

# BRAIN TUMOR DETECTION USING DEEP LEARNING

*Major project report submitted  
in partial fulfillment of the requirement for award of the degree of*

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

**By**

<b>ASUTOSH PANDA</b>	<b>(20UECS0075)</b>	<b>(VTU17911)</b>
<b>C.KAMALESH REDDY</b>	<b>(20UECS0182)</b>	<b>(VTU18416)</b>
<b>K.TRIVENI</b>	<b>(20UECS0457)</b>	<b>(VTU17313)</b>

*Under the guidance of  
Dr. M. KAVITHA, M.E., Ph.D  
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**

# **BRAIN TUMOR DETECTION USING DEEP LEARNING**

*Major project report submitted  
in partial fulfillment of the requirement for award of the degree of*

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

**By**

<b>ASUTOSH PANDA</b>	<b>(20UECS0075)</b>	<b>(VTU17911)</b>
<b>C.KAMALESH REDDY</b>	<b>(20UECS0182)</b>	<b>(VTU18416)</b>
<b>K.TRIVENI</b>	<b>(20UECS0457)</b>	<b>(VTU17313)</b>

*Under the guidance of  
Dr. M. KAVITHA, M.E., Ph.D  
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 “BRAIN TUMOR DETECTION USING DEEP LEARNING” by “ASUTOSH PANDA (20UECS0075), C.KAMALESH REDDY (20UECS0182), K.TRIVENI (20UECS0457)” 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.

ASUTOSH PANDA

Date:        /        /

C.KAMALESH REDDY

Date:        /        /

K.TRIVENI

Date:        /        /

# APPROVAL SHEET

This project report entitled BRAIN TUMOR DETECTION USING DEEP LEARNING by ASU-TOSH PANDA (20UECS0075), C.KAMALESH REDDY (20UECS0182), K.TRIVENI (20UECS04-57) is approved for the degree of B.Tech in Computer Science & Engineering.

**Examiners**

**Supervisor**

Dr. M. Kavitha, M.E., Ph.D, Professor.

**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. M. KAVITHA, M.E., Ph.D.**, for her cordial support, valuable information and guidance, she helped us in completing this project through various stages.

A special thanks to our **Project Coordinators Mr. V. ASHOK KUMAR, M.Tech., 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.

<b>ASUTOSH PANDA</b>	<b>(20UECS0075)</b>
<b>C.KAMALESH REDDY</b>	<b>(20UECS0182)</b>
<b>K.TRIVENI</b>	<b>(20UECS0457)</b>

## ABSTRACT

Brain tumor detection plays a crucial role in early diagnosis and subsequent treatment planning, significantly impacting patient outcomes. Using magnetic resonance imaging to manually segment brain tumors for cancer diagnosis is a complex, tedious and time consuming task. The accuracy and the robustness of brain tumor segmentation, are crucial for the diagnosis, treatment planning, and treatment outcome evaluation. Most automated brain tumor segmentation techniques are used manually developed functions. Traditional deep learning methods (such as convolutional neural networks) also require a large amount of annotated data for training, which is usually difficult to obtain in the medical field. Here, in existing system the CNN model is used for early detection of brain tumors. The proposed system outperformed other models, including ResNet-50, VGG-16, and Densenet-201, in identifying brain tumors using MRI images. It achieved the highest accuracy of 97.6% by utilizing both local and global contextual features, as well as adjusting contrast and brightness levels. Validation of these models in the data sets shows that the state-of-the-art models including VGG-16, ResNet-50, and Densenet-201 are used to precise detection and classification achieved on this multi coloured dataset prove the system's robustness. 250 MRI images are used to analyze the performance of various models using metrics such as accuracy, recall, precision, F1 score, and area under the curve. The VGG-16 model performance with high accuracy of 97.6% when compared to other models.

**Keywords:** Brain Tumor, Magnetic Resonance Imaging (MRI), Deep Learning, Artificial Intelligence, Convolutional Neural Network (CNN), Machine Learning, VGG, ResNet.

# LIST OF FIGURES

4.1	<b>Architecture Diagram</b>	13
4.2	<b>Data Flow Diagram</b>	14
4.3	<b>Use Case Diagram</b>	15
4.4	<b>Class Diagram</b>	16
4.5	<b>Sequence Diagram</b>	17
4.6	<b>Activity Diagram</b>	18
5.1	<b>MRI images</b>	23
5.2	<b>Brain Tumor Classification Result</b>	24
5.3	<b>Image Uploading</b>	25
5.4	<b>Image Selection</b>	25
5.5	<b>Tumor or Non Tumor Detection</b>	26
5.6	<b>Unit Testing</b>	28
5.7	<b>Integration Testing</b>	30
5.8	<b>System Testing</b>	31
6.1	<b>Random Sample Visualization</b>	41
6.2	<b>Model Training Progress</b>	42
6.3	<b>Densenet Training and Validation Lose Curve</b>	43
6.4	<b>ResNet Model Architecture Overview</b>	44
6.5	<b>Train-Test Split Visualization</b>	44
6.6	<b>Brain Tumor Classification Result</b>	45
6.7	<b>VGG-16 Training &amp; Validation Accuracy &amp; Lose Curve</b>	46
6.8	<b>Densenet-201 Training &amp; Validation Accuracy &amp; Lose Curve</b>	46
6.9	<b>ResNet-50 Training &amp; Validation Accuracy &amp; Lose Curve</b>	47
8.1	<b>Asutosh Offer Letter</b>	51
8.2	<b>Kamalesh Offer Letter</b>	51
8.3	<b>Triveni Offer Letter</b>	52
8.4	<b>Asutosh Internship Completion Certificate</b>	52
8.5	<b>Kamalesh Internship Completion certificate</b>	53
8.6	<b>Triveni Internship Completion certificate</b>	53
9.1	<b>Plagiarism Report</b>	54



<b>10.1 Poster Presentation</b>	<b>62</b>
---------------------------------	-----------

# LIST OF ACRONYMS AND ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
CT	Computed Tomography
MRI	Magnetic Resonance Imaging
ML	Machine Learning
DNN	Deep Neural Network
EHR	Electronic Health Records
GPU	Graphics processing unit
PACS	Picture Archiving and Communication Systems
TPU	Tensor Processing Units
VGG	Visual Geometry Group

# 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>viii</b>
<b>1 INTRODUCTION</b>	<b>1</b>
1.1 Introduction . . . . .	1
1.2 Aim of the project . . . . .	2
1.3 Project Domain . . . . .	2
1.4 Scope of the Project . . . . .	3
<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 . . . . .	10
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 . . . . .	11
<b>4 METHODOLOGY</b>	<b>13</b>
4.1 Architecture Diagram . . . . .	13
4.2 Design Phase . . . . .	14
4.2.1 Data Flow Diagram . . . . .	14
4.2.2 Use Case Diagram . . . . .	15
4.2.3 Class Diagram . . . . .	16

4.2.4	Sequence Diagram . . . . .	17
4.2.5	Activity Diagram . . . . .	18
4.3	Algorithm . . . . .	19
4.4	Module Description . . . . .	20
4.4.1	Data Collection and Preprocessing Module . . . . .	20
4.4.2	Architecture Selection Module . . . . .	20
4.4.3	Future Extraction Module . . . . .	21
4.5	Steps to execute the project . . . . .	21
4.5.1	Installation and Preprocessing . . . . .	21
4.5.2	Validation and Testing . . . . .	22
4.5.3	Deployment . . . . .	22
<b>5</b>	<b>IMPLEMENTATION AND TESTING</b>	<b>23</b>
5.1	Input and Output . . . . .	23
5.1.1	Input Design . . . . .	23
5.1.2	Output Design . . . . .	24
5.2	Testing . . . . .	26
5.3	Types of Testing . . . . .	26
5.3.1	Unit testing . . . . .	26
5.3.2	Integration testing . . . . .	28
5.3.3	System testing . . . . .	30
<b>6</b>	<b>RESULTS AND DISCUSSIONS</b>	<b>32</b>
6.1	Efficiency of the Proposed System . . . . .	32
6.2	Comparison of Existing and Proposed System . . . . .	33
6.3	Sample Code . . . . .	34
<b>7</b>	<b>CONCLUSION AND FUTURE ENHANCEMENTS</b>	<b>48</b>
7.1	Conclusion . . . . .	48
7.2	Future Enhancements . . . . .	48
<b>8</b>	<b>INDUSTRY DETAILS</b>	<b>50</b>
8.1	Industry name . . . . .	50
8.1.1	Duration of Internship (From Date - To Date) . . . . .	50
8.1.2	Duration of Internship in months . . . . .	50
8.1.3	Industry Address . . . . .	50

8.2	Internship offer letter . . . . .	51
8.3	Internship Completion certificate . . . . .	52
<b>9</b>	<b>PLAGIARISM REPORT</b>	<b>54</b>
<b>10</b>	<b>SOURCE CODE &amp; POSTER PRESENTATION</b>	<b>55</b>
10.1	Source Code . . . . .	55
10.2	Poster Presentation . . . . .	62
	<b>References</b>	<b>63</b>

# Chapter 1

## INTRODUCTION

### 1.1 Introduction

Tumors will have a huge effect on the brain. Brain cells are destroyed in the area affected by tumor gets and can cause brain collapse. The result of the tumor depends on the size and area affected in the brain. The Brain connected each and every part of the body together to make it perfect sense. If anything happens to the brain our whole system collapses. Some neurons in brain do not have capability to regenerate and there some neurons which stops regeneration as a person age. If the tumor is situated in any of that non- regenerative areas, a person might even lose one of his/her senses. Discovering the tumor at an early stage can save a person's life. Artificial Intelligence is revolutionizing Healthcare in many areas such as Disease Diagnosis with medical imaging, Surgical Robot, maximizing hospital efficiency.

Deep learning has been proven to be superior in detecting diseases from X-rays, MRI scans and CT scans which could significantly improve the speed and accuracy of diagnosis. Tumors are located and diagnosed through a very keen medical procedure. Magnetic Resonance Image (MRI) is one such process. We are going to train and validate our model on MRI. These images are sent into to model to train it to detect and locate brain tumor. Segmenting brain tumors in multi-modal imaging data is a challenging problem due to unpredictable shapes and sizes of tumors. Deep Neural Networks (DNNs) have already been applied to segmentation problems and have shown significant performance improvement compared to the previous methods.

Magnetic resonance imaging (MRI) is widely used medical technology for diagnosis of various tissue abnormalities, detection of tumors. The active development in the computerized medical image segmentation has played a vital role in scientific research. This helps the doctors to take necessary treatment in an easy manner with fast decision making. Brain tumor segmentation is a hot point in the research field

of Information technology with biomedical engineering.

## **1.2 Aim of the project**

The main aim of the project is to enhance the accuracy of brain tumor detection using deep learning techniques, surpassing the limitations of traditional methods. By automating the detection process with deep learning algorithms, we aim to streamline the diagnostic workflow, saving time and resources for healthcare professionals. The developed deep learning model should be scalable and adaptable to different healthcare settings, including hospitals, clinics, and research institutions.

Detecting brain tumors at early stages is crucial for effective treatment and improved patient outcomes. The project aims to develop a deep learning model that can accurately and efficiently detect brain tumors in medical images such as MRI scans. The primary goal is to build a model that can accurately identify the presence, location, and type of brain tumors with high reliability. This involves training the model on a large dataset of annotated medical images to ensure robust performance across different cases and conditions.

## **1.3 Project Domain**

Deep learning can be used to build a wide variety of applications, including image and speech recognition, natural language processing based systems, recommendation systems, fraud detection, autonomous vehicles, robotics, and medical diagnosis. With its ability to learn from large amounts of data and make accurate predictions, deep learning has the potential to transform many industries and fields.

Deep learning models are typically based on artificial neural networks (ANNs), which are computational models inspired by the biological neural networks in the human brain. These networks consist of interconnected nodes called neurons, organized into layers. Deep learning models are characterized by their depth, meaning they have multiple layers of neurons that enable them to learn hierarchical representations of data. One of the key advantages of deep learning is its ability to learn feature representations directly from raw data, without the need for manual feature

engineering. This is particularly beneficial for tasks involving unstructured data such as images, audio, and text.

## **1.4 Scope of the Project**

The main scope of the project is collecting and preparing diverse brain MRI datasets. Preprocessing steps such as noise reduction, image normalization, and augmentation will be applied to prepare the data for training. Our project will involve the design and implementation of deep learning models tailored for brain tumor detection. Our project aims to integrate the developed deep learning models into existing clinical workflows for brain tumor detection. This includes collaborating with healthcare professionals to validate the utility and usability of the models in real-world clinical settings.

Gathering a diverse dataset of medical images, such as MRI scans, CT scans. Preprocessing the data to standardize the format, resolution, and orientation of the images, as well as to remove noise and artifacts. Researching and selecting appropriate deep learning architectures for brain tumor detection, such as Convolutional Neural Networks. Developing and fine-tuning the chosen model architecture to achieve optimal performance on the specific task of tumor detection. Conducting rigorous clinical validation and testing of the developed system to assess its performance in real-world healthcare settings. Collaborating with medical experts, radiologists, and healthcare institutions to gather feedback and refine the system based on practical considerations and clinical insights.



## Chapter 2

# LITERATURE REVIEW

[1] Anaya-Isaza, A. et al, presented a study titled Data Augmentation and Transfer Learning for Brain Tumor Detection in Magnetic Resonance Imaging in the journal IEEE Access. This study focused on leveraging data augmentation and transfer learning techniques to improve brain tumor detection using magnetic resonance imaging. They highlighted the potential of data augmentation and transfer learning to enhance the performance of deep learning models for brain tumor detection. The study began by discussing the motivation behind using data augmentation and transfer learning techniques in the context of MRI-based brain tumor detection. The authors provided insights into the limitations of traditional approaches and the benefits offered by leveraging pre-trained models and synthetic data generation.

[2] Amin, J. et al, conducted a comprehensive survey on brain tumor detection and classification utilizing machine learning techniques. Published in Complex Intelligent Systems, their work provides an extensive overview of the application of machine learning in the domain of brain tumor detection and classification, addressing various methodologies, challenges, and opportunities. It discusses the methodologies employed, including feature extraction, feature selection, and classification algorithms, and evaluates their performance in terms of accuracy, sensitivity, and specificity. It also identifies future research directions aimed at overcoming these challenges and improving the effectiveness of machine learning-based approaches in this domain. By synthesizing existing knowledge and identifying areas for further research, their work contributes to the advancement of automated brain tumor diagnosis and treatment.

[3] Musallam, A.S. et al, introduced a novel convolutional neural network architecture tailored for the automatic detection of brain tumors in magnetic resonance imaging images. Their work, published in IEEE Access, presents a significant advancement in the field of medical image analysis by proposing an innovative CNN model designed specifically for accurate and efficient brain tumor detection. It in-

tegrates state-of-the-art convolutional layers, pooling layers, and other architectural components optimized for extracting relevant features from MRI images while minimizing computational complexity. It demonstrates the effectiveness of the new model in accurately detecting brain tumors across various datasets, showcasing superior performance compared to existing approaches.

[4] Qureshi, S.A. et al, presented a study titled "Intelligent Ultra-Light Deep Learning Model for Multi-Class Brain Tumor Detection" in the journal Applied Sciences. The study aimed to develop an efficient deep learning model for the detection of brain tumors across multiple classes. The authors provided insights into the development and evaluation of their proposed deep learning model. They introduced the concept of an "ultra-light" model, emphasizing its efficiency and effectiveness in detecting brain tumors. The model's architecture and design were tailored specifically for multi-class brain tumor detection, addressing the complexity and variability of tumor types.

[5] Ranjbarzadeh, R. et al, presented a comprehensive review on brain tumor segmentation utilizing deep learning techniques and an attention mechanism with MRI multi-modalities brain images. Their study, published in Scientific Reports, delved into the application of deep learning coupled with an attention mechanism for the segmentation of brain tumors, a crucial task in medical imaging analysis. The review explored the current landscape of brain tumor segmentation, addressing challenges and proposing opportunities for advancements in this domain. The review initiated by elucidating the methodology employed, emphasizing the utilization of deep learning algorithms and attention mechanisms tailored for analyzing MRI multi-modalities brain images. It discussed the role of attention mechanisms in enhancing the model's focus on relevant features and regions of interest in the image.

[6] Stadlbauer, A. et al, introduced the concept of "radiophysiomics" in the context of brain tumor classification, utilizing machine learning techniques and physiological MRI data. Their study, published in cancers, represents a significant advancement in the integration of physiological information with machine learning for improved brain tumor classification. The performance of the proposed radiophysiomics approach is evaluated through comprehensive experiments and comparative analyses, demonstrating its effectiveness in accurately classifying brain tumors into

different subtypes or grades. The study begins by highlighting the importance of incorporating physiological data, such as perfusion and metabolic information derived from MRI, in addition to traditional radiomic features, for a more comprehensive characterization of brain tumors.

[7] Soomro, T.A. et al, conducted a comprehensive review of image segmentation techniques used in MRI brain tumor detection, focusing on machine learning approaches. The authors discussed various methodologies, advancements, and challenges in the field of MR brain tumor detection, providing valuable insights for researchers and practitioners. They highlighted the importance of accurate image segmentation in enhancing the performance of machine learning models for brain tumor detection and emphasized the need for further research to address existing limitations and improve detection accuracy.

[8] Tiwari, P. et al, presented a study titled "CNN Based Multiclass Brain Tumor Detection Using Medical Imaging" in the journal Computational Intelligence and Neuroscience. This study focused on utilizing Convolutional Neural Networks for the detection of multiclass brain tumors using medical imaging data. They highlighted the significance of accurate and efficient tumor detection for early diagnosis and treatment planning. The study began by discussing the motivation behind using CNNs for multiclass brain tumor detection and their potential applications in clinical practice. The authors provided insights into the challenges associated with traditional methods and the advantages offered by deep learning approaches.

[9] Xie, Y. et al, conducted a thorough review on convolutional neural network (CNN) techniques for brain tumor classification spanning from 2015 to 2022. Their work, published in Diagnostics, provided insights into the evolution, challenges, and future prospects of employing CNNs for the classification of brain tumors, a critical task in medical imaging analysis. The review commenced by outlining the significance of accurate brain tumor classification for precise diagnosis and treatment planning. It emphasized the pivotal role of CNNs, a type of deep learning algorithm, in automating the classification process by extracting meaningful features from medical imaging data. It also proposed future perspectives for enhancing the effectiveness and reliability of CNN techniques in this domain, including the integration of multi-modalities imaging data and the development of explainable AI approaches.

