POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

# Brain MRI Tumor Segmentation with Vision Transformers

Tesi di Laurea Magistrale in
Computer Science and Engineering - Ingegneria Informatica

Author: **Massimo Maitan**

Student ID: 944771
Advisor: Prof. Daniele Loiacono
Co-advisors: Leonardo Crespi
Academic Year: 2022-2023

# Abstract

The glioma, or more broadly, brain tumor, represents one of the most debilitating and challenging to treat pathologies. Over the years, the development of medical imaging techniques has allowed for increasingly accurate diagnoses. Among these, one of the most widely used is Magnetic Resonance Imaging (MRI). The segmentation of tumor regions poses a complex challenge that requires significant expertise in the field and is vital for understanding its structure and evaluating therapeutic actions to be undertaken. In recent years, various techniques have been developed to assist medical personnel in this task, particularly methodologies based on Deep Learning and Artificial Intelligence. Concurrently, there has been a rise in the popularity of the Transformer in image analysis, in contrast to its traditional use for textual analysis. This architecture can capture long-range dependencies in input images and demonstrates significant parallelization potential, sometimes surpassing state-of-the-art architectures like Convolutional Neural Networks in both performance and efficiency. However, it should be noted that this architecture requires a substantial amount of data to be adequately trained, which is not always guaranteed in the medical field. In the context of this thesis, two architectures, namely Segmenter and SwinUNETR, will be employed to create models capable of addressing this challenge. The former is designed to analyze bi-dimensional RGB images, while the latter is specifically designed for the segmentation of 3D MRI scans. To address the limited data availability, regularization techniques, data augmentation, and transfer learning will be implemented. Thirteen models will be trained using a subset of the data provided in the 2019 edition of the BraTS Challenge. With 227 patient scans used for training, 49 for validation, and 59 for testing, the comparative analysis reveals that SwinUNETR is capable of providing more accurate and robust predictions, and also with less variability compared to Segmenter, which shows more heterogeneous results based on the pre-processing techniques applied to the training set. Furthermore, the results suggest that Transformer-based networks can challenge most of the state-of-the-art CNN-based algorithms addressing the same task.

**Keywords:** MRI, Brain Tumor Segmentation, Transformer, Deep Learning

# Abstract in lingua italiana

Il glioma, o più in generale il tumore cerebrale, rappresenta una delle patologie più debilitanti e complesse da trattare. Nel corso degli anni, lo sviluppo di tecniche di imaging medico ha permesso diagnosi sempre più accurate. Di queste, una delle più utilizzate è la risonanza magnetica (MRI). La segmentazione delle regioni di un tumore costituisce una sfida complessa che richiede notevole esperienza nel settore ed è di vitale importanza per comprenderne la struttura e per valutare le azioni terapeutiche da intraprendere. Negli ultimi anni, sono state sviluppate diverse tecniche per assistere il personale medico in questo compito, in particolare metodologie basate su Deep Learning e AI. Contestualmente, si è osservato un aumento della popolarità del Transformer nell'analisi di immagini, in contrasto con il suo utilizzo tradizionale per l'analisi testuale. Questa architettura è in grado di catturare dipendenze a lungo raggio nelle immagini di input e mostra un notevole potenziale di parallelizzazione, superando sia in prestazioni che in efficienza le architetture dello stato dell'arte come le CNN. Tuttavia, va notato che questa architettura richiede una quantità sostanziale di dati per essere adeguatamente addestrata, il che non è sempre scontato nel campo medico. Nel contesto di questa tesi, saranno impiegate due architetture, denominate Segmenter e SwinUNETR, per creare modelli capaci di affrontare questa sfida. Il primo è progettato per analizzare immagini RGB bidimensionali, mentre il secondo è concepito specificatamente per la segmentazione di risonanze 3D. Per bilanciare la limitata disponibilità di dati, saranno implementate tecniche di regolarizzazione, data augmentation e transfer learning. Verranno addestrati 13 modelli utilizzando un sottoinsieme dei dati forniti nell'edizione del 2019 della BraTS Challenge. Con le risonanze di 227 pazienti impiegate per l'addestramento, di 49 pazienti per la validazione e 59 pazienti per il test dei risultati, l'analisi comparativa rivela che SwinUNETR è in grado di fornire previsioni più accurate, più robuste con una minore variabilità rispetto al Segmenter, il quale mostra risultati più eterogenei in base alle tecniche di pre-processing applicate al training set. Inoltre, i risultati suggeriscono che le reti basate su Transformer hanno il potenziale per superare gli algoritmi CNN all'avanguardia che affrontano la stessa sfida.

**Parole chiave:** MRI, Segmentazione di tumori cerebrali, Deep Learning, Transformer

# Contents

# 1 | Introduction

In the last century, the progress in science and engineering has improved the standard of living of the human kind and, with it, also life expectation. In particular, diseases that were once considered lethal can now be treated with safe and effective treatments. Unfortunately, some types of diseases are still difficult to treat and have a very high mortality rate. Among these, we find brain cancer, which affects over 300,000 people worldwide each year. In these cases, the key to a good chance of survival is accurate diagnosis and effective treatment. Thanks to technology, we can use Magnetic Resonance Imaging (MRI), a non-invasive medical imaging technique, to not only detect the presence of a tumor but also assess its extent. This highly complex task can be facilitated by specific software that attempts to delineate the tumor areas through an analysis of the images produced by the MRI. The quantification of these areas is crucial for more accurate diagnoses and to define and monitor the treatment. The evolution of the field of Artificial Intelligence and, in particular, Machine Learning and Deep Learning, allows us to obtain increasingly accurate predictions year after year.

Over the past decade, the MICCAI (Medical Image Computing and Computer Assisted Intervention) society has annually hosted the BraTS [4–7, 20] challenge, where developers and engineers from around the world devise methods to address this problem.

## 1.1. Scope

This thesis aims to investigate two relatively recent approaches based on the Transformer [31] architecture, and evaluate their performances with respect to Convolutional Neural Networks (also called CNNs, which have become the gold standard for computer vision problems [9] in the last decade). Initially developed as an algorithm for text analysis, the Transformer has demonstrated outstanding performance in image analysis as well [11]. Specifically, this innovative architecture harnesses a mechanism known as *attention*. This introduces a novel approach to selecting and analyzing input features, addressing challenges present in other state-of-the-art architectures, such as issues related to parallelization and the locality of kernels' receptive fields [28] in CNNs.

The use of this architecture requires very large datasets for training, which is not a given in the medical field. Therefore, the use of transfer learning, regularization techniques and data augmentation [29] was necessary to achieve good results.

Two similar architectures were employed in this work:

- the Segmenter [30], which analyzes single bidimensional images (referred to as "slices") from an MRI scan
- the SwinUNETR [13], tailored for the analysis of three-dimensional scans

The following prediction results will be compared through the calculation of diverse metrics, aiming to elucidate the strengths and weaknesses of each methodology.

## 1.2. Thesis structure

This work is organized into the following chapters:

- Chapter 2 delves into the theoretical background underpinning the architectures utilized in this work. It explores key concepts in Machine Learning and Deep Learning, addressing core challenges and potential solutions. Additionally, it introduces pivotal papers that form the foundation of this thesis. The chapter concludes by elucidating the task of image segmentation, the focus of our research.
- Chapter 3 provides an overview of how Magnetic Resonance Imaging (MRI) functions, details the data collection process, and outlines the basic MRI semantic features. Following a brief explanation of gliomas diseases, it introduces the BraTS challenge, the benchmark on which our experiments are based.
- Chapter 4 outlines the Transformer-based approaches we have employed to address the Brain Tumor Segmentation task, discussing the data preparation implemented to achieve this objective.
- Chapter 5 presents the experimental design and the outcomes of our work, both quantitatively through tables and graphs and visually through comparisons of predictions. The models are extensively compared, revealing that every approach has some insightful nuances and distinctive strengths. Additionally, the study includes a comparison with other state-of-the-art models.
- Finally, chapter 6 draws insights from the obtained results and suggests future directions for this work.

# 2 | Theoretical Background

In this chapter, we will discuss the theoretical principles underlying the image segmentation task. In section 2.1 we will briefly address the topic of machine learning. Later in section 2.2 we will introduce the Deep Learning framework, the Transformer architecture and its computer vision counterpart, the Vision Transformer, and finally in section 2.3 we will tackle the image segmentation task.

## 2.1. Machine Learning and Deep Learning

### 2.1.1. Machine learning

Artificial intelligence has many different subfields. Among them, we'll find computer vision, speech recognition, fuzzy logic, robotics, natural language processing, and machine learning. The boundaries between these subareas are not well-defined, as they often overlap.

In particular, machine learning is a field that intersects with many of these areas.

According to the definition by Tom Mitchell [21] in 1997:

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E".

Essentially, based on this definition, a software "learns" if it can improve its performance in a given task with limited prior knowledge, relying on feedback related to the performance itself. Machine learning is, therefore, a sub-field of artificial intelligence that studies algorithms to create models capable of performing specific tasks based on experience. This is a very important milestone in the Computer Science field, since not all tasks can be solved by designing a software to predict all possibilities in the classical imperative approach. If the classic programming paradigm involves a computer and a program that, given input data, provides an output, machine learning introduces a new paradigm: instead of providing explicit instructions, we give the computer available data and the desired output.

The computer will then construct a model to perform this transformation.

This is a specific type of machine learning known as **supervised** learning. With this paradigm, we have knowledge of both the input and the output, and we seek a model capable of providing this transformation.

### 2.1.2. Supervised learning

In supervised learning, we use algorithms to find a model that, given a specific input, can provide the expected output. Within the realm of supervised learning, the most common tasks include *regression* and *classification*. Given these foundations, how can we determine if the model we have built is actually capable of making accurate predictions?

To do this we must split our data collection into training set, test set and validation set.

- **Training set**: a collection $X$ of $N$ samples, each of which consists of an array of features $\mathbf{x_i} =< x_{i1}, x_{i2}, ..., x_{iD} >$ and a target $t_i$ (which can be a value in a regression task or a class label in a classification task) that we use to train a model able to make predictions for each sample that are as close as possible to the corresponding target values. The training phase can be intended as an optimization problem, in which we adjust the parameters $\omega = [\omega_0, \omega_1, ..., \omega_M]$ of the model to minimize the error between the predicted values/labels and the targets, measured by mean of a **loss function**.
- **Validation set**: the loss value computed on the training set is not enough to verify the accuracy of the predictions of a model. Indeed, the value computed on these samples represents the ability of the model to predict the correct target value/label for the samples on which it is trained, but doesn't provide any information about the ability to generalize on unseen data. To assess the goodness of the model in this context, we need to calculate the metrics on a different collection of data which has not been used for the training. Such collection is called **validation set**.
- **Test set**: when we have trained all our models, we could use the performances on the validation set to choose which one is the best. By the way, also the metrics computed on the validation set won't be unbiased, since we used them to select the best model. For this reason, in order to have an unbiased estimation of the generalization error we use a third collection of data that have not been used neither for the training nor for the validation phase. We call this collection **test set.**

### 2.1.3. Overfitting, regularization and data augmentation

**Overfitting** is a phenomenon that occurs when a machine learning model is excessively trained on the training data, capturing not only the genuine patterns but also noise and random fluctuations. This results in a model that performs exceptionally well on the training data but struggles to generalize to new, unseen data. Overfit models are overly complex and essentially "memorize" the training data rather than learning the underlying relationships, leading to poor performance in real-world applications.

