



**BRAIN TUMOR SEGMENTATION USING
MULTIPLE MRI MODALITIES**

PROJECT REPORT

Submitted by

A.PAWAN

(111420243016)

P.YUVARAJ

(111420243026)

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

PRATHYUSHA ENGINEERING COLLEGE

(AN AUTONOMOUS INSTITUTION)

THIRUVALLUR – 602 025

APRIL 2024

PRATHYUSHA ENGINEERING COLLEGE
(AN AUTONOMOUS INSTITUTION)
TIRUVALLUR – 602 025

BONAFIDE CERTIFICATE

Certified that this project report “**BRAIN TUMOR SEGMENTATION USING MULTIPLE MRI MODALITIES**” is the bonafide work of "A PAWAN (111420243016), YUVARAJ P (111420243026) " who carried out the project work under my supervision.

SIGNATURE

Ms. PRIYANKA D , M.Tech.,
ASSISTANT PROFESSOR,
SUPERVISOR,
Department of AI & DS,
Prathyusha Engineering College,
Tiruvallur – 602 025.

SIGNATURE

Ms. R. KANNAMMA , M.Tech., (PhD)
ASSOCIATE PROFESSOR,
HEAD OF THE DEPARTMENT,
Department of AI & DS,
Prathyusha Engineering College,
Tiruvallur – 602 025.

Place: Tiruvallur

Date: 05-04-2024

Submitted for the Project Viva-Voce held on at
Prathyusha Engineering College, Thiruvallur – 602 025 .

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

Our sincere thanks to **Shri. P. RAJARAO**, Chairman, Prathyusha Engineering College for facilitating us to do this project.

We are grateful to our CEO, **Smt. P. PRATHYUSHA**, for being a source of Inspiration.

We are grateful to our **Director, Dr. P. M. BEULAH DEVAMALAR**, for supporting us during the development.

We are sincerely thanking **Dr. B. R. RAMESH BAPU, M.E., PhD., Principal**, Prathyusha Engineering College for encouraging our endeavors for this project.

We are sincerely thanking our HOD **Ms. R. KANNAMA, M.Tech., (PhD), Head of Department** for supporting us in all stages of project conduction.

We are grateful to our project coordinator **Ms. C. KAMATCHI, M.E.,(PhD) Assistant Professor**, for her valuable suggestions and guidance.

We are more thankful to our project guide **Ms. PRIYANKA D , M.Tech. ,Assistant Professor** , for assisting us with suggestions to complete our project .

We wish to express our sincere gratitude to **PARENTS AND FRIENDS** for valuable help, co-operation and encouragement during this project work.

Last but not least, we wish to express our sincere thanks to the entire teaching faculty and non-teaching staff for their support.

ABSTRACT

Recent advancements in healthcare, propelled by image processing and computer vision technologies, have ushered in notable improvements in diagnostic accuracy, cost efficiency, and time management. Gliomas, the most prevalent primary brain tumors, arise from the malignant transformation of glial cells in the brain and spinal cord. Magnetic Resonance Imaging (MRI) plays a pivotal role in medical imaging, particularly in segmenting brain tumors. Utilizing various pulse sequences during MRI scans yields distinct modalities (T1CE, T2, and FLAIR), each providing unique insights into tissue composition. The integration of the U-Net architecture, a convolutional neural network specialized in semantic segmentation in medical imaging, within a multimodal fusion framework enhances the identification of critical tumor features by prioritizing information across modalities. Employing two tailored model architectures, the "simple_unet_model" and the "improved_unet_model," aims to address segmentation intricacies. This synergistic approach aims to bolster the accuracy and reliability of brain tumor segmentation, offering clinicians precise diagnoses and facilitating personalized treatment plans. Ultimately, this advancement holds promise for optimizing patient care, potentially leading to improved treatment outcomes and increased survival rates for individuals affected by brain tumors.

TABLE OF CONTENTS

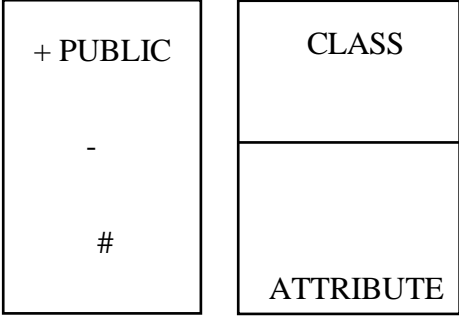
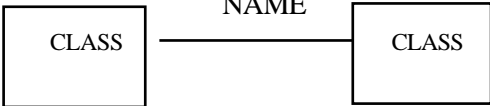

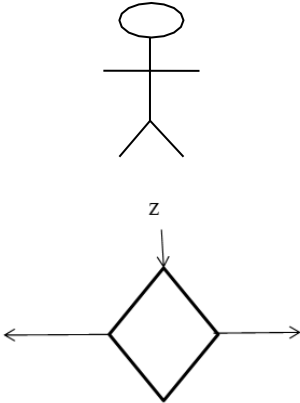

CH.NO	TITLE	PAGE NO
	ABSTRACT	iv
	LIST OF FIGURES	vii
	LIST OF SYMBOLS	viii
1	INTRODUCTION	1
	1.1 OVERVIEW	2
	1.2 OBJECTIVE	3
	1.3 LITERATURE SURVEY	3
2	SYSTEM ANALYSIS	6
	2.1 EXISTING SYSTEM	7
	2.1.1 DISADVANTAGES	7
	2.2 PROPOSED SYSTEM	8
	2.2.1 ADVANTAGES	9
3	SYSTEM REQUIREMENTS	10
	3.1 HARDWARE REQUIREMENTS	11
	3.2 SOFTWARE REQUIREMENTS	11
	3.3 SOFTWARE DESCRIPTION	12
4	SYSTEM DESIGN	14
	4.1 SYSTEM ARCHITECTURE	15
	4.2 UML DIAGRAM	16
	4.2.1 USE CASE DIAGRAM	16
	4.2.2 ACTIVITY DIAGRAM	17
	4.2.3 SEQUENCE DIAGRAM	18
5	SYSTEM IMPLEMENTATION	19



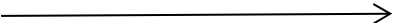
5.1	LIST OF MODULES	20
5.2	MODULE DESCRIPTION	20
5.2.1	DATA LOADING & UNDERSTANDING	21
5.2.2	DATA PREPROCESSING & SELECTION	21
5.2.3	MODEL TRAINING	21
5.2.4	WEB PAGE INTEGRATION	22
6	TESTING	24
6.1	UNIT TESTING	24
6.2	INTEGRATION TESTING	25
6.3	SYSTEM TESTING	25
7	RESULTS AND DISCUSSION	27
7.1	RESULTS	27
7.2	DISCUSSION	28
8	CONCLUSION AND FUTURE ENHANCEMENT	30
8.1	CONCLUSION	30
8.2	FUTURE ENHANCEMENT	31
9	ANNEXURE	32
	ANNEXURE 1 SOURCE CODE	33
	ANNEXURE 2 SCREENSHOT	43
10	REFERENCES	47

LIST OF FIGURES

FIG. NO.	FIGURE NAME	PAGE NO.
4.1	System architecture	14
4.2.1	Use case diagram	15
4.2.2	Activity diagram	16
4.2.3	Sequence diagram	17

LIST OF SYMBOLS

S.No	NOTATION NAME	NOTATION	DESCRIPTION
1.	Class		Represents a collection of similar entities grouped together.
2.	Association		Associations represent Static relationships between classes.
			Roles represent the way the two classes see each Other.
3.	Actor		Specifies a role played by a user that interacts with the subject.
4.	Communication		Communication between various use cases

S.No	NOTATION NAME	NOTATION	DESCRIPTION
5.	Initial State		Initial state of the object
6.	Final state		Final state of the object
7.	Control flow		Represents various control flow between the states.

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Brain tumors include the most threatening types of tumors around the world. A brain tumor is an abnormal growth of cells within the brain or the surrounding tissues that can cause a range of neurological symptoms and potentially life-threatening complications. Glioma, the most common primary brain tumor, occurs due to the carcinogenesis of glial cells in the spinal cord and brain. Brain tumors can be benign or malignant, and they can originate from different types of brain cells or from other organs that have spread to the brain . A brain tumor is an abnormal growth of cells within the brain or the surrounding tissues that can cause a range of neurological symptoms and potentially life-threatening complications. Glioma, the most common primary brain tumor, occurs due to the carcinogenesis of glial cells in the spinal cord and brain. Brain tumors can be benign or malignant, and they can originate from different types of brain cells or from other organs that have spread to the brain.

Magnetic Resonance Imaging (MRI) has become the cornerstone for diagnosing and managing brain tumors. However, accurate segmentation of these tumors from MRI scans remains a challenge. Traditional methods often rely on a single MRI modality, which can miss crucial details present in other modalities. This limitation can lead to inaccurate segmentation, hindering effective patient care, such as misdiagnosis or suboptimal treatment planning. To overcome this, a novel approach for brain tumor segmentation that leverages the power of multimodal fusion and a U-Net architecture. Multimodal fusion integrates information from T1, T1-contrast enhanced (T1CE), and T2-weighted MRI images. Each modality offers a unique perspective on the tumor: T1-weighted

images excel at revealing anatomical details, while T2-weighted images provide insights into water content, crucial for differentiating tumor tissue from surrounding fluids. T1CE images, with their contrast agent, further highlight tumor regions. By combining information from all three modalities, the proposed approach aims to create a more comprehensive picture of the tumor and its surrounding tissues, leading to improved segmentation accuracy. This can potentially translate into better clinical outcomes for patients through more precise diagnoses and personalized treatment plans.