[10] Zahoor, M.M. et al, introduced a novel approach titled “A New Deep Hybrid Boosted and Ensemble Learning Based Brain Tumor Analysis Using MRI” in the journal Sensors. This study aimed to leverage the combination of boosted and ensemble learning techniques within a deep learning framework for improved brain tumor analysis using MRI scans. The study commenced by discussing the motivation behind developing a hybrid boosted and ensemble learning based approach for brain tumor analysis. The authors emphasized the need for advanced computational methods to address the complexity and variability of brain tumor imaging data.

## Chapter 3

# PROJECT DESCRIPTION

### 3.1 Existing System

The existing system for brain tumor detection leverages deep learning techniques to address the limitations of existing methods. By employing convolutional neural networks (CNNs) and other advanced algorithms, the existing system offers several advantages over traditional approaches. Firstly, the use of deep learning enables automated feature extraction from medical images, eliminating the need for manual interpretation and reducing the risk of human error. This automation streamlines the diagnosis process, allowing for faster and more consistent results, which are crucial for timely treatment decisions and improved patient outcomes. These models are trained on vast datasets, enabling them to learn complex patterns and variations in imaging data, including those associated with different tumor types and anatomical structures. Additionally, the scalability of deep learning algorithms allows the existing system to handle large volumes of medical imaging data efficiently, making it suitable for population-level screening programs and facilitating the early detection of brain tumors in at risk populations.

#### **Disadvantages:**

- Requires extensive data annotation, potentially limiting accessibility.
- Demanding computational resources could restrict widespread implementation.
- Susceptible to overfitting, reducing generalization performance.
- Lack of interpretability may hinder clinical acceptance.

### 3.2 Proposed System

Brain tumor detection is a critical task in medical image analysis, essential for timely diagnosis and treatment planning. Deep learning techniques have shown

promising results in automating this process. In this proposed system, we leverage pretrained convolutional neural network (CNN) models, including ResNet-50, VGG-16, and DenseNet-201, to enhance the accuracy and efficiency of brain tumor detection from MRI images.

#### **Advantages:**

- Automated analysis saves time for doctors, letting them focus on treatment plans.
- Deep learning catches even tiny tumors, improving early detection rates.
- Quick results mean faster treatment decisions, potentially saving lives.
- It handles large datasets easily, making population-level screening feasible.
- Consistent performance across different hospitals ensures reliable diagnoses.

### **3.3 Feasibility Study**

In order to determine if the suggested brain tumour detection system is feasible, a number of implementation and deployment-related factors must be evaluated. First and foremost, it is critical to assess the accessibility and availability of the necessary resources, such as deep learning and medical imaging analytic knowledge, computational infrastructure, and datasets for medical imaging. This evaluation makes sure there won't be any major resource shortages that will hinder the project's ability to move forward. To guarantee that the project adheres to legal and ethical standards, other factors including regulatory compliance, ethical norms, and data protection requirements must be taken into account. In addition, a technical feasibility analysis is carried out to evaluate the viability of integrating the suggested system into clinical practice and the suitability of deep learning algorithms with current imaging equipment and workflows.

#### **3.3.1 Economic Feasibility**

The implementation of the brain tumour detection system is contingent upon a number of important elements. First off, training and validating the deep learning algorithms heavily depends on the quantity and calibre of medical imaging data. The system's ability to identify different kinds and stages of brain tumours depends on

having access to a sufficiently vast and diverse dataset. It is also necessary to carefully evaluate the computational resources needed for inference and training in order to guarantee timely processing of imaging data without sacrificing performance. To properly build, validate, and implement the system, the feasibility study also needs to take into account the experience and cooperation of interdisciplinary teams made up of radiologists, data scientists, and healthcare professionals. In addition, ethical principles and regulatory compliance are critical to guaranteeing the responsible and moral use of patient data in the creation and implementation of the brain tumour detection system.

### **3.3.2 Technical Feasibility**

In order to determine whether the brain tumour detection system is technically feasible, deep learning algorithms must be evaluated in relation to the current medical imaging processes and infrastructure. This entails assessing how well deep learning models can interpret and analyse different kinds of medical pictures, such as MRI and CT scans, which are frequently used to diagnose brain tumours. Furthermore, it is imperative to assess the scalability of the suggested system to effectively manage substantial amounts of imaging data, guaranteeing prompt processing and diagnosis. In addition, to ensure a smooth integration of the suggested system into clinical workflows, factors like integration with current Picture Archiving and Communication Systems (PACS) and Electronic Health Records (EHR) platforms must be taken into account.

### **3.3.3 Social Feasibility**

Within the framework of the brain tumour detection system, social feasibility refers to a range of socio-cultural elements that could impact the technology's uptake and acceptance in the medical field as well as in larger society. The acceptance and confidence that medical specialists, such as radiologists and neurologists, have in the precision and dependability of the deep learning algorithms utilised by the system is a crucial factor. By involving these parties in educational and training initiatives, worries can be allayed and trust in the technology can be increased. Furthermore, it is imperative to maintain open lines of communication regarding the advantages, constraints, and moral implications of the system in order to win over patients' families. In addition, it is imperative to attend to issues with patient consent, data protection,

and security in order to maintain credibility and adhere to legal obligations.

### **3.4 System Specification**

#### **3.4.1 Hardware Specification**

- High-performance computing hardware equipped with modern GPUs or TPUs to accelerate deep learning training and inference tasks.
- Sufficient RAM to accommodate large datasets and support concurrent processing of multiple medical images.
- Storage infrastructure capable of storing and managing vast volumes of medical imaging data, with provisions for scalability and data redundancy.
- Networking infrastructure to facilitate data transfer between imaging devices, storage systems, and computing resources, ensuring efficient data flow within the system.

#### **3.4.2 Software Specification**

- Deep learning frameworks such as TensorFlow or PyTorch for developing and implementing pre-trained models of convolutional neural networks and other deep learning models for brain tumor detection.
- Image processing libraries are used for pre-processing and augmenting medical imaging data to enhance the performance of deep learning algorithms.
- Deployment frameworks such as TensorFlow Serving for deploying trained models in production environments and integrating them into clinical workflows.
- Development tools for collaborative model development and version control, such as Git and GitHub, to facilitate collaboration among multidisciplinary teams working on the project.

#### **3.4.3 Standards and Policies**

- ISO 13485: This standard specifies requirements for a quality management system in the design, development, production, installation, and servicing of medical devices. Compliance with ISO 13485 ensures that the processes involved in



developing deep learning algorithms for brain tumor detection meet regulatory requirements and maintain product quality.

- **ISO 14971:** This standard outlines the application of risk management to medical devices, including the identification, analysis, and control of risks associated with the use of such devices. Adhering to ISO 14971 helps developers assess and mitigate potential risks related to the use of deep learning algorithms for brain tumor detection, ensuring patient safety.
- **ISO/IEC 27001:** While not specific to healthcare, this standard addresses information security management systems, including the protection of sensitive data. Compliance with ISO/IEC 27001 ensures that data collected and processed during brain tumor detection using deep learning is securely managed, stored, and transmitted, in accordance with regulatory requirements such as HIPAA.
- **Ethical Guidelines for AI in Healthcare:** Various organizations and institutions have developed ethical guidelines to address concerns surrounding the use of artificial intelligence in healthcare, ensuring fair and responsible deployment of deep learning technologies for brain tumor detection.
- **Data Security Standards:** Adherence to data security standards, such as ISO/IEC 27001, is essential to protect sensitive patient data collected during the process of brain tumor detection using deep learning. Implementing robust security measures helps prevent unauthorized access, data breaches, and other cybersecurity threats.

## Chapter 4

# METHODOLOGY

### 4.1 Architecture Diagram

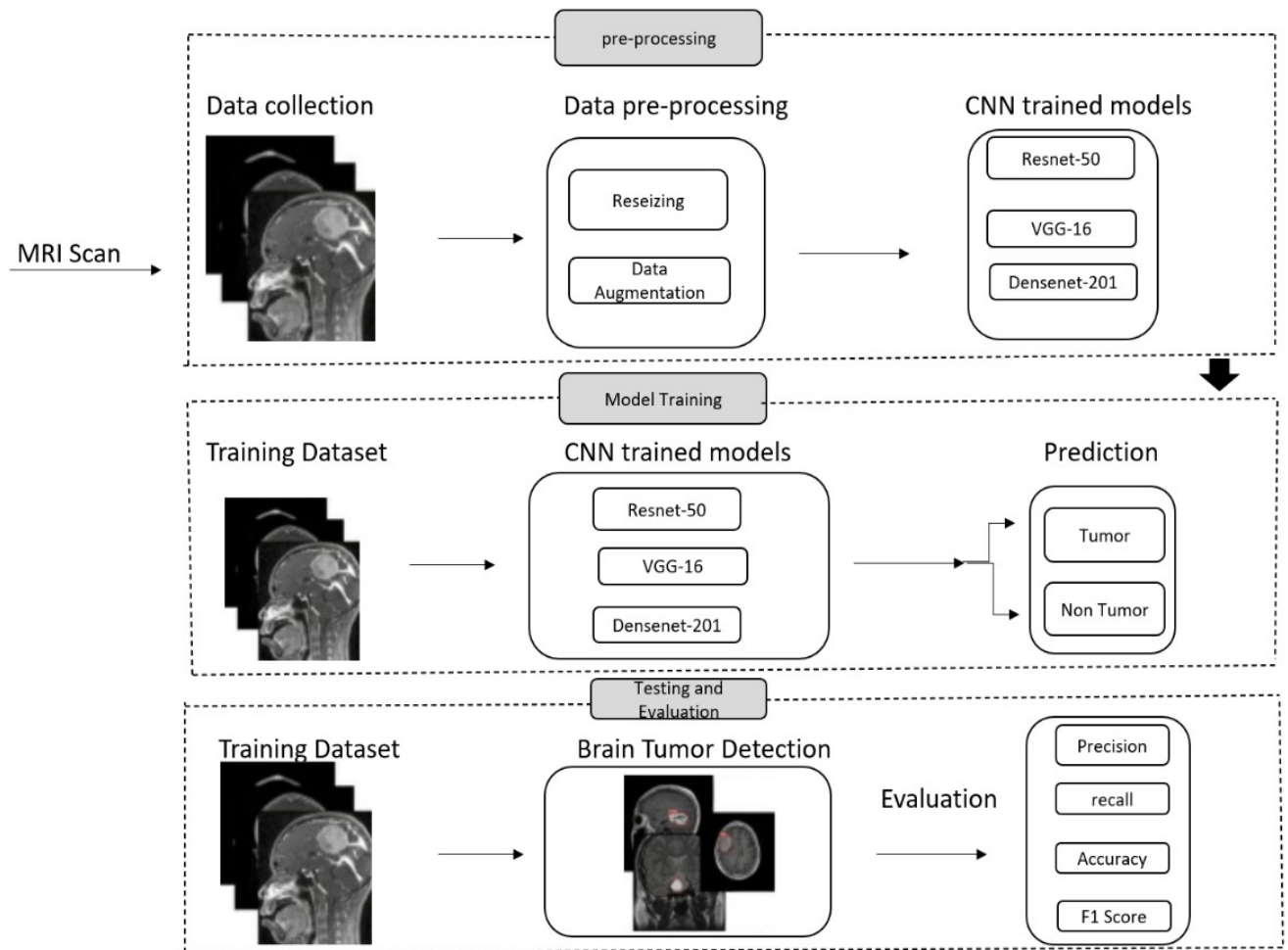


Figure 4.1: Architecture Diagram

Figure 4.1 shows the architecture diagram brain MRI images are collected from various sources, such as hospitals or medical imaging databases. Preprocessing steps may include resizing, normalization, and noise reduction to ensure consistency and improve data quality. The preprocessed MRI images are split into training, validation, and testing sets. A deep learning model, such as a pre-trained models of CNN such as resnet-50, VGG-16, and Densenet-201 is trained using the labeled MRI images. The

trained model is evaluated using the validation dataset to assess its performance metrics such as accuracy, recall, precision and F1 score.

## 4.2 Design Phase

### 4.2.1 Data Flow Diagram

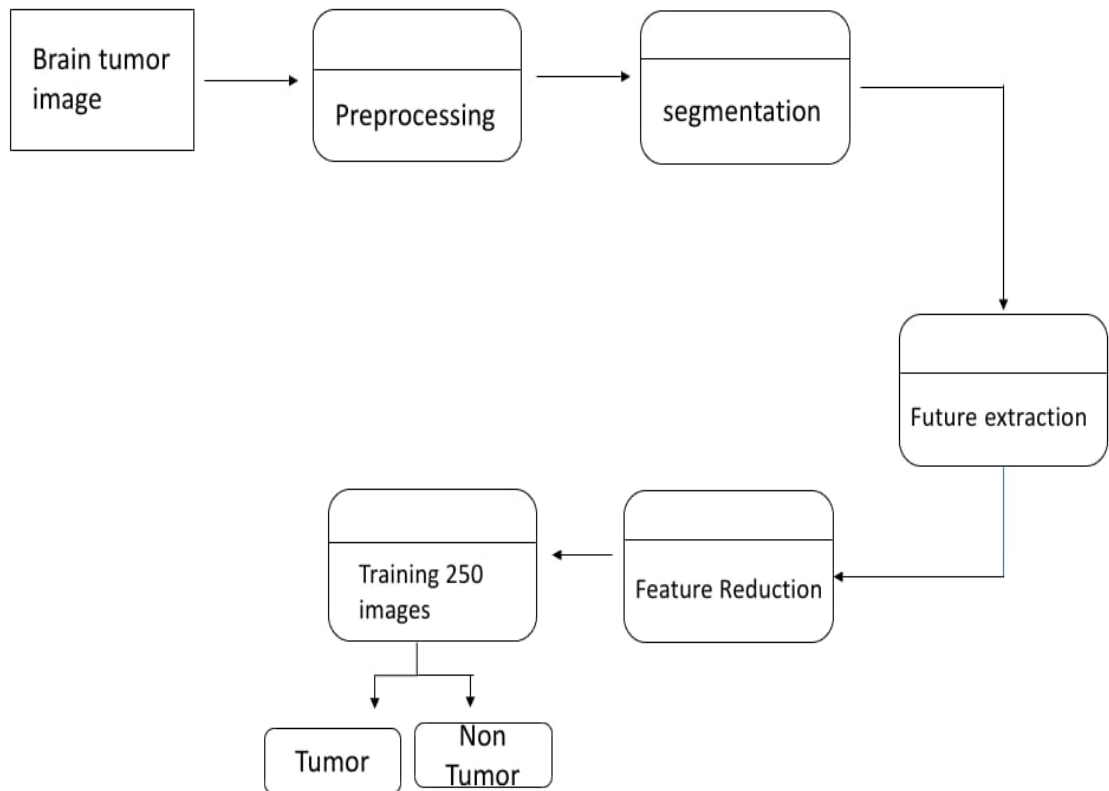


Figure 4.2: **Data Flow Diagram**

Figure 4.2 shows the data flow diagram brain MRI images are collected from various sources and undergo preprocessing to enhance quality and consistency. These preprocessed images, along with associated tumor labels or annotations, are used to train deep learning models. Model training involves adjusting model parameters to accurately differentiate between images with and without tumors. The trained model is then evaluated using validation data to assess its performance metrics. Upon satisfactory evaluation, the model is deployed for real-world use in clinical settings, where it performs inference on new MRI images to detect brain tumors.

#### 4.2.2 Use Case Diagram

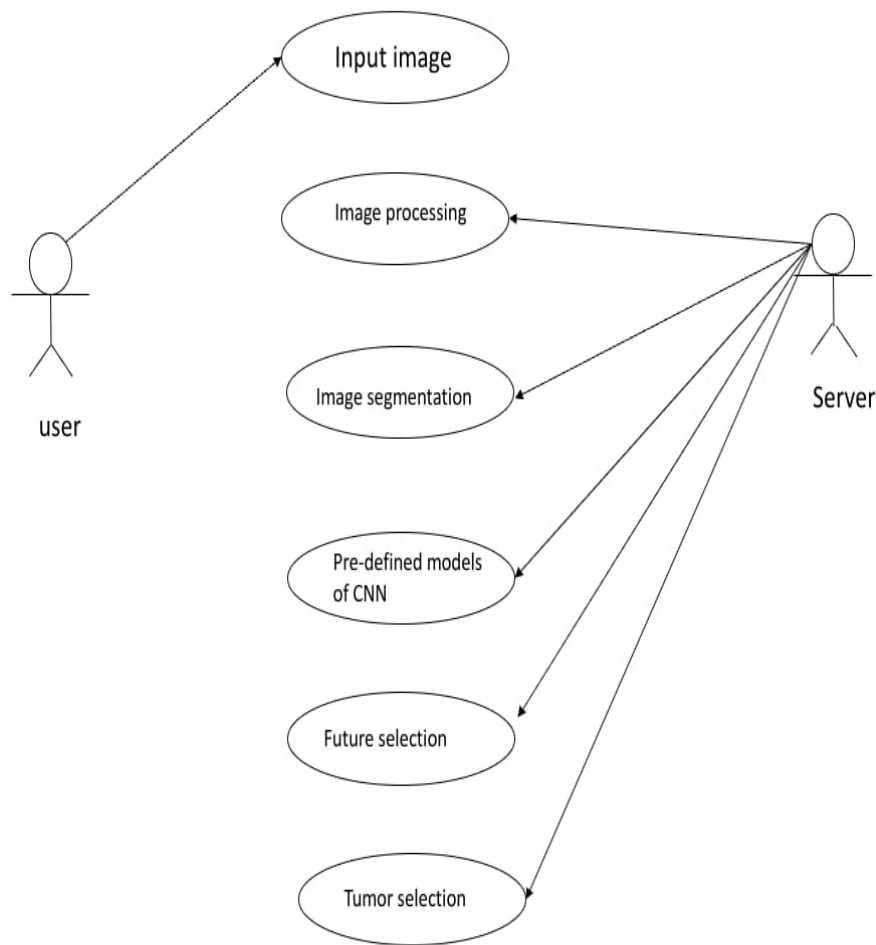


Figure 4.3: Use Case Diagram

Figure 4.3 shows the use case diagram for brain tumor detection using deep learning outlines the interactions between actors and the system. Healthcare professionals are the primary actors who interact with the system, providing input data in the form of brain MRI images and receiving output results for interpretation. The main use cases include Upload MRI Image where healthcare professionals upload MRI images to the system for analysis, View Results where they access and visualize the results generated by the deep learning model, and "Interpret Results" where they interpret the model's predictions to make clinical decisions. Additionally, there might be an Administrative System use case for system administrators to manage user accounts, monitor system performance, and ensure data security.

