SAPIENZA
UNIVERSITÀ DI ROMA

3D-UNet Based Approach for Brain Tumor Segmentation

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Corso di laurea in Applied Computer Science and Artificial Intelligence

Candidate

Veronica Di Gennaro
ID number 1984033

Thesis Advisor
Daniele Pannone

Co-Advisor
Danilo Avola

Academic Year 2023/2024

3D-UNet Based Approach for Brain Tumor Segmentation
Bachelor's thesis. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: verodige02@gmail.com

A mio nonno Flaminio,
che se fosse ancora qui
sarebbe in prima fila a guardarmi oggi.

Abstract

Brain tumor segmentation is a critical task in medical imaging, significantly impacting diagnosis, treatment planning, and monitoring of disease progression. This thesis focuses on using a U-Net-based approach for brain tumor segmentation. It begins with an overview of essential methodologies in medical imaging, discussing the latest advancements and challenges in the field. The exploration then delves into the theoretical basis of Convolutional Neural Networks (CNNs) and their use in image segmentation tasks. The core of the thesis involves experimental work, comparing 2D and 3D U-Net architectures to provide insights into the trade-offs between accuracy and computational cost in these architectures.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter is organized as follows: Section 1.1 provides a general overview of the context of the candidate's chosen topic, in Section 1.2 the related state of the art is summarized, and finally, in Section 1.3 the contributions of the proposed work are outlined together with an indication on how this thesis is structured.

1.1 Aims and scope

The human brain has a very complex anatomy with unique properties that distinguish it from the anatomy of all other organs in the human body [1]. It is composed of three major tissues: cerebrospinal fluid (CSF), white matter (WM), and gray matter (GM). The latter, that regulates brain activity, is composed of neurons and glial cells [2].

Brain tumors result from an uncontrolled and rapid cell proliferation and can significantly affect essential human functions such as movement, speech, and vision, especially when located in critical areas.

Among the most frequent types of brain tumors are gliomas which take this name from the fact that they arise from glial cells. These tumors can be classified into two categories based on their degree of aggressiveness: low-grade (LGG) and high-grade (HGG), with glioblastoma being the most common and malignant form of HGG [3].

Gliomas often present different degrees of malignancy in different regions with a high variability of cell structures within the same tumor. This becomes a problem when surgical biopsies are used with prognosis intents. There is, indeed, a risk of sampling a less malignant part of the tumor leading to underestimating the tumor grade and leading to inappropriate treatment decisions. Advanced medical imaging techniques such as multi-parametric magnetic resonance imaging (mpMRI) scans or employing more than one image modalities with varying contrasts, help overcome this issue by better reflecting the varying biological properties of a tumor. This surely improves diagnostic accuracy and possibly increases the chances of success of subsequent treatment plans.

Another big problem with gliomas, especially with glioblastomas, is their tendency to infiltrate surrounding brain tissue, making it challenging to delineate the exact boundaries of the tumor. This infiltration complicates tumor removal, increasing

the probability of residual cancerous cells likely causing tumor recurrences [1]. Furthermore, gliomas tend to grow rapidly and aggressively.

For all these presented reasons, early and precise segmentation is crucial for obtaining the necessary information to develop a specifically customized therapy plan, whether it is for surgical operations or for chemotherapy or radiotherapy treatments. This is also vital for accurately monitoring tumor responses to treatments and detecting any sign of recurrence as early as possible in the therapy.

However, segmentation of brain tumors in multi-modal MRI scans remains one of the most challenging tasks in medical image analysis given the high heterogeneity of tumors [4] in terms of location, extent, and shape, usually characterized by complex and fuzzy borders. Manual segmentation is possible only to a specialist with a thorough understanding of brain tumors, even though analyzing and delineating tumor boundaries, incorporating four different MRI modalities (T1, T1ce, T2, and FLAIR), for the same patient turns out to be a difficult, time-consuming, and error-prone task, even for well-trained experts [5].

Therefore, automatic segmentation techniques are a must if the objective is to speed up and enhance the diagnosis of brain tumors, and the proposed work aims to address this big challenge through the exploration of some of these techniques.

1.2 State of the art

1.2.1 BraTS Challenge

Brain tumor segmentation aims to accurately delineate the areas of brain tumors [2]. This task involves identifying cancerous brain tissues and labelling various tumor sub-regions. Due to the clinical significance of such task and its challenging nature, particularly when concerning gliomas, there has been considerable attention on automatic brain tumor segmentation. This has led to a growing body of related publications and literature over the past few decades.

Before 2012 and the advent of the Multimodal Brain Tumor Image Segmentation Benchmark (BraTS) and of the international conference on Medical Image Computing and Computer Assisted Interventions (MICCAI), it was difficult to estimate the state-of-the-art in automated brain tumor segmentation. This was primarily because comparing the different methods was challenging, given that they were trained on different datasets, often not made publicly available, and their performance was validated using varying metrics. In this context, the need for benchmarks in the field of medical image analysis became evident. Benchmarks allow different groups to optimize their own methods on a shared training dataset provided by the organizers and then apply them on a common test dataset [6].

This is exactly the foundation on which the BraTS challenge was first established in 2012 and since then it has been used, as part of the MICCAI conference, to assess state-of-the-art machine learning methods for brain tumor image analysis in multi-parametric magnetic resonance imaging (mpMRI) scans [4]. Moreover, BraTS

[1] PubMed is a free online database primarily comprising life science and biomedical related literature. It is maintained by the United States National Library of Medicine (NLM) at the National Institutes of Health (NIH).

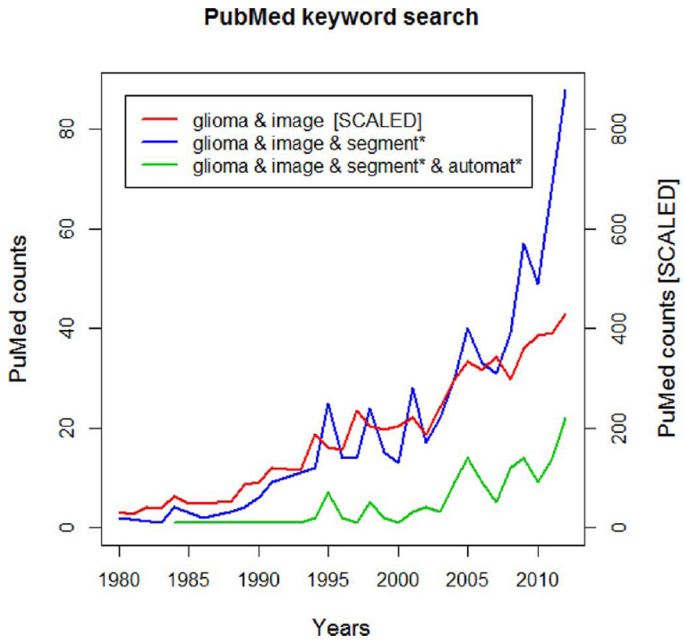

Figure 1.1 Results of PubMed[1]searches related to glioma imaging (red), tumor image segmentation (blue), and automated tumor segmentation (blue) up to 2012, that is, before the advent of the BraTS challenge. Figure taken from [6].

datasets have become a standard in the field, being also updated over the years, and BraTS2021 is exactly what we are going to be using in this work (Section 2.3).

1.2.2 Related works

Reviewing the evolution of methods for brain tumor segmentation over the past years, driven by challenges such as BraTS, reveals the significant advances in various types of architectures, including CNNs, U-Nets, transformers, and many hybrid models.

Today, many researchers focus more on optimizing these models, addressing computational limitations, and trying to make the most out of the potential of multi-modality MRI data to improve segmentation accuracy even further, such that in recent years, deep learning models, particularly Convolutional Neural Networks (CNNs) and U-Net architectures, have significantly advanced the state-of-the-art in the domain of brain tumor segmentation.

Around 2013 and 2014, initial applications of CNNs to medical imaging began appearing in research papers, following their success in other computer vision tasks. This tendency significantly influenced submissions to the BraTS challenges during those years and continued to do so in subsequent years [5], marking the shift from traditional machine learning models, such as Random Forest based methods that were prevalent in earlier years of the challenge, to deep learning. Since then, CNNs have been the foundation for many brain tumor segmentation models due to their ability to learn hierarchical features, from low-level features like edges and textures to high-level features such as shapes and structures in images, and because of the use of convolutional filters that enable the network to detect features irrespective of

their position in the image. This last property is essential for tumor segmentation, where the position of tumors can vary significantly.

Early models focused on standard CNN architectures, such as the one proposed in [7]. These were later enhanced with deeper and more complex structures to improve accuracy and robustness. One noteworthy example is DeepMedic [8], proposed in the 2016 challenge, an 11-layer 3D CNN with residual connections.

However, while deep learning (DL) algorithms like CNN can achieve high precision, they typically require large training datasets, whose collection can be costly and time-consuming, especially in the medical field. Various researchers have developed various DL architectures, primarily based on CNN structures, employing different techniques to address the mentioned limitation while ensuring high accuracy and efficiency of medical image segmentation. One example is the U-Net architecture [9], introduced by Ronneberger et al., which is designed to be efficient with data and has become a cornerstone in medical image segmentation. It consists of an encoder (downsampling path), similar to a traditional CNN, which reduces the image size to capture fine-grained context features, and a decoder (upsampling path), which restores the size to recover spatial information, enabling precise localization of the segmentation mask. During the upsampling process, the reconstruction of the spatial details is achieved through the skip connections in the network that link the feature maps from the encoder to those in the decoder. A similar result could also be obtained following the approach proposed by Chang et al. [10], which reported a fully convolutional neural network (FNN) with residual connections that, similar to skip connection, allow both low- and high-level feature maps to contribute to the final segmentation.

One difference between a U-Net and a traditional CNN is that while the latter generally requires large amounts of training data to perform well, the U-Net is designed to be efficient with data, meaning that it can achieve good performance even with a relatively small amount of annotated data, which is often the case in medical imaging.

The introduction of U-Net marked a significant advancement in medical image segmentation such that it has been widely adopted and modified for various segmentation tasks, for example, with the introduction of a three-dimensional (3D) version of the U-Net to extract information directly from the original volumetric data [11; 12; 13].

Subsequent BraTS challenges saw extensive use of U-Net, or 3D U-Net, and their variants. For example, the winning model of BraTS2018 employed an asymmetric encoder-decoder architecture with a variational auto-encoder branch [14].

In addition to these architectural advancements, ensemble models, which combine predictions from multiple models, have also emerged as a powerful technique to enhance the stability and accuracy of model predictions and mitigate the models' dependency on the choice of their hyperparameters. For example, the top-ranked model in BraTS 2017 was an ensemble of multiple architectures, including DeepMedic [8] and U-Net [9], with the final output obtained by averaging the class-confidence scores of the individual models for each voxel. Similarly, Sun et al. [15] developed an ensemble of deep learning-based models using the BraTS 2018 dataset, achieving considerable segmentation accuracy. Their ensemble comprised three previously successful models, but they used majority voting to ensemble these independently

trained networks.

The second top-ranked model in the BraTS 2017 challenge was a cascaded model proposed by Wang et al. [16]. Cascaded models use a sequential training approach where each model in the sequence takes the output of the previous one as input. Wang et al.'s aim was to break down the multiclass segmentation task into a sequence of hierarchical binary segmentation tasks. Essentially, the model consisted of a cascade of fully convolutional networks (FCNs), each focused on segmenting a specific tumor region (whole tumor, tumor core, and enhanced tumor core in order) and each taking as input only the region extracted by the previous one. This approach highlights the capacity of cascaded models to contribute to robustness and precision in segmentation tasks by reducing the possibility of false positives in the prediction, as the segmented regions are forced to be hierarchically contained within one another [5], as shown in Figure 1.2.

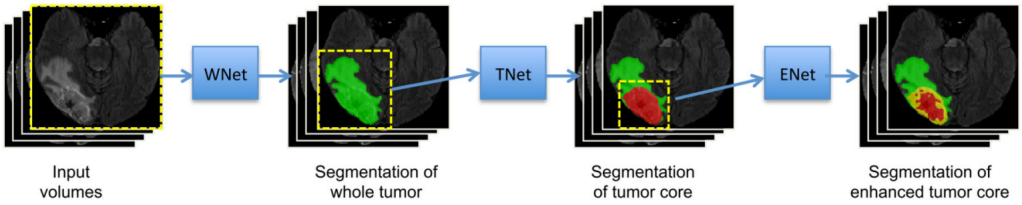

Figure 1.2 Framework proposed by [16] with triple cascaded CNNs for multi-modal brain tumor segmentation.

The success of transformers in image processing has also encouraged researchers to explore their potential in computer vision tasks, particularly in medical image segmentation. Vision Transformers (ViT) [17], which use a mechanism called Attention [18], have shown a remarkable capacity to retain the global context of the subject at hand that is useful for capturing long-distance dependencies in segmentation tasks and to avoid the loss of spatial information.

However, transformers generally require larger amounts of data and computational resources compared to CNNs to reach optimal performance [19]. For this reason, there has been increasing use of hybrid models combining transformers and CNNs, which have demonstrated promising results by capturing both local and global features, thereby enhancing segmentation performance while maintaining computational efficiency. One first example of integrating transformer blocks into the U-Net structure for medical image segmentation is the TransUNet, proposed by Chen et al. [20]. These hybrid approaches typically use pre-trained transformers as encoders to extract relevant features, which are then processed by the decoder for final segmentation output.