1.2 OBJECTIVE

The primary objective is to develop a robust and accurate brain tumor segmentation system using multimodal MRI data. The system integrate information from T1, T1CE, and T2-weighted MRI images, providing a more comprehensive view of the tumor and its surrounding tissues. Utilize a U-Net architecture for effective segmentation. U-Net's architecture is well-suited for medical image segmentation tasks due to its ability to capture both high-resolution details and broader anatomical context. Generate interpretable results for improved clinical decision-making. By understanding the model's reasoning behind the segmentation, physicians can make more informed treatment decisions.

1.3 LITERATURE SURVEY

[1] Liu, Jin, Min Li, Jianxin Wang, Fangxiang Wu, Tianming Liu, and Yi Pan. "A survey of MRI-based brain tumor segmentation methods." Tsinghua science and technology 19, no. 6 (2014): 578-595.

It provided an extensive review of MRI-based brain tumor segmentation methods, which aimed to distinguish various tumor tissues from normal brain tissues in MRI images. With the growing interest in non-invasive imaging, these techniques advanced significantly over the past two decades, approaching routine

clinical use. The paper covered brain tumor basics, imaging modalities, preprocessing steps, state-of-the-art segmentation methods, and evaluation strategies. It concluded with an objective assessment and outlined future directions in MRI-based brain tumor segmentation.

[2] Toğaçar, Mesut, Burhan Ergen, and Zafer Cömert. "BrainMRNet: Brain tumor detection using magnetic resonance images with a novel convolutional neural network model." *Medical hypotheses* 134 (2020): 109531.

BrainMRNet - a novel convolutional neural network designed for the precise detection and treatment of brain tumors. Incorporating attention modules and the hypercolumn technique, it efficiently selected vital features from MRI images. Through rigorous image preprocessing and augmentation, BrainMRNet highlighted critical areas, enhancing diagnostic accuracy. BrainMRNet demonstrated superior performance in brain tumor detection, showcasing its potential in advancing biomedical imaging technology.

[3] Amin, Javeria, Muhammad Sharif, Mussarat Yasmin, and Steven Lawrence Fernandes. "Big data analysis for brain tumor detection: Deep convolutional neural networks." *Future Generation Computer Systems* 87 (2018): 290-297.

This study proposed a methodology for brain tumor segmentation and classification in MRI images using Deep Neural Networks (DNN). The model employed a 7-layer architecture with 3 convolutional layers, 3 ReLU layers, and a softmax layer for classification. Input MR images were divided into patches, and the center pixel value of each patch was used for segmentation. The DNN assigned labels based on center pixels to perform segmentation. The model divided an MR image into multiple patches, assigned labels, and performed segmentation.

[4] Havaei M, Davy A, Warde-Farley D, Biard A, Courville A, Bengio Y, Pal C, Jodoin PM, Larochelle H. Brain tumor segmentation with Deep Neural Networks. Med Image Anal. 2017 Jan;35:18-31. doi: 10.1016/j.media.2016.05.004. Epub 2016 May 19. PMID: 27310171.

Fully automatic brain tumor segmentation method using Deep Neural Networks tailored to glioblastomas in MR images is used. The approach incorporated novel CNN architectures, leveraging both local and global contextual features efficiently. A 2-phase training procedure addressed label imbalance, while a cascade architecture enhanced segmentation accuracy by integrating outputs from multiple CNNs. This method achieved competitive performance by efficiently capturing diverse tumor shapes, sizes, and contrasts in brain images.

[5] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. ArXiv.

A network and training approach leveraging data augmentation is incorporated to improve the efficiency. The network architecture featured a contracting path for context and an expanding path for precise localization. Demonstrating its efficacy, the network outperformed previous methods in the ISBI challenge for neuronal structure segmentation, even with minimal training data. It also excelled in the ISBI cell tracking challenge 2015, using transmitted light microscopy images. Additionally, the network boasted impressive speed, capable of segmenting a 512x512 image. The full implementation, based on Caffe, and trained networks were made openly accessible for further research.

CHAPTER 2

SYSTEM ANALYSIS

CHAPTER 2

SYSTEM ANALYSIS

The system study is to provide the description about the existing system, its limitation and proposed system.

2.1 EXISTING SYSTEM

The current method employs YOLO, a prominent object identification framework, for segmenting and classification of brain tumors using Contrast-enhanced T1 (T1-ce) weighted images from Magnetic Resonance Imaging (MRI). T1-ce sequences involve injecting a contrast agent, typically gadolinium, before the scan. This agent accumulates in regions with heightened blood flow or disruptions in the blood-brain barrier, such as tumors or areas of inflammation or infection. It is a type of deep convolutional neural network, where various layers extract meaningful features, and object localization is achieved within images. It generates bounding box predictions allowing the model to detect objects of various sizes and aspect ratios. While YOLO demonstrates efficiency in general object detection, its application in brain tumor segmentation requires a nuanced understanding of its functionalities and limitations.

2.1.1 DISADVANTAGES

- It lack interpretability, hindering understanding of their predictions, critical for medical applications.
- Effective training demands large, well-annotated datasets.
- Sensitivity to Image Quality.
- Struggles to detect small objects.
- Hard to detect the tumor in early stages.

2.2 PROPOSED SYSTEM

A novel approach for brain tumor segmentation that capitalizes on the synergistic information gleaned from a combination of magnetic resonance imaging (MRI) modalities is proposed. These modalities include T1-weighted with contrast enhancement (T1CE), T2-weighted, and FLAIR (Fluid-Attenuated Inversion Recovery) images. Each modality offers a distinct window into the tumor's characteristics, providing valuable insights that, when combined, can lead to more accurate segmentation. The system leverages two distinct models within the U-Net architecture, tailored to handle the complexities of medical image analysis. The first model, referred to as the "Simple UNet," offers a streamlined approach optimized for efficient computation and resource utilization. In contrast, the second model, known as the "Improved UNet," incorporates enhancements such as batch normalization and increased network depth to further improve segmentation accuracy and robustness.

T1CE images: The administration of a contrast agent during T1CE sequences alters the signal intensity of tissues based on blood flow and blood-brain barrier integrity. This technique effectively highlights tumor regions due to their typically increased vascularization and disruption of the blood-brain barrier.

T2-weighted images: These images excel at revealing water content within the body. Since both fat and water appear bright on T2-weighted images, while bone and air appear dark, they play a crucial role in differentiating tumor tissue from surrounding structures based on their water content.

FLAIR images: Similar to T2-weighted images, FLAIR sequences provide insights into water content. However, FLAIR employs an inversion pulse that suppresses the signal from cerebrospinal fluid (CSF), making it appear dark on the image. This distinction is particularly valuable for differentiating tumors from surrounding fluid-filled regions.

By employing the U-Net architecture, which is a convolutional neural network (CNN) specifically designed for semantic segmentation in medical image analysis, the proposed method aims to achieve precise delineation of tumor boundaries across all three MRI modalities. U-Net's strengths lie in its ability to effectively handle complex medical images and generate interpretable results. This interpretability is crucial for clinical adoption, as physicians need to understand the rationale behind the model's predictions to confidently use AI in their decision-making processes.

2.2.1 ADVANTAGES

- Pixel-wise segmentation provides precise delineation of brain tumor boundaries
- Handles arbitrary input sizes, making it more flexible
- Effectively integrate information from multi-modal fusion
- Ability to leverage Multi-scale feature integration
- Versatility in handling various MRI modalities
- Produces high-resolution segmentation masks that preserve fine details of brain tumor regions
- Easier to interpret which enhancing trust and clinical acceptance

CHAPTER 3

SYSTEM REQUIREMENTS

CHAPTER 3

SYSTEM REQUIREMENTS

The requirements specification is a technical specification of requirements for the software products. It is the first step in the requirements analysis process it lists the requirements of a particular software system including functional, performance and security requirements. The requirements also provide usage scenarios from a user, an operational and an administrative perspective.

3.1 HARDWARE REQUIREMENTS

3.1.1 MINIMUM REQUIREMENTS

CPU	: Intel i5 7th Gen / Ryzen 5 1600
GPU	: Nvidia GTX 1050
RAM	: 16GB
OS	: Windows / Linux / Mac

3.1.2 RECOMMENDED REQUIREMENTS

CPU	: Intel i5 12 th Gen / AMD Ryzen 5600X
GPU	: Nvidia RTX 2060 / Nvidia RTX A2000
RAM	: 32GB
OS	: Windows / Linux / Mac

3.2 SOFTWARE REQUIREMENTS

Languages	: Python 3.7+
Frameworks/ Modules	: Tensorflow , matplotlib , nibabel , sklearn , numpy , segmentation-models

3.3 SOFTWARE DESCRIPTION

Software Description is a technical specification of requirement of software product. This specifies the environment for development, operation and maintenance of the product.

3.3.1 Python3.7

Python 3.7 is a version of the Python programming language, released on June 27th, 2018. It is the latest stable release in the 3.x series of Python. Python is a popular high-level, general-purpose programming language that is widely used for web development, scientific computing, data analysis, artificial intelligence, machine learning, and more. Python 3.7 includes many new features, improvements, and bug fixes compared to its previous versions.