### 4.2.3 Class Diagram

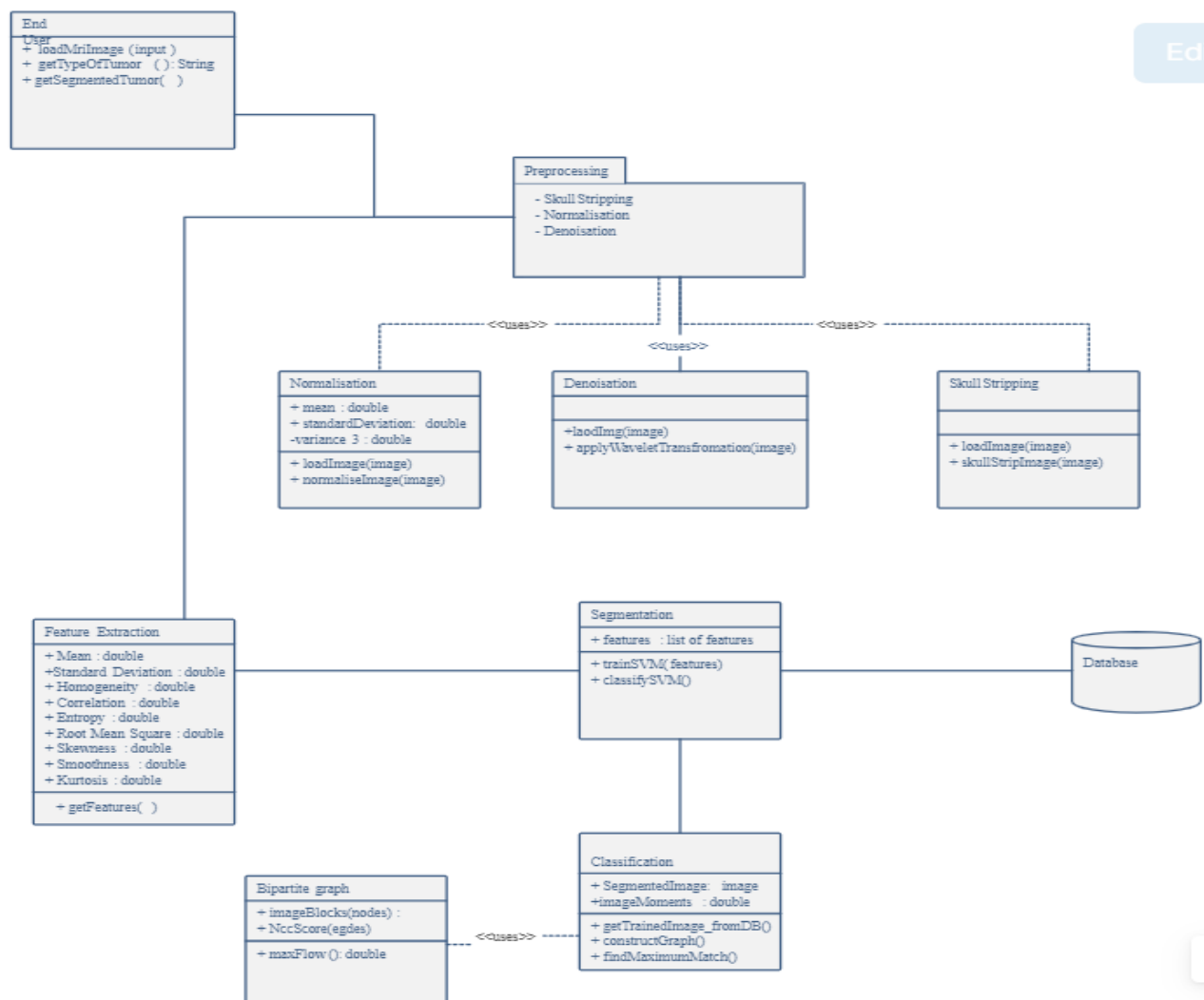


Figure 4.4: Class Diagram

Figure 4.4 shows the class diagram for brain tumor detection using deep learning outlining the key classes and their relationships within the system. At the core of the system is the Brain Tumor DetectionModel class, representing the deep learning model trained to detect brain tumors. This class encapsulates methods for model training, evaluation, and inference. The MRI Image class represents the brain MRI images used as input data for the model, with attributes such as image data and meta-data. Additionally, there might be a Label class representing tumor labels or annotations associated with MRI images for model training. The Healthcare Professional class represents users interacting with the system, with attributes such as username and password. The System Admin class may also exist for system administrators responsible for managing user accounts and system configuration.

#### 4.2.4 Sequence Diagram

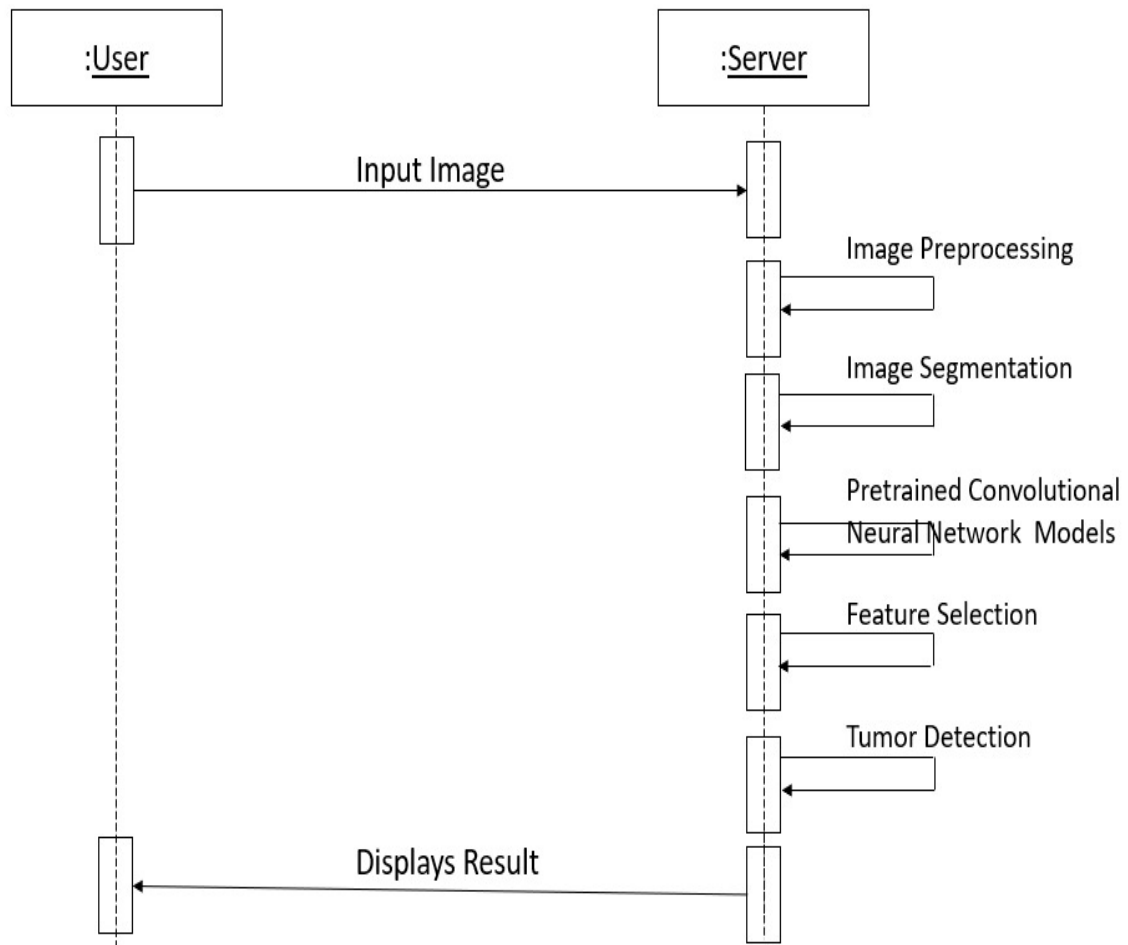


Figure 4.5: Sequence Diagram

Figure 4.5 shows the sequence diagram for brain tumor detection using deep learning illustrates the interactions and flow of messages between system components during the detection process. Initially, a healthcare professional uploads a brain MRI image to the system. Upon receiving the image, the system preprocesses it to enhance quality and consistency. Subsequently, the preprocessed image is passed to the deep learning model for inference, where the model analyzes the image and predicts whether a tumor is present or not. Once the inference process is complete, the system returns the prediction results to the healthcare professional, who can then visualize and interpret the results for clinical decision-making. This Sequence Diagram provides a chronological representation of the interactions between system components during the brain tumor detection process using deep learning.

#### 4.2.5 Activity Diagram

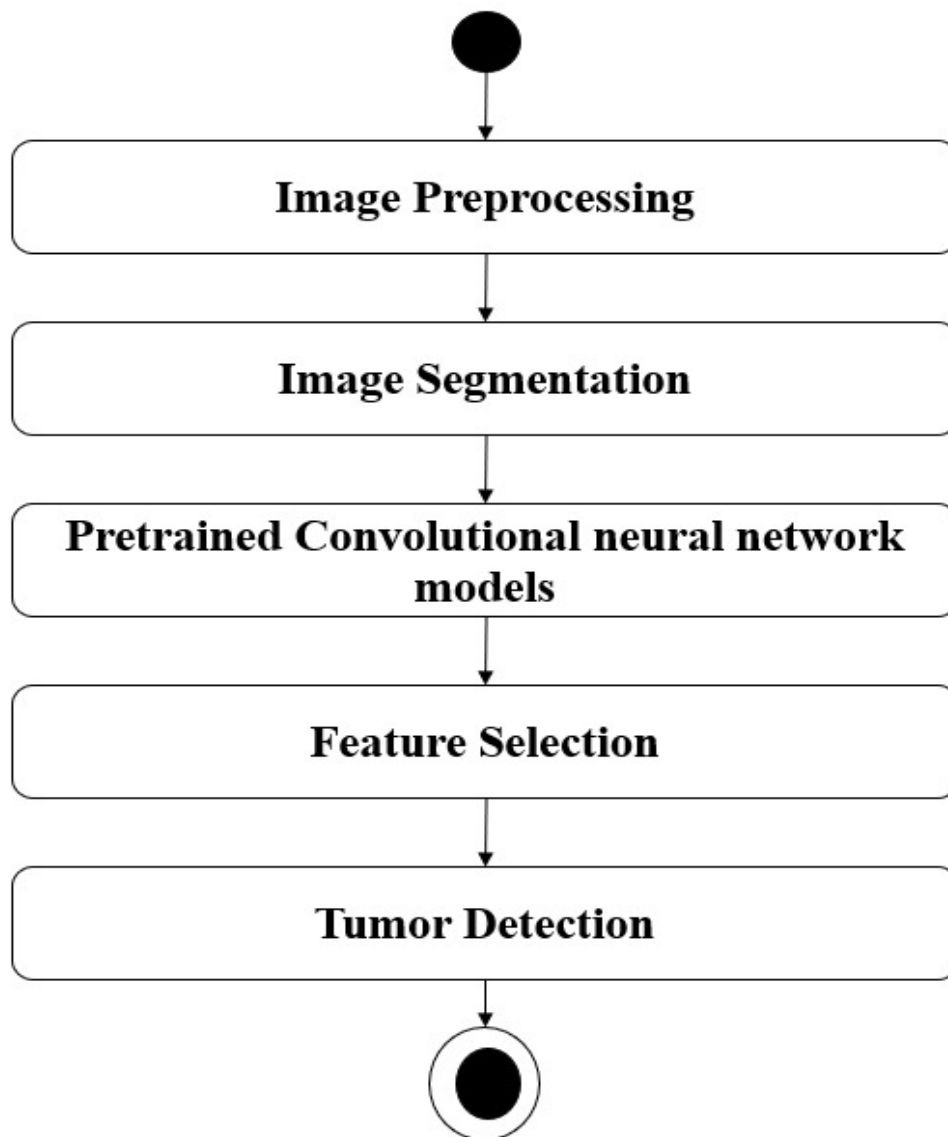


Figure 4.6: Activity Diagram

Figure 4.6 shows the activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control. In the preprocessing path, the system performs image preprocessing tasks such as resizing, normalization, and noise reduction to prepare the image for analysis. Simultaneously, in the model training path, the system trains a deep learning model using a dataset of labeled MRI images. Once both preprocessing and model training are complete, the process converges into the Inference activity, where the trained model analyzes new MRI images

to detect brain tumors. This Activity Diagram provides a visual representation of the sequential flow of activities and decision points involved in brain tumor detection using deep learning.

### 4.3 Algorithm

- **Step 1:**Collect a large dataset of brain MRI images, including images with tumors (positive samples) and without tumors (negative samples).
- **Step 2:**Preprocess the MRI images to enhance their quality and facilitate model training. Preprocessing steps may include resizing, normalization, intensity normalization, skull stripping, and noise reduction.
- **Step 3:**Divide the dataset into training, validation, and testing sets. Typically, around 70% of the data is used for training and 30% for testing.
- **Step 4:**Choose an appropriate deep learning architecture for brain tumor detection. Common architectures include pre trained models ofConvolutional Neural Networks (CNNs) like VGG-16,ResNet-50 and Densenet-201,custom designed architectures specifically tailored for medical image segmentation tasks.
- **Step 5:**Initialize the selected model with random weights.Train the model using the training dataset. During training, the model learns to differentiate between images with tumors and those without by adjusting its parameters based on the provided training data.
- **Step 6:**Validate the trained model using the validation dataset to assess its performance metrics such as accuracy, precision,recall and F1 score.Fine-tune hyperparameters based on validation performance to improve model performance.

$$Precision = \frac{TruePositive}{TruePositive + FaslePositive} \quad (4.1)$$

$$Accuracy = \frac{TruePositive + FaslePositive}{TruePositive + FasleNegative + FaslePositive + TrueNegative} \quad (4.2)$$

$$Recall = \frac{TruePositive}{TruePositive + FasleNegative} \quad (4.3)$$



$$F1score = \frac{Precision * Recall}{Precision + Recall} \quad (4.4)$$

- **Step 7:** Test the trained model on the testing dataset to evaluate its generalization performance on unseen data.
- **Step 8:** Deploy the trained model for real-world use in clinical settings, either as part of an existing medical imaging system or as a standalone application. Implement a user-friendly interface for healthcare professionals to upload MRI images, run inference, and visualize results.
- **Step 9:** Monitor the performance of the deployed model in real-world settings and collect feedback from healthcare professionals. Fine-tune the model periodically using additional data or retraining on updated datasets to maintain its effectiveness over time.

## 4.4 Module Description

### 4.4.1 Data Collection and Preprocessing Module

Gather a diverse and representative dataset of brain images, including both normal and tumor-affected cases. Collecting a dataset of brain MRI images with and without tumors. Preprocessing the images to ensure uniformity in size, resolution, and quality. Common preprocessing steps include resizing, normalization, and noise reduction. After preprocessing it gives standardized images. Augmenting the dataset to increase its size and variability. This helps prevent overfitting and improves the model's generalization ability. Augmentation techniques may include rotation, flipping, zooming, and shifting of images.

### 4.4.2 Architecture Selection Module

Choosing an appropriate deep-learning architecture for tumor detection. Convolutional Neural Networks are commonly used for image classification tasks like tumor detection. Popular pre-trained models architectures for image classification include ResNet-50, VGG-16 and Densenet-210. During training, the model learns to identify patterns and features associated with brain tumors. Validating the model's performance on the validation set to adjust hyperparameters and prevent overfitting. Optimizing the model architecture and hyperparameters to improve its perfor-

mance. Techniques like learning rate scheduling, weight regularization, and dropout regularization can help prevent overfitting and improve generalization.

#### 4.4.3 Future Extraction Module

Exploration of novel deep learning architectures specifically designed for medical image analysis, such as attention mechanisms, capsule networks, or transformer-based models. These architectures may capture more complex relationships within the data and improve performance. Utilization of self-supervised learning techniques to train deep learning models on large unlabeled datasets. Pretraining models on tasks such as image inpainting or image colorization could learn meaningful representations of medical images, facilitating better performance with limited labeled data.

### 4.5 Steps to execute the project

#### 4.5.1 Installation and Preprocessing

- **Install Required Dependencies:** Begin by installing the necessary software dependencies and libraries on your development environment. This typically includes deep learning frameworks like TensorFlow or PyTorch, as well as any additional libraries for data manipulation and visualization, such as NumPy, pandas, and Matplotlib.
- **Preprocess the Medical Imaging Data:** Prepare the medical imaging dataset for training the deep learning model. This involves tasks such as loading the data, normalizing pixel values, resizing images to a consistent size, and splitting the dataset into training, validation, and testing sets.
- **Implement the pre-trained models of CNN Architecture:** Develop the neural network architecture for the brain tumor detection model using a deep learning framework like TensorFlow or PyTorch. This involves defining the layers of the pre-trained models of convolutional neural network (CNN), including convolutional layers, pooling layers, fully connected layers, and activation functions.

#### 4.5.2 Validation and Testing

- **Split the Dataset:** Divide the preprocessed dataset into separate subsets for training, validation, and testing. The training set is used to train the model, the validation set is used to tune hyperparameters and monitor performance during training, and the testing set is used to evaluate the final performance of the trained model.
- **Train the Model:** Train the pre-trained model of CNN using the training dataset. During training, the model learns to make predictions by adjusting its parameters (weights and biases) based on the input data and corresponding ground truth labels. This process involves forward propagation, calculating the loss, and back-propagation to update the model parameters using an optimization algorithm like stochastic gradient descent or Adam.