Many questions remain regarding the optimal architecture between traditional CNNs and more advanced deep learning structures like transformers in the medical image segmentation task [19]. However, some studies explore the potential that CNN methods hold to outperform ViT-based architectures in image processing tasks [21] and how they can still yield substantial improvements in efficiency and accuracy, continuing to hold value in biomedical imaging [22].

Still, recently, there have been proposals that incorporate the powerful attention

mechanism, first introduced with transformers, into the standard UNet-based models, enabling the network to prioritize important features and to focus on the region of interest, making it more adept at handling complex medical images (e.g., SA-Net [23]).

The U-Net model, which remains a landmark in medical image segmentation, has undergone numerous modifications to enhance its performance, leading to the development of many U-Net variants for brain tumor segmentation and medical image segmentation in general. Table 1.1 shows some notable ones in chronological order. However, despite these variants, the technological core of the UNet, based on the encoder-decoder structure with information flowing from the downsampling to the upsampling path, has seen only modest innovations [19]. Proof of this comes from the proposal of the no-new-U-Net (nnU-Net) by Isensee et al. [24], which has won competitions in various segmentation tasks, years after the original U-Net was first introduced, with a slightly modified U-Net by focusing more on components outside the model architecture itself, such as data preprocessing and hyperparameter tuning. The nnU-Net performance underscores the interconnectedness of network structure, preprocessing, and training stages and their combined impact on segmentation performance, highlighting the simplicity and efficiency of the U-Net architecture while still achieving excellent results on various medical image segmentation tasks. This success supports the theory that "a well-trained U-net is hard to beat" [25].

REFERENCE	ARCHITECTURE	YEAR
[26]	Attention U-Net	2018
[27]	R2U-Net	2018
[28]	U-Net++	2018
[29]	Dense-U-Net	2019
[30]	U-GAN	2019
[31]	Residual U-Net	2019
[32]	DENSE-INception U-net	2020
[33]	BU-Net	2020
[34]	Optimized U-Net	2021
[35]	D-UNet	2021

Table 1.1 Relevant Variants of U-Net Architecture

1.3 Contributions and thesis outline

In this work, a U-Net-based approach for brain tumor segmentation will be implemented. We will explore both 2D and 3D U-Net architectures, as each approach has

demonstrated potential in medical image segmentation, even though a comprehensive comparison and benchmarking of these methods for auto-segmentation of brain MRIs is lacking. Therefore, we will evaluate the performance of both 2D and 3D U-Nets on this specific task. Additionally, various modifications to the original U-Net architecture will be experimented with to enhance performance as much as possible. These modifications will include adjustments to the hyperparameters, incorporation of attention mechanisms, and application of advanced training techniques, all aimed at improving segmentation accuracy and robustness.

The rest of this thesis is structured as follows. Chapter 2 provides an in-depth examination of the data used in this work, focusing on MRI modalities together with tumor regions, and the specifics of the BraTS dataset. Chapter 3 delves into the basics of Convolutional Neural Networks (CNNs) and the U-Net architecture. Chapter 4 presents the experiments of this thesis, discussing the design choices made during the implementation followed by the actual implementation details for various approaches adopted. A comparison between the different approaches is also made in the same chapter.

Chapter 2

Understanding the data

In this chapter, we will delve into the specifics of the data utilized throughout this work, which is necessary for the subsequent model development. In Section 2.1, we will explore the various MRI modalities used in brain tumor imaging and how they provide unique insights into the tissue characteristics of the specific regions of tumors described in this same section. Section 2.2 will focus on the annotation protocol adopted by the BraTS challenge. Finally, in Section 2.3, we will provide an in-depth overview of the BraTS dataset itself, setting the stage for its application in the subsequent chapters of this work.

2.1 MRI modalities and tumor regions

The key diagnostic tool for brain tumor analysis is magnetic resonance imaging (MRI) which is a non-invasive technique for medical imaging. Unlike other techniques such as computed tomography (CT) or positron emission tomography (PET) scans, it does not involve exposure to ionizing radiation (IR) which is by now known to be damaging to the human body, especially to the DNA possibly causing a variety of human diseases, including cancer [36; 37].

MRI exploits the magnetic properties of human tissue[1] to visualize contrasts and to distinguish between the various brain structures [38]. It is based on the physical phenomenon of nuclear magnetic resonance (NMR) in which nuclei are placed in a strong constant magnetic field, such that they align (or magnetize), and then perturbed with a radio frequency (RF) pulse. This energy is absorbed by the nuclei disrupting their alignment. After a while, when the RF pulse is turned off, the nuclei return to their equilibrium state (alignment) going through various relaxation processes in which they emit RF energy. The latter is measured and used to create images: differences in the signals from the nuclei of different tissue types result in varying shades and intensities of gray in the images [39; 40], allowing for clear distinction between different tissues and improved visualization of internal structures within the body, in our case within human brain.

[1] Specifically, here we refer to the properties of atoms of elements present in large quantities within the human body. The latter is primarily composed of water and fat, both containing many hydrogen atoms. Hydrogen nuclei (protons) produce a strong NMR signal, making them ideal for clinical MRI measurements.

Over the years numerous RF pulse sequences have been developed for MRI to create different types of images in terms of brightness and contrast, increasing MRI versatility in highlighting more specific tissues and abnormalities (e.g. fluids, fat or tumors) [40].

Some common MRI sequences include:

- **T1[2]-weighted imaging**. Ideal for visualizing normal brain anatomy since fat appears bright (enhanced signal) while water or fluid-filled areas, like cerebral ventricles containing cerebrospinal fluid (CSF)[3], appear dark (suppressed signal).

- **T2-weighted imaging**. Particularly sensitive to fluid-containing structures that appear bright on the image. It is particularly useful when increased extracellular fluid is index of inflammation and other abnormalities including tumor edema. T2-weighted images can be easily distinguished from T1-weighted images just by the appearance of CSF (Figure 2.1).

- **Fluid Attenuated Inversion Recovery (FLAIR) imaging**. Similar to the T2-weighted imaging, but while abnormalities remain bright, the CSF fluid signal is attenuated, allowing for a better distinction of abnormalities near CSF.

- **Contrast-enhanced T1-weighted imaging**. A variation of the previously presented T1-weighted imaging. It involves the use of a contrast agent, typically a gadolinium-based one, that is injected into the patient's bloodstream before the MRI scan. Gadolinium is a heavy metal able to change signal intensities, enhancing the MRI signal in areas where it accumulates, highlighting vessels and any degradation of the blood-brain barrier (BBB)[4].

Other commonly used sequences include Diffusion-Weighted Imaging (DWI) or Gradient Echo (GRE) Imaging. However, the sequences mentioned above are the ones relevant to the BraTS2021 dataset.

The importance of the existence of different types of MRI sequences lies in their ability to provide a better understanding of abnormalities and distinguish between various tumor sub-regions, which is crucial for tailored clinical decision-making [42].

Gliomas are highly diverse tumors, exhibiting both inter-tumor heterogeneity (differences between tumors) and intra-tumor heterogeneity (differences within a single tumor) due to their infiltrative nature. This diversity, especially within a single tumor, makes choosing an effective treatment a challenging task because different parts of the tumor may respond differently to the same therapy. By identifying

[2]The time required for the atom to return to its original state during relaxation is called Relaxation Time (RT). The two types are: 1) longitudinal (T1) and 2) transverse (T2). The brightness and contrast of the acquired images are given according to the properties of tissues with respect to T1 and T2.

[3]Cerebrospinal fluid (CSF) is a colorless body fluid produced in brain's ventricles (hollow cavities). It flows in the four cavities, around the brain and in spinal cord.

[4]The BBB is a selective, semi-permeable membrane that regulates the movement of molecules and ions between the blood and the brain. Tumors can disrupt the BBB, increasing its permeability [41]. This disruption is exploited in diagnostic imaging using contrast agents, which leak into the brain tissue through the compromised BBB, highlighting the tumor during scans.

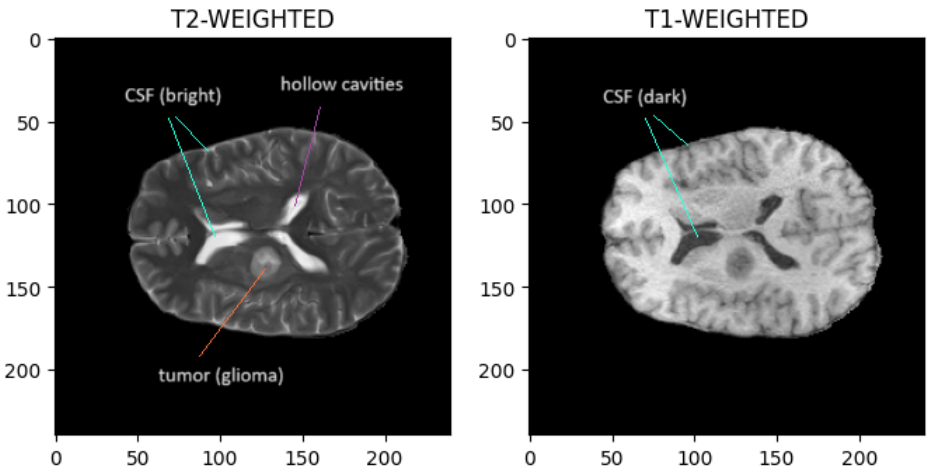

Figure 2.1 CSF appears bright on T2-weighted images and dark on T1-weighted images. Note that this sample has been selected from the BraTS2021 (Section 2.3) training set.

different tumor regions and their extent, clinicians can adapt treatments to target each region more effectively. This distinction between regions depends, of course, on their specific biomolecular features and micro-environments. However, even just by looking at the MRI scans, it is possible to identify some differences. For example, in the contrast-enhanced scan in Figure 2.2, it is possible to distinguish hyper-intense regions from hypo-intense ones within the same tumor. This brings attention to a well-known issue: the radiologic definition and delineation of tumor boundaries is a complex and still debated topic. In this work, we refer, once again, to the standardization introduced by the BraTS initiative which, after consultation with expert neuro-radiologists [4], defined four tumor sub-regions:

1. **(Peritumoral) Edema/Invasion**. This refers to the swelling caused by the accumulation of fluid in the tissue surrounding a tumor, a typical feature of brain tumors, especially gliomas. It is generally caused by the tumor growing and influencing normal fluid flow, possibly causing leakage. Edema, often an indicator of tumor spread [43], can interfere with blood supply and oxygenation to healthy brain tissue, making delineation of its extent crucial for treatment selection and planning [44]. This sub-region is easily visible as a hyper-intense signal on T2-weighted and FLAIR images, and as hypo-intense on T1-weighted images.

2. **Tumor Core**. Best visualized on T2-weighted imaging, it appears as a lower intensity signal and with a more heterogeneous[5] texture, characteristics that make it distinguishable from the surrounding edema. This region is what it is usually defined as the "gross abnormality" and can include both enhancing regions and necrotic/cystic components.

[5]Heterogeneity here is caused by the varying cellular composition and differences in blood supply within the tumor core. The presence of both necrotic areas and cysts, which are a mix of fluid and dead cell-filled cavities, makes the intensities in the T2-weighted image highly variable.

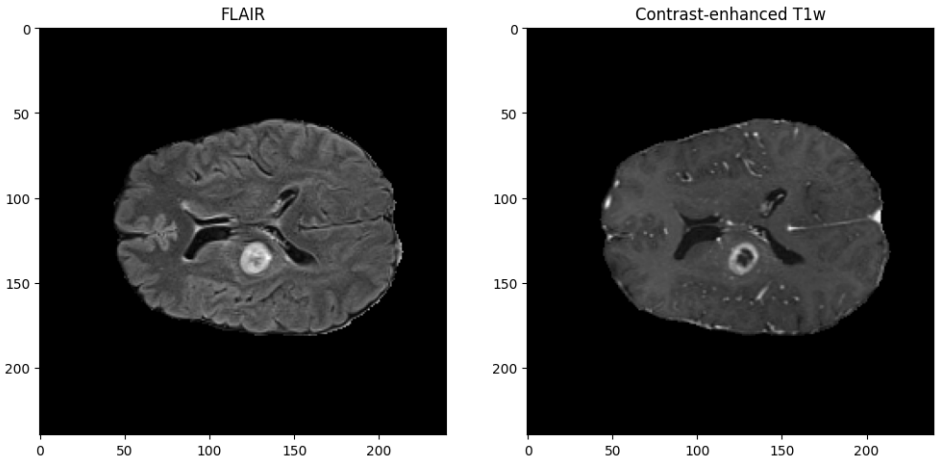

Figure 2.2 FLAIR and contrast-enhanced MRI scans acquired from the same patient of Figure 2.1. The FLAIR imaging modality clearly distinguishes abnormalities, such as the tumor in this example.

3. **Active Tumor**. This refers to the enhancing regions within the gross tumor abnormality excluding the necrotic core.

4. **Necrotic Core (or Necrocyst)**. It usually corresponds to areas within a tumor where cells are dead. In contrast-enhanced T1-weighted images, it is fairly easy to distinguish the enhancing regions from these necrotic components. In fact, enhancing tumor regions are areas where tumor cells accumulate and retain the injected contrast agent, indicating an increased permeability of blood vessels within the tumor and resulting in a hyper-intensity region. Oppositely, non-enhancing/necrotic tumor regions lack any blood flow and, for this reason, appear hypo-intense.

