Interactive Brain Tumor Segmentation Leveraging Localized Focus and Preexisting Masks

Ali Sedaghi Arghoun

Artifical Intelligence & Robotics Department

School of Computer Engineering

Iran University of Science and Technology

Submitted in conformity with the requirements for the
Degree of Bachelor of Science
in Computer Engineering

Supervisor Dr. MohammadReza Mohammadi

March, 2023

Abstract

Accurately segmenting 3D images of the human brain is crucial for medical applications such as tumor diagnosis and treatment planning. Deep neural networks have emerged as the most advanced automatic segmentation method, but they may require modifications for clinical use. Interactive segmentation allows for greater accuracy by incorporating user interactions, but current methods are not suitable for low-power systems and may require significant user effort. This study utilizes a novel method that performs a coarse segmentation on a low-resolution section of the target region, followed by a local refinement to restore lost resolution. To optimize efficiency, morphological analysis is used to modify only areas that need updating while preserving previously obtained results for other regions. Experimental results demonstrate that this method achieves more accurate results with less user interaction and requires less computing power and time compared to other methods. Additionally, this method shows acceptable generalization in tasks not encountered during the training phase.

Keywords: Deep Learning, Computer Vision, Interactive Image Segmentation, Brain Tumor Segmentation (BraTS), 3D Medical Images

Statement of Originality

I, Ali Sedaghi Arghoun, student ID 97521378, studying in the field of Computer Engineering at the Bachelor of Science level, hereby certify that all the results of this thesis/dissertation are my own work without any alteration, and all copied content from others' works has been fully cited with complete source information. In case of any proven violation of the above, the university will deal with me according to the applicable rules and regulations (including the Law on the Protection of Authors' and Composers' Rights, the Law on Translation and Reproduction of Books, Periodicals, and Audio Works, educational, research, and disciplinary regulations, etc.), and I waive any right to object regarding the enforcement of acquired rights, determination of violations, and penalties. Furthermore, I bear full responsibility for any response to individuals, whether natural or legal persons, and competent authorities, and the university will not be held responsible in this regard.

Statement of Availability

I hereby acknowledge the availability of any part of this thesis/dissertation for viewing, photocopying or incorporation into future studies, providing that full reference is given to the origins of any information contained herein. I further give permission for a copy of this work to be deposited to any repository authorized for use by Iran University of Science and Technology. I acknowledge that Iran University of Science and Technology may make the title and a summary of this thesis/dissertation freely available.

Signed: Ali Sedaghi Arghoun

Date: March, 2023

Acknowledgements

I feel obliged to thank all those who have been by my side during this period. First, I am grateful to my family, especially my dear father and mother, who have always supported me throughout my life.

I also thank the professors of the Faculty of Computer Engineering, especially Dr. Mohammadi, who not only helped me complete this project but whose guidance and advice have been invaluable throughout my studies. I wish health and success for him and his respected family.

Furthermore, I express my gratitude to Dr. Mozayani, who kindly agreed to review this project and generously dedicated his valuable time to me.

Table of Contents

1 Introduction **1**

1.1 Introduction 1

2 Literature Review **4**

2.1 Image Segmentation 4

2.1.1 Semantic Segmentation 5

2.1.2 Instance Segmentation 5

2.1.3 Panoptic Segmentation 5

2.2 Interactive Image Segmentation 6

2.3 Interactive Segmentation Methods 7

2.3.1 Classification Based on the Type of User Interaction 7

Seed-Based Interaction 8

ROI-Based Interaction 8

Active and Passive Interaction 9

2.3.2 Classification Based on Methodology 10

Contour-based Methods 11

GraphCut-based Methods 11

Random Walk-based Methods 12

Region Growing and Merging-based Methods 13

Deep Learning and Hybrid Methods 13

2.4 Evaluation Metrics 14

2.4.1 Error Rate (ERR) 15

2.4.2 True Positive Rate (TPR) 15

2.4.3 Precision 15

2.4.4 F1-Score 16

2.4.5 Intersection over Union (IoU) 16

2.4.6 Dice Similarity Coefficient (DSC) 17

2.4.7 Variation of Information (VI) 18

2.4.8 Number of Clicks (NoC) 18
2.4.9 Number of Failures (NoF) 18
2.4.10 Time per Click (TPC) 18
2.4.11 Mean IoU (mIoU) and Mean Dice (mDice) 19
2.5 The BraTS Dataset 19
2.5.1 Introduction 19
2.5.2 Features of BraTS 20
2.5.3 Challenges of Working with BraTS 22
2.5.4 Applications 23
2.6 Backpropagating Refinement Scheme (BRS) 24
2.6.1 Introduction 24
2.6.2 Algorithm 24
2.6.3 Network Architecture 25
2.6.4 Training Phase 26
2.6.5 Inference Phase 26
2.6.6 Refinement 26
2.6.7 Conclusion 27

3 Methodology **28**
3.1 Introduction 28
3.2 Problems of Previous Methods 29
3.2.1 Poor Efficiency on Low-Computing-Power Devices 29
3.2.2 Lack of Use of Existing Pre-Existing Masks 29
3.3 Features of the Proposed Algorithm 30
3.4 Algorithm at a Glance 31
3.5 Algorithm Details 32
3.5.1 Target Crop 33
3.5.2 Coarse Segmentation 33
3.5.3 Focus Crop 35
3.5.4 Local Refinement 36
3.5.5 Progressive Merge 37
3.6 Loss Function 38
3.7 Combining Different MRI Modalities 38

4 Experiments and Results **40**
4.1 Preparing the BraTS Dataset 40
4.2 Distribution of the Dataset into Training, Validation, and Testing Sets 40

4.3 Training Environment 41
4.4 Defining and Running Experiments 41
4.5 General Notes on Experimental Results 43
4.6 Exp 1: HRNet18s-FLAIR-WT-Pretrained 44
4.7 Exp 2: HRNet32-FLAIR-WT-Pretrained 46
4.8 Exp 3: SegFormerB3-FLAIR-WT-Pretrained 48
4.9 Exp 4: SegFormerB3-Mix-WT and Modality Fusion 50
4.10 Exp 5: Evaluation on an Unseen Task During Training 52
4.11 Exp 6: Evaluation on Unseen Modality and Task 54
4.12 Comparison with Related Work Results 57
4.12.1 The *Enhanced U-Net* Paper 57
4.12.2 The *DeepIGeoS* Paper 58
4.12.3 The *MIDeepSeg* Paper 59
4.12.4 Traditional and Non-Deep Methods 60

5 Conclusion **61**
5.1 Conclusion 61
5.2 Suggestions and Future Work 62
5.2.1 Adapting FocalClick for 3D Input 62
5.2.2 Using BRS in FocalClick 62
5.2.3 Using Exponential Geodesic Distance Transform 62
5.2.4 Other Suggestions 63

Appendices **64**

References **65**

List of Tables

2.1 Summary of BraTS data distribution over the years 20
2.2 Brightness levels of different brain parts in each modality 21

4.1 Distribution of Training, Validation, and Testing Sets 41
4.2 NoC and NoF metrics in Experiment 1 44
4.3 mIoU and mDice metrics in Experiment 1 44
4.4 NoC and NoF metrics in Experiment 2 46
4.5 mIoU and mDice metrics in Experiment 2 46
4.6 NoC and NoF metrics in Experiment 3 48
4.7 mIoU and mDice metrics in Experiment 3 48
4.8 NoC and NoF metrics in Experiment 4 50
4.9 mIoU and mDice metrics in Experiment 4 50
4.10 NoC and NoF metrics in Experiment 5 52
4.11 mIoU and mDice metrics in Experiment 5 52
4.12 NoC and NoF metrics in Experiment 6 54
4.13 mIoU and mDice metrics in Experiment 6 54
4.14 Results of *Enhanced U-Net* Paper 58
4.15 Results of *DeepIGeoS* Paper 58
4.16 Dataset distribution in the *MIDeepSeg* paper 59
4.17 Results of the *MIDeepSeg* paper 59
4.18 Results of traditional and non-deep methods 60

List of Figures

2.1 An example of medical image segmentation 5
2.2 Segmentation types 6
2.3 Types of seed-based and ROI-based interaction 7
2.4 Examples of different user interaction methods 9
2.5 Classification of interactive segmentation methods 11
2.6 Mathematical representation of IoU 16
2.7 Example of S and G segmentations 16
2.8 Example of intersection and union areas 17
2.9 Mathematical representation of Dice coefficient 17
2.10 Comparison of three modalities in medical images 21
2.11 An example of BraTS sample in all four modalities 21
2.12 Regions corresponding to different classes in BraTS 22
2.13 Overview of the BRS algorithm 25
2.14 Proposed architecture based on DenseNet 25
2.15 Example of points generated by clustering during training 26
2.16 Mathematical representation of the BRS relation 27

3.1 Types of clicks 28
3.2 Unintended changes when generating new output 30
3.3 A general overview of the *FocalClick* algorithm 32
3.4 Architecture of the Coarse Segmentation part 33
3.5 Architecture of the SegFormer network 34
3.6 Architecture of the HRNet network 34
3.7 Architecture of the U-Net network 35
3.8 Architecture of the Focus Crop part 36
3.9 Architecture of the Local Refinement part 36
3.10 Architecture of the Progressive Merge part 37
3.11 Example of the image created in the modality combination mode . . 39

4.1 How to define variables in the training phase 42
4.2 How to define variables in the evaluation phase 42
4.3 Algorithm output at one of the training phase steps in Experiment 1 45
4.4 Algorithm output on one of the evaluation samples in Experiment 1 . 45
4.5 Detail recovery of one evaluation sample image in Experiment 1 . . . 45
4.6 Algorithm output at one of the training phase steps in Experiment 2 47
4.7 Algorithm output on one of the evaluation samples in Experiment 2 . 47
4.8 Detail recovery of one evaluation sample image in Experiment 2 . . . 47
4.9 Algorithm output at one of the training phase steps in Experiment 3 49
4.10 Algorithm output on one of the evaluation samples in Experiment 3 . 49
4.11 Detail recovery of one evaluation sample image in Experiment 3 . . . 49
4.12 Algorithm output at one of the training phase steps in Experiment 4 51
4.13 Algorithm output on one of the evaluation samples in Experiment 4 . 51
4.14 Detail recovery of one evaluation sample image in Experiment 4 . . . 51
4.15 Algorithm output at one of the training phase steps in Experiment 5 53
4.16 Algorithm output on one of the evaluation samples in Experiment 5 . 53
4.17 Detail recovery of one evaluation sample image in Experiment 5 . . . 54
4.18 Algorithm output at one of the training phase steps in Experiment 6 56
4.19 Algorithm output on one of the evaluation samples in Experiment 6 . 56
4.20 Detail recovery of one evaluation sample image in Experiment 6 . . . 56

5.1 Comparison of different transformations for user clicks 63

Chapter 1

Introduction

1.1 Introduction

In recent decades, the field of medical imaging has undergone significant transformation through the use of machine learning (ML) and deep learning (DL) algorithms for image segmentation (IS) of medical scans and the detection of brain tumors or other diseases. This field assists physicians in the early diagnosis of brain malignancies and improves disease prognosis. Adults are often affected by gliomas, which originate from glial cells and may subsequently invade surrounding healthy tissues [21].

Magnetic Resonance Imaging (MRI) modalities are commonly examined manually by radiologists to extract quantitative information and to delineate tumor boundaries and internal regions. However, this manual inspection—followed by three-dimensional (3D) segmentation—is a time-consuming, costly, error-prone process that requires highly trained specialists. Moreover, diagnostic outcomes may vary among experts. This challenge becomes even more critical when tumors vary in size, shape, and location.

Enhancements in disease diagnosis, treatment planning, monitoring, and clinical decision-making rely heavily on precise segmentation of brain tumors in medical images, which in turn requires accurate localization and size estimation. However, the inherent characteristics of brain tumors make accurate segmentation a difficult task [9]. Tumors can appear in any brain region, with diverse sizes and shapes. Fur-

thermore, tumor tissue may resemble healthy, vital brain tissue in texture and intensity, often showing low contrast differences. Consequently, distinguishing healthy tissue from tumor regions remains challenging [27].

Automated segmentation methods, combined with imaging technologies and deep learning, enable specialists to work more efficiently and accurately. As a result, rapid and precise brain tumor segmentation can help physicians treat tumors safely—particularly during surgery—without compromising healthy brain regions, thereby increasing patient survival rates and improving quality of life. In the domain of automatic segmentation, various convolutional neural network (CNN) architectures are currently applied to 3D medical and MRI images. Previously, the use of deep networks was challenging due to the lack of sufficient medical data and the high computational demands. The release of datasets such as the Brain Tumor Segmentation (BraTS) dataset—which contains labeled 3D MRI scans—has facilitated progress in this field. Additionally, the availability of more powerful graphics processing units (GPUs) has enabled faster data processing. These advancements have increased research interest in developing methods to support physicians in this area.

Nevertheless, because tumor cells can develop anywhere in brain tissue and vary in size, appearance, and shape, automated segmentation of brain tumors and their subregions and boundaries remains difficult [17]. Moreover, in medical applications—where human lives and health are at stake—physician involvement in the automated segmentation process is essential. This need introduces the concept of semi-automatic segmentation.

Interactive image segmentation (IIS) is a method that incorporates user-provided guidance regarding the target object, known as the foreground (FG), as well as cues about the background (BG), to more accurately and efficiently segment the region of interest (ROI). In such methods, user interaction can continue until the desired outcome is achieved. One goal of interactive segmentation is to minimize user effort while maximizing accuracy. The use of deep learning and CNN architectures—alongside

innovative approaches for injecting user knowledge into the network—appears to be one of the most promising strategies to achieve this goal.

The aim of this research is to propose a deep learning–based method for interactive segmentation of brain tumors across various MRI modalities, delivering high accuracy and speed while maintaining low computational cost. Furthermore, a key consideration is minimizing the user's required effort. This work also examines whether the proposed method, once trained on a single modality and task, can generalize effectively to other modalities and tasks.

The remainder of this thesis is structured as follows:

- Chapter 2 presents the literature review on interactive segmentation and discusses related work.
- Chapter 3 details the proposed algorithm, including the network architecture, loss function, evaluation metrics, and related components.
- Chapter 4 reports the results of various experiments along with their analysis, interpretation, and generated outputs.
- Chapter 5 concludes the study and provides suggestions for future work.

Chapter 2

Literature Review

2.1 Image Segmentation

Image segmentation is one of the fundamental tasks in computer vision and has applications in various fields. The main objective of image segmentation is to partition an image into similar regions. These similar regions share common properties such as spatial location, color, shape, texture, and motion (in the case of video segmentation) [28].

More precisely, image segmentation is a process in which each pixel of the image is assigned a label. These labels are assigned in such a way that pixels with the same label share similar and common characteristics, as mentioned above. Simulating the human perceptual system to divide an image into meaningful regions has been one of the core challenges in computer vision and has been extensively studied since the early days of the field.

The reason for such extensive research is the wide range of applications of image segmentation. For example, in the medical domain, image segmentation can be used to process Computed Tomography (CT) and Magnetic Resonance Imaging (MRI) scans to detect damaged areas and tumors, which can then assist in determining the severity of the disease and in planning surgery.

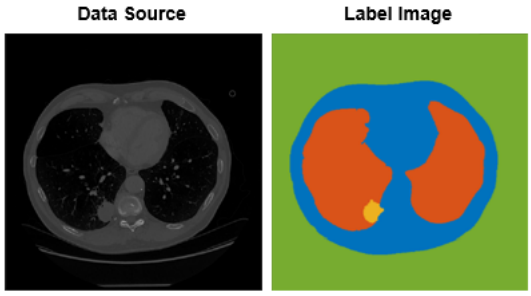


Figure 2.1: An example of medical image segmentation

2.1.1 Semantic Segmentation

In semantic segmentation, all pixels belonging to the same class are assigned a single fixed label [10]. For example, if an image contains three cats in a field, the pixels corresponding to the cats would be assigned label "1," while the pixels of the field area would be assigned label "0."

2.1.2 Instance Segmentation

In instance segmentation, all pixels belonging to each individual instance of the target object are assigned a distinct label [46]. The aim is to distinguish between different instances of the same object. For example, in the above image, the pixels for the first cat could be labeled "0," the second cat labeled "1," and the third cat labeled "2."

2.1.3 Panoptic Segmentation

Panoptic segmentation is a combination of the previous two approaches [19]. Similar to semantic segmentation, each pixel is assigned the label of its corresponding class. At the same time, similar to instance segmentation, different instances of the same class are also distinguished from one another.

