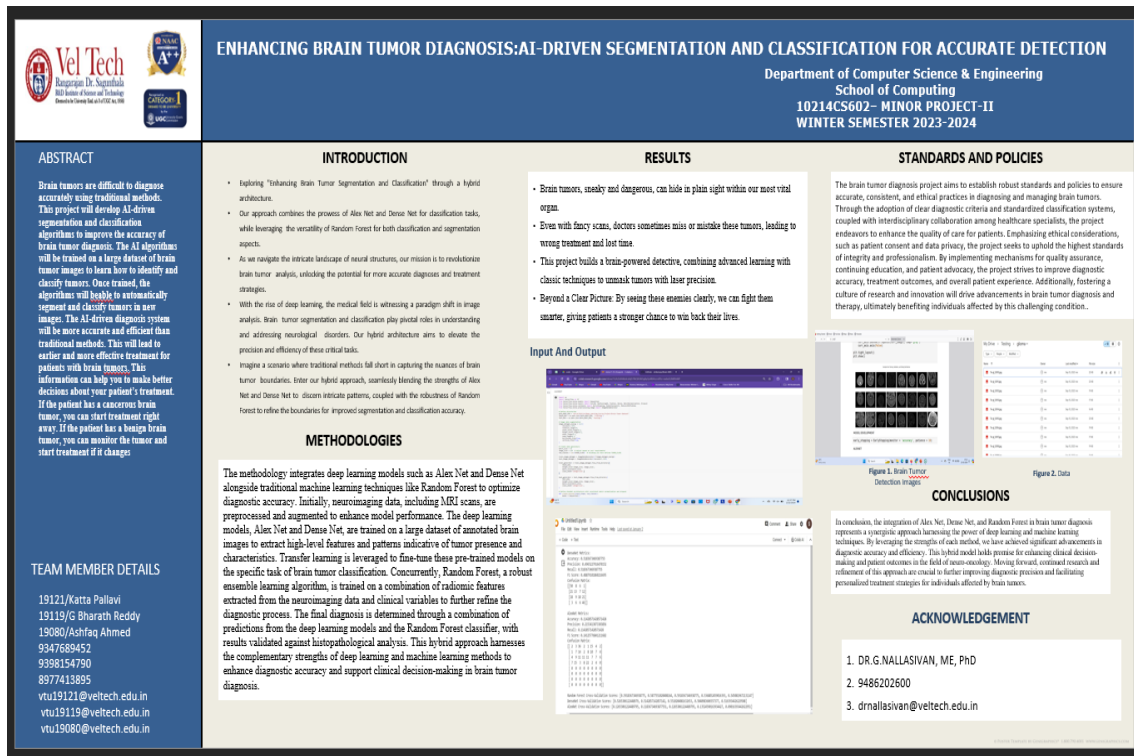


Figure 9.1: Poster Presentation

# References

- [1] AYESHA JABBAR et al. "Implemented Brain Tumor Detection and Multi-Grade Segmentation through Hybrid Caps-VGGNet Model", IEEE access, pp. 10.1099, 2017.
- [2] Abdulaziz Salamah Aljaloud et al. "Early Diagnosis of Brain Tumour MRI Images Using Hybrid Techniques between Deep and Machine Learning", Research Article , May 2022.
- [3] Angela Zhang et-al. "Brain Tumor Segmentation and Tractographic Feature Extraction from Structural MR Images for Overall Survival Prediction", Research Gate, November 2018.
- [4] Dinthisrang Daimary et al. "Identifying the cancerous brain tissues and labeling them automatically based on the tumor types", pp. 2419–2428, August 2019.
- [5] Feyza Altunbey Ozbay et al., (2023), conducted a comprehensive literature review on the detection of brain tumors using deep learning techniques. The authors discuss various studies that have been conducted in this field, highlighting the different methods and models proposed for brain tumor classification
- [6] Mohammed Rasool et al. "A Hybrid Deep Learning Model for Brain Tumour Classification. Brain tumours are classified using biopsy", Entropy 2022, 24, 799., June 2022.
- [7] Nagwa M. Aboelenein et al. "Propose a Hybrid Two-Track U-Net(HTTU-Net) architecture for brain tumor segmentation", IEEE access, pp.10.1109, June 2020.
- [8] Shakir Khan et al. "U-Net-Based Models towards Optimal MR Brain Image Segmentation", Article, pp.10.3390, May 2023.

- [9] T. Balamurugan et al. "Provides Brain Tumor Segmentation and Classifier. Early diagnosis of brain tumors is critical to enhancing patient survival and prospects.", Research Square, May 2022.
- [10] Zengxin Liu et al. "TransMVU Multi-view 2D U-Nets with transformer for brain tumour segmentation", IEEE access, pp.10-1049, January 2023.