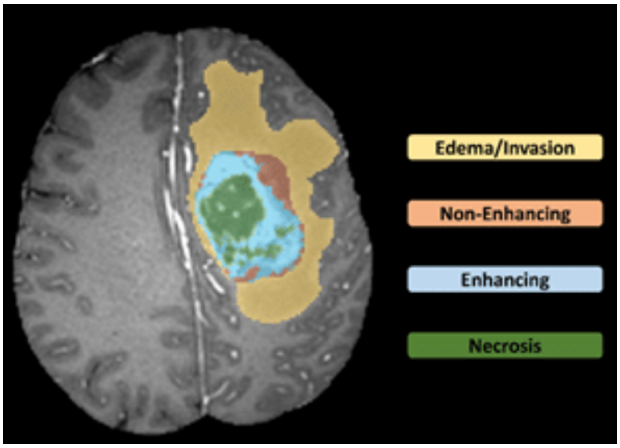

Figure 2.3 The four glioma subregions

2.2 BraTS annotation protocol

The just introduced tumor sub-regions, identified by the BraTS initiative and used as labels in the dataset, are mainly based on imaging. For this reason, they may not directly correspond to specific biological entities, although their appearance on certain scans likely reflects their molecular composition. To overcome any possible confusion during annotation and to standardize the annotation process for datasets from 2017 onwards, BraTS introduced an ulterior division of three mutually inclusive tumor regions [4]:

- ET (Enhancing Tumor);
- TC (Tumor Core): consists of the ET and the necrotic tumor core (NCR);
- WT (Whole Tumor): consists of the edema (ED) and the TC.

Standardization was necessary mainly because the brain tumor MRI scans were collected from multiple centers under standard clinical conditions but still using different equipment and imaging protocols. This variability could lead to heterogeneous image quality and consequentially to varying segmentations of the same tumor regions.

To address this, BraTS developed a tumor annotation protocol for manual segmentation by experts working on the dataset. According to this protocol, one should start by loading the FLAIR images and delineating the boundaries of the WT. Then, using T2-weighted images, the TC boundaries should be outlined (the remaining area of the WT will correspond to the edema sub-region). Finally, to delineate the ET, contrast-enhanced T1 scans should be used together with the existing TC outline, to distinguish between the high intensity of the active (enhancing) tumor and the low intensity of the necrotic (non-enhancing) core [4].

Putting together these annotated regions it is possible to reidentify the original labels. Refer to Figure 2.4.

2.3 BraTS dataset

The BraTS challenge aims to evaluate state-of-the-art automatic brain tumor segmentation methods by providing a 3D MRI dataset with voxel-wise ground truth labels.

Since its inception in 2012, the BraTS dataset has continually evolved. For this study, our focus lies on the BraTS2021 challenge training dataset[6] (available at [45]), which contains multi-institutional multi-parametric magnetic resonance imaging (mpMRI) scans of glioma from where we extract both training and test sets.

The BraTS2021 challenge included a total of 2040 glioma cases distributed across training, validation, and test sets. Specifically, the training dataset consists of 1251

[6]The BraTS2021 dataset was updated since BraTS2020 to include routine clinically acquired mpMRI scans from institutions that had not previously contributed to BraTS. This has increased the demographic diversity of the patient population represented in the dataset and also contributed to a substantial dataset expansion from 660 cases to more than 2000.

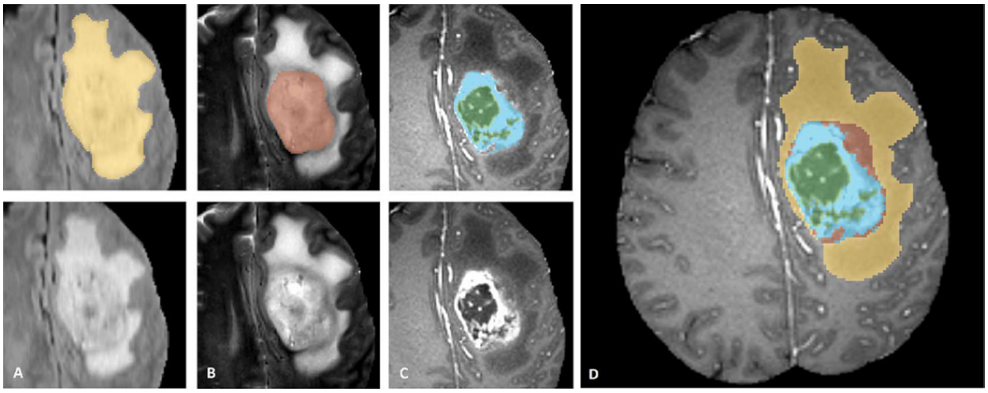

Figure 2.4 The figure illustrates three imaging modalities and the corresponding annotated tumor structures: (A) FLAIR scan, where the whole tumor (yellow) can be delineated; (B) T2-weighted scan, in which the tumor core (red) is distinguishable from edema; and (C) contrast-enhanced T1 scan, highlighting the active tumor (light blue) and differentiating it from the non-enhancing cystic/necrotic core (green). These segmentations are then combined to generate the final labels of the tumor sub-regions (D).
Figure taken from [6].

cases, amounting to 5004 mpMRI scans, given that each subject was scanned in the four modalities: native T1, post-contrast T1, T2, and FLAIR.

After collection, these scans underwent standardized pre-processing routines, as described in [4]. The first step is converting DICOM (Digital Imaging and Communications in Medicine) files to NIfTI (Neuroimaging Informatics Technology Initiative) format. This conversion allows the image data to be stored directly in a 3D format, making it easier and faster to load and process in neuroimaging analysis pipelines. Next, the scans are co-registered to the same anatomical template (to the T1c MRI) to ensure that corresponding anatomical structures are aligned across all modalities and subjects. Following this, the scans are adjusted to have a uniform isotropic resolution of $1mm^3$, such that the voxel dimensions are equal in all three spatial directions. Finally, skull-stripping removes the skull and non-brain structures from the images to keep only brain tissue that must be analyzed.

After completing the standardized pre-processing, the ground truth data was created. The annotation process for BraTS 2021 was facilitated by creating initial segmentations by fusing top-performing automated methods of previous challenges, trained on the BraTS 2020 dataset: DeepMedic [8], DeepSCAN [46], and nn-UNet [24]. Next, neuroradiology experts manually refined the segmentations by looking at the four mpMRI scans along with the automated segmented volume. Two senior neuroradiologists finally reviewed the segmentations and either approved or returned them for further refinement. This iterative process continued until the approval of the segmentations for the challenge [47].

So, in the end, each sample in the dataset contains four modal data and one segmentation label data, all in NIFTI format (.nii.gz).

BraTS 2021 focuses on two tasks, but we are only interested in Task 1: segmentation of heterogeneous brain glioma sub-regions in mpMRI scans. The sub-regions considered for evaluation are the "enhancing tumor" (ET), the "tumor core" (TC),

and the "whole tumor" (WT), as described in Section 2.2, and the provided segmentation labels have values of 1 for NCR, 2 for ED, 4 for ET, and 0 for everything else.

Chapter 3

Basics of CNN and U-Net

In this section, we will first present the Convolutional Neural Network (CNN) architecture that, despite being, nowadays, one of the most commonly known architectures, it is still essential to understand the rationale behind it since it forms the foundation for the architecture of our interest: the U-Net, which, along with some of its variants that will be used in this work, will be described later in this section.

3.1 CNN

A CNN (or ConvNet) is a type of deep learning model; mainly, it is a class of neural networks inspired by the biological vision system, particularly how the human brain processes visual information. Similar to how neurons in the brain's visual cortex respond only to specific tiny regions of the visual field (known as receptive fields) and how they are connected to cover the entire visual field, each neuron in a CNN is responsible for processing data only from a small portion of the input image. These neurons are then arranged in layers, enabling the network to detect simpler patterns, such as edges and lines in the initial layers, and more complex structures as data progresses through the network, ultimately leading to the identification of objects.

Due to these capabilities, CNNs have become widely used in computer vision tasks that require image and video analysis, such as image classification, object detection, and segmentation.

The core building blocks of the CNN are the following:

- *Convolutional layer.* This layer generates a feature map by applying some convolutions defined as:

  Definition 3.1.1 (Convolution)**.** Given a matrix M and another matrix k, called *kernel* or *filter*, the convolution of M by k is the matrix obtained by taking the Hadamard product of k with every submatrix of M of the same size as k.

  Remark. We can also take convolutions with bigger *strides* that are the number of steps we move the kernel before applying the next product.

  Remark. The term convolution refers to both the result function and to the process of computing it.

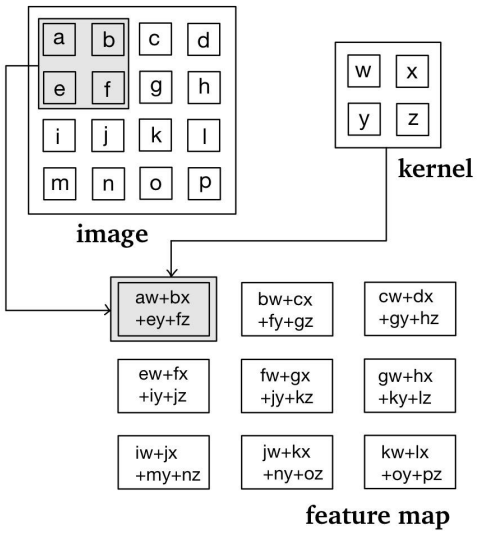

Figure 3.1 Example of how convolution operations are performed.

Typically, a CNN has multiple convolutional layers, such that input images go through a set of convolutional filters, each activating specific features within the images.

- *Non-linearity layer.* The rationale behind this layer, as the name indicates, is to introduce non-linearity into the feature map. The most common activation function to insert in CNNs after each convolutional layer is the **Rectified Linear Unit (ReLU)**, defined mathematically as:

$$\mathrm{ReLU}(x) = \max(0, x) = \begin{cases} x & \text{if } x > 0, \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

  So this function maps negative values to zero while the positive ones are maintained. This can accelerate convergence during training, enabling a faster and improved learning phase.

- *Pooling Layer (or downsampling layer).* This layer is used to reduce the dimensionality of the feature map while retaining important information. This helps reduce the number of parameters the network has to learn, consequentially lowering the computational overhead and avoiding the degradation of computation speed as data advances through the network. The most popular pooling functions are max pooling and average pooling.

- *Fully Connected (FC) Layer.* In a typical CNN for a classification task, this corresponds to the second-to-last layer that outputs a vector with dimension K, where K is the number of possible classes able to be predicted.

- *Final Layer.* It is also referred to as the Softmax Layer since it typically employs the **Softmax function (or softargmax)** defined mathematically as:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{for } i = 1, \ldots, K \tag{3.2}$$

where z is a vector of K real numbers and the output is a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers.

The aforementioned K-dimensional vector, in output from the FC layer, is passed to this final layer to obtain the probability scores for all the available classes for the given input image. These scores specify how likely this image is to belong to each class. The class with the highest probability is then selected as the final prediction for the input image.

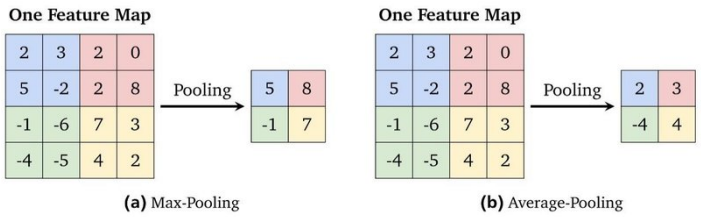

Figure 3.2 Examples of max pooling and average pooling. Figure taken from [48].

For a CNN employed for a segmentation task, the last layers are slightly modified and adapted to the specific need of segmentation. Instead of generating a single vector representing class probabilities, segmentation tasks require pixel-wise classification to produce an output mask highlighting the regions of interest in the image. To this end, the softmax layer is applied to each pixel, and the output is typically a multi-dimensional array where each pixel has a probability distribution over the classes.

Additionally, other layers may be introduced, such as:

- *Upsampling Layers.* These layers (e.g., transposed convolutional layers) are used to upsample the feature maps back to the original image resolution, which means increasing the spatial dimensions of the feature maps back to the original image resolution, enabling pixel-wise predictions in the softmax layer in fact while downsampling through pooling layers helps capture high-level features, it leads to a loss of spatial information, making classifying objects at the pixel level challenging.

At the same time, this pixel-wise classification often fails to capture global context and dependencies among neighboring pixels, resulting in over- or under-segmentation issues.

As a solution to this limitation of CNNs, there are other specialized architectures for image segmentation. One example is the U-Net.

3.2 U-Net

The U-net is a fully convolutional neural network architecture designed primarily for medical image segmentation. It was first introduced in [9]. The basic structure

of a U-net architecture, shown in figure 1, consists of two paths:

- A *contracting path*, also known as the encoder, that is similar to a traditional CNN. This path consists of several convolutional layers followed by max pooling layers, allowing the encoder to extract relevant contextual information from the input image while reducing its spatial dimension.
- An *expanding path*, also known as the decoder, that uses transposed convolutions to capture the higher-level context and localized features, as opposed to the fine-grained details acquired in the encoder. Moreover, the decoder increases the output's resolution by progressively upsampling the feature maps so that the final output can then be used to create a fully segmented image.