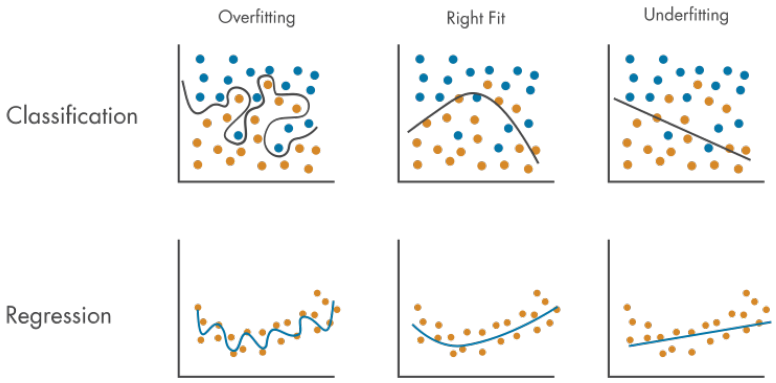



Figure 2.1: Examples of overfitting and underfitting in supervised learning tasks [19]

This problem introduces the need to find a **bias-variance trade-off**, a fundamental principle in building machine learning models. Bias represents the error introduced by the model due to excessive simplifications in its predictions, while variance reflects how sensitive the model is to fluctuations in the training data. Models that closely align with the training data tend to exhibit high variance, leading to generalization issues and overfitting. On the other hand, models with lower variance often generate oversimplified and inaccurate predictions, a condition known as a **underfitting**.

Model complexity and dataset size have a close relationship with overfitting. A larger dataset provides the model with a broader and more representative sample of the underlying data distribution, making it harder for the model to memorize noise and thus promoting better generalization. On the other hand, by constraining the model's complexity (e.g., the number of parameters), we encourage it to focus on the essential patterns in the data and reduce its tendency to overfit.

The trade-off involves finding a balance between bias and variance. A balanced model minimizes the overall error, yielding accurate and generalizable predictions.

The two most commonly employed methods for mitigating overfitting when expanding the dataset size is not feasible are:

- **Regularization**: this technique consists in imposing constraints on the model's complexity, limiting its ability to fit the training data too closely. Common types of regularization include L1 (Lasso) and L2 (Ridge), which reduce the model's parameter coefficients.
- **Data augmentation**: this strategy tries to enhance a model's generalization ability creating variations of the training data by making small changes, such as rotations, reflections, cropping, and color adjustments in images. This increases the diversity of the training dataset, helping the model better recognize real data variations and reduce overfitting.


Figure 2.2: Example of data augmentation

## 2.2. Deep learning

Machine learning algorithms allow to construct models that are able to make predictions given a set of features as input. There are tasks in which features can be found through a process called **feature engineering**, but there are cases in which it happens to be very difficult to provide handcrafted features. For example, if we have a dataset composed by a small set of features, we can use some techniques to identify which ones are important for the prediction task, but if we have a dataset composed by images it can be very impractical, since each pixel corresponds to a feature.

### 2.2.1. Artificial Neural Networks

The introduction of Deep Learning solved this problem by allowing models to automatically extract the features from complex data. In particular, Deep Learning is a sub-field of machine learning based on **Artificial Neural Networks**, a particular kind of algorithm that allows the model to extract the features of complex types of data and build intermediate representations to solve the task.

During the training phase of a neural network, the model learns to make accurate predictions from data. This process involves iterating through the dataset multiple times, with each iteration referred to as an *"epoch"*.

The novelty introduced by neural networks lies in the **hierarchical organization** of its primary units, known as *neurons*. This hierarchical arrangement allows the network to capture and process information in a structured and layered manner. As data passes through the various layers, the extracted features become progressively more abstract and semantically meaningful.

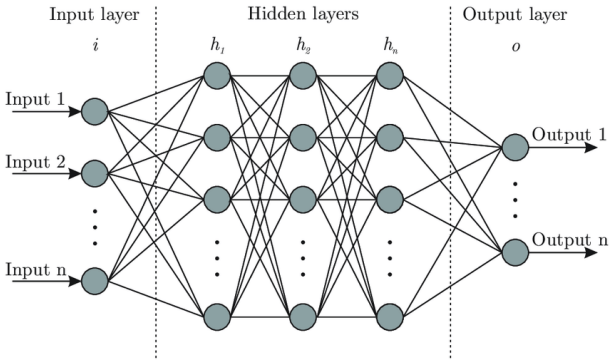



Figure 2.3: Structure of an artificial neural network

For instance, in natural language processing neural networks, the initial layers may capture features like individual words, while upper layers recognize phrases and concepts. In image processing tasks, the initial layers of the network may identify elementary features like edges, while higher layers discern complex shapes, object components, and ultimately complete objects or scenes. This hierarchical feature extraction process is a fundamental strength of deep learning models in image analysis and recognition tasks.

The network's parameters are initially set randomly, and during each epoch, the model makes predictions, computes the loss (the difference between predicted and actual values), and then adjusts its parameters through a technique called "backpropagation". Backpropagation calculates how much each parameter contributed to the error and updates them in a way that reduces the error. This iterative process continues for many epochs, gradually improving the model's ability to make accurate predictions.

In the last decade, the most common architectures used for Computer Vision have been by far Convolutional Neural Networks, also called CNNs. These kind of networks, proposed by Y. Le Cun in 1989 with the paper "Handwritten Digit Recognition with a Back-Propagation Network" [9], use an operation called **convolution** to produce intermediate representations of the input images that could be given as an input to some classifier

(typically a *fully connected neural network*, but it can be any classification algorithm) to predict the class.

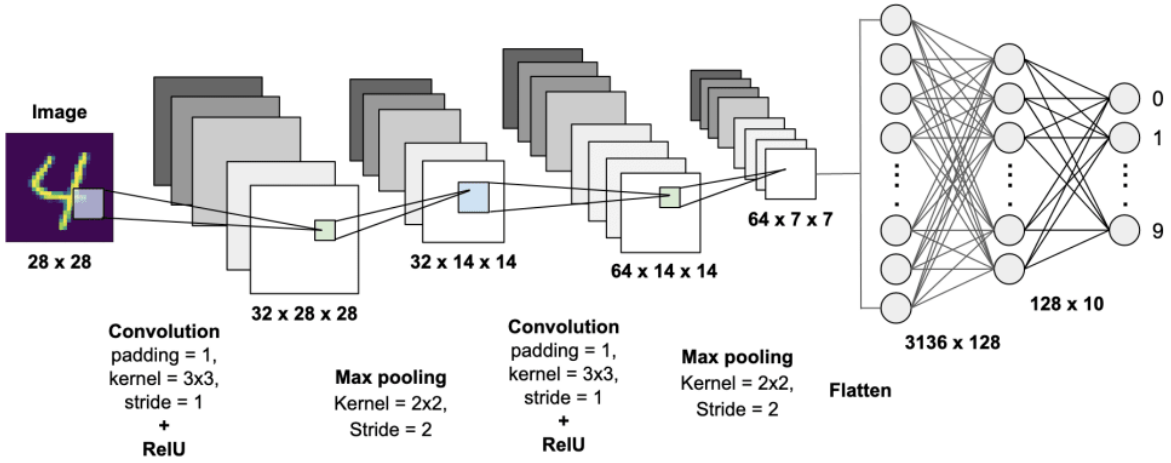



Figure 2.4: Structure of a Convolutional Neural Network

This algorithm is very powerful but processes the input in a sequential way, because it involves applying a filter (or *kernel*) to a local region of the input image and sliding it across the entire image. The issue with models that process images sequentially is that they are less inclined to parallelization, leading to significantly longer training times.

### 2.2.2. Attention is all you need: the Transformer

In 2017, Google Brain team published a paper called "Attention is all you need" [31] that described a novel architecture for language modeling called **Transformer**, that introduced a new mechanism for extracting the features of the input, called *attention*. This technique basically consists in the computation of a dynamic filter that is able to weigh more some specific parts of the input sequence that are relevant for the prediction task. The main advantage of this kind of architecture is the potential of parallelization of the computation with respect to the gold standard architecture of the time, the Recurrent Neural Networks (that used to analyze sentences sequentially and thus taking a long time to train).

The Transformer is composed of two main components:

- **Encoder**: takes the input words as a batch (thus, in parallel) and processes them in the following main steps
  - **Word encoding**: looking up on a *dictionary*, each word is converted into a *token*, i.e. an integer value. At the end of this step, the whole input text is

represented as an array.

- **Word embedding**: we apply to our input array a linear transformation that projects each token in a d-dimensional vector space that encodes primary semantic features of each word. To make things more intuitive, for example the vector that encodes the word "king" will be quite similar to the one encoding the word "queen", with a difference on the dimension that represents the gender.
- **Positional embedding**: at this point we have a vector for each word that encodes the meaning of the word. Anyway, this encoding doesn't include any information about the position of the word in the text, and this can lead to ambiguity. For example, if we compare the sentences "Although they had not won the tournament, they were happy" and "Although they had won the tournament, they were not happy", they have the same words but the position of the word *"not"* completely changes the meaning of the sentence. For this reason, the architecture enriches each vector with positional information with respect to the text with a positional embedding layer.
- **Multi-headed self-attention layer**: after the embedding phase, the processed input is ready to be fed to a series of Multi-headed self attention layers. Each block is composed by three independent linear layers that compute respectively the vectors of **values**, **keys** and **queries** for each input vector. To do an analogy, when we search something on YouTube we put, in the search bar, a query. YouTube compares this query to a set of keys, which are (for example) the titles of the videos. If some video has a key that has a similarity with the query, its content (the value) is retrieved. Once we have computed these vectors we use them to compute the **attention** with the following formula:

$$\text{Attention(Q,K,V)} = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

where $Q, K$ and $V$ are respectively the matrices of the computed queries, keys and values, and $d_k$ is the dimension of the key vectors, which is used as normalization factor. The attention filter, computed with the dot product $QK^T$, is a matrix that encodes the importance of each word with respect to all the other words in the text (hence, the name **self-attention**), and its product with the value matrix computes a filtered version of the sentence. If we imagine this

concept applied to an image, we could visualize the attention filter as a mask that gives more weight to the parts of the image on which the algorithm wants to focus to extract its features, like in the figure 2.5.

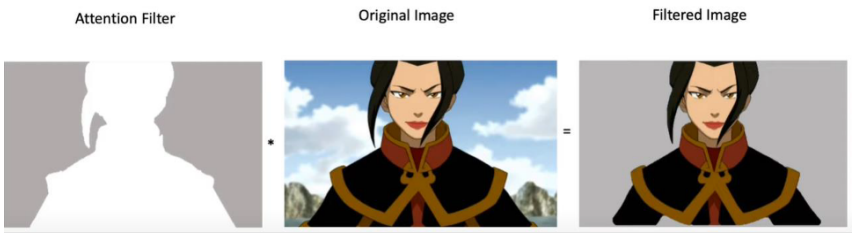



Figure 2.5: Attention filter visualization

These matrices are *learnable*: through the training phase, the model learns how to compute the keys, the queries and the values to extract the features that are important for solving the task.

A single attention filter is able to capture the relevant information about a single concept. In order to analyze the different concepts of the input, we need to compute attention with many independent attention filters. For this reason, this stage of the processing is called **multi-headed self-attention**.

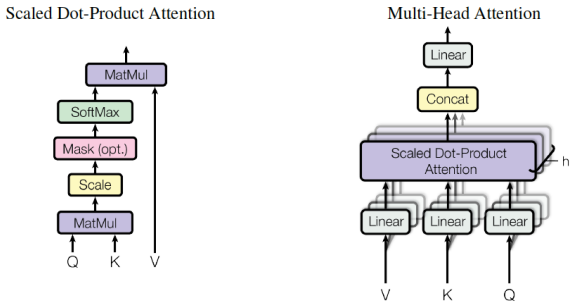



Figure 2.6: Single and multi-headed self-attention [31]

After the computation of the multi-headed self-attention, the filtered versions of the input computed by each head are enriched with original information through *skip connections*, normalized, concatenated and passed to a fully connected layer that performs a linear transformation that shapes the tokens with the desired dimension before passing the processed input to the next multi-headed self attention layer.

- **Decoder**: the encoder transforms raw data into a series of tokens that encode semantic and contextual information about the input. At this point, the decoder

will process such tokens to predict the output (in the case of text translation, a sequence of words), compare them to the ground-truth labels and compute the loss.