3.3.2 Tensorflow

TensorFlow is an open-source machine learning framework developed by Google. It is widely used for various tasks in machine learning and deep learning, including neural networks, natural language processing, and computer vision.

3.3.3 Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It provides a wide range of plotting functions to generate plots, histograms, scatterplots, and more.

3.3.4 NiBabel

NiBabel is a Python library for reading and writing neuroimaging data formats. It supports various formats used in neuroimaging research, such as NIfTI and ANALYZE, making it a valuable tool for processing and analyzing brain imaging data

3.3.5 sklearn

scikit-learn is a popular machine learning library that provides simple and efficient tools for data mining and data analysis. It includes various algorithms for classification, regression, clustering, dimensionality reduction, and model selection.

3.3.6 NumPy

NumPy is a fundamental package for scientific computing in Python. It provides support for multidimensional arrays, matrices, mathematical functions, linear algebra, random number generation, and more. NumPy is widely used in scientific and numerical computing tasks.

3.3.7 Segmentation-models

Segmentation-models is a Python library that provides pre-trained segmentation models for image segmentation tasks. It includes various state-of-the-art architectures for semantic segmentation, instance segmentation, and other related tasks, making it easier for developers to implement and experiment with segmentation algorithms.

CHAPTER 4

SYSTEM DESIGN

CHAPTER 4

SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

The system adopts a modular architecture for brain tumor segmentation, employing a U-Net deep learning model. The system begins by loading raw T1CE, T2, and FLAIR MRI data. A preprocessing module then cleans and prepares the data through skull stripping, intensity normalization, and co-registration to ensure consistency across modalities. This preprocessed data is then fed into the U-Net model, which extracts features at different scales and combines them to generate a pixel-wise segmentation mask that identifies tumor regions within the brain. Finally, evaluation metrics like Jaccard coefficient assess the model's segmentation accuracy. This modular architecture with a U-Net model facilitates efficient training, leverages multi-modal information for comprehensive tumor segmentation, and offers a flexible framework for potential future enhancements.

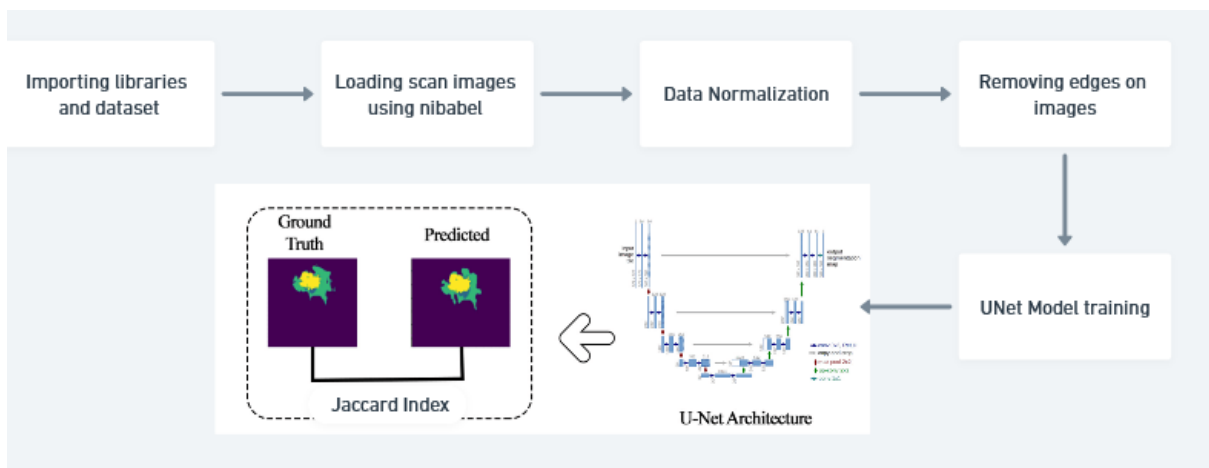


Fig 4.1 System architecture for UNet model

4.2 UML DIAGRAM

4.2.1 USE CASE DIAGRAM

Use Cases are used to describe the visible interactions that the system will have with users and external systems. The use case for brain tumor segmentation involves the application of advanced image processing techniques to accurately delineate brain tumors from medical imaging scans, such as MRI or CT images. This segmentation process enables clinicians to diagnose and characterize brain tumors with greater precision, facilitating treatment planning, monitoring disease progression, and evaluating treatment response. Additionally, brain tumor segmentation supports clinical research endeavors by providing valuable insights into tumor biology, prognostic factors, and therapeutic targets.

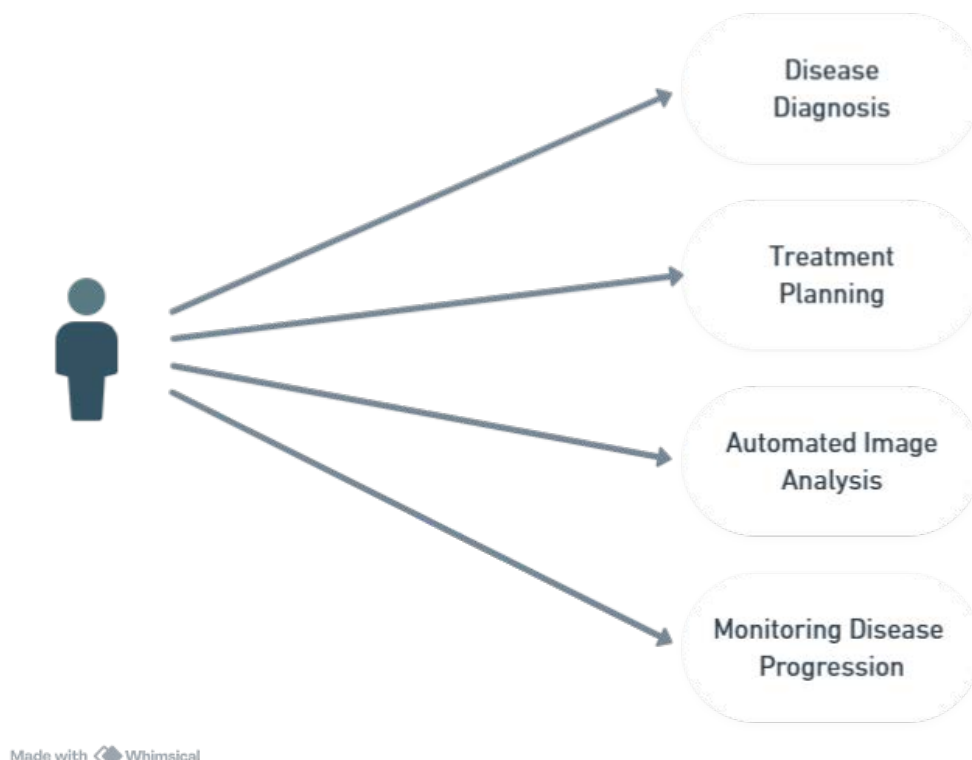


Fig 4.2.1 Use case diagram for Brain Tumor Segmentation

4.2.2 ACTIVITY DIAGRAM

The Activity Diagram illustrates the flow of activities within the proposed system. . This diagram provides a dynamic view, highlighting the sequence and conditions of activities, aiding in the understanding of system workflows. The process begins with the user providing an image for analysis. Once preprocessed, the image is fed into the trained model to generate a prediction. The model utilizes its learned parameters to make an inference based on the provided image. Subsequently, the prediction result is displayed, showcasing the outcome determined by the model. Finally, the process concludes, having successfully analyzed the provided image and generated a prediction based on the trained model's assessment.

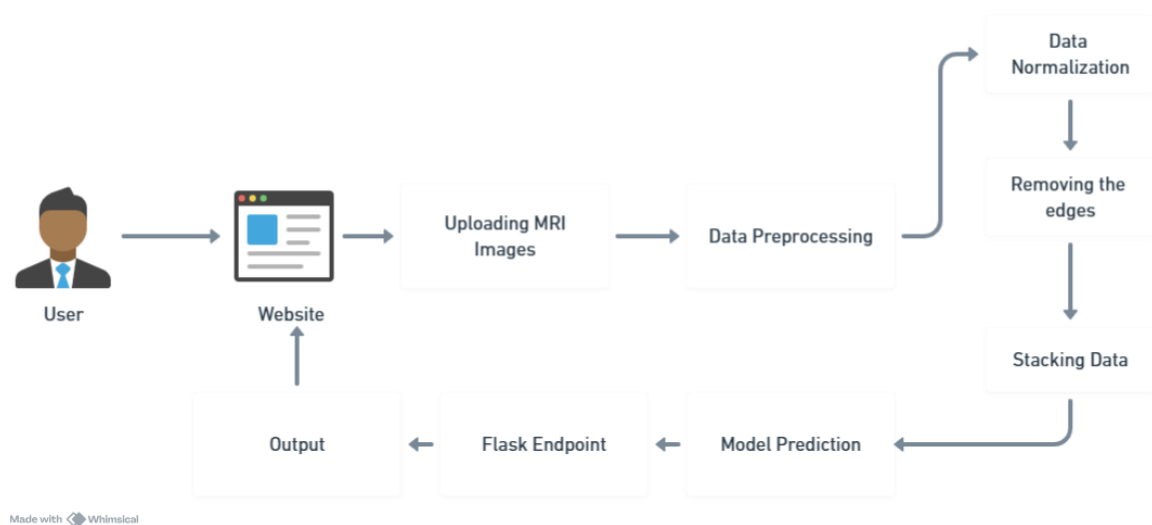


Fig 4.2.2 Activity diagram for Brain Tumor Segmentation

4.2.3 SEQUENCE DIAGRAM

The Sequence Diagram delves into the interactions between different components of the system during specific processes. The sequence diagram illustrates the flow of interactions , beginning with the user providing an image,

which triggers the processing of the uploaded image function. This function preprocesses the image, ensuring it is appropriately formatted for the model. Subsequently, the preprocessed image is fed into the trained model, which generates a prediction based on its learned parameters. Once the prediction is obtained, it is displayed to the user, completing the sequence of interactions in the system workflow.