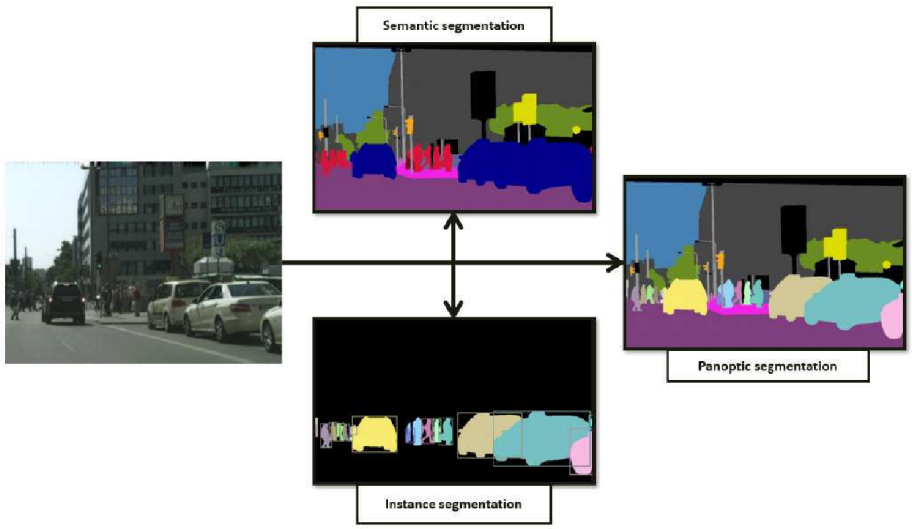


Figure 2.2: Different types of segmentation [8]

2.2 Interactive Image Segmentation

In addition to the segmentation approaches mentioned in the previous section, some research—although less extensive—has focused on interactive image segmentation methods. Interactive image segmentation, also referred to as supervised segmentation or semi-automatic segmentation, refers to the extraction of a region from an image, a target object, or the foreground (FG) from the background (BG) with the aid of prior knowledge provided through user interaction [31].

User interaction with the model can take various forms. For example, points or scribbles can be used to indicate the desired object or the background image (BG). A bounding box (BB) or a polygon can also be employed to define the regions of interest (ROI) and interact with the model, thereby assisting the automatic segmentation process. Such user interactions not only improve the results but also reduce segmentation execution time. Consequently, interactive segmentation typically achieves higher accuracy and speed compared to fully automatic segmentation [12]. In fact, in many computer vision applications, such as in the medical field, achieving accurate segmentation requires this user involvement.

2.3 Interactive Segmentation Methods

Interactive segmentation methods can be classified in various ways. In the following, we will discuss the details of two such classification approaches.

2.3.1 Classification Based on the Type of User Interaction

The way a user interacts with the model can be examined from two perspectives. From the first perspective, user interaction with the model can be based on either the Seed or the ROI approach [42]. From the second perspective, user interaction with the model can be either active or passive [3].

In the first perspective, the user can specify the background (BG) and foreground (FG) using a set of points, line segments, or disconnected lines. This form of user interaction is called seed-based interaction. Another way of interaction involves drawing approximate boundaries such as a bounding box, a polygon, or any other closed curve to separate the target object from the BG. These boundaries are drawn by the user to define the ROI, and therefore, this form of interaction is called ROI-based interaction.

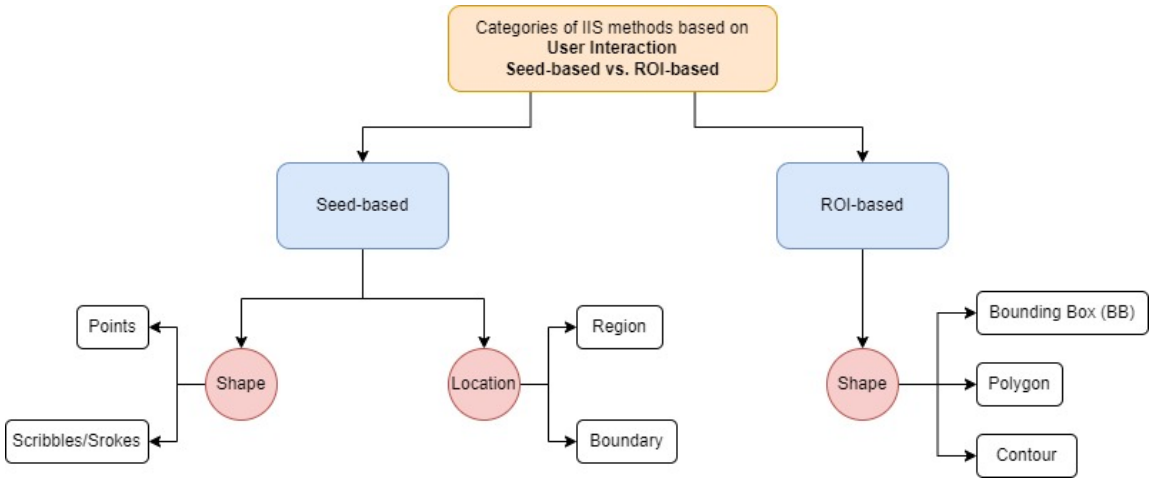


Figure 2.3: Types of seed-based and ROI-based interaction

Seed-Based Interaction

Early seed-based models required points located on the boundaries of the target object. These points are known by different names, two of which are seed points and anchor points. This method is also referred to as boundary seed-based interactive segmentation.

Another subgroup of seed-based methods relies on the region. In this case, there is no restriction on the placement of points by the user. However, the segmentation output becomes highly sensitive to the number of points, and different point sets may result in significantly different outputs. The seeds in this case can be either a set of discrete points or continuous strokes (scribbles).

It is also possible to create a hybrid approach by combining boundary seeds and region seeds [39]. In this hybrid method, the model first uses boundary seeds to roughly determine the object's contour and then applies region seeds to produce a refined final output.

ROI-Based Interaction

In this approach, prior knowledge about the FG and BG is obtained from the pixels inside and outside the ROI. The GrabCut method [35] is the most well-known and widely used interactive segmentation method, allowing the user to draw a bounding box to separate the FG from the BG.

The performance of ROI-based methods depends on how tightly the user's drawn curve encloses the FG. If the curve fails to tightly separate the FG, the model's performance will drop significantly. As a result, the concept of "tightness" has emerged for this category of methods.

In general, defining an ROI by drawing a curve around it is easier for the user, but the accuracy and performance of the model are highly dependent on the tightness of this curve. On the other hand, seed-based algorithms, given enough seeds, can

extract FGs with complex shapes. Overall, ROI-based methods usually require more user interactions compared to seed-based methods [15].

The main challenge, regardless of whether the method is seed-based or ROI-based, is to achieve high accuracy with minimal user effort. Recent works based on deep learning have shown good progress toward this goal.

A summary of the seed-based and ROI-based interaction types, along with their subcategories, is shown in Figure 2.4. Figure (a) shows the boundary seed case, Figure (b) shows region point seeds, Figure (c) shows region scribble seeds, Figure (d) shows ROI contour, Figure (e) shows ROI BB, and Figure (f) shows the ROI BB tight case.

Figure 2.4: Examples of different user interaction methods [31]

Active and Passive Interaction

From the second perspective, some interactive segmentation models allow active interaction and online assistance during the segmentation process. These methods fall into the active category. For example, the model may first output an initial segmentation and then identify regions with low confidence, prompting the user to refine them. This interaction could take the form of yes/no feedback on uncertain regions, or the model could generate multiple outputs and, through binary questions

(similar to the "20 Questions" game), process the user's responses to produce the final improved result [36]. In active methods, interaction continues until the user is satisfied with the result or the model reaches a predefined threshold.

In contrast, interaction can also occur passively or offline, which is referred to as passive interaction [3]. In fact, the majority of interactive segmentation methods are passive. In this case, whenever the user creates new seeds or ROIs, or edits existing ones, the model's output is updated accordingly.

2.3.2 Classification Based on Methodology

Interactive segmentation algorithms can be divided into two main categories based on methodology. The first category includes traditional and older approaches such as contour-based algorithms, graph-cut-based algorithms, random-walk-based algorithms, and region-based algorithms such as region growing and region merging. The second category consists of more recent approaches based on deep learning algorithms, where convolutional neural networks (CNNs) are widely used.

In some works, both types of approaches are combined to leverage the strengths of each. For example, deep learning can be used to generate an initial segmentation (without user interaction), and then the user interactions can be processed using a graph-cut method to refine the results [23]. While the deep-learning-based method initially produces a relatively good output, its high computational cost makes it impractical to re-run for each user interaction. Therefore, in the interaction stage, more efficient traditional algorithms are employed.

Since the main focus of this project is interactive segmentation using deep learning, the traditional, non-deep methods are reviewed briefly here, with short explanations of contour-based, graph-cut-based, random-walk-based, region-growing, and region-merging algorithms.

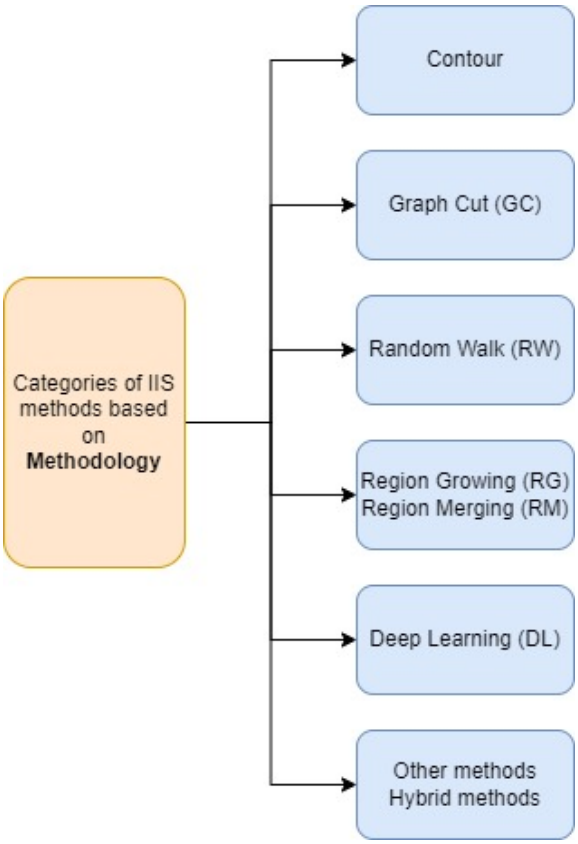


Figure 2.5: Classification of interactive segmentation methods based on methodology

Contour-based Methods

In contour-based methods, the key principle is to extract the object's contours using edge-related features and prior knowledge provided through user interaction. The object contours are extracted by computing the shortest path connecting the seed points. The shortest path is calculated using Dijkstra's algorithm or faster alternatives.

GraphCut-based Methods

GraphCut-based methods are among the most popular in the field of interactive segmentation. In these methods, an energy function is defined with the goal of minimizing it. This energy minimization can be approximated by solving a max-flow problem in a graph [2]. According to the max-flow min-cut theorem, it is sufficient to find the minimum cut in the graph.

The GrabCut algorithm [34] is the most well-known GraphCut-based interactive segmentation method. In GrabCut, the user first creates a bounding box around the target object. The algorithm then uses a Gaussian Mixture Model (GMM) to estimate the color distributions of the foreground (FG) and background (BG). Based on this estimation, a Markov Random Field (MRF) is constructed over the pixel labels, with an energy function that favors connected regions having the same label. Finally, a GraphCut-based optimization is performed to obtain the labels.

Random Walk-based Methods

In the original Random Walk model [32], an undirected graph is created in which the nodes represent the pixels of the input image, and each edge connects two pixels. The weight of each edge represents the probability of a random step between the two connected pixels. The goal of the algorithm is to find a set of probabilities (x), where each element (x_i) is the probability that a random walk will eventually remain at node i.

This goal can be formulated as minimizing the following function:

$$E(x) = x^T L x \tag{2.1}$$

where L is the combinatorial Laplacian matrix.

Let V_m be the set of vertices (pixels) chosen by the user as seeds, and V_u the set of unmarked vertices. These two sets are disjoint, and their union covers all pixels. Let f_j denote the vector of pixel labels, and partition the matrix L as:

$$L = \begin{bmatrix} L_m & B \\ B^T & L_u \end{bmatrix} \tag{2.2}$$

To minimize Eq. 2.1, it is sufficient to solve the following linear system:

$$L_u x_u = -B^T f \tag{2.3}$$

Region Growing and Merging-based Methods

The main idea in region growing and region merging algorithms is to start from the regions where the user has placed seeds and iteratively merge adjacent, similar regions. At each merging step, the principle of region homogeneity must be preserved [1]. The merging process is repeated until all regions are labeled according to the user's desired segmentation.

Region-merging methods are generally easy to understand but have many limitations and challenges. For example, the order of merges can lead to different results [1]. Additionally, accurate results often require sufficient user input to cover the main regions. This becomes challenging when parts of the foreground are very similar to the background. These methods also tend to perform poorly on images with shadows or low quality [29].

Deep Learning and Hybrid Methods

In recent years, CNNs have achieved significant success in many computer vision applications, including image segmentation. Many interactive segmentation methods based on deep architectures have been proposed, which outperform traditional models. Below, a few deep methods in this field are briefly introduced, while some of them are reviewed in more detail in the related work section.

The Deep Extreme Cut (DEXTR) [24] architecture uses a CNN-based model for interactive segmentation by incorporating extreme points—top, bottom, left, and right—of the target object. These points are provided alongside the RGB channels as input to the model. A ResNet-101 [14] network is used as the backbone in this model.

The Deep Interactive Object Selection method [44] converts positive clicks (on the target object) and negative clicks (outside the target object) provided by the user into separate Euclidean distance maps. These maps are then combined with the RGB channels of the input image to form an image-interaction pair. This pair is

fed into a Fully Convolutional Network (FCN) to generate an initial segmentation. Finally, a refinement stage using GraphCut optimization is applied to produce the final output.

The Deep GrabCut for Object Selection method [45] takes a bounding box from the user instead of clicks. This model is trained end-to-end and does not require a post-processing stage to refine the FCN outputs.

The SeedNet method [20] provides an interactive segmentation system that assists the user in segmenting an object using an automatic seed generation framework. The automatic seed generation in this system is performed via deep reinforcement learning (Deep RL).

With the rise of deep learning and the high-quality segmentation results that deep learning-based methods can achieve with just a few clicks, research in this area is very promising. However, a common problem with most deep learning-based methods—one that some works have tried to address—is that excellent results are obtained on the training set, while performance on unseen classes is poor.

Moreover, the requirement for large amounts of training data to train models with a vast number of parameters raises doubts about the practicality of using such methods in industrial applications. Nevertheless, with the increasing availability of datasets and the use of data augmentation techniques, deep learning-based methods have been successfully deployed in industry and smart factories.

2.4 Evaluation Metrics

For objective quantitative and observable evaluation of segmentation methods, including interactive segmentation methods, a comparison between the model's output and the human-annotated labels (ground truth, GT) is required. This comparison must be based on a metric capable of measuring the similarity between the GT and the model's output.

In this section, a summary of common metrics used for such evaluation is provided. In the following formulas, S represents the model's segmentation output, and G denotes the ground truth.

2.4.1 Error Rate (ERR)

The error rate indicates the ratio or percentage of pixels inside the ROI that are incorrectly labeled. It is defined as:

$$ERR = \frac{|S \cup G - G|}{\text{Number of pixels in ROI}} \tag{2.4}$$

2.4.2 True Positive Rate (TPR)

Also known as Recall, this metric represents the ratio of correct positive predictions by the model for a positive class. Recall is defined as:

$$R = \frac{|S \cap G|}{|G|} \tag{2.5}$$

The reason for using Recall instead of simple Accuracy is that, for example, if 98% of pixels in a brain scan are healthy, then a model that predicts all pixels as healthy will have 98% accuracy, which seems good. However, in medical applications, the remaining 2% of pixels containing a tumor are critical because detecting them can lead to treatment.

2.4.3 Precision

Precision indicates the ratio of correct positive predictions among all positive predictions made by the model:

$$P = \frac{|S \cap G|}{|S|} \tag{2.6}$$

2.4.4 F1-Score

The F1-Score is the harmonic mean of Recall and Precision, combining these two metrics as:

$$F = \frac{2 \times R \times P}{R + P} \tag{2.7}$$

2.4.5 Intersection over Union (IoU)

IoU is the ratio of correctly predicted foreground pixels (intersection) to the total pixels labeled as foreground in either S or G (union). It is calculated as:

$$IoU = \frac{|S \cap G|}{|S \cup G|} \tag{2.8}$$

Also known as the Jaccard Index or Overlap Coefficient, IoU is one of the most common metrics for evaluating segmentation models.