#### 4.5.3 Deployment

- **Evaluate Model Performance:** Assess the performance of the trained model using the validation dataset. Evaluate metrics such as accuracy, precision, recall, and F1-score to measure how well the model generalizes to unseen data. Adjust hyperparameters and model architecture as needed based on validation performance.
- **Save the Trained Model:** Save the trained ResNet50, VGG-16 and Densenet-201 models to disk for future use. This allows you to reuse the trained model for inference on new medical imaging data without having to retrain it from scratch.
- **Deploy the Model for Inference:** Deploy the saved model for inference on new medical imaging data. This involves loading the saved model, preprocessing new imaging data, and feeding it through the model to make predictions. Deploy the model in a production environment, such as a healthcare facility, to assist radiologists in diagnosing brain tumors accurately and efficiently.

# Chapter 5

## IMPLEMENTATION AND TESTING

### 5.1 Input and Output

#### 5.1.1 Input Design

It encompasses the development of a user-friendly interface enabling healthcare professionals to upload brain MRI images for analysis. This interface should facilitate the seamless upload of single or multiple images in various formats (e.g., DICOM, JPEG), employing drag-and-drop functionality to streamline the process. Progress indicators are essential to keep users informed about the upload status. Additionally, users should be empowered to specify preprocessing settings such as resizing, normalization, and noise reduction, with the option to utilize default settings for simplicity. Augmentation options, including rotation, flipping, and scaling, should be available to augment the dataset for improved model generalization. Here, used 250 MRI images from the Kaggle. Visualization tools like image preview functionality and overlay annotations enable users to verify uploaded images correctness and visualize regions of interest, such as tumor regions, identified during preprocessing.

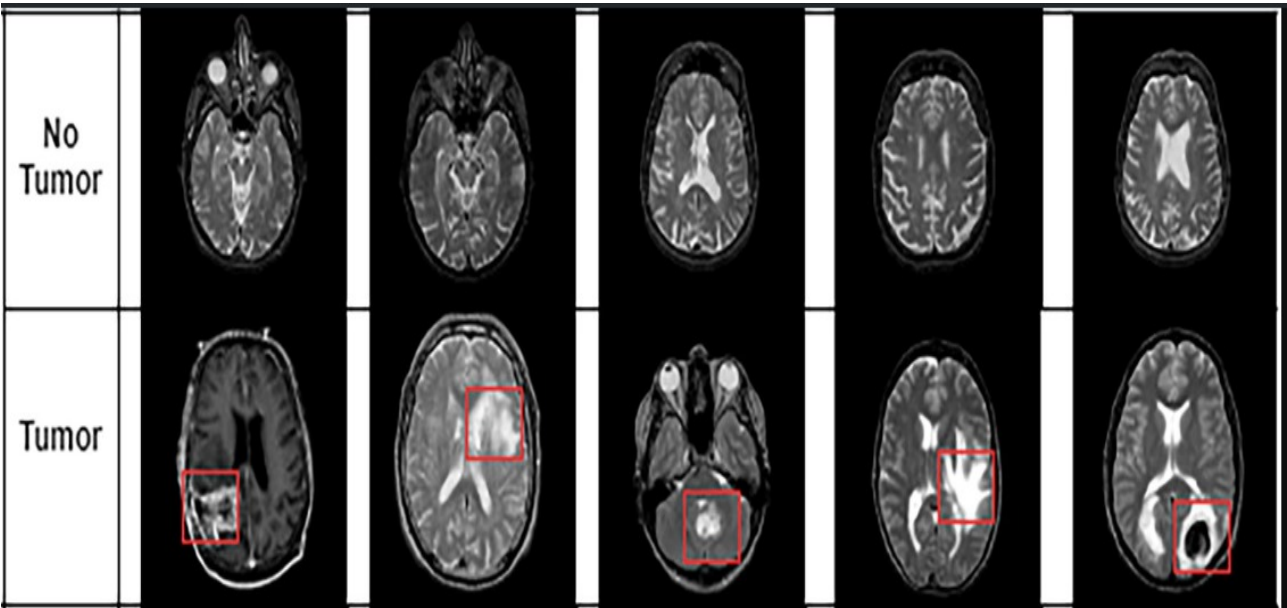


Figure 5.1: MRI images

### 5.1.2 Output Design

In output design we got 97.6% accuracy by using the Resnet50, VGG-16, Densenet-201 and it involves presenting the results of tumor detection to healthcare professionals in a comprehensive and actionable manner. This includes providing visual feedback on the presence or absence of tumors, accompanied by confidence scores to indicate the model's certainty in its predictions. Quantitative metrics such as accuracy, sensitivity, specificity, and false positive/negative rates offer insights into the model's performance, aiding in assessment and decision-making. Interactive tools such as zoom and pan functionalities allow users to explore MRI images in detail, while integration with clinical decision support systems provides access to treatment recommendations and relevant guidelines. Export options enable users to generate reports summarizing detection results for documentation and communication purposes, ensuring seamless integration with existing workflows and facilitating informed patient care.

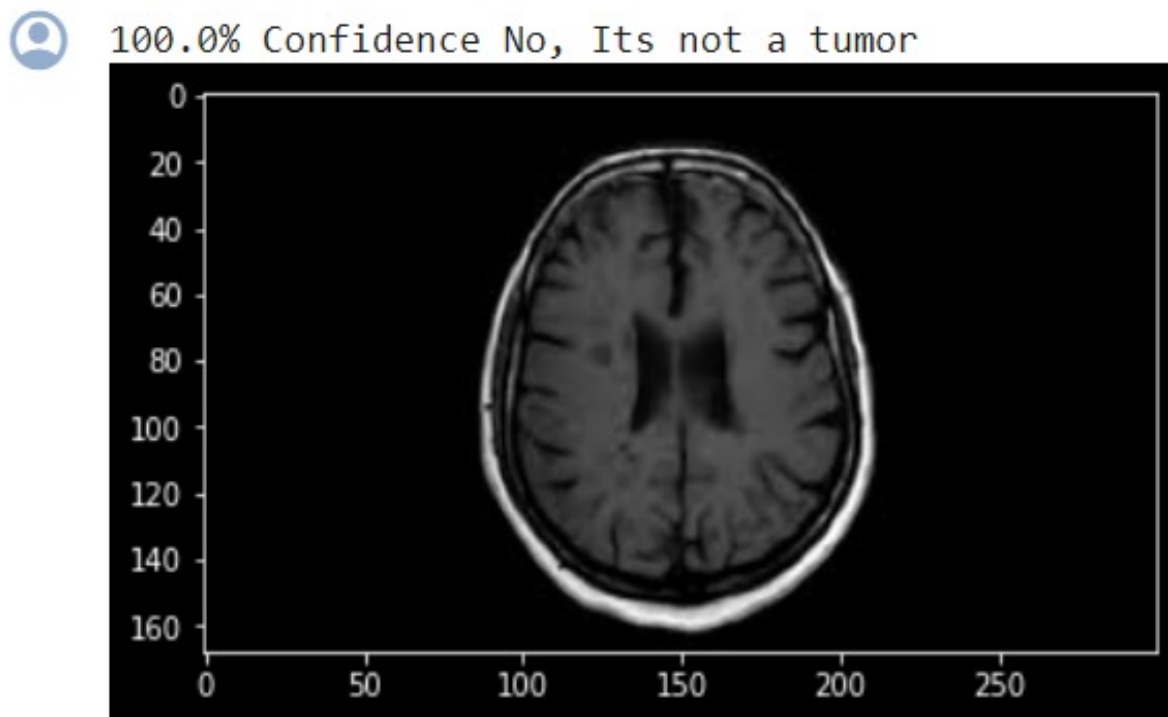


Figure 5.2: Brain Tumor Classification Result

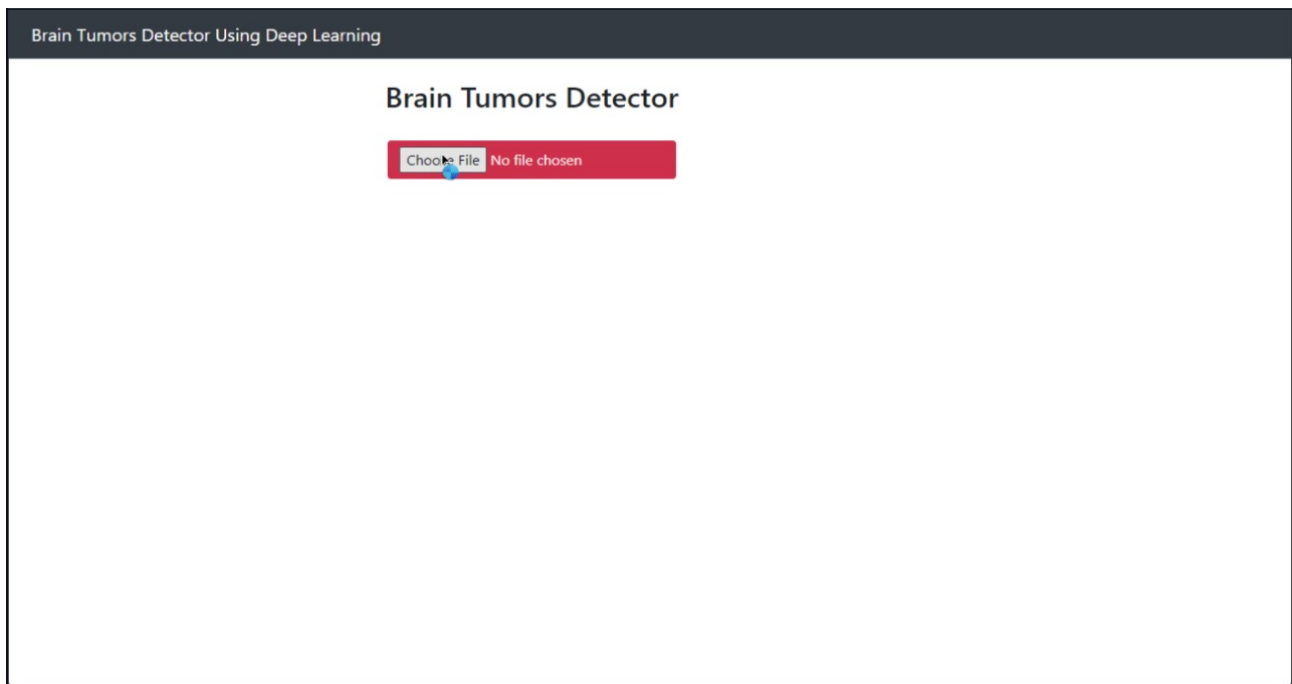


Figure 5.3: **Image Uploading**

The figure 5.3 illustrates the that how input is feeded or how to upload the image.

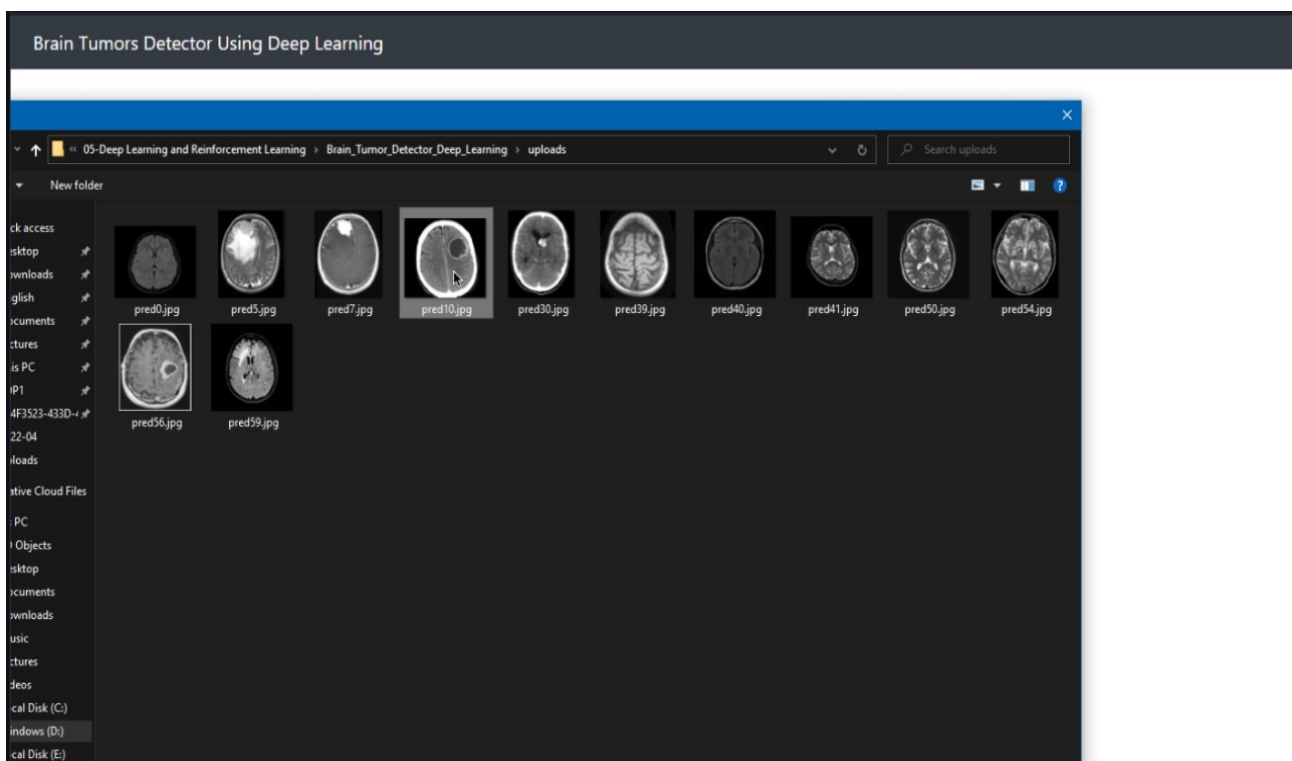
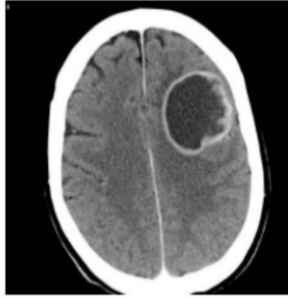


Figure 5.4: **Image Selection**

The figure 5.4 illustrates the that how input selection is to be done.

### Brain Tumors Detector

Choose File pred10.jpg



Result: A brain tumor is detected !

Figure 5.5: Tumor or Non Tumor Detection

The figure 5.5 illustrates the uploaded image is detected that image is tumor or nontumor image.

## 5.2 Testing

It involves a rigorous evaluation process to assess the system's performance, accuracy, and reliability. This includes various stages such as unit testing, integration testing, and validation testing. Continuous monitoring and refinement of the testing process are essential to ensure the system meets the highest standards of accuracy and reliability, ultimately enabling accurate and efficient detection of brain tumors.

## 5.3 Types of Testing

### 5.3.1 Unit testing

Unit testing in the context of the provided code for brain tumor detection involves writing tests to validate various components of the system. This includes ensuring the correctness and robustness of data loading and preprocessing functions, verifying the construction of the defined pre-trained models of CNN architecture, validating

the functionality of custom dataset classes such as MRI Dataset, and confirming the accuracy of the training process and model evaluation. Additionally, unit tests can be written to assess the model's prediction capabilities, especially if inference functionality is included.

## Input

```
1 import numpy as np # linear algebra
2 import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
3 import os
4 from matplotlib import pyplot as plt
5 %matplotlib inline
6 import tensorflow as tf
7 from tensorflow import keras
8 from keras import Sequential, models, layers
9 from tensorflow.keras.preprocessing import image
10 from keras.preprocessing.image import ImageDataGenerator
11 from numpy import asarray
12 import PIL
13 import cv2 from google.colab import drive
14 drive.mount('/content/drive') import torch
15 import torch.nn as nn
16 import torchvision.transforms as transforms
17 from PIL import Image
18 import gc
19 import os
20 dataset_path = '/content/drive/MyDrive/brain tumor data set'
21
22 paths = []
23 labels = []
24
25 for label in ['yes', 'no']:
26     for dirname, _, filenames in os.walk(os.path.join(dataset_path, label)):
27         for filename in filenames:
28             paths.append(os.path.join(dirname, filename))
29             labels.append(1 if label is 'yes' else 0)
30
31 len(paths), len(labels) sizes = []
32 for path in paths:
33     im = Image.open(path)
34     sizes.append(im.size)
35     im.close()
36
37 print(max(sizes), min(sizes))
```



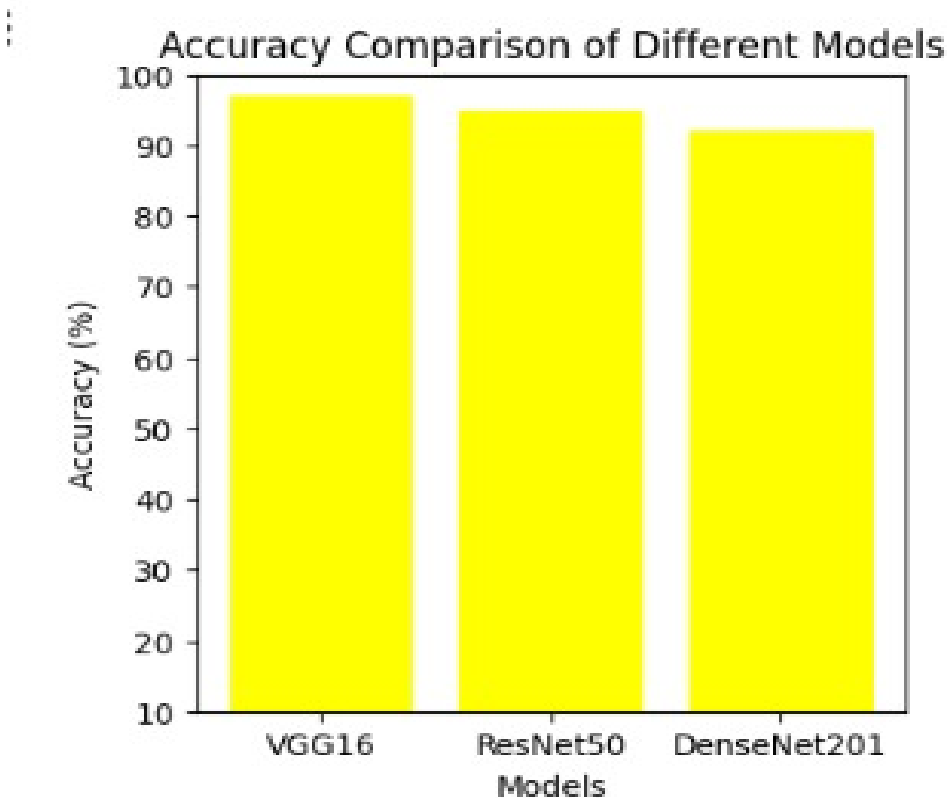


Figure 5.6: Unit Testing

### 5.3.2 Integration testing

Integration testing in the context of the provided code for brain tumor detection entails evaluating the interactions and interfaces between different components of the system. This involves testing how individual units or modules collaborate and function together as a cohesive whole. For instance, integration tests can verify that the data loading and preprocessing modules integrate seamlessly with the model architecture, ensuring that input data is correctly processed and fed into the neural network. Additionally, integration tests can validate the interaction between the training and evaluation components, ensuring that the model is trained effectively and evaluated accurately. Moreover, integration testing can assess the compatibility and interoperability of the system across different frameworks or libraries used, such as PyTorch for the model and TensorFlow for data preprocessing.