A key characteristic of UNet is the use of skip connections between the contracting and the expanding paths. These connections link matching layers from the encoder to the decoder, passing and concatenating lower-level feature maps with higher-level ones. This results in integrating fine-grained details with contextual information, consequently improving the localization accuracy of segmented regions and allowing for precise object boundaries.

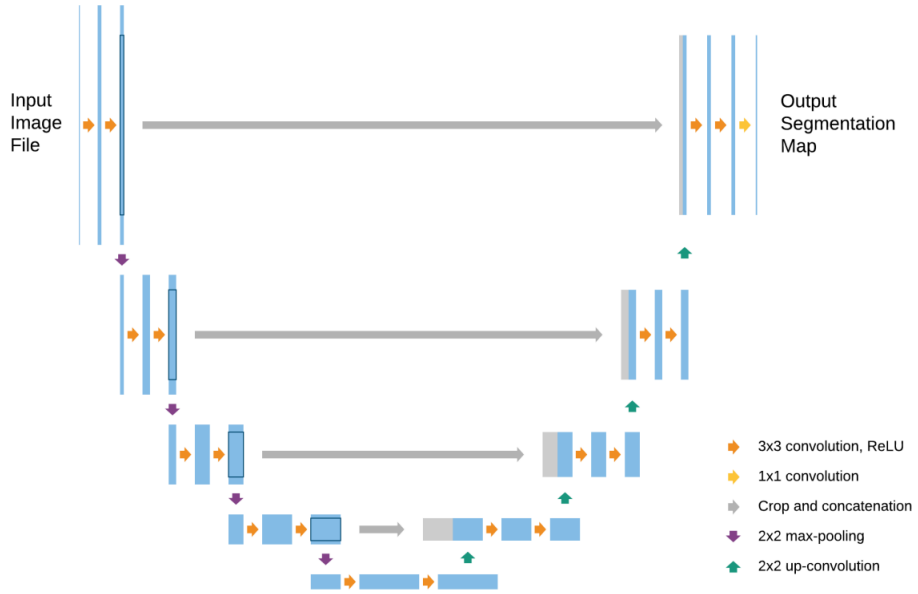

Figure 3.3 The architecture is named after its U-shaped (symmetrical) look when depicted in a diagram. The arrows represent the different operations, the blue boxes represent the feature map at each layer, and the gray boxes represent the cropped feature maps from the contracting path. Figure taken from [49].

In detail, the original proposed U-Net consists of an encoder with repeated convolutional blocks, each composed of two successive 3x3 convolutional layers followed by a ReLU activation unit and a 2×2 max-pooling layer with stride 2 for downsampling. After each block is applied, the number of feature channels is doubled: 64 channels after the first convolutional block, 128 after the second convolutional layer, 256 ls after the third convolutional layer, and 512 after the

fourth convolutional layer. At the bottleneck, the lowest point in the U-Net, there is another convolutional block with no pooling layer, after which the channel dimension is 1024, to process the feature maps at the lowest spatial resolution but with the highest number of channels.

The expanding path, instead, increases the spatial dimensions back to the original input size while decreasing the number of feature channels, from 512 back to 64. Also, this is made of repeated blocks, each composed of a 2x2 up (transposed)-convolution that halves the channel dimension and two convolutional layers with a 3x3 kernel, each followed by a ReLU. The feature map in output from each decoder block is concatenated with the corresponding feature map from the contracting path over the channel dimension.

Finally, a 1x1 convolution is applied to reduce the feature channels to the number of classes required for segmentation, and the softmax function is used to generate the final probability map for each class.

3.3 3D U-Net

3D U-net is an adaptation of the fundamental U-net framework to enable 3D (volumetric) segmentation. The only modification is that all 2D operations are substituted with their respective 3D operations, namely 3D convolutions, 3D max pooling, and 3D up-convolutions. As a result, the encoder-decoder structure will now produce a 3-dimensional segmented image [49].

3.4 Attention U-Net

Attention mechanisms have been incorporated into the U-Net architecture to enhance feature learning and overall performance by focusing on the most relevant features for segmentation tasks. These modifications are referred to as Attention U-Net, where attention mechanisms are integrated into the standard 3D U-Net framework while maintaining its characteristic U-shape.

The attention mechanism can be introduced in various forms, such as attention gates or blocks, and can utilize different types of attention, including spatial and self-attention, to address task-specific needs. Both attention gates and attention blocks work by applying attention coefficients to the feature maps learned during training, attributing varying levels of importance to different parts of this feature map.

Attention gates

In the context of 3D U-Net, attention gates are placed at the skip connections between the encoder and decoder paths. This ensures that only the most relevant features are passed from the contracting path to the expanding path, thereby enhancing the segmentation performance.

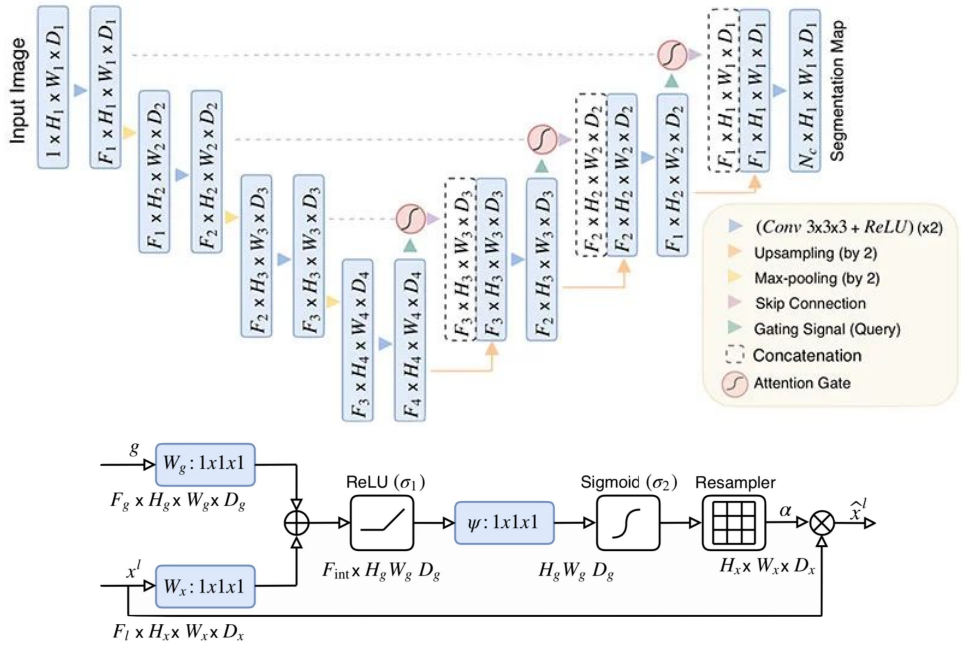

Figure 3.4 The top figure shows the original Attention U-Net architecture proposed in [26]. The architecture presents three attention gates the structure of which is showed in the bottom figure.

Attention blocks

Attention blocks can be introduced within the U-Net architecture at various stages of the network, either within the encoder, the decoder, or at the skip connections. An attention block typically consists of an attention mechanism, such as a spatial or channel attention module [50], which recalibrates the feature maps by focusing on the most "informative" parts.

- The *Self-Attention mechanisms*, also known as intra-attention, compute attention weights by comparing each feature with all other features in the image, regardless of their distance. This allows the network to capture long-range dependencies and contextual information, which can benefit tasks like language modeling, translation, and text summarization. This is why Self-attention is mainly used in transformers for NLP (Natural Language Processing)

- The *Spatial Attention mechanism* focuses on the spatial relationships within the feature maps. It highlights the critical regions in the spatial domain, which is particularly useful for medical image segmentation, where the precise location of abnormalities is crucial.

- The *Channel Attention mechanisms* focus on the inter-channel relationships in the feature maps. By recalibrating the importance of each channel, the network can emphasize the most relevant feature channels for the segmentation task.

Figure 3.5 illustrates the detailed structures of the Channel Attention Module

and the Spatial Attention Module, highlighting the specific layers and operations involved.

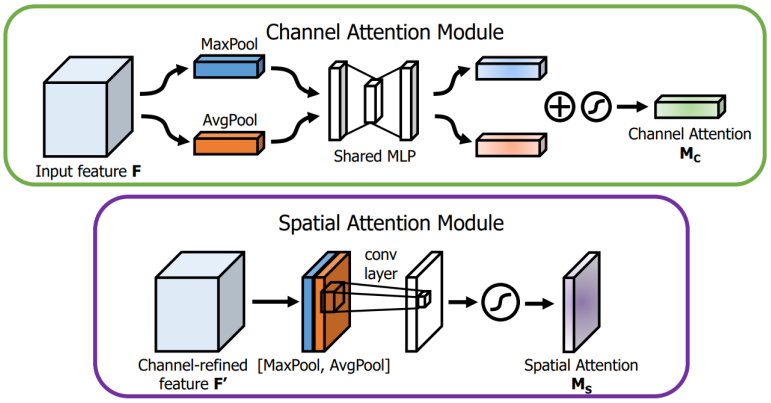

Figure 3.5 Diagram of the Channel Attention Module (top) and the Spatial Attention Module (bottom). The Channel Attention Module uses max pooling and average pooling followed by a shared MLP. The outputs are then concatenated and passed through a sigmoid operation to generate channel attention. The Spatial Attention Module uses a convolutional layer, followed by a sigmoid operation, on concatenated pooled (max and avg) features to generate spatial attention.

Chapter 4

Experiments and Results

While the previous chapters primarily addressed theoretical aspects of the segmentation task we are dealing with, the current chapter's focus shifts towards the actual implementations and the conducted experiments.

4.1 Design Choices

Python was chosen as the programming language for conducting the experiments. Python is an object-oriented programming language that combines ease of use, due to its dynamic typing, with expressiveness. The decisive factor for selecting Python as the programming language for the experiments is its accompanying framework. Known as PyTorch, this framework is among the most widely used for research in machine learning and deep learning. Also influential are the numerous libraries available for machine and deep learning, and for the visualization of complex data like the MRI scans we are dealing with.

4.1.1 PyTorch

PyTorch [51] is an open-source framework originally developed by Facebook's Artificial Intelligence Research group. It was created using Python, C++, and CUDA, and is widely applied in machine learning and deep learning. It is built upon Torch, a Lua-based library which provides a wide range of algorithms for deep learning. Before choosing PyTorch, other solutions were also evaluated, with TensorFlow being prominent among them. The reason why PyTorch was ultimately selected is its strong focus on research applications, in addition to its intuitive and easy-to-learn nature. PyTorch distinguishes itself because of its tensors, multi-dimensional arrays of numbers. Moreover, PyTorch integrates well with Python and this is evident in the similarities between PyTorch tensors and NumPy arrays, and the ease with which they can interact.

PyTorch framework was mainly used to build and train the neural networks. Libraries like *torch* and *torch.nn*, in fact, provide the necessary modules for creating specific layers, performing tensor operations, and other neural network functionalities.

4.1.2 NiBabel

NiBabel [52] is an open-source Python library for brain imaging, providing read and write access to neuroimaging file formats, including NIfTI, CIFTI and ANALYZE, among many others. It allows users to manipulate and analyze neuroimaging data stored in these formats, supporting tasks such as data preprocessing, transformation, visualization, and statistical analysis.

4.2 Dataset

The dataset utilized for both training and validation, as well as for evaluation purposes, is the BraTS2021 dataset, already presented and extensively described in Chapter 2.

4.2.1 Input data preparation

Of course, the first step in our method is preparing the data from the just mentioned dataset and making it suitable for training our model.

The preprocessing process is very straightforward and consists of iterating over each image in the dataset and doing the following passages. Each MRI modality of the image, apart from native T1, is loaded as an array of float64 values with a size of 240x240x155. The reason why we left out T1 is that while it can help define the anatomical structure of the brain, it does not help highlight the tumor regions. For each modality, the intensity values are scaled to the range [0, 1] using Min-Max scaling to ensure consistency across modalities and avoid issues like numerical instability, difficulty in converging, or bias toward features with larger numerical ranges. Min-Max scaling transforms the data to fit within a specified range and is defined by the following formula.

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{4.1}$$

Where X is the original intensity value, $X_{\min}$ is the minimum intensity value in the dataset or volume, $X_{\max}$ is the maximum intensity value in the dataset or volume,, and X' is the resulting scaled value in the range [0, 1].

The three MRI modalities, T2, T1ce, and FLAIR, are combined into a single 4D array of size 240x240x155x3. This creates a multi-channel 3D volume where the fourth dimension represents the channels since each modality will be treated as a channel. The segmentation mask corresponding to the image is also loaded and adjusted. In fact, as anticipated in Section 2.3, the unique values in the mask are [0, 1, 2, 4]. We replace label 4 with label 3. Before this, the mask array values should also be converted from float64 to uint8, as the labels are integers, and storing them as float64 would only be a waste of memory. At this point, the volumes and masks obtained up to now are cropped in all three dimensions (height, width, and depth), typically around the central region of the volume, where there is a higher likelihood of containing tumors, to ensure meaningful volumes and to remove non-informative background regions. This can also help address class imbalance issues typical for medical image datasets, in addition to reducing the computational complexity of

working with a bigger volume. The final size after cropping is 128x128x128x3 for the volumes and 128x128x128 for the masks. Here, it is necessary to crop to a size divisible by 64 to facilitate later extraction of 64x64x64 patches during training.