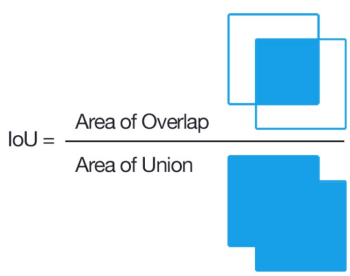


Figure 2.6: Mathematical representation of IoU

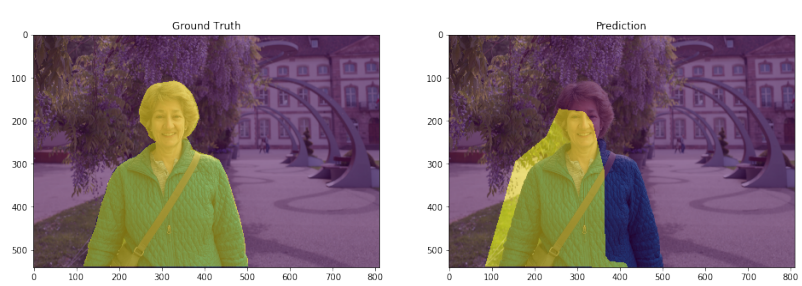


Figure 2.7: Example of S and G segmentations

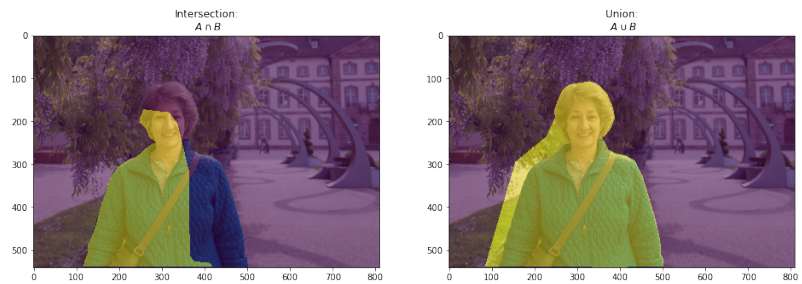


Figure 2.8: Example of intersection and union areas

2.4.6 Dice Similarity Coefficient (DSC)

This metric is similar to IoU but differs by using the sum of all foreground pixels in S and G in the denominator instead of the union, and the ratio is multiplied by 2:

$$DSC = 2 \times \frac{|S \cap G|}{|S| + |G|} \tag{2.9}$$

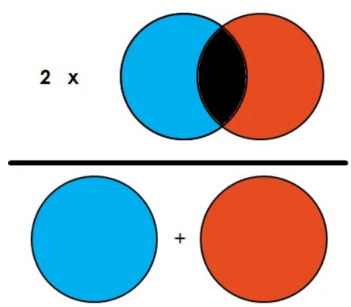


Figure 2.9: Mathematical representation of Dice coefficient

From its definition, the Dice coefficient is equivalent to the F1 score. Unlike IoU, Dice can be used as a loss function during network training because it is differentiable. The Dice loss function is defined as:

$$Dice\ Loss = 1 - DSC \tag{2.10}$$

2.4.7 Variation of Information (VI)

This metric measures the distance between two segmentations S and G using the average conditional entropy. It is based on Information Theory and is mostly applied in clustering problems rather than segmentation tasks [25].

2.4.8 Number of Clicks (NoC)

This metric expresses the average number of clicks a user needs to reach a desired IoU or Dice score. For example:

$$NoC(Dice = 80\%) = 7.87 \tag{2.11}$$

means that on average, 7.87 clicks are needed to achieve 80% Dice score.

2.4.9 Number of Failures (NoF)

Assuming the model has a maximum allowed number of clicks max_clicks from the user, NoF counts how many samples fail to reach the desired IoU or Dice within this limit. These are called failed samples. For example:

$$NoF_{max-clicks=20}(Dice = 85\%) = 851 \tag{2.12}$$

means 851 samples did not achieve 85% Dice even after 20 clicks.

2.4.10 Time per Click (TPC)

This metric measures the average time the model takes to produce a new output after receiving each user click. It corresponds to the model's response time per click, measured in seconds.

2.4.11 Mean IoU (mIoU) and Mean Dice (mDice)

These metrics represent the average IoU or Dice score that the model achieves after a specified number of clicks per sample. For example:

$$mDice_{C=20} = 62\% \tag{2.13}$$

means the model achieves an average Dice score of 62% after 20 clicks per sample.

2.5 The BraTS Dataset

2.5.1 Introduction

Brain tumors are among the most threatening diseases and are a significant cause of death worldwide. Early detection and accurate diagnosis of brain tumors are essential for effective treatment and improving survival rates. Medical imaging techniques such as MRI play a vital role in the diagnosis of brain tumors. MRI provides non-invasive and accurate imaging of the brain and offers high-resolution 3D images of the brain structure and any abnormalities. However, manual segmentation of brain tumors from MRI images is a challenging and time-consuming task. Therefore, automatic segmentation of brain tumors from MRI images is a crucial research area in medical imaging.

The Brain Tumor Segmentation dataset, abbreviated as BraTS [26], is a publicly available dataset for developing and evaluating automatic brain tumor segmentation algorithms. This dataset has been widely used for research and benchmarking in medical imaging, especially for developing deep learning-based brain tumor segmentation models. In this section, an overview of the BraTS dataset is provided, including its features, challenges, and potential applications.

2.5.2 Features of BraTS

The BraTS dataset is a publicly available collection of MRI brain tumor scans with manual annotations. The dataset is provided in the NIfTI format, which stands for Neuroimaging Informatics Technology Initiative. The 2020 version of this dataset includes 660 MRI scans of brain tumors, divided into training (369), validation (125), and testing (166) sets. The following table summarizes the distribution of training, validation, and testing sets across different years.

Table 2.1: Summary of BraTS data distribution over the years

Year	**Total Data**	**Training Data**	**Validation Data**	**Testing Data**
2012	50	35	NA	15
2013	60	35	NA	25
2014	238	200	NA	38
2015	253	200	NA	53
2016	391	200	NA	191
2017	477	285	46	146
2018	542	285	66	191
2019	626	335	125	166
2020	660	369	125	166
2021	2040	1251	219	577

The dataset includes 2D and 3D MRI images from multiple institutions, each having different scanner types and imaging protocols, providing a diverse collection of imaging data. It also includes annotations for each tumor subregion, such as grade size, location, and shape, which can be used for tumor segmentation and classification.

The dataset includes four imaging modalities for MRI images, which are:

- T2 Fluid Attenuated Inversion Recovery (FLAIR)
- Native T1-weighted (T1)

- T1-weighted contrast-enhanced (T1ce)
- T2-weighted (T2)

Figure 2.10 provides an example and intuitive comparison of the three modalities. Table 2.2 gives a more detailed comparison of these modalities.

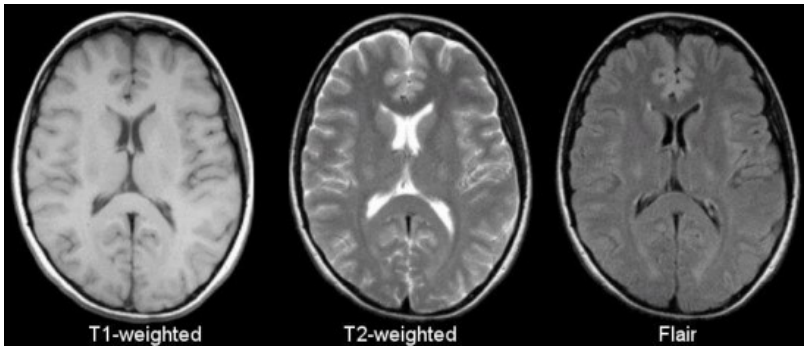


Figure 2.10: Comparison of three modalities in medical images [30]

Table 2.2: Brightness levels of different brain regions in each modality [30]

	T1	**T2**	**FLAIR**
Cerebrospinal fluid (CSF)	Dark	Bright	Dark
White Matter	Light	Dark Gray	Dark Gray
Cortex	Gray	Light Gray	Light Gray
Fat	Bright	Light	Light
Inflammation	Dark	Bright	Bright

Figure 2.11 shows an example from this dataset in 2D across all four MRI modalities. From left to right, the first image corresponds to the FLAIR modality, the second to T1, the third to T1ce, and the last to T2.

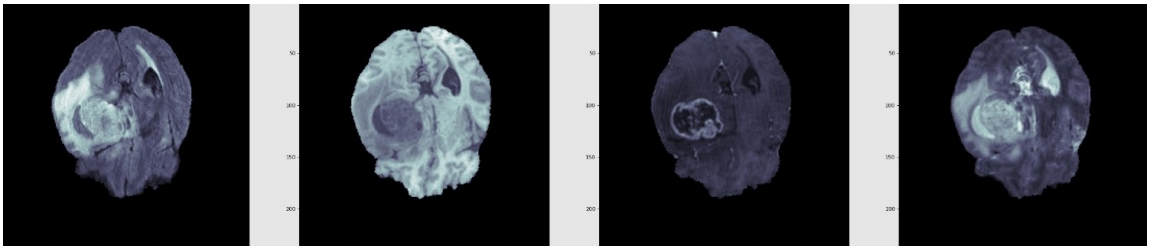


Figure 2.11: An example of BraTS sample in all four modalities

Each MRI scan in this dataset is accompanied by a corresponding binary segmentation mask, where voxels (the 3D equivalent of pixels) are labeled as one of the following:

- Necrotic and non-enhancing tumor core (NCR/NET - Label 1)
- Peritumoral edema (ED - Label 2)
- GD-enhancing tumor (ET - Label 4)

The first category includes the tumor core that is necrotic and non-enhancing. The second refers to the surrounding edema, which is the area where the tumor invades healthy tissue. The third represents the enhancing part of the tumor, which is usually the most aggressive part.

By combining the NCR, ED, and ET regions, the whole tumor area (Whole Tumor - WT) can be obtained. Also, by combining NCR and ET, the tumor core (Tumor Core - TC) area is derived.

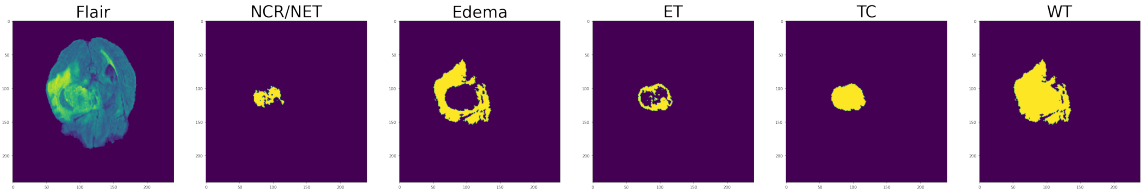


Figure 2.12: Regions corresponding to different classes in BraTS

In each modality, specific brain regions are highlighted more prominently. For example in T1, the enhancing tumor (ET) regions are more visible, and in FLAIR, the edema regions stand out more. Also, the FLAIR and T1ce modalities provide the most useful and richest information in this dataset [27].

2.5.3 Challenges of Working with BraTS

This dataset presents several challenges for developing automatic tumor segmentation algorithms. One major challenge is class imbalance in the segmentation maps. The

tumor core and enhancing tumor regions are relatively small compared to edema and background regions, making segmentation more difficult.

Another challenge is the variation in tumor appearance across different MRI modalities. Tumors can appear differently across modalities, making accurate tumor region detection in each modality difficult.

Another challenge is the large size of this dataset. Each sample has a width of 240 pixels, height of 240 pixels, and depth of 155. Considering 4 MRI modalities, each sample contains $240 \times 240 \times 155 \times 4 = 35,712,000$ data points, whereas a 2D RGB image of the same width and height contains $240 \times 240 \times 3 = 172,800$ data points. In other words, each sample in this dataset is equivalent to 206 2D images. This makes training deep models on this dataset difficult without specialized and powerful hardware. However, this 3D dataset can be treated as 2D by considering the depth dimension separately. For example, if the dataset contains 369 3D samples, it can be viewed as $369 \times 155 = 57,195$ 2D samples. In that case, each 2D sample has dimensions $240 \times 240 \times 4$.

In addition, tumor annotation is subjective and depends on the annotator's perspective. Different annotators may label tumors differently, which can lead to the problem known as Inter-Annotator Variability.

Finally, the BraTS dataset only contains a limited number of MRI scans, which may not fully represent the complete diversity of brain tumors. This challenge can be partially addressed using data augmentation techniques.

2.5.4 Applications

The BraTS dataset has many applications in medical imaging and computer vision. It is widely used for developing and evaluating algorithms for brain tumor segmentation, tumor classification, and survival prediction. It has also been used for developing deep learning models for automated diagnosis, treatment planning, and surgical guidance.

2.6 Backpropagating Refinement Scheme (BRS)

2.6.1 Introduction

BRS [18] presents an interactive image segmentation algorithm that receives user-provided points indicating the target object and background. By computing the spatial distance of each pixel to the user-provided points, interaction maps are generated. Then, a forward pass is performed in a convolutional neural network to obtain an initial segmentation map. It is possible that some input points by the user are mislabeled in this initial result. Therefore, in this research, a backpropagating refinement scheme has been developed. This scheme corrects pixels that have been mislabeled. Experimental results show that the proposed algorithm performs better than conventional methods on four challenging datasets.

2.6.2 Algorithm

The proposed interactive segmentation algorithm outputs a binary mask of the object specified by the user. This mask is generated from the cribble method, which requires foreground and background clicks. These clicks indicate the expected label for the corresponding pixels.

Figure 2.13 summarizes the proposed algorithm. Given user points as input, foreground and background interaction maps are first generated. Then the input image and these interaction maps are fed into a CNN network. Although these interaction maps clearly indicate the labels specified at the clicked locations, the probability mask may contain incorrect information from the clicked locations. Therefore, using BRS, the clicked locations are forced to have the labels specified by the user. Finally, the segmentation mask of the target object is obtained by running the forward pass again.

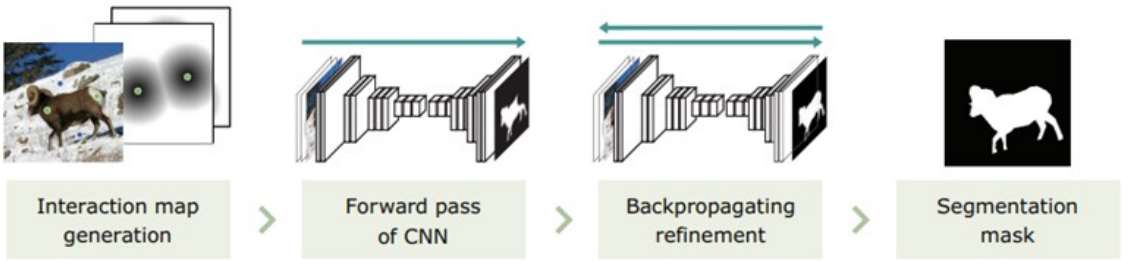


Figure 2.13: Overview of the BRS algorithm [18]

This process starts when the user performs the first click on the target object. Then, after obtaining the segmentation result, the user may click on a new location on the target object or background. This algorithm runs repeatedly, producing increasingly accurate segmentations. These two steps are executed recursively until the user stops clicking.

2.6.3 Network Architecture

The presented convolutional neural network uses an encoder-decoder architecture. The architecture of this network is shown in Figure 2.14. This network takes two interaction maps and the original image as inputs. A DenseNet [16] is used as the encoder network to extract both high-level and low-level features. Skip connections are also used to preserve low-level features in the deep layers of the network. At the end of each dense block, squeeze and excitation modules are also used.

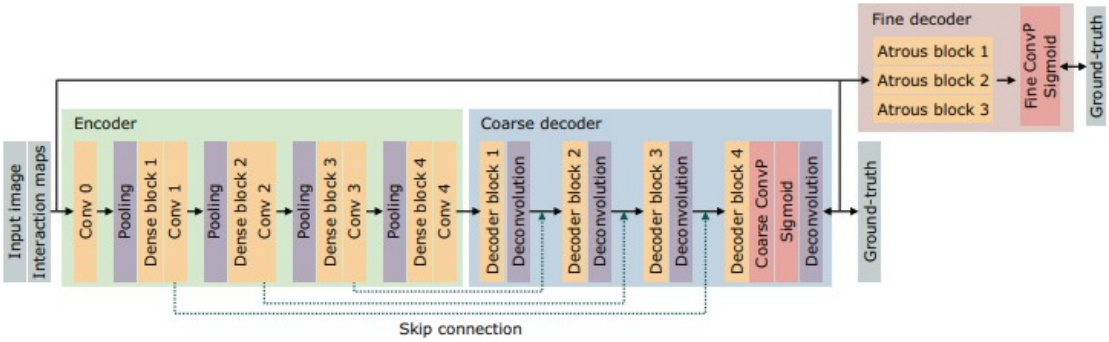
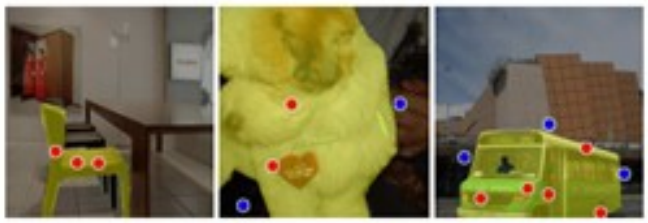


Figure 2.14: Proposed architecture based on DenseNet [16]

2.6.4 Training Phase

The Semantic Boundaries Dataset (SBD) [11] was used for training the proposed network. This dataset includes 8498 training images. Around each object, a random crop of size 360 by 360 pixels, representing the image and target mask, was extracted. Horizontal flipping was also used for data augmentation.