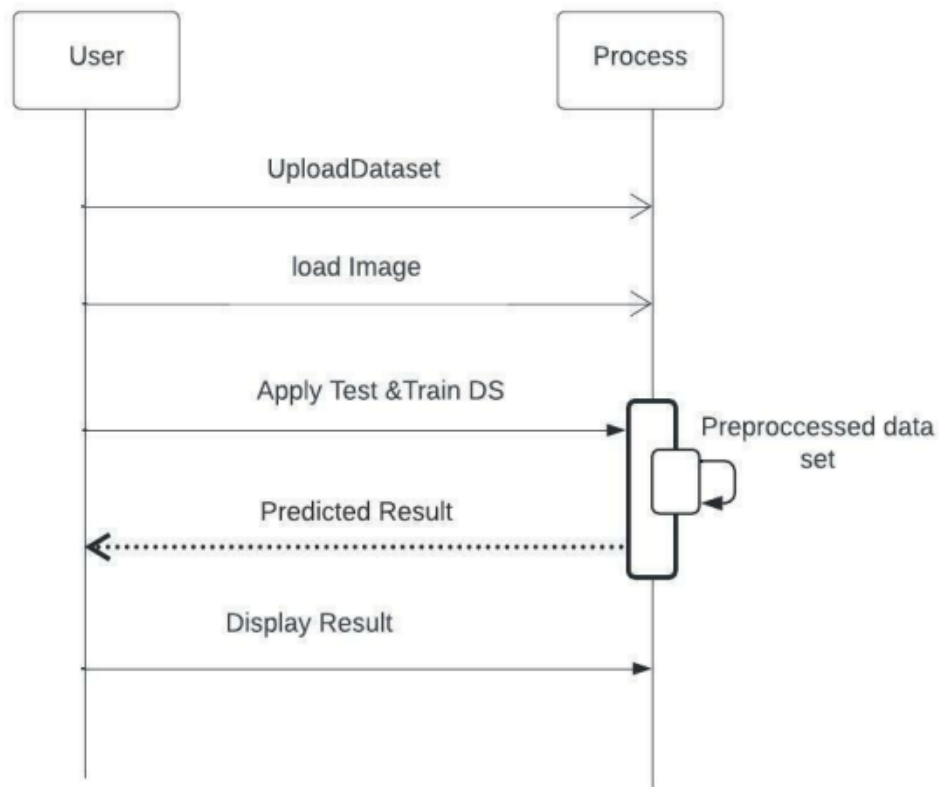


Fig 4.2.3 Sequence diagram for Brain Tumor Segmentation

CHAPTER 5

SYSTEM IMPLEMENTATION

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 LIST OF MODULES

1. DATA LOADING & UNDERSTANDING
2. DATA PREPROCESSING & SELECTION
3. MODEL TRAINING
4. WEB PAGE INTEGRATION

5.2 MODULE DESCRIPTION

The system leverages a modular approach, guiding the brain tumor segmentation process through four key stages. The first stage involves data loading and understanding, where raw MRI data from various modalities is loaded and analyzed to grasp the clear understanding about the data. Next, the data preprocessing and selection module takes center stage. Here, the data is meticulously prepared for training by applying techniques like intensity normalization, resizing and assigning the mask value. And selecting a high information modalities among the different modalities. The third module focuses on model training, where a deep learning architecture like U-Net is trained on the preprocessed data. Lastly, the evaluation metrics module assesses the trained model's performance on a separate validation dataset. Metrics like Jaccard coefficient and sensitivity quantify segmentation accuracy, while visualization tools allow for qualitative comparison of predicted and ground truth masks.

5.2.1 DATA LOADING & UNDERSTANDING

This module loads raw MRI data (T1CE, T2, FLAIR) from various sources. It performs initial exploration to grasp the clear understanding about the data. The raw MRI data is stored in (.nii) file format which is commonly used medical imaging format to store brain imaging data obtained using MRI. Each modalities provide different information about the brain. The multiple modalities is loaded using nibabel, a python module which provides read and write access to neuroimaging file formats. Finally, loading the patient's multiple modalities and visualizing with different features provides more information about the brain and the data. Each image contains a 255 features. Visualizing the MRI scans displays the scanned images of the brain and mask images displays a tumor segmentation over the brain.

5.2.2 DATA PREPROCESSING & SELECTION

This module prepares the data for training by applying techniques like intensity normalization, cropping the 3D image from 255 to 128 where cropping removes empty borders and changing mask pixel values (labels) from 4 to 3. Selecting the modalities where different modalities provide different information such as tissue, fluids, and tumor. T1ce , an version of T1 where a contrast agent gadolinium is injected over the blood which highlights tumor regions due to their typically increased vascularization and disruption of the blood-brain barrier. Stacking three MRI scans into a single 3D image volume. Finally, saving preprocessed image and mask as NumPy arrays.

5.2.3 MODEL TRAINING

This module employs a deep learning architecture like U-Net for training. It was designed for image segmentation in the biomedical field. It consists of a contraction path and an expansion path with skip connections. In contracting path feature maps get spatially smaller, whereas in an expanding path, the feature maps are expanded back to their original size. From the top to the bottom of the encoder, the number of feature maps increases and the size of feature maps reduces. The top of the decoder is the output layer consisting of a convolution layer and a sigmoid activation function.

The improved U-Net model has a deeper architecture with more filters in its convolutional layers and includes batch normalization after each convolutional layer, which can potentially enhance training stability and convergence speed compared to the simple U-Net model.

5.2.4 WEB PAGE INTEGRATION

The web page integration module enables user interaction through an interface built with HTML and CSS. Users can upload MRI scans (T1CE, T2, FLAIR) for segmentation, potentially receive visual feedback during processing, and view the final results with the original MRI alongside the generated segmentation mask. HTML structures the page layout, while CSS styles the interface for a user-friendly experience. Integration with a backend (optional) allows processing of uploaded data and retrieval of segmentation results, potentially using form submission mechanisms. Future enhancements could include user authentication, result history, or more informative visualizations.

CHAPTER 6

TESTING

CHAPTER 6

TESTING

Testing is the process of executing a program or application with the intent of finding software bugs, and to verify that the software product is fit for use.

6.1 UNIT TESTING

Unit testing is an essential component of software development, ensuring that individual units of code function correctly in isolation. For our brain tumor segmentation project, unit testing will be conducted to verify the functionality of key modules and components.

1. **Data Loading & Understanding Module:** Unit tests will validate that the module correctly loads raw MRI data from various sources, including T1CE, T2, and FLAIR modalities. Tests will verify that the data is loaded in the expected format (e.g., NIfTI format), and initial exploration functions accurately extract relevant metadata and characteristics from the MRI scans.
2. **Data Preprocessing & Selection Module:** Unit tests will ensure that preprocessing techniques such as intensity normalization, cropping, and modality selection are applied correctly. Tests will validate that the preprocessing steps produce the expected output, such as resized images with normalized intensities and appropriate label values for tumor segmentation.
3. **Model Training Module:** Unit tests will verify the functionality of the model training process, including the instantiation of the U-Net architecture, compilation with appropriate loss functions and optimizers, and execution of training loops. Tests will check that the model trains without errors and that key performance metrics (e.g., loss, accuracy) are

within expected ranges during training.

4. **Webpage Integration:** Unit tests will ensure seamless integration of the system with web interfaces. Tests will validate that the webpages correctly display input forms, handle user interactions, and process uploaded data for further analysis. Additionally, tests will verify that the generated outputs are accurately presented to users, maintaining consistency with the underlying model predictions and segmentation results.

6.2 INTEGRATION TESTING

Integration testing ensures that individual modules or components of a system work together seamlessly as a cohesive unit. In the context of your brain tumor segmentation project, integration testing would involve verifying the interaction and interoperability between the different modules. This includes testing data flow between modules, ensuring correct preprocessing and selection of MRI data, validating model training with preprocessed data, and assessing the performance of evaluation metrics on the trained models. By conducting integration testing, you can ensure that all modules function correctly when integrated into the overall system, leading to a robust and reliable brain tumor segmentation solution.

6.3 SYSTEM TESTING

System testing involves rigorous validation to ensure its functionality, reliability, and accuracy. This process encompasses various stages, including unit testing, integration testing, and end-to-end testing. This phase validates the system's ability to handle diverse inputs, adapt to different scenarios, and produce accurate segmentation results. Additionally, the testing process involves validating the system's robustness against outliers, noise, and variations in input data. It also includes stress testing to assess the system's performance under high load and resource constraints.

