BRAIN TUMOR DETECTION USING UNET FOR IMAGE SEGMENTATION ON MRI IMAGES

A PROJECT REPORT

Submitted by
HARINARAYANAN R [RA2111026010424]
MANIKANTA SAI PATEL [RA2111026010433]

Under the Guidance of
DR. OM PRAKASH P.G
(Assistant Professor, Department of Computational Intelligence)

in partial fulfillment of the requirementsfor the degree of
BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING
with specialization in ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

DEPARTMENT OF COMPUTATIONAL INTELLIGENCE COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR- 603 203

NOVEMBER 2024

Department of Computational Technologies
SRM Institute of Science & Technology
Own Work Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/ Course	**:**	**B. TECH Computer Science Engineering (AIML)**
Student Name	**:**	**Harinarayanan R, Manikanta Sai Patel**
Registration Number	**:**	**RA2111026010424, RA2111026010433**
Title of Work	:	**Brain Tumor Detection using UNET for Image Segmentation on MRI Images**

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is my / our own except where indicated, and that we have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc.)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. Fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the university policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is our own work, except were indicated by referring, and that I have followed the good academic practices noted above.
If you are working in a group, please write your registration numbers and sign the date for every student in your group.

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that 18CSP107L project report titled "**BRAIN TUMOR DETECTION USING UNET FOR IMAGE SEGMENTATION ON MRI IMAGES"** is the bonafide work of "**Harinarayanan R [RA2111026010424], Manikanta Sai Patel [RA2111026010433]**" who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

Dr. OM PRAKASH PG
Assistant Professor
Department of Computational Intelligence

Dr. ANNIE UTHRA
Professor and Head
Department of Computational Intelligence

Examiner I **Examiner II**

ACKNOWLEDGEMENT

We express our humble gratitude to **Dr. C. Muthamizhchelvan,** Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr.T.V. Gopal,** for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman, Professor & Chairperson**, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We are incredibly grateful to our Head of the Department, **Dr. Annie Uthra**, Professor and Head, Department of Computational Intelligence, School of Computing, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our Project Coordinator, **Dr. M.S Abhirami**, **Associate Professor**, Panel Head, **Dr. S Sadagopan**, **Associate Professor**, Professor and members, **Dr. Om Prakash PG, Assistant Professor**, **Dr. R Siva, Associate Professor,** Department of Computational Intelligence, School of Computing, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor, **Dr. Om Prakash PG,** Assistant Professor, Department of Computational Intelligence, School of Computing, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide**, Dr. Om Prakash PG,** Assistant Professor, Department of Computational Intelligence, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under his mentorship. He provided us with the freedom and support to explore the research topics of our interest. His passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank the Computational Intelligence department staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

Harinarayanan R [RA2111026010424]

Manikanta Sai Patel [RA2111026010433]

ABSTRACT

This study presents a model for automated segmentation of Brain Tumor from MRI Images using deep learning framework called the U-NET architecture, which is primarily used for analysis of scans in the medical field. And the main objective is to demarcate the tumor regions from the MRI scans for an accurate diagnosis and planning of the treatment. The task of manually segmenting brain tumors is a labour intensive, complex process and is hindered by MRI data variability between institutions and by the intricacies of the tumor boundaries. Therefore, leveraging U-Net offers an effective solution due to its unique encoder-decoder architecture with skip connections, enabling it to capture fine details and spatial information critical to medical imaging.

The BraTS 2020 dataset, containing diverse, preoperative MRI images with expert-annotated ground truth labels, serves as the foundation for this model's training and validation [1]. Preprocessing steps included co-registering scans, interpolating resolutions, and standardizing image dimensions to ensure compatibility with the U-Net model. Throughout training, key metrics for evaluation like accuracy, dice coefficient, precision were used, indicating the model's high performance and reliable segmentation accuracy. The U-Net model consistently outperformed traditional methods, achieving a final Dice coefficient near 0.99, demonstrating its capability to accurately delineate tumor regions.

Results highlight the U-Net model's potential for clinical applications, though future enhancements are suggested to improve robustness and scalability. Incorporating attention mechanisms, transformer architectures, and data augmentation could refine the model's precision and adaptability across more diverse clinical contexts. Additionally, extending the U-Net to 3D processing could enable more precise volume estimations, providing an invaluable tool for personalized patient care. This study's findings contribute to advancing automated brain tumor segmentation, offering a promising step toward more efficient, accurate diagnosis and treatment planning in neuro-oncology

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

Medical and Imaging Terms

1. **MRI** - Magnetic Resonance Imaging
2. **GBM** - Glioblastoma
3. **HGG** - High-Grade Glioma
4. **LGG** - Lower-Grade Glioma
5. **NIfTI** - Neuroimaging Informatics Technology Initiative
6. **T1** - Native T1-Weighted MRI Sequence
7. **T1Gd** - Post-contrast T1-Weighted MRI Sequence
8. **T2** - T2-Weighted MRI Sequence
9. **FLAIR**- T2 Fluid-Attenuated Inversion Recovery MRI Sequence
10. **ET** - Enhancing Tumor
11. **ED** - Peritumoral Edema
12. **NCR** - Necrotic/Non-Enhancing Tumor Core
13. **ROI** - Region of Interest
14. **TCGA**- The Cancer Genome Atlas

Technical and Model Architecture Terms

1. **CNN** - Convolutional Neural Network
2. **DC** - Dice Coefficient
3. **ADAM**- Adaptive Moment Estimation (Optimizer)
4. **GAN** - Generative Adversarial Network
5. **IOU** - Intersection over Union
6. **FN** - False Negative
7. **FP** - False Positive
8. **TP** - True Positive
9. **TN** - True Negative
10. **AI** - Artificial Intelligence
11. **ML** - Machine Learning

CHAPTER 1

INTRODUCTION

1.1 Introduction to Brain Tumor Detection:

Brain tumors are one of the most severe forms of cancer, often leading to fatal outcomes if not diagnosed and treated in time. Early detection is an important step in improving patient outcomes, as timely intervention can significantly increase the chances of survival. The detailed imaging capabilities of MRI make this widely used to diagnose tumors. While manual interpretation of MRI images for tumor detection is a laborious and error prone task, in practice on large data sets within clinical environments. The growing demand for rapid and accurate medical diagnoses has brought attention to the role of automation in healthcare, particularly the application of AI and DL for analysis in the biomedical field.

In this research, we focus on utilizing an advanced neural computing technique known as U-Net, which is specifically made for segmentation on Biomedical images. This is an encoder-decoder architecture makes it an ideal choice for segmenting complex structures, such as tumors, from medical images. Its success comes from the skip connections between the encoder and decode, which enable the model to learn both local and global features of an image at the same time [2]. With its ability to produce precise segmentation masks, U-Net showed great results in different biomedical imaging applications, including the delineation of brain tumor areas from MRI images [3].

The project explores the use of U-Net to create an automated brain tumor detection system that provides accurate segmentation of tumor regions in MRI images. This system aims to assist radiologists and healthcare professionals in making more informed decisions, improving diagnostic accuracy while reducing the time spent analyzing images manually. The growing interest in AI-based solutions for medical diagnostics has pushed the boundaries of conventional imaging techniques, enabling rapid and accurate results that can assist in better patient management. By applying the U-Net architecture, the project aims to contribute to this ongoing revolution in medical image processing.

The implications of this project extend beyond just detecting tumors. By creating an effective segmentation system, we can also lay the groundwork for future advancements in personalized medicine, where tumor characteristics could be analyzed in real time, enabling tailored treatment

plans for patients. Furthermore, the model can be adapted and extended to other forms of medical imaging, making it a versatile tool in the broader field of diagnostic radiology. The potential to streamline medical workflows, reduce human error, and ultimately improve patient outcomes forms the foundation of this research initiative.

In conclusion, the project's focus is on using advanced neural computing techniques to solve a critical healthcare challenge. The U-Net-based brain tumor segmentation system represents an important step toward achieving faster and more reliable diagnostic processes, ensuring that medical professionals can spend less time analyzing images and more time developing effective treatment strategies for their patients [4]. By automating tumor detection, the project aims to deliver a real-world solution that contributes to improving the efficiency of medical services globally.

1.2 Motivation

The motivation for this project arises from the increasing burden on healthcare systems to deliver timely and accurate diagnoses, particularly in the field of oncology, where early detection is critical for effective treatment. Brain tumors present unique challenges in terms of detection due to their variability in size, shape, and location. The manual analysis of MRI scans requires significant expertise and is subject to variability in interpretation among radiologists. Additionally, as medical imaging technology continues to advance, the volume of data generated by hospitals and clinics is rapidly increasing, making it difficult for human experts to keep pace. These factors emphasize the need for AI-driven solutions that can automate and accelerate the diagnostic process without compromising accuracy.

Another driving factor is the proven potential of computer science in transforming the field of medical scanning. We have shown that convolutional neural networks can be trained to detect patterns humans fail to identify. One of its most popular use cases is medical image segmentation and for this U-Net was proven a good choice since it can work with limited datasets and produce very precise results. This project aims to harness this capability of U-Net to address the challenges associated with brain tumor segmentation, thereby offering a tool that can support radiologists in their diagnostic tasks. The project's focus on accuracy, efficiency, and clinical applicability makes it a timely and relevant contribution to modern healthcare.

There is also a personal motivation driving this research, stemming from the desire to improve patient outcomes through innovation. The emotional toll of a brain tumor diagnosis on patients and their

families cannot be understated, and providing medical professionals with advanced tools that can deliver faster and more reliable diagnoses can alleviate some of that burden. By reducing the time it takes to diagnose brain tumors and enhancing the precision of tumor delineation, this project seeks to positively impact the lives of patients by enabling earlier and more targeted treatments. This human-centered motivation ensures that the project maintains a focus on real-world clinical applicability.

Moreover, the demand for healthcare services is rapidly increasing due to a growing global population and an aging demographic, especially in regions with limited access to specialized medical care. In such contexts, AI-based solutions can democratize access to advanced diagnostic tools, allowing hospitals and clinics with limited resources to improve the quality of care they offer. By developing a brain tumor segmentation system that can be deployed in various settings, the project addresses the broader goal of reducing healthcare disparities and ensuring that cutting-edge medical technology is accessible to all.

Finally, this project is also motivated by the potential for interdisciplinary collaboration. The integration of AI with medical diagnostics is a multidisciplinary endeavor that brings together expertise from computer science, radiology, and healthcare systems. By advancing the state of AI in healthcare, this project opens doors for further collaboration between these fields, encouraging innovation that can ultimately lead to more sophisticated, life-saving medical technologies.

1.3 Sustainable Development Goal of the Project

The sustainable goal of this project is to contribute to the United Nations Sustainable Development Goal 3 (Good Health and Well-Being) by improving access to quality healthcare services through the use of advanced technology. Brain tumor detection and treatment are critical aspects of healthcare that, when enhanced, can save lives and reduce the burden on healthcare systems. This project aims to create a sustainable solution that leverages AI to improve diagnostic accuracy and speed, thereby reducing the likelihood of misdiagnoses and ensuring that patients receive timely and appropriate medical attention.

One key aspect of the project's sustainability is its focus on efficiency. By automating the segmentation process for brain tumors, the U-Net model reduces the time and resources required for manual analysis, making it a more sustainable option for healthcare providers. Additionally, the model can be deployed in a variety of healthcare settings, from large urban hospitals to rural clinics,

ensuring that even regions with limited access to specialist care can benefit from advanced diagnostic tools. This scalability makes the project a sustainable solution for improving global health outcomes.

Furthermore, the project promotes sustainable healthcare by aiming to reduce the strain on medical professionals. As the demand for healthcare services continues to rise, the need for efficient, reliable diagnostic tools becomes more pressing. By providing radiologists with an AI-driven solution that can quickly and accurately segment tumors from MRI scans, the project helps ease the work of the medical professionals to work on to more critical works and it helps in general smooth healthcare delivery.

Another sustainable aspect of the project is its potential for future expansion. The U-Net model developed in this project is highly adaptable and can be extended to other forms of medical imaging beyond brain tumor segmentation. By creating a flexible and reusable AI architecture, this project lays the groundwork for future innovations in medical diagnostics. As more data becomes available, the model can be retrained and fine-tuned for new applications, ensuring that it remains relevant and effective in the long term.

Finally, the project's emphasis on sustainability is also reflected in its commitment to ethical AI development. The model is designed with patient privacy and data security in mind, ensuring that medical data is handled responsibly and in accordance with regulatory guidelines. This commitment to ethical AI not only ensures that the project aligns with sustainable healthcare goals but also fosters trust in AI-driven medical technologies, encouraging their broader adoption in clinical practice. In conclusion, this project's sustainable goal is to create a lasting positive impact on global health by leveraging AI to improve diagnostic processes while ensuring that the solution is scalable, adaptable, and ethically sound.