- **Masked Multi-Head Self-Attention**: In the first step of the decoder, each target position (the words of the correct output sequence) attends to all the positions in the input sequence. However, it's crucial to prevent "cheating" by allowing a position to attend to future input positions during training. To address this, a masking mechanism is applied. It ensures that each position in the target sequence can only attend to positions that come before it in the input sequence, making the model auto-regressive. In other words, the self-attention mechanism is not allowed to peek into the future.
- **Add and Norm**: After self-attention, residual connections are added, and layer normalization is applied. This step helps stabilize training.
- **Encoder-Decoder Cross-Attention with Masking**: In the second step of the decoder, the model needs to establish connections with the source sequence (the encoder's output) while still adhering to the causal mask to maintain autoregressive behavior. This is done through the cross-attention mechanism, specifically the "multi-head cross-attention".

  Similar to the self-attention step, the cross-attention step also involves masking. However, in this case, the masking serves a different purpose. Instead of preventing future positions in the target sequence from being attended to, it prevents positions that are padded or beyond the current position in the source sequence from being attended to. This ensures that the model focuses on relevant information in the source sequence while maintaining the autoregressive property in the generation of the target sequence.
- **Add and Norm**: Residual connections and layer normalization are applied to the feed-forward layer's output.
- **Feed Forward**: Position-wise feed-forward networks apply a linear transformation followed by a non-linear activation function to each position separately.
- **Add and Norm**: Similar to the previous step, residual connections are added, and layer normalization is applied.
- **Linear Layer**: The linear layer is responsible for mapping the model's internal representation to the desired output dimension.
- **Softmax**: The softmax function generates a probability distribution over the

output vocabulary, allowing the model to predict the next token in the sequence.

In summary, the Transformer decoder is designed for sequential output generation using self-attention and feed-forward layers, with a focus on causal generation and maintaining sequential dependencies.

### 2.2.3. Vision Transformers

In 2021, in an article titled "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", Alexey Dosovitskiy and his team [11] proposed to apply the architectural approach of the Transformer to image analysis, departing from its conventional use in text processing.

The fundamental concept behind this research is that partitioning an image into square **patches** and embedding them into a vector space, following a process similar to the one proposed in the previous subsection, we can feed them into a Transformer encoder that builds intermediary representations. Such representations can then be subject to analysis by a classifier, such as a Multi-Layer Perceptron (a fully connected neural network), to predict the label.

The proposed architecture is named **Vision Transformer** (or **ViT**) and leverages a Transformer encoder to perform image classification.

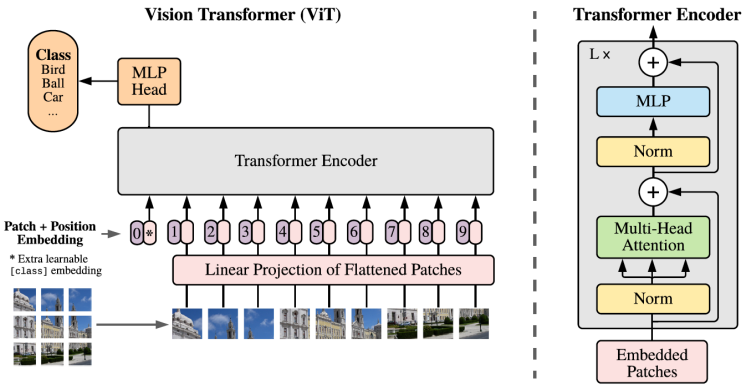



Figure 2.7: Architecture of the Vision Transformer [11]

There are many advantages to using a Vision Transformer (ViT) over a Convolutional Neural Network (CNN), including:

- **Handling Long-Range Relationships**: Vision Transformers excel in handling long-range relationships in images due to their self-attention mechanism. This capability allows them to capture global correlations between distant pixels/patches, which is often challenging for Convolutional Neural Networks (CNNs) [28].

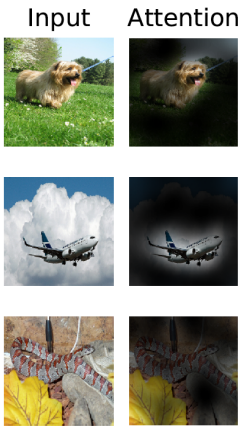



Figure 2.8: Attention maps [11]

Indeed, a CNN analyzes the input image by applying a set of kernels to a local region and sliding it across the entire image. This implies that in CNN architecture the convolution operation is able to gather global information only in the higher layers, while in the lower layers the convolution is computed across near pixels. In a ViT, self-attention allows each pixel/patch in the image to attend all other pixels/patches simultaneously, regardless of their spatial proximity. This is a crucial advantage in computer vision tasks because it enables ViTs to understand how different parts of an image relate to each other in a holistic manner. For instance, in an image of a beach scene, ViTs can capture the interplay between the distant horizon, the sky, the shoreline, and the people on the beach.

This holistic view enables ViTs to recognize patterns, context, and relationships that might be missed by traditional CNNs, which typically rely on local feature extraction through convolutional filters.

- **Scalability**: ViTs are designed to handle a wide range of image sizes. This adaptability is particularly beneficial because real-world images can vary significantly in dimensions. The ability of ViTs to handle images of varying sizes is derived from their architecture based on self-attention. In a ViT network, self-attention allows each element of the input (pixel or token) to interact with all other elements simultaneously. This means that there is no need to specify the exact dimensions of the image in advance, and the network will dynamically adapt to the input's dimensions. This flexibility is particularly valuable in scenarios where images may vary in terms of resolution or dimensions. For example, a ViT trained on low-resolution images can be seamlessly used on high-resolution images without the need for significant reconfigurations. This greatly simplifies the development process and the practical application of ViTs in a wide range of computer vision contexts.

- **Greater Parallelism**: ViTs can perform attention in parallel across all image positions, which can lead to higher computational efficiency, especially on hardware accelerators, like GPUs and TPUs, leading to faster training and inference times. It enables ViTs to handle large-scale data and complex tasks with remarkable efficiency.

In the original paper [11], the architecture has been proven to generally outperform the CNN-based architecture ResNet (with a lower computational budget) when pre-trained on large datasets (see Appendix A for additional information about transfer learning). This is shown in figure 2.9, that compares some large ViT models pre-trained repectively on JFT-300M (that comprises more than 300 millions of images, and isn't publicly available) and ImageNet21k (14 millions of images) datasets to a large ResNet model. We can see that although the TPUv3-core-days spent for pre-training ViTs are significantly smaller with respect to the ResNet architecture, the latter is significantly outperformed by the Vision Transformers that are pre-trained on the JFT dataset.

| | Ours-JFT (ViT-H/14) | Ours-JFT (ViT-L/16) | Ours-I21k (ViT-L/16) | BiT-L (ResNet152x4) | Noisy Student (EfficientNet-L2) |
|---|---|---|---|---|---|
| ImageNet | **88.55** ± 0.04 | 87.76 ± 0.03 | 85.30 ± 0.02 | 87.54 ± 0.02 | 88.4/88.5* |
| ImageNet ReaL | **90.72** ± 0.05 | 90.54 ± 0.03 | 88.62 ± 0.05 | 90.54 | 90.55 |
| CIFAR-10 | **99.50** ± 0.06 | 99.42 ± 0.03 | 99.15 ± 0.03 | 99.37 ± 0.06 | – |
| CIFAR-100 | **94.55** ± 0.04 | 93.90 ± 0.05 | 93.25 ± 0.05 | 93.51 ± 0.08 | – |
| Oxford-IIIT Pets | **97.56** ± 0.03 | 97.32 ± 0.11 | 94.67 ± 0.15 | 96.62 ± 0.23 | – |
| Oxford Flowers-102 | 99.68 ± 0.02 | **99.74** ± 0.00 | 99.61 ± 0.02 | 99.63 ± 0.03 | – |
| VTAB (19 tasks) | **77.63** ± 0.23 | 76.28 ± 0.46 | 72.72 ± 0.21 | 76.29 ± 1.70 | – |
| TPUv3-core-days | 2.5k | 0.68k | 0.23k | 9.9k | 12.3k |

Figure 2.9: Transfer of large datasets on large models [11]

The ViT model trained on the smaller ImageNet21k dataset delivers results comparable to those of the CNN architecture, and it accomplishes this with only 1/40th of the computational power. In contrast, the following figure illustrates that when ViTs are pre-trained on a smaller dataset (ImageNet with 1K classes, comprising 1.3 million images), they are unable to match the performance of ResNet models. Furthermore, the image demonstrates that with a larger pre-trained dataset, the size of the model becomes a crucial factor, and larger models achieve higher accuracy.

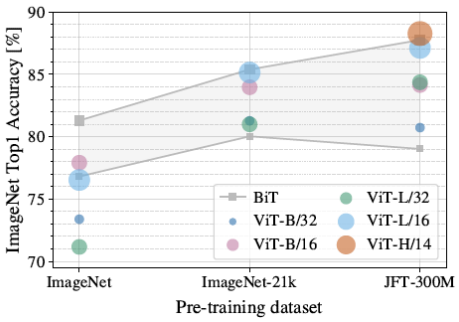



Figure 2.10: ViT and CNNs comparison with different pre-trained datasets sizes [11]

### 2.2.4. How to train a ViT: regularization and data augmentation

The study's findings indicate that ViT excels when trained on extensive datasets. However, a challenge arises in certain application domains where the available datasets are notably smaller, as is the case with this thesis. In 2021, Google Research Brain team [29] found that the combination of increased compute and techniques of Data Augmentation and Regularization can yield models with the same performance as models trained on an order of magnitude more training data. Furthermore, the study points out how transfer learning is always the best option when it comes to small datasets.

In particular, the paper shows that

- for models pre-trained with the mid-sized ImageNet-1k dataset, any kind of Data Augmentation or regularization improves the performances on all models
- for models pre-trained with ImageNet-21k dataset (which is 10x larger) there are two scenarios:
  - When training for a limited number of epochs, such as 30 epochs, any form of AugReg negatively impacts performance except for the largest models.
  - Expanding the training time to 300 epochs, AugReg benefits a broader range of models, although it still has a detrimental effect on the smaller ones.

## 2.3. Image segmentation

**Image semantic segmentation** is a fundamental task in the field of computer vision. It involves the identification, classification, and separation of specific regions within an image to recognize objects or structures of interest. This technique finds applications in a wide range of fields, including object recognition, medical diagnostics, autonomous driving, anomaly detection, and much more.


Figure 2.11: Example of image segmentation

In semantic segmentation, the process begins with a raw image and aims to divide it into different regions or segments, each representing a particular object or feature. This segmentation allows computer systems to better understand the image's content, identifying contours, shapes, and structures within it. This is particularly useful in scenarios where detailed information needs to be extracted from images, such as detecting tumors in a medical scan or isolating objects in a surveillance image.

Formally, we can define the goal of semantic segmentation task as the following:

**Proposition 2.1.** *Given an image and a finite set of classes* $\Lambda = \{K_1, K_2, ..., K_m\}$*, associate to each pixel of the image a label from* $\Lambda$*.*

Several methods have been proposed to address this task, and one prominent approach is machine learning. In particular, we can train a model to label each pixel in a supervised fashion by supplying the algorithm with a set of input-output pairs:

- Input: the source image
- Output: the desired segmentation map

Throughout the training phase, the model attempts to generate segmentation maps and continuously refines itself by assessing its error using a specific loss function, ultimately achieving accurate predictions.

# 3 | MRI Scans and Medical Image Segmentation

Magnetic Resonance Imaging (MRI) scans of the brain have revolutionized our understanding of the intricate structures and functions of the human brain. This medical imaging technique employs powerful magnetic fields and radio waves to produce detailed, high-resolution images of the brain's anatomy, allowing for the visualization of various brain regions, abnormalities, and pathologies. MRI scans are an essential tool in clinical diagnostics, neuroscience research, and the evaluation of neurological conditions, providing invaluable insights into brain health and facilitating the early detection and treatment of disorders affecting this vital organ. In section 3.1 we will introduce the basics concepts around MRI and brain scans, while in section 3.2 we will dive into the core of the thesis presenting the Brain Tumor Segmentation (often referred to as **BraTS**) challenge and its dataset.

