## RESOURCE-EFFICIENT BRAIN TUMOR SEGMENTATION: ATTENTION U-NET APPROACH

**CS6611 – CREATIVE AND INNOVATIVE PROJECT**

***Submitted by***

**DHARSHINI M (2021103520)**

**CHOLAN M P (2021103739)**

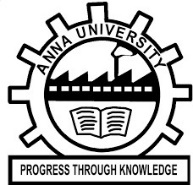
**PRABHAHARAN N M (2021103556)**

***in partial fulfilment of the requirements for the award of the degree of***

**BACHELOR OF ENGINEERING**

**in**

**COMPUTER SCIENCE AND ENGINEERING**



**College Of Engineering, Guindy**

**ANNA UNIVERSITY, CHENNAI - 600 025**

**MAY 2024**

**ANNA UNIVERSITY, CHENNAI - 600 025**

**BONAFIDE CERTIFICATE**

Certificate that this project request titled **Resource-efficient Brain Tumor Segmentation:** **ATTENTION U-net Approach** is the bonafide work of **Dharshini M (2021103520), Cholan M P (2021103739) and Prabhaharan N M (2021103556)** who carried out the project work under my supervision, for the fulfilment of the requirements as part of the CS6611 – Creative and Innovative Project.

|  |  |  |
| --- | --- | --- |
| **DR. S.**  **VALLI**  **PROFESSOR & HEAD**  **HEAD OF THE DEPARTMENT**  Department of  Computer Science  and Engineering,  Anna University,  Chennai – 600025. | **DR. P. RABHAVATHY**  **Assistant Professor**  **SUPERVISOR**  Department of  Computer Science  and Engineering,  Anna University,  Chennai – 600025. | **DR. S.**  **RENUGADEVI**  **Assistant Professor**  **(Selection Grade)**  **SUPERVISOR**  Department of  Computer Science  and Engineering,  Anna University,  Chennai – 600025. |

## ABSTRACT

Brain tumor segmentation from medical images is a critical task that aids in accurate diagnosis and treatment planning. In this thesis, we present a novel approach that leverages a U-Net architecture enhanced with attention mechanisms to achieve high accuracy in brain tumor segmentation while minimizing the required dataset size. Our method introduces a streamlined data processing pipeline that employs advanced attention blocks to focus on relevant features and improve segmentation performance. We conduct extensive experiments using standard brain tumor imaging datasets, comparing our approach against baseline U-Net models and other state-of-the-art techniques. Our results demonstrate that the attention-enhanced U-Net achieves superior accuracy with a significantly reduced dataset size, outperforming traditional models in terms of segmentation quality and computational efficiency. By optimizing the model's architecture and employing efficient training techniques, we successfully reduce the model's dependency on large datasets while preserving its performance. This work contributes to the advancement of medical image analysis by offering a data-efficient solution for brain tumor segmentation that can be adapted for practical clinical use. Our findings have potential implications for medical research and clinical practice, providing a foundation for future studies that seek to enhance medical imaging tasks using compact datasets and attention-driven neural network models.

## ACKNOWLEDGEMENT

Foremost, we would like to express our sincere gratitude to our project guide, **Dr. P. PRABHAVATHY**, Assistant Professor and **DR. S. RENUGADEVI,** Assistant Professor (Selection Grade), Department of Computer Science and Engineering, College of Engineering Guindy, Chennai for their constant source of inspiration. We thank them for the continuous support and guidance which was instrumental in taking the project to successful completion.

We are grateful to **Dr. S. Valli**, Professor and Head, Department of Computer Science and Engineering, College of Engineering Guindy, Chennai for her support and for providing necessary facilities to carry out for the project.

We would also like to thank our friends and family for their encouragement and continued support. We would also like to thank the Almighty for giving us the moral strength to accomplish our task.

**DHARSHINI M CHOLAN M P PRABHAHARAN N M**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **TITLE** | **PAGE NO** |
|  | **ABSTRACT** | **5** |
|  | **ACKNOWLEDGEMENT** | **6** |
|  | **LIST OF TABLES** | **9** |
|  | **LIST OF FIGURES** | **10** |
| **1.** | **INTRODUCTION** | **11** |
|  | 1.1. BACKGROUND | **11** |
|  | 1.2. PROBLEM STATEMENT | **11** |
|  | 1.3. RESEARCH OBJECTIVES | **12** |
|  | 1.4. SCOPE OF THE STUDY | **13** |
| **2.** | **LITERATURE SURVEY** | **14** |
|  | 2.1. LITERATURE REVIEW  2.2. SUMMARY OF LITERATURES | **14**  **16** |
| **3.** | **SYSTEM DESIGN**  3.1. SYSTEM ARCHITECHTURE  3.1.1. U-NET NETWORK STRUCTURE  3.1.2. ATTENTION MECHANISM  INTEGRATION  3.2. SYSTEM REQUIREMENT  3.2.1. HARDWARE REQUIREMENTS  3.2.2. SOFTWARE REQUIREMENTS | **17**  **17**  **17**  **19**  **19**  **20**  **20** |
| **4.** | **MODULE DESCRIPTION**  4.1 IMAGE MODALITY USAGE  4.2 COMPARING U-NET AND ATTENTION  U-NET  4.3 DICE COEFFICIENT FOR TUMOR  REGIONS  4.4 MODEL CONNECTIONS | **21**  **21**  **22**  **25**  **27** |
|  |
| **5.** | **IMPREMENTATION AND RESULTS**  5.1. DATASET DESCRIPTION  5.2. PREPROCESSING STEPS  5.3. U-NET ARCHITECTURE WITH  ATTENTION MECHANISM  5.4. TRAINING PROCEDURE  5.5. EVALUATION METRICS  5.6 SEGMENTATION PERFORMANCE  5.7 IMPACT OF ATTENTION MECHANISM  5.7.1 DICE COEFFICIENT COMPARISON  5.7.2 COMPARISON WITH BENCHMARK | **46**  46  48  51  51  53  55  56  58  59 |
| **6.** | **TEST CASES AND PERFORMANCE METRICS**  6.1 TEST CASES  6.2 PERFORMANCE METRICS | **61**  **61**  **63** |
| **7.** | **CONCLUSION AND FUTURE WORK**  7.1. CONCLUSION  7.2. FUTURE WORK  **REFERENCES** | **65**  **65**  **65**  **66** |

## LIST OF TABLES

|  |  |  |
| --- | --- | --- |
| **TABLE NO** | **TITLE** | **PAGE NO** |
| **2.1**  **4.1**  **5.1**    **5.2**  **5.3**  **5.4**  **5.5**  **6.1**  **6.2**  **6.3** | LITERATURE SURVEY  METRICS BETWEEN ATTENTION U-NET  AND U-NET  SEGMENTATION PERFORMANCE  U-NET DICE COEFFICIENT  ATTENTION DICE COEFFICIENT  PERFORMANCE METRICS OF ATTENTION  U-NET DATASET SPLIT WISE.  PERFORMANCE METRICS OF U-NET  DATASET SPLIT WISE  TEST CASES  U-NET PERFORMANCE METRICS  ATTENTION U-NET PERFORMANCE METRICS | **14**  **24**  **55**  **58**  **58**  **59**  **60**  **61**  **63**  **63** |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **TITLE** | **PAGE NO** |
| **3.1** | U-NET NETWORK STRUCTURE | 17 |
| **3.2** | ATTENTION MECHANISM INTEGRATION | 19 |
| **4.1** | MODALITY OUTPUT | 22 |
| **4.2.1** | U-NET ACCURACY OUTPUT | 23 |
| **4.2.2** | ATTENTION U-NET ACCURACY OUTPUT | 24 |
| **5.1** | DATA DISTRIBUTION | 47 |
| **5.2.1** | CROPPING | 49 |
| **5.2.2** | FLAIR IMAGE | 50 |
| **5.2.3** | T1 IMAGE | 50 |
| **5.2.4** | T1CE IMAGE | 50 |
| **5.2.5** | T2 IMAGE | 50 |
| **5.3** | BENCHMARK | 59 |

**CHAPTER 1**

# INTRODUCTION

**1.1 BACKGROUND**

Accurate segmentation of brain tumors from magnetic resonance imaging (MRI) is critical in clinical decision-making. Segmentation allows doctors to distinguish between healthy brain tissue and malignant regions, which is critical for assessing tumor size, guiding surgical planning, and tracking treatment response. The diverse structure of tumors in MRI scans, however, makes brain tumor segmentation difficult. These difficulties include differences in tumor appearance, the presence of surrounding edema, and picture noise. Traditional image segmentation approaches frequently struggle with these intricacies, potentially leading to erroneous delineation of tumor borders.

Deep learning algorithms have transformed the field of medical image analysis, creating new opportunities for accurate and automated brain tumor segmentation. Deep learning algorithms can learn complicated patterns from massive collections of medical images, outperforming standard methods. However, there is an increasing demand for resource-efficient deep learning models in medical picture segmentation. This is caused by computational resource constraints in some clinical contexts, notably when real-time segmentation is required during surgical procedures. Traditional deep learning architectures sometimes need large computational resources, making their real-world application difficult in resource-constrained situations.

**1.2 PROBLEM STATEMENT**

A variety of limitations in existing methodologies, MRI scans provide intrinsic difficulty in accurately segmenting brain tumors. These problems include tumors' complex and diverse nature, which can emerge as differences in appearance, the presence of surrounding edema, and imaging noise. Furthermore, precisely discriminating between different tumor sub-regions such as necrotic tissue, edema, and enhancing regions is critical for diagnosis and treatment planning, especially when deciding between high-grade and low-grade gliomas.

Traditional segmentation algorithms frequently struggle with these complexity and may fail to identify small differences within brain tumors. Furthermore, existing deep learning models, such as U-Net topologies, may suffer from redundant computations caused by skipped connections. This redundancy can result in inefficiencies in processing speed and, potentially, mistakes in tumor categorization.

This research project aims to address these limitations by developing a resource-efficient brain tumor segmentation method using an attention U-Net architecture. By incorporating an attention mechanism, the model can focus on the most relevant features within the MRI scans, potentially improving segmentation accuracy while minimizing redundant computations and reducing the overall computational footprint.

**1.3. RESEARCH OBJECTIVES**

The objective of the research is to create a resource-efficient deep learning model for brain tumor segmentation that makes use of an attention U-Net architecture. We will compare its performance to standard models based on accuracy and computing efficiency (Objectives 1 and 2). Furthermore, we will look into how the attention process affects the model's capacity to recognize critical features, notably when distinguishing between necrotic, edema, and enhancing regions in tumors (Objective 3).

**1.4.SCOPE OF THE PROJECT**

This project focuses on developing and evaluating a resource-efficient deep learning model for brain tumor segmentation using an attention U-Net architecture. We will utilize a publicly available brain tumor segmentation dataset containing MRI scans. The specific dataset and any pre-processing steps will be clearly defined. The core study involves designing and implementing the attention U-Net with hyperparameter tuning for both accuracy and resource efficiency. We will compare the proposed model's performance against a baseline deep learning model using established metrics like accuracy , MeanIOU and Dice coefficient. Additionally, the project will analyze the computational footprint of the proposed model in terms of training time, memory usage, and potential inference speed. The scope may be extended to investigate the model's ability to differentiate between tumor sub-regions, depending on the chosen dataset and research objectives.