1.4 Research Goal

The Research goal is to model a systematic and adept automated to figure out the distribution of brain tumor using the U-Net framework. By leveraging deep learning techniques, this project seeks to create a robust model that can analyze MRI images and precisely segment tumor regions with high accuracy. The ultimate aim is to support healthcare professionals by providing them with a reliable diagnostic tool that can expedite the identification of tumors, thus facilitating early intervention and treatment. The research focuses on improving segmentation performance for both certainty and

efficiency, ensuring the model can be integrated into real-world clinical environments where timely diagnosis is critical.

Additionally, this project aims to commit to the broader field of AI in biomedical imaging by demonstrating the efficacy of U-Net in complex segmentation tasks. Another sub-goal of the research is to develop a scalable solution that can be adapted for other medical imaging applications, such as lung cancer or retinal disease detection. By doing so, the research extends beyond brain tumor segmentation, positioning itself as a stepping stone for further advancements in AI-driven medical diagnostics. The project aims to lay a foundation for future research into AI models that can improve healthcare outcomes while minimizing resource requirements, ensuring a wider impact across various healthcare domains.

CHAPTER 2

LITERATURE REVIEW

2.1 U-Net framework for Medical Image Segmentation

U-Net has become an indispensable tool in medical image segmentation, especially for tasks requiring precise delineation of complex structures such as brain tumors. Its encoder-decoder architecture, characterized by down-sampling the input image into lower-dimensional representations and then reconstructing the original image with pixel-wise accuracy, makes it particularly effective for pixel-level segmentation tasks. What sets U-Net apart is its use of skip connections, which carry spatial information from the encoder to the decoder, helping to retain crucial image details lost during down-sampling. This architecture allows U-Net to capture contextual information, which is critical for medical tasks like brain tumor segmentation, where small but significant features may define the tumor boundaries.

Additionally, U-Net has been extended through various modifications to improve performance. For example, Attention U-Net includes attention gates that screen the important elements from the image to ignore the bothersome components, containing attention accuracy increase. This attention mechanism has proven to be particularly useful in segmenting brain tumors, which are often irregularly shaped and located in complex brain regions. Another variation, Res-UNet, adds residual connections to the basic U-Net architecture, enabling the model to train faster and more accurately by preventing gradient vanishing issues. Both of these enhancements have shown great promise in improving the identification of tumors, especially in MRI images of gliomas and other brain abnormalities [3].

Given the nature of brain tumor segmentation tasks, where accurate identification of tumor boundaries is crucial for surgical planning and treatment, U-Net's ability to deliver high-resolution segmentation has made it a favored model in this field. The flexibility of the U-Net architecture also allows it to be adapted for different scanning techniques, such as X-Ray and MRI scans. The BraTS challenge datasets, which feature multimodal MRI data of brain tumors, have been particularly instrumental in the application and evaluation of U-Net models in this domain. Researchers have demonstrated that U-Net not only outperforms traditional segmentation techniques but also provides a robust framework for further innovations in medical image analysis.

U-Net's success in brain tumor segmentation can also be attributed to its generalizability across diverse medical imaging datasets. By training well-annotated datasets such as BraTS, we demonstrate that U-Net models can generalize well to unseen data, successfully identifying tumor regions in different subjects and MRI protocols at high accuracy [5]. This is especially important given the variability in MRI scan quality, patient anatomy, and tumor heterogeneity. The continuous improvements in U-Net and its variants signify the model's potential to become an integral part of clinical workflows, where automated segmentation can assist radiologists in making faster, more accurate diagnoses.

Ultimately, U-Net's effectiveness in image segmentation, especially in the setting of brain tumors, stems from its well-designed architecture and its adaptability to various image processing challenges. As deep learning continues to evolve, U-Net and its enhanced variants are expected to play an even greater role in improving speed, and reproducibility of brain tumor diagnoses and treatment planning. First, this has tremendous implications not only for brain tumor research but for the larger field of biomedical analysis more generally, by opening up the possibility of automated, precise and large-scale image segmentation.

2.2 Deep Learning in Brain Tumor Segmentation

Unlike traditional image segmentation methods, which rely heavily on manual intervention and expert knowledge, deep learning models automate the segmentation process, allowing for faster and more consistent results. The rise of CNNs, and more specifically architectures like U-Net, has addressed many of the challenges that come with segmenting medical images, particularly MRI scans, which often feature complex textures and irregular tumor shapes

These models can identify features, such as edges and textures, and other high-level features like the overall shape and structure of a tumor. This makes CNNs especially useful in differentiating between healthy brain tissue and tumor regions, which can often have subtle differences in appearance. Moreover, by training these models on large, annotated datasets such as the BraTS challenge datasets, researchers can fine-tune the model to achieve high accuracy in detecting gliomas, meningiomas, and other brain tumor types.

Another significant advancement has been the use of multi-modal MRI data. Since different MRI modalities highlight different aspects of brain tissue and tumor characteristics, combining data from multiple MRI scans (e.g., T1, T2, FLAIR) allows CNNs to learn more comprehensive features. By

employing this multi-modal approach, the model is better able to separate out different parts of the tumor, such as the tumor core, edema, and necrotic tissue. In particular, U-Net and its variants have been shown to excel at processing multi-modal data, effectively integrating information from different MRI sequences to enhance segmentation accuracy.

The principal problems in neural net-based tumor segmentation are brain MRI scan variability across patients and different institutions. [6]. Differences in scanner hardware, scanning protocols, and anatomy of the inmate can bring up noise and deviations in the data, making it difficult for models to generalize across diverse datasets. To overcome this, data augmentation and transfer learning are several other techniques that enable models to train on a larger variety of images, and more easily adapt to new data. These techniques, combined with the robust architecture of models like U-Net, have enabled deep learning models to generalize better across different clinical settings, improving their utility in real-world applications.

2.3 Challenges and Advances in MRI-Based Tumor Segmentation

MRI brain tumor segmentation poses unique challenges because of the complexity and heterogeneity of the scans. Tumors such as glioblastomas are often irregularly shaped, with fuzzy boundaries that make them difficult to differentiate from healthy brain tissue. Additionally, tumors may infiltrate surrounding areas, further complicating the segmentation process. These challenges have driven the need for automated segmentation solutions, which can provide more consistent and accurate results.

Properties of CNNs have shown to be particularly useful in terms of learning high level features from MRI images in order to capture the subtle nuances of brain tumors [7]. For example, U-Net and its variants have been used to segment not only the tumor core but also surrounding edema and necrotic regions, which are critical for treatment planning. These models can process MRI images from multiple modalities allowing them to identify different information about the tumor's structure and improve segmentation accuracy. The use of attention mechanisms has further improved the accuracy of segmentation, particularly in cases where tumors are located in regions of the brain with complex anatomy [8].

CHAPTER 3

DATASET

3.1 BraTS Dataset: Structure and Composition

BraTS is a comprehensive free resource developed for benchmarking and training of models for brain tumor segmentation [9]. This dataset has undergone several iterations since its inception, growing in both complexity and comprehensiveness. This study uses the BraTS 2020 version, that consists of pre-operative multi-institutional MRI scans with LGG and HGG diagnosis. This dataset is structured to address the critical challenges, providing an invaluable benchmark for developing and evaluating deep learning models.

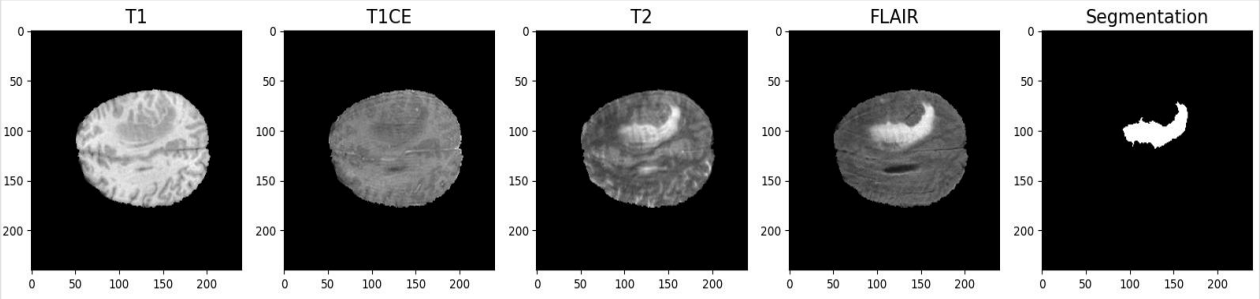


Figure 1: Visualization of MRI Modalities and Ground truth for Tumor Detection

Each scan, available in NIfTI format (.nii.gz), consists of four channels: T1 native, T1-weighted post-contrast (T1Gd), T2 Fluid Attenuated Inversion Recovery (T2-FLAIR) and T2-weighted (T2) [10]. This diversity enables models to learn multi-modal feature representations, which enhances their performance and generalizability across different imaging conditions.

BraTS dataset MRI scans come with meticulously annotated tumor sub-regions, labeled by expert neuro-radiologists. These annotations identify tumor sub-regions, such as the peritumoral edema, gadolinium-enhancing tumor and the non-enhancing tumor core along with necrotic tissue. The consistency and high quality of these labels make BraTS one of the most reliable datasets for brain tumor research, serving as a foundation for various automated segmentation techniques. The expert annotations improve the robustness of model training by providing accurate ground truth labels against which the predictions can be evaluated, ultimately making the BraTS dataset a critical resource for developing models capable of achieving high accuracy in brain tumor segmentation.

3.2 Preprocessing and Data Standardization

Data preprocessing is essential in brain tumor segmentation tasks, especially when dealing with complex, multimodal MRI data like the BraTS dataset. The preprocessing pipeline for BraTS data typically includes steps such as co-registering all scans to a common anatomical template, interpolating to a uniform resolution of $1mm^3$, and skull-stripping to remove non-brain tissues. These steps help standardize the images, ensuring that the model is not influenced by irrelevant variations due to differing patient anatomy, imaging protocols, or equipment. This preprocessing approach allows for consistency across MRI scans, improving model accuracy and robustness when trained on the dataset.

Label	Description	Class ID
4	GD-Enhancing Tumor (ET)	Tumor
2	Peritumoral Edema (ED)	Tumor
1	Necrotic/Non-Enhancing Core	Tumor
0	Background (Non-Tumor)	Non-Tumor

Table 1: Tumor Label Annotations

For further compatibility, the images are changed to a fixed resolution of 240x240 pixels. This consistency is needed considering the typical U-Net framework which requires input images of fixed dimensions so that features align within layers. Additionally, resizing reduces the computational load, allowing for more efficient training on high-performance GPU architectures without sacrificing segmentation detail.

Another vital aspect of preprocessing involves the normalization of pixel intensity values across MRI modalities. Given that MRI scans from different institutions or equipment may have variations in intensity, normalization ensures that the model focuses on the structural and anatomical details rather than variations in pixel intensity.

CHAPTER 4

SPRINT PLANNING AND EXECUTION

4.1 Sprint 1

4.1.1 Sprint Goal with User Stories of Sprint 1

Sprint Goal: Develop and validate a U-Net model for segmenting MRI images to detect brain tumors, establishing baseline accuracy and refining initial parameters.

User Stories:

- Story 1: As a medical researcher, I need a model that accurately segments tumor regions in MRI images to aid diagnosis.
- Story 2: As a data scientist, I need to preprocess MRI images to ensure consistency and improve model training.
- Story 3: As a software engineer, I need to implement the U-Net architecture for segmentation tasks.

4.1.2 Functional Document

In Sprint 1, our objective was to develop a foundational U-Net model for accurately detecting brain tumors in MRI images. The primary focus was on preprocessing MRI scans to ensure uniformity and consistency across all input data. This preprocessing phase included essential steps like scaling and normalizing images, which are critical to enhance model training and ensure accurate segmentation outcomes. Proper preprocessing reduces noise and optimizes input quality, allowing the U-Net model to better identify tumor regions in the early stages of development.

The sprint also focused on implementing the U-Net model architecture itself, starting with initial configurations and training parameters. With this architecture in place, we could generate segmentation outputs to test the model's initial performance. The output segmentation maps formed the baseline against which we'll measure all future model improvements. This sprint set the groundwork for refining the model and enhancing performance through further optimization and data augmentation in the following sprints.