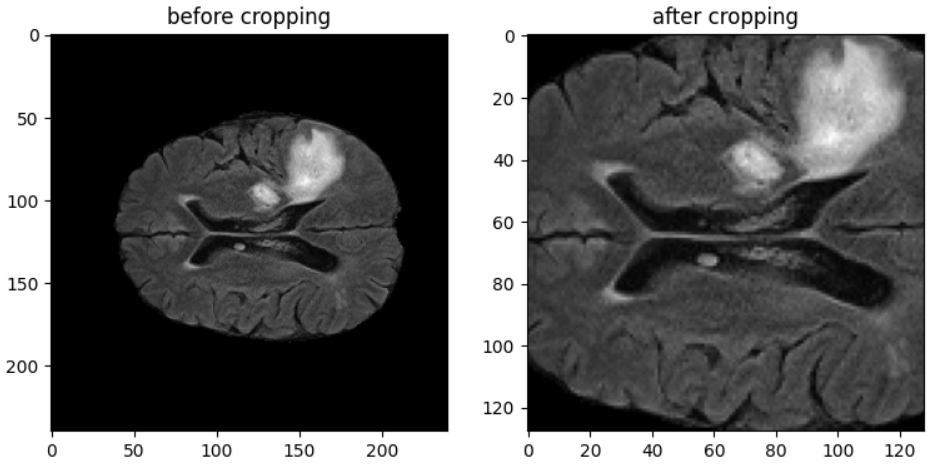

Figure 4.1 Central slice of a MRI volume (in FLAIR modality) before and after cropping.

The mask is checked to ensure it contains at least 1% useful volume with labels other than 0. If this condition is met, the masks are converted to a categorical format using one-hot encoding, resulting in an array of size 128x128x128x4, where 4 represents the number of possible classes, one for each tumor region label plus the background. Contrarily, if the condition is not met, the corresponding sample is discarded from the dataset. The processed volumes and masks are saved as numpy arrays, ready to be displayed using the matplotlib library tools and used in the segmentation process.

As previously mentioned, in addition to 3D models, we have also worked with 2D models, which require some changes in the preprocessing pipeline. For each image, after loading the corresponding modalities, combining them, loading the corresponding mask, adjusting it, and resizing both the volume and the mask, the 155 slices of the volume and the corresponding 155 slices of the mask are extracted one by one and saved as numpy arrays, but not before having converted each mask slice into the categorical format. This process results in a 2D dataset instead of a 3D one. It is worth noting that the 2D dataset will contain a significantly larger number of samples compared to the 3D dataset.

4.2.2 Data generator

Since we experimented with different models that will be presented in the next section (Section 4.3) it is easy to understand that they require different data formats for training. To simplify, we can distinguish these models as 3D models, 2D models, and 3D models trained on patches (rather than the entire volume in the case of 3D). Due to these different requirements, we have created different types of custom data generators.

Full volume generator

A custom generator was developed to handle the loading and processing of large 3D medical imaging data. When the 2D dataset was created, as described before, this same generator was used for that without requiring any adaptation. This generator focuses on loading entire images and their corresponding segmentation masks. The generator class is initialized with directories for images and masks stored as numpy arrays and a list of image and mask filenames retrieved from these directories to easily locate and load the necessary data. When the generator is called to get a sample (*__getitem__* method), it constructs the full paths to the image and mask files using the provided directories and filenames. It then loads the images and masks as numpy arrays, which are subsequently converted into PyTorch tensors, the required input format for the neural network. To streamline the data feeding process, an *imageLoader* function was defined. This function creates a PyTorch DataLoader that wraps around the CustomDataset. The DataLoader iterates over the dataset in batches, shuffling the data to ensure that the model is exposed to diverse examples during training. This shuffling enhances the model's ability to generalize to new, unseen data by preventing it from overfitting to the order of the training data, as there is continuous iteration over the dataset during training.

Patch extraction generator

Another custom generator was created to handle the large 3D volumes more efficiently and extract patches for training. This generator is specifically designed to load images and corresponding masks, extract patches with a specified size and stride, and feed these patches into the neural network. The generator class is, as before, initialized with directories for images and masks stored as numpy arrays, a list of image and mask filenames retrieved from these directories, the desired patch size, which we already mentioned to be 64, and the stride for patch extraction. A stride of 32 was opted for, meaning that there is an overlap of 32 voxels between consecutive patches, given the patch size is 64). This overlap is crucial because, as explained later in Section 4.3, all the patches extracted from a single volume will be fed as a single batch. The overlap allows the model to learn contextual information from neighboring patches, which can improve consistency in segmentation results. So, the number of volumes available, along with the hyperparameters of patch size and stride, defines the length of the effective dataset. When the generator is called to get a sample (*__getitem__* method), it loads an image and its corresponding mask as numpy arrays from the directories. The method then extracts patches from the 3D volumes of both the image and the mask. The extraction is done by iterating over the 3D volume based on the specified patch size and stride. The extracted patches are then converted into PyTorch tensors, to be suitable inputs for the neural network during training.

As illustrated in Figure 4.2, some extracted patches may contain only background pixels. However, the occurrence of such patches is significantly lower than those containing at least 1% of useful volume, meaning not labeled as background (remember that this consideration was a crucial part of our data preparation process for creating the dataset). To achieve this balance, we selected a big enough patch

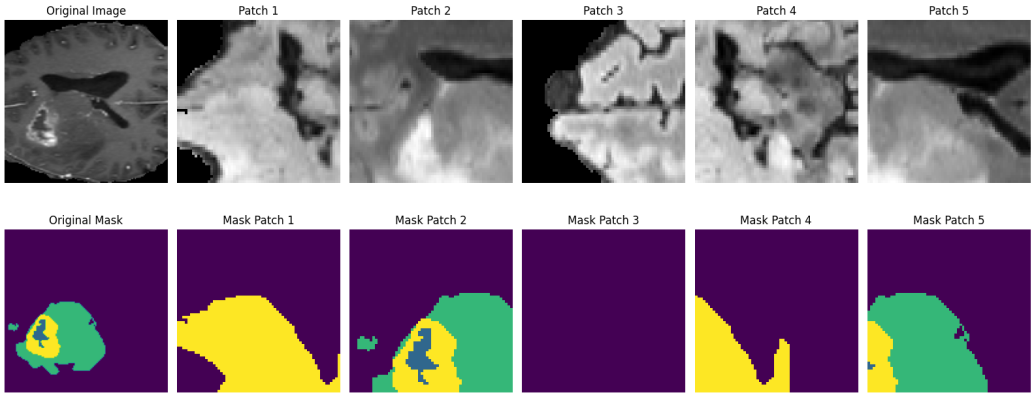

Figure 4.2 This figure illustrates, in the first column, a slice of a MRI volume and its corresponding mask. The subsequent columns display various extracted volume patches (again we are just displaying a significant slice within the entire volume) and their respective mask patches.

size of 64x64x64.

4.3 Implementation details

In the experiments, the 3D U-Net model, described in the previous chapter, was implemented to evaluate its performance. The initial choice of the 3D model stemmed from the hypothesis that it could effectively utilize the complete 3D information present in MRI data, potentially improving segmentation accuracy. Processing data directly in three dimensions was seen as advantageous for capturing spatial dependencies and not losing crucial information, as certain properties may vary depending on direction. Subsequent modifications were made to this base model, leading to the development of three additional architectures, one of which is simply the 2D version of the U-Net. Below is a detailed description of each model.

4.3.1 3D U-Net based approach

This first proposed model follows the fundamental workflow of the U-Net architecture but is specifically adapted to 3D (volumetric) data, such as the original MRI scans we are working with. The base structure includes an encoder-decoder architecture with skip connections.

Encoder

The encoder consists of a series of five 3D convolutional blocks. Each block includes:

- two 3D convolutional layers with a 3x3x3 kernel and a stride of 1x1x1;
- ReLU activation;
- dropout (with 30% probability), added to mitigate overfitting that might happen due to the small dataset we are forced to use given the computational limitations.

The tensor that the network gets as input has three channels (one for each modality), which are expanded to 16 channels after the first convolutional block and doubles after each subsequent block. The sequence of channels is [16, 32, 64, 128, 256].

Downsampling is achieved using a 3D max-pooling layer after each of the encoder's five convolutional blocks. These layers reduce the spatial dimensions of the feature maps, allowing the model to capture hierarchical features at different scales. The kernel size and stride for these max-pooling layers are not fixed but vary between 1x2x2 and 2x2x2 for both the kernel size and the stride at different points in the network to effectively reduce the dimensions to capture finer-grained details while preserving important information.

The outputs of these five convolutional blocks (before the max-pooling operation) are maintained as skip connections to be later concatenated with the decoder features. Before passing to the decoder part of the architecture, there is an additional 3D convolutional block called the bottleneck. This doubles the feature channels from 256 to 512 and is not followed by any max-pooling layer.

Decoder

The decoder consists of a series of five upsampling blocks, making the upsampling path symmetrical to the downsampling path. Each block includes:

- an upsampling operation using trilinear interpolation;
- two 3D convolutional layers;;
- ReLU activation;
- dropout (introduced for the same reason as in the encoder).

The upsampling operation increases the spatial resolution of the feature maps. The five upsampling blocks in the decoder behave symmetrically to the convolutional blocks in the encoder. They decrease the output feature channels at each level from 512 back to 16.

After each upsampling block, concatenation occurs with the corresponding skip connection from the encoder: the first upsampling block's output is concatenated with the last encoder block's output, the second with the second last, and so on. This continues until the network reaches the ends of the U-shaped structure, where the spatial resolution of the final output matches the original input resolution.

After the last concatenation there is the final layer. This is a 3D convolutional layer that outputs a tensor with the same resolution as the input with four channels to represent the one-hot encoded four classes.

4.3.2 3D U-Net with spatial attention

To further enhance the segmentation performance, an attempt was made to innovatively incorporate the spatial attention mechanisms into the 3D U-Net architecture. The primary goal of integrating spatial attention was to improve the network's

ability to focus on relevant features while not giving too much "attention" to others, thereby improving its ability to segment target tumor regions accurately.

The proposed implementation includes a 3D version of the spatial attention mechanism, operating directly on the volumetric data processed by the network. This mechanism computes attention weights based on both average and maximum feature activations across channels and spatial dimensions, facilitating enhanced feature representation and localization. Specifically, average-pooling and max-pooling operations are performed along the channel axis and then concatenated. Subsequently, this pooled feature map goes through a convolutional layer followed by a sigmoid activation, generating a spatial attention map with attention scores that encode the specific regions to emphasize or suppress. This spatial attention module was, in fact, integrated after the final convolutional block in the 3D U-Net architecture immediately before the decoder. Here, the obtained attention map is multiplied by the output of the convolutional block, modulating the input features before their propagation to subsequent decoder layers.

In addition to the 3D spatial attention, the effectiveness of its 2D counterpart within the same framework was also explored. The 2D spatial attention operates on individual slices of the 3D data, applying analogous principles of feature weighting and modulation but in a slice-by-slice manner. However, no significant disparities were visible when comparing the performance gains achieved by 3D versus 2D attention mechanisms in the context of our volumetric segmentation task. Consequently, the 3D spatial attention was only chosen because its implementation is more straightforward within the 3D U-Net framework.

4.3.3 2D U-Net based approach

After experimenting with the 3D U-Net for our segmentation tasks, the 2D version, which is actually the original version of U-Net, was also explored. We maintained the same encoder-decoder structure with skip connections as in the 3D model but with a key change: 2D operations instead of 3D ones. Specifically, 3D convolutions were replaced with 2D convolutions, 3D max-pooling with 2D max-pooling, and trilinear interpolation with bilinear interpolation for the upsampling operation. These changes were necessary because the depth dimension is no longer a concern in the 2D approach, so the stride and kernel sizes are also different. For instance, in the 2D model, the stride and kernel size in the max-pooling layer are always 2x2, unlike in the 3D architecture, where they vary in the depth dimension according to the level in the network.

We decided to try the 2D approach because the 3D version of the U-Net took a long time to train and required more computational resources. As a result, we had to reduce the number of volumes in the training dataset to 100, which increased the risk of overfitting and reduced the model's ability to generalize to unseen data. Another motivation for trying the 2D approach was the need to increase the batch size to achieve more stable and smooth gradients and once again improve the model's generalization capabilities. In the 3D U-Net model, due to computational limitations, we are restricted to a very small batch size, typically around 2, but this problem is mitigated with the 2D dataset.

By slicing the 3D data into 2D data, as already explained in Section 4.2.1, we

could adapt the dataset to the 2D network. This straightforward conversion allowed us to experiment with the advantages of the 2D U-Net architecture.

4.3.4 3D U-Net trained on patches

Similarly to the 2D U-Net approach, training the 3D U-Net on patches instead of the entire volume addresses the primary challenge of data augmentation when adding more 3D volumes to the dataset is not feasible. By working with smaller patches, we can increase the number of training samples from a single given volume. So, this technique helps generate a more varied training dataset from a limited number of volumes, making the model more robust against overfitting, thereby enhancing the model's generalization ability.