## 3.1. Magnetic Resonance Imaging

### 3.1.1. MRI basics

Magnetic Resonance Imaging (MRI) is a non-invasive medical imaging technique that relies on the abundant presence of hydrogen atoms in the human body (water, fat, etc..). The fundamental principle of MRI is based on the magnetic properties of hydrogen nuclei, or protons. Within the body, these protons are typically randomly aligned. When a patient enters the MRI scanner, a strong external **magnetic field** is applied. This magnetic field causes the hydrogen protons to align themselves along its direction. This alignment sets the stage for image generation.

To create an image, a short burst of **radiofrequency (RF) energy** is directed at the patient's body. This RF pulse is precisely tuned to the resonant frequency of the hydrogen protons and points towards a different direction than the constant magnetic field [1, 8]. When the RF pulse is applied, it temporarily disrupts the alignment of the protons

obtained previously through the application of the magnetic field. As the RF pulse ends, the protons gradually return to their original aligned state, emitting RF signals in the process. Since water distribution is not homogeneous in the body, the variation in the detected electromagnetic field allow to determine the differences in tissue density and morphology across the entire body.

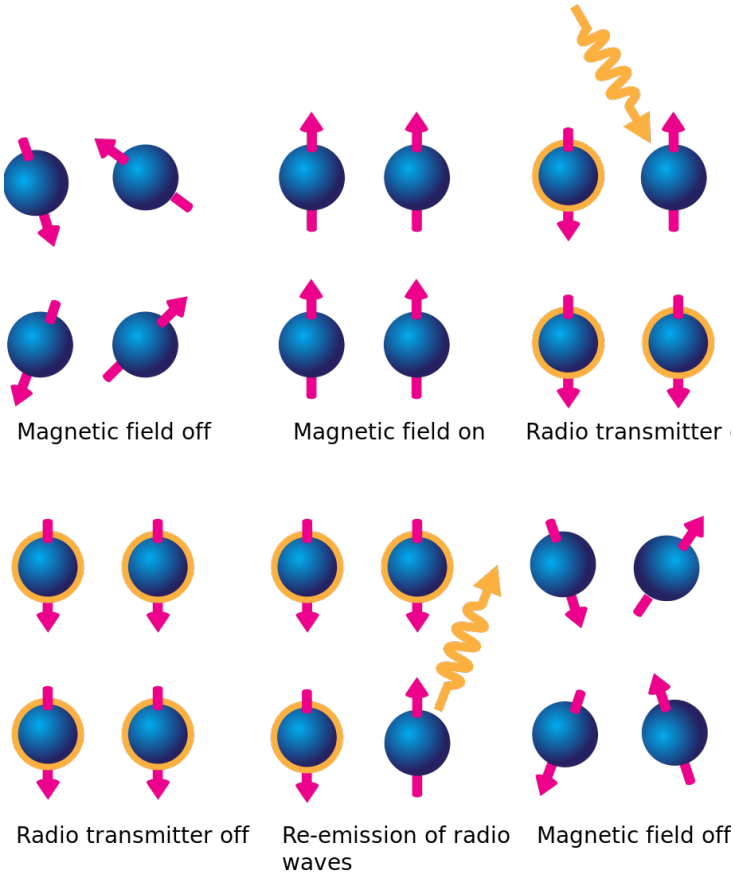



Figure 3.1: Effects of the application of magnetic field and burst of RF energy [1]

These emitted RF signals are picked up by receiver coils within the MRI machine. The signals contain information about the density and distribution of hydrogen protons within various tissues of the body.

To add spatial information to the images, gradient coils are utilized. These coils create varying magnetic field gradients across the body. By doing so, the MRI scanner can pinpoint the exact location of the emitted RF signals. This enables the creation of highly detailed 3D images, typically displayed as grayscale representations, where differences in brightness indicate variations in the density of hydrogen atoms in different tissues of the body.

Three-dimensional images are often represented slicing the scans through three planes:

- **Axial Plane**: a horizontal plane that divides the body or object into upper and lower parts.
- **Sagittal Plane**: a vertical plane that divides the body into left and right halves.
- **Coronal Plane**: a vertical plane that divides the body into front and back sections.

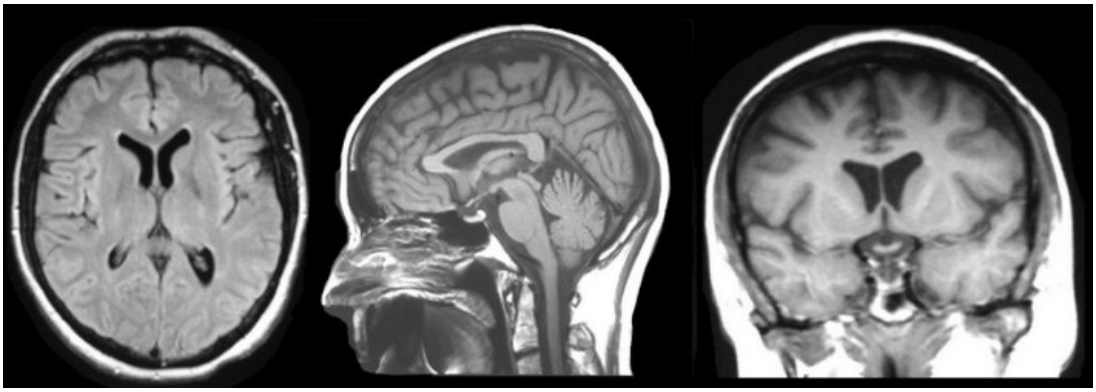


Figure 3.2: Axial, sagittal and coronal planes of a brain MRI scan [10]

Different types of images can be produced by manipulating the sequence of applied and captured RF pulses. This is done through two main parameters [10]:

- **Repetition Time (TR)**: refers to the duration between consecutive pulse sequences administered to the same slice.
- **Time to Echo (TE)**: indicates the time between the delivery of the RF pulse and the receipt of the echo signal.

In particular, different tissues can be characterized by two distinct relaxation times – T1 and T2 [10].

- **T1 (longitudinal relaxation time)**: the time constant governing the speed at which excited protons revert to their equilibrium state. It signifies the duration it takes for spinning protons to reestablish alignment with the external magnetic field.
- **T2 (transverse relaxation time)**: the time constant that dictates how quickly excited protons attain equilibrium or lose synchronicity with each other. T2 represents the duration it takes for spinning protons to lose phase coherence among nuclei spinning at right angles to the primary magnetic field.

### 3.1.2. MRI modalities

Based on this premise, various types of images can be produced by modulating the TR and TE times and measuring T1 and T2 values. In particular, we mention:

- **T1-weighted images**: during a T1 sequence, protons aligned along the main magnetic field gradually return to their equilibrium position. This sequence is ideal for visualizing the anatomical structures of the body since tissues with longer T1 relaxation times appear brighter, while those with shorter T1 relaxation times appear darker [27]. For instance, in T1 brain images, brain tissue appears brighter than cerebrospinal fluid (CSF), allowing for a clear visualization of cerebral structures. These images are produced using short TR and TE times.
- **T1ce (T1 with Contrast Enhancement)**: The T1ce sequence is a variant of the standard T1 sequence used in combination with a **contrast agent**, such as **gadolinium**. This contrast agent is administered to the patient and accumulates in areas of interest, such as lesions, tumors, or inflammations, mostly because of leaky blood vessels. During the acquisition of T1ce images, areas where the contrast agent is concentrated appear very bright, while the rest of the tissues maintain their standard T1 appearance (comparison in figure 3.3). This sequence is crucial for highlighting pathologies and lesions that might otherwise go undiagnosed.

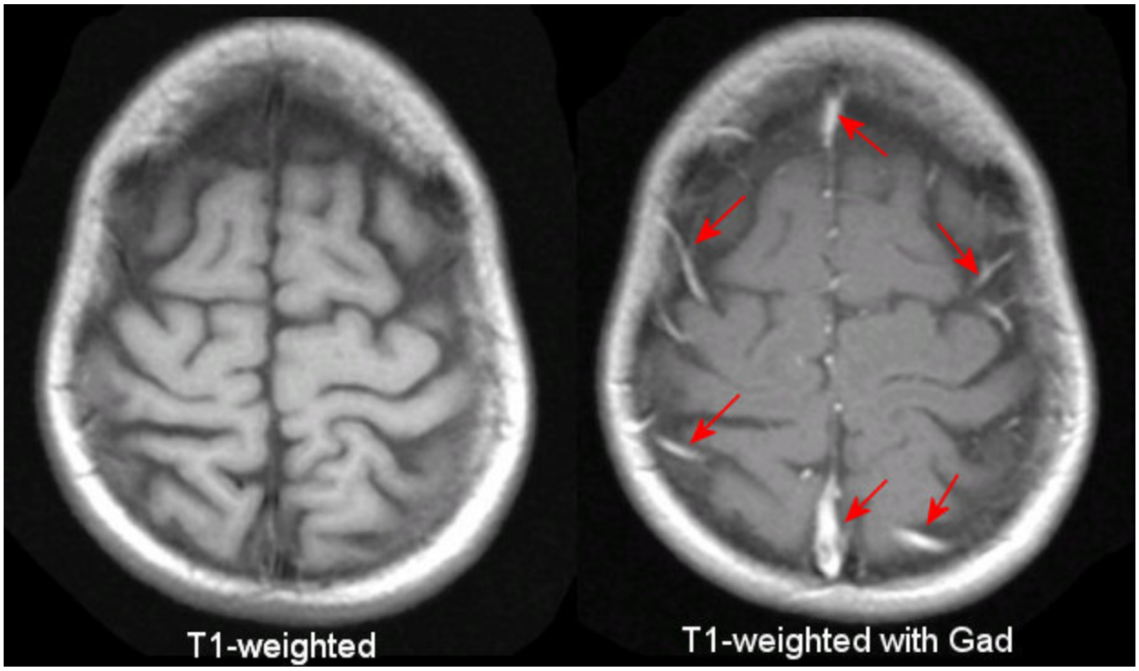



Figure 3.3: Differences in brightness of blood vessels between T1 and T1ce scans [10]

- **T2-weighted images**: these kind of images, produced using longer TR and TE times, represent the difference in the T2 (transverse relaxation) time across the scanned area. In these images, tissues with longer T2 relaxation times appear brighter, while those with shorter T2 relaxation times appear darker. The T2 sequence is widely used to detect lesions, inflammations, edema, and other pathological conditions in soft tissues. This is due to the fact that pathological processes often lead to an accumulation of fluids within the tissues. Since fluids tend to have longer T2 relaxation times, in these images brain lesions or edema appear brighter than the surrounding brain tissue, aiding in diagnosis and assessment [25]. With respect to T1-weighted images, cerebrospinal fluid appears brighter.
- **FLAIR (Fluid Attenuated Inversion Recovery)**: Similar to T2 sequence, except for the longer TR and TE times [10], the FLAIR sequence is designed to suppress the signal from fluids of the body, in particular cerebrospinal fluid (CSF) within the brain [24]. This suppression enhances the visualization of lesions or pathologies in the brain, as CSF can often obscure critical details. In these images, CSF appears very dark, while the surrounding tissues appear brighter, making any anomalies more apparent. The FLAIR sequence is commonly used in the diagnosis of neurological diseases, including brain tumors.