4.1.3 Architecture Document

- Overview: The architecture was focused on establishing the foundational U-Net model, designed for semantic segmentation of brain tumors in MRI images.
- Components:
- Input Layer: Accepts pre-processed MRI images.
- U-Net Model: Consists of down sampling and up sampling layers, along with skip connections for retaining spatial information.
- Output Layer: Produces binary segmentation maps highlighting tumor regions.
- Dependencies: Required libraries included TensorFlow/Keras for model construction and MRI dataset for training and testing.

4.1.4 Functional Test Cases

- Test Case 1: Verify preprocessing correctness.
- Test Case 2: Confirm U-Net model construction.
- Test Case 3: Validate segmentation accuracy.

4.1.5 Daily Call Progress

- Days 1-2: Kicked off with a detailed review of the sprint objectives, breaking down tasks among team members. Initial focus was on image preprocessing, discussing techniques like scaling and normalization for consistent MRI input quality.
- Days 3-4: Worked on building the U-Net model architecture, configuring the model layers based on initial design goals. Verified dependencies (TensorFlow/Keras) and ensured compatibility across project requirements.
- Days 5-6: Conducted initial model training and examined early results, particularly accuracy and Dice coefficient improvements. Began refining model parameters to enhance performance based on observed metrics.
- Day 7: Held a sprint retrospective to evaluate preprocessing effectiveness and the initial U-Net model's setup. Concluded that foundational goals were met, setting the stage for the next sprint.

4.1.6 Committed vs. Completed User Stories

We committed to three key user stories: creating a U-Net model for brain tumor segmentation, preprocessing MRI images, and establishing the foundational architecture for segmentation tasks. These stories were essential to kickstarting our project, aiming to set a solid baseline model and refine initial training parameters. By the end of the sprint, we successfully completed the preprocessing pipeline and constructed the U-Net model, achieving an initial segmentation accuracy that aligned with our expectations. This solid start gave us a strong foundation to build on, allowing us to focus on more advanced features and improvements in the upcoming sprints.

4.1.7 Sprint Retrospective

We were able to establish a solid foundation by implementing the U-Net model architecture and completing the preprocessing pipeline. The sprint went smoothly, with both preprocessing and model setup meeting our initial expectations for accuracy. However, we did face a challenge with limited computational resources, which slowed down model training. Despite this, our team adapted quickly and efficiently, which helped us meet our sprint goals and set a strong groundwork for the next phase.

4.2 Sprint 2

4.2.1 Sprint Goal with User Stories of Sprint 2

Sprint Goal: Enhance the U-Net model's performance by implementing data augmentation techniques and optimizing hyperparameters.
User Stories:

- Story 1: I need to apply data augmentation to improve model robustness as an data scientist.
- Story 2: I want to experiment with different parameters to increase model accuracy as an ml engineer.
- Story 3: As a project manager, I need to document the effects of augmentation on model performance.

4.2.2 Functional Document

Sprint 2 aimed to strengthen the U-Net model's accuracy and generalizability by integrating data augmentation techniques and tuning hyperparameters. Data augmentation involved applying various

transformations to the MRI images, such as flipping, rotating, scaling, and adjusting brightness. This process was essential to make the model robust against variations and improve its ability to generalize to diverse tumor presentations. By adding synthetic variations to the training data, we expected to enhance the model's resilience and minimize overfitting, ensuring a stronger and more reliable segmentation performance.

In addition to data augmentation, we focused on hyperparameter optimization. We experimented with different learning rates, batch sizes, and dropout values to achieve an ideal balance between model accuracy and computational efficiency. These optimizations allowed us to iteratively fine-tune the model and assess its performance across various parameter settings. Re-evaluation of the model using these augmented data inputs helped us gauge the impact of each adjustment on segmentation accuracy, providing valuable insights for future enhancements.

4.2.3 Architecture Document

- Overview: This sprint expanded the data pipeline to include data augmentation techniques and refined model training with hyperparameter optimization.
- Components:
- Augmentation Module: Applied transformations, such as orientation adjustment, flipping, and scaling, to diversify the training set.
- U-Net Model with Enhanced Parameters: Adjustments in parameters like learning rate and dropout were made to improve model generalization.
- Dependencies: Utilized data augmentation libraries (such as Albumentations), with the U-Net model setup consistent with Sprint 1.

4.2.4 Functional Test Cases

- Test Case 1: Validate that augmentation correctly alters input images.
- Test Case 2: Ensure hyperparameter tuning affects model performance as expected.

4.2.5 Daily Call Progress

- Days 1-2: Outlined the sprint objectives, focusing on data augmentation strategies like orientation adjustment and resizing. Team members conducted tests to gauge the impact on model performance.

- Days 3-4: Continued applying data augmentation, introducing techniques like inversion and random cropping to diversify training data. Began hyperparameter tuning, experimenting with batch size and learning rate adjustments.
- Days 5-6: Refined the model based on tuning outcomes, monitoring accuracy, and preventing overfitting. Fine-tuned dropout rates and assessed their effect on model generalization.
- Day 7: Reviewed augmentation and hyperparameter tuning outcomes. Concluded with an analysis of model accuracy improvements, documenting changes to help guide future adjustments.

4.2.6 Committed vs. Completed User Stories

Our commitments centered on enhancing model performance through data augmentation and hyperparameter optimization. We aimed to make the U-Net model more robust by diversifying the training data with augmented variations and tuning key parameters to achieve greater segmentation accuracy. By the end of the sprint, we successfully implemented these enhancements, witnessing an improvement in model robustness and accuracy as a result. However, the time-intensive nature of hyperparameter tuning posed some challenges, leading us to prioritize specific combinations that delivered the best initial results within our timeline.

4.2.7 Sprint Retrospective

During Sprint 2, we saw significant improvements in model robustness by adding data augmentation and optimizing hyperparameters. These adjustments led to noticeable boosts in our accuracy and sensitivity metrics. One of the main challenges we encountered was managing time constraints while still experimenting effectively with different parameters. Although this required some balancing, we stayed focused and made process adjustments that allowed us to complete the sprint successfully, positioning our model well for further testing and refinement.

4.3 Sprint 3

4.3.1 Sprint Goal with User Stories of Sprint 3

Sprint Goal: Develop a user-friendly interface for real-time tumor detection and analysis of model performance metrics.

User Stories:

- Story 1: As a clinician, I need a simple interface to upload MRI images and receive segmentation results quickly.
- Story 2: As a project stakeholder, I want to visualize model performance metrics in an easily interpretable format.
- Story 3: As a software developer, I need to integrate the U-Net model with the front-end interface.

4.3.2 Functional Document

The primary objective of Sprint 3 was to develop a user interface (UI) that would enable seamless interaction with the trained U-Net model for brain tumor detection. This UI was designed to allow users, such as clinicians or researchers, to upload MRI images, view segmentation results, and access performance metrics easily. Core functionalities included real-time image upload and processing, ensuring that tumor detection results are presented quickly and accurately. This interface was also built to accommodate different MRI formats, making it versatile for a range of imaging needs.
Alongside the display of segmentation results, we implemented a system to show performance metrics—such as accuracy and Dice coefficient—in a visually interpretable format for end-users. These metrics help users assess the reliability of segmentation outputs, providing insights into model performance in a clinical context. The sprint emphasized a user-friendly design and intuitive layout, focusing on a seamless experience that connects technical outputs to practical diagnostic applications.

4.3.3 Architecture Document

- Overview: The architecture was extended to integrate a user interface that allows real-time tumor detection, displaying model performance metrics and segmentation results.
- Components:
- UI Layer: Interface for MRI image upload, segmentation result visualization, and user feedback.
- U-Net Model Integration: The model was linked to the front end for real-time segmentation processing.
- Performance Metrics Display: Displays metrics like accuracy, Dice coefficient, and MeanIoU for user analysis.

4.3.4 Functional Test Cases

- Test Case 1: Verify the upload functionality for different MRI formats.
- Test Case 2: Confirm segmentation results display correctly.
- Test Case 3: Validate performance metrics calculation.

4.3.5 Daily Call Progress

- Days 1-2: Launched the sprint with a focus on defining UI requirements for MRI image upload and real-time segmentation result display. Established initial layout and integration points between the model backend and frontend.
- Days 3-4: Developed the core UI structure, including upload functionality, and tested the initial connection to the model. Discussed additional features to enable user feedback for each segmentation result.
- Days 5-6: Implemented a display for model performance metrics, improving clarity for clinical interpretation. Addressed minor integration issues and refined the responsiveness of the UI.
- Day 7: Conducted final testing of UI components and backend integration, ensuring seamless functionality for real-time tumor detection. The sprint concluded with a comprehensive review and readiness for deployment.

4.3.6 Committed vs. Completed User Stories

We committed to three user stories focused on developing a user-friendly interface for real-time tumor detection, displaying performance metrics, and integrating the trained U-Net model into the interface. This sprint's goal was to make the model outputs accessible and interpretable for end-users, such as clinicians and researchers. By the end of the sprint, we had a functional UI that allowed users to upload MRI images, view segmentation outputs, and analyze performance metrics. Integration posed some initial challenges, but we managed to create a responsive interface that met all user story requirements, marking a successful sprint completion.

4.3.7 Sprint Retrospective

In Sprint 3, we focused on developing and integrating a user-friendly interface, which made tumor detection results more accessible and interpretable for end-users. Integrating the UI with the backend

was a success, creating a seamless real-time segmentation display. Initially, we faced some issues with UI responsiveness across different devices, but iterative testing and adjustments helped us resolve them. Overall, we ended the sprint with a responsive and functional UI, making the system more effective and practical for real-world use.

CHAPTER 5

METHODOLOGY

5.1 U-Net Framework

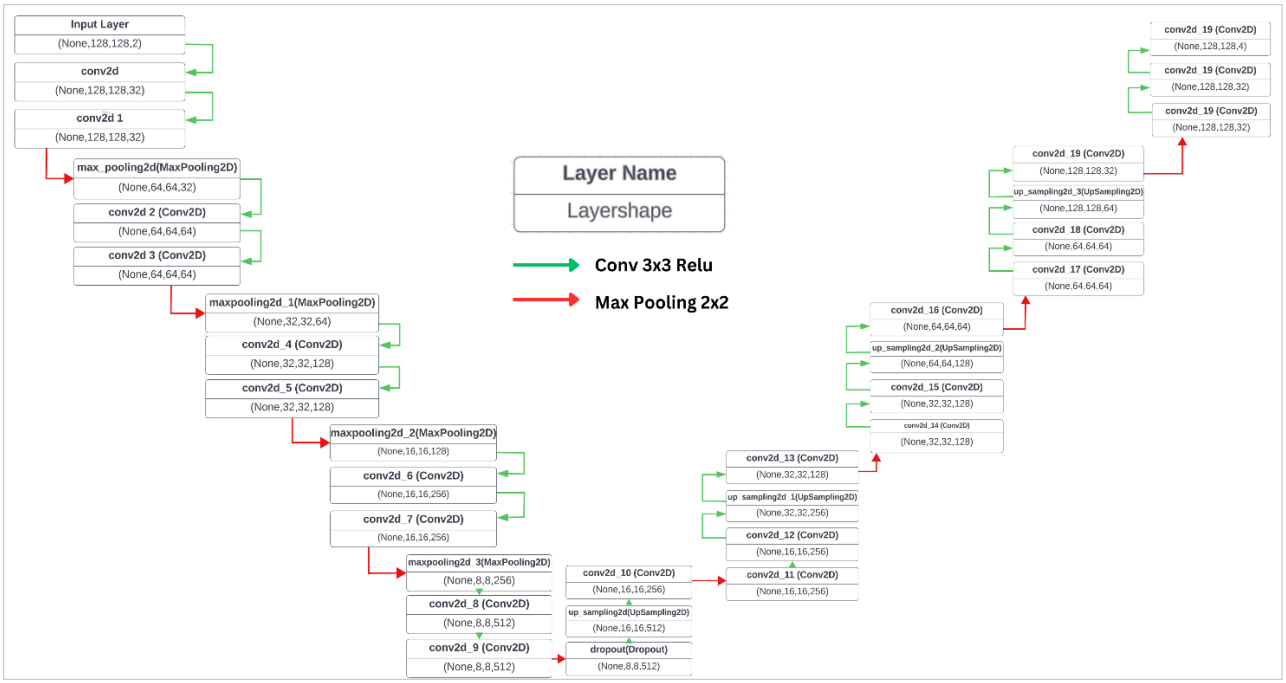


Figure 2: Layer-Wise Architecture of the U-Net Framework for Segmentation of Brain Tumors

5.1.1 Encoder Path (Contracting Path)

The Encoder Path, also known as the contracting path, is integral to feature extraction, which is vital for accurately segmenting complex structures like brain tumors in MRI images. Each block in the encoder helps the model retain spatial information and detect finer details like edges, textures, and shapes. These convolutional layers then have a 2x2 max pooling layer, which helps in decreasing dimensions of the image while retaining its core information, effectively down-sampling the data. As the model progresses through the encoder, the number of filters is gradually increased, starting from 32 and reaching up to 256.