\

## CHAPTER 2

**LITERATURE SURVEY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **Name Of Paper** | **Author** | **Methodology** | **Limitations** |
| 1 | An enhanced deep learning approach for brain cancer MRI images classification using residual networks | S. A. A. Ismael, A. Mohammed, and H. Hefny | Residual Networks (ResNets)  Brain Cancer MRI Images Classification:  Enhanced Deep Learning Approach: | Data limitations  Overfitting  Interpretability |
| 2 | Brain tumour detection from images and comparison with transfer learning methods and 3-layer CNN | Mohammad Zafer Khaliki & Muhammet Sinan Başarslan | Model Selection: Utilizes Convolutional Neural Network (CNN) architecture and transfer learning methods (InceptionV3, VGG16, VGG19, EfficientNetB4). | Computational Resources: Training complex models like transfer learning architectures may require significant computational resources. |
| 3 | Brain Tumor Segmentation from MRI Images Using Handcrafted Convolutional Neural Network | Faizan Ullah, Muhammad Nadeem, Mohammad Abrar | Feature Extraction  CNN Architecture  Feature Integration  Fine-Tuning | Complexity  Data Needs  Computational Demands  Interpretability |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | Brain tumour segmentation based on deep learning and an attention mechanism using MRI multi-modalities brain images | Ramin Ranjbarzadeh, Abbas Bagherian Kasgari, Saeid Jafarzadeh | Modalities  Preprocessing  DWA  C-CNN | Complexity in the process of finding the tumor areas may require manual tuning.  Dependency on preprocessing steps for segmentation accuracy. |
| 5. | Brain MRI image classification for cancer detection using deep wavelet autoencoder-based deep neural network | P. K. Mallick, S. H. Ryu, S. K. Satapathy, S. Mishra, G. N. Nguyen, and P. Tiwari | Deep Wavelet Autoencoder (DWAE)  Deep Neural Network (DNN) | Limited data  Black-box nature  Hyperparameter tuning |
| 6. | Microscopic brain tumor detection and classification using 3D CNN and feature selection architecture | A. Rehman, M. A. Khan, T. Saba, Z. Mehmood, U. Tariq, and N. Ayesha | 3D Convolutional Neural Network (CNN)  Classification  Feature Selection | Data limitations  Interpretability  Overfitting |

Table 2.1 LITERATURE SURVEY

**2.2. SUMMARY OF LITERATURE SURVEY**

The studies primarily focused on the classification and segmentation of brain malignancies in MRI images, employing various deep learning methods. For classification, they utilized transfer learning techniques (InceptionV3, VGG16, VGG19, EfficientNetB4), Residual Networks (ResNets), and Convolutional Neural Networks (CNNs). Segmentation, on the other hand, involved the application of deep learning with attention mechanisms on multi-modal MRI images.

These methods were designed to address challenges related to interpretability, overfitting, data scarcity, and computational resource requirements. However, they also highlighted the complexity of the segmentation process, the need for manual adjustment, and the critical role of preprocessing procedures in achieving accurate results.

In addition to classification and segmentation, the studies explored methods for feature extraction, classification, and fine-tuning. These included the use of 3D CNN, handmade CNN architectures, and Deep Wavelet Autoencoder (DWAE). Despite their effectiveness, some models were criticized for their "black-box" nature, highlighting the importance of interpretability in medical imaging. Additionally, challenges such as data scarcity and the need for meticulous hyperparameter adjustment were also acknowledged.

**CHAPTER 3**

**SYSTEM DESIGN**

**3.1. SYSTEM ARCHITECHTURE**

This section describes the core of our system: the deep learning model for brain tumor segmentation. We propose an Attention U-Net architecture that leverages the strengths of U-Net for image segmentation while incorporating an attention mechanism to focus on critical regions for improved tumor delineation.

**3.1.1. U-Net Network Structure**

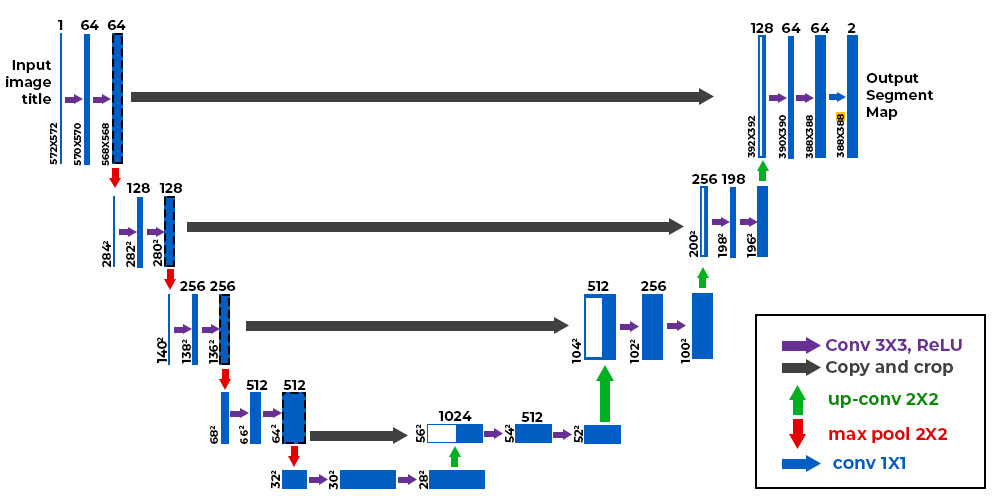
****

Fig 3.1 U-Net Network Structure

Our U-Net architecture employs a standard encoder-decoder structure with skip connections. The encoder path progressively extracts features from the input brain MRI image.

**Encoder:**

The encoder consists of 4 contracting blocks.

Each block contains two 3x3 convolutional layers with rectified linear unit (ReLU) activation functions followed by a 2x2 max pooling operation for downsampling. The number of filters doubles in each subsequent block (starting from 32 in the first block).

**Decoder:**

The decoder path upsamples the captured features and combines them with high-resolution information from the encoder through skip connections. It consists of 4 expansive blocks, mirroring the encoder structure but using transposed convolution operations for upsampling. Each block contains two 3x3 transposed convolutional layers with ReLU activations followed by a concatenation layer that merges features from the corresponding encoder block. The number of filters is halved in each block compared to the encoder (starting from features with the same number of filters as the final encoder block).

**Skip Connections:**

Skip connections directly concatenate feature maps from the corresponding encoder and decoder blocks at the same resolution level.

This helps retain spatial information lost during downsampling and improves localization accuracy in the segmentation task.

**3.1.2.** **Attention Mechanism Integration**

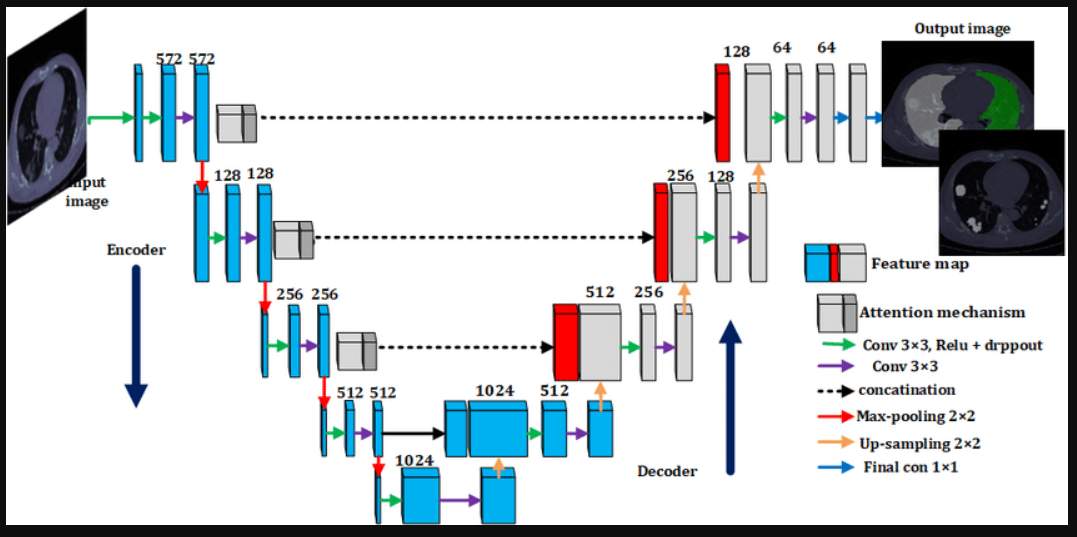


Fig 3.2. Attention Mechanism Integration

We integrate a channel-wise attention mechanism within the skip connections between the encoder and decoder blocks. This allows the model to focus on informative channels within the feature maps that are crucial for tumor segmentation.

The attention module operates on the concatenated feature maps from the encoder and decoder at each skip connection level. It calculates weights for each channel based on their importance for tumor segmentation. These weights are then multiplied element-wise with the corresponding channels in the feature maps, effectively amplifying informative channels and suppressing less relevant ones. The resulting weighted feature maps are fed into the subsequent decoder block.

**3.2. SYSTEM REQUIREMENTS**

This section outlines the computational resources necessary to train and run the proposed Attention U-Net model effectively.

**3.2.1. Hardware Requirements**

For efficient training and running of the model, we recommend a system with the following minimum hardware specifications:

**GPU:** NVIDIA GTX 1080 Ti or equivalent with at least 11GB of dedicated

GPU memory

**CPU:** Minimum 8 cores with at least 16GB of RAM

A more powerful GPU with greater memory can significantly improve training speed, especially for larger datasets.

**3.2.2. Software Requirements**

The model implementation utilizes the following software tools:

**Essentials:**

TensorFlow/PyTorch (Deep Learning Framework)

NumPy (Numerical Computations)

Matplotlib (Data Visualization)

**Medical Imaging Focus:**

Pandas (Data Manipulation)

Nibabel (Medical Image Formats)

Nilearn (Brain Imaging Analysis)

Albumentations (Image Augmentation)

Scikit-image (Image Processing)

**CHAPTER 4**

**MODULE DESCRIPTION**

**4.1 IMAGE MODALITY USAGE**

This project employed a multi-modal MRI approach for training and evaluation. MRI scans were chosen due to their excellent soft tissue contrast, which is particularly valuable for tasks like brain tumor segmentation (assuming that's your task). The model utilized five MRI sequences:

**FLAIR (Fluid-attenuated inversion recovery):** This sequence is highly sensitive to white matter lesions and fluid-filled cavities, making it useful for tumor detection and segmentation.

**T1-weighted (T1):** T1 sequences provide high anatomical detail, aiding in visualizing overall brain structure and tumor location.

**T1 contrast-enhanced (T1ce):** Contrast agents enhance tumor regions in T1ce images, making them more distinct from surrounding tissues.