In particular, a practical and straightforward approach was adopted: feeding patches extracted from a single volume as a single batch for training the 3D U-Net model. This method can also be beneficial apart from practicality. Extracting multiple patches from a single volume and feeding them as a batch maintains spatial consistency and ensures that the model sees various parts of the same volume during each training iteration, which can lead to the model learning both local and global features. The latter is exactly the original purpose of the U-Net and the main goal in medical imaging analysis. Additionally, this approach allows for better utilization of computational resources. Training on smaller patches requires less memory and enables the use of larger batch sizes, which can result in more stable and smooth gradients.

It is important to note that apart from the notion of batch, the architecture of the 3D U-Net remains unchanged when using patches instead of entire volumes. So, the core model structure and its ability to learn hierarchical features from the data are preserved.

4.4 Evaluation metrics and loss function

Multiple metrics exist in the literature to evaluate the performance of segmentation models, but the most prevalent are the Dice Score and the Hausdorff Distance, which are also used in the BraTS challenge to evaluate submitted models.

- The **Dice Score** (also known as Dice Similarity Coefficient, DSC) is a measure of similarity or overlap between two sets. A score of 1 indicates perfect overlap and, thus, perfect segmentation performance, while a score of 0 indicates no overlap at all. It compares the area, in terms of the set of pixels (or voxels in the case of 3D segmentation), of the ground truth and predicted regions with the total area of both regions. The Dice Score is calculated as follows:

$$\text{Dice} = \frac{2|Y_{\text{true}} \cap Y_{\text{pred}}|}{|Y_{\text{true}}| + |Y_{\text{pred}}| + \epsilon} \tag{4.2}$$

  where Y_{true} is the ground truth, Y_{pred} is the prediction, and ϵ is a small number used to avoid division by zero.

- The **Hausdorff Distance** measures the distance between segmentation boundaries. A smaller Hausdorff Distance indicates better segmentation. It calculates the maximum value among the shortest least squared distances (LSDs) $d(p,t)$ where p is any point belonging to the predicted surface and t is any point on the ground truth surface. Essentially, it is the maximum distance between a point in one set and the nearest point in the other set. The Hausdorff Distance is computed as follows:

$$Haus(P,T) = \max\left\{\sup_{p\in P}\inf_{t\in T} d(p,t), \sup_{t\in T}\inf_{p\in P} d(t,p)\right\} \tag{4.3}$$

  In words, to calculate the Hausdorff Distance:

  1. For each point p in the predicted set, find the shortest distance to any point t in the ground truth set, and then take the maximum among these distances.
  2. For each point t in the ground truth set, find the shortest distance to any point p in the predicted set, and then take the maximum among these distances.
  3. Finally, take the maximum between these two distances.

Of course all the metric computations need to account for the multi-class nature of the segmentation task we are working on. This requires calculating every metric for each class separately and then averaging them.

Initially, the Dice Loss, extended to handle multiple classes, was used as the loss function in the training phase, and the IoU (Intersection over Union) score was used as the accuracy metric to test the model. However, a significant discrepancy was observed between the Dice and IoU scores for the same models. The Dice Score tends to show more sensitivity towards small regions and classes with fewer pixels, making it beneficial for tasks like ours and medical image analysis in general, where segmenting even small areas or less frequent classes is crucial.

The dataset used exhibits high class imbalance, as visible in the pie chart in Figure 4.3, and we aim to ensure that less frequent classes, such as necrotic regions, are not neglected. IoU score, on the other hand, is more useful for applications where the overall accuracy of a larger region is more important than the single regions, but this is not our case.

Therefore, in the end, the Dice Score was selected as the most indicative accuracy metric. It can be calculated class by class separately, as in the BraTS challenge, allowing us to compare our model's performance with those benchmarked by BraTS.

Additionally, metrics such as Specificity, Sensitivity (also known as Recall), Precision, and the F1 score are used to provide a more comprehensive understanding of the model's performance. For example, metrics of Sensitivity (or Recall) and Specificity are employed to determine potential over-segmentation or under-segmentation for each tumor region.

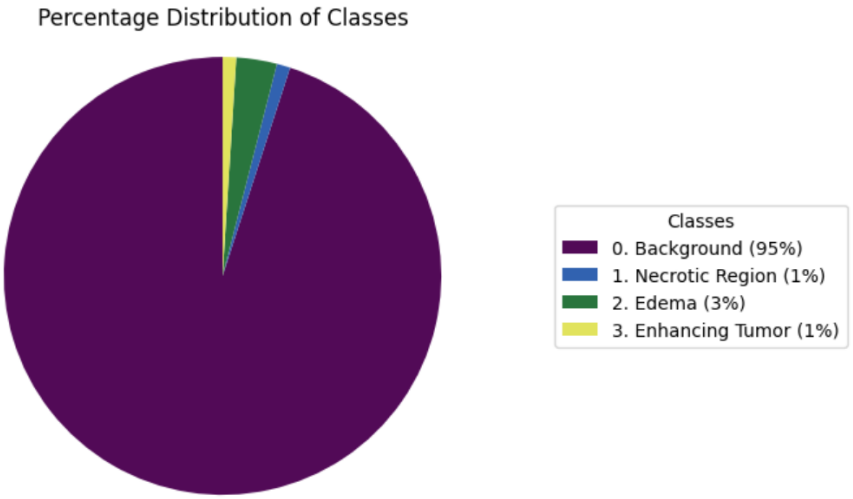

Figure 4.3

4.5 Training step

During the training phase of the experiments, starting from the 3D U-Net model, careful consideration was given to the selection of hyperparameters to optimize model performance.

Number of epochs

Initially, the number of epochs was set at around 25. However, analysis of the training curves (visible in Figure 4.4) revealed a consistent decrease in loss and an increase in accuracy up to epoch 25, suggesting that further training could yield additional improvements. This observation, combined with the lack of performance degradation in the validation set, indicated no signs of overfitting, justifying an increase in the number of epochs. Consequently, the number of epochs was extended first to 50, leading to similar observations as before, and then to 90. While this extension continued to improve the model's performance, the rate of improvement slowed, with fluctuations in the loss and accuracy metrics becoming more pronounced as training progressed in the epochs.

Learning Rate

To address these fluctuations, various learning rate (LR) scheduling strategies were employed. Initially, a multiplicative LR scheduler (MultiplicativeLR) was used, which adjusts the learning rate by a fixed percentage each epoch. Starting with a base learning rate of 0.1, the rate was reduced by 5% per epoch, reaching approximately 0.001 by the final epoch. However, this approach proved to be not optimal as the high initial learning rate caused overshooting, leading to instability and oscillation in the training process. Probably as consequence of this, at epoch 45, the loss remained still stuck around 0.81, indicating no progress being made.

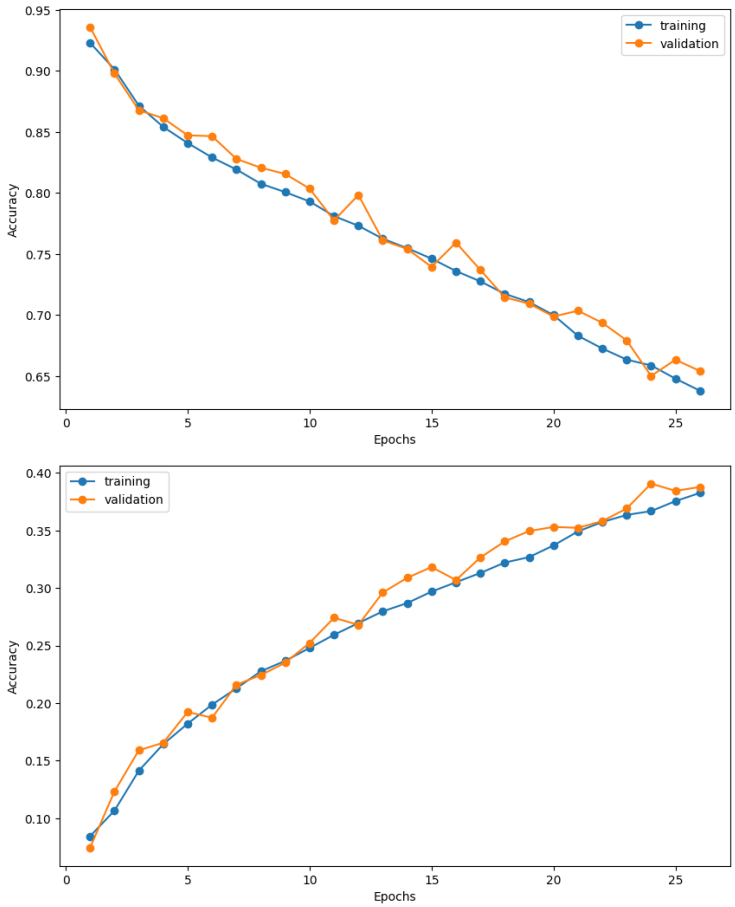

Figure 4.4 The plot on the top shows the loss values for both training and validation datasets as the training progresses over 26 epochs. The plot on the bottom illustrates the accuracy values for both training and validation datasets throughout the training process.

Alternative strategies, including the StepLR scheduler and ReduceLROnPlateau, were also tested. The StepLR scheduler reduces the learning rate by a factor at fixed intervals, in our case, the number of epochs. Although this method provides a systematic reduction, it does not sufficiently address the oscillation issue. ReduceLROnPlateau, contrarily, adjusts the learning rate based on some performance evaluation. When the monitored metric (epoch mean loss in our case) stops improving for a specified number of epochs, called *patience*, the learning rate is reduced. Despite this more adaptive approach, it was ineffective due to the loss still gradually improving although slowly, which did not trigger the learning rate reduction frequently enough or did so too often, according to how the hyperparameters of the scheduler like patience were set, leading us back to the initial problem and failing to provide a consistent improvement. In response, a learning rate warm-up strategy was employed, gradually increasing the learning rate at the start of training to stabilize the initial updates.

Batch size

Additionally, the choice of the batch size was thought to be fundamental to encourage stable and smooth gradients and prevent instability in the training process. Unfortunately, due to limited resources, particularly memory, a batch size of 2 was used. To compensate, gradient accumulation was introduced.

Gradient accumulation

Gradient accumulation is used to simulate a larger batch size while managing memory usage constraints, and it is particularly useful when training on GPUs with limited memory, like in our case. This technique consists in accumulating gradients over multiple forward and backward passes before updating model weights, yielding more stable and smooth gradient updates.

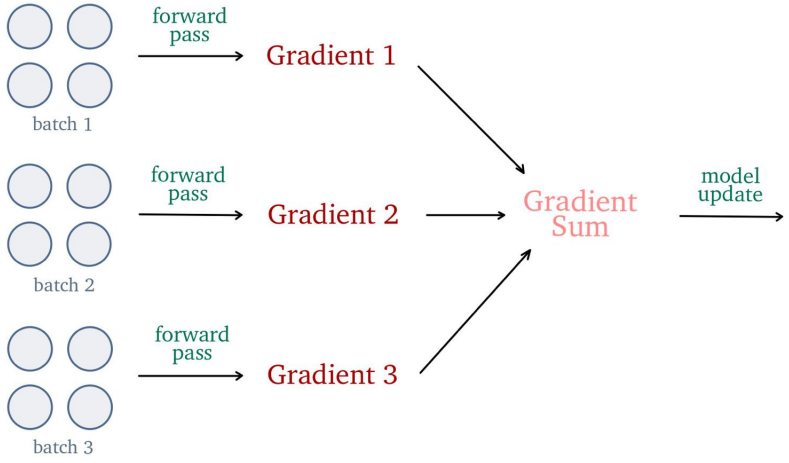

Figure 4.5 In this example the batch size is 4 and the accumulation step is 3 indicating that the model weights are updated only once every 3 batches, obtaining an effective batch size of 12 (=4*3).

The integration of gradient accumulation required a reevaluation of the LR scheduling strategy. Given the improved stability and reduced noise in the training process, it was possible to delay the learning rate reductions, allowing the model to benefit from a higher learning rate for a more extended period during training. This combination of techniques ultimately facilitated more effective training, leading to slightly enhanced model performance.

4.6 3D Vs 2D U-Net

One fundamental experiment in this thesis is a comparison between the 3D version of the U-Net and its 2D counterpart. The need for this comparison arises from a gap in the literature, where there is a lack of comprehensive studies directly comparing the two, despite the widespread adoption of 2D-CNNs more than 3D ones in brain tumor segmentation methods.

A key aspect of this comparison was not only the segmentation performance but also the computational requirements.

3D models and their limitations

The 3D U-Net was the model with which the experimentation started. The reason behind this choice is that ideally, by directly processing the original 3-dimensional volumes, it is possible to capture the spatial dependencies within them, a capacity further increased by introducing spatial attention mechanisms, as we did.

Unfortunately, our 3D models faced some significant challenges during training, leading to lower-than-expected segmentation accuracy. The most evident issue was that processing entire volumes with the 3D model proved to be computationally expensive, making the model training process feasible only with a smaller dataset and batch size than initially planned. The faced computational resource constraints further exacerbated these limitations.

To address these issues, we experimented with training the 3D U-Net (both with and without spatial attention) on patches rather than whole volumes. Specifically, we extracted 27 3D patches from each volume and used these as a batch in training, as described in Section 4.2.1, to increase both the batch size and the effective dataset size. While this approach somewhat mitigated the computational burdens, the results were still unsatisfactory, obtaining only a minimum validation loss of 0.48 after 90 training epochs.