The down-sampling process within the Encoder Path is crucial for compressing the input data and providing a high-level feature representation. By reducing the spatial resolution, the model avoids overfitting on small details and focuses on the most important patterns and textures relevant to tumor identification. This contraction is particularly useful in medical imaging, where segmentation needs

to be precise for reliable diagnosis. Additionally, this layered approach enables the U-Net model to handle variations in tumor shapes and sizes, improving its robustness in segmenting irregularly shaped tumors across diverse MRI scans.

Max-pooling layers in the Encoder Path play a role in creating scale-invariant features by retaining only the highest values from each pooling window. This not only reduces computational load but also contributes to capturing invariant features crucial for accurate segmentation. By the end of the encoder path, the model reaches a bottleneck where all essential data features are consolidated. This ability to condense information while preserving key features makes the Encoder Path foundational to the model's segmentation power, allowing it to capture both broad structures and subtle changes across different MRI slices.

5.1.2 Bottleneck layer

The bottleneck layer serves as the most compressed representation of the input image. Positioned at the deepest point in the U-Net, the bottleneck layer comprises convolutional layers with 512 filters, which help the model capture abstract, high-level features. The increased number of filters in this layer allows the model to consolidate detailed spatial relationships and semantic features, making the segmentation of complex brain tumor structures more accurate.

This abstraction within the bottleneck layer is essential for handling variations in MRI scans from different patients. By compressing the data into a feature-dense representation, the bottleneck enables the model to generalize better across varied tumor appearances and intensities. The model, therefore, becomes more robust and adaptable, able to recognize similar tumor features despite differences in shape, size, or location. This characteristic is especially beneficial in clinical settings, where consistent accuracy across diverse patient data is crucial for diagnostic reliability.

Additionally, the bottleneck layer's structure minimizes the risk of overfitting, by capturing only the essential features, the model avoids learning noise or irrelevant details. This is particularly important for clinical applications where the model will be used on new patient data, and ensuring consistency is key. The bottleneck layer, therefore, acts as the model's memory bank of critical tumor features, ensuring that only significant information is passed to the Decoder Path for precise reconstruction and segmentation.

5.1.3 Decoder Path (Expansive Path) with Skip Connections

The Decoder Path, or expansive path, reconstructs image and refines it to match the dimensions of the input image. This phase mirrors the Encoder Path in structure but operates in the opposite direction, progressively increasing spatial resolution through up-sampling layers. Up-sampling is achieved through transposed convolution layers, which increase the image size while refining the learned features. These layers are crucial for generating high-resolution segmentations necessary for medical applications where tumor boundaries need to be precisely delineated for treatment planning.

Skip connections are one of the most important features of the U-Net architecture. These connections allow the model to consolidate details from the contracting path, which would otherwise be lost during down-sampling. By concatenating feature maps from the contracting path to the respective layers in the expansive, the model preserves spatial context and finer details. This feature is vital for segmenting small structures, such as tumor boundaries.

The decoder path continues the process of reducing filter numbers in each layer, reversing the encoding path's pattern. Beginning with 512 filters at the condensed layer, every next layer reduces the number of filters (256, 128, 64, 32), gradually refining the spatial features and restoring the image to its original dimensions. The final layer applies a 1x1 convolution and a SoftMax activation function to classify each pixel into categories: edema, non-tumor areas, and both non-enhancing and enhancing tumor cores [5,9].

5.1.4 Reason for UNET over other Architectures

In brain MRI scans, tumors often have irregular and diffuse shapes, making accurate boundary delineation challenging. U-Net's design allows it to capture both high-level abstractions and low-level details, making it effective at isolating tumor regions from surrounding brain tissue. This ability is further enhanced by skip connections, which enable the model to retain spatial context and avoid losing important features during down-sampling, a common limitation in other CNN architectures.

Furthermore, U-Net's design addresses key challenges in medical image segmentation, such as variability in imaging modalities and differences across patient anatomy. Its encoder-decoder structure ensures that each level of detail, from broad shapes to intricate edges, is preserved and contributes to a comprehensive segmentation map. The model's encoder focuses on learning complex features, while the decoder reconstructs these features with accuracy, resulting in segmentation outputs that are both spatially accurate and semantically meaningful. This quality is essential in

clinical settings, where high accuracy is needed for diagnosis and treatment planning, particularly in cases where tumor boundaries are close to vital brain regions like the motor cortex or speech areas.

U-Net's applicability also extends to longitudinal studies, where tumor progression or treatment response needs to be monitored over time. By achieving consistent segmentation accuracy across multiple scans, U-Net facilitates the tracking of tumor changes, helping clinicians make informed decisions. The U-Net model's adaptability, accuracy, and spatial precision make it an ideal choice for brain tumor segmentation, improving diagnostic accuracy and enhancing patient care by allowing more precise treatment planning and monitoring of disease progression.

5.2 Data Augmentation and Generator Techniques

Data augmentation is essential for enhancing the generalizability of deep learning models, particularly in biomedical scanning. In this study, data amplification techniques are applied to prevent overfitting and enhance performance. The effectiveness of these augmentations is especially relevant in brain tumor segmentation, as tumors can vary significantly in size, shape, and orientation.

The Data Generator class is used to dynamically load and preprocess MRI images during training. This approach streamlines the process, enabling the model to train on batches of data without excessive memory usage. Additionally, it includes a shuffle function that randomizes the data order after each epoch, further reducing the risk of overfitting. The generator ensures that each batch of data is standardized to fit the input requirements of the U-Net model.

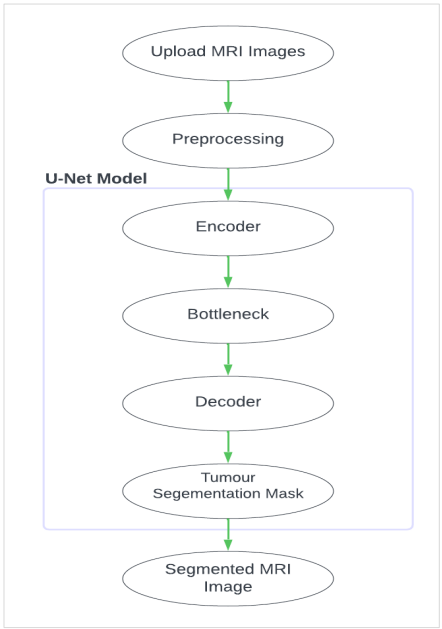


Figure 3: Workflow of the U-Net Framework for Segmenting Brain Tumor

To prepare the segmentation masks for multi-class classification, they are transformed into a one-hot encoded format. This conversion allows the model to treat each pixel in the mask as a distinct class, improving the accuracy of multi-class segmentation. The one-hot encoding process assigns specific pixel values to different tumor classes, allowing the model to make accurate predictions at the pixel level. By combining data augmentation and an efficient data generator, it assures that the data is robust and distinct, enhancing its segmentation performance on the BraTS dataset.

5.3 Model Training and Evaluation

Model training for brain tumor segmentation is conducted using the Adam optimizer, that is particularly well-suited for handling spare gradients and noisy data. The lr is set to 0.001, to balance convergence speed with model stability [8]. During training, cross-entropy loss function evaluates the performance, as it aligns well with the multi-class nature of the segmentation task. This loss function allows the U-Net model to learn from misclassifications, thereby increase its accuracy in delineating tumor boundaries across different sub-regions.

Throughout the training process, the model's performance is monitored on a validation set, enabling early stopping if there are signs of overfitting. This approach is critical for medical image segmentation tasks, where overfitting has a bad effect on real-world clinical data. By tuning the model parameters and employing early stopping, this study ensures that the U-Net model achieves high accuracy without sacrificing robustness.

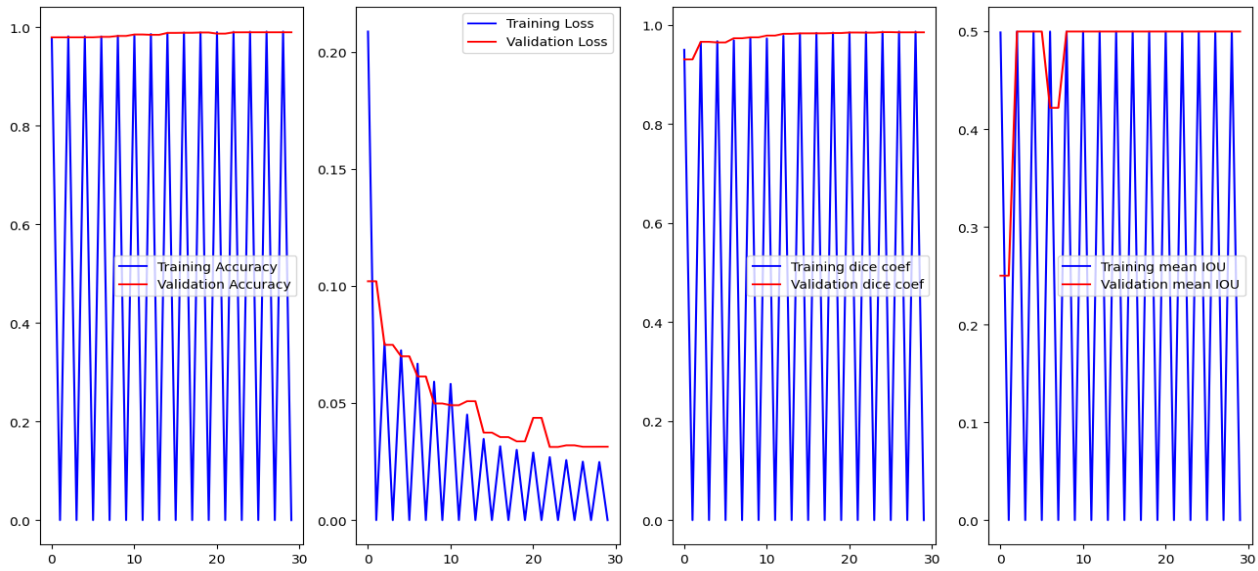


Figure 4:Training and Validation Metrics for U-Net Model on Brain Tumor Segmentation.

After training, the model's effectiveness is validated on a test dataset, comparing its performance to other segmentation methods like Fully Convolutional Networks (FCNs) and SegNet. Several Metrics are used to quantitatively assess segmentation accuracy, particularly in challenging regions with low contrast or complex tumor structures. The robust training and evaluation framework in this study highlights the model's potential for automated segmentation of tumors, offering a valuable tool for radiologists and enhancing diagnostic accuracy and treatment planning in clinical practice.

CHAPTER 6

CONCLUSION

6.1 Results

In this study, we developed and evaluated a U-Net-based model for brain tumor segmentation on MRI scans, specifically targeting the comprehensive BraTS 2020 dataset. The model effectively segmented tumor sub-regions, which is crucial for accurate diagnosis and treatment planning. The study measured performance through a variety of metrics, including accuracy, Dice coefficient, precision, sensitivity, specificity, and MeanIoU, yielding promising results across training and validation datasets. The results of the study reinforce the utility of the U-Net architecture for medical image segmentation, particularly due to its encoder-decoder structure with skip connections that preserve spatial information and fine detail.

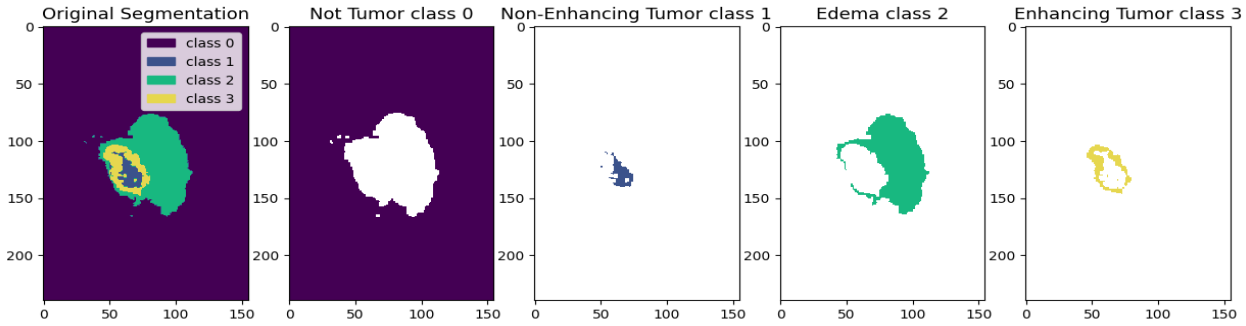


Figure 5:Multi-Class Brain Tumor Segmentation Results

6.1.1 Model Performance

The model evaluation focused on the segmentation performance using multiple metrics, which are essential for understanding the reliability of predictions in medical imaging. Over the course of 30 training epochs, the U-Net model demonstrated an impressive accuracy improvement, reaching a final accuracy of 99.14%, a high value that suggests a robust segmentation model. These results align with the trend of accuracy metrics in similar studies, where U-Net architectures have shown to handle fine-grained segmentation well by balancing both precision and recall. The accuracy results indicate that the model was able to correctly classify a high proportion of tumor and non-tumor pixels, which is especially valuable for sensitive clinical applications.