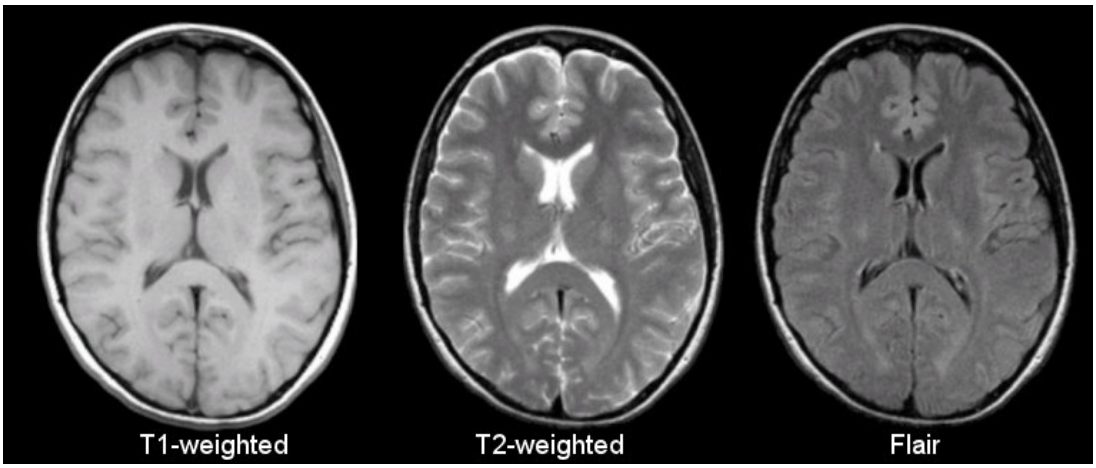



Figure 3.4: T1, T2 and FLAIR axial plan comparison [10]

### 3.1.3. Brain Tumors

A brain tumor is an abnormal growth of cells within the brain. These growths can be benign (non-cancerous) or malignant (cancerous) and can originate from brain tissue or spread from other parts of the body. Brain tumors can vary in size, location, and behavior. The most prevalent type of brain tumor in adults is the **glioma**, accounting for almost 80 percent of malignant brain tumors. They arise from the supporting cells of the brain, called the glia. In particular, **Glioblastoma multiforme (GBM)** represents the most invasive form of glial tumors. These tumors exhibit rapid growth, a propensity to spread to surrounding tissues, and a generally unfavorable prognosis. They can consist of various cell types, including astrocytes and oligodendrocytes. GBM primarily affects individuals between the ages of 50 and 70 and shows a higher incidence in men compared to women [26].

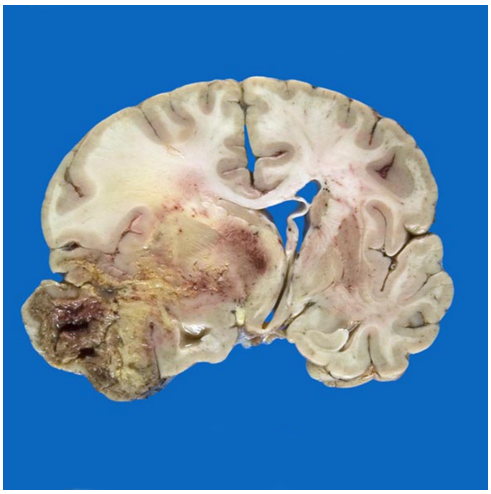


Figure 3.5: Glioblastoma multforme [12]

Brain tumors are believed to originate from the damage to specific genes on a cell's chromosomes, leading to their malfunction. These genes are in charge to control the rate of cell division, repair defects in other genes, and trigger self-destruction if the damage is irreparable. If they are damaged and can't function normally, once a cell starts dividing rapidly it can progress into a tumor. The body's immune system can serve as an initial defense, ideally detecting and eliminating abnormal cells. Tumors may produce substances that obstruct the immune system's recognition of these abnormal cells and then grow uncontrollably [26].

## 3.2. Brain Tumor Segmentation challenge

Brain tumor segmentation is a critical task in medical image analysis that involves the identification and delineation of various tumor components within brain MRI scans. Accurate segmentation is essential for diagnosis, treatment planning, and monitoring the progression of brain tumors. One of the well-known challenges associated with brain tumor segmentation is the **Brain Tumor Segmentation Challenge (BraTS)**.

The BraTS challenge [4–7, 20] is an annual competition in the field of medical image analysis. It is organized by the Medical Image Computing and Computer Assisted Interventions (MICCAI) community and focuses on the segmentation of brain tumors using multi-modal MRI scans. The challenge aims to promote research and the development of advanced algorithms for brain tumor segmentation, as well as to benchmark and compare the performance of different methods.

Participants are provided with MRI scans that include various modalities, such as

- T1-weighted modality
- T2-weighted modality
- fluid-attenuated inversion recovery (FLAIR) modality
- post-contrast T1-weighted (T1ce or, equivalently, T1Gd) modality

These different modalities offer complementary information for accurate tumor segmentation.

The challenge addresses the segmentation of different tumor types, including

- **Necrotic Tumor Core (NCR - label 1)**: This label corresponds to the core of the tumor, where the tissue is necrotic, meaning the cells in this region have died. Necrosis is a common feature in aggressive brain tumors and is often associated with tissue death and lack of blood supply. Accurate segmentation of this core is crucial for understanding the tumor's inner structure.
- **Edematous/Invaded Tissue (ED - label 2)**: This label represents the region surrounding the tumor where the brain tissue has been affected by edema. Edema is characterized by the accumulation of fluid, which can lead to swelling and changes in tissue density. In the context of brain tumors, the edematous region typically indicates areas where the tumor has invaded and impacted surrounding healthy brain tissue. Accurate segmentation of this region is vital for assessing the tumor's impact on the surrounding brain and planning surgical or therapeutic interventions.

- **GD-Enhancing Tumor (ET - label 4)**: This label refers to the region of the tumor that exhibits enhancement after the administration of a contrast agent during MRI imaging. The GD-enhancing tumor core typically represents the most actively growing and vascularized part of the tumor. This region is of particular interest in treatment planning and assessment because it often corresponds to the most aggressive and actively growing tumor cells. Accurate segmentation of this area is essential for planning surgical resections and evaluating the response to therapies.

The BraTS challenge uses specific evaluation metrics to assess the accuracy of segmentation results, including the Dice Similarity Coefficient (DSC), on three different sub-regions [3]:

- **Enhancing tumor (ET)**: described previously, is the area defined by regions exhibiting hyper-intensity in T1ce when compared to T1, as well as when contrasted with "healthy" white matter in T1ce.
- **Tumor core (TC)**: defines the central mass of the tumor, typically the area targeted for surgical removal. It encompasses both the necrotic part of the tumor and the enhancing tumor.
- **Whole tumor (WT)**: the full extent of the disease. It entails the tumor core and the peritumoral edematous region.

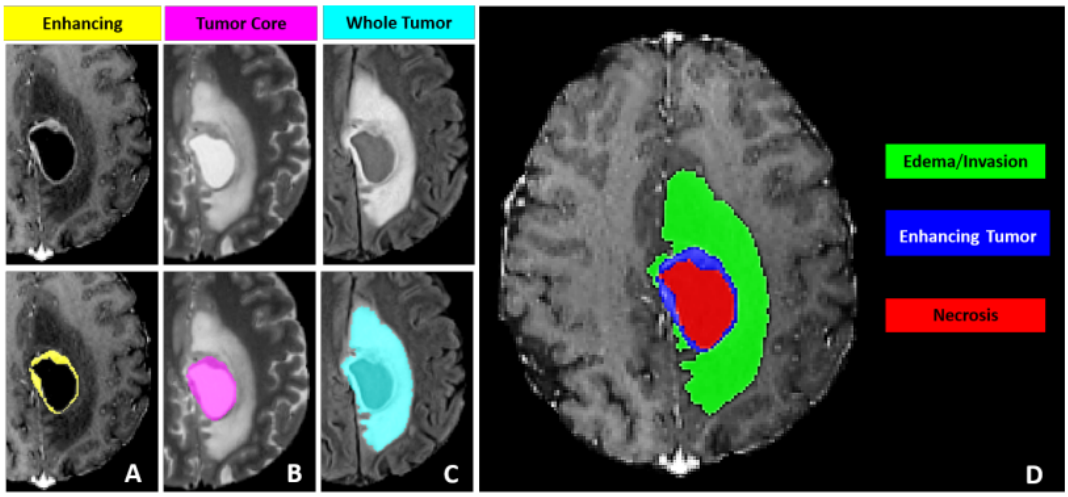



Figure 3.6: Representation [3] of the three sub-regions (A, B, C) considered for the evaluation and the three labels (D)

The challenge has significantly advanced the field of brain tumor segmentation by serving as a benchmark for state-of-the-art methods. It has encouraged the development of

deep learning models, including convolutional neural networks (CNNs), that have shown impressive results in recent editions. The developments and innovations emerging from the BraTS challenge have the potential to improve clinical practice by providing more accurate and reproducible tumor segmentations. This can lead to better patient outcomes, treatment planning, and assessment.

In 2020, Andriy Myronenko and Ali Hatamizadeh from NVIDIA proposed an architecture [23] based on Convolutional Neural Networks for achieving the brain tumor segmentation task using the BraTS2019 challenge dataset, and other teams from around the world proposed alternative approaches [2, 15, 16, 32, 33] to address the same problem. In the following chapter we will expose the approaches that we have adopted to face this challenge, and in Chapter 5 we will compare the results with these studies.

# 4 | Transformer-based Brain Tumor Segmentation

In this section, we outline the Transformer-based approaches that we have adopted in this research to tackle the brain tumor segmentation task. In particular, we focused on two distinct architectures. The first approach, as detailed in Section 4.1, seeks to accomplish segmentation by analyzing a series of slices taken along the axial plane of MRI scans, and we will provide an explanation of the pre-processing techniques employed. The second approach, as detailed in Section 4.2, achieves the segmentation task by effectively using complete three-dimensional scans as input.

The models are trained on two variations of the same dataset, respectively in slice and volumetric format. The dataset, provided by the thesis advisor and assistants as a set of .tfrecords files, is a pre-processed subset of the BraTS2019 challenge data collection.

Finally, in section 4.3 we will introduce the metrics used for the evaluation of our models.

## 4.1. Segmenter: 2D slice segmentation

This experiment is inspired by the work [30] of Robin Strudel, Ricardo Garcia, Ivan Laptev and Cordelia Schmid. The proposed architecture, named Segmenter, extends the Vision Transformer [11] to the task of semantic segmentation. We'll start with the paper's architecture presentation, and subsequently we will cover the data preparation employed to adapt this approach to the task of segmentation of medical images.

### 4.1.1. Segmenter architecture

The Segmenter architecture is inspired by the one proposed in the original Vision Transformer paper [11], and it is composed by an **encoder**, which extracts the semantic meaning of the input, and a **decoder**, which utilizes contextualized information to reconstruct a segmentation map of the input image.

#### Encoder

This component is a Transformer Encoder, described in the Vision Transformer subsection. To provide a more detailed explanation of its operation, let's break it down step by step:

- **Splitting into patches**: the original image, with dimension $W \times H \times C$ (where C represents the number of channels, typically RGB) is split into $N = \frac{HW}{P^2}$ square patches, each having dimension $P \times P \times C$. This initial transformation of the input image can be represented $\mathbf{x} \in \mathbb{R}^{W \times H \times C} \rightarrow \mathbf{x} = [\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_N}] \in \mathbb{R}^{N \times P \times P \times C}$, where $\forall i \in [1, N]\ \mathbf{x_i} \in \mathbb{R}^{P \times P \times C}$.
- **Flattening**: each patch is is flattened from a $P \times P \times C$ tensor into a one-dimensional vector with dimension $P^2 \cdot C$. This results in a transformation of the input image $\mathbf{x} = [\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_N}] \in \mathbb{R}^{N \times P \times P \times C} \rightarrow \mathbf{x} = [\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_N}] \in \mathbb{R}^{N \times P^2 C}$, i.e. a matrix composed by $N$ arrays, each with length $P^2 C$.
- **Linear projection into embedding space**: each patch undergoes a learnable linear application represented by $\mathbf{E} \in \mathbb{R}^{D \times P^2 C}$ projecting them in a D-dimensional vector space. Consequently, the input image is transformed into a set of N vectors, each with D dimensions: $\mathbf{x} = [\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_N}] \in \mathbb{R}^{N \times P^2 C} \rightarrow \mathbf{x_0} = [\mathbf{E} \cdot \mathbf{x_1}, \mathbf{E} \cdot \mathbf{x_2}, ..., \mathbf{E} \cdot \mathbf{x_N}] = [\mathbf{d_1}, \mathbf{d_2}, ..., \mathbf{d_N}] \in \mathbb{R}^{D \times N}$.
- **Position embedding**: to each vector, we add learnable positional information $\mathbf{z_0} = \mathbf{x_0} + \mathbf{pos}$ where $\mathbf{pos} \in \mathbb{R}^{N \times D}$.