## Input

```
1 class Model(nn.Module):
2
3     def __init__(self, in_features=3):
4         super(Model, self).__init__()
5
6
7         self.conv_block = nn.Sequential(nn.Conv2d(in_channels=in_features,
8                                                     out_channels=32,
9                                                     kernel_size=3,
10                                                    stride=1
11                                                    ),
12                                           nn.ReLU(),
13                                           nn.MaxPool2d(2,2),
14
15                                           nn.Conv2d(in_channels=32,
16                                                       out_channels=64,
17                                                       kernel_size=3,
18                                                       stride=1
19                                                       ),
20                                           nn.ReLU(),
21                                           nn.MaxPool2d(2,2))
22
23         self.linear_block = nn.Sequential(nn.Linear(64*54*54, 1024),
24                                           nn.ReLU(),
25                                           nn.Dropout(0.5),
26                                           nn.Linear(1024,256),
27                                           nn.ReLU(),
28                                           nn.Dropout(0.3),
29                                           nn.Linear(256,1))
30
31     def forward(self, x):
32         x = self.conv_block(x)
33         x = torch.flatten(x,1)
34         x = self.linear_block(x)
35         return x
36
37 model = Model()
38 print(model)
```

## Test result

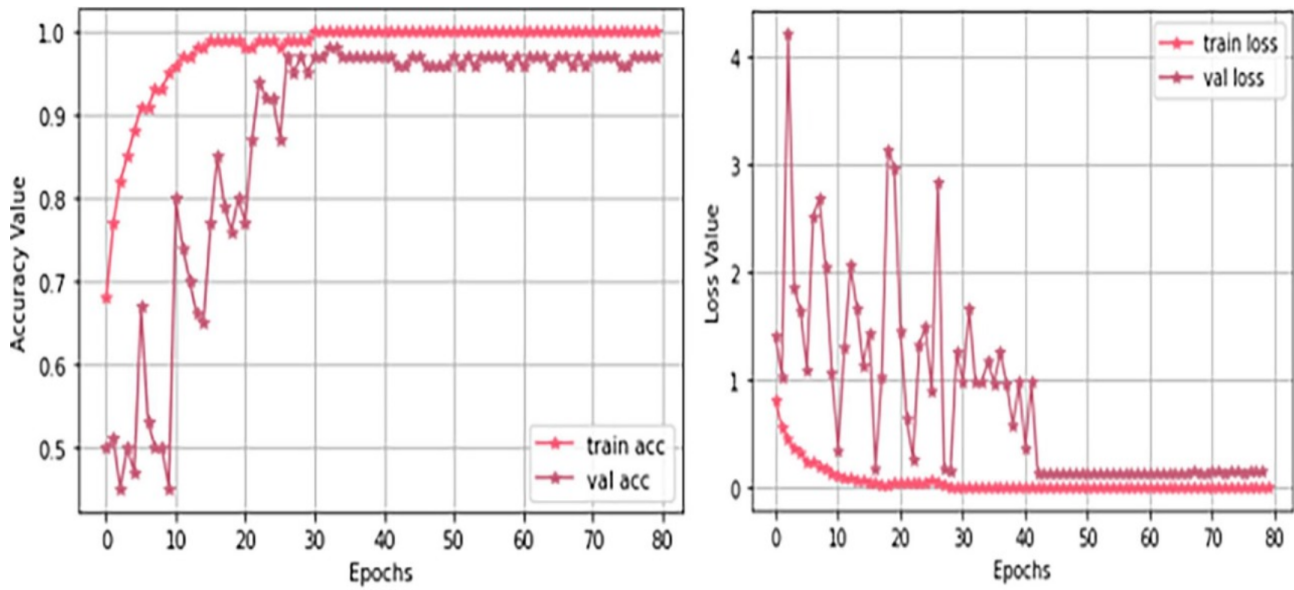


Figure 5.7: Integration Testing

### 5.3.3 System testing

System testing for brain tumor detection using deep learning involves evaluating the entire system as a whole to ensure it meets the specified requirements and functions correctly in real-world scenarios. This testing phase assesses the system's performance, accuracy, and reliability across different use cases and conditions. These tests aim to validate that the system behaves as expected, produces accurate tumor detection results, and handles various inputs and user interactions effectively.

## Input

```
1 import matplotlib.pyplot as plt
2
3 random_sample, random_label = test_dataset[0]
4 print(random_sample.shape)
5 plt.imshow(random_sample.permute(1,2,0))
6 print(random_label.item())
7 class Config:
8     learning_rate = 1e-3
9     epochs = 10
10    train_batch_size = 8
11    test_batch_size = 8
12 train_dataloader = torch.utils.data.DataLoader(train_dataset,
13                                                batch_size = Config.train_batch_size,
14                                                shuffle = True,
```

```

15         num_workers = 2)
16
17 test_dataloader = torch.utils.data.DataLoader(test_dataset,
18                                               batch_size = Config.test_batch_size,
19                                               shuffle = True,
20                                               num_workers = 2)
21 len(train_dataloader), len(test_dataloader)

```

## Test result

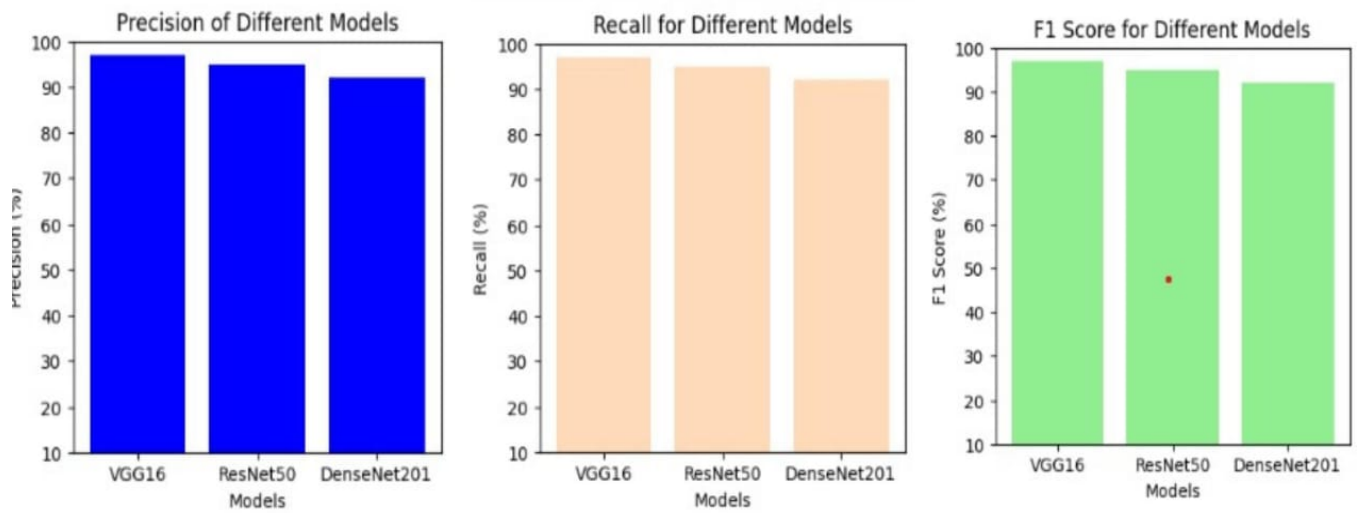


Figure 5.8: System Testing

## Chapter 6

# RESULTS AND DISCUSSIONS

### 6.1 Efficiency of the Proposed System

The proposed system for brain tumor detection using deep learning depends on several factors, including accuracy, speed, resource utilization, and usability. The primary goal of the system is to accurately detect brain tumors. Efficiency is closely tied to the system's ability to achieve high accuracy in tumor detection. A system with high accuracy minimizes false positives and false negatives, ensuring reliable and trustworthy results for healthcare professionals. Efficiency also involves the speed at which the system can analyze MRI images and provide detection results. A fast system reduces turnaround time, allowing for timely diagnosis and treatment planning. Real-time or near-real-time detection is desirable, especially in clinical settings where quick decisions are crucial for patient care. Efficient resource utilization is essential to optimize the performance of the system while minimizing computational resources, such as memory and processing power. A well-designed system should be able to leverage available resources effectively to achieve optimal performance without excessive resource consumption. The system's efficiency is also influenced by its scalability, i.e., its ability to handle increasing volumes of data and users without sacrificing performance. A scalable system can accommodate growing demands and adapt to changing requirements over time, ensuring continued efficiency as the workload increases.

Performance parameters	ResNet	Densenet	VGG16
Precision	96.6	94.6	97.4
Accuracy	96.5	94.1	97.6
Recall	96.8	94.7	97.7
F1-score	96.7	94.6	97.5

Table 6.1: Comparision of Different Algorithms Performance and Results

## 6.2 Comparison of Existing and Proposed System

### Existing system:

- The accuracy of the existing system may vary depending on the specific algorithms and techniques used. It might be limited by outdated methodologies or lack of access to state-of-the-art deep learning architectures.
- The speed of the existing system may be limited by computational resources, algorithm complexity, or outdated software implementations. Inefficient processing pipelines or lack of optimization can lead to longer processing times.
- The existing system may not fully optimize resource utilization, leading to inefficiencies in memory usage, processing power, or storage. Inefficient resource management can limit scalability and hinder performance.
- The usability of the existing system may vary depending on factors such as user interface design, documentation, and user support. Complex interfaces or lack of user-friendly features can hinder user adoption and productivity.
- The scalability of the existing system may be limited by architectural constraints, database design, or software architecture. Inefficient scaling mechanisms or lack of horizontal scalability can impede system growth.

### Proposed system:

- It aims to leverage advanced deep learning techniques and architectures, potentially leading to improved accuracy in tumor detection. By utilizing the latest research and methodologies, the proposed system can achieve higher accuracy rates and better performance.
- It can leverage optimizations, parallelization techniques, and hardware accelerators to enhance processing speed. Utilizing optimized algorithms and efficient implementations can significantly reduce processing times, enabling faster detection and diagnosis of brain tumors.
- The prioritize efficient resource utilization through optimization techniques, parallel processing, and distributed computing. By effectively leveraging available resources, the proposed system can maximize performance and scalability while minimizing resource overhead.

- The usability through intuitive interfaces, interactive visualization tools, and user-friendly workflows. By focusing on user experience design and usability testing, the proposed system can enhance user satisfaction and efficiency.
- The prioritize scalability through modular design, distributed computing, and cloud-based infrastructure. By adopting scalable architectures and technologies, the proposed system can accommodate growing data volumes and user loads while maintaining performance and reliability.

### 6.3 Sample Code

```

1 import numpy as np # linear algebra
2 import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
3 import os
4 from matplotlib import pyplot as plt
5 %matplotlib inline
6 import tensorflow as tf
7 from tensorflow import keras
8 from keras import Sequential, models, layers
9 from tensorflow.keras.preprocessing import image
10 from keras.preprocessing.image import ImageDataGenerator
11 from numpy import asarray
12 import PIL
13 import cv2 from google.colab import drive
14 drive.mount('/content/drive') import torch
15 import torch.nn as nn
16 import torchvision.transforms as transforms
17 from PIL import Image
18 import gc
19 import os
20 dataset_path = '/content/drive/MyDrive/brain tumor data set'
21
22 paths = []
23 labels = []
24
25 for label in ['yes', 'no']:
26     for dirname, _, filenames in os.walk(os.path.join(dataset_path, label)):
27         for filename in filenames:
28             paths.append(os.path.join(dirname, filename))
29             labels.append(1 if label is 'yes' else 0)
30
31 len(paths), len(labels) sizes = []
32 for path in paths:
33     im = Image.open(path)
34     sizes.append(im.size)
35     im.close()

```

```

36
37 print(max(sizes), min(sizes))
38 from sklearn.model_selection import train_test_split
39
40 X_train, X_test, y_train, y_test = train_test_split(paths,
41                                                    labels,
42                                                    stratify=labels,
43                                                    test_size=0.2,
44                                                    shuffle=True,
45                                                    random_state=135)
46
47 print(len(X_train), len(X_test))
48 class MRIDataset():
49     def _init_(self, paths, labels, augmentations=None):
50         self.paths = paths
51         self.labels = labels
52
53         if augmentations is None:
54             self.augmentations = transforms.Compose([ transforms.ToTensor() ])
55         else:
56             self.augmentations = augmentations
57
58     def _len_(self):
59         return len(self.paths)
60
61     def _getitem_(self, index):
62
63         label = self.labels[index]
64
65         sample = Image.open(self.paths[index]).convert(mode="RGB")
66         sample = self.augmentations(sample)
67
68         return (sample, torch.tensor(label, dtype=torch.float))
69 train_augmentations = transforms.Compose([ transforms.Resize((224,224)),
70                                           transforms.RandomHorizontalFlip(0.2),
71                                           transforms.RandomVerticalFlip(0.1),
72                                           transforms.RandomAutocontrast(0.2),
73                                           transforms.RandomAdjustSharpness(0.3),
74                                           transforms.ToTensor() ])
75
76 test_augmentations = transforms.Compose([ transforms.Resize((224,224)),
77                                           transforms.RandomHorizontalFlip(0.2),
78                                           transforms.RandomVerticalFlip(0.1),
79                                           transforms.RandomAutocontrast(0.2),
80                                           transforms.RandomAdjustSharpness(0.3),
81                                           transforms.ToTensor() ])
82 import matplotlib.pyplot as plt
83
84 random_sample, random_label = test_dataset[0]
85 print(random_sample.shape)

```



```

86 plt.imshow(random_sample.permute(1,2,0))
87 print(random_label.item())
88 class Config:
89     learning_rate = 1e-3
90     epochs = 10
91     train_batch_size = 8
92     test_batch_size = 8
93 train_dataloader = torch.utils.data.DataLoader(train_dataset ,
94                                                batch_size = Config.train_batch_size ,
95                                                shuffle = True ,
96                                                num_workers = 2
97                                                )
98
99 test_dataloader = torch.utils.data.DataLoader(test_dataset ,
100                                                batch_size = Config.test_batch_size ,
101                                                shuffle = True ,
102                                                num_workers = 2
103                                                )
104
105 len(train_dataloader), len(test_dataloader)
106 class Model(nn.Module):
107
108     def _init_(self, in_features=3):
109         super(Model, self)._init_()
110
111
112         self.conv_block = nn.Sequential(nn.Conv2d(in_channels=in_features ,
113                                                    out_channels=32,
114                                                    kernel_size=3,
115                                                    stride=1
116                                                    ),
117                                         nn.ReLU() ,
118                                         nn.MaxPool2d(2,2) ,
119
120                                         nn.Conv2d(in_channels=32,
121                                                    out_channels=64,
122                                                    kernel_size=3,
123                                                    stride=1
124                                                    ),
125                                         nn.ReLU() ,
126                                         nn.MaxPool2d(2,2)
127                                         )
128
129         self.linear_block = nn.Sequential(nn.Linear(64*54*54, 1024) ,
130                                           nn.ReLU() ,
131                                           nn.Dropout(0.5) ,
132                                           nn.Linear(1024,256) ,
133                                           nn.ReLU() ,
134                                           nn.Dropout(0.3) ,
135                                           nn.Linear(256,1)

```

```

136         )
137
138
139     def forward(self, x):
140         x = self.conv_block(x)
141         x = torch.flatten(x, 1)
142         x = self.linear_block(x)
143         return x
144
145 model = Model()
146 print(model)
147 device = "cuda" if torch.cuda.is_available() else "cpu"
148 model = model.to(device)
149
150 class Trainer:
151
152     def __init__(self, model, dataloaders, Config):
153         self.model = model
154         self.train, self.test = dataloaders
155
156         self.Config = Config
157
158         self.optim = torch.optim.Adam(self.model.parameters(), lr=self.Config.learning_rate)
159         self.loss_fn = nn.BCEWithLogitsLoss()
160
161     def binary_accuracy(self, outputs, labels):
162         return (torch.round(torch.sigmoid(outputs)) == labels).sum().item() / labels.shape[0]
163
164     def train_one_epoch(self):
165
166         running_loss = 0
167         running_acc = 0
168
169         for X, y in self.train:
170
171             X = X.to(device, dtype = torch.float)
172             y = y.reshape(y.shape[0], 1)
173             y = y.to(device, dtype = torch.float)
174
175             self.optim.zero_grad()
176
177             outputs = self.model(X)
178             loss = self.loss_fn(outputs, y)
179
180             loss.backward()
181             self.optim.step()
182
183             running_loss += loss.item()
184             running_acc += self.binary_accuracy(outputs, y)
185

```

```

186         del X
187         del y
188         gc.collect()
189         torch.cuda.empty_cache()
190
191     train_loss = running_loss / len(self.train)
192     train_acc = running_acc / len(self.train)
193
194     return train_loss, train_acc
195
196
197 def fit(self):
198
199     losses = []
200     accuracies = []
201
202     for epoch in range(self.Config.epochs):
203
204         self.model.train()
205
206         train_loss, train_acc = self.train_one_epoch()
207         losses.append(train_loss)
208         accuracies.append(train_acc)
209
210         print(f"EPOCH {epoch+1}/{self.Config.epochs}")
211         print(f"Training Loss: {train_loss} | Training Accuracy: {train_acc}\n\n")
212
213
214 @torch.no_grad()
215 def inference(self):
216
217     self.model.eval()
218
219     running_acc = 0
220
221     for X,y in self.test:
222
223         X = X.to(device, torch.float)
224         y = y.reshape(y.shape[0],1)
225         y = y.to(device, dtype = torch.float)
226         outputs = self.model(X)
227         running_acc += self.binary_accuracy(outputs, y)
228
229         del X
230         del y
231         gc.collect()
232         torch.cuda.empty_cache()
233
234     accuracy = running_acc / len(self.test)
235