The Dice coefficient and MeanIoU further demonstrated the model's efficiency in segmentation. Starting with an initial Dice coefficient of 0.9074, the model reached 0.9898 by the end of training. MeanIoU, a more challenging metric representing intersection over union for all classes, showed a lower initial value (around 0.4) but gradually improved, reaching 0.51 in the test evaluation. The Dice coefficient's high final value reflects how well different tumor morphologies in the BraTS dataset could be understood, providing a reliable means of segmenting complex brain tumor shapes.

These findings underscore the significance of using U-Net for segmentation tasks involving irregular, complex shapes like brain tumors. By employing categorical cross-entropy loss, the model benefited from efficient training convergence, producing segmented outputs that are critical for identifying tumor boundaries. The model's strong performance across all metrics highlights its potential clinical utility in assisting neuro-radiologists with MRI-based tumor segmentation, which can facilitate improved diagnostics and treatment planning.

Metric	Initial Value (Epoch 1)	Final Value (Epoch 30)	Test Set Evaluation
Accuracy	96.86%	99.14%	98.94%
Precision	95.96%	99.11%	99.11%
Sensitivity	91.2%	98.69%	98.69%
Specificity	99.45%	99.7%	99.7%
Dice Coefficient	0.9074	0.9898	0.9856
MeanIoU	~0.40	0.51	0.5166
Loss	0.68	0.0243	0.0329

Table 2: Model Performance on Test Set and Training Overview

6.1.2 Training and Validation Results

The model's training results demonstrate a consistent improvement across multiple evaluation metrics, beginning with an accuracy of 96.86% in the first epoch and concluding with a peak of 99.14%. The validation accuracy remained stable, fluctuating within the range of 97.95% to 98.98% across epochs, indicating minimal overfitting and strong model generalization. The decreasing trend in loss function values supports this observation, with the training loss declining from 0.68 to 0.0243 and the validation loss reducing from 0.1021 to 0.0314 over the 30 epochs. This trend reflects the ability to optimize and converge with minimal error, further verified by the stable and reliable validation metrics.

The metrics for precision, sensitivity, and specificity provide further insight into the model's segmentation performance. Precision started at 95.96% and improved to 99.11% by epoch 25, demonstrating the model's increasing capacity to accurately identify tumor pixels without incorrectly classifying non-tumor pixels. Sensitivity, essential for detecting tumor regions accurately, reached 98.69%, and specificity reached a final value of 99.7%. These metrics emphasize the model's reliability in identifying both tumor and non-tumor areas, which is crucial for diagnostic accuracy in medical imaging.

The Dice coefficient and MeanIoU results further confirm the model's segmentation effectiveness. Starting at 0.9074, the Dice score steadily increased, reaching 0.9856 in the test evaluation. The MeanIoU, which began lower, still exhibited a notable improvement, ending at 0.51, a substantial accomplishment given the complexity of tumor segmentation. The consistent increase in Dice and MeanIoU values reflects the U-Net model's capability to capture tumor boundaries accurately and to generate meaningful predictions that match ground-truth annotations, an essential aspect for high-quality medical image segmentation.

6.2 Discussion and Future Research

The U-Net model's robust performance on the BraTS 2020 dataset demonstrates its efficacy in MRI-based brain tumor segmentation, offering high accuracy and sensitivity essential for clinical applications. The model's skip connections and encoder-decoder structure effectively preserved spatial information, allowing it to perform well in distinguishing complex tumor boundaries across diverse imaging protocols. However, challenges in segmenting irregular tumor shapes and handling the variability of MRI scans from different institutions suggest areas for improvement to make the model more universally applicable.

Future research could enhance model performance by integrating attention mechanisms and transformer-based architectures, which may help refine the U-Net's focus on relevant tumor features while capturing long-range dependencies. These additions could better equip the model to handle complex boundaries and variable imaging contexts, potentially making it more robust across different medical imaging environments.

Developing a 3D U-Net variant is another promising avenue, as it would enable the processing of volumetric MRI data, thus improving tumor structure analysis across multiple slices and providing more precise volume estimations. For clinical deployment, lightweight model versions could be

developed to meet the speed and computational demands of real-time applications, supporting efficient integration into clinical workflows. Collectively, these advancements could significantly improve the practical utility of U-Net in brain tumor segmentation, contributing to more accurate diagnosis and personalized treatment planning in neuro-oncology.

REFERENCES

[1] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J.S. Kirby, et al., "Advancing the Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features", Nature Scientific Data, 4:170117 (2017) DOI: 10.1038/sdata.2017.117

[2] Y. Ding, F. Chen, Y. Zhao, Z. Wu, C. Zhang and D. Wu, "A Stacked Multi-Connection Simple Reducing Net for Brain Tumor Segmentation," in IEEE Access, vol. 7, pp. 104011-104024, 2019, doi: 10.1109/ACCESS.2019.2926448.

[3] Crimi, Alessandro, and Spyridon Bakas, eds. Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: 6th International Workshop, BrainLes 2020, Held in Conjunction with MICCAI 2020, Lima, Peru, October 4, 2020, Revised Selected Papers, Part I. Vol. 12658. Springer Nature, 2021

[4] Ronneberger, O., Fischer, P., Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab, N., Hornegger, J., Wells, W., Frangi, A. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science(), vol 9351. Springer, Cham. https://doi.org/10.1007/978-3-319-24574-428.

[5] Bakas, Spyridon & Reyes, Mauricio & Jakab, András & Bauer, Stefan & Rempfler, et al. (2019). Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction in the BRATS Challenge, Research Gate

[6] M. Ghaffari, A. Sowmya and R. Oliver, "Automated Brain Tumor Segmentation Using Multimodal Brain Scans: A Survey Based on Models Submitted to the BraTS 2012–2018 Challenges," in IEEE Reviews in Biomedical Engineering, vol. 13, pp. 156-168, 2020, doi: 10.1109/RBME.2019.2946868.

[7] Ali Işın, Cem Direkoğlu, Melike Şah, Review of MRI-based Brain Tumor Image Segmentation Using Deep Learning Methods, Procedia Computer Science, Volume 102, 2016, Pages 317-324, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2016.09.407.

[8] Fidon, L., Ourselin, S., Vercauteren, T. (2021). Generalized Wasserstein Dice Score, Distributionally Robust Deep Learning, and Ranger for Brain Tumor Segmentation: BraTS 2020 Challenge. In: Crimi, A., Bakas, S. (eds) Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries. BrainLes 2020. Lecture Notes in Computer Science(), vol 12659. Springer, Cham. https://doi.org/10.1007/978-3-030-72087-2_18.

[9] S. Bakas, M. Reyes, A. Jakab, S. Bauer, M. Rempfler, A. Crimi, et al., "Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction in the BRATS Challenge", arXiv preprint arXiv:1811.02629 (2018)

[10] N. Haj-Hosseini, P. Milos, C. Hildesjö, M. Hallbeck, J. Richter, and K. Wårdell, 'Fluorescence spectroscopy and optical coherence tomography for brain tumor detection', presented at the SPIE Photonics Europe, Biophotonics: Photonic Solutions for Better Health Care, Brussels, Belgium, 3 - 7 April 2016, 2016, pp. 9887–96.

[11] V. Anitha and S. Murugavalli, "Brain tumour classification using two-tier classifier with adaptive segmentation technique", IET Computer Vision, vol. 10, no. 1, p. 9-17, 2016. https://doi.org/10.1049/iet-cvi.2014.0193

[12] Peter E. Ricci, David H. Dungan, Imaging of low- and intermediate-grade gliomas, Seminars in Radiation Oncology, Volume 11, Issue 2, 2001, Pages 103-112, ISSN 1053-4296, https://doi.org/10.1053/srao.2001.21420.

[13] A. Wulandari, R. Sigit and M. M. Bachtiar, "Brain Tumor Segmentation to Calculate Percentage Tumor Using MRI," 2018 International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC), Bali, Indonesia, 2018, pp. 292-296, doi: 10.1109/KCIC.2018.8628591.

APPENDIX

A. PATENT DISCLOSURE FORM / CONFERENCE PAPER

Outlook

Re: Submission of Conference Paper: "Brain Tumor Detection Using U-Net Architecture for Image Segmentation on MRI Images"

From icscn conf <icscnconf@gmail.com>
Date Fri 10/25/2024 11:32 AM
To Harinarayanan R <harinarayananr@outlook.com>

Dear Author
We received your paper
Thank you

On Fri, Oct 25, 2024 at 12:04 AM Harinarayanan R <harinarayananr@outlook.com> wrote:

> **Greetings,**
>
> I hope this email finds you well.
>
> I am writing to submit my conference paper titled "Brain Tumor Detection Using U-Net Architecture for Image Segmentation on MRI Images" for your review and consideration. The paper is attached to this email as a PDF file, as per the submission guidelines.
>
> Please let me know if you require any additional information or documents. I would be grateful for the opportunity to present my work at the upcoming conference, and I look forward to your feedback.
>
> Thank you for considering my submission.
>
> **Best regards,**
> Harinarayanan R,
> B.Tech in Computer Science w/s in AI & ML,
> SRM Institute of Science and Technology,
> Kattankulathur, India.

--

Regards,

Conference chair

ICSCN

Brain Tumor Detection Using UNet for Semantic Segmentation on MRI Images

Harinarayanan R,
dept. Computational Intelligence,
SRM Institute of Science and Technology,
Chennai, India,
harinarayananr@outlook,com.

Manikanta Sai Patel,
dept. Computational Intelligence,
SRM Institute of Science and Technology,
Chennai, India,
manikantasai26244@gmail.com

Dr. Om Prakash P.G
dept. Computational Intelligence,
SRM Institute of Science and Technology,
Chennai, India,
omp@srmist.edu.in

***Abstract*— The presence of brain tumors also often is accompanied by the formation of blood clots in the brain and early detection of these clots has been associated with significant mortality and morbidity linked with brain cancer. Precise segmentation of tumor tissue on magnetic resonance imaging (MRI) is necessary for effective diagnosis and treatment. Many of these deep learning paradigms for brain tumor segmentation were developed to outline tumor boundary and achieve high accuracy. We use in this paper the U-Net architecture, a convolutional neural network which is specifically designed for semantic segmentation of images, to process MRI scans of brain tumors. The aim is to delineate the critical tumor regions; in particular, the non-enhancing tumor core, the area of edema and the enhancing tumor. While the method performs well at accurately segmenting these regions, its application to assist clinicians in the diagnosis and treatment of brain tumors is demonstrated. As an assistive technology, the U-Net model offers substantial promise to assist radiologists in performing more accurate segmentation of brain tumor tissues.**

Keywords*— *Brain tumor detection, Deep learning, U-Net architecture, Convolutional neural networks (CNN)

I. Introduction

Brain tumors are when cells in the brain grow and multiply out of control leading to a mass. Since these tumors are confined to an area in the skull, pressure can be applied to the adjacent structures of the brain causing life threatening neurological complications [1]. Early detection and rapid diagnosis are critical to initiating treatment early, which decreases mortality and improves patient outcomes [2]. Gliomas, meningiomas, and pituitary tumors are among the most widespread of brain tumors [3].

Magnetic Resonance Imaging (MRI) is the most reliable modality for visualizing brain anatomy and tumor location and displaying detailed images of brain tissue [4]. Manual segmentation of MRI scans however is a time and error prone process, and especially difficult in regions such as complex tumor structures or at the low contrast interfaces with the tumor. The sheer volume of imaging data in modern health care further complicates the problem of manual interpretation and drives a need for automated solutions.