Since the SBD dataset does not include user input points, a simple clustering method called k-Medoids was used to generate these points. The result of this clustering algorithm is shown in Figure 2.15.



(a) 3 FG / 0 BG (b) 2 FG / 2 BG (c) 5 FG / 3 BG

Figure 2.15: Example of points generated by clustering during training [18]

2.6.5 Inference Phase

The proposed algorithm takes the original image and foreground and background interaction maps as input. First, the interaction maps are updated by computing the distance of each pixel to the nearest point. Then the updated interaction maps are fed into the neural network to produce a probability map. Pixels with probability greater than 0.5 are detected as the target object.

2.6.6 Refinement

User clicks may be mislabeled. Therefore, a criterion to compute the error and a procedure to backpropagate and reduce the error are required. The BRS algorithm performs iterative backpropagation of the error until all clicks are correctly labeled.

First, the symbols of this algorithm are defined by several tensors, whose overview

is shown in Figure 2.16. Tensors y^{r-1} and z^{r-1} are concatenated, and parameters θ^r and ϕ^r are used to obtain y^r.

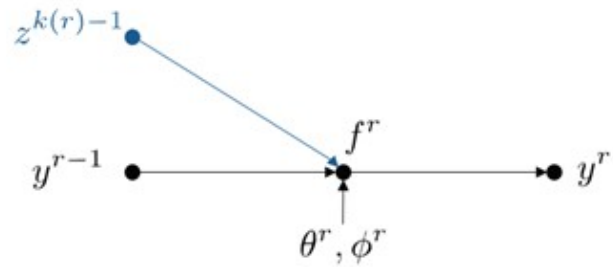


Figure 2.16: Mathematical representation of the BRS relation [18]

This parameter represents the response of the r-th layer of the network. The value of y^r is expressed in Equation 2.14.

$$y^r = f^r(y^{r-1}, z^{r-1}, \theta^r, \phi^r) \tag{2.14}$$

By using a suitable loss function and performing backpropagation on the above function, this network can be trained.

2.6.7 Conclusion

Although this method has not been tested on the BraTS dataset, experiments show that the proposed algorithm outperforms conventional and traditional algorithms on several datasets. In fact, using BRS significantly improved the NoC metric, and user input points were used more efficiently.

Chapter 3

Methodology

3.1 Introduction

This section focuses on the click-based interactive segmentation method. Click-based interaction is essentially seed point-based interaction, which is referred to here simply as clicks. Under this method, the user sequentially places positive clicks, such as the red points in Figure 3.1, to define the foreground (FG), and negative clicks, such as the green points in Figure 3.1, to define the background (BG). After each user click, the model outputs updated predictions.

Figure 3.1: Types of clicks [5]

The main paradigm for click-based interactive segmentation is the use of Gaussian maps or disks to represent user clicks [44]. In this method, a Gaussian map centered at the user click with a specified radius is generated. Then, the click maps are concatenated with the input image and fed into a segmentation model to predict the initial mask.

Based on this method, previous works that were based on other paradigms have been improved in various aspects. However, when applying these methods in practical and industrial scenarios, the results are not satisfactory in some respects, which are discussed below.

3.2 Problems of Previous Methods

3.2.1 Poor Efficiency on Low-Computing-Power Devices

A good segmentation tool is expected to produce quality masks in a short amount of time. Most previous works focus solely on accuracy and use large models with high-resolution inputs. However, when deployed on personal laptops, edge devices, or web-based applications with heavy requests, they face difficulties. Additionally, when trying to reduce input size for faster performance, the accuracy of these models significantly decreases.

3.2.2 Lack of Use of Existing Pre-Existing Masks

In practical and real-world applications, the image to be segmented may already have an initial segmentation mask. These existing masks can be provided by a prior offline model or other preprocessing methods. The previously generated mask by the same model can also be considered. Utilizing these existing masks and making modifications on them can facilitate the segmentation task. However, previous methods are not compatible with external and pre-existing masks. Many of these methods do not even consider the existence of a prior mask as input. While in some works the prior mask is concatenated with the input image and the maps generated from user clicks as input to the model, these methods still do not perform well in utilizing pre-existing masks.

Not using pre-existing masks causes many problems. For example, in Figure 3.2, the user intends to remove the ship's cabin from the previous mask by placing several negative clicks (green) on the cabin to delete that part. But after running the model

again, it is observed that the model not only removes the cabin but also removes other parts that are far from the user clicks and were correctly labeled in the previous output. In some cases, it even mistakenly adds new parts to the output mask.

(a) Initial Mask (b) After Editing

Figure 3.2: Unintended changes when generating new output [5]

In fact, when the user clicks on a region, they intend the model to focus on that region. For example, in Figure 3.2, the user wants the model to pay attention to the ship's cabin, not the surrounding environment. However, many previous methods ignore the user's intention and treat all pixels in the image equally when producing the new output. This causes two phenomena. First, regions that were previously well segmented require re-segmentation, a phenomenon called computational redundancy. Second, all pixels of the previous mask must be segmented again, and the model loses the previously existing information about them, which is called loss of information.

3.3 Features of the Proposed Algorithm

The features of the proposed algorithm must be aligned with and correspond to solving the mentioned problems. The attention and focus of this algorithm should be on specific local regions. Also, new predictions should be made only for the regions that require recomputation, and the algorithm should update only the outputs of the regions that the user intends to update. Besides these, the algorithm should preserve the previous output in other regions. Considering these points, problems, and goals, this project uses the *FocalClick* model and algorithm [5] for interactive brain tumor

segmentation. The following explains different parts of this algorithm according to the *FocalClick* paper [5].

3.4 Algorithm at a Glance

When the user makes a new click, first a crop from the original image, called the *Target Crop*, is created. This crop is constructed based on the pre-existing mask. In Figure 3.3, this crop is shown with a yellow bounding box.

Then, the *Target Crop* is resized, and its resolution is also reduced. A coarse segmentation is performed on this smaller, low-resolution image, a step known as *Coarse Segmentation*.

Next, a smaller region where the user intends to modify the segmentation output is identified. Since the model's focus should be on this region, it is called the *Focus Crop*, marked by a red boundary in Figure 3.3. This focused region then goes into the refinement stage, called *Local Refinement*, to recover its lost quality and resolution. Therefore, the heavy and time-consuming operation of segmenting a large image is converted into two local, low-cost operations.

Finally, the refined region is fed into the *Progressive Merge* step, where a morphological analysis is performed on the pre-existing mask and the new output to decide whether to update or preserve previous labels. Hence, regions that were previously correctly predicted are not altered.

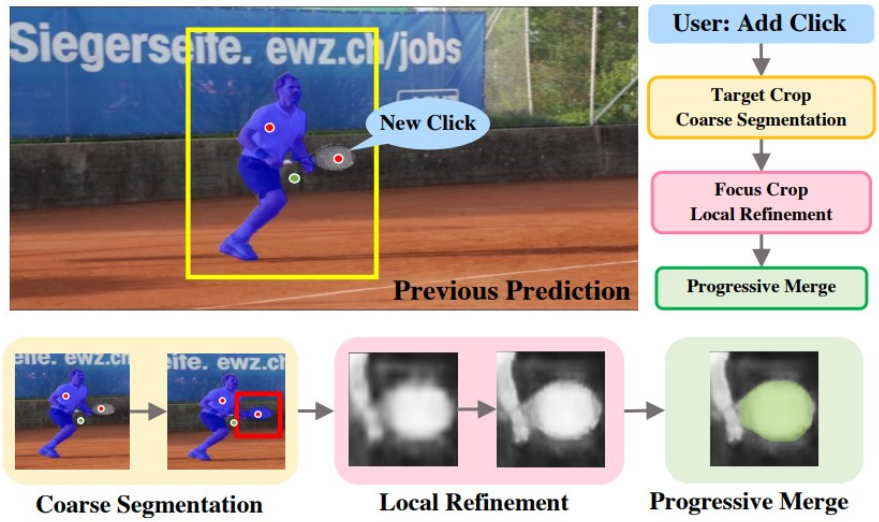


Figure 3.3: A general overview of the *FocalClick* algorithm [5]

From the above explanations, it is clear that the method relies heavily on the cropping operation, which is also seen in some other segmentation works. However, the difference of this algorithm compared to others is that other methods require preserving the resolution of the cropped image and their segmenters produce detailed outputs, which is time-consuming and slow. But in this algorithm, the resolution of the cropped region is reduced and the segmenter's output is fast and coarse.

3.5 Algorithm Details

In this section, the parts and stages of the algorithm are examined with details. First, the model receives the input image, two maps corresponding to negative and positive clicks, and the pre-existing mask as inputs. To create the map related to the user clicks, a binary disk with a radius of 2 is used. In other words, the area inside a circle with radius 2 and centered at the click is labeled 1, and outside this circle labeled 0. Then, the *Target Crop* is created around the target object, and its size and resolution are reduced. This crop is then fed into the segmentation network to produce a coarse output. The *Focus Crop* is created by calculating the difference between regions of the previous mask and the coarse segmentation. This crop is used

to refine and recover details lost in the coarse image. Finally, the *Progressive Merge* stage merges the previous mask and the refined region. The details of each step of this algorithm are explained below.

3.5.1 Target Crop

The goal of this part is to filter the background region from the target object region. To create this region, a minimum enclosing bounding box that includes the pre-existing mask (target object) and the user's new click is generated. This bounding box is then expanded by a hyperparameter called *ratio_TC*. The value of this hyperparameter is set to 1/4. As seen in Figure 3.1, the yellow rectangle contains the user's new click and the target object, with slightly expanded dimensions. After expanding this bounding box and obtaining its coordinates, three tensors related to the image, previous mask, and user click maps are cropped according to this bounding box. Finally, the resolution and size of these cropped tensors are reduced to prepare for the next stage.

3.5.2 Coarse Segmentation

In this part, the binary inputs corresponding to negative clicks, positive clicks, and the pre-existing mask are passed through two convolutional layers to become channel-wise compatible with the input image. The input image also passes through two convolutional layers to extract low-level features. Then the extracted image features and the three binary inputs are combined.

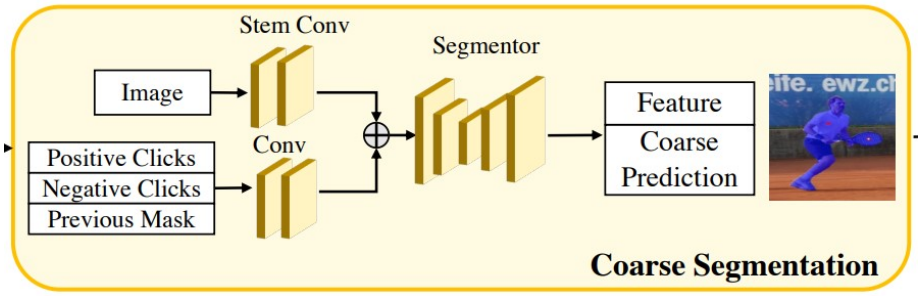


Figure 3.4: Architecture of the Coarse Segmentation part [5]

These combined features are fed into a segmentation network. This network can be any segmentation network such as *U-Net* [33], *HRNet* [41] (suitable for high-resolution images), or transformer-based segmentation networks like *SegFormer* [43]. A common characteristic of these networks is the encoder-decoder architecture along with skip connections to preserve input image information. Finally, the coarse segmentation output is produced.

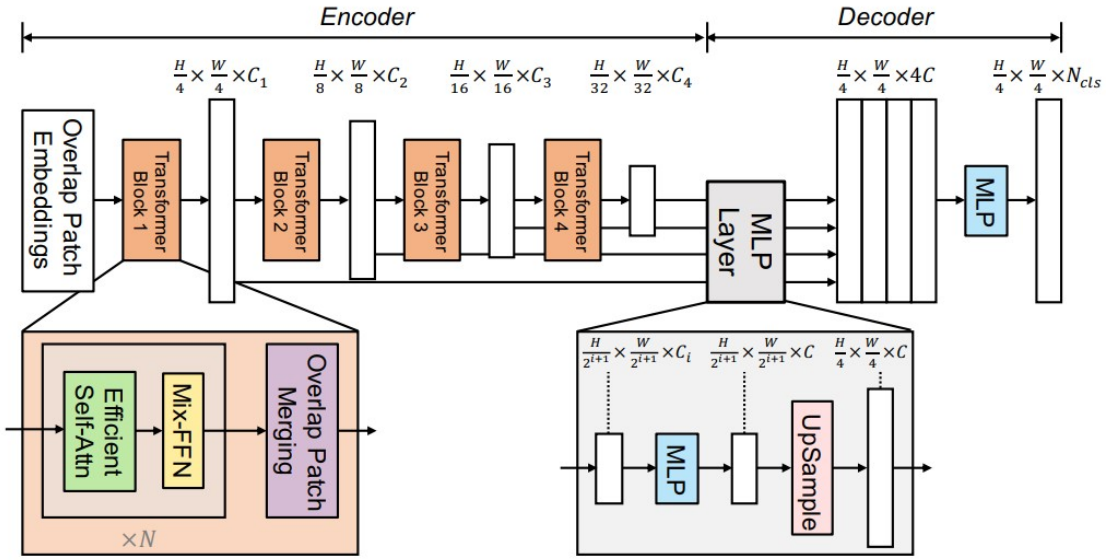


Figure 3.5: Architecture of the SegFormer network [43]

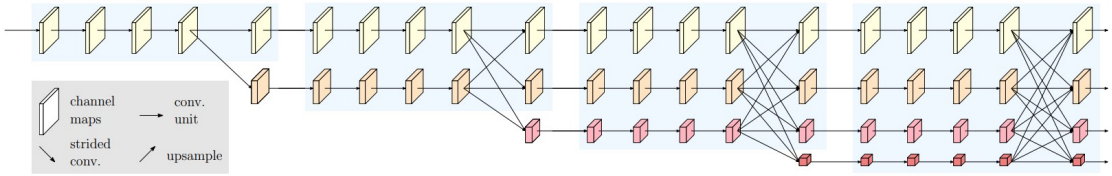


Figure 3.6: Architecture of the HRNet network [41]

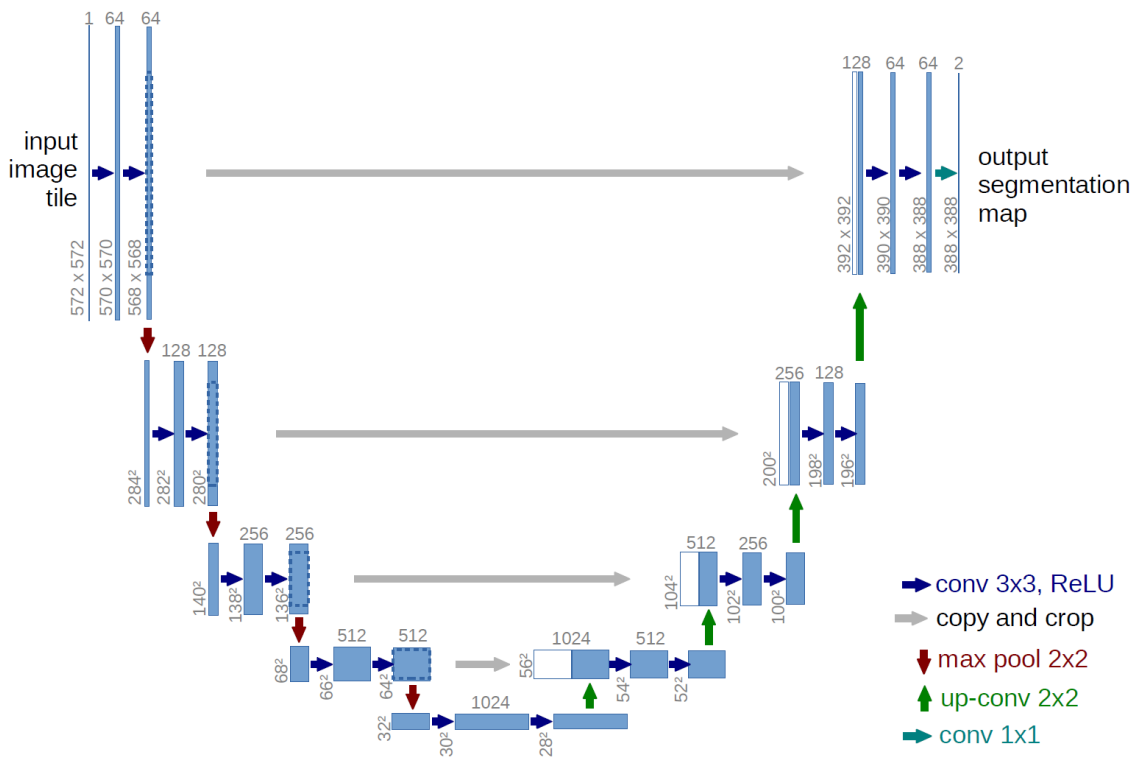


Figure 3.7: Architecture of the U-Net network [33]

3.5.3 Focus Crop

This part first calculates the difference between the pre-existing mask and the coarse segmentation output. This comparison is done using the logical XOR function, resulting in a new mask called the *Difference Mask (DM)*. Then, the largest connected region within this mask that contains the user's new click is calculated. A bounding box is created around this largest connected region. Similar to the Target Crop step, this bounding box is expanded using a hyperparameter called *ratio_FC*, which is also set to 1/4. Finally, the *Focus Crop*, shown in red in Figure 3.3, is obtained. According to the obtained Focus Crop, the input image and user click maps are also cropped. Furthermore, the *RoiAlign* module [13] is used to crop features and logits from the segmentation network output.