- **Multiple layers of multi-headed self-attention and MLP**: At this stage, the input image has been transformed into a series of N vectors within a D-dimensional embedding space. The Transformer encoder then processes these tokens through multiple layers of Multi-Head Self-Attention, which weight the input patches by computing self-attention, and subsequently pass them through a Multi-Layer Perceptron (MLP). Both of these blocks incorporate layer normalization to stabilize the training process and skip connections to preserve original information.

$$\mathbf{a_{i-1}} = \mathbf{MSA}(\mathbf{LN}(\mathbf{z_{i-1}})) + \mathbf{z_{i-1}}$$

$$\mathbf{z_i} = \mathbf{MLP}(\mathbf{LN}(\mathbf{a_{i-1}})) + \mathbf{a_{i-1}}$$

$\forall i \in [1, L]$, where L is the number of encoder layers.

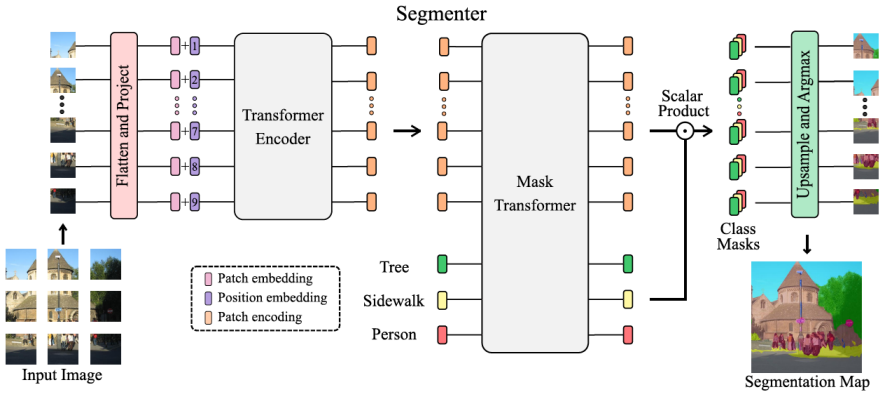



Figure 4.1: Segmenter architecture [30]

## Decoder

At this point, the encoder feeds the encoded sequence $\mathbf{z_L} \in \mathbb{R}^{N \times D}$, containing semantic and contextualized information about the input patches, to the decoder, that produces the **segmentation map s** $\in \mathbb{R}^{H \times W \times K}$, where $H$ and $W$ are the original height and width of the input image and $K$ is the number of classes. The paper proposes two approaches for implementing the decoder:

- **Linear decoder**: feeds the patch encodings $\mathbf{z_L} \in \mathbb{R}^{N \times D}$ to a point-wise linear layer that produces patch-level class logits $\mathbf{z_{linear}} \in \mathbb{R}^{N \times K}$. In other words, this step outputs, for each of the N patches, a score for each class. At this point, these N tokens are reshaped in K bi-dimentional images that still preserve the patch-level resolution $\mathbf{s_{linear}} \in \mathbb{R}^{H/P \times W/P \times K}$. Next, the feature maps are bilinearly upsampled

to the original image resolution $\mathbf{s} \in \mathbb{R}^{H \times W \times K}$ and then fed into a softmax block that transforms the logits into probabilities. Finally, we utilize the argmax function to select, for each pixel, the class with the highest probability, and compute the final segmentation map.

- **Mask Transformer**: this approach introduces a set of $K$ randomly initialized learnable D-dimensional vectors known as **class embeddings**, denoted as $\mathbf{c} = [\mathbf{c_1}, \mathbf{c_2}, ..., \mathbf{c_K}] \in \mathbb{R}^{D \times K}$, with each vector assigned to a specific class. The Mask Transformer is basically designed as a Transformer Encoder with $M$ layers that takes as input:
  - The patch encodings $\mathbf{z_L}$ outputed by the encoder
  - The "raw" class embeddings $\mathbf{c}$

  Following the final layer of the decoder, we have the processed class embeddings $\mathbf{c'} \in \mathbb{R}^{D \times K}$ and patch tokens $\mathbf{z'_M} \in \mathbb{R}^{D \times N}$. At this stage, the assignment of classes to each patch is achieved by computing the dot product between the respective class embedding vector and the patch token.

$$\mathbf{m} \in \mathbb{R}^{N \times K} = \mathbf{z'_M} \cdot \mathbf{c'}^T$$

  Subsequently, through a pipeline of
  - reshaping: $\mathbf{m} \in \mathbb{R}^{N \times K} \rightarrow \mathbf{s_{mask}} \in \mathbb{R}^{H/P \times W/P \times K}$
  - bilinear upsampling: $\mathbf{s_{mask}} \in \mathbb{R}^{H/P \times W/P \times K} \rightarrow \mathbf{s} \in \mathbb{R}^{H \times W \times K}$
  - softmax: $\mathbf{s}$ values are transformed such that masks sequences are **softly exclusive** (i.e., scores for all the classes of the same pixel sum up to 1)
  - argmax (only at inference time): for each pixel, we assign the class with the highest probability: $\mathbf{s} \in \mathbb{R}^{H \times W \times K} \rightarrow \mathbf{s} \in \mathbb{R}^{H \times W}$.

  we derive the ultimate segmentation map.

### Loss function

The loss function used for computing the training error is the **per-pixel Cross-Entropy loss**, that averages the classification score on each pixel:

$$Loss = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{C} t_{ni}\log(y_{ni}) \tag{4.1}$$

where:

- $y_{ni}$ is the predicted probability that pixel $n$ belongs to the class $i$, computed with the softmax function, which standardizes the score values for a particular pixel across all classes, ensuring that they collectively add up to 1 (see Appendix B for additional information).
- $t_{ni}$ equals 1 if the actual class of pixel $n$ in the ground-truth is class $i$, and it equals 0 otherwise
- $C$ and $N$ represent the number of classes and the number of pixels of the batch respectively

## 4.1.2. Data preparation

The dataset utilized for the experiment comprises a training set consisting of 14,894 slices extracted from the axial plane of MRI scans obtained from 227 patients, along with an additional set of 6,272 slices reserved for validation. For the test set, we ultimately have 7,552 slices. Each slice comprises four single-channel (greyscale) images with a resolution of $180 \times 180$ pixels, representing the four MRI scan modalities (T1, T1 with contrast or T1ce, T2, FLAIR). Each slice is accompanied by a ground truth segmentation map that the model aims to reconstruct during the training phase. While the training set consists of a variable number of slices for each patient, resulting from the dataset cleaning process that removed slices with only the background class, the validation and test sets comprise a fixed number of 128 slices for each patient.

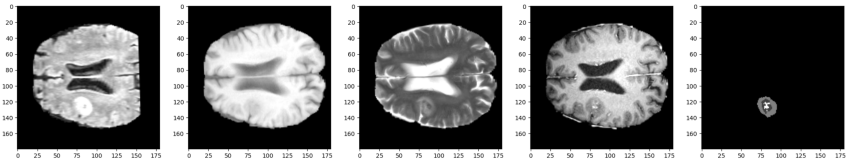
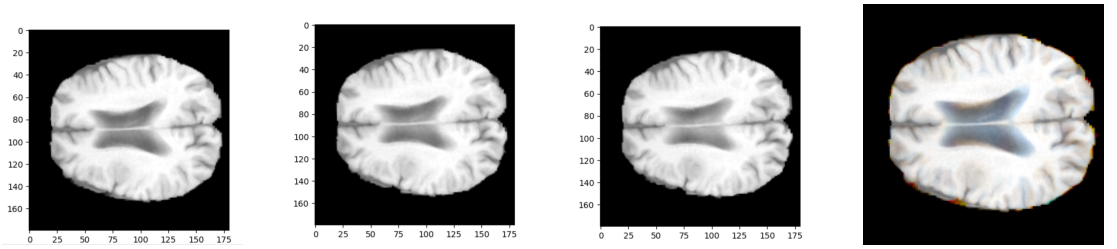


Figure 4.2: FLAIR, T1, T2 and T1ce of a single slice and the ground-truth segmentation

## Pre-processing

The dataset's input images are made up of four single-channel images, but the Segmenter code provided in the paper is designed to analyze standard RGB images, consisting of three channels. In order to maximize the analysis capabilities across these three channels, we opted to combine three distinct single-channel images into a single RGB image. We achieved this using two approaches:

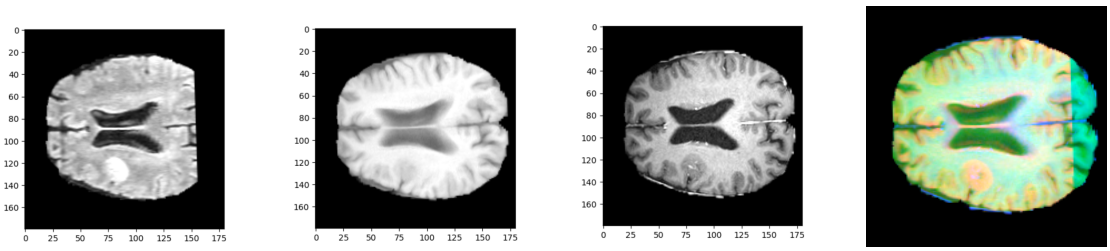
- **Consequent slices merging**: the fundamental concept is to construct 3-channel images by placing three consecutive slices of a specific modality in the R, G, and B channels, respectively.



(a) $(i-1)^{th}$ T1 slice (b) $i^{th}$ T1 slice (c) $(i+1)^{th}$ T1 slice (d) $i^{th}$ T1-merged slice

Figure 4.3: An example of consequent slice merging

- **Modality merging**: with this approach we create RGB images by assigning three different modalities to each of the three color channels respectively. Specifically, we have chosen to focus on FLAIR, T1 and T1ce modalities.