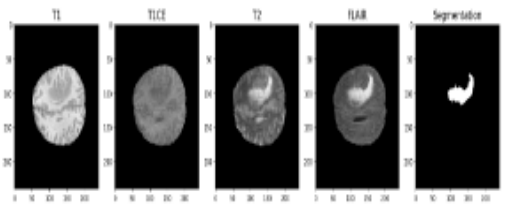


Fig. 1. Visualization of MRI Modalities and Ground truth for Tumor Detection

Drivers for automated image segmentation for the purpose of improving the accuracy of brain tumor classification and diagnosis are crucial. By segmenting MRI images, we can eliminate confounding structures in healthy brain tissues for further diagnostic clarity in delineating tumor subtypes to establish treatment choices. Because precise tumor boundaries must be identified during radiotherapy and surgical planning, accurate segmentation is of particular importance when the tumor is near areas crucial to speech, motor, and sensory functions. Further, longitudinal MRI scans can be segmented to the track developmental progression (growth, shrinkage, etc.) of tumors to assist in the evaluation of the success of treatment.

Current methods heavily reliant on manual delineation by radiologists is due to the importance of segmentation in clinical practice. This labor intensive, subjective, and operator experience dependent process involves slice-by-slice analysis of MRI scans. Finally, manual segmentation performed by the same operator can be reproducible, but reproducibility of manual segmentation results can vary. Due to the increasing complexity of multimodal, multi-institution, and longitudinal clinical data, a fully automated, objective and reproducible segmentation method is needed to facilitate large scale clinical trials and, subsequently, improve the accuracy of brain tumor diagnosis and treatment.

Improvements in the accuracy of medical image segmentation have greatly been improved by recent breakthroughs in machine learning and deep learning techniques. However, traditional methods of segmentation, K-means, and Fuzzy C-Means (FCM), fail to perform well for noisy images and have poor edge detection precision [6]. On the other hand, Convolutional Neural Networks (CNNs), in particular, are the prevailing technology for brain tumor segmentation tasks.

Especially for pixel-level segmentation tasks U-Net is a popular CNN architecture [7]. From its use of an encoder decoder architecture, with skip connections, the model is able to learn to capture spatial details and semantic information at both fine grain and high levels, making it a good fit for segmenting irregular tumor shapes in medical image. This is extended further with enhancements to U-Net variants, such as Attention Res-UNet, incorporating multi scale feature extraction and optimized loss function [8,9].

This study considers the use of U-Net architecture for brain tumor detection using MRI scans based on BraTS dataset [10] for comprehensive training and evaluation. The cells of glioblastoma (GBM/HGG) and lower grade glioma (LGG) from several institutions featured in the preoperative multimodal MRI scans of the BraTS 2020 dataset come together with expertly annotated ground truth labels. However, this dataset forms an ideal benchmark for training

the U-Net model such that it predicts the segmentation of key tumor components, i.e. non tumor regions, edema, and enhancing and non-enhancing tumor cores [11,12].

We validate the performance of the U-Net model by unseen data through a robust validation framework, showing that it performs much better in accuracy, robustness, and computational efficiency than other segmentation methods such as Fully Convolutional Networks (FCNs) and SegNet. Our results suggest the promise of deep learning-based automated segmentation to partially relieve radiologist burden, objectivity in treatment planning, and precision of tumor margin identification

II. Methodology

A. Dataset and Preprocessing

The work employs the BraTS (Brain tumor Segmentation) 2020 database, which includes multi-institutional pre-operative clinically acquired MRI scans of glioblastoma (GBM/HGG) and lower-grade gliomas (LGG). Each scan is provided in NIfTI format (.nii.gz) and includes four channels: native (T1), post-contrast T1-weighted (T1Gd), T2-weighted (T2), and T2 Fluid Attenuated Inversion Recovery (T2-FLAIR) [10]. This data was captured from different clinical protocols and scanners within 19 different institutions, making the dataset robust and diverse.

All imaging datasets were manually segmented by 1 to 4 expert raters, with their annotations approved by experienced neuro-radiologists. The annotations label tumor sub-regions, including the GD-enhancing tumor (ET — label 4), peritumoral edema (ED — label 2), and the necrotic and non-enhancing tumor core (NCR/NET — label 1) [11]. The data were preprocessed by co-registering all scans to a common anatomical template, interpolating to a uniform resolution of 1mm³, and skull-stripping to remove non-brain tissues. Each scan was subsequently resized to 240x240 pixels to ensure compatibility with the U-Net architecture, which was originally developed for higher-resolution microscopy images [13].

The BraTS dataset has undergone significant improvements over the years, with the data used from BraTS'12-'13 remaining a key part of the challenge, while pre-operative scans from TCGA-GBM and TCGA-LGG collections have been re-evaluated and included in the BraTS'17-'20 datasets. This dataset provides a robust foundation for evaluating the U-Net model, given its comprehensive annotations and pre-processing pipeline, which make it ideal for tumor segmentation tasks.

B. U-Net Architecture

This architecture is particularly suitable for the case of capturing finer details based on limited data, which are critical in the case of brain tumor segmentation. Below, it is described with an explanation both of its encoder-decoder structure and of special adaptations in order to apply it to the BraTS dataset.

In Encoder (Contracting Path); The U-Net mainly has used several end numbers of blocks of convolutional layers as well as max pooling layers only in the encoder part. Two consecutive convolutional layers with a filter size of 3x3, activation function of ReLU and padding 'same' are used within the block. Prominent features including the edges and textures of images are therefore accentuated while at the same time, maintaining the spatial relationships of the input data. There is only 32 filters, which increase progressively to higher values of 64, 128 and 256 as the network becomes deeper. In this manner the model builds up its model of the problem with an improved representation at each level of abstraction. For every set of layers, the execution follows an uncompressed ReLu layer and a 2x2 max pooling layer for the purpose of down sampling the feature map. It shrinks them but retains their core and all-important data. The encoder also stops short at a bottleneck layer composed of 512 filters hereby capturing all the abstract representations of the input data. It is only when the U-Net is moving from the contracting to the expansive phase that this is the chance for the model to learn the tumor features at a high level.

In Decoder (Expansive Path); It uses up-convolution layers that are also known as up sampling layers along with skip connections were implemented through concatenation with corresponding feature maps in the decoder. This is why skip connections are so important because they allow the network to retain spatial data and minor details for segmenting. Like the Encoder but in the reversed order with

Fig. 2. Layer-Wise Architecture of the U-Net Model for Brain Tumor Segmentation

B. SAMPLE CODING

1. Loading the Necessary Libraries

```
import nibabel as nib
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from skimage.transform import rotate
from skimage.util import montage
import os
from sklearn.model_selection import train_test_split
import keras
import cv2
import tensorflow
import random
from tensorflow.keras.layers import *
from tensorflow.keras.models import *
import numpy as np
from keras.callbacks import CSVLogger
import keras.backend as K
import zipfile
import pandas as pd
from keras import utils as np_utils
```

2. Downloading and Loading the Dataset

```
!kaggle datasets download -d awsaf49/brats20-dataset-training-validation
```

```
Dataset URL: https://www.kaggle.com/datasets/awsaf49/brats20-dataset-training-validation
License(s): CC0-1.0
Downloading brats20-dataset-training-validation.zip to /content
... resuming from 479199232 bytes (3989371709 bytes left) ...
100% 4.16G/4.16G [03:04<00:00, 17.0MB/s]
100% 4.16G/4.16G [03:04<00:00, 21.7MB/s]
```

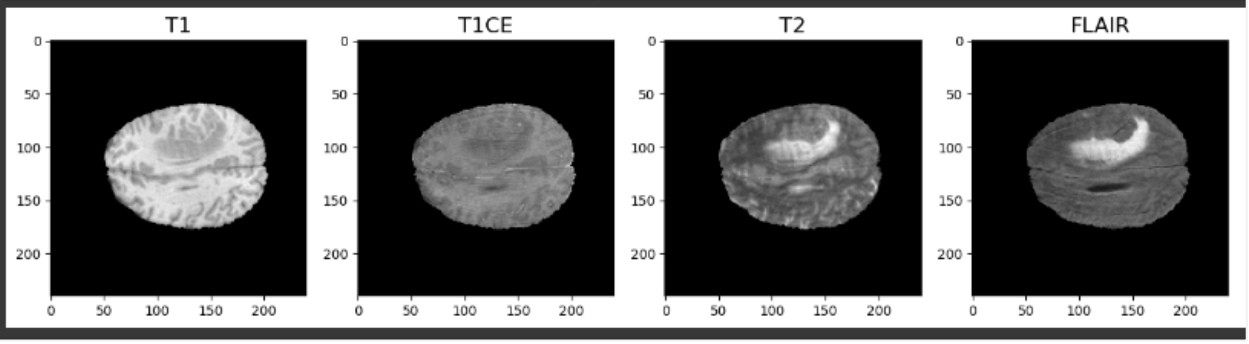
```
[ ] # Unzip dataset
    path_to_zip_file = "brats20-dataset-training-validation.zip"
    with zipfile.ZipFile(path_to_zip_file, 'r') as zip_ref:
        zip_ref.extractall("brats20-dataset-training-validation")
```

```
[ ] # Absolute path of the incorrectly named file
    old_name = "brats20-dataset-training-validation/BraTS2020_TrainingData/MICCAI_BraTS2020_TrainingData/BraT
    new_name = "brats20-dataset-training-validation/BraTS2020_TrainingData/MICCAI_BraTS2020_TrainingData/BraT

    try:
        os.rename(old_name, new_name)
        print("The file has been successfully renamed")
    except FileNotFoundError:
        print("Maybe you have already renamed the file or the file is not misspelled on the dataset anymore")
```

```
The file has been successfully renamed
```

3. Data Visualisation

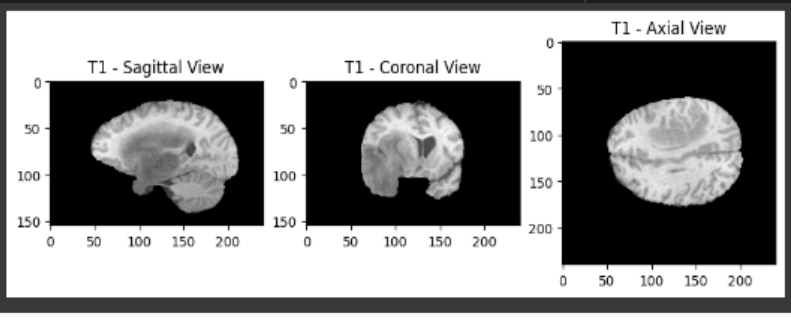
```
# Specify a sample path (here we will take the first patient of the Training dataset)
sample_path = 'brats20-dataset-training-validation/BraTS2020_TrainingData/MICCAI_BraTS2020_TrainingData/BraTS20_Training_001/BraTS20_Training_001_'

# Load the 4 MRI modalities and the segmentation located in the patient's path using the nibabel library
t1_img=nib.load(sample_path + 't1.nii')
t1ce_img=nib.load(sample_path + 't1ce.nii')
t2_img=nib.load(sample_path + 't2.nii')
flair_img=nib.load(sample_path + 'flair.nii')
seg_img=nib.load(sample_path + 'seg.nii')

# Get the image data
t1_data = t1_img.get_fdata()
t1ce_data = t1ce_img.get_fdata()
t2_data = t2_img.get_fdata()
flair_data = flair_img.get_fdata()
seg_data = seg_img.get_fdata()

# Plot the 100th slice of the 4 MRI modalities and the segmentation
slice_nb = 100

fig, axs = plt.subplots(1, 5, figsize=(20,20))
axs[0].imshow(t1_data[:,:,slice_nb], cmap="gray")
axs[0].set_title('T1', fontsize=16)
axs[1].imshow(t1ce_data[:,:,slice_nb], cmap="gray")
axs[1].set_title('T1CE',fontsize=16)
axs[2].imshow(t2_data[:,:,slice_nb], cmap="gray")
axs[2].set_title('T2',fontsize=16)
axs[3].imshow(flair_data[:,:,slice_nb], cmap="gray")
axs[3].set_title('FLAIR',fontsize=16)
axs[4].imshow(seg_data[:,:,slice_nb], cmap="gray")
axs[4].set_title('Segmentation',fontsize=16)
plt.show()
```




```
# Modality shape
print(t1_data.shape)

# Segmentation shape
print(seg_data.shape)
```

```
(240, 240, 155)
(240, 240, 155)
```

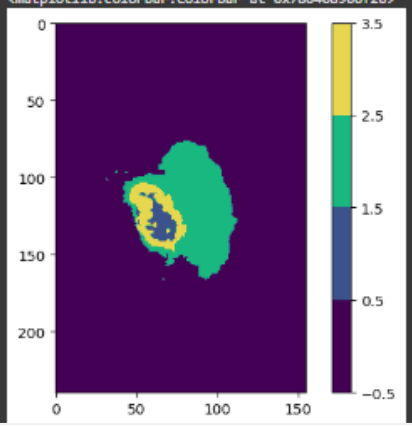
```
# Plot a MRI modality through all planes
slice_nb = 100

fig, axs2 = plt.subplots(1, 3, figsize=(10,10))

# Apply a 90° rotation with an automatic resizing, otherwise the display is less obvious to analyze
axs2[0].imshow(rotate(t1_data[slice_nb,:,:], 90, resize=True), cmap="gray")
axs2[0].set_title('T1 - Sagittal View')

axs2[1].imshow(rotate(t1_data[:,slice_nb,:], 90, resize=True), cmap="gray")
axs2[1].set_title('T1 - Coronal View')

axs2[2].imshow(t1_data[:,:,slice_nb], cmap="gray")
axs2[2].set_title('T1 - Axial View')

plt.show()
```