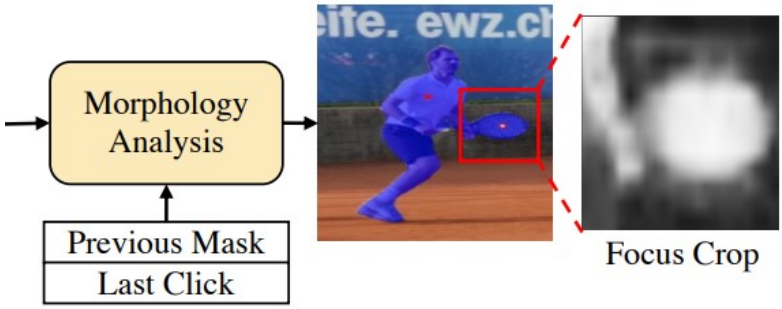


Figure 3.8: Architecture of the Focus Crop part [5]

3.5.4 Local Refinement

This part recovers and refines the details of the coarse prediction made on the Focus Crop. First, low-level features are extracted from the cropped tensors using convolutional layers of *Xception* [6]. Meanwhile, the number of channels of the *RoiAlign* features is adjusted and then merged with the obtained low-level features.

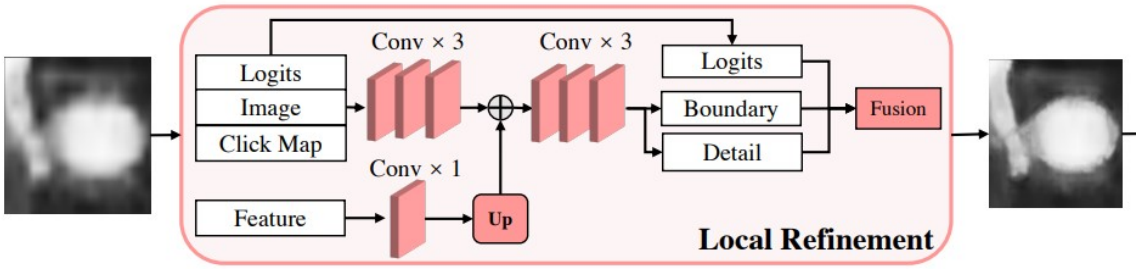


Figure 3.9: Architecture of the Local Refinement part [5]

To obtain the refined predictions, two heads are used to predict a detail map M_d and a boundary map M_b. Then the refined prediction map M_r is calculated using the following formula. In this formula, M_l represents the logits predicted by the Coarse Segmentor network.

$$M_r = Sigmoid(M_b) * M_d + (1 - Sigmoid(M_b)) * M_l \tag{3.1}$$

3.5.5 Progressive Merge

When labeling or editing masks, we do not expect the model to update the mask for all pixels after every click. Otherwise, all details that the algorithm has correctly labeled would be updated again, and even previously correct labels may be lost. The user only wants updates in limited areas where they intend to make corrections. In the Progressive Merge stage, similar to the Target Crop and Focus Crop parts, morphology-based analysis is used to find the region needing correction.

After receiving a user click, the new predicted mask is binarized with a threshold of 0.5, and the differing area between the new prediction and existing mask is computed. Then the largest connected region containing the new click is selected as the update region (the green area in Figure 3.10). In this region, the new predicted mask is placed over the previous mask and updated. The previous mask remains unchanged in other regions.

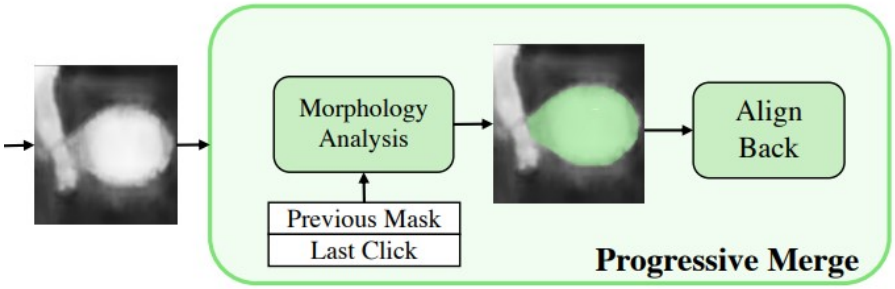


Figure 3.10: Architecture of the Progressive Merge part [5]

When the algorithm is run with a pre-existing mask or an initial mask from other segmentation tools, the Progressive Merge step is applied to preserve correct details. However, during initial labeling without a prior mask, the Progressive Merge part activates after 10 clicks.

3.6 Loss Function

Since this algorithm consists of different parts, a loss function is considered for each of these sections, and by combining these functions, the final loss function is obtained. The algorithm is trained under the supervision of this function. The loss function for the boundary map M_b is obtained by downsampling the ground truth segmentation 8 times and then upsampling it back to the original size. In this way, the changed pixels can represent regions that require more accuracy and detail. The *Binary Cross Entropy (BCE)* loss is used to supervise the *Boundary Head*. Also, to supervise the output of the coarse segmentation, the *Normalized Focal Loss (NFL)* [37] is used. For supervising the *Local Refinement* part, a weighted version of the *NFL* loss is used. This weight is set to $b = 1.5$, and the new weighted loss function is abbreviated as *BNFL*. Finally, the overall loss function of the entire network is obtained by summing these three loss functions.

$$L = L_{BCE} + L_{NFL} + L_{BNFL} \tag{3.2}$$

3.7 Combining Different MRI Modalities

As mentioned in Chapter 2, each MRI image sample in the BraTS dataset is available in four modalities: *FLAIR*, *T1ce*, *T2*, and *T1*. Each of these modalities highlights specific parts of the brain or brain tumor better. These modalities exist as grayscale maps, with each pixel having an intensity between 0 and 1. In other studies focusing on brain MRI image segmentation, usually only one of these modalities is used, and the modality used is often either *FLAIR* or *T1ce* [23, 40].

One effective innovation employed in this project is the simultaneous use of three modalities: *FLAIR*, *T1ce*, and *T2*. To do this, a similar approach to normal RGB images—which are formed by merging three red, green, and blue channels—is ap-

plied. That is, the three modalities *FLAIR*, *T1ce*, and *T2* are concatenated to create a three-channel image. For convenience, this is called the *Mix* mode.

Although the image produced by merging these three modalities (an example is shown in Figure 3.11) may not be meaningful or easily interpretable for humans and can even be misleading, a deep learning model can extract the necessary features from each channel to produce better outputs. Another advantage of this approach is that the input image to the model resembles RGB images, which makes it easier to use pretrained models that were trained on large datasets such as ImageNet [7]. Previously, to use pretrained models, the selected modality had to be repeated three times, which not only provided no new information to the model but also significantly increased the number of model parameters.

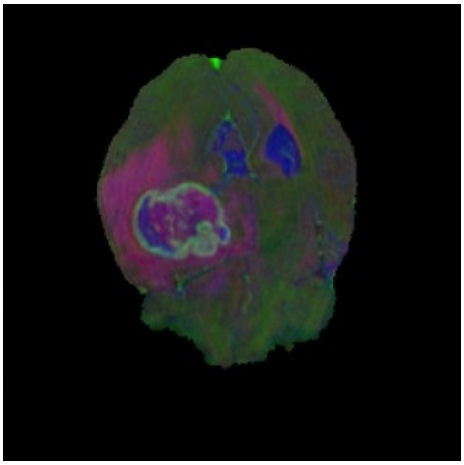


Figure 3.11: Example of the image created in the modality combination mode

Chapter 4

Experiments and Results

4.1 Preparing the BraTS Dataset

The FocalClick algorithm is suitable for high-resolution natural color images, and the inputs to the Refiner and Segmentor parts remain at size 256×256 even after cropping. This algorithm is designed for 2D RGB images. However, the BraTS dataset consists of 3D images in four modalities with dimensions $240 \times 240 \times 155$. Therefore, adapting this algorithm to this dataset requires modifications in both. First, we change the input size of the Refiner and Segmentor parts from 256×256 to 96×96 to better fit the BraTS image sizes. Also, since processing and training 3D images requires very high computational power and large memory, and due to the absence of 3D modules like RoiAlign in deep learning frameworks such as PyTorch and TensorFlow, we feed the BraTS dataset to the network as 2D. For this, each depth slice of the dataset is considered a new sample. The modalities FLAIR, T1ce, and the novel Mix mode are also used in the experiments.

4.2 Distribution of the Dataset into Training, Validation, and Testing Sets

Since the BraTS test set is not publicly available, the 2020 BraTS training dataset is used for training, validation, and testing the algorithm. This version of the dataset contains 369 3D samples in four modalities. Of these 369 samples, 75% are used

for training the model, 20% for validating the model, and 5% for testing the model. Table 4.1 summarizes the number of samples in 2D and 3D forms for the training, validation, and testing sets.

Table 4.1: Distribution of Training, Validation, and Testing Sets

	Percentage	**3D Cases**	**2D Cases**
Training Set	75%	276	42780
Validation Set	20%	73	11315
Testing Set	5%	20	3100
Total	100%	369	57195

4.3 Training Environment

Some hyperparameters and settings remain fixed throughout all conducted experiments and are mentioned here. All models were trained for 20 epochs. Also, samples are fed into the model in batches of size 32 to enable parallel processing. Using larger batches causes memory overflow and algorithm failure. Cropping and resolution reduction are performed so that the Target Crop and Focus Crop regions become 96×96 in size. The hyperparameters ratio_TC and ratio_FC, which are used to expand the bounding box dimensions in yellow and red respectively, are set to 1/4. Experiments were conducted in two environments: Google Colab with Nvidia K80/T4 GPUs and Kaggle with Nvidia P100 GPUs. Links to these environments, experimental results, model output images at each training stage (Visualization), model weights, and algorithm implementation are provided in the appendices.

4.4 Defining and Running Experiments

To define and run experiments, it is sufficient to run one of the Google Colab or Kaggle notebooks. These notebooks are designed so that by changing existing variables, various experiments can be conducted. For example, one can select one of the

existing segmentation networks as the Backbone. It is also possible to select different modalities during both training and evaluation phases. The user's task can also be defined, allowing them to specify their desired region and mask. Details on how to define experiments are shown in Figures 4.1 and 4.2. Additionally, simultaneous training on multiple GPUs is supported.

```
MODE = "FocalClick"  # CDNet, FocalClick
BACKBONE = "segformerB3"  # cdnet_res34, hrnet18s, hrnet32, segformerB3

# CDNet-ResNet34, FocalClick-HRNet18s, FocalClick-HRNet32, FocalClick-SegFormerB3
EXP_NAME = "FC-SF-Mix-WT"

CHANNEL = "mix"  # flair, t1, t1ce, t2, mix
LABEL = "wt"  # net, ed, et, wt, tc
EPOCHS = 20
BATCH_SIZE = 32
NGPUS = 1
WORKERS = 2

MODEL_FILE = f"{REPO_PATH}/models/brats/{BACKBONE}.py"
MODEL_DIR = f"{REPO_PATH}/experiments/brats/{BACKBONE}/000_{EXP_NAME}/checkpoints/"

# Set --pretrained in below cell
```

Figure 4.1: How to define variables in the training phase

```
MODE = "FocalClick"  # CDNet, FocalClick
BACKBONE = "segformerB3"  # cdnet_res34, hrnet18s, hrnet32, segformerB3
EXP_NAME = "000_FocalClick-SegFormerB3"
CHECKPOINT = "epoch-19-val-loss-1.28.pth"

CHANNEL = "t1ce"  # flair, t1, t1ce, t2, mix
LABEL = "tc"  # net, ed, et, wt, tc

MODEL_DIR = f"{REPO_PATH}/experiments/brats/{BACKBONE}/{EXP_NAME}/checkpoints/"
```

Figure 4.2: How to define variables in the evaluation phase

4.5 General Notes on Experimental Results

The Dice metric is always higher than the IoU metric because Dice contains a factor of 2 in its formula. The analysis and interpretation of results focus more on Dice, while IoU is used for comparing this algorithm's results with other papers that report results only in IoU format.

Also, the high value of NoF is due to many 3D image slices converted to 2D lacking any foreground pixels. In such cases, the algorithm counts these samples as Failures.

In the first image provided for each experiment, starting from the top left, the parts represent Full Image, then Full Ground Truth, then Full Prediction, and on the right, Previous Full Prediction. At the bottom, starting from the left, the parts represent Refine Image, then Refine Focus Prediction, then Refine Prediction, and on the right, Trimap Prediction (a combination of negative and positive click maps).

In the second image for each experiment, the region and IoU achieved after 20 clicks are shown. This image also displays the locations of positive and negative clicks. The Target Crop and Focus Crop regions are marked with yellow and red bounding boxes. Additionally, Ground Truth, the model's final output, and the penultimate model output are shown.

In the third and final image for each experiment, the Focus Crop region is presented in two forms: Coarse and Refined.

4.6 Exp 1: HRNet18s-FLAIR-WT-Pretrained

In this experiment, the FocalClick algorithm is executed using the HRNet18s backbone and segmenter, which is pretrained on the ImageNet dataset. The FLAIR modality is used to create the dataset. The task and objective of the experiment is to find the region corresponding to the whole tumor (WT). After training the model is completed, the test set evaluation is performed on the same modality and the same task as in the training phase.

Table 4.2: NoC and NoF metrics in Experiment 1

Metric	**NoC80%**	**NoC85%**	**NoC90%**	**NoF85%**	**NoF90%**	**TPC**	**Time**
IoU	3.47	4.31	6.00	325	568	0.061	0:56:38
Dice	2.50	2.92	3.72	180	259		

Table 4.3: mIoU and mDice metrics in Experiment 1

Metric	**Clicks=1**	**Clicks=2**	**Clicks=3**	**Clicks=5**	**Clicks=10**	**Clicks=20**
mIoU	84.27 %	85.77 %	86.59 %	88.08 %	89.79 %	90.94 %
mDice	88.39 %	89.67 %	90.28 %	91.36 %	92.63 %	93.55 %

Since the HRNet18s segmenter is a lightweight segmenter with a small number of parameters, the algorithm is able to produce new outputs quickly after receiving a new click. Also, to reach an average Dice score of 90%, 3.72 clicks are required on average, which is reasonably acceptable. The algorithm, after receiving 20 clicks, achieves an average Dice score of 93.55%, which again indicates good performance. Moreover, as the number of clicks increases, the model's performance improves, demonstrating the impact of user clicks and the model's effective use of them.

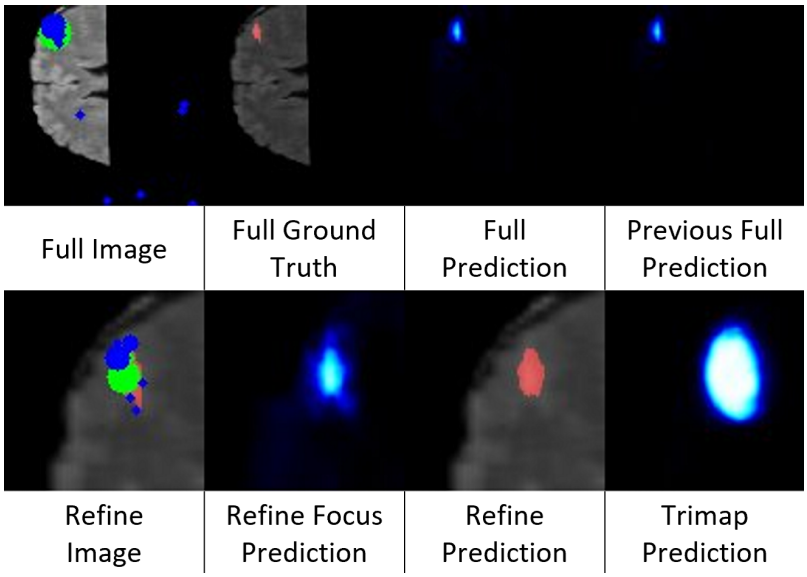


Figure 4.3: Algorithm output at one of the training phase steps in Experiment 1

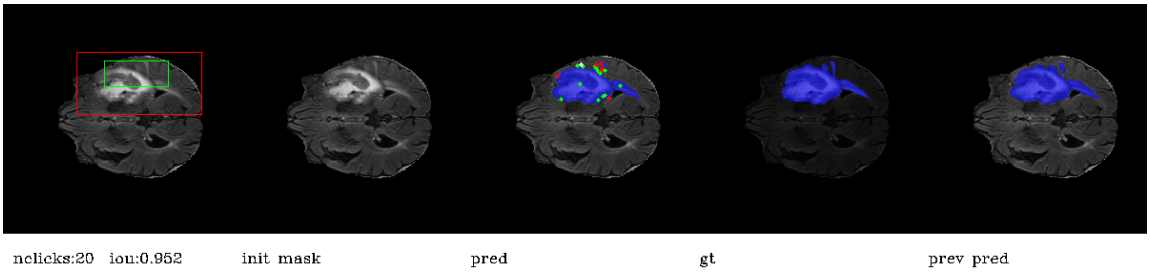


Figure 4.4: Algorithm output on one of the evaluation samples in Experiment 1

Figure 4.5: Detail recovery of one evaluation sample image in Experiment 1

4.7 Exp 2: HRNet32-FLAIR-WT-Pretrained

In this experiment, the FocalClick algorithm is executed using the Backbone and the HRNet32 segmenter, which is pretrained on the ImageNet dataset. The FLAIR modality is also used to create the dataset. The task and goal of the experiment is to find the region corresponding to the whole tumor (WT). After the model training is complete, the test set evaluation is performed on the same modality and the same task as the training phase.