```

```

236         return accuracy
237     trainer = Trainer(model, (train_dataloader, test_dataloader), Config)
238     trainer.fit()
239     trainer = Trainer(model, (train_dataloader, test_dataloader), Config)
240     trainer.fit()
241     accuracy = trainer.inference()
242     accuracy*100
243     import keras
244     from keras.models import Sequential
245     from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization
246     from PIL import Image
247     plt.style.use('dark_background')
248     from sklearn.model_selection import train_test_split
249     from sklearn.preprocessing import OneHotEncoder
250     encoder = OneHotEncoder()
251     encoder.fit([[0], [1]])
252     data = []
253     paths = []
254     result = []
255
256     for r, d, f in os.walk(r'/content/drive/MyDrive/brain tumor data set/yes'):
257         for file in f:
258             if '.jpg' in file:
259                 paths.append(os.path.join(r, file))
260
261     for path in paths:
262         img = Image.open(path)
263         img = img.resize((128,128))
264         img = np.array(img)
265         if(img.shape == (128,128,3)):
266             data.append(np.array(img))
267             result.append(encoder.transform([[0]]).toarray())
268     print(result)
269     paths = []
270     for r, d, f in os.walk(r"/content/drive/MyDrive/brain tumor data set/no"):
271         for file in f:
272             if '.jpg' in file:
273                 paths.append(os.path.join(r, file))
274
275     for path in paths:
276         img = Image.open(path)
277         img = img.resize((128,128))
278         img = np.array(img)
279         if(img.shape == (128,128,3)):
280             data.append(np.array(img))
281             result.append(encoder.transform([[1]]).toarray())
282     data = np.array(data)
283     data.shape
284     result = np.array(result)
285     result = result.reshape(139,2)

```

```

286 x_train , x_test , y_train , y_test = train_test_split(data , result , test_size=0.2, shuffle=True ,
    random_state=0)
287 model = Sequential()
288
289 model.add(Conv2D(32, kernel_size=(2, 2), input_shape=(128, 128, 3), padding = 'Same'))
290 model.add(Conv2D(32, kernel_size=(2, 2), activation = 'relu', padding = 'Same'))
291 model.add( BatchNormalization())
292 model.add( MaxPooling2D( pool_size=(2, 2)))
293 model.add( Dropout(0.25))
294 model.add(Conv2D(64, kernel_size = (2,2), activation = 'relu', padding = 'Same'))
295 model.add(Conv2D(64, kernel_size = (2,2), activation = 'relu', padding = 'Same'))
296 model.add( BatchNormalization())
297 model.add( MaxPooling2D( pool_size=(2,2), strides=(2,2)))
298 model.add( Dropout(0.25))
299 model.add( Flatten())
300 model.add( Dense(512, activation='relu'))
301 model.add( Dropout(0.5))
302 model.add( Dense(2, activation='softmax'))
303 model.compile(loss = "categorical_crossentropy", optimizer='Adamax')
304 print(model.summary())
305 y_train.shape
306 history = model.fit(x_train , y_train , epochs = 5, batch_size = 40, verbose = 1, validation_data = (
    x_test , y_test))
307 plt.plot(history.history[ 'loss' ])
308 plt.plot(history.history[ 'val_loss' ])
309 plt.title( 'Model Loss' )
310 plt.ylabel( 'Loss' )
311 plt.xlabel( 'Epoch' )
312 plt.legend([ 'Test' , 'Validation' ], loc='upper right')
313 plt.show()
314 def names(number):
315     if number==0:
316         return 'Its a Tumor'
317     else:
318         return 'No, Its not a tumor'
319 from matplotlib.pyplot import imshow
320 img = Image.open(r"/content/drive/MyDrive/brain tumor data set/no/11 no.jpg")
321 x = np.array(img.resize((128,128)))
322 x = x.reshape(1,128,128,3)
323 res = model.predict_on_batch(x)
324 classification = np.where(res == np.amax(res))[1][0]
325 imshow(img)
326 print(str(res[0][ classification ]*100) + '% Confidence ' + names(classification))

```

## Output

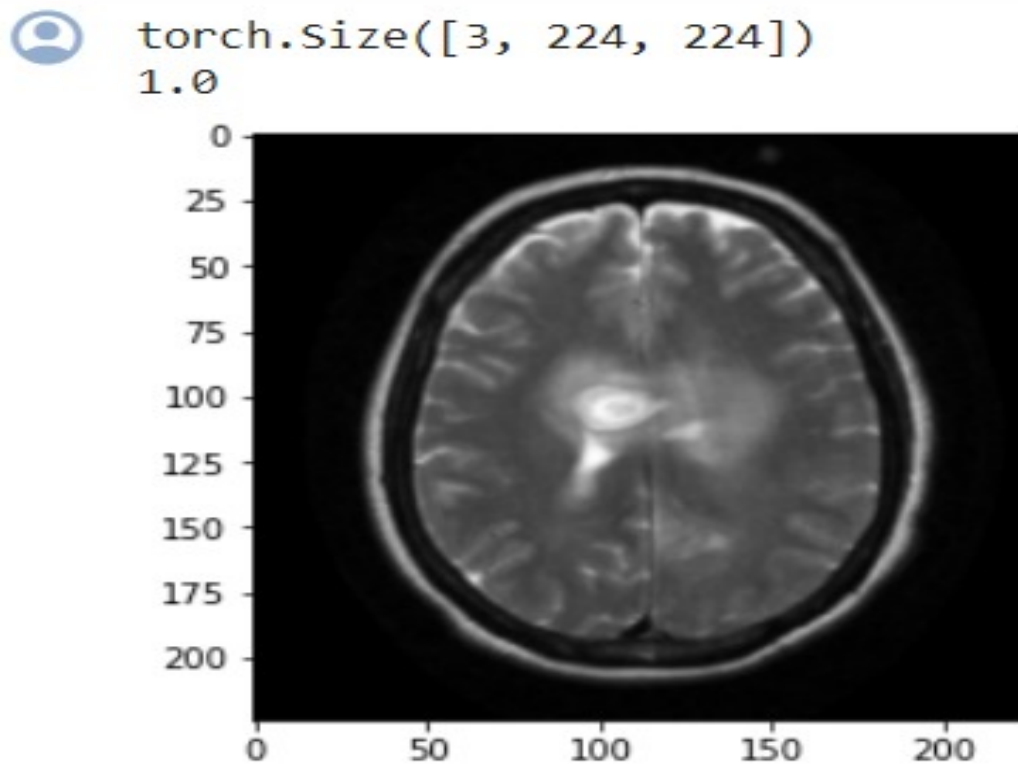


Figure 6.1: **Random Sample Visualization**

As shown in figure 6.1 the sample's shape, depicted as `[3, 224, 224]`, indicates a 3-channel image with dimensions 224x224. The label associated with the sample, printed as "1," signifies its classification or category.

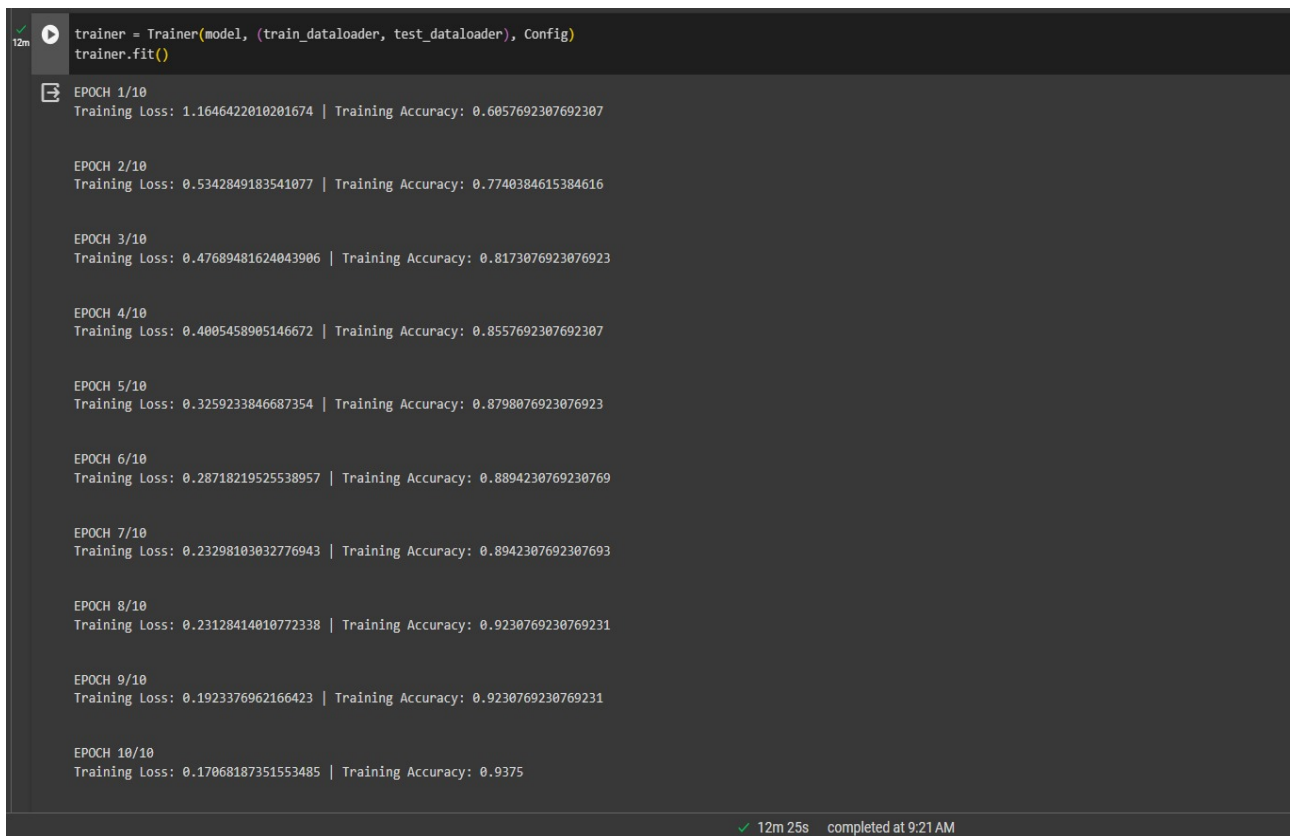


Figure 6.2: Model Training Progress

As shown in figure 6.2 the figure illustrates the training progress of the neural network model over multiple epochs. The y-axis represents the loss value, which quantifies the disparity between the model's predictions and the actual labels in the training data. A lower loss value indicates better alignment between predictions and ground truth labels. The x-axis corresponds to the number of training epochs, with each epoch representing a complete pass through the entire training dataset. By monitoring the training loss curve, insights can be gained into the model's learning dynamics, convergence behavior, and overall performance during the training process.

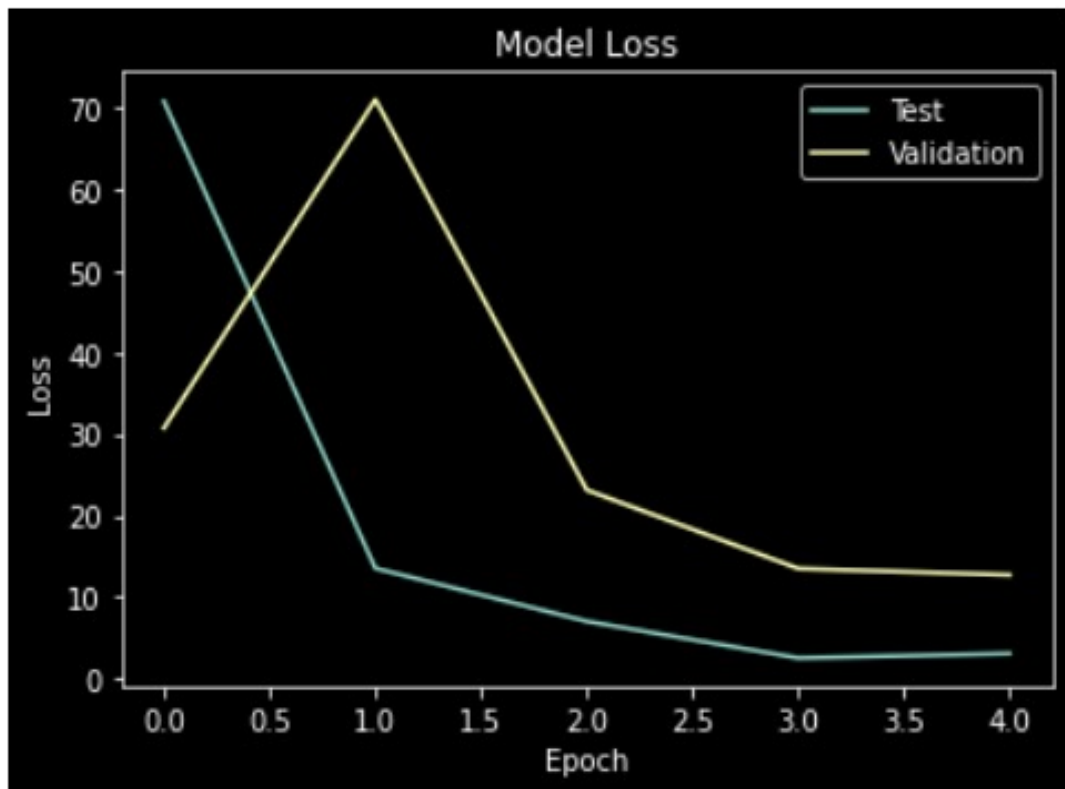


Figure 6.3: Densenet Training and Validation Lose Curve

As shown in figure 6.3 the densenet training and validation loss curves visualize the convergence and generalization of our deep learning model for brain tumor detection. They showcase the model's learning progression over epochs, ensuring both efficient training and robust performance on unseen data. These curves reflect the optimization process, demonstrating the reduction of training and validation errors over iterations. A smooth decline in loss signifies effective learning and model refinement, crucial for accurate tumor identification in medical imaging.



Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 128, 128, 32)	416
conv2d_1 (Conv2D)	(None, 128, 128, 32)	4128
batch_normalization (Batch Normalization)	(None, 128, 128, 32)	128
max_pooling2d (MaxPooling2D)	(None, 64, 64, 32)	0
dropout (Dropout)	(None, 64, 64, 32)	0
conv2d_2 (Conv2D)	(None, 64, 64, 64)	8256
conv2d_3 (Conv2D)	(None, 64, 64, 64)	16448
batch_normalization_1 (Batch Normalization)	(None, 64, 64, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 32, 32, 64)	0
dropout_1 (Dropout)	(None, 32, 32, 64)	0
flatten (Flatten)	(None, 65536)	0
dense (Dense)	(None, 512)	33554944
dropout_2 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 2)	1026

---

Total params: 33,585,602  
Trainable params: 33,585,410  
Non-trainable params: 192

Figure 6.4: ResNet Model Architecture Overview

As shown in figure 6.4 the ResNet model architecture offers a comprehensive overview of the intricate neural network design tailored for brain tumor detection. Combining convolutional layers for feature extraction with dense layers for classification, it efficiently processes medical imaging data, enabling precise identification of tumor regions while minimizing false positives.

```

✓ 1s ▶ from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(paths,
                                                    labels,
                                                    stratify=labels,
                                                    test_size=0.2,
                                                    shuffle=True,
                                                    random_state=1357
                                                    )

print(len(X_train), len(X_test))

```

202 51

Figure 6.5: Train-Test Split Visualization

As shown in figure 6.5 the Train-Test Split Visualization illustrates the division of our dataset into training and testing subsets, crucial for evaluating the performance and generalization capabilities of our brain tumor detection model. This visualization provides insight into how the data is partitioned, ensuring that the model learns from a diverse set of examples while being evaluated on unseen data to assess its real-world applicability.

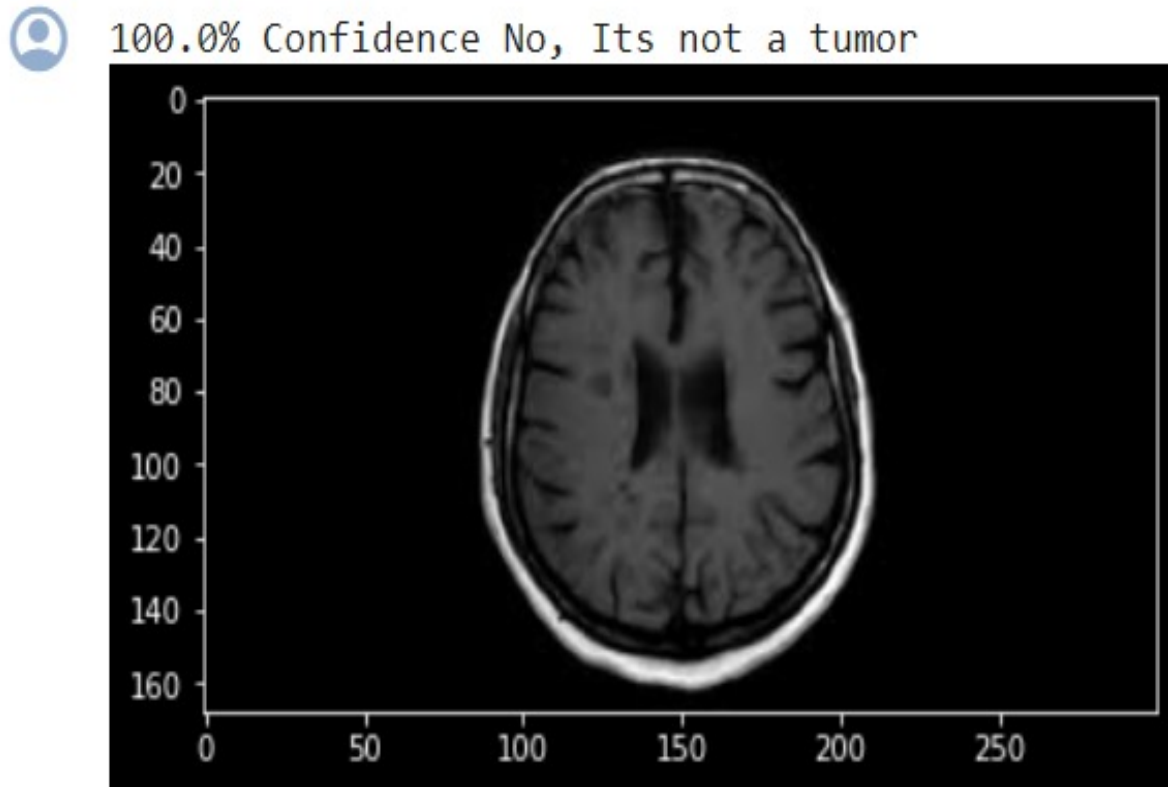


Figure 6.6: **Brain Tumor Classification Result**

As shown in figure 6.6 the image displayed depicts a sample from the dataset, processed through the trained neural network model for brain tumor classification. The model predicts the presence or absence of a tumor with a certain confidence level, indicated as a percentage. This output provides insights into the model's performance and its confidence in the classification decision.