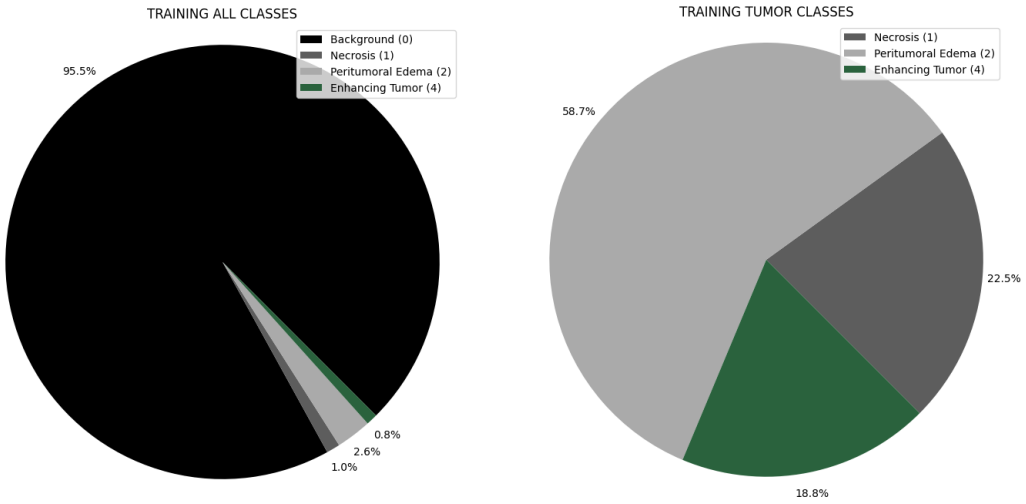


(a) FLAIR (b) T1 (c) T1ce (d) Merged

Figure 4.4: An example of modality merging

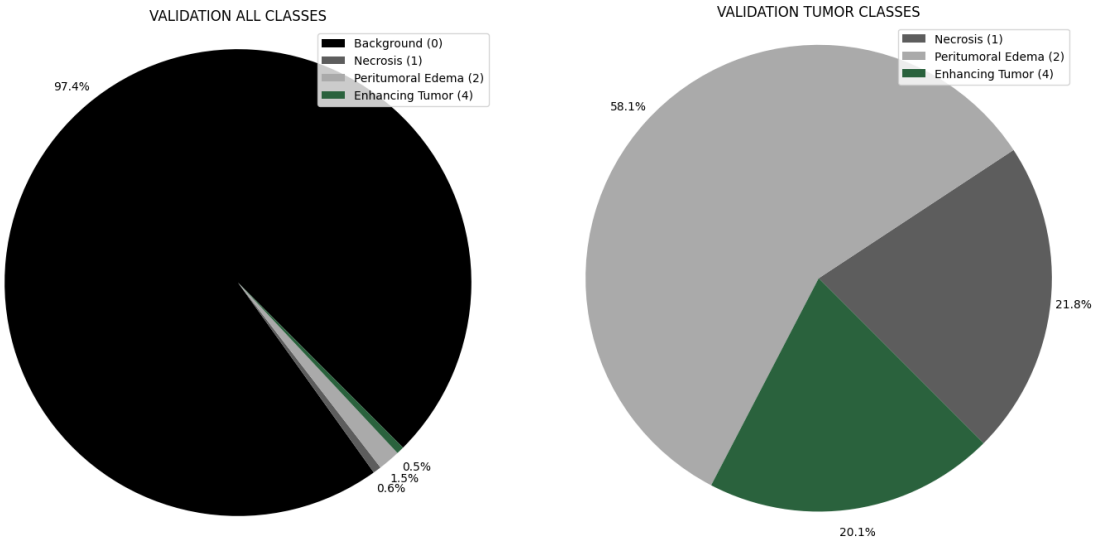
## Class imbalance

The dataset used for the experiment exhibits a significant class imbalance, primarily stemming from the inherent small size of the tumor region in comparison to the broader brain and magnetic resonance imaging context. Below, we provide the class proportions for the training and validation sets.




(a) All classes (b) Tumor classes

Figure 4.5: Training set class proportion




(a) All classes (b) Tumor classes

Figure 4.6: Validation set class proportion

## Data augmentation

To enhance the model's performance, we employ various data augmentation techniques on the input images. These techniques encompass:

- **Resizing**: The images are resized with a maximum scale of $180 \times 180$ and an aspect ratio ranging from 0.5 to 2.0. This resizing provides variability in image dimensions, making the model more robust to variations in input image sizes.
- **Random Flip**: The images are horizontally flipped with a 50 percent probability. This augmentation introduces diversity by mirroring images, thus increasing the model's ability to generalize.
- **Photometric Distortion**: Random photometric distortions are applied to the images. This includes adjustments in contrast and brightness, contributing to an expanded range of data variability. These distortions help the model learn to accommodate changes in lighting and image quality.

Additionally, we normalize the input images to ensure consistency in pixel values, and the training set is shuffled to eliminate any order-related biases during training. These combined data augmentation and preprocessing steps result in a more robust and generalizable model.

## 4.2. SwinUNETR: 3D scan segmentation

In this experiment we use the SwinUNETR [13] architecture proposed by NVIDIA for training models that predict the tumor sub-regions for the BraTS challenge on three-dimensional MRI scans. We will first briefly discuss the proposed architecture, and then we will present the data preparation.

### 4.2.1. SwinUNETR architecture

The SwinUNETR architecture is composed by an **encoder**, that analyze the input at different resolutions and using a shifting window mechanism, and a CNN-based decoder that reconstructs the three-dimensional segmentation map of the input scan.

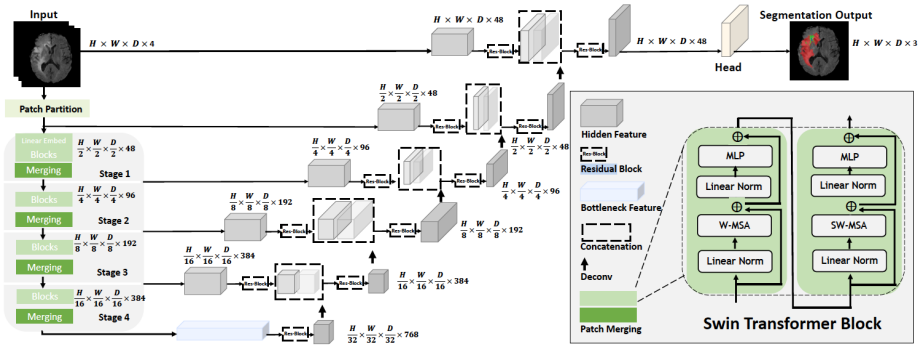



Figure 4.7: Swin-UNETR architecture [13]

#### Encoder

The encoder has a structure similar to the one presented for the Segmenter, except for the application of some interesting techniques.

- **Splitting into patches**: the original scan is represented by four different $H \times W \times D$ 3D scans, one for each MRI modality: FLAIR, T1, T1ce, T2. Each of the four tokens is split into $\frac{H}{2} \times \frac{W}{2} \times \frac{D}{2}$ patches with a dimension of $(2 \times 2 \times 2)$ voxels.
- **Flattening and projection into embedding space**: each patch is is flattened into a vector space of $2 \times 2 \times 2 \times 4 = 32$ dimensions.
- **Linear projection into embedding space**: each patch undergoes a learnable linear application that projects them into a $C$-dimentional embedding space, in a way that is similar to the one discussed in the Segmenter.

- **Swin Transformer blocks**: At this stage, the input tokens are processed by a series of 4 Swin Transformer blocks, each composed by the following steps:
  - **Window self-attention**: we evenly partition the tokens into windows of $7 \times 7 \times 7$ patches, and compute self-attention on each window. Subsequently, the tokens are passed through a normalization layer and a multi-layer perceptron (MLP).

$$\hat{z}_l = \text{W-MSA}(\text{LN}(z^{l-1})) + z^{l-1}$$

$$z_l = \text{MLP}(\text{LN}(\hat{z}^l)) + \hat{z}^l$$

  - **Shifted-window self-attention**: the mechanism is similar to the W-MSA, but each window is shifted by $\lfloor \frac{M}{2} \times \frac{M}{2} \times \frac{M}{2} \rfloor$ patches. In order to make the computation of the shifted window mechanism more efficient, a 3D cyclic-shifting [17] is applied.

$$\hat{z}_{l+1} = \text{SW-MSA}(\text{LN}(z^l)) + z^l$$

$$z_{l+1} = \text{MLP}(\text{LN}(\hat{z}^{l+1})) + \hat{z}^{l+1}$$

    This mechanism allows to make connections between neighboring non-overlapping windows of the previous layer.

  - **Patch-merging**: in order to maintain the hierarchical structure of the encoder, the number of patches is reduced as the network gets deeper. To do this, at the end of each stage a patch-merging layer concatenates groups of $2 \times 2 \times 2$ patches into a series of $4C$-dimensional vectors and passes them to a linear layer that shrinks them in a $2C$-dimensional vector space.

This means that in the second transformer block, we will have a number of patches equal to $\frac{H}{4} \times \frac{W}{4} \times \frac{D}{4}$. Each consists in $4 \times 4 \times 4$ voxels and is embedded in a $2C$-dimensional vector space. Similarly, the third layer will half the number of patches, each composed by $8 \times 8 \times 8$ voxels, and project them into a $4C$-dimensional vector space. Finally, the last layer will project a smaller number of patches, each containing $16 \times 16 \times 16$ voxels, into a $8C$-dimensional vector space.

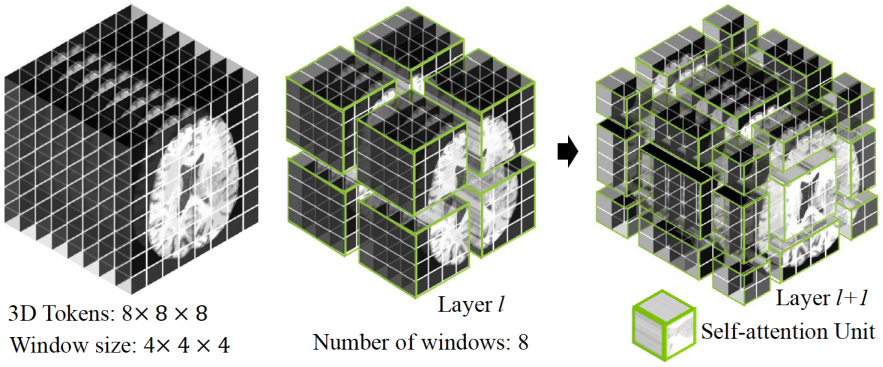



**Fig. 2.** Overview of the shifted windowing mechanism. Note that $8 \times 8 \times 8$ 3D tokens and $4 \times 4 \times 4$ window size are illustrated.

Figure 4.8: Shifted-window self-attention mechanism [13]

## Decoder

At each layer $i$ of the encoder we take the extracted feature, reshape it into a tensor of size $\frac{H}{2^i} \times \frac{W}{2^i} \times \frac{D}{2^i}$, and feed them into a Residual Block [14], composed by two convolutional layers with $3 \times 3 \times 3$ kernels. The output of the convolution is upsampled by a factor of 2 using a deconvolutional layer and concatenated with the output of the $(i-1)^{\text{th}}$ encoder-layer. The concatenated feature map is then passed through another residual block, and concatened with the output of the previous layer, and so on.

The final segmentation outputs are generated using a $1 \times 1 \times 1$ convolutional layer followed by a sigmoid activation function.

## Loss function

For determining the training error the paper proposes the **voxel-wise soft DICE Loss**

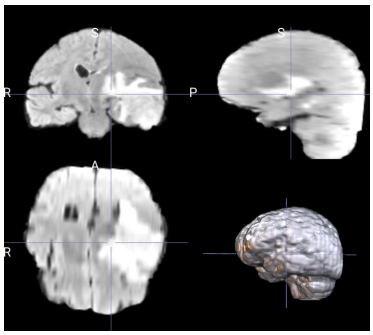
$$Loss = 1 - \frac{2}{J} \sum_{j=1}^{J} \frac{\sum_{i=1}^{I} G_{i,j} Y_{i,j}}{\sum_{i=1}^{I} G_{i,j}^2 + \sum_{i=1}^{I} Y_{i,j}^2} \tag{4.2}$$

where:

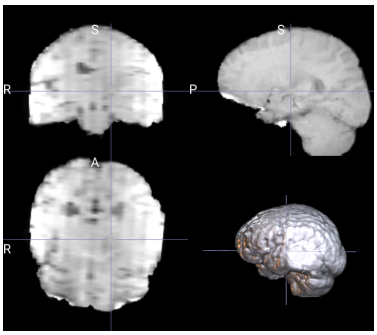
- $I$ and $J$ denote respectively the number of voxels and the number of classes
- $Y_{i,j}$ represents the probability predicted by the model that the voxel $i$ belongs to class $j$
- $G_i$ is a one-hot-encoded array that represents the ground-truth for voxel $i$.