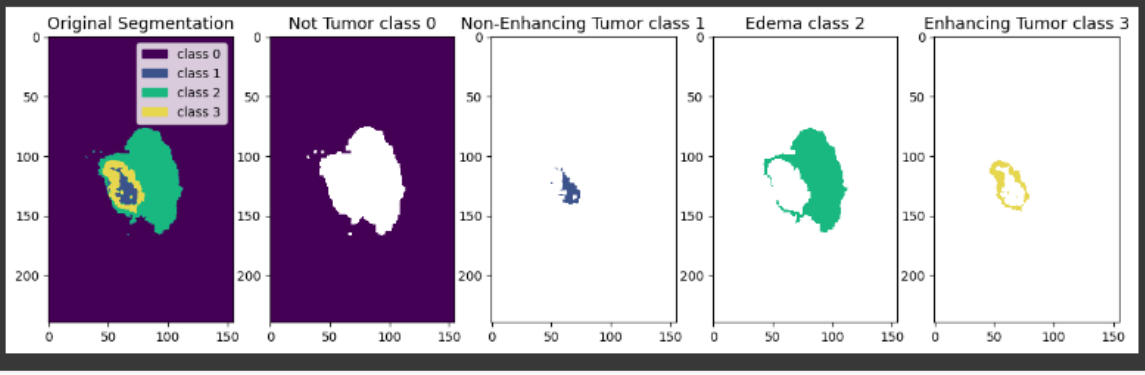
4. Sample Segmentation

```
some_seg_img = nib.load("brats20-dataset-training-validation/BraTS2020_TrainingData/MICCAI_BraTS2020_TrainingData/BraTS20_Training_001/BraTS20_Training_

cmap = mpl.colors.ListedColormap(['#440054', '#3b528b', '#18b880', '#e6d74f'])
norm = mpl.colors.BoundaryNorm([-0.5, 0.5, 1.5, 2.5, 3.5], cmap.N)

plt.imshow(some_seg_img[100,:,:], cmap=cmap, norm=norm)
plt.colorbar()
```

```
<matplotlib.colorbar.Colorbar at 0x786408906f20>
```




```
# Deletion of class 0
seg_0 = some_seg_img.copy()
seg_0[seg_0 != 0] = np.nan

# Isolation of class 1
seg_1 = some_seg_img.copy()
seg_1[seg_1 != 1] = np.nan

# Isolation of class 2
seg_2 = some_seg_img.copy()
seg_2[seg_2 != 2] = np.nan

# Isolation of class 4
seg_3 = some_seg_img.copy()
seg_3[seg_3 != 4] = np.nan

# Define legend
class_names = ['class 0', 'class 1', 'class 2', 'class 3']
legend = [plt.Rectangle((0, 0), 1, 1, color=cmap(i), label=class_names[i]) for i in range(len(class_names))]

fig, axs3 = plt.subplots(1, 5, figsize=(15, 15))

axs3[0].imshow(some_seg_img[100,:,:], cmap=cmap, norm=norm)
axs3[0].set_title('Original Segmentation',fontsize=13)
axs3[0].legend(handles=legend, loc='upper right')

axs3[1].imshow(seg_0[100,:,:], cmap=cmap, norm=norm)
axs3[1].set_title('Not Tumor class 0',fontsize=13)

axs3[2].imshow(seg_1[100,:,:], cmap=cmap, norm=norm)
axs3[2].set_title('Non-Enhancing Tumor class 1',fontsize=13)

axs3[3].imshow(seg_2[100,:,:], cmap=cmap, norm=norm)
axs3[3].set_title('Edema class 2',fontsize=13)

axs3[4].imshow(seg_3[100,:,:], cmap=cmap, norm=norm)
axs3[4].set_title('Enhancing Tumor class 3',fontsize=13)

plt.show()
```



5. Data Generator

```
[ ] IMG_SIZE = 128
    class DataGenerator(np_utils.Sequence):
        'Generates data for Keras'
        def __init__(self, list_IDs, dim=(IMG_SIZE,IMG_SIZE), batch_size = 1, n_channels = 2, shuffle=True):
            'Initialization'
            self.dim = dim # Resized image dimensions (128 x 128)
            self.batch_size = batch_size #  Number of images to load each time
            self.list_IDs = list_IDs # Patients IDs
            self.n_channels = n_channels # Number of channels (T1CE + FLAIR)
            self.shuffle = shuffle # Indicates if data is shuffled for each epoch
            self.on_epoch_end() # Updates indexes after each epoch

        def __len__(self):
            'Denotes the number of batches per epoch'
            return int(np.floor(len(self.list_IDs) / self.batch_size))

        def __getitem__(self, index):
            'Generate one batch of data'
            # Generate indexes of the batch
            indexes = self.indexes[index*self.batch_size:(index+1)*self.batch_size]

            # Find list of IDs
            Batch_ids = [self.list_IDs[k] for k in indexes]

            # Load & Generate data
            X, y = self.__data_generation(Batch_ids)

            return X, y

        def on_epoch_end(self):
            'Updates indexes after each epoch'
            self.indexes = np.arange(len(self.list_IDs))
            if self.shuffle == True:
                np.random.shuffle(self.indexes)

        def __data_generation(self, Batch_ids):
            'Generates data containing batch_size samples'
            # Initialization
            X = np.zeros((self.batch_size*VOLUME_SLICES, *self.dim, self.n_channels))
            y = np.zeros((self.batch_size*VOLUME_SLICES, 240, 240))

            for c, i in enumerate(Batch_ids):

                sample_path = os.path.join(data_path, i, i)
                t1ce_path = sample_path + '_t1ce.nii'
                flair_path = sample_path + '_flair.nii'
                seg_path = sample_path + '_seg.nii'

                t1ce = nib.load(t1ce_path).get_fdata()
                flair = nib.load(flair_path).get_fdata()
                seg = nib.load(seg_path).get_fdata()

                for j in range(VOLUME_SLICES):
                    X[j +VOLUME_SLICES*c,:,:,0] = cv2.resize(flair[:,:,j+VOLUME_START_AT], (IMG_SIZE, IMG_SIZE))
                    X[j +VOLUME_SLICES*c,:,:,1] = cv2.resize(t1ce[:,:,j+VOLUME_START_AT], (IMG_SIZE, IMG_SIZE))
                    y[j +VOLUME_SLICES*c] = seg[:,:,j+VOLUME_START_AT]
            y[y==4] = 3
            mask = tensorflow.one_hot(y, 4)
            Y = tensorflow.image.resize(mask, (IMG_SIZE, IMG_SIZE))
            return X/np.max(X), Y

    training_generator = DataGenerator(samples_train)
    valid_generator = DataGenerator(samples_val)
    test_generator = DataGenerator(samples_test)
```

```
# Split the dataset into train and validation sets
samples_train, samples_val = train_test_split(samples, test_size=0.2, random_state=42)

# Split the train set into the real train set and in a test set
samples_train, samples_test = train_test_split(samples_train, test_size=0.15, random_state=42)

# Print data distribution (Train: 68%, Test: 12%, Val: 20%)
print(f"Train length: {len(samples_train)}")
print(f"Validation length: {len(samples_val)}")
print(f"Test length: {len(samples_test)}")
```

```
Train length: 250
Validation length: 74
Test length: 45
```

6. Model Architecture

```
def build_unet(inputs, ker_init, dropout):
    conv1 = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(inputs)
    conv1 = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv1)

    pool = MaxPooling2D(pool_size=(2, 2))(conv1)
    conv = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(pool)
    conv = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv)

    pool1 = MaxPooling2D(pool_size=(2, 2))(conv)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(pool1)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv2)

    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(pool2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv3)

    pool4 = MaxPooling2D(pool_size=(2, 2))(conv3)
    conv5 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(pool4)
    conv5 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv5)
    drop5 = Dropout(dropout)(conv5)

    up7 = Conv2D(256, 2, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(UpSampling2D(size = 2)(drop5))
    merge7 = concatenate([conv3,up7], axis = 3)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(merge7)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv7)

    up8 = Conv2D(128, 2, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(UpSampling2D(size = 2)(conv7))
    merge8 = concatenate([conv2,up8], axis = 3)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(merge8)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv8)

    up9 = Conv2D(64, 2, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(UpSampling2D(size = 2)(conv8))
    merge9 = concatenate([conv,up9], axis = 3)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(merge9)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv9)

    up = Conv2D(32, 2, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(UpSampling2D(size = 2)(conv9))
    merge = concatenate([conv1,up], axis = 3)
    conv = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(merge)
    conv = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv)

    conv10 = Conv2D(4, 1, activation = 'softmax')(conv)

    return Model(inputs = inputs, outputs = conv10)
```

```
[ ] input_layer = Input((IMG_SIZE, IMG_SIZE, 2)) #Define input data shape

    model = build_unet(input_layer, 'he_normal', 0.2)  #Build and compile the model

    model.compile(loss="categorical_crossentropy", optimizer=tensorflow.keras.optimizers.Adam(learning_rate=0.001), metrics = ['accuracy',tensorflow.keras.metrics.MeanIoU(num_classes=4), dice_coef, prec
```

```
[ ] callbacks = [
        keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.2,
                                patience=2, min_lr=0.000001, verbose=1),

        keras.callbacks.ModelCheckpoint(filepath = 'model_.{epoch:02d}-{val_loss:.6f}.weights.h5',
                                verbose=1, save_best_only=True, save_weights_only = True),

        CSVLogger('training.log', separator=',', append=False)
    ]
```

```
[ ] import tensorflow as tf

    def dice_coef(y_true, y_pred, smooth=1.0):
        y_true_f = tf.reshape(y_true, [-1]) # Flatten the tensors for all classes at once
        y_pred_f = tf.reshape(y_pred, [-1])
        intersection = tf.reduce_sum(y_true_f * y_pred_f)     # Compute intersection and union using TensorFlow functions
        dice = (2. * intersection + smooth) / (tf.reduce_sum(y_true_f) + tf.reduce_sum(y_pred_f) + smooth)
        return dice

    def precision(y_true, y_pred):
        true_positives = tf.reduce_sum(tf.round(tf.clip_by_value(y_true * y_pred, 0, 1)))
        predicted_positives = tf.reduce_sum(tf.round(tf.clip_by_value(y_pred, 0, 1)))
        precision = true_positives / (predicted_positives + tf.keras.backend.epsilon())
        return precision

    def sensitivity(y_true, y_pred):
        true_positives = tf.reduce_sum(tf.round(tf.clip_by_value(y_true * y_pred, 0, 1)))
        possible_positives = tf.reduce_sum(tf.round(tf.clip_by_value(y_true, 0, 1)))
        return true_positives / (possible_positives + tf.keras.backend.epsilon())

    def specificity(y_true, y_pred):
        true_negatives = tf.reduce_sum(tf.round(tf.clip_by_value((1 - y_true) * (1 - y_pred), 0, 1)))
        possible_negatives = tf.reduce_sum(tf.round(tf.clip_by_value(1 - y_true, 0, 1)))
        return true_negatives / (possible_negatives + tf.keras.backend.epsilon())
```