CHAPTER 7

RESULTS AND DISCUSSION

CHAPTER 7

RESULTS AND DISCUSSION

7.1 RESULTS

The model proposed was constructed using the Keras library, leveraging TensorFlow as its backend. During optimization, the ADAM optimizer was selected with a conservative learning rate set to 0.0001. For stability and normalization during the training process, the ReLU activation function combined with batch normalization techniques was employed. The model was trained over a span of 100 epochs, utilizing a batch size of 2 to accommodate computational constraints effectively. Upon evaluation, the models exhibited promising outcomes. Specifically, the Simple UNet architecture achieved an intersection over union (IOU) score of 0.8123, indicating substantial performance. Notably, the Improved UNet model demonstrated even more remarkable results, boasting an IOU score of 0.8502. These findings underscore the potential of deep learning approaches in advancing medical image analysis, particularly in the domain of brain tumor segmentation.

7.2 DISCUSSION

By amalgamating T1CE, T2, and FLAIR MRI images, the system achieved a more nuanced understanding of brain tissue and tumor characteristics, surpassing the insights gleaned from individual modalities alone. T1CE sequences, renowned for their capacity to highlight contrast agent enhancement, effectively delineated tumor regions by accentuating increased vascularization and disruption of the blood-brain barrier. T2-weighted images enriched the analysis by providing crucial information on water content, facilitating the differentiation of tumor tissue from surrounding structures. FLAIR images, akin to T2-weighted images but with cerebrospinal fluid (CSF) signal suppression, further contributed to the differentiation of tumors from fluid-filled regions. Through the holistic integration of information from all three modalities, the model likely achieved heightened segmentation accuracy, leveraging the complementary details offered by each imaging technique to capture a comprehensive representation of tumor morphology and surrounding brain tissue characteristics.

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

CHAPTER 8

CONCNLUSION AND FUTURE ENHANCEMENT

8.1 CONCLUSION

The system has effectively formulated a method for brain tumor segmentation by harnessing the potential of multiple MRI modalities, including T1CE, T2, and FLAIR. Through the amalgamation of information from these complementary imaging techniques, the model has demonstrated remarkable proficiency in precisely delineating brain tumors, as evidenced by its outstanding performance on the validation dataset. This success underscores the transformative impact of deep learning methodologies on medical image analysis, offering clinicians and researchers a powerful tool for automating tumor segmentation with high precision. By leveraging diverse modalities, the system represents a significant stride toward more accurate and efficient diagnostic processes, holding promise for enhancing patient care outcomes and advancing the field of neuroimaging.

8.2 FUTURE ENHANCEMENT

For future enhancements, additional MRI modalities or imaging techniques can be integrated to capture a more comprehensive view of brain tumor characteristics, potentially improving segmentation accuracy. Data augmentation and transfer learning strategies may enhance model generalization, while real-time processing capabilities could facilitate clinical deployment for timely and accurate tumor segmentation. Collaboration with medical professionals to collect annotated datasets and validate the system's performance in clinical settings would ensure its efficacy and relevance in real-world scenarios, advancing the field of medical image analysis and improving patient care in neuro-oncology.

CHAPTER 9

ANNEXURE

ANNEXURE

APPENDIX 1

SOURCE CODE

```
import numpy as np
import nibabel as nib
import glob
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler

TRAIN_PATH = 'BraTS2020_TrainingData/MICCAI_BraTS2020
               _TrainingData/'
test_image_flair=nib.load(TRAIN_PATH + 'BraTS20_Training_355/
                           BraTS20_Training_355_flair.nii').get_fdata()
test_image_flair=scaler.fit_transform(test_image_flair.reshape(-1,
test_image_flair.shape[-1])).reshape(test_image_flair.shape)
test_image_t1=nib.load(TRAIN_PATH + 'BraTS20_Training_355/
                           BraTS20_Training_355_t1.nii').get_fdata()
test_image_t1=scaler.fit_transform(test_image_t1.reshape(-1,
test_image_t1.shape[-1])).reshape(test_image_t1.shape)
test_image_t1ce=nib.load(TRAIN_PATH + 'BraTS20_Training_355/
                           BraTS20_Training_355_t1ce.nii').get_fdata()
test_image_t1ce=scaler.fit_transform(test_image_t1ce.reshape(-1,
test_image_t1ce.shape[-1])).reshape(test_image_t1ce.shape)
test_image_t2=nib.load(TRAIN_PATH + 'BraTS20_Training_355/
                           BraTS20_Training_355_t2.nii').get_fdata()
test_image_t2=scaler.fit_transform(test_image_t2.reshape(-1,
test_image_t2.shape[-1])).reshape(test_image_t2.shape)
test_mask=nib.load(TRAIN_PATH + 'BraTS20_Training_355/
                           BraTS20_Training_355_seg.nii').get_fdata()
test_mask=test_mask.astype(np.uint8)
print(np.unique(test_mask)) #0, 1, 2, 4 (Need to reencode to 0, 1, 2, 3)
test_mask[test_mask==4] = 3 #Reassign mask values 4 to 3
print(np.unique(test_mask))

import random
n_slice=random.randint(0, test_mask.shape[2])
plt.figure(figsize=(12, 8))
plt.subplot(231)
```

```

plt.imshow(test_image_flair[:, :, n_slice], cmap='gray')
plt.title('Image flair')
plt.subplot(232)
plt.imshow(test_image_t1[:, :, n_slice], cmap='gray')
plt.title('Image t1')
plt.subplot(233)
plt.imshow(test_image_t1ce[:, :, n_slice], cmap='gray')
plt.title('Image t1ce')
plt.subplot(234)
plt.imshow(test_image_t2[:, :, n_slice], cmap='gray')
plt.title('Image t2')
plt.subplot(235)
plt.imshow(test_mask[:, :, n_slice])
plt.title('Mask')
plt.show()

t2_list = sorted(glob.glob('BraTS2020_TrainingData/MICCAI_BraTS2020_
                          TrainingData/*/*t2.nii'))
t1ce_list = sorted(glob.glob('BraTS2020_TrainingData/MICCAI_BraTS2020_
                          TrainingData/*/*t1ce.nii'))
flair_list = sorted(glob.glob('BraTS2020_TrainingData/MICCAI_BraTS2020_
                          TrainingData/*/*flair.nii'))
mask_list = sorted(glob.glob('BraTS2020_TrainingData/MICCAI_BraTS2020_
                          TrainingData/*/*seg.nii'))
for img in range(len(t2_list)): #Using t1_list as all lists are of same size
    temp_image_t2=nib.load(t2_list[img]).get_fdata()
    temp_image_t2=scaler.fit_transform(temp_image_t2.reshape(-1,
temp_image_t2.shape[-1])).reshape(temp_image_t2.shape)

    temp_image_t1ce=nib.load(t1ce_list[img]).get_fdata()
    temp_image_t1ce=scaler.fit_transform(temp_image_t1ce.reshape(-1,
temp_image_t1ce.shape[-1])).reshape(temp_image_t1ce.shape)
    temp_image_flair=nib.load(flair_list[img]).get_fdata()
    temp_image_flair=scaler.fit_transform(temp_image_flair.reshape(-1,
temp_image_flair.shape[-1])).reshape(temp_image_flair.shape)
    temp_mask=nib.load(mask_list[img]).get_fdata()
    temp_mask=temp_mask.astype(np.uint8)
    temp_mask[temp_mask==4] = 3
    temp_combined_images = np.stack([temp_image_flair, temp_image_t1ce,
temp_image_t2], axis=3)
    temp_combined_images=temp_combined_images[56:184, 56:184, 13:141]
    temp_mask = temp_mask[56:184, 56:184, 13:141]
    val, counts = np.unique(temp_mask, return_counts=True)

```

```

temp_mask= to_categorical(temp_mask, num_classes=4)

np.save('BraTS2020_TrainingData/input_data_3channels/images/image_'+str(i
mg)+'.npy', temp_combined_images)
np.save('BraTS2020_TrainingData/input_data_3channels/masks/mask_'+str(im
g)+'.npy', temp_mask)

import splitfolders
input_folder = 'BraTS2020_TrainingData/input_data_3channels/'
output_folder = 'BraTS2020_TrainingData/input_data_128/'
splitfolders.ratio(input_folder, output=output_folder, seed=42, ratio=(.75, .25),
group_prefix=None)

import os
import numpy as np
def load_img(img_dir, img_list):
    images=[]
    for i, image_name in enumerate(img_list):
        if (image_name.split('.')[1] == 'npy'):
            image = np.load(img_dir+image_name)
            images.append(image)
    images = np.array(images)
    return(images)

def imageLoader(img_dir, img_list, mask_dir, mask_list, batch_size):
    L = len(img_list)
    while True:
        batch_start = 0
        batch_end = batch_size
        while batch_start < L:
            limit = min(batch_end, L)
            X = load_img(img_dir, img_list[batch_start:limit])
            Y = load_img(mask_dir, mask_list[batch_start:limit])
            yield (X,Y)
            batch_start += batch_size
            batch_end += batch_size
from keras.models import Model
from keras.layers import Input, Conv3D, MaxPooling3D, concatenate,
Conv3DTranspose, BatchNormalization, Dropout, Lambda
from keras.optimizers import Adam
from keras.metrics import MeanIoU
kernel_initializer = 'he_uniform'