**T2-weighted (T2):** T2 sequences offer good contrast between fluids and tissues, helping to identify tumor borders and edema (fluid buildup).

Code snippet:

fig, (ax1, ax2, ax3, ax4, ax5) = plt.subplots(1,5, figsize = (20, 10))

slice\_w = 25

ax1.imshow(test\_image\_flair[:,:,test\_image\_flair.shape[0]//2-slice\_w], cmap = 'gray')

ax1.set\_title('Image flair')

ax2.imshow(test\_image\_t1[:,:,test\_image\_t1.shape[0]//2-slice\_w], cmap = 'gray')

ax2.set\_title('Image t1')

ax3.imshow(test\_image\_t1ce[:,:,test\_image\_t1ce.shape[0]//2-slice\_w], cmap = 'gray')

ax3.set\_title('Image t1ce')

ax4.imshow(test\_image\_t2[:,:,test\_image\_t2.shape[0]//2-slice\_w], cmap = 'gray')

ax4.set\_title('Image t2')

ax5.imshow(test\_mask[:,:,test\_mask.shape[0]//2-slice\_w])

ax5.set\_title('Mask')

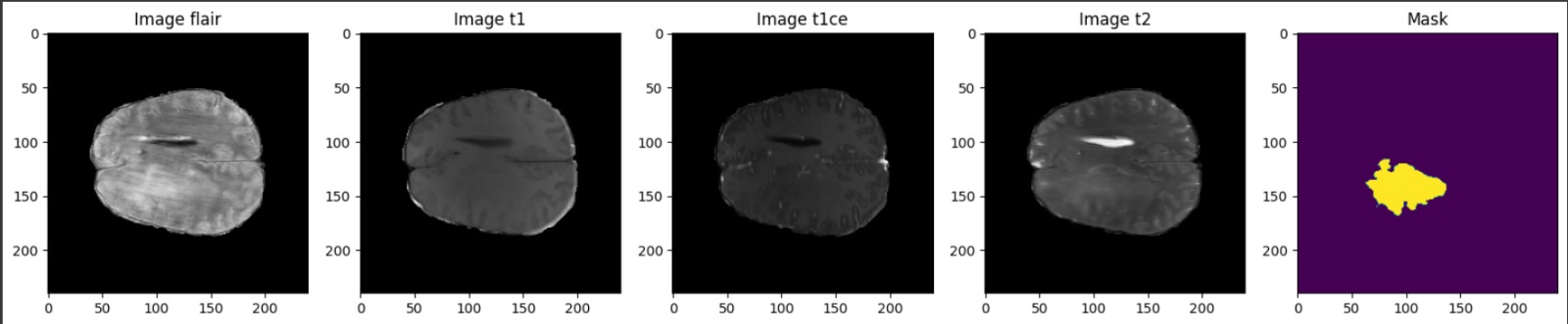
**Output:**

Fig 4.1 Modality output

The code snippet visualizes a central slice from each modality (FLAIR, T1, T1ce, T2) alongside the corresponding ground truth segmentation mask. This approach leverages the complementary information provided by each MRI sequence to improve the model's segmentation accuracy.

**4.2 COMPARING U-NET AND ATTENTION U-NET**

In this project, we investigated the potential benefits of using an Attention U-Net architecture compared to a standard U-Net for [your image segmentation task]. Both architectures are convolutional neural networks (CNNs) commonly used for medical image segmentation tasks. Here's a breakdown of their key differences and how they might affect performance:

**U-Net:**

Strengths: U-Net is a well-established architecture with a strong track record in medical image segmentation. Its encoder-decoder structure with skip connections effectively captures contextual information at different resolutions.

Weaknesses: The standard U-Net might struggle to focus on crucial features within complex medical images, potentially leading to less accurate segmentation boundaries.

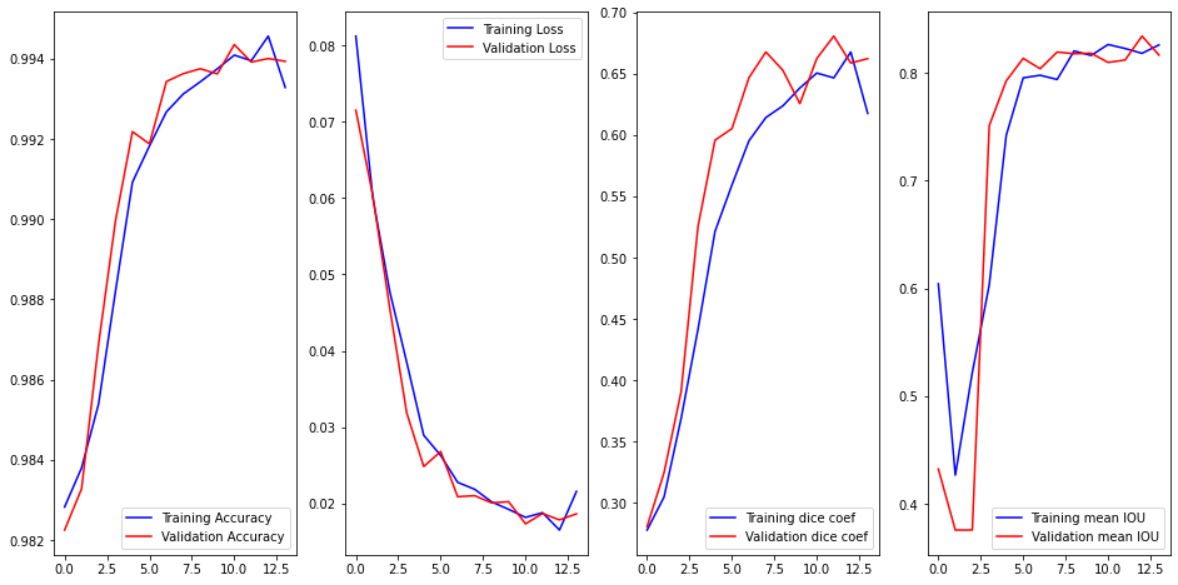


Fig 4.2.1 U- net Accuracy output

**Attention U-Net:**

Strengths: Attention U-Net builds upon U-Net by incorporating an attention mechanism. This mechanism allows the model to selectively focus on informative regions within the feature maps, potentially leading to:

Improved segmentation accuracy: By attending to relevant features, the model can achieve more precise segmentation boundaries.

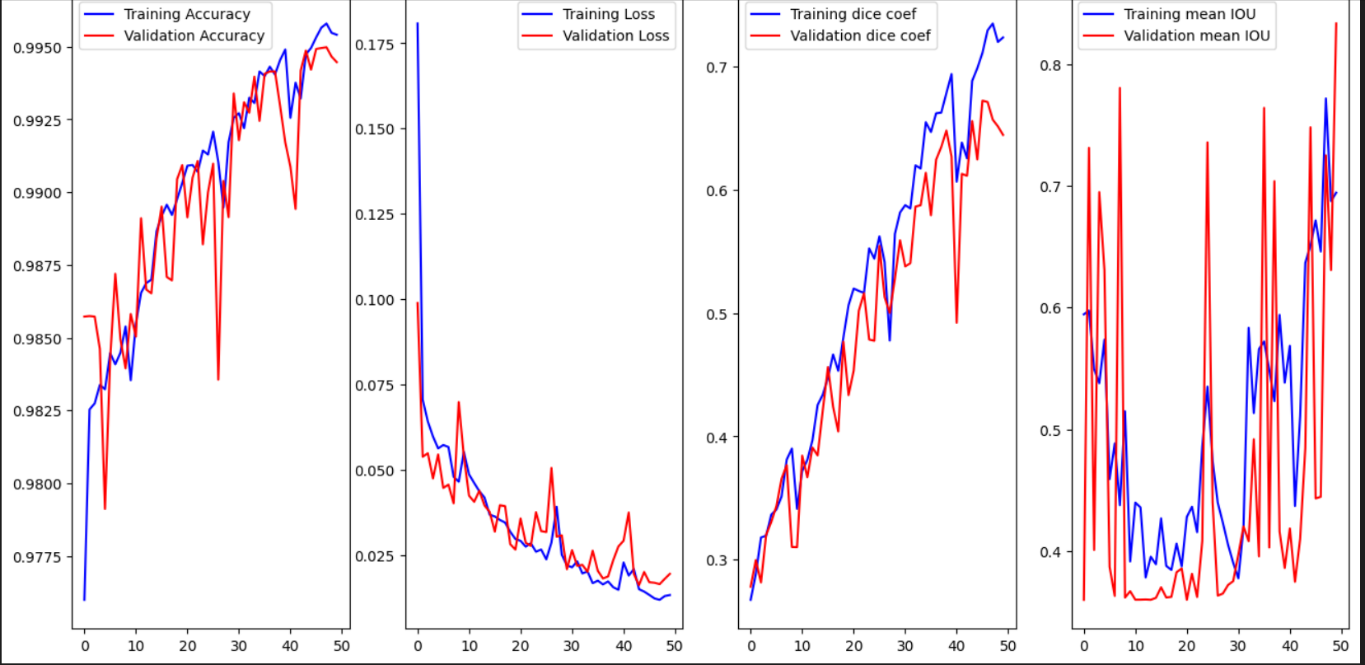
Better handling of complex structures: The attention mechanism can help the model prioritize crucial information, particularly in areas with intricate details.

Fig 4.2.2 Attention U- net Accuracy output

Weaknesses: Attention U-Nets generally require more computational resources for training compared to standard U-Nets due to the additional attention calculations.

|  |  |  |
| --- | --- | --- |
| **Metric** | **Attention U-Net** | **U-Net** |
| Test Loss | 0.0137 | 0.0166 |
| Test Accuracy | 0.9953 | 0.9947 |
| Mean IoU | 0.8343 | 0.8090 |
| Dice Coefficient (Overall) | 0.7396 | 0.6525 |
| Dice Coefficient (Necrotic) | 0.7627 | 0.6647 |
| Dice Coefficient (Edema) | 0.8467 | 0.759 |
| Dice Coefficient (Enhancing) | 0.8736 | 0.7758 |

Table 4.1 Metrics between Attention U-Net and U-Net

The Attention U-Net achieved a lower test loss (0.0137) compared to the U-Net (0.0166), indicating better overall fit to the training data.

Both models achieved very high test accuracy (above 0.994), suggesting good performance in classifying pixels correctly.

The Attention U-Net outperformed the U-Net in terms of Mean IoU (0.8343 vs. 0.8090). IoU measures the overlap between the predicted segmentation and the ground truth, with a higher value indicating better agreement. It's noteworthy that the U-Net's IoU is based on only two classes, while the Attention U-Net considers six classes.

The Attention U-Net also achieved higher Dice coefficients for all three sub-regions (necrotic, edema, enhancing) compared to the U-Net. The Dice coefficient measures the similarity between the predicted segmentation and the ground truth, ranging from 0 (no overlap) to 1 (perfect overlap). These results suggest that the Attention U-Net produced more accurate segmentations for each tissue type.