The 3D models, in fact, suffered a lot from the extreme class imbalance present in the dataset, making the training process slow and full of oscillations. This imbalance requires more epochs for the model to learn adequately to segment minority classes, especially when the initial model heavily favors the majority classes. However, the introduction of weights in the loss function significantly improved the overall model performance, decreasing the validation loss to a new minimum of 0.39 after 90 epochs. This was the 3D model that yielded the best results, and for this reason, we used it for comparison with the 2D models.

One important consideration to make here is that we still expect the 3D models to perform better with additional epochs since neither the training loss nor the validation loss stopped decreasing, and only after a long training period could the expected results be achieved. Unfortunately, we did not have the resources to train for more than 90 epochs, which was clearly insufficient.

Comparison between 2D and 3D models

As a solution, the 2D model mitigated many of the issues mentioned above by training on slices.

For example, class imbalance in 2D is less evident as each slice can have a more balanced class distribution than the overall volume.

Across various hyper-parameter configurations, the 2D U-Net was able to obtain better results than 3D U-Net in a way smaller number of training epochs, suggesting that the 2D architecture effectively resolves the computational challenges while delivering fairly good segmentation results. Refer to figure 4.6.

	Metric	Value
0	Dice Score	0.606189
1	IoU Score	0.497137
2	Sensitivity	0.633039
3	Specificity	0.923766
4	Precision	0.684303
5	F1 Score	0.617855

	Metric	Value
0	Dice Score	0.699996
1	IoU Score	0.578461
2	Sensitivity	0.811090
3	Specificity	0.956143
4	Precision	0.645264
5	F1 Score	0.702814

Figure 4.6 These tables show the metrics of the 3D model (on the left) compared to those of the 2D model (on the right). We point out once again that every metric is calculated for each class separately (as a pixel-wise classification) and then averaged.

- The 2D model has a higher dice score, indicating that its predictions are closer to the true labels of the mask compared to the 3D model.
- The 2D model has a higher IoU score, suggesting that it correctly classifies more pixels overall compared to the 3D model.
- The 2D model has a much higher sensitivity (or recall), meaning it identifies a higher proportion of actual pixels belonging to a specific class correctly. This indicates increased ability to identify true positive cases.
- The 2D model has a higher specificity, indicating it is better at correctly identifying actual negatives. This means fewer false positives.
- The 2D model has slightly lower precision than the 3D model. This indicates a slight trade-off between precision and recall. However, in medical imaging segmentation, recall is often more crucial than precision to ensure that no anomalies are missed.
- The 2D model has a higher F1 Score. This suggests once again improved overall performance and a more robust model.

It is crucial to note that both architectures were evaluated on the same validation dataset to ensure a fair comparison. This dataset consisted of 3D volumes, and for the 2D model, this involved creating a 3D mask by generating 2D masks for each slice of a volume and then stacking them together for evaluation.

Initially, there were some doubts associated with this approach. The first concern was that using this method to predict slices one by one might take a lot longer or need more computational resources than directly predicting the entire volume using the 3D model, thereby negating the computational advantages of training the 2D model. The second concern was the potential loss of important 3D contextual information from adjacent slices in the volume data, which could result in discontinuous predictions. Fortunately, both concerns proved to be unfounded. The computational efficiency of the 2D model remained advantageous, and as demonstrated in Figure 4.11, the 2D model successfully preserved the continuity of the predictions.

Limitations of the 2D model

During the testing of the trained 2D U-Net, some limitations impacting the performance and generalization ability of the proposed model still emerged.

Firstly, the model sometimes struggled with handling complex and irregularly shaped masks, leading to inaccuracies in the segmentation outputs. Additionally, artifacts and noise occasionally appeared in the segmented images, necessitating post-processing steps to enhance the final results.

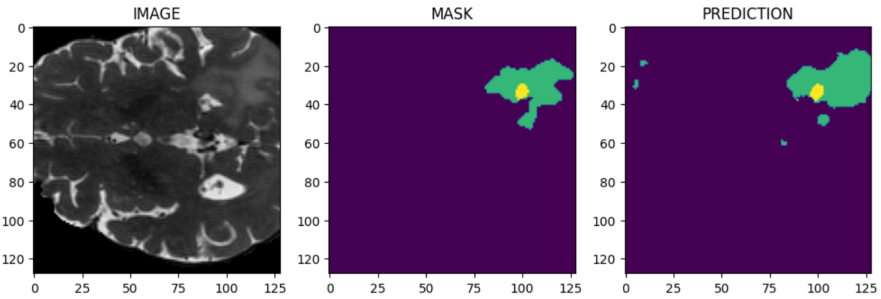

Figure 4.7 Segmentation of a 3D volume with 2D U-Net (only a slice plotted). This shows the noise introduced in the segmentation mask.

Moreover, the models exhibited higher Dice loss in samples where tumor regions were either absent or extremely small, as the background regions were easier to predict. This highlights the need for more robust techniques to handle such scenarios effectively.

Finally, it is not to be excluded that the 3D U-Net, trained for a large enough number of epochs and with adequate computational resources, could eventually outperform the 2D model by fully taking advantage of the 3D information of the MRI data and addressing the limitations encountered with the 2D model.

4.7 Impact of Network Depth

To investigate the importance of the depth of feature extraction in our 2D U-Net model, we conducted an ablation study by simplifying the model's architecture.

The starting 2D U-Net architecture consisted of an encoder-decoder structure with five convolutional blocks and a bottleneck layer, progressively increasing the feature channels from 3 to 512. This configuration allowed the model to capture fine-grained details by downsampling the input image and expanding the feature channels in the following order: [16, 32, 64, 128, 256, 512]. The five upsampling blocks in the decoder are just symmetrical. The performance of this model, as already shown, yielded a loss of 0.3000, an accuracy of 0.5785, a sensitivity of 0.8111, a specificity of 0.9561, a precision of 0.6453, and an F1 score of 0.7028.

For the ablation study, we modified the architecture by reducing the number of convolutional blocks in both the encoder and decoder from five to three, resulting in a shallower network that progressed through the feature channels [16, 32, 64, 128]. This simplified model was also validated on the same dataset as the original one, producing a loss of 0.3972, an accuracy of 0.4825, a sensitivity of 0.8014, a specificity

of 0.9475, a precision of 0.5334, and an F1 score of 0.6058.

	Metric	Value
0	Dice Score	0.699996
1	IoU Score	0.578461
2	Sensitivity	0.811090
3	Specificity	0.956143
4	Precision	0.645264
5	F1 Score	0.702814

	Metric	Value
0	Dice Score	0.602803
1	IoU Score	0.482531
2	Sensitivity	0.801412
3	Specificity	0.947466
4	Precision	0.533361
5	F1 Score	0.605777

Figure 4.8

The results of the ablation study demonstrate the importance of deeper feature extraction layers in the original 2D U-Net model. The reduced depth model, in fact, exhibited a perceptible decline in performance across all evaluated metrics. Specifically, the increase in loss and decrease in accuracy, precision, and F1 score indicate that the simplified architecture was less effective at correctly classifying the pixels in the input images and so at segmenting the input images.

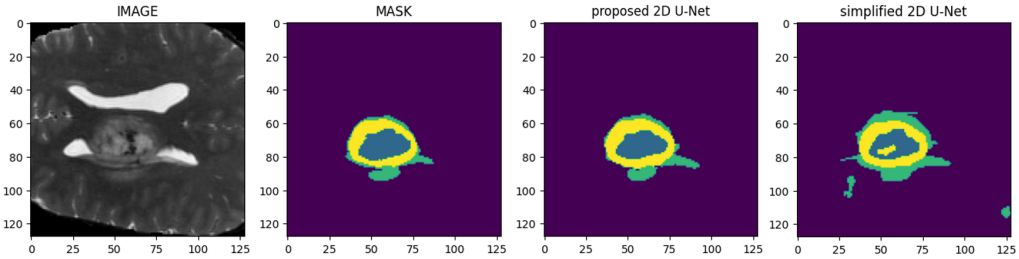

Figure 4.9 The first column shows the original MRI image. The second column displays the ground truth mask of the brain tumor regions. The third column presents the segmentation result from the proposed 2D U-Net architecture. The fourth column shows the segmentation result from a simplified 2D U-Net architecture. This visual comparison highlights the differences in segmentation performance between the proposed and simplified models.

These findings underscore the necessity of a deeper network to achieve high levels of detail and accuracy in the segmentation task, validating our initial design choice of a more complex and deeper architecture.

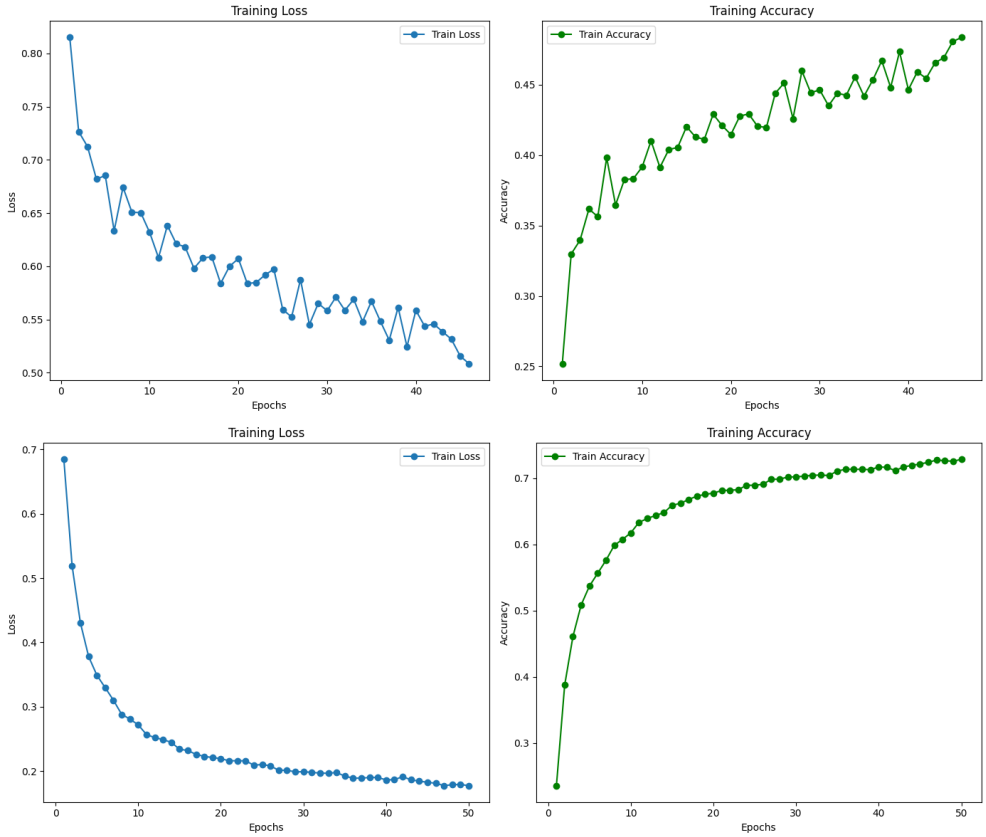

Figure 4.10 This plot compares the training loss and accuracy over epochs for 3D (top row) and 2D (bottom row) models. The 3D model (top left) shows a gradual decrease in loss over 50 epochs, indicating slow but steady learning progress. The 2D model (bottom left) exhibits a rapid drop in loss initially and continues to decline over 50 epochs, suggesting a faster convergence compared to the 3D model. The 3D model (top right) demonstrates a consistent increase in accuracy, reaching around 50% by the end of 50 epochs, suggesting the possibility of better classification capability with more training epochs. The 2D model (bottom right) achieves a higher accuracy more quickly, stabilizing around 70% within 50 epochs, indicating more efficient learning and better performance.