```

```

def simple_unet_model(IMG_HEIGHT, IMG_WIDTH, IMG_DEPTH,
IMG_CHANNELS, num_classes):
    inputs = Input((IMG_HEIGHT, IMG_WIDTH, IMG_DEPTH,
IMG_CHANNELS))
    s = inputs
    c1 = Conv3D(16, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(s)
    c1 = Dropout(0.1)(c1)
    c1 = Conv3D(16, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(c1)
    p1 = MaxPooling3D((2, 2, 2))(c1)

    c2 = Conv3D(32, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(p1)
    c2 = Dropout(0.1)(c2)
    c2 = Conv3D(32, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(c2)
    p2 = MaxPooling3D((2, 2, 2))(c2)

    c3 = Conv3D(64, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(p2)
    c3 = Dropout(0.2)(c3)
    c3 = Conv3D(64, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(c3)
    p3 = MaxPooling3D((2, 2, 2))(c3)

    c4 = Conv3D(128, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(p3)
    c4 = Dropout(0.2)(c4)
    c4 = Conv3D(128, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(c4)
    p4 = MaxPooling3D(pool_size=(2, 2, 2))(c4)

    c5 = Conv3D(256, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(p4)
    c5 = Dropout(0.3)(c5)
    c5 = Conv3D(256, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(c5)

    #Expansive path
    u6 = Conv3DTranspose(128, (2, 2, 2), strides=(2, 2, 2), padding='same')(c5)
    u6 = concatenate([u6, c4])

```

```

    c6 = Conv3D(128, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(u6)
    c6 = Dropout(0.2)(c6)
    c6 = Conv3D(128, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(c6)

    u7 = Conv3DTranspose(64, (2, 2, 2), strides=(2, 2, 2), padding='same')(c6)
    u7 = concatenate([u7, c3])
    c7 = Conv3D(64, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(u7)
    c7 = Dropout(0.2)(c7)
    c7 = Conv3D(64, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(c7)

    u8 = Conv3DTranspose(32, (2, 2, 2), strides=(2, 2, 2), padding='same')(c7)
    u8 = concatenate([u8, c2])
    c8 = Conv3D(32, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(u8)
    c8 = Dropout(0.1)(c8)
    c8 = Conv3D(32, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(c8)

    u9 = Conv3DTranspose(16, (2, 2, 2), strides=(2, 2, 2), padding='same')(c8)
    u9 = concatenate([u9, c1])
    c9 = Conv3D(16, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(u9)
    c9 = Dropout(0.1)(c9)
    c9 = Conv3D(16, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(c9)
    outputs = Conv3D(num_classes, (1, 1, 1), activation='softmax')(c9)
    model = Model(inputs=[inputs], outputs=[outputs])
    model.summary()
    return model

def improved_unet_model(IMG_HEIGHT, IMG_WIDTH, IMG_DEPTH,
IMG_CHANNELS, num_classes):

    inputs = Input((IMG_HEIGHT, IMG_WIDTH, IMG_DEPTH,
IMG_CHANNELS))
    s = inputs

    #Contraction path

```

```

c1 = Conv3D(32, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(s)
c1 = BatchNormalization()(c1)
c1 = Conv3D(32, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(c1)
c1 = BatchNormalization()(c1)
p1 = MaxPooling3D((2, 2, 2))(c1)

c2 = Conv3D(64, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(p1)
c2 = BatchNormalization()(c2)
c2 = Conv3D(64, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(c2)
c2 = BatchNormalization()(c2)
p2 = MaxPooling3D((2, 2, 2))(c2)

c3 = Conv3D(128, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(p2)
c3 = BatchNormalization()(c3)
c3 = Conv3D(128, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(c3)
c3 = BatchNormalization()(c3)
p3 = MaxPooling3D((2, 2, 2))(c3)

c4 = Conv3D(256, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(p3)
c4 = BatchNormalization()(c4)
c4 = Conv3D(256, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(c4)
c4 = BatchNormalization()(c4)
drop4 = Dropout(0.5)(c4)
p4 = MaxPooling3D((2, 2, 2))(drop4)

c5 = Conv3D(512, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(p4)
c5 = BatchNormalization()(c5)
c5 = Conv3D(512, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(c5)
c5 = BatchNormalization()(c5)
drop5 = Dropout(0.5)(c5)

#Expansion path
u6 = Conv3DTranspose(128, (2, 2, 2), strides=(2, 2, 2), padding='same')(c5)

```

```

u6 = concatenate([u6, c4])
c6 = Conv3D(256, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(u6)
c6 = BatchNormalization()(c6)
c6 = Dropout(0.2)(c6)
c6 = Conv3D(256, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(c6)
c6 = BatchNormalization()(c6)

u7 = Conv3DTranspose(64, (2, 2, 2), strides=(2, 2, 2), padding='same')(c6)
u7 = concatenate([u7, c3])
c7 = Conv3D(128, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(u7)
c7 = BatchNormalization()(c7)
c7 = Dropout(0.2)(c7)
c7 = Conv3D(128, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(c7)
c7 = BatchNormalization()(c7)

u8 = Conv3DTranspose(32, (2, 2, 2), strides=(2, 2, 2), padding='same')(c7)
u8 = concatenate([u8, c2])
c8 = Conv3D(64, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(u8)
c8 = BatchNormalization()(c8)
c8 = Dropout(0.2)(c8)
c8 = Conv3D(64, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(c8)
c8 = BatchNormalization()(c8)

u9 = Conv3DTranspose(16, (2, 2, 2), strides=(2, 2, 2), padding='same')(c8)
u9 = concatenate([u9, c1])
c9 = Conv3D(32, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(u9)
c9 = BatchNormalization()(c9)
c9 = Dropout(0.2)(c9)
c9 = Conv3D(32, (3, 3, 3), activation='relu',
kernel_initializer=kernel_initializer, padding='same')(c9)
c9 = BatchNormalization()(c9)

outputs = Conv3D(num_classes, (1, 1, 1), activation='softmax')(c9)
model = Model(inputs=[inputs], outputs=[outputs])

train_img_dir = "BraTS2020_TrainingData/input_data_128/train/images/"

```



```

train_mask_dir = "BraTS2020_TrainingData/input_data_128/train/masks/"
val_img_dir = "BraTS2020_TrainingData/input_data_128/val/images/"
val_mask_dir = "BraTS2020_TrainingData/input_data_128/val/masks/"

train_img_list=os.listdir(train_img_dir)
train_mask_list = os.listdir(train_mask_dir)
val_img_list=os.listdir(val_img_dir)
val_mask_list = os.listdir(val_mask_dir)
batch_size = 2

train_img_datagen = imageLoader(train_img_dir, train_img_list,
                                train_mask_dir, train_mask_list, batch_size)

val_img_datagen = imageLoader(val_img_dir, val_img_list,
                              val_mask_dir, val_mask_list, batch_size)

wt0, wt1, wt2, wt3 = 0.25,0.25,0.25,0.25
import segmentation_models_3D as sm
dice_loss = sm.losses.DiceLoss(class_weights=np.array([wt0, wt1, wt2, wt3]))
focal_loss = sm.losses.CategoricalFocalLoss()
total_loss = dice_loss + (1 * focal_loss)
metrics = ['accuracy', sm.metrics.IOUScore(threshold=0.5)]
LR = 0.0001
optim = keras.optimizers.Adam(LR)
steps_per_epoch = len(train_img_list)//batch_size
val_steps_per_epoch = len(val_img_list)//batch_size

model = simple_unet_model(IMG_HEIGHT=128, IMG_WIDTH=128,
                          IMG_DEPTH=128,IMG_CHANNELS=3,
                          num_classes=4)
model = improved_unet_model(IMG_HEIGHT=128, IMG_WIDTH=128,
                             IMG_DEPTH=128,IMG_CHANNELS=3,num_classes=4)
model.compile(optimizer = optim, loss=total_loss, metrics=metrics)

history=model.fit(train_img_datagen,
                  steps_per_epoch=steps_per_epoch,
                  epochs=100,
                  verbose=1,
                  validation_data=val_img_datagen,
                  validation_steps=val_steps_per_epoch,
                  )

model.save(unet_brain_tumor.hdf5')

```

```

loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'y', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title("Training and validation loss")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

```

```

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
plt.plot(epochs, acc, 'y', label='Training accuracy')
plt.plot(epochs, val_acc, 'r', label='Validation accuracy')
plt.title("Training and validation accuracy")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

```

```

from keras.metrics import MeanIoU
batch_size=8
test_img_datagen = imageLoader(val_img_dir, val_img_list,
                                val_mask_dir, val_mask_list, batch_size)
test_image_batch, test_mask_batch = test_img_datagen.__next__()
test_mask_batch_argmax = np.argmax(test_mask_batch, axis=4)
test_pred_batch = my_model.predict(test_image_batch)
test_pred_batch_argmax = np.argmax(test_pred_batch, axis=4)
n_classes = 4
IOU_keras = MeanIoU(num_classes=n_classes)
IOU_keras.update_state(test_pred_batch_argmax, test_mask_batch_argmax)
print("Mean IoU =", IOU_keras.result().numpy())
img_num = 82

```

```

test_img = np.load("BraTS2020_TrainingData/input_data_128/
                    val/images/image_"+str(img_num)+".npy")

test_mask = np.load("BraTS2020_TrainingData/input_data_128/
                    val/masks/mask_"+str(img_num)+".npy")
test_mask_argmax=np.argmax(test_mask, axis=3)
test_img_input = np.expand_dims(test_img, axis=0)
test_prediction = my_model.predict(test_img_input)