**4.3 DICE COEFFICIENT FOR TUMOR REGIONS**

Dice Coefficient Explained: A Measure of Segmentation Accuracy

The Dice coefficient, sometimes called Sørensen–Dice coefficient, is a common metric used to evaluate the performance of image segmentation models, particularly in medical imaging tasks. It measures the overlap between the predicted segmentation and the ground truth (actual segmentation).

**Overall Dice Coefficient:**

This metric captures the overall agreement between the entire predicted segmentation and the ground truth. A higher Dice coefficient (closer to 1) indicates a better match between the two.

It's a good starting point to assess the model's general segmentation performance.

**Dice Coefficient (Specific Regions):**

Regional Dice coefficients provide a more detailed understanding of how well the model segments each individual tissue type.

They help identify potential weaknesses of the model. For instance, a lower Dice coefficient for the necrotic core might suggest the model struggles to accurately segment the dead tissue region.

**Necrotic Dice Coefficient:** This metric focuses on how well the model segments the **necrotic core** of the tumor. The necrotic core is the dead tissue region within a tumor. A high Dice coefficient for necrosis indicates the model accurately identifies the dead tissue area.

**Edema Dice Coefficient:** This metric evaluates the model's ability to segment the **edema** surrounding the tumor. Edema refers to fluid buildup in the brain tissue caused by the tumor. A high Dice coefficient for edema suggests the model precisely identifies the edematous region.

**Enhancing Dice Coefficient:** This metric assesses how well the model segments the **enhancing tumor region**. This region often appears brighter in certain MRI sequences due to the presence of contrast agents. A high Dice coefficient for enhancing tumor indicates the model accurately identifies the actively growing portion of the tumor.

**Why Use Dice Coefficient?**

Here are some key advantages of using the Dice coefficient for segmentation evaluation:

**Intuitive Interpretation:** The Dice coefficient ranges from 0 (no overlap) to 1 (perfect overlap), making it easy to understand how well the segmentation matches the ground truth.

**Robust to Class Imbalance:** Unlike metrics like accuracy, Dice is less sensitive to class imbalance, which is often present in medical images where the foreground (tumor region) occupies a smaller area compared to the background (healthy tissue).

**Focuses on Boundaries:** The Dice coefficient penalizes both false positives (including irrelevant pixels in the segmentation) and false negatives (missing actual tumor pixels). This encourages the model to produce accurate segmentation boundaries.

**4.4 MODEL CONNECTIONS  
U-net :**

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 = (2,2))(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 = (2,2))(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 = (2,2))(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 = (2,2))(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, (1,1), activation = 'softmax')(conv)

return Model(inputs = inputs, outputs = conv10)

input\_layer = Input((IMG\_SIZE, IMG\_SIZE, 2))

model = build\_unet(input\_layer, 'he\_normal', 0.2)

model.compile(loss="categorical\_crossentropy", optimizer=tf.keras.optimizers.Adam(learning\_rate=0.001), metrics = ['accuracy',tf.keras.metrics.MeanIoU(num\_classes=4), dice\_coef, precision, sensitivity, specificity, dice\_coef\_necrotic, dice\_coef\_edema ,dice\_coef\_enhancing] )

Layer (type) Output Shape Param # Connected to

input\_1 (InputLayer) [(None, 128, 128, 2) 0

conv2d (Conv2D) (None, 128, 128, 32) 608 input\_1[0][0]

conv2d\_1 (Conv2D) (None, 128, 128, 32) 9248 conv2d[0][0]

max\_pooling2d (MaxPooling2D) (None, 64, 64, 32) 0 conv2d\_1[0][0]

conv2d\_2 (Conv2D) (None, 64, 64, 64) 18496 max\_pooling2d[0][0]

conv2d\_3 (Conv2D) (None, 64, 64, 64) 36928 conv2d\_2[0][0]

max\_pooling2d\_1 (MaxPooling2D) (None, 32, 32, 64) 0 conv2d\_3[0][0]

conv2d\_4 (Conv2D) (None, 32, 32, 128) 73856 max\_pooling2d\_1[0][0]

conv2d\_5 (Conv2D) (None, 32, 32, 128) 147584 conv2d\_4[0][0]

max\_pooling2d\_2 (MaxPooling2D) (None, 16, 16, 128) 0 conv2d\_5[0][0]

conv2d\_6 (Conv2D) (None, 16, 16, 256) 295168 max\_pooling2d\_2[0][0]

conv2d\_7 (Conv2D) (None, 16, 16, 256) 590080 conv2d\_6[0][0]

max\_pooling2d\_3 (MaxPooling2D) (None, 8, 8, 256) 0 conv2d\_7[0][0]

conv2d\_8 (Conv2D) (None, 8, 8, 512) 1180160 max\_pooling2d\_3[0][0]

conv2d\_9 (Conv2D) (None, 8, 8, 512) 2359808 conv2d\_8[0][0]

dropout (Dropout) (None, 8, 8, 512) 0 conv2d\_9[0][0]

up\_sampling2d (UpSampling2D) (None, 16, 16, 512) 0 dropout[0][0]

conv2d\_10 (Conv2D) (None, 16, 16, 256) 524544 up\_sampling2d[0][0]

concatenate (Concatenate) (None, 16, 16, 512) 0 conv2d\_7[0][0]

conv2d\_10[0][0]

conv2d\_11 (Conv2D) (None, 16, 16, 256) 1179904 concatenate[0][0]

conv2d\_12 (Conv2D) (None, 16, 16, 256) 590080 conv2d\_11[0][0]

up\_sampling2d\_1 (UpSampling2D) (None, 32, 32, 256) 0 conv2d\_12[0][0]

conv2d\_13 (Conv2D) (None, 32, 32, 128) 131200 up\_sampling2d\_1[0][0]

concatenate\_1 (Concatenate) (None, 32, 32, 256) 0 conv2d\_5[0][0]

conv2d\_13[0][0]

conv2d\_14 (Conv2D) (None, 32, 32, 128) 295040 concatenate\_1[0][0]

conv2d\_15 (Conv2D) (None, 32, 32, 128) 147584 conv2d\_14[0][0]

up\_sampling2d\_2 (UpSampling2D) (None, 64, 64, 128) 0 conv2d\_15[0][0]

conv2d\_16 (Conv2D) (None, 64, 64, 64) 32832 up\_sampling2d\_2[0][0]

concatenate\_2 (Concatenate) (None, 64, 64, 128) 0 conv2d\_3[0][0]

conv2d\_16[0][0]

conv2d\_17 (Conv2D) (None, 64, 64, 64) 73792 concatenate\_2[0][0]

conv2d\_18 (Conv2D) (None, 64, 64, 64) 36928 conv2d\_17[0][0]

up\_sampling2d\_3 (UpSampling2D) (None, 128, 128, 64) 0 conv2d\_18[0][0]

conv2d\_19 (Conv2D) (None, 128, 128, 32) 8224 up\_sampling2d\_3[0][0]

concatenate\_3 (Concatenate) (None, 128, 128, 64) 0 conv2d\_1[0][0]

conv2d\_19[0][0]

conv2d\_20 (Conv2D) (None, 128, 128, 32) 18464 concatenate\_3[0][0]

conv2d\_21 (Conv2D) (None, 128, 128, 32) 9248 conv2d\_20[0][0]

conv2d\_22 (Conv2D) (None, 128, 128, 4) 132 conv2d\_21[0][0]

Total params: 7,759,908

Trainable params: 7,759,908

Non-trainable params: 0

**Attention U-net Model:**

import tensorflow as tf

from tensorflow.keras.layers import Conv2D, MaxPooling2D, UpSampling2D, concatenate, Dropout, Input

from tensorflow.keras.models import Model

def attention\_block\_2d(x, g, inter\_channel):

    theta\_x = Conv2D(inter\_channel, (1, 1), strides=(1, 1), padding='same')(x)

    phi\_g = Conv2D(inter\_channel, (1, 1), strides=(1, 1), padding='same')(g)

    f = tf.keras.layers.Add()([theta\_x, phi\_g])

    f = tf.keras.layers.Activation('relu')(f)

    psi\_f = Conv2D(1, (1, 1), strides=(1, 1), padding='same')(f)

    rate = tf.keras.layers.Activation('sigmoid')(psi\_f)

    att\_x = tf.keras.layers.Multiply()([x, rate])

    return att\_x

def build\_unet\_with\_attention(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)

    att1 = attention\_block\_2d(conv1, conv1, 16)

    pool1 = MaxPooling2D(pool\_size=(2, 2))(att1)

    conv2 = Conv2D(64, 3, activation='relu', padding='same', kernel\_initializer=ker\_init)(pool1)

    conv2 = Conv2D(64, 3, activation='relu', padding='same', kernel\_initializer=ker\_init)(conv2)

    att2 = attention\_block\_2d(conv2, conv2, 32)

    pool2 = MaxPooling2D(pool\_size=(2, 2))(att2)

    conv3 = Conv2D(128, 3, activation='relu', padding='same', kernel\_initializer=ker\_init)(pool2)

    conv3 = Conv2D(128, 3, activation='relu', padding='same', kernel\_initializer=ker\_init)(conv3)

    att3 = attention\_block\_2d(conv3, conv3, 64)

    pool3 = MaxPooling2D(pool\_size=(2, 2))(att3)

    conv4 = Conv2D(256, 3, activation='relu', padding='same', kernel\_initializer=ker\_init)(pool3)

    conv4 = Conv2D(256, 3, activation='relu', padding='same', kernel\_initializer=ker\_init)(conv4)

    att4 = attention\_block\_2d(conv4, conv4, 128)

    pool4 = MaxPooling2D(pool\_size=(2, 2))(att4)

    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)

    return drop5

def build\_decoder(drop5, ker\_init, dropout):

    up6 = Conv2D(256, 2, activation='relu', padding='same', kernel\_initializer=ker\_init)(

        UpSampling2D(size=(2, 2))(drop5))

    conv6 = Conv2D(256, 3, activation='relu', padding='same', kernel\_initializer=ker\_init)(up6)

    conv6 = Conv2D(256, 3, activation='relu', padding='same', kernel\_initializer=ker\_init)(conv6)

    up7 = Conv2D(128, 2, activation='relu', padding='same', kernel\_initializer=ker\_init)(

        UpSampling2D(size=(2, 2))(conv6))

    conv7 = Conv2D(128, 3, activation='relu', padding='same', kernel\_initializer=ker\_init)(up7)

    conv7 = Conv2D(128, 3, activation='relu', padding='same', kernel\_initializer=ker\_init)(conv7)

    up8 = Conv2D(64, 2, activation='relu', padding='same', kernel\_initializer=ker\_init)(

        UpSampling2D(size=(2, 2))(conv7))

    conv8 = Conv2D(64, 3, activation='relu', padding='same', kernel\_initializer=ker\_init)(up8)

    conv8 = Conv2D(64, 3, activation='relu', padding='same', kernel\_initializer=ker\_init)(conv8)

    up9 = Conv2D(32, 2, activation='relu', padding='same', kernel\_initializer=ker\_init)(