Table 4.4: NoC and NoF metrics in Experiment 2

Metric	**NoC80%**	**NoC85%**	**NoC90%**	**NoF85%**	**NoF90%**	**TPC**	**Time**
IoU	4.23	5.11	6.53	515	697	0.103	1:35:37
Dice	3.02	3.53	4.52	326	443		

Table 4.5: mIoU and mDice metrics in Experiment 2

Metric	**Clicks=1**	**Clicks=2**	**Clicks=3**	**Clicks=5**	**Clicks=10**	**Clicks=20**
mIoU	85.58%	79.77%	79.02%	79.39%	80.66%	81.69%
mDice	89.03%	82.69%	81.84%	81.65%	82.46%	83.34%

Considering that HRNet32 is a large network with a very high number of parameters and is suitable for high-resolution images, it is observed that the algorithm response time reaches 1.03 seconds, which is twice as long compared to other experiments. Additionally, this network performs worse in all available metrics compared to the previous experiment. Given that HRNet32 is a larger version of HRNet18s, it can be inferred that the model in this experiment has overfitted.

Another interesting point is that with increasing the number of clicks, the IoU and Dice values worsen. This means that the network could not make good use of user interactions, and these guidance clicks even degraded the result. By examining the output images of the model at each stage, it can be seen that due to the model size, an appropriate segmentation is initially produced, but over time, the overfitting

phenomenon dominates the user clicks, and the result deteriorates with time and more clicks.

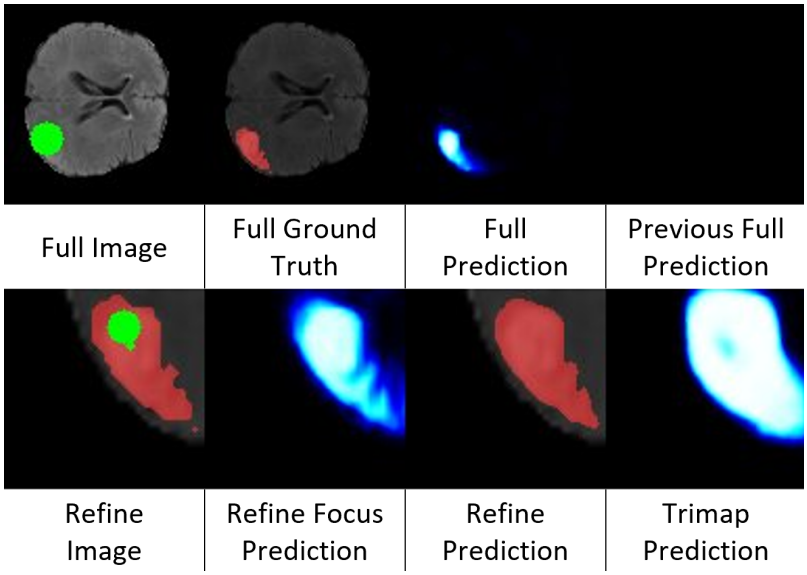


Figure 4.6: Algorithm output at one of the training phase steps in Experiment 2

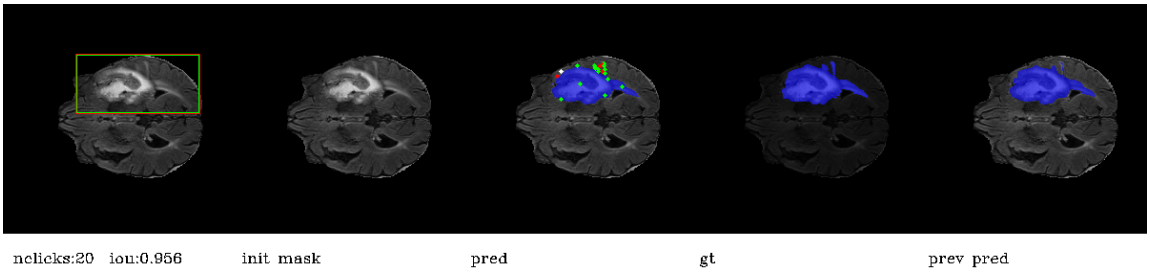


Figure 4.7: Algorithm output on one of the evaluation samples in Experiment 2

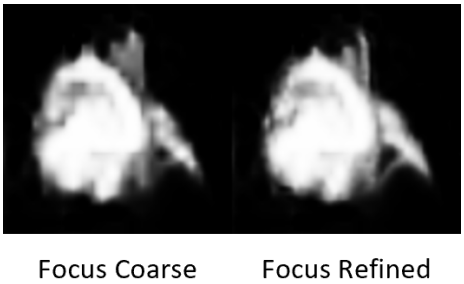


Figure 4.8: Detail recovery of one evaluation sample image in Experiment 2

4.8 Exp 3: SegFormerB3-FLAIR-WT-Pretrained

In this experiment, the FocalClick algorithm is executed using the Backbone and the SegFormerB3 segmenter, which is pretrained on the ImageNet dataset. The FLAIR modality is used to create the dataset. The task and goal of the experiment is to find the region corresponding to the whole tumor (WT). After the model training is complete, the test set evaluation is performed on the same modality and the same task as the training phase.

Table 4.6: NoC and NoF metrics in Experiment 3

Metric	**NoC80%**	**NoC85%**	**NoC90%**	**NoF85%**	**NoF90%**	**TPC**	**Time**
IoU	3.28	4.03	5.75	320	530	0.059	0:55:07
Dice	2.45	2.81	3.48	176	246		

Table 4.7: mIoU and mDice metrics in Experiment 3

Metric	**Clicks=1**	**Clicks=2**	**Clicks=3**	**Clicks=5**	**Clicks=10**	**Clicks=20**
mIoU	85.35%	85.30%	86.46%	88.37%	90.29%	91.75%
mDice	88.96%	88.84%	89.75%	91.16%	92.78%	94.12%

Considering that SegFormer is a lighter segmenter with fewer parameters compared to HRNet32, the algorithm was able to produce new outputs in good time (approximately similar to Experiment 1 but slightly better) after receiving a new click. Also, to reach an average Dice score of 90%, 3.48 clicks are needed, which is a good improvement compared to the previous two experiments. With 20 clicks, the algorithm achieves an average Dice of 94.12%, which again shows better performance compared to the previous two experiments. Moreover, as the number of clicks increases, the model performance improves, indicating the influence of user clicks and the model's effective use of these clicks. In summary, using the SegFormerB3 segmenter has improved the model's performance in all aspects.

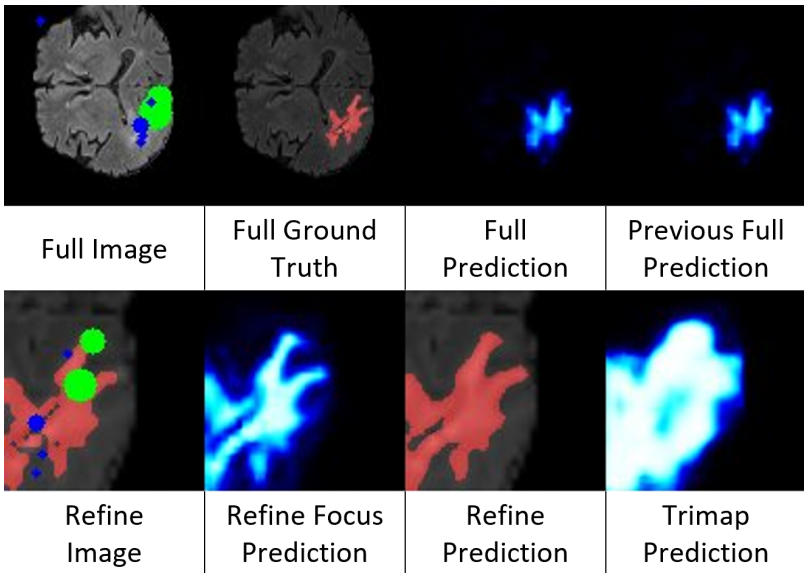


Figure 4.9: Algorithm output at one of the training phase steps in Experiment 3

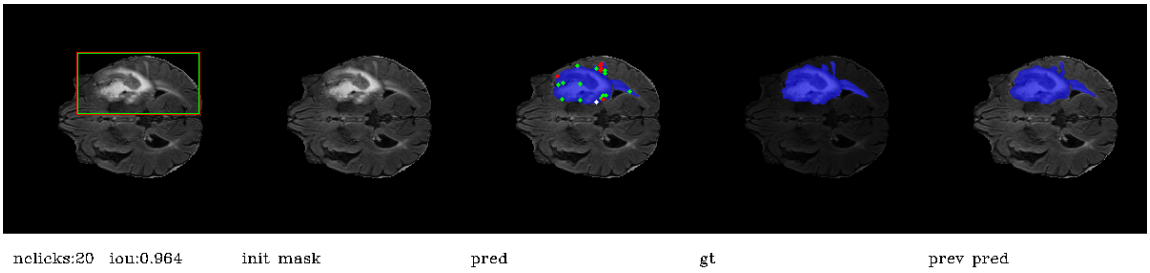


Figure 4.10: Algorithm output on one of the evaluation samples in Experiment 3

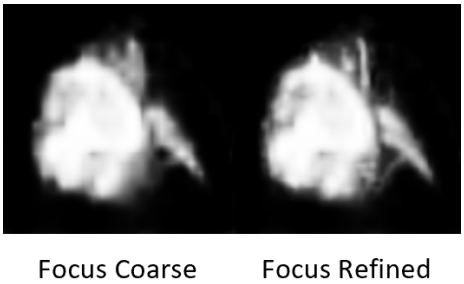


Figure 4.11: Detail recovery of one evaluation sample image in Experiment 3

4.9 Exp 4: SegFormerB3-Mix-WT and Modality Fusion

In this experiment, the FocalClick algorithm is used with the Backbone and the SegFormerB3 segmenter without any pretraining. Also, the Mix modality, which combines FLAIR, T1ce, and T2 modalities, is used to create the dataset. The goal of this experiment is to observe the effect of simultaneous use of modalities on the model's performance. The task is to find the region corresponding to the whole tumor (WT). After training, the test set evaluation is performed on the same modalities and the same task as in the training phase.

Table 4.8: NoC and NoF metrics in Experiment 4

Metric	**NoC80%**	**NoC85%**	**NoC90%**	**NoF85%**	**NoF90%**	**TPC**	**Time**
IoU	3.24	4.15	5.63	319	533	0.061	0:56:54
Dice	2.21	2.66	3.53	138	233		

Table 4.9: mIoU and mDice metrics in Experiment 4

Metric	**Clicks=1**	**Clicks=2**	**Clicks=3**	**Clicks=5**	**Clicks=10**	**Clicks=20**
mIoU	86.17%	87.31%	88.36%	89.73%	91.74%	93.07%
mDice	89.57%	90.86%	91.74%	92.82%	94.49%	95.56%

Since this experiment uses the SegFormerB3 segmenter as in Experiment 3, with no change in model parameters, the response time and execution time are similar to Experiment 3, which is very reasonable. Comparing the NoC and mDice results of this experiment with the previous one shows overall improvement in both metrics. This confirms that simultaneous and combined use of modalities is effective. For example, in Experiment 3, to reach an IoU of 90%, the model needed on average 5.75 clicks, while in this experiment it requires 5.63 clicks. Also, in Experiment 3 with 20 clicks, the mIoU was 91.75%, whereas in this experiment it improved to 93.07%.

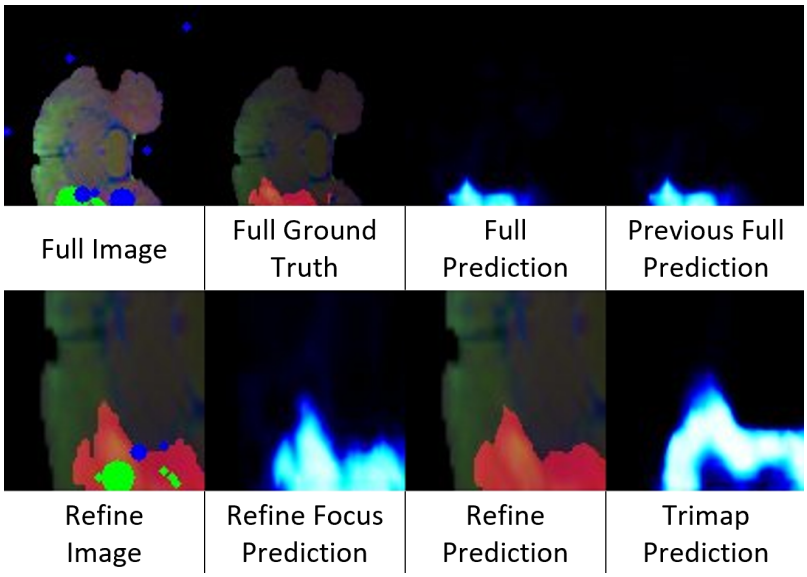


Figure 4.12: Algorithm output at one of the training phase steps in Experiment 4

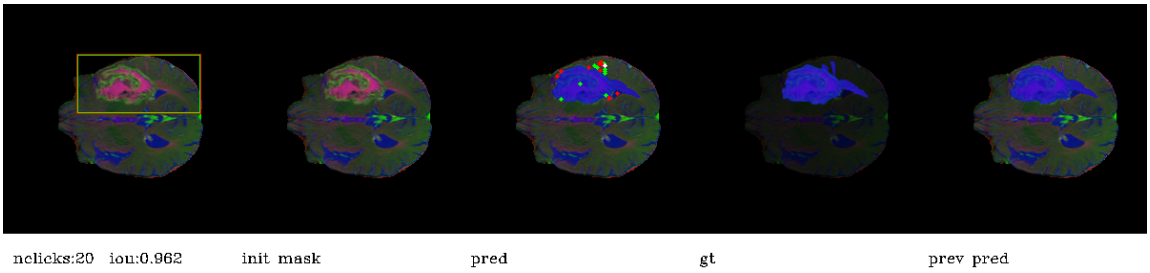


Figure 4.13: Algorithm output on one of the evaluation samples in Experiment 4

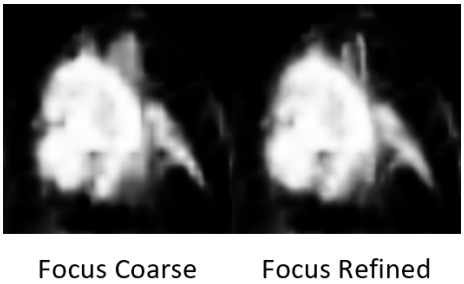


Figure 4.14: Detail recovery of one evaluation sample image in Experiment 4

4.10 Exp 5: Evaluation on an Unseen Task During Training

In this experiment, we aim to evaluate the model's performance on a task that it has not encountered during training (Unseen Task). For this purpose, the weights of the model trained in Experiment 3 are used to evaluate this new task. In Experiment 3, the defined task was finding the region corresponding to the whole tumor (WT), and the model was trained accordingly. However, in this experiment, the task is to find the region corresponding to the tumor core (TC). By analyzing and interpreting the results of this experiment, we can understand how useful user clicks are for guiding the model towards the new task despite the task change.