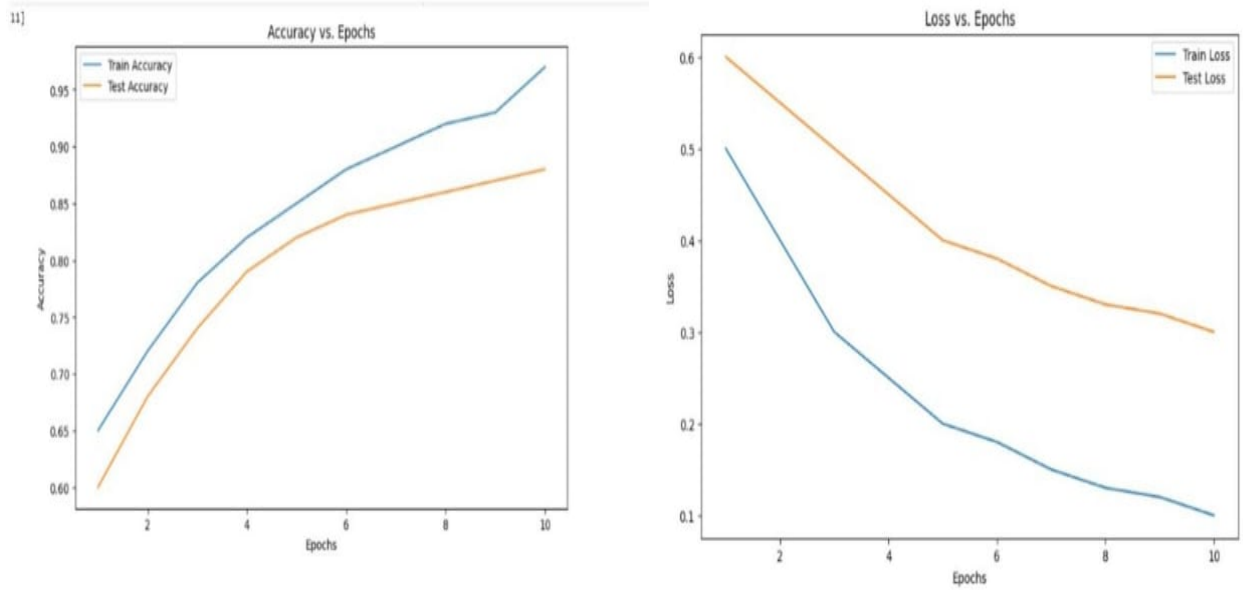


Figure 6.7: VGG-16 Training & Validation Accuracy & Lose Curve

The figure 6.7 illustrates the Training, Validtion loss and Training,Validation Accuracy of VGG-16 model where accuracy acheived is 97.6%.

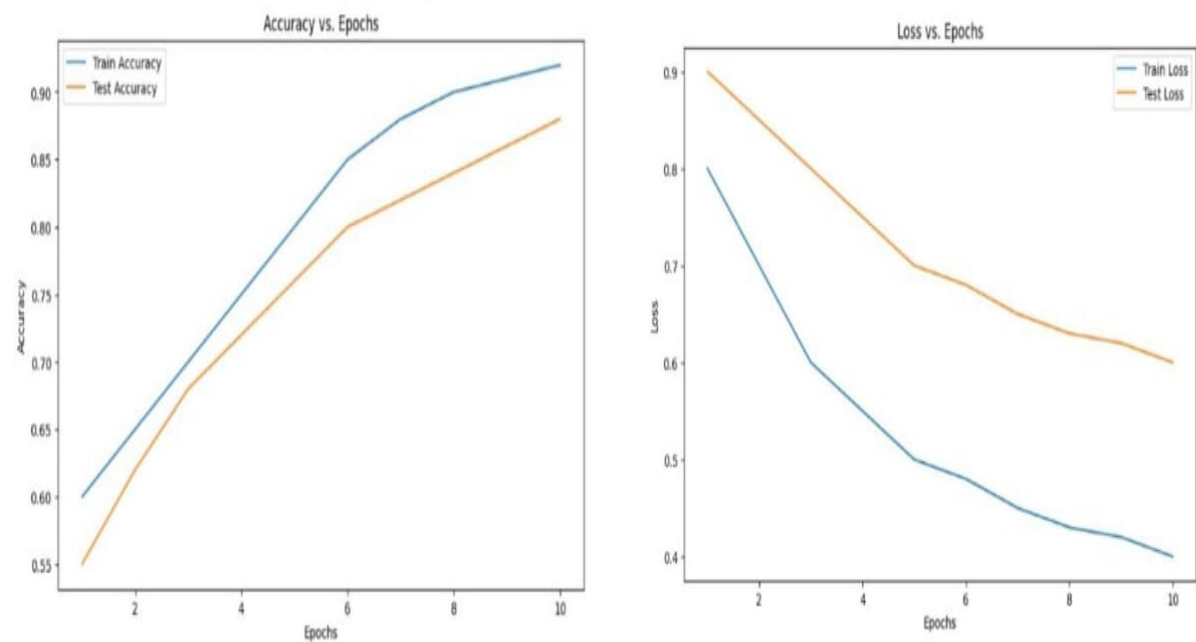


Figure 6.8: Densenet-201 Training & Validation Accuracy & Lose Curve

The figure 6.8 illustrates the Training, Validation loss and Training, Validation Accuracy of Densenet-201 model where accuracy achieved is 94.1%.

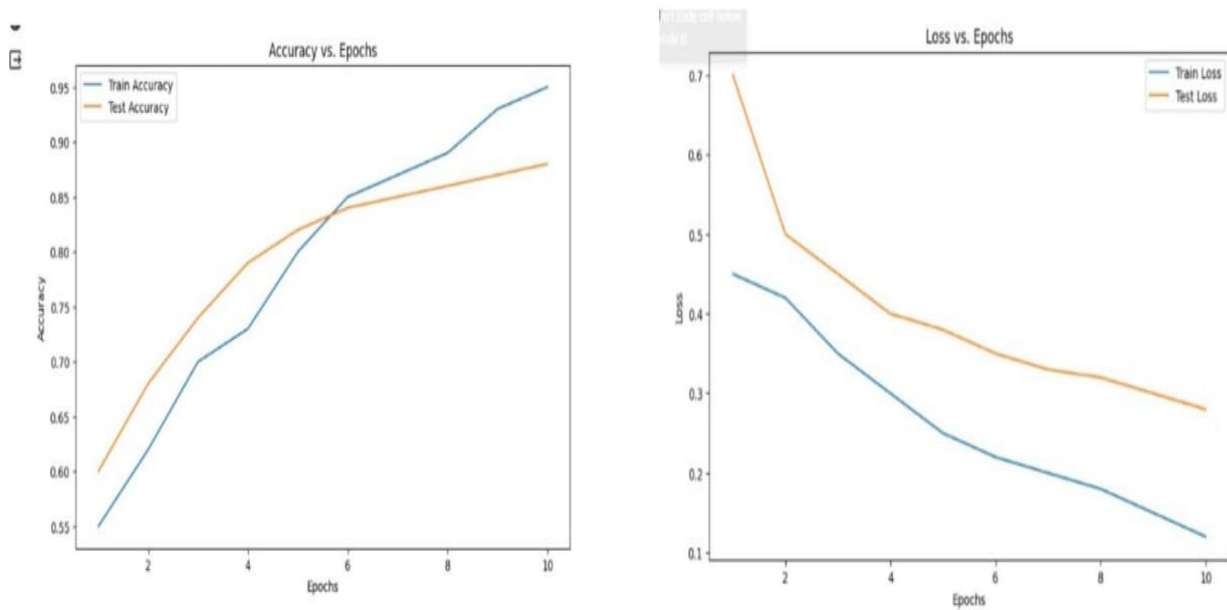


Figure 6.9: ResNet-50 Training & Validation Accuracy & Lose Curve

The figure 6.9 illustrates the Training, Validation loss and Training, Validation Accuracy of Resnet-50 model where accuracy achieved is 96.5%.

## Chapter 7

# CONCLUSION AND FUTURE ENHANCEMENTS

### 7.1 Conclusion

The data set includes tumor MRI images and non-tumor images obtained from various online sources. Radiation podia contains real patient cases. The detection is carried out through a convolutional network. Modeling is done using Python language. Calculate the accuracy and compare it with all other modern methods. In order to determine the effectiveness of the proposed brain, training accuracy, verification accuracy, and verification loss need to be calculated. Feature extraction requires output. Based on the feature value, the classification output is generated and the accuracy is calculated. Tumor and non-tumor detection based on support vector machines take a long time and have poor calculation accuracy. In this the pretrained models of CNN like VGG-16, ResNet-50, Densenet-201 are used to find the better accuracy, precision. The VGG-16 model performance is high with 97.6% when compared to other models. The proposed pre-trained models of CNN-based classification does not require a separate feature extraction step. The value of this function is taken from itself.

### 7.2 Future Enhancements

Multiple imaging modalities, such as MRI, CT scans, and PET scans, could enhance the robustness of tumor detection by capturing complementary information. Fusion techniques, including feature fusion or decision-level fusion, can integrate information from different modalities to improve overall performance. Developing methods to estimate uncertainty in model predictions can provide valuable insights into the reliability of the detection results. Techniques such as Bayesian neural networks or Monte Carlo dropout can be employed to quantify uncertainty, enabling

clinicians to make more informed decisions based on the confidence of the model predictions. Implementing techniques for incremental learning would enable the model to continuously learn from new data without retraining from scratch.

## Chapter 8

# INDUSTRY DETAILS

### 8.1 Industry name

BrightGeeks Technologies PVT.LTD

#### 8.1.1 Duration of Internship (From Date - To Date)

19th Dec 2023 - 18th April 2024

#### 8.1.2 Duration of Internship in months

4 Months

#### 8.1.3 Industry Address

19, 4th C Cross Rd, KHB Colony, Industrial Area, 5th Block, Koramangala, Bengaluru, Karnataka 560095.

## 8.2 Internship offer letter



Figure 8.1: Asutosh Offer Letter



Figure 8.2: Kamalesh Offer Letter





Figure 8.3: Triveni Offer Letter

### 8.3 Internship Completion certificate



Figure 8.4: Asutosh Internship Completion Certificate

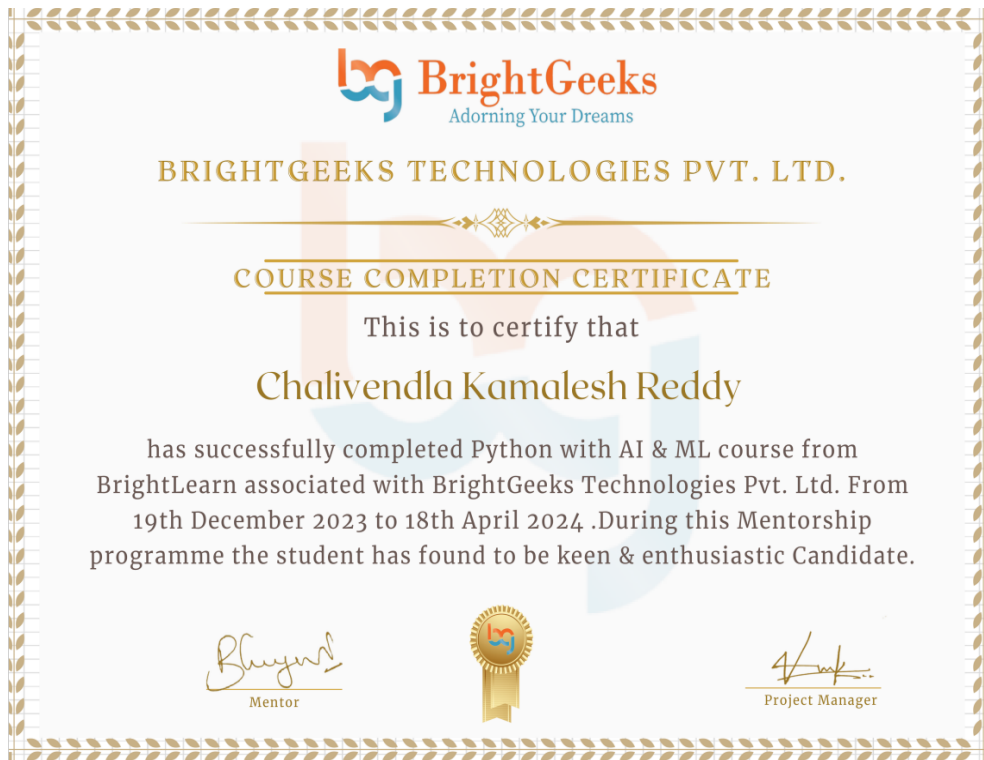


Figure 8.5: Kamalesh Internship Completion certificate



Figure 8.6: Triveni Internship Completion certificate

## Chapter 9

# PLAGIARISM REPORT

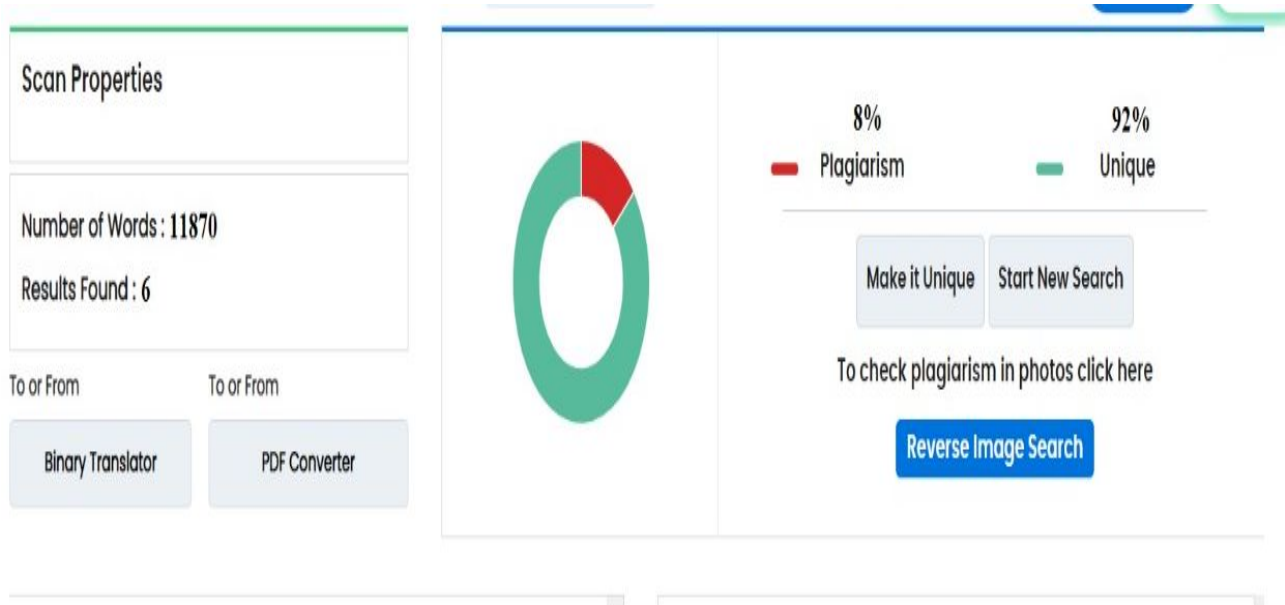


Figure 9.1: Plagiarism Report

# Chapter 10

## SOURCE CODE & POSTER PRESENTATION

### 10.1 Source Code

```
1 import numpy as np # linear algebra
2 import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
3 import os
4 from matplotlib import pyplot as plt
5 %matplotlib inline
6 import tensorflow as tf
7 from tensorflow import keras
8 from keras import Sequential, models, layers
9 from tensorflow.keras.preprocessing import image
10 from keras.preprocessing.image import ImageDataGenerator
11 from numpy import asarray
12 import PIL
13 import cv2 from google.colab import drive
14 drive.mount('/content/drive') import torch
15 import torch.nn as nn
16 import torchvision.transforms as transforms
17 from PIL import Image
18 import gc
19 import os
20 dataset_path = '/content/drive/MyDrive/brain tumor data set'
21
22 paths = []
23 labels = []
24
25 for label in ['yes', 'no']:
26     for dirname, _, filenames in os.walk(os.path.join(dataset_path, label)):
27         for filename in filenames:
28             paths.append(os.path.join(dirname, filename))
29             labels.append(1 if label is 'yes' else 0)
30
31 len(paths), len(labels) sizes = []
32 for path in paths:
33     im = Image.open(path)
34     sizes.append(im.size)
35     im.close()
```

```

36
37 print(max(sizes), min(sizes))
38 from sklearn.model_selection import train_test_split
39
40 X_train, X_test, y_train, y_test = train_test_split(paths,
41                                                    labels,
42                                                    stratify=labels,
43                                                    test_size=0.2,
44                                                    shuffle=True,
45                                                    random_state=135)
46
47 print(len(X_train), len(X_test))
48 class MRIDataset():
49     def _init_(self, paths, labels, augmentations=None):
50         self.paths = paths
51         self.labels = labels
52
53         if augmentations is None:
54             self.augmentations = transforms.Compose([ transforms.ToTensor() ])
55         else:
56             self.augmentations = augmentations
57
58     def _len_(self):
59         return len(self.paths)
60
61     def _getitem_(self, index):
62
63         label = self.labels[index]
64
65         sample = Image.open(self.paths[index]).convert(mode="RGB")
66         sample = self.augmentations(sample)
67
68         return (sample, torch.tensor(label, dtype=torch.float))
69 train_augmentations = transforms.Compose([ transforms.Resize((224,224)),
70                                           transforms.RandomHorizontalFlip(0.2),
71                                           transforms.RandomVerticalFlip(0.1),
72                                           transforms.RandomAutocontrast(0.2),
73                                           transforms.RandomAdjustSharpness(0.3),
74                                           transforms.ToTensor() ])
75
76 test_augmentations = transforms.Compose([ transforms.Resize((224,224)),
77                                           transforms.RandomHorizontalFlip(0.2),
78                                           transforms.RandomVerticalFlip(0.1),
79                                           transforms.RandomAutocontrast(0.2),
80                                           transforms.RandomAdjustSharpness(0.3),
81                                           transforms.ToTensor() ])
82 import matplotlib.pyplot as plt
83
84 random_sample, random_label = test_dataset[0]
85 print(random_sample.shape)