7. Model Training

```
[ ] model.fit(training_generator,
              epochs=30,
              steps_per_epoch=len(samples_train),
              callbacks=callbacks,
              validation_data=valid_generator)
```

```
250/250 ━━━━━━━━━━━━━━━━━━━━ 0s 492ms/step - accuracy: 0.9895 - dice_coef: 0.9846 - loss: 0.0301 - mean_io_u: 0.5279 - precision: 0.9920 - sensitivity: 0.9865 - specificity: 0.9973
Epoch 19: val_loss improved from 0.03551 to 0.03368, saving model to model_.19-0.033680.weights.h5
250/250 ━━━━━━━━━━━━━━━━━━━━ 158s 620ms/step - accuracy: 0.9895 - dice_coef: 0.9846 - loss: 0.0301 - mean_io_u: 0.5278 - precision: 0.9920 - sensitivity: 0.9865 - specificity: 0.9973 - val_accuracy:
Epoch 20/30

Epoch 20: val_loss improved from 0.03368 to 0.03368, saving model to model_.20-0.033680.weights.h5
250/250 ━━━━━━━━━━━━━━━━━━━━ 41s 166ms/step - accuracy: 0.0000e+00 - dice_coef: 0.0000e+00 - loss: 0.0000e+00 - mean_io_u: 0.0000e+00 - precision: 0.0000e+00 - sensitivity: 0.0000e+00 - specificity:
Epoch 21/30
250/250 ━━━━━━━━━━━━━━━━━━━━ 0s 497ms/step - accuracy: 0.9896 - dice_coef: 0.9844 - loss: 0.0303 - mean_io_u: 0.5313 - precision: 0.9920 - sensitivity: 0.9863 - specificity: 0.9973
Epoch 21: ReduceLROnPlateau reducing learning rate to 4.0000001899898055e-05.

Epoch 21: val_loss did not improve from 0.03368
250/250 ━━━━━━━━━━━━━━━━━━━━ 159s 617ms/step - accuracy: 0.9896 - dice_coef: 0.9844 - loss: 0.0303 - mean_io_u: 0.5312 - precision: 0.9920 - sensitivity: 0.9863 - specificity: 0.9973 - val_accuracy:
Epoch 22/30

Epoch 22: val_loss did not improve from 0.03368
250/250 ━━━━━━━━━━━━━━━━━━━━ 32s 127ms/step - accuracy: 0.0000e+00 - dice_coef: 0.0000e+00 - loss: 0.0000e+00 - mean_io_u: 0.0000e+00 - precision: 0.0000e+00 - sensitivity: 0.0000e+00 - specificity:
Epoch 23/30
250/250 ━━━━━━━━━━━━━━━━━━━━ 0s 499ms/step - accuracy: 0.9895 - dice_coef: 0.9850 - loss: 0.0312 - mean_io_u: 0.5314 - precision: 0.9914 - sensitivity: 0.9868 - specificity: 0.9971
Epoch 23: val_loss improved from 0.03368 to 0.03129, saving model to model_.23-0.031288.weights.h5
250/250 ━━━━━━━━━━━━━━━━━━━━ 159s 623ms/step - accuracy: 0.9895 - dice_coef: 0.9850 - loss: 0.0311 - mean_io_u: 0.5313 - precision: 0.9914 - sensitivity: 0.9868 - specificity: 0.9971 - val_accuracy:
Epoch 24/30

Epoch 24: val_loss did not improve from 0.03129
250/250 ━━━━━━━━━━━━━━━━━━━━ 31s 122ms/step - accuracy: 0.0000e+00 - dice_coef: 0.0000e+00 - loss: 0.0000e+00 - mean_io_u: 0.0000e+00 - precision: 0.0000e+00 - sensitivity: 0.0000e+00 - specificity:
Epoch 25/30
250/250 ━━━━━━━━━━━━━━━━━━━━ 0s 502ms/step - accuracy: 0.9914 - dice_coef: 0.9869 - loss: 0.0246 - mean_io_u: 0.5300 - precision: 0.9931 - sensitivity: 0.9888 - specificity: 0.9976
Epoch 25: ReduceLROnPlateau reducing learning rate to 8.000000525498762e-06.

Epoch 25: val_loss did not improve from 0.03129
250/250 ━━━━━━━━━━━━━━━━━━━━ 171s 667ms/step - accuracy: 0.9914 - dice_coef: 0.9869 - loss: 0.0246 - mean_io_u: 0.5299 - precision: 0.9931 - sensitivity: 0.9888 - specificity: 0.9976 - val_accuracy:
Epoch 26/30

Epoch 26: val_loss did not improve from 0.03129
250/250 ━━━━━━━━━━━━━━━━━━━━ 31s 125ms/step - accuracy: 0.0000e+00 - dice_coef: 0.0000e+00 - loss: 0.0000e+00 - mean_io_u: 0.0000e+00 - precision: 0.0000e+00 - sensitivity: 0.0000e+00 - specificity:
Epoch 27/30
250/250 ━━━━━━━━━━━━━━━━━━━━ 0s 497ms/step - accuracy: 0.9913 - dice_coef: 0.9869 - loss: 0.0251 - mean_io_u: 0.5303 - precision: 0.9929 - sensitivity: 0.9886 - specificity: 0.9976
Epoch 27: ReduceLROnPlateau reducing learning rate to 1.6000001778593287e-06.

Epoch 27: val_loss did not improve from 0.03129
250/250 ━━━━━━━━━━━━━━━━━━━━ 158s 620ms/step - accuracy: 0.9913 - dice_coef: 0.9869 - loss: 0.0251 - mean_io_u: 0.5302 - precision: 0.9929 - sensitivity: 0.9886 - specificity: 0.9976 - val_accuracy:
Epoch 28/30

Epoch 28: val_loss did not improve from 0.03129
250/250 ━━━━━━━━━━━━━━━━━━━━ 33s 132ms/step - accuracy: 0.0000e+00 - dice_coef: 0.0000e+00 - loss: 0.0000e+00 - mean_io_u: 0.0000e+00 - precision: 0.0000e+00 - sensitivity: 0.0000e+00 - specificity:
Epoch 29/30
250/250 ━━━━━━━━━━━━━━━━━━━━ 0s 502ms/step - accuracy: 0.9914 - dice_coef: 0.9870 - loss: 0.0243 - mean_io_u: 0.5314 - precision: 0.9931 - sensitivity: 0.9887 - specificity: 0.9976
Epoch 29: ReduceLROnPlateau reducing learning rate to 1e-06.

Epoch 29: val_loss did not improve from 0.03129
250/250 ━━━━━━━━━━━━━━━━━━━━ 161s 632ms/step - accuracy: 0.9914 - dice_coef: 0.9870 - loss: 0.0243 - mean_io_u: 0.5313 - precision: 0.9931 - sensitivity: 0.9887 - specificity: 0.9976 - val_accuracy:
Epoch 30/30

Epoch 30: val_loss did not improve from 0.03129
250/250 ━━━━━━━━━━━━━━━━━━━━ 30s 122ms/step - accuracy: 0.0000e+00 - dice_coef: 0.0000e+00 - loss: 0.0000e+00 - mean_io_u: 0.0000e+00 - precision: 0.0000e+00 - sensitivity: 0.0000e+00 - specificity:
<keras.src.callbacks.history.History at 0x786408708a30>
```

8. Results Evaluation

```
results = model.evaluate(test_generator, batch_size=100, callbacks= callbacks)

descriptions = ["Loss", "Accuracy", "MeanIOU", "Dice coefficient", "Precision", "Sensitivity", "Specificity"]

# Combine results list and descriptions list
results_list = zip(results, descriptions)

# Display each metric with its description
print("\nModel evaluation on the test set:")
print("==================================")
for i, (metric, description) in enumerate(results_list):
    print(f"{description} : {round(metric, 4)}")
```

```
45/45 ━━━━━━━━━━━━━━━━━━━━ 20s 440ms/step - accuracy: 0.9908 - dice_coef: 0.9867 - loss: 0.0264 - mean_io_u: 0.6720 - precision: 0.9926 - sensitivity: 0.9882 - specificity: 0.9975

Model evaluation on the test set:
==================================
Loss : 0.0329
Accuracy : 0.9894
MeanIOU : 0.5166
Dice coefficient : 0.9856
Precision : 0.9911
Sensitivity : 0.9869
Specificity : 0.997
```

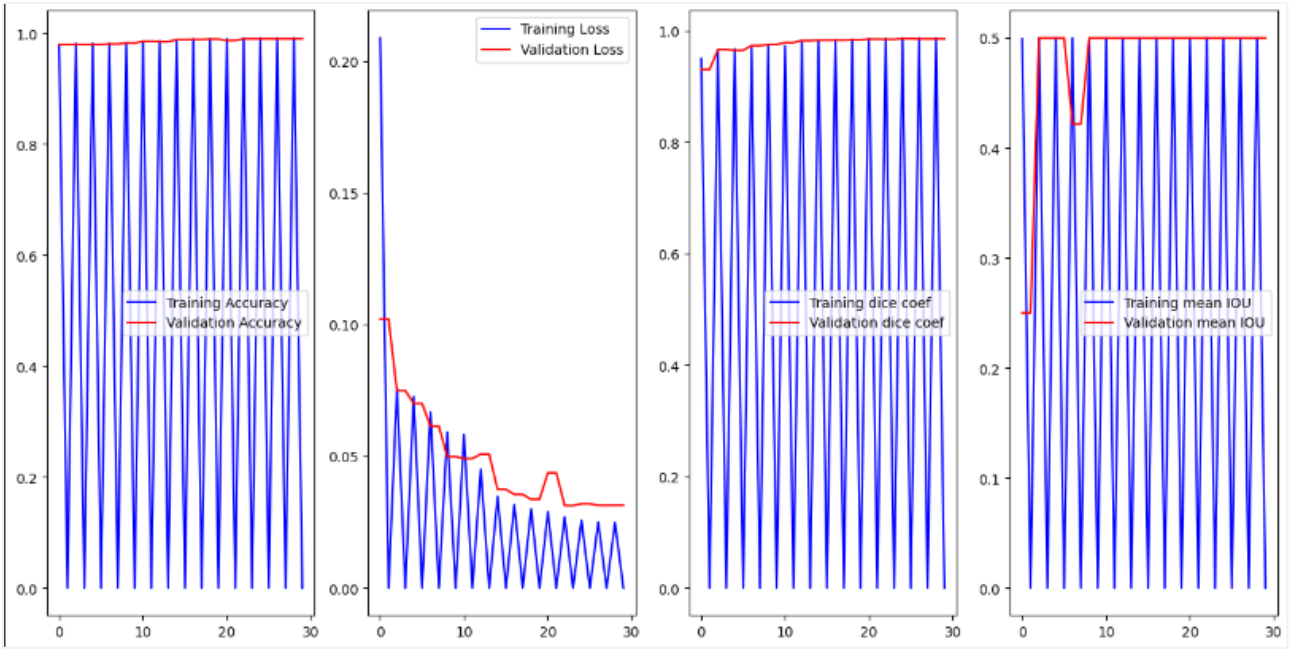
```
model.save("Brain_tumor_segmentation_model.keras")
```

```
[ ] model = keras.models.load_model('/content/Brain_tumor_segmentation_model.keras',
                                    custom_objects={"accuracy" : tf.keras.metrics.MeanIoU(num_classes=4),
                                                    "dice_coef" : dice_coef,
                                                    "precision" : precision,
                                                    "sensitivity" :sensitivity,
                                                    "specificity" :specificity,
                                                   }, compile=False)
```

9. Training and Validation metrics

```
[ ] history = pd.read_csv('/content/training.log', sep=',', engine='python')

    hist=history

    acc=hist['accuracy']
    val_acc=hist['val_accuracy']

    epoch=range(len(acc))

    loss=hist['loss']
    val_loss=hist['val_loss']

    train_dice=hist['dice_coef']
    val_dice=hist['val_dice_coef']

    f,ax=plt.subplots(1,4,figsize=(16,8))

    ax[0].plot(epoch,acc,'b',label='Training Accuracy')
    ax[0].plot(epoch,val_acc,'r',label='Validation Accuracy')
    ax[0].legend()

    ax[1].plot(epoch,loss,'b',label='Training Loss')
    ax[1].plot(epoch,val_loss,'r',label='Validation Loss')
    ax[1].legend()

    ax[2].plot(epoch,train_dice,'b',label='Training dice coef')
    ax[2].plot(epoch,val_dice,'r',label='Validation dice coef')
    ax[2].legend()

    ax[3].plot(epoch,hist['mean_io_u'],'b',label='Training mean IOU')
    ax[3].plot(epoch,hist['val_mean_io_u'],'r',label='Validation mean IOU')
    ax[3].legend()

    plt.show()
```

```
from tensorflow.keras.utils import plot_model

plot_model(model,
           show_shapes = True,
           show_dtype=False,
           show_layer_names = True,
           rankdir = 'TB',
           expand_nested = False,
           dpi = 70)
```



C.PLAGIARISM REPORT

424 433 V2

424433v2

Quick Submit

Quick Submit

SRM Institute of Science & Technology

Document Details

Submission ID
trn:oid:::1:3069012395

Submission Date
Nov 6, 2024, 1:15 PM GMT+5:30

Download Date
Nov 6, 2024, 1:18 PM GMT+5:30

File Name
Report_424_433_v2.docx

File Size
1.8 MB

35 Pages

6,781 Words

41,529 Characters

10% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

Small Matches (less than 8 words)

Match Groups

46 Not Cited or Quoted 10%
Matches with neither in-text citation nor quotation marks

2 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

8% Internet sources

9% Publications

5% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.