Table 4.10: NoC and NoF metrics in Experiment 5

Metric	**NoC80%**	**NoC85%**	**NoC90%**	**NoF85%**	**NoF90%**	**TPC**	**Time**
IoU	5.96	6.39	6.82	667	790	0.059	0:54:53
Dice	4.93	5.49	6.13	456	596		

Table 4.11: mIoU and mDice metrics in Experiment 5

Metric	**Clicks=1**	**Clicks=2**	**Clicks=3**	**Clicks=5**	**Clicks=10**	**Clicks=20**
mIoU	78.04%	78.29%	78.96%	80.30%	82.85%	86.92%
mDice	81.76%	82.12%	82.71%	83.83%	86.09%	89.75%

First, it should be noted that the TC region is smaller than the WT region, which makes segmenting the TC area more difficult. Therefore, lower IoU and Dice values compared to previous experiments are expected. In fact, even if the model is trained on the TC task, its IoU and Dice values will still be lower.

Observing the results, we see that initially, when the model has received only one click, its performance is very poor with an mDice of 81.76%. This is due to the task change. However, with an increasing number of clicks, the model manages to understand the new task and act accordingly. When the number of clicks reaches

20, the mDice value improves to 89.75%, which is a suitable value for the TC task. Also, in this experiment, the effect of the number of user clicks on mIoU and mDice values is more pronounced compared to previous experiments. The output images generated at different algorithm stages also show that when the number of user clicks is low, the model outputs the WT region, but with more clicks, it realizes that the user is interested in a different region, i.e., the TC.

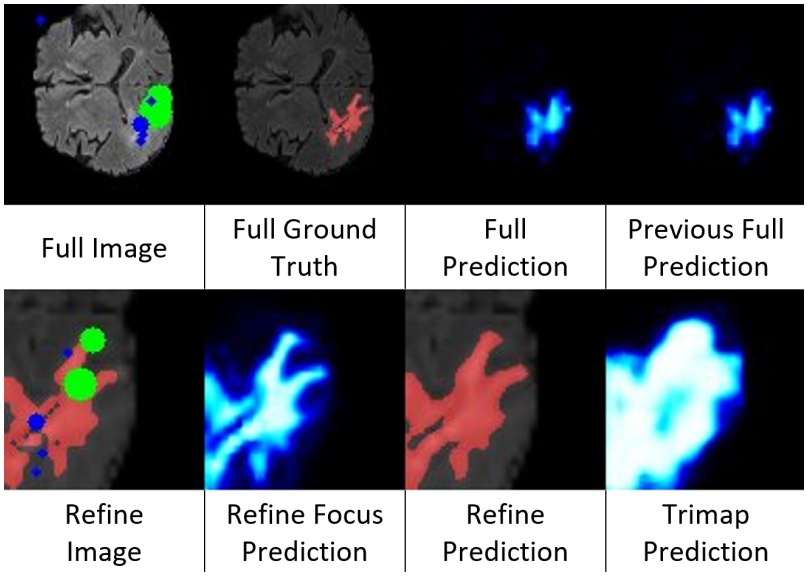


Figure 4.15: Algorithm output at one of the training phase steps in Experiment 5 (same as Experiment 3)

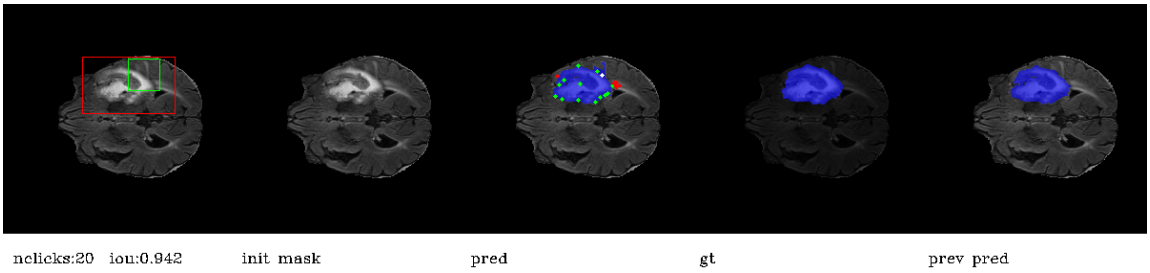


Figure 4.16: Algorithm output on one of the evaluation samples in Experiment 5

Figure 4.17: Detail recovery of one evaluation sample image in Experiment 5

4.11 Exp 6: Evaluation on Unseen Modality and Task

In this experiment, we aim to simultaneously evaluate the model's performance on a modality and task that were not encountered during training (Unseen Modality and Task). To this end, we use the weights of the model trained in Experiment 3 to evaluate the new modality and task. In Experiment 3, the defined task was to find the region corresponding to the Whole Tumor (WT) based on the FLAIR modality, and the model was trained accordingly on that modality and task. However, in this experiment, the task is to find the Tumor Core (TC) region using the T1ce modality. By analyzing and interpreting the results of this experiment, we can understand how useful the user clicks have been for guiding the model toward the new modality and task despite the changes.

Table 4.12: NoC and NoF metrics in Experiment 6

Metric	**NoC80%**	**NoC85%**	**NoC90%**	**NoF85%**	**NoF90%**	**TPC**	**Time**
IoU	4.56	4.87	5.42	480	576	0.058	0:54:22
Dice	4.20	4.38	4.68	421	458		

Table 4.13: mIoU and mDice metrics in Experiment 6

Metric	**Clicks=1**	**Clicks=2**	**Clicks=3**	**Clicks=5**	**Clicks=10**	**Clicks=20**
mIoU	79.80 %	79.27 %	80.06 %	81.31 %	82.96 %	85.49 %
mDice	81.52 %	80.77 %	81.47 %	82.58 %	84.21 %	86.77 %

It should be noted that the TC region is smaller in size compared to the WT region, making segmentation of the TC part more challenging. Therefore, having lower IoU and Dice values compared to previous experiments is natural. In fact, even if the model is trained on the TC task, the IoU and Dice values will still be lower. Additionally, as mentioned in Chapter 2, the FLAIR modality highlights the Edema region. The Whole Tumor region includes the Edema area, whereas the Tumor Core region does not. The T1ce modality highlights the enhancing tumor, which is present in both the WT and TC masks. All these factors make the model face a challenging and difficult task.

By observing the results, we see that initially when the model receives only one click, its performance is very poor with an mDice value of 81.52%. This poor performance is related to the change in both modality and task. However, as the number of clicks increases, the model appears to understand the new modality and task and performs accordingly. When the number of clicks reaches 20, the mDice value improves to 86.77%. This value is lower than Experiment 5, where the modality was unchanged, indicating the negative impact of modality change. Nevertheless, ultimately the model does not perform poorly given a high number of clicks. One of the project goals is to reduce the required user effort, and the result of this experiment contradicts that goal, since the user must exert considerable effort to achieve satisfactory results with changes in modality and task.

Also, in this experiment, the effect of the number of user clicks on mIoU and mDice values is more pronounced compared to previous experiments. From the output images produced in different algorithm stages, it can also be seen that when the number of user clicks is low, the model outputs the WT region; with increasing clicks, the model realizes that another region, namely TC, is the user's intended target.

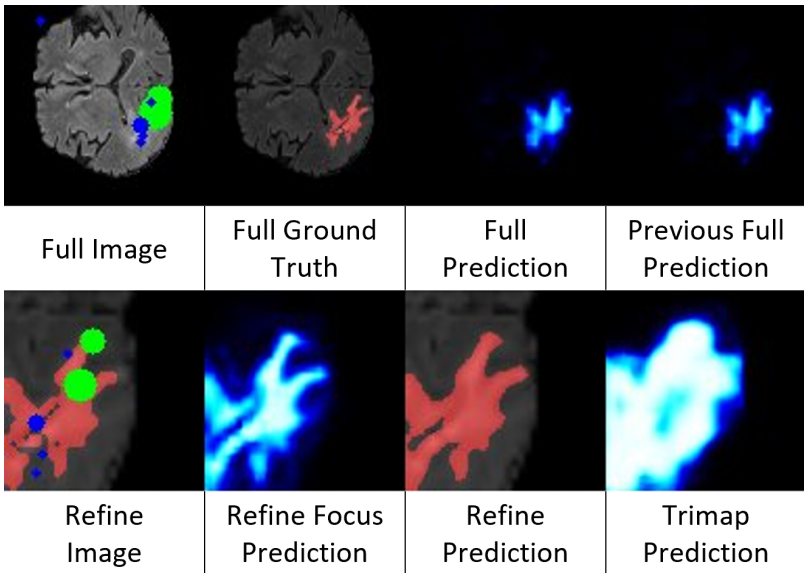


Figure 4.18: Algorithm output at one of the training phase steps in Experiment 6 (same as Experiment 3)

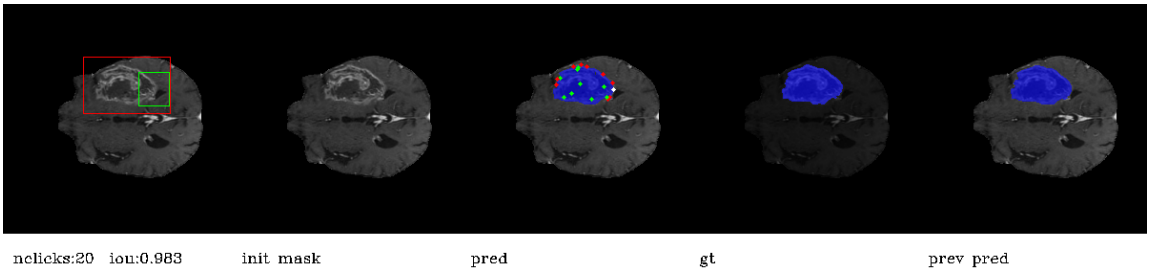


Figure 4.19: Algorithm output on one of the evaluation samples in Experiment 6

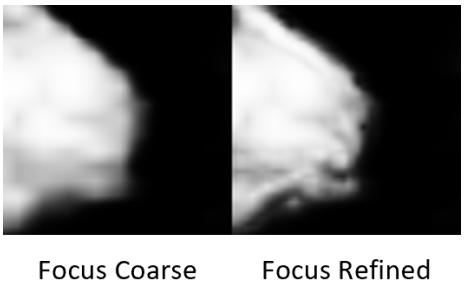


Figure 4.20: Detail recovery of one evaluation sample image in Experiment 6

4.12 Comparison with Related Work Results

There are four issues that make it difficult to have a definitive and precise comparison between different works on the *BraTS* dataset. These four issues are:

1. The existence of annual versions of the *BraTS* dataset (from 2012 to 2022) and the use of different versions by various articles.
2. Different ways articles split the *BraTS* dataset into training, validation, and test sets.
3. Presence of four modalities for each image and different articles using different subsets of these modalities.
4. Existence of several different masks and articles using different masks from this dataset.

Despite these challenges, it is still possible to provide an approximate comparison between the results of our proposed method and other methods. In this section, the results of our proposed method are compared with those of three papers: *Enhanced U-Net* [27], *DeepIGeoS* [40], and *MIDeepSeg* [23]. Finally, a brief comparison with older and non-deep learning papers is presented.

4.12.1 The *Enhanced U-Net* Paper

The paper *Enhanced U-Net* [27] proposes a non-interactive segmentation method. In this approach, the *U-Net* network is used for segmentation. Also, separate annual versions of the dataset from 2017 to 2020 are used. This paper divides the 2019 version of the dataset into 68% training, 20% validation, and 12% testing. Furthermore, the modalities *FLAIR* and *T1ce* are utilized. As shown in Table 4.14, this paper achieves an approximate Dice score of 84%, which is lower than our interactive proposed method. The significant difference between this paper's result and ours is due to its non-interactive nature, highlighting the importance and impact of user interaction.

Table 4.14: Results of *Enhanced U-Net* [27]

Dataset	Loss	Accuracy	mIoU	Precision	Sensitivity	Specificity	Dice
BraTS 2017	0.0056	0.9980	0.9637	0.9973	0.9970	0.9972	0.8453
BraTS 2018	0.0057	0.9979	0.8927	0.9972	0.9970	0.9940	0.8160
BraTS 2019	**0.0054**	**0.9981**	**0.9130**	**0.9974**	**0.9971**	**0.9991**	**0.8409**
BraTS 2020	0.0056	0.9980	0.8935	0.9973	0.9970	0.9983	0.8300

4.12.2 The *DeepIGeoS* Paper

The paper *DeepIGeoS* [40] proposes an interactive segmentation method. This method uses geodesic distance to convert user clicks into interaction maps. Also, it employs the *P-Net* [38], *R-Net* [4], and *CRF-Net* [22] networks simultaneously for segmentation.

This paper uses the 2015 version of the *BraTS* dataset. The 2015 dataset is split into 85% training and 15% validation and testing. Only the *FLAIR* modality is used. The defined task is segmentation on the *WT* mask. As shown in Table 4.15, this paper achieves a best Dice score of 89.93%, which is slightly lower than our proposed method.

Table 4.15: Results of *DeepIGeoS* [40]

Method	Dice (%)	ASSD (pixels)
Before refinement	87.55 ± 6.72	2.04 ± 1.70
Min-cut user-editing	88.41 ± 7.05	1.74 ± 1.53
3D R-Net (Euc)	88.82 ± 7.68	1.60 ± 1.56
3D R-Net	89.30 ± 6.82	1.52 ± 1.37
3D R-Net (Euc) + CRF-Net (fu)	89.27 ± 7.32	1.48 ± 1.22
3D R-Net + CRF-Net (fu)	$\mathbf{89.93 \pm 6.49}$	$\mathbf{1.43 \pm 1.16}$

4.12.3 The *MIDeepSeg* Paper

The paper *MIDeepSeg* [23] proposes an interactive segmentation method. In this approach, the Exponentialized Geodesic Distance (EGD) is used to convert user clicks into interaction maps. Additionally, the *U-Net* network [33] is used for the initial segmentation. To apply user clicks on the generated initial segmentation, a non-deep method called *Graph Cut* is employed.

This paper uses the 2018 version of the *BraTS* dataset. The dataset is split into 78% training and 22% validation and testing. Moreover, the modalities *FLAIR* and *T1ce* are used separately. The defined tasks are segmentation on the *WT* and *TC* masks.

Table 4.16: Dataset distribution in the *MIDeepSeg* paper [23]

Object	Modality	N_{train}	N_{test}	Dataset
Tumor core	MRI (T1ce)	170	47	BraTS2018
Whole tumor	MRI (FLAIR)	0	60	BraTS2018

This paper, similar to our proposed method, evaluates its performance on an unseen modality and task. As shown in Table 4.17, this paper achieves a best Dice score of 88.71%, which is lower than our proposed method.

Table 4.17: Results of the *MIDeepSeg* paper [23]

Method	Dice (%)	ASSD (mm)	Time (s)
3D Graph Cuts	78.91 ± 14.98	3.46 ± 5.10	99.4 ± 36.7
ITK-SNAP	82.34 ± 11.42	1.99 ± 1.31	173.0 ± 75.5
DeepIGeoS	82.47 ± 10.78	3.34 ± 3.81	82.2 ± 44.7
DeepGrabCut	82.58 ± 11.79	2.89 ± 2.37	$\mathbf{10.5 \pm 8.3}$
DIOS	83.34 ± 10.47	2.57 ± 1.79	67.5 ± 23.6
DEXTR	86.39 ± 9.03	1.59 ± 1.11	34.7 ± 18.6
MIDeepSeg	$\mathbf{88.71 \pm 7.00}$	$\mathbf{1.24 \pm 0.88}$	28.6 ± 12.2

4.12.4 Traditional and Non-Deep Methods

In this section, the Dice scores for traditional and non-deep methods are presented. Due to the large number of these methods, their details are not discussed. A noteworthy point is the large difference in Dice scores in these methods compared to deep methods. For the WT task, Dice scores range from 44% to 82%. Also, for the TC task, this metric ranges from 17% to 70%.