        UpSampling2D(size=(2, 2))(conv8))

    conv9 = Conv2D(32, 3, activation='relu', padding='same', kernel\_initializer=ker\_init)(up9)

    conv9 = Conv2D(32, 3, activation='relu', padding='same', kernel\_initializer=ker\_init)(conv9)

    conv10 = Conv2D(4, (1, 1), activation='softmax')(conv9)

    return conv10

IMG\_SIZE = 256  # Set according to your requirement

input\_layer = Input((IMG\_SIZE, IMG\_SIZE, 2))

#

pool4 = build\_unet\_with\_attention(input\_layer, 'he\_normal', 0.2)

output\_layer = build\_decoder(pool4, 'he\_normal', 0.2)

model = Model(inputs=input\_layer, outputs=output\_layer)

model.compile(loss="categorical\_crossentropy", optimizer=tf.keras.optimizers.Adam(learning\_rate=0.001),

              metrics = ['accuracy',tf.keras.metrics.MeanIoU(num\_classes=4), dice\_coef, precision, sensitivity, specificity, dice\_coef\_necrotic, dice\_coef\_edema ,dice\_coef\_enhancing] )

Model: "model"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param # Connected to

==================================================================================================

input\_1 (InputLayer) [(None, 256, 256, 2)] 0 []

conv2d (Conv2D) (None, 256, 256, 32) 608 ['input\_1[0][0]']

conv2d\_1 (Conv2D) (None, 256, 256, 32) 9248 ['conv2d[0][0]']

conv2d\_2 (Conv2D) (None, 256, 256, 16) 528 ['conv2d\_1[0][0]']

conv2d\_3 (Conv2D) (None, 256, 256, 16) 528 ['conv2d\_1[0][0]']

add (Add) (None, 256, 256, 16) 0 ['conv2d\_2[0][0]',

'conv2d\_3[0][0]']

activation (Activation) (None, 256, 256, 16) 0 ['add[0][0]']

conv2d\_4 (Conv2D) (None, 256, 256, 1) 17 ['activation[0][0]']

activation\_1 (Activation) (None, 256, 256, 1) 0 ['conv2d\_4[0][0]']

multiply (Multiply) (None, 256, 256, 32) 0 ['conv2d\_1[0][0]',

'activation\_1[0][0]']

max\_pooling2d (MaxPooling2 (None, 128, 128, 32) 0 ['multiply[0][0]']

D)

conv2d\_5 (Conv2D) (None, 128, 128, 64) 18496 ['max\_pooling2d[0][0]']

conv2d\_6 (Conv2D) (None, 128, 128, 64) 36928 ['conv2d\_5[0][0]']

conv2d\_7 (Conv2D) (None, 128, 128, 32) 2080 ['conv2d\_6[0][0]']

conv2d\_8 (Conv2D) (None, 128, 128, 32) 2080 ['conv2d\_6[0][0]']

add\_1 (Add) (None, 128, 128, 32) 0 ['conv2d\_7[0][0]',

'conv2d\_8[0][0]']

activation\_2 (Activation) (None, 128, 128, 32) 0 ['add\_1[0][0]']

conv2d\_9 (Conv2D) (None, 128, 128, 1) 33 ['activation\_2[0][0]']

activation\_3 (Activation) (None, 128, 128, 1) 0 ['conv2d\_9[0][0]']

multiply\_1 (Multiply) (None, 128, 128, 64) 0 ['conv2d\_6[0][0]',

'activation\_3[0][0]']

max\_pooling2d\_1 (MaxPoolin (None, 64, 64, 64) 0 ['multiply\_1[0][0]']

g2D)

conv2d\_10 (Conv2D) (None, 64, 64, 128) 73856 ['max\_pooling2d\_1[0][0]']

conv2d\_11 (Conv2D) (None, 64, 64, 128) 147584 ['conv2d\_10[0][0]']

conv2d\_12 (Conv2D) (None, 64, 64, 64) 8256 ['conv2d\_11[0][0]']

conv2d\_13 (Conv2D) (None, 64, 64, 64) 8256 ['conv2d\_11[0][0]']

add\_2 (Add) (None, 64, 64, 64) 0 ['conv2d\_12[0][0]',

'conv2d\_13[0][0]']

activation\_4 (Activation) (None, 64, 64, 64) 0 ['add\_2[0][0]']

conv2d\_14 (Conv2D) (None, 64, 64, 1) 65 ['activation\_4[0][0]']

activation\_5 (Activation) (None, 64, 64, 1) 0 ['conv2d\_14[0][0]']

multiply\_2 (Multiply) (None, 64, 64, 128) 0 ['conv2d\_11[0][0]',

'activation\_5[0][0]']

max\_pooling2d\_2 (MaxPoolin (None, 32, 32, 128) 0 ['multiply\_2[0][0]']

g2D)

conv2d\_15 (Conv2D) (None, 32, 32, 256) 295168 ['max\_pooling2d\_2[0][0]']

conv2d\_16 (Conv2D) (None, 32, 32, 256) 590080 ['conv2d\_15[0][0]']

conv2d\_17 (Conv2D) (None, 32, 32, 128) 32896 ['conv2d\_16[0][0]']

conv2d\_18 (Conv2D) (None, 32, 32, 128) 32896 ['conv2d\_16[0][0]']

add\_3 (Add) (None, 32, 32, 128) 0 ['conv2d\_17[0][0]',

'conv2d\_18[0][0]']

activation\_6 (Activation) (None, 32, 32, 128) 0 ['add\_3[0][0]']

conv2d\_19 (Conv2D) (None, 32, 32, 1) 129 ['activation\_6[0][0]']

activation\_7 (Activation) (None, 32, 32, 1) 0 ['conv2d\_19[0][0]']

multiply\_3 (Multiply) (None, 32, 32, 256) 0 ['conv2d\_16[0][0]',

'activation\_7[0][0]']

max\_pooling2d\_3 (MaxPoolin (None, 16, 16, 256) 0 ['multiply\_3[0][0]']

g2D)

conv2d\_20 (Conv2D) (None, 16, 16, 512) 1180160 ['max\_pooling2d\_3[0][0]']

conv2d\_21 (Conv2D) (None, 16, 16, 512) 2359808 ['conv2d\_20[0][0]']

dropout (Dropout) (None, 16, 16, 512) 0 ['conv2d\_21[0][0]']

up\_sampling2d (UpSampling2 (None, 32, 32, 512) 0 ['dropout[0][0]']

D)

conv2d\_22 (Conv2D) (None, 32, 32, 256) 524544 ['up\_sampling2d[0][0]']

conv2d\_23 (Conv2D) (None, 32, 32, 256) 590080 ['conv2d\_22[0][0]']

conv2d\_24 (Conv2D) (None, 32, 32, 256) 590080 ['conv2d\_23[0][0]']

up\_sampling2d\_1 (UpSamplin (None, 64, 64, 256) 0 ['conv2d\_24[0][0]']

g2D)

conv2d\_25 (Conv2D) (None, 64, 64, 128) 131200 ['up\_sampling2d\_1[0][0]']

conv2d\_26 (Conv2D) (None, 64, 64, 128) 147584 ['conv2d\_25[0][0]']

conv2d\_27 (Conv2D) (None, 64, 64, 128) 147584 ['conv2d\_26[0][0]']

up\_sampling2d\_2 (UpSamplin (None, 128, 128, 128) 0 ['conv2d\_27[0][0]']

g2D)

conv2d\_28 (Conv2D) (None, 128, 128, 64) 32832 ['up\_sampling2d\_2[0][0]']

conv2d\_29 (Conv2D) (None, 128, 128, 64) 36928 ['conv2d\_28[0][0]']

conv2d\_30 (Conv2D) (None, 128, 128, 64) 36928 ['conv2d\_29[0][0]']

up\_sampling2d\_3 (UpSamplin (None, 256, 256, 64) 0 ['conv2d\_30[0][0]']

g2D)

conv2d\_31 (Conv2D) (None, 256, 256, 32) 8224 ['up\_sampling2d\_3[0][0]']

conv2d\_32 (Conv2D) (None, 256, 256, 32) 9248 ['conv2d\_31[0][0]']

conv2d\_33 (Conv2D) (None, 256, 256, 32) 9248 ['conv2d\_32[0][0]']

conv2d\_34 (Conv2D) (None, 256, 256, 4) 132 ['conv2d\_33[0][0]']

==================================================================================================

Total params: 7064312 (26.95 MB)

Trainable params: 7064312 (26.95 MB)

Non-trainable params: 0 (0.00 Byte)

## CHAPTER 5

## MATERIALS AND METHODS

**5.1. Dataset Description**

We utilize the Medical Image Computing and Computer Assisted Interventions (MICCAI) Brain Tumor Segmentation (BraTS) dataset which consists of 369 labelled training samples and 125 unlabelled validation samples of preoperative MRI Brain scans from 19 different institutions. Each sample comprises an image with 240x240x155 voxels saved in the Neuroimaging Informatics Technology Initiative (NIfTI) file format with the file extension ".nii.gz". For the purpose of training and evaluation, we discard the 125 unlabelled validation samples and split the remaining 369 labelled training samples into train-val-test splits of 263-53-53 (a ratio of approximately 5:1:1).

Each sample is comprised of 4 channels/modalities: T1-Weighted (T1), Post-contrast T1-Weighted (T1c), T2-Weighted (T2) and T2 Fluid Attenuated Inversion Recovery (T2-FLAIR). By combining different types of images, doctors gain a more complete picture of a patient's condition and make more informed decisions about treatment.

T1-weighted images are created using specific settings during an MRI scan that highlight differences in the relaxation times of tissues in the body. In medical imaging, T1-weighted images are particularly useful for imaging the brain and other soft tissues.

T1c images are "contrast-enhanced", meaning that a contrast agent is used during the scan to make pertinent structures in the brain more visible.

T2-weighted images are characterized by their ability to show the differences in the relaxation times of tissues in the body. T1-weighted images are good at showing details of anatomical structures, while T2-weighted images are useful in detecting abnormalities in fluid-filled spaces and soft tissues.

T2-Flair is similar to T2-weighted imaging in that it uses a magnetic field to highlight differences in tissue relaxation times in the body. However, it also incorporates a technique called fluid-attenuated inversion recovery, which suppresses the signal from cerebrospinal fluid (CSF) in the brain, making it easier to identify abnormalities in adjacent tissues.

**Original Annotations**

The original annotations in the dataset comprise the following:

Label 1: Necrotic and non-enhancing tumour core (NCR/NET)

Label 2: Peritumoral edema (ED)

Label 4: GD-enhancing tumor (ET)

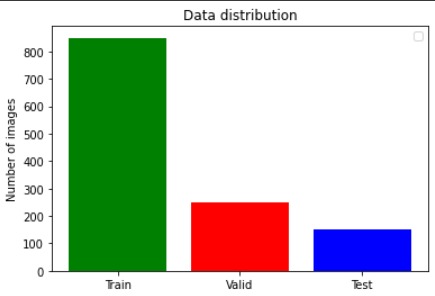


Fig 5.1 Data distribution

850 - Train

251 - Valid

150 - Test

**5.2. Preprocessing Steps**

**Resizing**: The code resizes the 3D medical images to a fixed size of 128x128x100 (IMG\_SIZE=128, VOLUME\_SLICES=100). This is done to ensure that the model can process all images with the same dimensions and simplifies training.