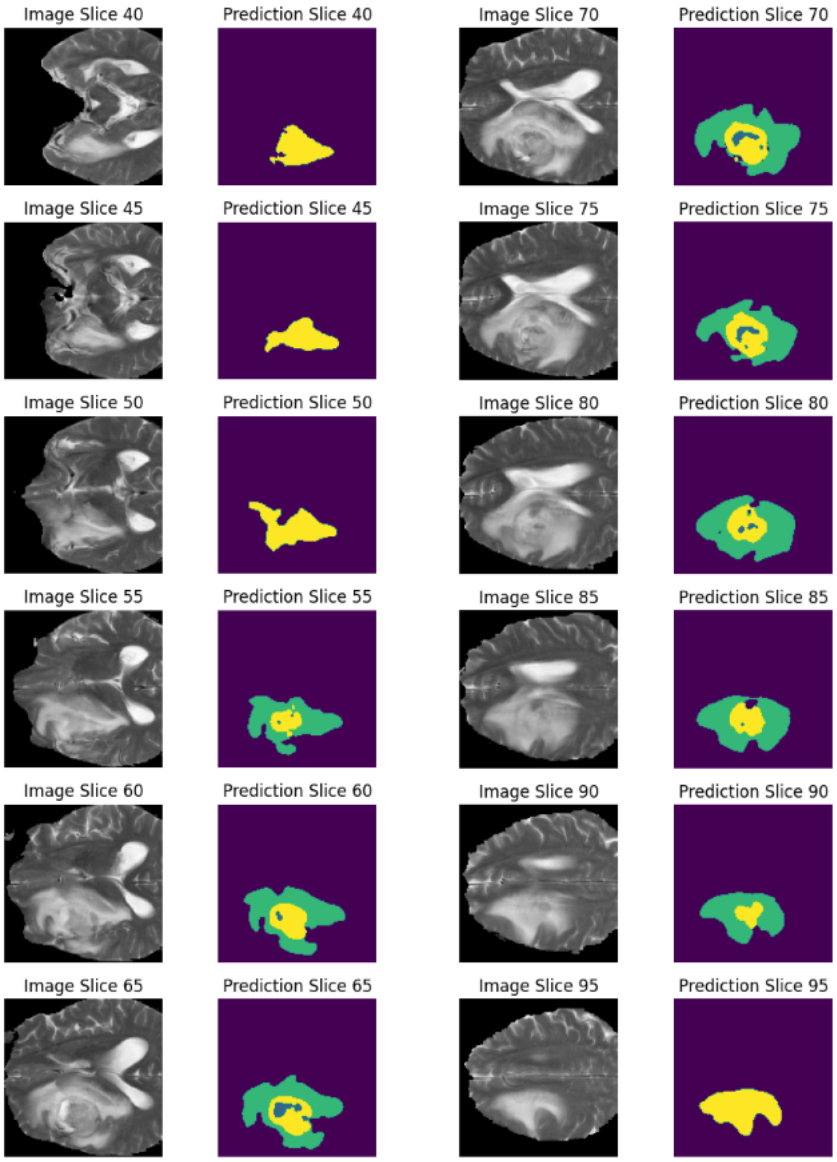

Figure 4.11 Segmentation of a 3D volume with 2D U-Net. This figure provides an overview of the model's performance across the volume. The slices are sampled at regular intervals in the central area where the tumor is located.

Chapter 5

Conclusions

5.1 Summary of findings

In this thesis, two neural networks based on the U-Net architecture were primarily implemented: a 3D U-Net, which was further modified with the introduction of a spatial attention mechanism, and a 2D U-Net. The objective of this work was to achieve accurate segmentation of images extracted from the BraTS2021 dataset, with a particular focus on comparing the performance of the 2D model against the 3D model. This comparison encompassed not only segmentation accuracy but also execution times, to explore the computational efficiency and feasibility of employing 2D models for 3D volume segmentation in the context of brain tumor analysis, providing insights into their practical application and potential advantages in medical imaging.

Unfortunately the computational resources available posed a clear limitation, particularly for the 3D U-Net, as the training and evaluation of this model required significant processing power and memory, limiting the size of the dataset and the batch sizes used during training.

5.2 Future research

Future research should focus on several key areas to address the limitations identified in the models implemented in this thesis. One potential direction is using pre-processing and post-processing techniques to possibly improve the overall quality of the segmented images. Pre-processing steps could include advanced normalization and augmentation techniques to better prepare the data for training. Data augmentation, for example, can be used to artificially balance the dataset when class imbalance is a big problem and slows down training exactly like in our case, but it needs to be applied carefully to avoid introducing bias or noise. Post-processing could, instead, involve techniques such as conditional random fields (CRFs) or other methods to refine the segmentation boundaries after prediction by removing noise and artifacts. False positives in the prediction could be also removed after segmentation, as in brain tumor segmentation specific regions should be contained within specific others.

Moreover, exploring ways to optimize the computational efficiency of the 3D U-Net and similar models will be crucial. This might include investigating more efficient

network architectures, leveraging advanced hardware accelerators, or employing techniques such as model pruning and quantization to reduce the computational burden.

Overall, future research should aim to improve both the accuracy and efficiency of segmentation models in brain tumor MR imaging.

Bibliography

[1] K. Herholz, K.-J. Langen, C. Schiepers, and J. M. Mountz. Brain tumors. *Seminars in Nuclear Medicine*, 42(6):356–370, 2012. doi: 10.1053/j.semnuclmed.2012.06.001.

[2] R. Kaifi. A review of recent advances in brain tumor diagnosis based on ai-based classification. *Diagnostics*, 13(18):3007, 2023. doi: 10.3390/diagnostics13183007.

[3] E. Gonzales-Aloy, A. Ahmed-Cox, M. Tsoli, D. S. Ziegler, and M. Kavallaris. From cells to organoids: The evolution of blood-brain barrier technology for modelling drug delivery in brain cancer. *Advanced Drug Delivery Reviews*, 196:114777, 2023. doi: 10.1016/j.addr.2023.114777.

[4] S. Bakas et al. Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the brats challenge. *arXiv preprint arXiv:1811.02629*, 2018. URL https://arxiv.org/abs/1811.02629.

[5] 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*, 13:156–168, 2020. doi: 10.1109/rbme.2019.2946868.

[6] B. H. Menze et al. The multimodal brain tumor image segmentation benchmark (brats). *IEEE Transactions on Medical Imaging*, 34(10):1993–2024, 2015. doi: 10.1109/tmi.2014.2377694.

[7] G. Urban, M. Bendszus, F. Hamprecht, , and J. Kleesiek. Multi-modal brain tumor segmentation using deep convolutional neural networks. In *MICCAI-BRATS*, pages 31–35, 2014.

[8] K. Kamnitsas et al. Efficient multi-scale 3d cnn with fully connected crf for accurate brain lesion segmentation. *Medical Image Analysis*, 36:61–78, 2017. doi: 10.1016/j.media.2016.10.004.

[9] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention (MICCAI)*, page 234–241, 2015.

[10] P. D. Chang. Fully convolutional deep residual neural networks for brain tumor segmentation. In *International workshop on Brainlesion: Glioma, multiple sclerosis, stroke and traumatic brain injuries*, page 108–118, 2016.

[11] A. Casamitjana, S. Puch, A. Aduriz, and E. V. V. Sayrol. 3d convolutional networks for brain tumor segmentation. In *MICCAI-BRATS*, page 65–68, 2016.

[12] F. Milletari, N. Navab, and S. A. Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *Fourth International Conference on 3D Vision (3DV)*, pages 565–571, 2016.

[13] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, and O. Ronneberger. 3d u-net: Learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention-MICCAI*, pages 424–432, 2016.

[14] A. Myronenko. 3d mri brain tumor segmentation using autoencoder regularization. In *BrainLes MICCAI 2018. Lecture Notes in Computer Science*, page 311–320, 2018.

[15] L. Sun, S. Zhang, H. Chen, and L. Luo. Brain tumor segmentation and survival prediction using multimodal mri scans with deep learning. *Frontiers in Neuroscience*, 13:1–9, 2019. doi: 10.3389/fnins.2019.00810.

[16] G. Wang, W. Li, S. Ourselin, and T. Vercauteren. Automatic brain tumor segmentation using cascaded anisotropic convolutional neural networks. In *Proc. 6th MICCAI BraTS Challenge*, pages 297–303, 2017.

[17] A. Dosovitskiy et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. URL https://arxiv.org/abs/2010.11929.

[18] A. Vaswani et al. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[19] M. J. Willemink, H. R. Roth, and V. Sandfort. Toward foundational deep learning models for medical imaging in the new era of transformer networks. *Radiology. Artificial Intelligence*, 4(6), 2022. doi: 10.1148/ryai.210284.

[20] J. Chen et al. Transunet: Transformers make strong encoders for medical image segmentation. *arXiv preprint arXiv:2102.04306*, 2021. URL https://arxiv.org/abs/2102.04306.

[21] M. Tan and Q. Le. Efcientnetv2: Smaller models and faster training. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[22] R.-G. Dumitru, D. Peteleaza, and C. Craciun. Using duck-net for polyp image segmentation. *Scientific Reports*, 13(1), 2023.

[23] S. Roy, R. Saha, S. Sarkar, R. Mehera, R. K. Pal, and S. K. Bandyopadhyay. Brain tumour segmentation using s-net and sa-net. *IEEE Access*, 11: 28658–28679, 2023. doi: 10.1109/access.2023.3257722.

[24] F. Isensee, P. F. Jaeger, S. A. Kohl, J. Petersen, and K. H. Maier-Hein. nnu-net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature Methods*, 18(2):203–211, 2020. doi: 10.1038/s41592-020-01008-z.

[25] F. Isensee, P. Kickingereder, W. Wick, M. Bendszus, and K. H. Maierhein. No new-net. In *Proc. Pre-Conf. 7th MICCAI BraTS Challenge*, pages 222–231, 2018.

[26] O. Oktay et al. Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*, 2018.

[27] M. Z. Alom, M. Hasan, C. Yakopcic, T. M. Taha, and V. K. Asari. Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation. *arXiv preprint arXiv:1802.06955*, 2018.

[28] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang. Unet++: A nested u-net architecture for medical image segmentation. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 3–11, 2018.

[29] M. Kolarik, R. Burget, V. Uher, and L. Povoda. Superresolution of mri brain images using unbalanced 3d dense-u-net network. In *Proc. 42nd International Conference on Telecommunications and Signal Processing (TSP)*, pages 643–646, 2019.

[30] C. Wu, Y. Zou, and Z. Yang. U-gan: Generative adversarial networks with u-net for retinal vessel segmentation. In *Proc. 14th International Conference on Computer Science & Education (ICCSE)*, pages 642–646, 2019.

[31] W. Yu, B. Fang, Y. Liu, M. Gao, S. Zheng, and Y. Wang. Liver vessels segmentation based on 3d residual u-net. In *Proc. IEEE International Conference on Image Processing (ICIP)*, pages 250–254, 2019.

[32] Z. Zhang, C. Wu, S. Coleman, and D. Kerr. Dense-inception u-net for medical image segmentation. *Comput. Methods Programs Biomed.*, 192:105395, 2020.

[33] M. U. Rehman, S. Cho, J. H. Kim, and K. T. Chong. Bu-net: Brain tumor segmentation using modified u-net architecture. *Electronics*, 9(12):2203, 2020.

[34] M. Futrega, A. Milesi, M. Marcinkiewicz, and P. Ribalta. Optimized u-net for brain tumor segmentation. *arXiv preprint arXiv:2110.03352*, 2021.

[35] Y. Zhou, W. Huang, P. Dong, Y. Xia, and S. Wang. D-unet: A dimension-fusion u shape network for chronic stroke lesion segmentation. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 18(3):940–950, 2021.

[36] J. O. S. H. Cleary and A. R. Guimarães. Magnetic resonance imaging. In *Pathobiology of Human Disease*, pages 3987–4004. 2014.

[37] A. A. Goodarzi, A. Anikin, and D. D. Pearson. Environmental sources of ionizing radiation and their health consequences. In *Genome Stability*, chapter 33, pages 569–581. 2016.

[38] C. Duran, P. S. Sobieszczyk, and F. J. Rybicki. Magnetic resonance imaging. In *Vascular Medicine: A companion to Braunwald's Heart Disease*, chapter 13, pages 166–183. Elsevier, 2 edition, 2013.

[39] E. T. Chou and J. A. Carrino. Magnetic resonance imaging. In *Pain Management*, chapter 10, pages 106–117. W.B. Saunders, 2007.

[40] D. C. Preston. Magnetic resonance imaging (mri) of the brain and spine: Basics. URL https://case.edu/med/neurology/NR/MRI%20Basics.htm.

[41] A. K. Dotiwala, C. McCausland, and N. S. Samra. Anatomy, head and neck: Blood brain barrier. In *StatPearls [Internet]*. StatPearls Publishing, 2023.

[42] J. P. Vieira de Mello et al. Deep learning-based type identification of volumetric mri sequences. *25th International Conference on Pattern Recognition (ICPR)*, pages 1–8, 2021. doi: 10.1109/ICPR48806.2021.9413120.

[43] X. Qin et al. Peri-tumoral brain edema associated with glioblastoma correlates with tumor recurrence. *Journal of Cancer*, 12(7):2073–2082, 2021. doi: 10.7150/jca.53198.

[44] K. Ohmura, H. Tomita, and A. Hara. Peritumoral edema in gliomas: A review of mechanisms and management. *Biomedicines*, 11(10):2031, 2023. doi: 10.3390/biomedicines11102731.

[45] Brats 2021 task 1 dataset, 2021. URL https://www.kaggle.com/datasets/dschettler8845/brats-2021-task1.

[46] R. McKinley, R. Meier, and R. Wiest. Ensembles of densely-connected cnns with label-uncertainty for brain tumor segmentation. *International MICCAI Brainlesion Workshop*, pages 456–465, 2018.

[47] U. Baid et al. The rsna-asnr-miccai brats 2021 benchmark on brain tumor segmentation and radiogenomic classification. *arXiv preprint arXiv:2107.02314*, 2021. URL https://arxiv.org/abs/2107.02314.

[48] Q. Mohammad, R. Hamnah, F. S. Afaan, and B. A. Swagat. Sentiment analysis using deep learning: A domain independent approach. 2023. doi: 10.1109/ICEARS56392.2023.10085676.

[49] N. Siddique, S. Paheding, C. P. Elkin, and V. Devabhaktuni. U-net and its variants for medical image segmentation: A review of theory and applications. *IEEE Access*, 9:82031–82057, 2021. doi: 10.1109/access.2021.3086020.

[50] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon. Cbam: Convolutional block attention module. *arXiv preprint arXiv:1807.06521*, 2018.

[51] A. Paszke et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[52] M. Brett et al. nipy/nibabel: 5.2.1. *Zenodo*, 10.5281/zenodo.10714563.