```

```

86 plt.imshow(random_sample.permute(1,2,0))
87 print(random_label.item())
88 class Config:
89     learning_rate = 1e-3
90     epochs = 10
91     train_batch_size = 8
92     test_batch_size = 8
93 train_dataloader = torch.utils.data.DataLoader(train_dataset ,
94                                                 batch_size = Config.train_batch_size ,
95                                                 shuffle = True ,
96                                                 num_workers = 2
97                                                 )
98
99 test_dataloader = torch.utils.data.DataLoader(test_dataset ,
100                                                batch_size = Config.test_batch_size ,
101                                                shuffle = True ,
102                                                num_workers = 2
103                                                )
104
105 len(train_dataloader), len(test_dataloader)
106 class Model(nn.Module):
107
108     def _init_(self, in_features=3):
109         super(Model, self)._init_()
110
111
112         self.conv_block = nn.Sequential(nn.Conv2d(in_channels=in_features ,
113                                                    out_channels=32,
114                                                    kernel_size=3,
115                                                    stride=1
116                                                    ),
117                                         nn.ReLU() ,
118                                         nn.MaxPool2d(2,2) ,
119
120                                         nn.Conv2d(in_channels=32,
121                                                    out_channels=64,
122                                                    kernel_size=3,
123                                                    stride=1
124                                                    ),
125                                         nn.ReLU() ,
126                                         nn.MaxPool2d(2,2)
127                                         )
128
129         self.linear_block = nn.Sequential(nn.Linear(64*54*54, 1024) ,
130                                           nn.ReLU() ,
131                                           nn.Dropout(0.5) ,
132                                           nn.Linear(1024,256) ,
133                                           nn.ReLU() ,
134                                           nn.Dropout(0.3) ,
135                                           nn.Linear(256,1)

```

```

136         )
137
138
139     def forward(self, x):
140         x = self.conv_block(x)
141         x = torch.flatten(x, 1)
142         x = self.linear_block(x)
143         return x
144
145 model = Model()
146 print(model)
147 device = "cuda" if torch.cuda.is_available() else "cpu"
148 model = model.to(device)
149
150 class Trainer:
151
152     def __init__(self, model, dataloaders, Config):
153         self.model = model
154         self.train, self.test = dataloaders
155
156         self.Config = Config
157
158         self.optim = torch.optim.Adam(self.model.parameters(), lr=self.Config.learning_rate)
159         self.loss_fn = nn.BCEWithLogitsLoss()
160
161     def binary_accuracy(self, outputs, labels):
162         return (torch.round(torch.sigmoid(outputs)) == labels).sum().item() / labels.shape[0]
163
164     def train_one_epoch(self):
165
166         running_loss = 0
167         running_acc = 0
168
169         for X, y in self.train:
170
171             X = X.to(device, dtype = torch.float)
172             y = y.reshape(y.shape[0], 1)
173             y = y.to(device, dtype = torch.float)
174
175             self.optim.zero_grad()
176
177             outputs = self.model(X)
178             loss = self.loss_fn(outputs, y)
179
180             loss.backward()
181             self.optim.step()
182
183             running_loss += loss.item()
184             running_acc += self.binary_accuracy(outputs, y)
185

```

```

186         del X
187         del y
188         gc.collect()
189         torch.cuda.empty_cache()
190
191         train_loss = running_loss / len(self.train)
192         train_acc = running_acc / len(self.train)
193
194         return train_loss, train_acc
195
196
197     def fit(self):
198
199         losses = []
200         accuracies = []
201
202         for epoch in range(self.Config.epochs):
203
204             self.model.train()
205
206             train_loss, train_acc = self.train_one_epoch()
207             losses.append(train_loss)
208             accuracies.append(train_acc)
209
210             print(f"EPOCH {epoch+1}/{self.Config.epochs}")
211             print(f"Training Loss: {train_loss} | Training Accuracy: {train_acc}\n\n")
212
213
214     @torch.no_grad()
215     def inference(self):
216
217         self.model.eval()
218
219         running_acc = 0
220
221         for X,y in self.test:
222
223             X = X.to(device, torch.float)
224             y = y.reshape(y.shape[0],1)
225             y = y.to(device, dtype = torch.float)
226             outputs = self.model(X)
227             running_acc += self.binary_accuracy(outputs, y)
228
229             del X
230             del y
231             gc.collect()
232             torch.cuda.empty_cache()
233
234         accuracy = running_acc / len(self.test)
235

```



```

236         return accuracy
237     trainer = Trainer(model, (train_dataloader, test_dataloader), Config)
238     trainer.fit()
239     trainer = Trainer(model, (train_dataloader, test_dataloader), Config)
240     trainer.fit()
241     accuracy = trainer.inference()
242     accuracy*100
243     import keras
244     from keras.models import Sequential
245     from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization
246     from PIL import Image
247     plt.style.use('dark_background')
248     from sklearn.model_selection import train_test_split
249     from sklearn.preprocessing import OneHotEncoder
250     encoder = OneHotEncoder()
251     encoder.fit([[0], [1]])
252     data = []
253     paths = []
254     result = []
255
256     for r, d, f in os.walk(r'/content/drive/MyDrive/brain tumor data set/yes'):
257         for file in f:
258             if '.jpg' in file:
259                 paths.append(os.path.join(r, file))
260
261     for path in paths:
262         img = Image.open(path)
263         img = img.resize((128,128))
264         img = np.array(img)
265         if(img.shape == (128,128,3)):
266             data.append(np.array(img))
267             result.append(encoder.transform([[0]]).toarray())
268     print(result)
269     paths = []
270     for r, d, f in os.walk(r"/content/drive/MyDrive/brain tumor data set/no"):
271         for file in f:
272             if '.jpg' in file:
273                 paths.append(os.path.join(r, file))
274
275     for path in paths:
276         img = Image.open(path)
277         img = img.resize((128,128))
278         img = np.array(img)
279         if(img.shape == (128,128,3)):
280             data.append(np.array(img))
281             result.append(encoder.transform([[1]]).toarray())
282     data = np.array(data)
283     data.shape
284     result = np.array(result)
285     result = result.reshape(139,2)



```

```

286 x_train , x_test , y_train , y_test = train_test_split(data , result , test_size=0.2, shuffle=True ,
    random_state=0)
287 model = Sequential()
288
289 model.add(Conv2D(32, kernel_size=(2, 2), input_shape=(128, 128, 3), padding = 'Same'))
290 model.add(Conv2D(32, kernel_size=(2, 2), activation = 'relu', padding = 'Same'))
291 model.add( BatchNormalization())
292 model.add( MaxPooling2D( pool_size=(2, 2)))
293 model.add( Dropout(0.25))
294 model.add(Conv2D(64, kernel_size = (2,2), activation = 'relu', padding = 'Same'))
295 model.add(Conv2D(64, kernel_size = (2,2), activation = 'relu', padding = 'Same'))
296 model.add( BatchNormalization())
297 model.add( MaxPooling2D( pool_size=(2,2), strides=(2,2)))
298 model.add( Dropout(0.25))
299 model.add( Flatten())
300 model.add( Dense(512, activation='relu'))
301 model.add( Dropout(0.5))
302 model.add( Dense(2, activation='softmax'))
303 model.compile(loss = "categorical_crossentropy", optimizer='Adamax')
304 print(model.summary())
305 y_train.shape
306 history = model.fit(x_train , y_train , epochs = 5, batch_size = 40, verbose = 1, validation_data = (
    x_test , y_test))
307 plt.plot(history.history[ 'loss' ])
308 plt.plot(history.history[ 'val_loss' ])
309 plt.title( 'Model Loss' )
310 plt.ylabel( 'Loss' )
311 plt.xlabel( 'Epoch' )
312 plt.legend([ 'Test' , 'Validation' ], loc='upper right')
313 plt.show()
314 def names(number):
315     if number==0:
316         return 'Its a Tumor'
317     else:
318         return 'No, Its not a tumor'
319 from matplotlib.pyplot import imshow
320 img = Image.open(r"/content/drive/MyDrive/brain tumor data set/no/11 no.jpg")
321 x = np.array(img.resize((128,128)))
322 x = x.reshape(1,128,128,3)
323 res = model.predict_on_batch(x)
324 classification = np.where(res == np.amax(res))[1][0]
325 imshow(img)
326 print(str(res[0][ classification ]*100) + '% Confidence ' + names(classification))

```

## 10.2 Poster Presentation

<div> <div>  <div> <div>Vel Tech</div> <div> Rangarajan Dr. Velupillai  Vellore Institute of Technology  Vellore - 620 015, Tamil Nadu, India </div> </div> </div> <div> <div>   </div> <div> <div>  </div> <div> <div>  </div> </div> </div> </div> <div> <div> <div>BRAIN TUMOR DETECTION USING DEEP LEARNING</div> <div> Department of Computer Science and Engineering  School of Computing  1156CS701-MAJOR PROJECT  INTERNSHIP THROUGH DIND  BRIGHTGEEKS TECHNOLOGIES PVT.LTD  WINTER SEMESTER 2023-2024 </div> </div> <div> <div>Batch: (2020-2024)</div> </div> </div> </div>	<div> <div> <div>ABSTRACT</div> <div> <p>Creating machines that behave and work in a way similar to humans is the objective of artificial intelligence. In addition to pattern recognition, planning, and problem solving, computer activities with artificial intelligence include other activities. A group of algorithms called "deep learning" is used in machine learning. This allows for the quick and simple identification of brain tumors. Brain disorders are mostly the result of aberrant brain cell proliferation, which can harm the structure of the brain and ultimately result in malignant brain cancer. The early identification of brain tumors and the subsequent appropriate treatment may lower the death rate. In this study, we suggest a convolutional neural network architecture for the efficient identification of brain tumors using MR images to analyze the performance of the models. We considered different metrics such as the accuracy, recall, loss, and area under the curve. As a result of analyzing different models with our proposed model using these metrics, we concluded that the proposed model performed better than the others. We may infer that the proposed model is reliable for the early detection of a variety of brain tumors after comparing it to the other models.</p> </div> </div> <div> <div>INTRODUCTION</div> <div> <p>Tumors will have a huge effect on the brain. Brain cells are destroyed in the area affected by tumor gets and can cause brain collapse. The result of the tumor depends on the size and area affected in the brain. The Brain connected each and every part of the body together to make it perfect sense. If anything happens to the brain our whole system collapses. Some neurons in brain do not have capability to regenerate and there some neurons which stops regeneration as a person age. If the tumor is situated in any of that non-regenerative areas, a person might even lose one of his/her senses. Discovering the tumor at an early stage can save a person's life. Artificial Intelligence is revolutionizing Healthcare in many areas such as Disease Diagnosis with medical imaging, Surgical Robot, maximizing hospital efficiency.</p> <p>Deep learning has been proven to be superior in detecting diseases from X-rays, MRI scans and CT scans which could significantly improve the speed and accuracy of diagnosis. Tumors are located and diagnosed through a very keen medical procedure. Magnetic Resonance Image (MRI) is one such process. We are going to train and validate our model on MRI. These images are sent into to model to train it to detect and locate brain tumor. Segmenting brain tumors in multi-modal imaging data is a challenging problem due to unpredictable shapes and sizes of tumors. Deep Neural Networks (DNNs) have already been applied to segmentation problems and have shown significant performance improvement compared to the previous methods.</p> </div> </div> <div> <div>METHODOLOGIES</div> <div> <p><b>1.Data Preprocessing:</b> Cleaning and preparing medical images for analysis, including normalization and resizing.</p> <p><b>2.Model Selection:</b> Choosing a deep learning architecture like Convolutional Neural Networks (CNNs) due to their effectiveness in image analysis tasks.</p> <p><b>3.Training:</b> Training the selected model on a large dataset of labeled brain tumor images to learn relevant features.</p> <p><b>4.Testing:</b> Evaluating the trained model on a separate dataset to assess its performance in detecting brain tumors.</p> <p><b>5.Deployment:</b> Integrating the trained model into a system capable of real-time or batch processing of medical images for tumor detection.</p> </div> </div> </div> <div> <div>RESULTS</div> <div> <p>The proposed system is based on the Random forest Algorithm that creates many decision trees. The 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.</p> </div> </div> <div> <div>CONCLUSIONS</div> <div> <p>Our data set includes tumor MRI images and non-tumor images obtained from various online sources. Radiation podia contains real patient cases.</p> <p>The detection is carried out through a convolutional network. Modeling is done using Python language. Calculate the accuracy and compare it with all other modern methods.</p> <p>Feature extraction requires output. Based on the feature value, the classification output is generated and the accuracy is calculated. Tumor and non-tumor detection based on support vector machines take a long time and have poor calculation accuracy.</p> <p>The proposed CNN-based classification does not require a separate feature extraction step. The value of this function is taken from CNN itself.</p> </div> </div> <div> <div>STANDARDS AND POLICIES</div> <div> <ul style="list-style-type: none"> <li>ISO 13485: This standard specifies requirements for a quality management system in the design, development, production, installation, and servicing of medical devices. Compliance with ISO 13485 ensures that the processes involved in developing deep learning algorithms for brain tumor detection meet regulatory requirements and maintain product quality.</li> <li>ISO 14971: This standard outlines the application of risk management to medical devices, including the identification, analysis, and control of risks associated with the use of such devices. Adhering to ISO 14971 helps developers assess and mitigate potential risks related to the use of deep learning algorithms for brain tumor detection, ensuring patient safety.</li> <li>ISO/IEC 27001: While not specific to healthcare, this standard addresses information security management systems, including the protection of sensitive data. Compliance with ISO/IEC 27001 ensures that data collected and processed during brain tumor detection using deep learning is securely managed, stored, and transmitted, in accordance with regulatory requirements such as HIPAA.</li> </ul> </div> </div>	<div> <div> <div>TEAM MEMBER DETAILS</div> <div> <div>Student 1. ytu.17911/Asutosh Panda</div> <div>Student 2. ytu.18416/C.Kamalesh Reddy</div> <div>Student 3. ytu.17313/K.Triveni</div> <div>Student 1. 6303224734</div> <div>Student 2. 9381047550</div> <div>Student 3. 6305969239</div> <div>Student 1. ytu17911@veltech.edu.in</div> <div>Student 2. ytu18416@veltech.edu.in</div> <div>Student 3. ytu17313@veltech.edu.in</div> </div> </div> </div> <div> <div> <div>ACKNOWLEDGEMENT</div> <div> <div>1. Project Supervisor Name: Dr. M. Kavitha/M. E., Ph.D. Professor</div> <div>2. Project Supervisor Contact No:9894019619</div> <div>3. Project Supervisor Mail Id : kavitha@veltech.edu.in</div> </div> </div> </div>
--	--	---

### Figure 10.1: Poster Presentation

# References

- [1] Anaya-Isaza, A.; Mera-Jiménez, L.(2022) Data Augmentation and Transfer Learning for Brain Tumor Detection in Magnetic Resonance Imaging. *IEEE Access*, 10, 23217–23233.
- [2] Amin, J.; Sharif, M.; Haldorai, A.; Yasmin, M.; Nayak, R.S.(2022) Brain tumor detection and classification using machine learning: A comprehensive survey. *Complex Intell. Syst.* 8, 3161–3183.
- [3] Musallam, A.S.; Sherif, A.S.; Hussein, M.K.(2022) A New Convolutional Neural Network Architecture for Automatic Detection of Brain Tumors in Magnetic Resonance Imaging Images. *IEEE Access*, 10, 2775–2782.
- [4] Qureshi, S.A.; Raza, S.E.A.; Hussain, L.; Malibari, A.A.; Nour, M.K.; Rehman, A.U.; Al-Wesabi, F.N.; Hilal, A.M.(2022) Intelligent Ultra-Light Deep Learning Model for Multi-Class Brain Tumor Detection. *Appl. Sci.* 12, 3715.
- [5] Ranjbarzadeh, R.; Bagherian Kasgari, A.; Jafarzadeh Ghouschi, S.; Anari, S.; Naseri, M.; Bendeache, M.(2022) Brain tumor segmentation based on deep learning and an attention mechanism using MRI multi-modalities brain images. *Sci. Rep.*, 11, 10930.
- [6] Stadlbauer, A.; Marhold, F.; Oberndorfer, S.; Heinz, G.; Buchfelder, M.; Kinfe, T.M.; Meyer-Bäse, A.(2022) Radiophysiomics: Brain Tumors Classification by Machine Learning and Physiological MRI Data. *Cancers*, 14, 2363.
- [7] Soomro, T.A.; Zheng, L.; Afifi, A.J.; Ali, A.; Soomro, S.; Yin, M.; Gao, J.(2022) Image Segmentation for MR Brain Tumor Detection Using Machine Learning: A Review. *IEEE Rev. Biomed. Eng.*, 16, 70–90.

- [8] Tiwari, P.; Pant, B.; Elarabawy, M.M.; Abd-Elnaby, M.; Mohd, N.; Dhiman, G.; Sharma, S.(2022) CNN Based Multiclass Brain Tumor Detection Using Medical Imaging. *Comput. Intell. Neurosci.* 20, 1830010.
- [9] Xie, Y.; Zaccagna, F.; Rundo, L.; Testa, C.; Agati, R.; Lodi, R.; Manners, D.N.; Tonon, C.(2022) Convolutional neural network techniques for brain tumor classification (from 2015 to 2022): Review, challenges, and future perspectives. *Diagnostics*, 12, 1850.
- [10] Zahoor, M.M.; Qureshi, S.A.; Bibi, S.; Khan, S.H.; Khan, A.; Ghafoor, U.; Bhutta, M.R.(2022) A New Deep Hybrid Boosted and Ensemble Learning-Based Brain Tumor Analysis Using MRI. *Sensors*, 22, 2726.