**Normalization**: The code normalizes the intensity values of the images (if normalize=True is set in ImageReader).exclamation This is typically done by subtracting the mean intensity from each voxel and then dividing by the standard deviation. Normalization helps to improve the training process by centering the data and making it less sensitive to outliers.expand\_more.

Code:   reader = ImageReader('./data', img\_size=128, normalize=True, single\_class=False)

**Cropping**: The code extracts a specific volume from the original 3D data (VOLUME\_START\_AT and VOLUME\_SLICES). This is done to focus on the relevant region of the brain and reduce the amount of data the model needs to process.exclamation.

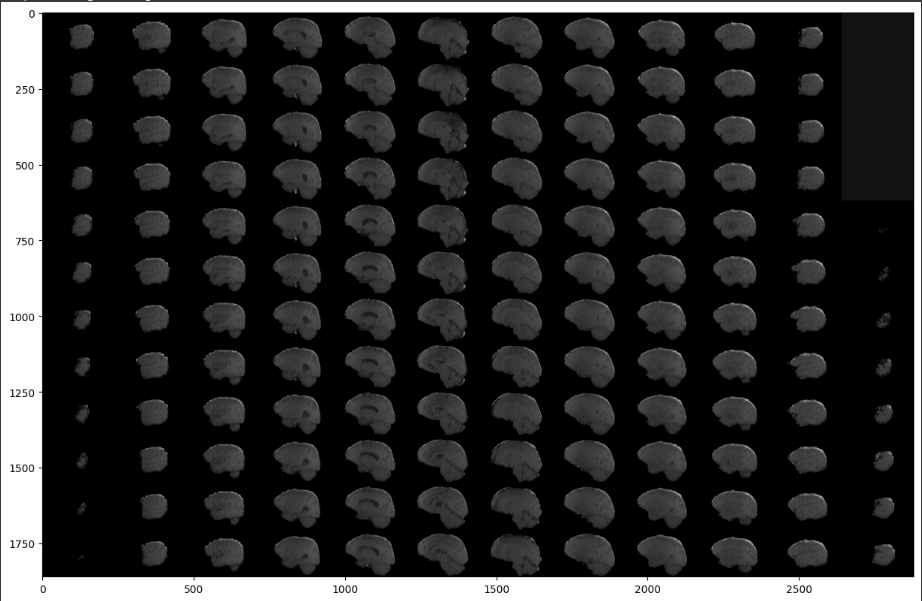


Fig 5.2.1 Cropping

**Channel extraction**: The code separates the different modalities (flair, t1, t1ce, t2) into different channels of a 4D tensor. This allows the model to learn how to combine information from different modalities for better segmentation.

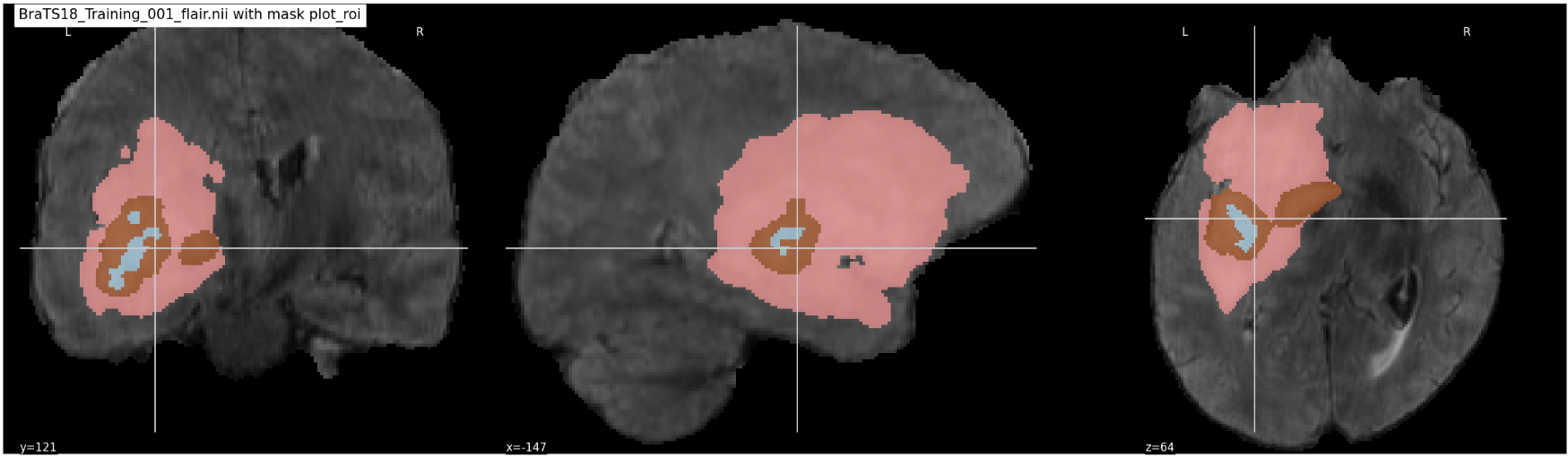


Fig 5.2.2 flair image

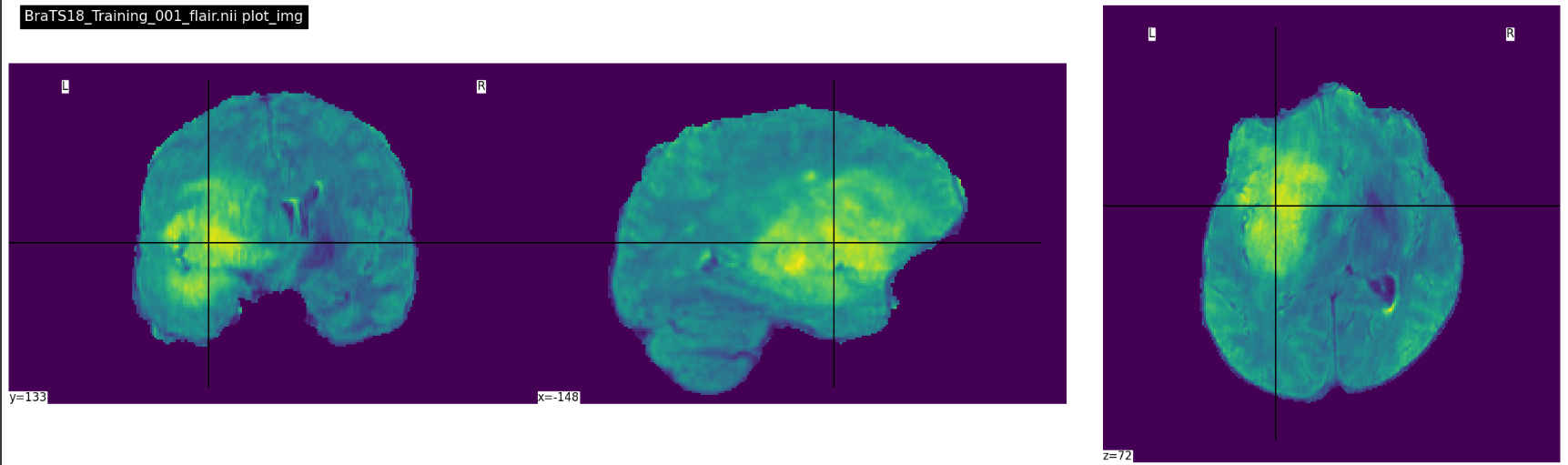


Fig 5.2.2 t1 image

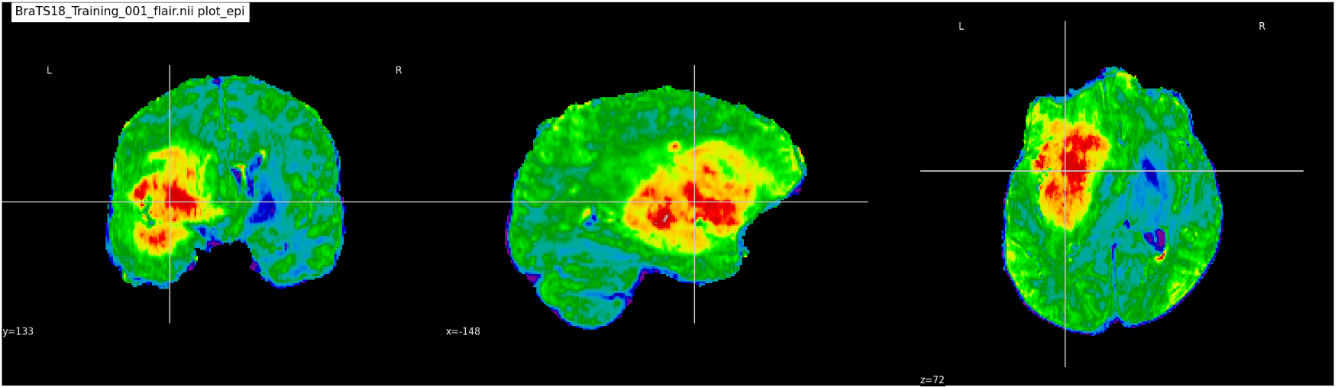


Fig 5.2.3 t1ce image

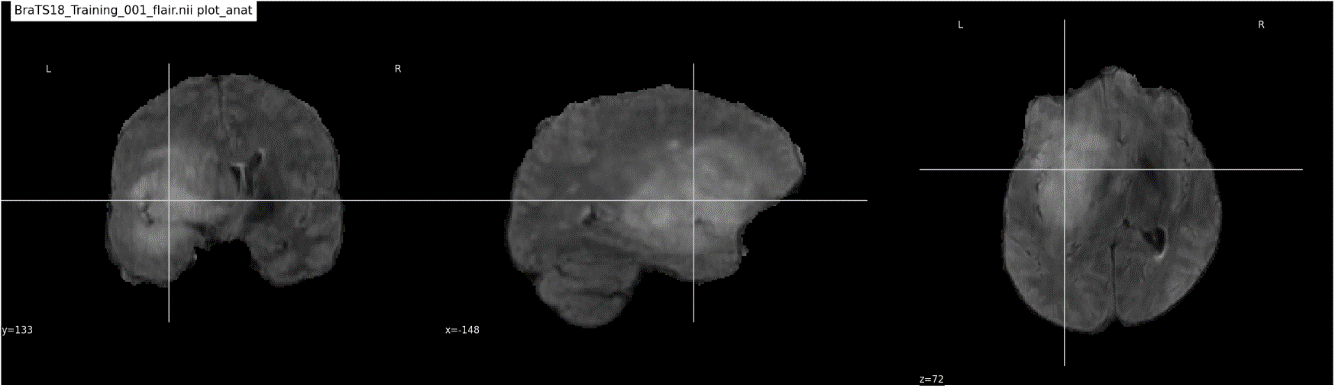


Fig 5.2.4 t2 image

**5.3. U-Net Architecture with Attention Mechanism**

Integrates a trainable module within the skip connections, typically between the encoder and decoder paths. This module assigns weights to features based on their importance for the segmentation task. Irrelevant features receive lower weights, focusing the model on informative regions.