Table 4.18: Results of traditional and non-deep methods [26]

	whole		core		active	time (min) (arch)
Dice (in %)		LG / HG		LG / HG		
Bauer	68	49 / 74	48	30 / 54	57	8 (CPU)
Buendia	57	19 / 71	42	8 / 54	45	0.3 (CPU)
Cordier	68	60 / 71	51	41 / 55	39	20 (Cluster)
Doyle	74	63 / 78	44	41 / 45	42	15 (CPU)
Festa	62	24 / 77	50	33 / 56	61	30 (CPU)
Geremia	62	55 / 65	32	34 / 31	42	10 (Cluster)
Guo	74	71 / 75	65	59 / 67	49	<1 (CPU)
Hamamci	72	55 / 78	57	40 / 63	59	20 (CPU)
Meier	69	46 / 77	50	36 / 55	57	6 (CPU)
Menze (D)	78	81 / 76	58	58 / 59	54	20 (CPU)
Menze (G)	69	48 / 77	33	9 / 42	53	10 (CPU)
Reza	70	52 / 77	47	39 / 50	55	90 (CPU)
Riklin Raviv	74	na / 74	50	na / 50	58	8 (CPU)
Shin	30	28 / 31	17	22 / 15	5	8 (CPU)
Subbanna	75	55 / 82	70	54 / 75	59	70 (CPU)
Taylor	44	24 / 51	28	11 / 34	41	1 (Cluster)
Tustison	75	68 / 78	55	42 / 60	52	100 (Cluster)
Zhao (I)	82	78 / 84	66	60 / 68	49	15 (CPU)
Zhao (II)	76	67 / 79	51	42 / 55	52	20 (CPU)
Zikic	75	62 / 80	47	33 / 52	56	2 (CPU)
Best Combination	88	86 / 89	78	66 / 82	71	
Fused	82	68 / 87	73	62 / 77	65	

Chapter 5

Conclusion

5.1 Conclusion

In this project, the method and algorithm FocalClick [5] were used as a click-based deep interactive segmentation approach to segment MRI brain medical images and identify tumor regions. The conducted experiments showed that the proposed algorithm performs significantly better than previous interactive and non-interactive algorithms. These experiments were carried out on the BraTS 2020 dataset, and in all tests, the results obtained were superior to prior methods. This improved performance is attributed to the presence of user auxiliary clicks, which, by providing a method for effectively utilizing these clicks, demonstrated how beneficial user interaction can be. Moreover, the satisfactory performance of the FocalClick algorithm in metrics such as TPC and NoC indicates that this algorithm, unlike previous methods, is fully suitable for industrial and real-world applications. The FocalClick algorithm significantly enhances the efficiency of existing pipelines and enables their deployment on low-power devices. In addition to the above, the experimental results showed that the proposed algorithm generalizes well across different tasks. This generalization is such that the model, trained on one task, can achieve satisfactory results on an unseen task. The following section discusses potential future directions for this project.

5.2 Suggestions and Future Work

5.2.1 Adapting FocalClick for 3D Input

In this project, due to the absence of 3D morphology modules and the undefined parts of the FocalClick algorithm for the 3D case, as well as the challenging nature of designing these parts in 3D, the 3D BraTS dataset was converted to 2D by slicing along the depth dimension. This can cause loss of spatial information and ultimately reduce performance. However, due to memory limitations and the large size of each 3D sample, feeding the full 3D samples into the model is not feasible. To solve this issue, instead of only cropping as used in FocalClick, a combination of cropping and slicing can be applied. In this way, rather than converting each data sample of size $240 \times 240 \times 155$ entirely to 96×96 and losing the third dimension completely, the sample can be resized to $96 \times 96 \times 48$, preserving spatial information along the third dimension. This approach has been performed in the DeepIGeoS [40] and MIDeepSeg [23] papers with effective results.

5.2.2 Using BRS in FocalClick

The BRS paper [18] demonstrates the adaptability and positive impact of the BRS method on several CNN networks, showing that the use of BRS reduces the NoC metric. Reducing the required number of clicks and the user effort is one of the most important goals in designing interactive segmentation models. One of the future works of this research could be incorporating the BRS method into the FocalClick algorithm. However, this would require significant changes to the existing algorithm.

5.2.3 Using Exponential Geodesic Distance Transform

In the algorithm used in this project, user clicks are converted to binary maps. In other words, each user click is converted to a circle with a radius of 2. This is one of the simplest available transformations and provides less information to the model compared to other common methods like Euclidean or Gaussian distance. The

DeepIGeoS paper uses geodesic distance to convert user clicks to interaction maps. Furthermore, the MIDeepSeg paper uses an improved form of this distance called Exponential Geodesic Distance (EGD). In Figure 5.1, interaction maps created by different methods are shown. As observed, the EGD transformation provides the best interaction map for the model, almost accurately representing the user's intended region. Using the EGD transform can be considered as one of the future works.

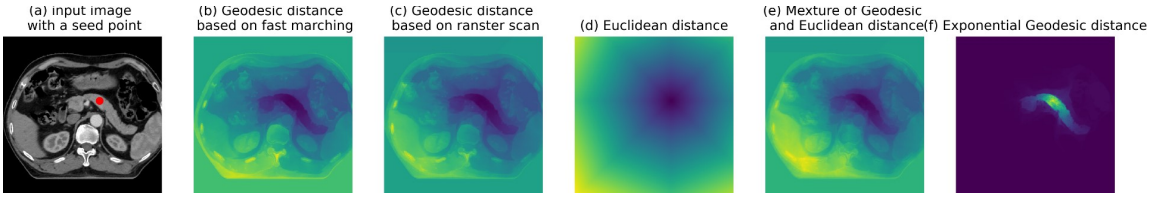


Figure 5.1: Comparison of different transformations for user clicks

5.2.4 Other Suggestions

Besides the above, using the 2022 version of the BraTS dataset, which contains a total of 5880 MRI scans (approximately 10 times more than the 2020 version), could be very suitable for model training. In this project, very simple data augmentation methods such as flipping and rotation were used. In the future, more advanced augmentation techniques can be applied. Using other segmenters and backbones could also be part of future work.

Appendices

- GitHub repository: Link
- Experiment results, trained model, and output images: Link
- Google Colab environment: Link
- Kaggle training & evaluation environment: Link
- Kaggle pre-trained model evaluation environment: Link
- BraTS dataset: Link

References

[1] R. Adams and L. Bischof, 'Seeded region growing,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 641–647, 1994. DOI: 10.1109/34.295913 (cit. on p. 13).

[2] Y. Boykov, O. Veksler and R. Zabih, 'Fast approximate energy minimization via graph cuts,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222–1239, 2001. DOI: 10.1109/34.969114 (cit. on p. 11).

[3] D.-J. Chen, H.-T. Chen and L.-W. Chang, 'Interactive segmentation from 1-bit feedback,' in *Computer Vision–ACCV 2016*, S.-H. Lai, V. Lepetit, K. Nishino and Y. Sato, Eds., Cham: Springer International Publishing, 2017, pp. 261–274, ISBN: 978-3-319-54181-5 (cit. on pp. 7, 10).

[4] S. Chen, Z. Chen, L. Yan and W. Guo, 'R-net: A novel fully convolutional network-based infrared image segmentation method for intelligent human behavior analysis,' *Infrared Physics & Technology*, vol. 123, p. 104 164, Apr. 2022. DOI: 10.1016/j.infrared.2022.104164 (cit. on p. 58).

[5] X. Chen, Z. Zhao, Y. Zhang, M. Duan, D. Qi and H. Zhao, 'Focalclick: Towards practical interactive image segmentation,' in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2022, pp. 1290–1299. DOI: 10.1109/CVPR52688.2022.00136 (cit. on pp. 28, 30–33, 36, 37, 61).

[6] F. Chollet, 'Xception: Deep learning with depthwise separable convolutions,' in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA: IEEE Computer Society, Jul. 2017, pp. 1800–1807. DOI: 10.1109/CVPR.2017.195 (cit. on p. 36).

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, 'Imagenet: A large-scale hierarchical image database,' in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848 (cit. on p. 39).

[8] O. Elharrouss, S. A. Al-Maadeed, N. Subramanian, N. Ottakath, N. Almaadeed and Y. Himeur, 'Panoptic segmentation: A review,' *ArXiv*, vol. abs/2111.10250, 2021 (cit. on p. 6).

[9] 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,' *IEEE Reviews in Biomedical Engineering*, vol. 13, pp. 156–168, 2020, ISSN: 1937-3333. DOI: 10.1109/RBME.2019.2946868 (cit. on p. 1).

[10] D. Guo, Y. Pei, K. Zheng, H. Yu, Y. Lu and S. Wang, 'Degraded image semantic segmentation with dense-gram networks,' *IEEE Transactions on Image Processing*, vol. 29, pp. 782–795, 2020. DOI: 10.1109/TIP.2019.2936111 (cit. on p. 5).

[11] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji and J. Malik, 'Semantic contours from inverse detectors,' in *2011 International Conference on Computer Vision*, 2011, pp. 991–998. DOI: 10.1109/ICCV.2011.6126343 (cit. on p. 26).

[12] J. He, C.-S. Kim and C.-C. J. Kuo, 'Interactive image segmentation techniques,' in *Interactive Segmentation Techniques: Algorithms and Performance Evaluation*. Singapore: Springer Singapore, 2014, pp. 17–62, ISBN: 978-981-4451-60-4. DOI: 10.1007/978-981-4451-60-4_3. [Online]. Available: https://doi.org/10.1007/978-981-4451-60-4_3 (cit. on p. 6).

[13] K. He, G. Gkioxari, P. Dollár and R. Girshick, 'Mask r-cnn,' in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988. DOI: 10.1109/ICCV.2017.322 (cit. on p. 35).

[14] K. He, X. Zhang, S. Ren and J. Sun, 'Deep residual learning for image recognition,' in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90 (cit. on p. 13).

[15] A. Hernandez-Vela, A. Hernández-Vela, C. Primo and S. Escalera, 'Automatic user interaction correction via multi-label graph cuts,' in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2011, pp. 1276–1281. DOI: 10.1109/ICCVW.2011.6130398 (cit. on p. 9).

[16] G. Huang, Z. Liu and K. Q. Weinberger, 'Densely connected convolutional networks,' *CoRR*, vol. abs/1608.06993, 2016. [Online]. Available: http://arxiv.org/abs/1608.06993 (cit. on p. 25).

[17] K. T. Islam, S. Wijewickrema and S. O'Leary, 'A deep learning framework for segmenting brain tumors using mri and synthetically generated ct images,' *Sensors*, vol. 22, no. 2, 2022, ISSN: 1424-8220. DOI: 10.3390/s22020523 (cit. on p. 2).

[18] W.-D. Jang and C.-S. Kim, 'Interactive image segmentation via backpropagating refinement scheme,' in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019 (cit. on pp. 24–27, 62).

[19] A. Kirillov, K. He, R. B. Girshick, C. Rother and P. Dollár, 'Panoptic segmentation,' *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9396–9405, 2018 (cit. on p. 5).

[20] K. M. Lee, H. Myeong and G. Song, 'Seednet: Automatic seed generation with deep reinforcement learning for robust interactive segmentation,' in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1760–1768. DOI: 10.1109/CVPR.2018.00189 (cit. on p. 14).

[21] A. Y. Li and M. Iv, 'Conventional and advanced imaging techniques in post-treatment glioma imaging,' *Frontiers in Radiology*, vol. 2, 2022, ISSN: 2673-8740. DOI: 10.3389/fradi.2022.883293 (cit. on p. 1).

[22] H. Li and P. Peers, 'Crf-net: Single image radiometric calibration using cnns,' in *Proceedings of the 14th European Conference on Visual Media Production (CVMP 2017)*, Dec. 2017, pp. 1–9, ISBN: 978-1-4503-5329-8. DOI: 10.1145/3150165.3150170 (cit. on p. 58).

[23] X. Luo *et al.*, 'Mideepseg: Minimally interactive segmentation of unseen objects from medical images using deep learning,' *Medical Image Analysis*, vol. 72, p. 102 102, 2021, ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2021.102102 (cit. on pp. 10, 38, 57, 59, 62).

[24] K.-K. Maninis, S. Caelles, J. Pont-Tuset and L. Van Gool, 'Deep extreme cut: From extreme points to object segmentation,' in *Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018, pp. 616–625. DOI: 10.1109/CVPR.2018.00071 (cit. on p. 13).

[25] M. Meila, 'Comparing clusterings: An axiomatic view,' in *Proceedings of the 22nd International Conference on Machine learning*, Jan. 2005, pp. 577–584. DOI: 10.1145/1102351.1102424 (cit. on p. 18).

[26] B. H. e. a. Menze, 'The multimodal brain tumor image segmentation benchmark (brats),' *IEEE Transactions on Medical Imaging*, vol. 34, no. 10, pp. 1993–2024, 2015. DOI: 10.1109/TMI.2014.2377694 (cit. on pp. 19, 60).

[27] M. A. A. Nasim, A. Munem, M. Islam, M. A. H. Palash, M. M. A. Haque and F. M. Shah, 'Brain tumor segmentation using enhanced u-net model with empirical analysis,' *ArXiv*, vol. abs/2210.13336, 2022 (cit. on pp. 2, 22, 57, 58).

[28] F. Nielsen and R. Nock, 'On region merging: The statistical soundness of fast sorting, with applications,' in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 2, 2003, pp. II–19. DOI: 10.1109/CVPR.2003.1211447 (cit. on p. 4).

[29] J. Ning, L. Zhang, D. Zhang and C. Wu, 'Interactive image segmentation by maximal similarity based region merging,' *Pattern Recognition*, vol. 43,

no. 2, pp. 445–456, 2010, Interactive Imaging and Vision, ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2009.03.004 (cit. on p. 13).

[30] D. C. Preston, *Magnetic resonance imaging (mri) of the brain and spine: Basics*, https://case.edu/med/neurology/NR/MRI%20Basics.htm, Accessed: 2023-03-07, 2006 (cit. on p. 21).

[31] H. Ramadan, C. Lachqar and H. Tairi, 'A survey of recent interactive image segmentation methods,' *Computational Visual Media*, Aug. 2020. DOI: 10.1007/s41095-020-0177-5 (cit. on pp. 6, 9).

[32] Rayleigh, 'The problem of the random walk,' *Nature*, vol. 72, pp. 318–318, 1905 (cit. on p. 12).

[33] O. Ronneberger, P. Fischer and T. Brox, 'U-net: Convolutional networks for biomedical image segmentation,' in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Cham: Springer International Publishing, 2015, pp. 234–241, ISBN: 978-3-319-24574-4 (cit. on pp. 34, 35, 59).

[34] C. Rother, V. Kolmogorov and A. Blake, 'Grabcut -interactive foreground extraction using iterated graph cuts,' *ACM Transactions on Graphics (SIGGRAPH)*, Aug. 2004 (cit. on p. 12).

[35] C. Rother, V. Kolmogorov and A. Blake, 'Grabcut: Interactive foreground extraction using iterated graph cuts,' *ACM Trans. Graph.*, vol. 23, pp. 309–314, Aug. 2004. DOI: 10.1145/1186562.1015720 (cit. on p. 8).

[36] C. Rupprecht, L. Peter and N. Navab, 'Image segmentation in twenty questions,' in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3314–3322. DOI: 10.1109/CVPR.2015.7298952 (cit. on p. 10).

[37] K. Sofiiuk, I. A. Petrov and A. Konushin, 'Reviving iterative training with mask guidance for interactive segmentation,' in *2022 IEEE International Conference on Image Processing (ICIP)*, 2022, pp. 3141–3145. DOI: 10.1109/ICIP46576.2022.9897365 (cit. on p. 38).

[38] W. Song, H. Yu and J. Wu, *Patch network for medical image segmentation*, 2023. DOI: 10.48550/ARXIV.2302.11802. [Online]. Available: https://arxiv.org/abs/2302.11802 (cit. on p. 58).

[39] T. V. Spina, P. A. V. de Miranda and A. Xavier Falcão, 'Hybrid approaches for interactive image segmentation using the live markers paradigm,' *IEEE Transactions on Image Processing*, vol. 23, no. 12, pp. 5756–5769, 2014. DOI: 10.1109/TIP.2014.2367319 (cit. on p. 8).

[40] G. Wang *et al.*, 'Deepigeos: A deep interactive geodesic framework for medical image segmentation,' *IEEE Transactions on Pattern Analysis and Machine*

Intelligence, vol. PP, Jul. 2017. DOI: 10.1109/TPAMI.2018.2840695 (cit. on pp. 38, 57, 58, 62).

[41] J. Wang *et al.*, *Deep high-resolution representation learning for visual recognition*, 2019. DOI: 10.48550/ARXIV.1908.07919. [Online]. Available: https://arxiv.org/abs/1908.07919 (cit. on p. 34).

[42] M. Xian, Y. Zhang, H.-D. Cheng, F. Xu and J. Ding, 'Neutro-connectedness cut,' *Trans. Img. Proc.*, vol. 25, no. 10, pp. 4691–4703, Oct. 2016, ISSN: 1057-7149. DOI: 10.1109/TIP.2016.2594485. [Online]. Available: https://doi.org/10.1109/TIP.2016.2594485 (cit. on p. 7).

[43] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez and P. Luo, 'Segformer: Simple and efficient design for semantic segmentation with transformers,' in *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*, Dec. 2021 (cit. on p. 34).

[44] N. Xu, B. Price, S. Cohen, J. Yang and T. Huang, 'Deep interactive object selection,' in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2016, pp. 373–381. DOI: 10.1109/CVPR.2016.47 (cit. on pp. 13, 28).

[45] N. xu, B. Price, S. Cohen, J. Yang and T. Huang, 'Deep grabcut for object selection,' in *2017 British Machine Vision Conference*, Jul. 2017. DOI: 10.5244/C.31.182 (cit. on p. 14).

[46] J. Yi, P. Wu, M. Jiang, Q. Huang, D. J. Hoeppner and D. N. Metaxas, 'Attentive neural cell instance segmentation,' *Medical image analysis*, vol. 55, pp. 228–240, 2019 (cit. on p. 5).