```

```

test_prediction_argmax=np.argmax(test_prediction, axis=4)[0,:,:,:]
n_slice = 55
plt.figure(figsize=(12, 8))
plt.subplot(231)
plt.title('Testing Image')
plt.imshow(test_img[:, :, n_slice, 1], cmap='gray')
plt.subplot(232)
plt.title('Testing Label')
plt.imshow(test_mask_argmax[:, :, n_slice])
plt.subplot(233)
plt.title('Prediction on test image')
plt.imshow(test_prediction_argmax[:, :, n_slice])
plt.show()

```

index.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Brain Tumor Segmentation</title>
</head>
<body>
  <h1>Upload Brain MRI Images</h1>
  <form action="/upload" method="post" enctype="multipart/form-data"
class="upload-form">
    <div class="file-upload">
      <label for="flair">FLAIR:</label>
      <input type="file" name="flair" id="flair" accept=".nii, .nii.gz">
    </div>
    <div class="file-upload">
      <label for="t1ce">T1CE:</label>
      <input type="file" name="t1ce" id="t1ce" accept=".nii, .nii.gz">
    </div>
    <div class="file-upload">
      <label for="t2">T2:</label>
      <input type="file" name="t2" id="t2" accept=".nii, .nii.gz">
    </div>
    <button type="submit">Submit</button>
  </form>
</body>

```

```
</html>
```

result.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Brain Tumor Segmentation Result</title>
</head>
<body>
  <h1>Brain Tumor Segmentation Result</h1>
  <div class="image-container">
    
    
  </div>
</body>

<style>
  img{
    height: 350px;
    width: 350px;
  }
</style>
</html>
```

app.py

```
import os
import nibabel as nib
import numpy as np
from flask import Flask, render_template, request, redirect, url_for
from unet import unet_model
from matplotlib import pyplot as plt
from PIL import Image as im
import nilearn
import nilearn.plotting as nlplt

app = Flask(__name__)
```

```

UPLOAD_FOLDER = 'uploads'
RESULT_FOLDER = 'results'
ALLOWED_EXTENSIONS = {'nii', 'nii.gz'}

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['RESULT_FOLDER'] = RESULT_FOLDER

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in
ALLOWED_EXTENSIONS

@app.route('/')
def index():
    return render_template('index.html')
@app.route('/upload', methods=['POST'])
def upload():
    if 'flair' not in request.files or 't1ce' not in request.files or 't2' not in
request.files:
        return redirect(request.url)

    flair = request.files['flair']
    t1ce = request.files['t1ce']
    t2 = request.files['t2']

    if flair.filename == "" or t1ce.filename == "" or t2.filename == "":
        return redirect(request.url)

    if flair and allowed_file(flair.filename) and t1ce and
allowed_file(t1ce.filename) and t2 and allowed_file(t2.filename):
        # Save uploaded files
        flair_file = os.path.join(app.config['UPLOAD_FOLDER'], flair.filename)
        t1ce_file = os.path.join(app.config['UPLOAD_FOLDER'], t1ce.filename)
        t2_file = os.path.join(app.config['UPLOAD_FOLDER'], t2.filename)

        flair.save(flair_file)
        t1ce.save(t1ce_file)
        t2.save(t2_file)

        # Load NIfTI files using nibabel
        flair_img = nib.load(flair_file).get_fdata()
        t1ce_img = nib.load(t1ce_file).get_fdata()
        t2_img = nib.load(t2_file).get_fdata()

```

```

# Stack the MRI modalities along the fourth dimension
mri_stack = np.stack([flair_img, t1ce_img, t2_img], axis=3)
mri_stack = mri_stack[56:184, 56:184, 13:141] # Adjust the cropping as
needed

# Process the MRI stack (e.g., perform segmentation)
my_model = unet_model()
my_model.load_weights('dhanesh_2batch_30epoch.weights.h5')

test_img_input = np.expand_dims(mri_stack, axis=0)
test_prediction = my_model.predict(test_img_input)
test_prediction_argmax=np.argmax(test_prediction, axis=4)[0,:,:,:]
n_slice = 55
array = test_prediction_argmax[:, :, n_slice]

plt.imsave('static/results/mri.png',mri_stack[:, :, n_slice,0])
plt.imsave('static/results/output.png',array)
return redirect(url_for('result', filename='output.png'))

@app.route('/result/<filename>')
def result(filename):

    return render_template('result.html', segmented_image=url_for('static',
filename='results/' + filename),
                        mri_image=url_for('static', filename='results/' + 'mri.png' ))

if __name__ == '__main__':
    app.run(debug=True)