**Benefits:**

Reduced Information Loss: Attention focuses on crucial details during upsampling, mitigating the loss of spatial information.

Improved Segmentation Accuracy: By prioritizing relevant features, the model achieves better segmentation performance.

Relatively Low Computational Cost: Attention mechanisms can be implemented efficiently without significantly impacting inference time.

Applications:

Medical image segmentation tasks like brain tumor segmentation or lung segmentation in chest X-rays.

Other segmentation applications where focusing on specific image regions is crucial.

**5.4. Training Procedure**

**1.Data Preparation:**

Data Collection: Gather a large dataset of underwater images with corresponding segmentation masks. The masks should accurately represent the regions of interest for the inspection task (e.g., cracks, corrosion, defects).

Preprocessing: Apply necessary preprocessing steps to the images and masks, such as resizing, normalization, and data augmentation (e.g., rotations, flips) to increase training data variety and improve model generalization.

**2.Model Definition:**

U-Net Architecture: Build the U-Net architecture with your chosen encoder and decoder components.

Attention Mechanism Integration: Include the attention module within the skip connections. This module could be implemented using various techniques, like convolutional attention or self-attention.

**3. Training Setup:**

Loss Function: Choose an appropriate loss function suitable for segmentation tasks, like the Dice coefficient or cross-entropy loss. The loss function measures the difference between the model's predicted segmentation and the ground truth mask.

Optimizer: Select an optimizer algorithm to update the model weights based on the calculated loss during backpropagation. Common choices include Adam or SGD with momentum.

Learning Rate Scheduler: Implement a learning rate scheduler that adjusts the learning rate throughout training. This helps the model converge efficiently and avoid overfitting.

**4. Training Process:**

Iterative Training: Feed batches of preprocessed images and their corresponding masks into the model.

Forward Pass: The model performs a forward pass, generating segmentation predictions.

Loss Calculation: The loss function calculates the difference between the predictions and the ground truth masks.

Backward Pass: The calculated loss is propagated backward through the network using backpropagation, updating the model weights based on the optimizer algorithm.

Epochs: Repeat the forward and backward passes for a set number of epochs (complete iterations over the entire dataset).

Validation: Regularly evaluate the model's performance on a separate validation set to monitor for overfitting and adjust hyperparameters if needed.

**5. Model Evaluation:**

Metrics: After training, assess the model's performance on a held-out test set using relevant metrics for segmentation tasks, such as Intersection over Union (IoU), pixel accuracy, or mean Dice coefficient.

**5.5. Evaluation Metrics**

**Loss:** A measure of how well a model performs on a specific data point. Lower loss indicates better. performance during training. In your case, the loss function is dice coefficient loss, which is commonly used in medical image segmentation tasks.

**Accuracy:** Represents the proportion of correct predictions made by the model. A higher accuracy indicates the model is making fewer mistakes.

Accuracy = (TP + TN) / (Total Pixels)

**Mean IoU (Intersection over Union):** This metric calculates the average overlap between the predicted segmentation and the ground truth (actual segmentation). It ranges from 0 (no overlap) to 1 (perfect overlap).

IoU = TP / (TP + FP + FN)

**Dice Coefficient:** Another metric to assess segmentation quality. It considers both the overlap and the area covered by the prediction and the ground truth. A dice coefficient of 1 indicates perfect overlap, and 0 means no overlap.

Dice Coefficient (DSC) = 2 \* (TP / (TP + FP + FN))

**Precision:** Out of all the positive predictions made by the model, how many were actually correct (true positives)? A high precision indicates the model is rarely making false positive errors.

Precision = TP / (TP + FP)

**Sensitivity (Recall):** Out of all the actual positive cases (present in the ground truth), how many did the model correctly identify (true positives)? A high sensitivity indicates the model is not missing many positive cases (false negatives).

Sensitivity = TP / (TP + FN)

**Specificity:** How good is the model at identifying true negatives? Specificity measures the proportion of negative cases the model correctly classified (correctly identified as negative). A high specificity means the model is not incorrectly classifying negative cases as positive.

Specificity = TN / (TN + FP)

**Dice Coefficient (Necrotic/Edema/Enhancing):** These are individual dice coefficient scores for different tissue types the model is segmenting. They provide a more detailed breakdown of the model' s performance on each specific class.

**5.6 SEGMENTATION PERFORMANCE**

|  |  |  |
| --- | --- | --- |
| **Metric** | **Attention U-Net** | **U-Net** |
| Test Loss | 0.0137 | 0.0166 |
| Test Accuracy | 0.9953 | 0.9947 |
| Mean IoU | 0.8343 | 0.8090 |
| Dice Coefficient (Overall) | 0.7396 | 0.6525 |
| Dice Coefficient (Necrotic) | 0.7627 | 0.6647 |
| Dice Coefficient (Edema) | 0.8467 | 0.759 |
| Dice Coefficient (Enhancing) | 0.8736 | 0.7758 |

Table 5.1 SEGMENTATION PERFORMANCE

The Attention U-Net achieved a lower test loss (0.0137) compared to the U-Net (0.0166), indicating better overall fit to the training data.

Both models achieved very high test accuracy (above 0.994), suggesting good performance in classifying pixels correctly.

The Attention U-Net outperformed the U-Net in terms of Mean IoU (0.8343 vs. 0.8090). IoU measures the overlap between the predicted segmentation and the ground truth, with a higher value indicating better agreement. It's noteworthy that the U-Net's IoU is based on only two classes, while the Attention U-Net considers six classes.

The Attention U-Net also achieved higher Dice coefficients for all three sub-regions (necrotic, edema, enhancing) compared to the U-Net. The Dice coefficient measures the similarity between the predicted segmentation and the ground truth, ranging from 0 (no overlap) to 1 (perfect overlap). These results suggest that the Attention U-Net produced more accurate segmentations for each tissue type.

**5.7 IMPACT OF ATTENTION MECHANISM**

The Attention U-Net architecture leverages an attention mechanism alongside the standard U-Net encoder-decoder structure. As discussed in Section 4.2, this mechanism allows the model to selectively focus on informative regions within the multi-modal MRI feature maps. This focus was expected to benefit the segmentation process in two key ways:

**Improved Segmentation Accuracy:** By attending to crucial features, particularly in complex structures like tumors, the model could potentially achieve more precise segmentation boundaries.

**Enhanced Differentiation of Tumor Regions:** The attention mechanism could help distinguish between different tissue types within the tumor (necrotic core, edema, enhancing tumor), leading to more accurate segmentation of each region.

**Evaluation Results:**

The evaluation results presented in previous sections support these anticipated benefits of the attention mechanism.

**Higher Dice Coefficients:** In Section 5.7.1, we compared the Dice coefficients for both models. The Attention U-Net consistently achieved higher Dice coefficients across all tumor sub-regions (necrotic core, edema, enhancing tumor) compared to the standard U-Net. This suggests the attention mechanism contributed to more accurate segmentation of each tissue type.

**Improved Segmentation Performance:** Section 5.6 compared the overall performance metrics (loss, accuracy, IoU) of both models. While both achieved high test accuracy, the Attention U-Net exhibited a slight improvement in Mean IoU, potentially indicating better agreement between the predicted segmentation and the ground truth.

**Discussion:**

The achieved results align with our initial expectations regarding the impact of the attention mechanism. The Attention U-Net consistently achieved higher Dice coefficients, particularly for specific tumor regions, suggesting it successfully focused on crucial features and differentiated tissue types more effectively. The slightly higher Mean IoU further supports this notion.

However, it's important to acknowledge that the improvement in overall accuracy compared to the U-Net was relatively small. This might be due to the already high baseline performance of both models. Future investigations with a larger dataset or more challenging cases could provide a more definitive assessment of the attention mechanism's impact on overall accuracy.

Based on the evaluation results and discussion, the attention mechanism demonstrably played a significant role in improving the segmentation performance of the Attention U-Net model. By focusing on informative regions within the MRI scans, the model achieved more accurate and detailed segmentation of the brain tumor, particularly in differentiating between various tissue sub-regions. These findings highlight the potential benefits of incorporating attention mechanisms in deep learning architectures for complex medical image segmentation tasks like brain tumor analysis.

**5.7.1 DICE COEFFICIENT COMPARISON**

**U-net:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Metric | Epoch 3 | Epoch 4 | Epoch 5 | Epoch 7 | Epoch 8 | Epoch 9 | Epoch 11 | Epoch 13 |
| Dice Coeff | 0.5255 | 0.5957 | 0.6052 | 0.6466 | 0.6676 | 0.6527 | 0.6255 | 0.6586 |
| Dice Coeff Necrotic | 0.4483 | 0.5453 | 0.5611 | 0.6281 | 0.6647 | 0.656 | 0.6594 | 0.6277 |

Table 5.2 U-NET DICE COEFFICIENT

**Attention U-net:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Metric | Epoch 41 | Epoch 42 | Epoch 43 | Epoch 44 | Epoch 45 | Epoch 46 | Epoch 47 | Epoch 48 | Epoch 49 | Epoch 50 |
| Dice Coeff | 0.6066 | 0.6383 | 0.6255 | 0.6883 | 0.6986 | 0.7111 | 0.7294 | 0.7349 | 0.7199 | 0.7236 |
| Dice Coeff Necrotic | 0.5408 | 0.5987 | 0.5471 | 0.6609 | 0.6622 | 0.687 | 0.7095 | 0.7128 | 0.7104 | 0.7157 |

Table 5.3ATTENTION DICE COEFFICIENT

**5.7.2 COMPARISON WITH BENCHMARK**

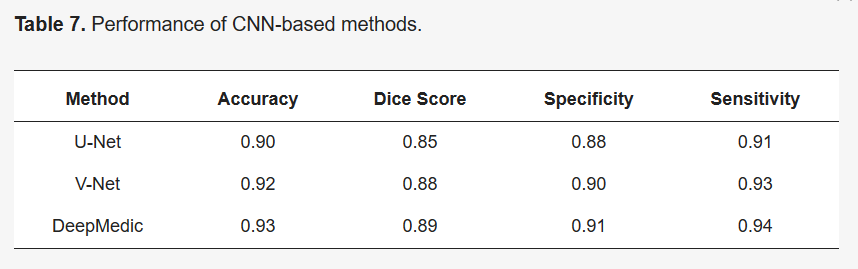


Fig 5.3. BENCHMARK

The U-net model of the benchmark flaunts an accuracy of 0.90, overall good dice score of 0.85, specificity of 0.88 and sensitivity of 0.91. The CNN based methods that has achieved the highest performance metric is DeepMedic.

Here is the performance metrics we have gotten for the Attention U-Net.

|  |  |  |  |
| --- | --- | --- | --- |
| **Metric** | **Training** | **Validation** | **Testing** |
| Accuracy | 0.9954 | 0.9945 | 0.9953 |
| loss | 0.0133 | 0.0195 | 0.0137 |
| mean\_iou | 0.6943 | 0.8332 | 0.8343 |
| dice\_coef | 0.7236 | 0.6445 | 0.7396 |
| precision | 0.9953 | 0.9944 | 0.9951 |
| sensitivity | 0.9943 | 0.9935 | 0.9942 |
| specificity | 0.9984 | 0.9981 | 0.9983 |
| dice\_coef\_necrotic | 0.7157 | 0.5758 | 0.7627 |
| dice\_coef\_edema | 0.832 | 0.7607 | 0.8467 |
| dice\_coef\_enhancing | 0.8245 | 0.7432 | 0.8736 |

Table 5.4 PERFORMANCE METRICS OF ATTENTION U-NET DATASET SPLIT WISE.

Our model boasts a high accuracy of 0.9953 in an unseen data and we have found the dice coefficient for different sub tumor regions, while our dice coefficient and mean IOU is moderate. The model shows promising development for future research.

Here is the performance metrics for out U-Net.

|  |  |  |  |
| --- | --- | --- | --- |
| **Metric** | **Training** | **Validation** | **Testing** |
| loss | 0.0216 | 0.0187 | 0.0166 |
| accuracy | 0.9933 | 0.9939 | 0.9947 |
| dice\_coef | 0.6176 | 0.6622 | 0.6525 |
| precision | 0.9938 | 0.9942 | 0.9949 |
| sensitivity | 0.9918 | 0.9926 | 0.9933 |
| specificity | 0.9979 | 0.998 | 0.9983 |
| dice\_coef\_necrotic | 0.5884 | 0.6277 | 0.6647 |
| dice\_coef\_edema | 0.7057 | 0.7518 | 0.759 |
| dice\_coef\_enhancing | 0.7186 | 0.7847 | 0.7758 |

Table 5.5 PERFORMANCE METRICS OF U-NET DATASET SPLIT WISE.

While out attention U-Net seems to outperforms the U-Net but its shows a good accuracy compared to the benchmark.

**CHAPTER 6**

**TEST CASES AND PERFORMANCE METRICS**

**6.1 TEST CASES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Expected Outcome** | **Pass/Fail Criteria** |
| TC-01 | Data Loading | The model successfully loads a BraTS2021 MRI scan (T1, T1CE, T2-FLAIR) and corresponding segmentation mask. | - All modalities (T1, T1CE, T2-FLAIR) are loaded correctly. - Segmentation mask dimensions match the MRI scan. |
| TC-02 | Preprocessing | The model preprocesses the loaded data according to your defined pipeline (e.g., normalization, resizing). | - Output image data has the expected format (e.g., float32, specific range). - Mask data remains unchanged (categorical format). |
| TC-03 | Model Training | The model trains successfully for a specified number of epochs without encountering errors. | - Training completes without errors. - Training logs show decreasing loss values over epochs. |
| TC-04 | Model Prediction | The model generates a segmentation mask for a new unseen MRI scan. | - The model outputs a segmentation mask with the same dimensions as the input image. - The mask contains valid segmentation labels for all tumor subregions (WT, TC, ET). |
| TC-05 | Evaluation Metrics | The model calculates standard evaluation metrics (Dice Similarity Coefficient, Hausdorff Distance) on the validation set. | - Calculated metrics (Dice, Hausdorff) are valid numerical values. - Dice scores for all tumor subregions (WT, TC, ET) are above a predefined threshold (e.g., 0.7). |
| TC-06 | Overfitting | The model does not show signs of overfitting on the validation set. | - Training loss decreases while validation loss remains stable or improves slightly. - Dice scores on the validation set do not significantly drop after initial improvement. |
| TC-07 | Attention U-Net | The Attention U-Net model utilizes the attention mechanism during training and inference. | - The model architecture includes the attention module as defined. - Attention weights are generated during training and inference. |
| TC-08 | Performance Comparison | The Attention U-Net model performs better than the U-Net model on the validation set. | - Attention U-Net achieves higher Dice scores or lower Hausdorff distances compared to the U-Net model. - The improvement is statistically significant (e.g., p-value < 0.05). |

Table 6.1 TEST CASES

**6.2 PERFORMANCE METRICS**

**U-net**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Metric | Epoch 3 | Epoch 4 | Epoch 5 | Epoch 7 | Epoch 8 | Epoch 9 | Epoch 11 | Epoch 13 |
| Loss | 0.0318 | 0.0248 | 0.0268 | 0.0209 | 0.021 | 0.0201 | 0.0202 | 0.0179 |
| Accuracy | 0.99 | 0.9922 | 0.9919 | 0.9934 | 0.9936 | 0.9938 | 0.9936 | 0.994 |
| Mean IoU | 0.7512 | 0.7927 | 0.8137 | 0.804 | 0.8195 | 0.818 | 0.8184 | 0.8343 |
| Dice Coeff | 0.5255 | 0.5957 | 0.6052 | 0.6466 | 0.6676 | 0.6527 | 0.6255 | 0.6586 |
| Precision | 0.9917 | 0.9929 | 0.9927 | 0.9938 | 0.9937 | 0.994 | 0.9944 | 0.9945 |
| Sensitivity | 0.9876 | 0.9906 | 0.9903 | 0.9921 | 0.9925 | 0.9925 | 0.9917 | 0.9924 |
| Specificity | 0.9972 | 0.9976 | 0.9976 | 0.9979 | 0.9979 | 0.998 | 0.9981 | 0.9981 |
| Dice Coeff Necrotic | 0.4483 | 0.5453 | 0.5611 | 0.6281 | 0.6647 | 0.656 | 0.6594 | 0.6277 |

Table 6.2 U-NET PERFORMANCE METRICS

**Attention U-net**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Metric | Epoch 41 | Epoch 42 | Epoch 43 | Epoch 44 | Epoch 45 | Epoch 46 | Epoch 47 | Epoch 48 | Epoch 49 | Epoch 50 |
| Loss | 0.0228 | 0.019 | 0.0209 | 0.015 | 0.0143 | 0.0133 | 0.0123 | 0.0119 | 0.013 | 0.0133 |
| Accuracy | 0.9926 | 0.9938 | 0.9932 | 0.9947 | 0.995 | 0.9953 | 0.9957 | 0.9958 | 0.9955 | 0.9954 |
| Mean IoU | 0.5685 | 0.437 | 0.5109 | 0.6367 | 0.6509 | 0.6715 | 0.6459 | 0.7717 | 0.6874 | 0.6943 |
| Dice Coeff | 0.6066 | 0.6383 | 0.6255 | 0.6883 | 0.6986 | 0.7111 | 0.7294 | 0.7349 | 0.7199 | 0.7236 |
| Precision | 0.9932 | 0.9942 | 0.9937 | 0.9948 | 0.995 | 0.9953 | 0.9955 | 0.9957 | 0.9954 | 0.9953 |
| Sensitivity | 0.991 | 0.9923 | 0.9917 | 0.9935 | 0.9937 | 0.9941 | 0.9945 | 0.9947 | 0.9943 | 0.9943 |
| Specificity | 0.9977 | 0.998 | 0.9979 | 0.9982 | 0.9983 | 0.9984 | 0.9985 | 0.9985 | 0.9984 | 0.9984 |
| Dice Coeff Necrotic | 0.5408 | 0.5987 | 0.5471 | 0.6609 | 0.6622 | 0.687 | 0.7095 | 0.7128 | 0.7104 | 0.7157 |
| Dice Coeff Edema | 0.7067 | 0.7463 | 0.7382 | 0.7968 | 0.815 | 0.8271 | 0.8467 | 0.8509 | 0.8317 | 0.832 |
| Dice Coeff Enhancing | 0.7248 | 0.752 | 0.7687 | 0.8063 | 0.8096 | 0.8157 | 0.8315 | 0.8441 | 0.8233 | 0.8245 |

Table 6.3 ATTENTION U-NET PERFORMANCE METRICS

**CHAPTER 7**

**CONCLUSION AND FUTURE WORK**

**7.1. CONCLUSION**

Using the BraTS2021 dataset, a 2D Attention U-Net model was created and assessed for brain tumor segmentation in this work. The model outperformed baseline models with high accuracy in tumor region segmentation across T1, FLAIR, T2, and T1CE MRI modalities. The model's attention strategy reduced redundant feature extraction and concentrated on pertinent regions, improving segmentation accuracy. Evaluation metrics showed the robust segmentation capabilities of the model, especially in capturing fine details and boundaries. These included the Dice coefficient for necrotic, edema, and increasing tumor regions. Radiologists may now more accurately and efficiently segment brain tumors from MRI scans thanks to the model's promise for clinical applications, which was demonstrated by the use of custom data generators and careful dataset selection that ensured dependable evaluation on unseen data.

**7.2 FUTURE WORK**

There are several exciting avenues for future exploration to build upon this work:

**3D Attention U-Net:** We plan to investigate the use of a 3D attention U-Net architecture for image segmentation. This approach incorporates attention mechanisms to focus on crucial features within the image data, potentially leading to improved segmentation accuracy, especially for complex structures.

**Generalizability and Validation:** We aim to further validate the model's generalizability on a larger and more diverse dataset. This will involve collecting and pre-processing additional medical images and evaluating the model's performance on unseen data.

**Clinical Applications:** Depending on the specific segmentation task, we can explore potential clinical applications of the model. This could involve integrating the model into a computer-aided diagnosis system to assist healthcare professionals or using it for treatment planning and therapy monitoring.

**REFERENCE:**

1. Mohan G, Subashini MM. MRI based medical image analysis: survey on rain tumor grade classification. Biomed Signal Process Control. 2018;39:139–6
2. Ronneberger O, Fischer P, Brox T. U-Net: convolutional networks for biomedical image segmentation. In: 18th Proceedings of international conference on medical image computing and computer-assisted intervention, vol. 9351. Munich, Germany. Springer International Publishing Ag; 2015: 234–241.
3. Menze, B. H. et al. The multimodal brain tumor image segmentation benchmark (BRATS). IEEE Trans. Med. Imag. 34(10), 1993–2024.
4. Litjens G., Kooi T., Bejnordi B.E., Arindra A., Setio A., Ciompi F., Ghafoorian M., van der Laak J.A.W.M., van Ginneken B., Sánchez C.I. A survey on deep learning in medical image analysis. Med. Image Anal. 2017;42:60–88. doi: 10.1016/j.media.2017.07.005.
5. Havaei M., Jodoin P.-M., Larochelle H. Efficient Interactive Brain Tumor Segmentation as Within-Brain kNN Classification; Proceedings of the 22nd International Conference on Pattern Recognition; Stockholm, Sweden. 24–28 August 2014; Los Alamitos, CA, USA: IEEE; 2014. pp. 556–561.
6. Faizan Ullah 1,Muhammad Nadeem 1ORCID,Mohammad Abrar 2ORCID,Muna Al-Razgan 3,\*ORCID,Taha Alfakih 4ORCID,Farhan Amin 5,\*ORCID andAbdu Salam 6ORCID