```

APPENDIX 2

SCREENSHOTS

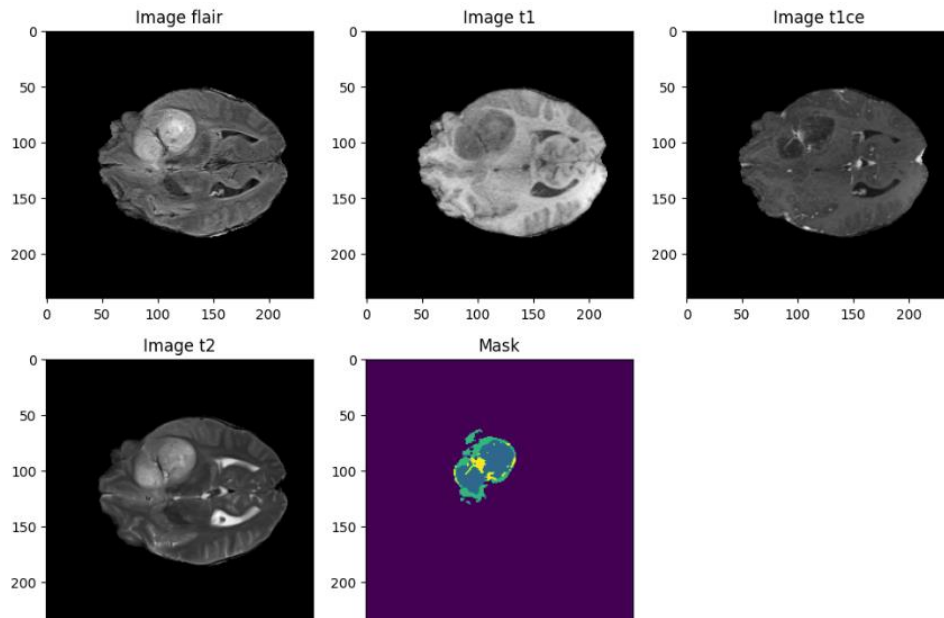


Fig 9.1 Visualizing the MRI Scan images

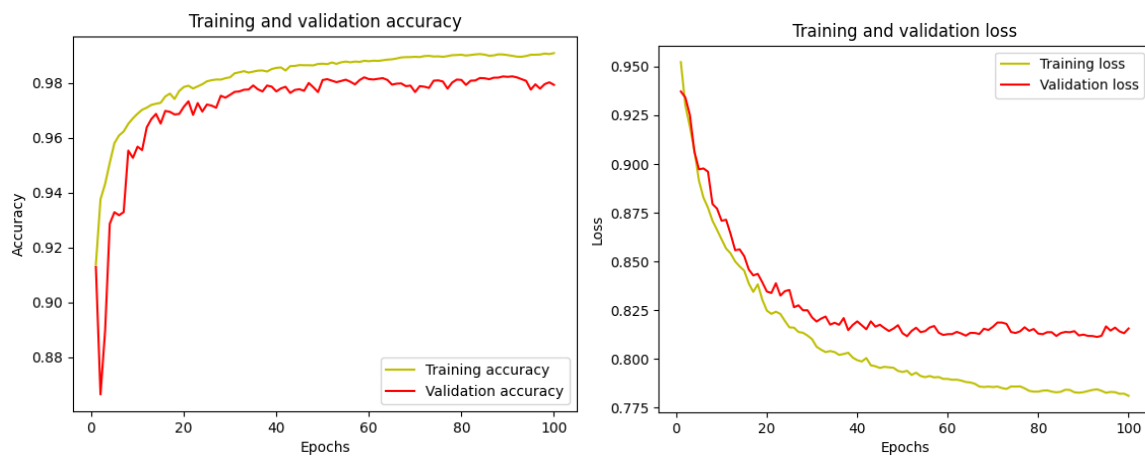


Fig 9.2.1 Simple UNet model loss and Accuracy

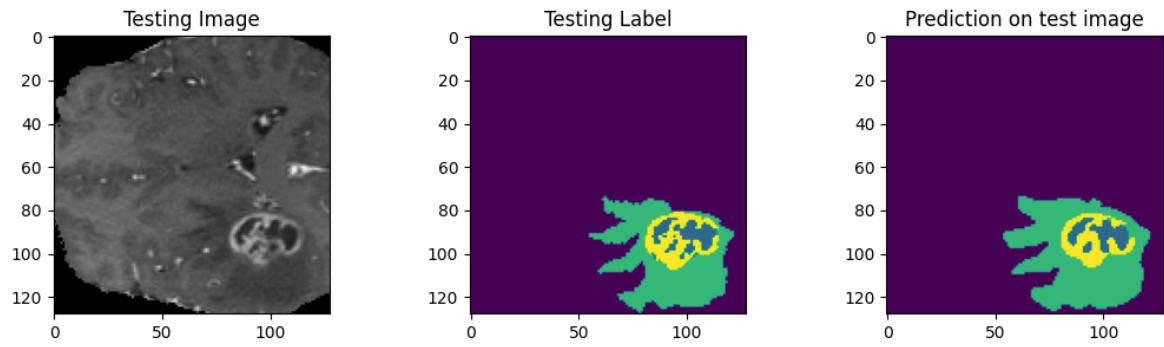


Fig 9.2.2 Simple UNet Predicition

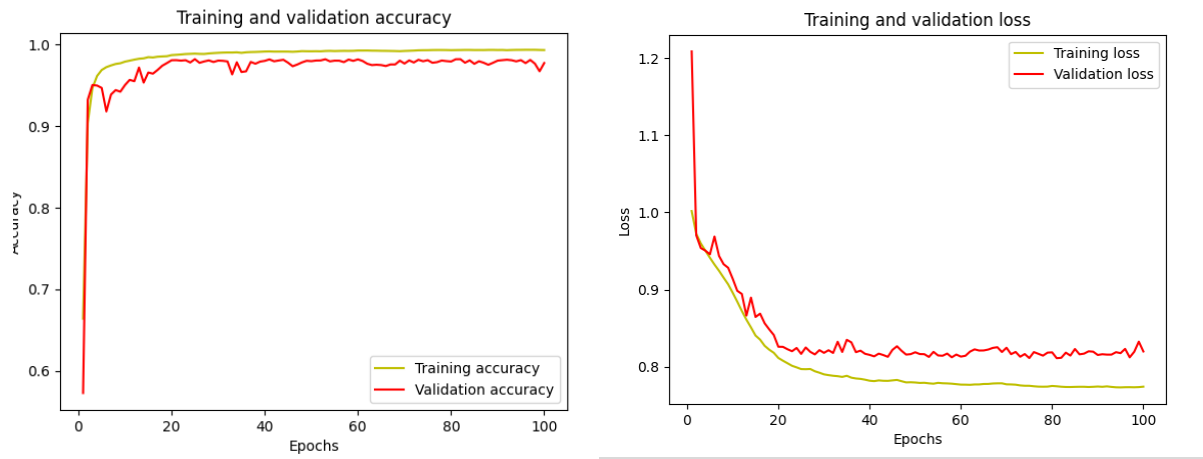


Fig 9.3.1 Improved UNet model loss and Accuracy

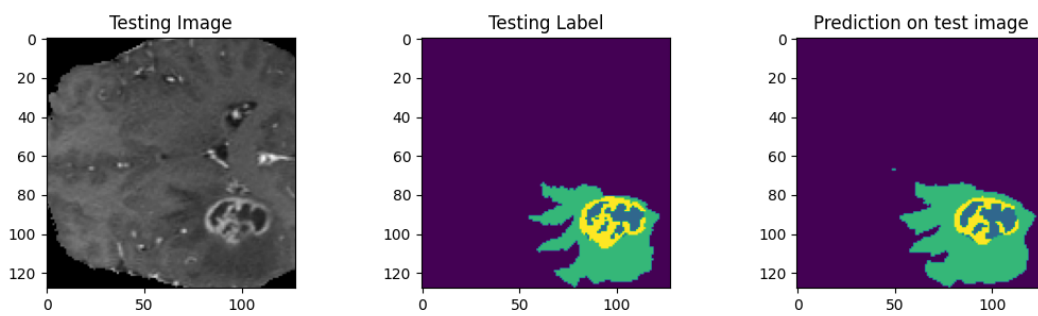


Fig 9.3.2 Improved UNet Predicition

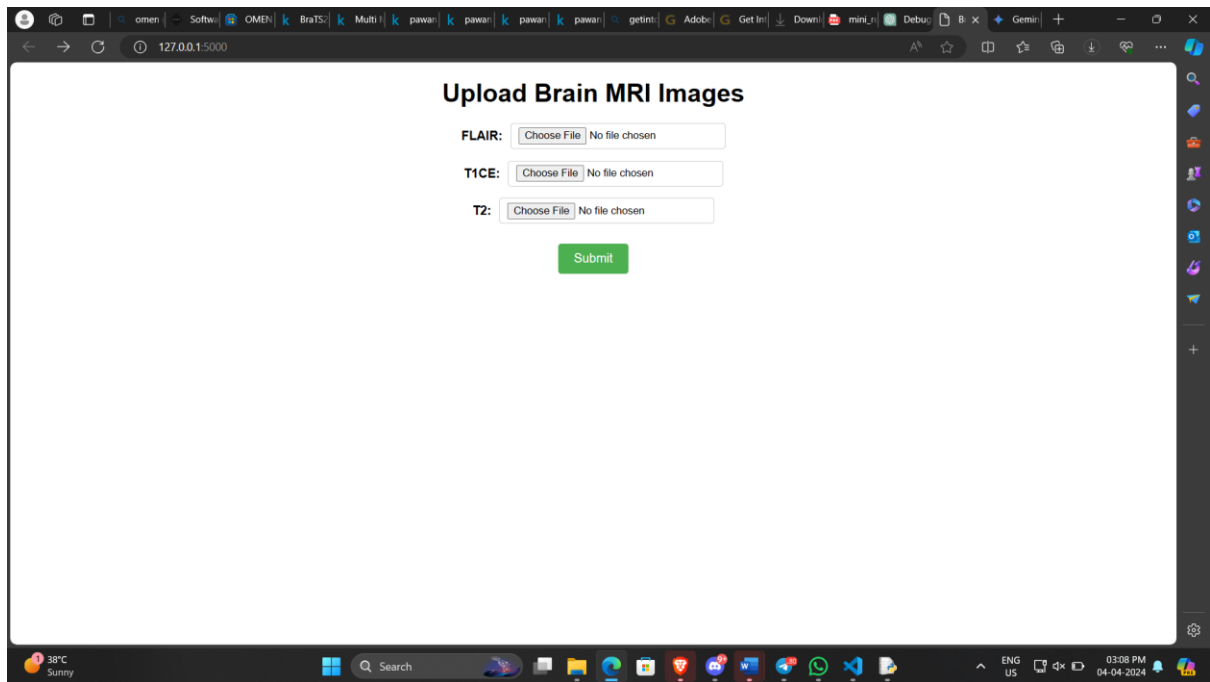


Fig 9.4.1 Web Interface

Brain Tumor Segmentation Result

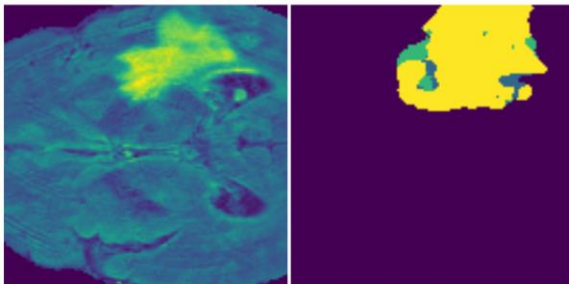


Fig 9.4.2 Segmentation of the brain tumor

CHAPTER 10

REFERENCES

REFERENCES

- [1] Liu, Jin, Min Li, Jianxin Wang, Fangxiang Wu, Tianming Liu, and Yi Pan. "A survey of MRI-based brain tumor segmentation methods." *Tsinghua science and technology* 19, no. 6 (2014): 578-595.
- [2] Toğaçar, Mesut, Burhan Ergen, and Zafer Cömert. "BrainMRNet: Brain tumor detection using magnetic resonance images with a novel convolutional neural network model." *Medical hypotheses* 134 (2020): 109531.
- [3] Amin, Javeria, Muhammad Sharif, Mussarat Yasmin, and Steven Lawrence Fernandes. "Big data analysis for brain tumor detection: Deep convolutional neural networks." *Future Generation Computer Systems* 87 (2018): 290-297.
- [4] Havaei, Mohammad, Axel Davy, David Warde-Farley, Antoine Biard, Aaron Courville, Yoshua Bengio, Chris Pal, Pierre-Marc Jodoin, and Hugo Larochelle. "Brain tumor segmentation with deep neural networks." *Medical image analysis* 35 (2017): 18-31.
- [5] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." In *Medical image computing and computer-assisted intervention—MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III* 18, pp. 234-241. Springer International Publishing, 2015.

- [6] Allah, Ahmed M. Gab, Amany M. Sarhan, and Nada M. Elshennawy. "Edge U-Net: Brain tumor segmentation using MRI based on deep U-Net model with boundary information." *Expert Systems with Applications* 213 (2023): 118833.
- [7] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III* 18, pp. 234-241. Springer International Publishing, 2015.
- [8] Chang, Yankang, Zhouzhou Zheng, Yingwei Sun, Mengmeng Zhao, Yao Lu, and Yan Zhang. "DPAFnet: A residual dual-path attention-fusion convolutional neural network for multimodal brain tumor segmentation." *Biomedical Signal Processing and Control* 79 (2023): 104037.
- [9] Tie, Juhong, Hui Peng, and Jiliu Zhou. "MRI brain tumor segmentation using 3D U-Net with dense encoder blocks and residual decoder blocks." *Computer Modeling in Engineering & Sciences* 128, no. 2 (2021): 427-445.
- [10] Miller, Kimberly D., Quinn T. Ostrom, Carol Kruchko, Nirav Patil, Tarik Tihan, Gino Cioffi, Hannah E. Fuchs et al. "Brain and other central nervous system tumor statistics, 2021." *CA: a cancer journal for clinicians* 71, no. 5 (2021): 381-406.