### 4.2.2. Data preparation

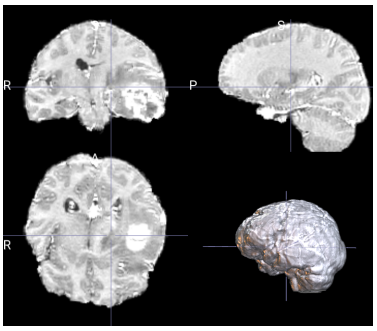
For this experiment, we utilized the same dataset employed for the Segmenter Models, organized into multiple volumes rather than slices. The training set comprises 227 patients, the validation set includes 49 patients, and the test set is comprised of 59 patients. Each patient's data is collected and processed into a set of four NIFTI files, corresponding to different modalities: FLAIR, T1, T1 with contrast (T1ce), and T2. Additionally, for each patient, there is a ground-truth NIFTI file containing the segmentation maps for the classes defined in the BraTS dataset. Each MRI scan has a dimension of $180 \times 180 \times 128$ voxels.



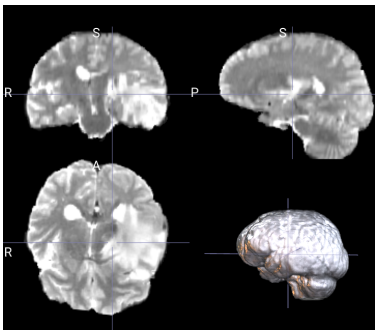
(a) FLAIR MRI scan.



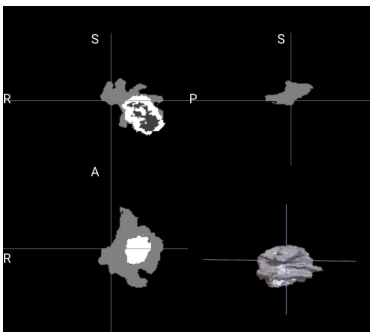
(b) T1 MRI scan.



(c) T1ce MRI scan.



(d) T2 MRI scan.

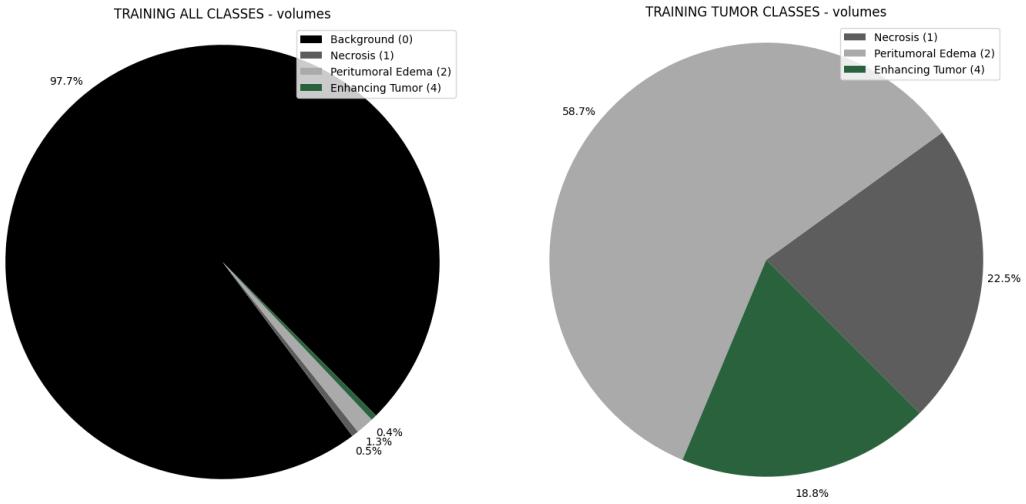


(e) Tumor ground-truth.

Figure 4.9: An example of MRI scan and its respective tumor ground-truth

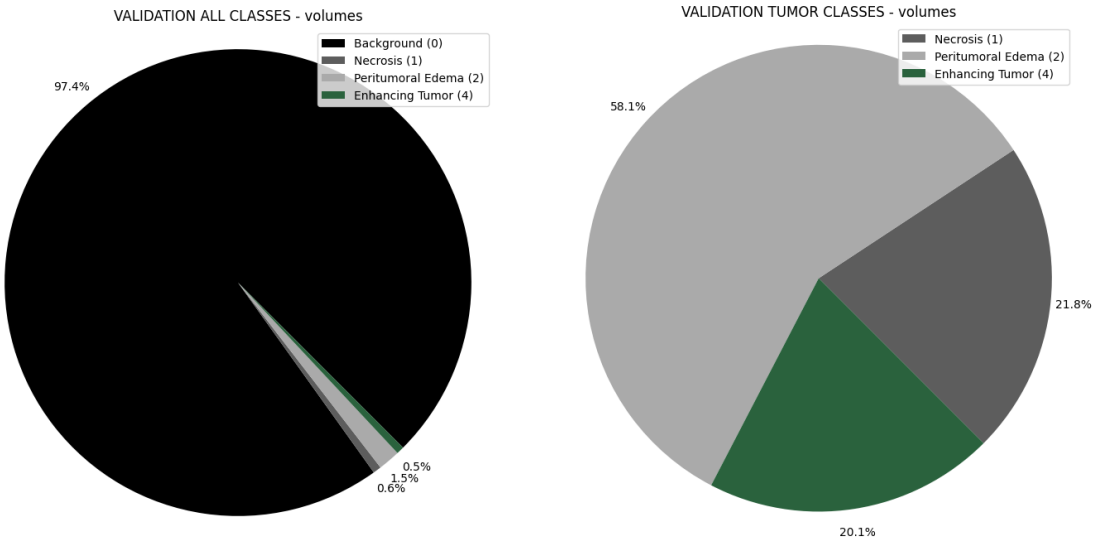
## Class imbalance

Exactly like the slice dataset, this dataset exhibits a substantial class imbalance. Notably, while the slice training set was cleared of all empty slices, all these volumes encompass the entire scan without any removal of background-only areas. Consequently, the class percentages for the training set differ slightly.




(a) All classes (b) Tumor classes

Figure 4.10: Training set class proportion



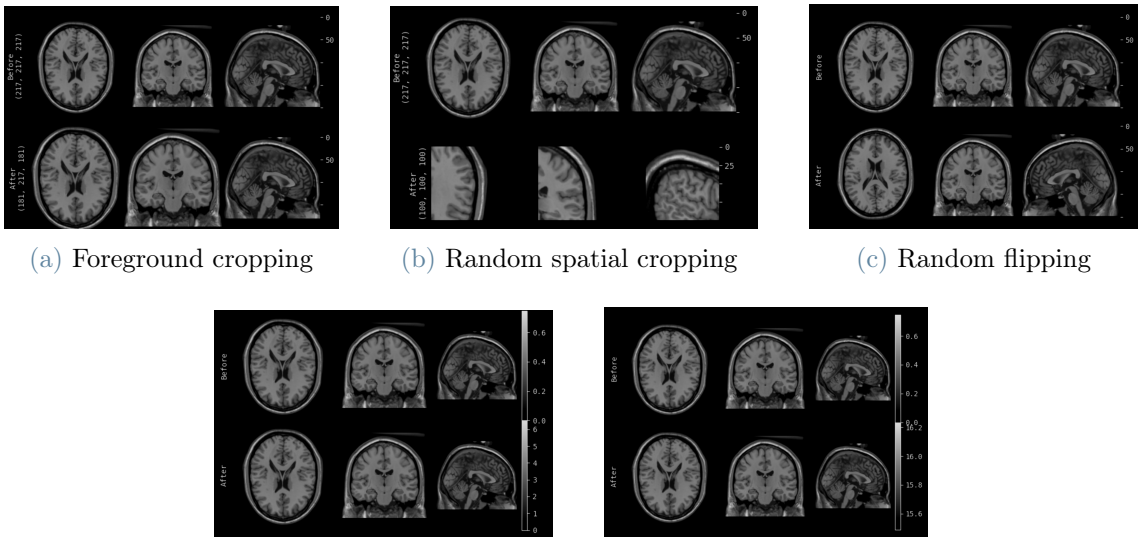

(a) All classes (b) Tumor classes

Figure 4.11: Validation set class proportion

## Preprocessing and data augmentation

To enhance the model's performance, we employ various preprocessing and data augmentation techniques on the input scans. These techniques are described in detail in the official documentation of MONAI [22] and encompass, in this order:

- **Foreground crop**: cropping the original scan using a bounding box that excludes most of the foreground, which is not interesting for the analysis. This function is commonly utilized to assist in training and evaluation when dealing with medical images where the relevant or valid part occupies a relatively small portion of the entire image.
- **Random Spatial Cropping**: crop the image with a randomly sized or a specific-sized Region of Interest (ROI).
- **Random Flipping**: the images are subject to mirroring, which occurs independently across all three spatial dimensions with a 50 percent probability.
- **Random intensity scaling and shifting**: each of the input scans undergoes a random intensity scaling and shifting operation.



(a) Foreground cropping (b) Random spatial cropping (c) Random flipping

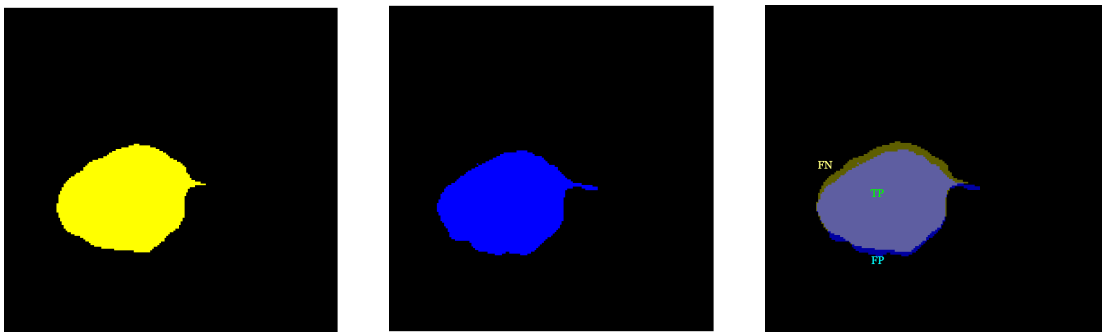
(d) Random intensity scaling (e) Random intensity shifting

Figure 4.12: The preprocessing tecnhiniques applied to the SwinUNETR dataset

## 4.3. Evaluation metrics

In the realm of machine learning, particularly in tasks like semantic segmentation, evaluation metrics are indispensable tools for quantifying the performance of models and algorithms. Central to these metrics are four fundamental quantities:

- **True Positives (TP)**: true positives in semantic segmentation refer to those pixels or regions that the model correctly identifies as part of a specific class or object. Essentially, TP indicates the model's ability to accurately capture and segment the regions of interest within an image or scene.
- **True Negatives (TN)**: true negatives, in this context, represent the pixels or regions that the model correctly recognizes as not belonging to a specific class or object. TN showcases the model's accuracy in excluding regions that do not pertain to the target class, ensuring precision in segmentation.
- **False Positives (FP)**: false positives occur when the model incorrectly identifies pixels or regions as part of a class when they do not belong to that class. In semantic segmentation, FP may lead to the model's misinterpretation of background elements as foreground objects, potentially resulting in over-segmentation.
- **False Negatives (FN)**: false negatives, in the context of semantic segmentation, are those pixels or regions that the model fails to identify as part of a specific class or object when they should have been. FN can lead to the omission of relevant objects or regions, causing under-segmentation and incomplete analysis.




(a) Ground truth tumor (b) Prediction (c) TP, FN and FP

Figure 4.13: Visual example of TP, FP and FN for the Whole Tumor class

In semantic segmentation tasks, these concepts are crucial for assessing the model's performance. We will now delve into how these concepts are employed to calculate two commonly used metrics in semantic segmentation tasks.

### 4.3.1. DICE score

The DICE score, also known as the "Sørensen–Dice coefficient", is a metric used to evaluate the accuracy of segmentation in various image processing tasks, including semantic segmentation. It quantifies the overlap between the predicted segmentation (by a model) and the ground truth (the actual segmentation in the dataset). The DICE score is calculated as the intersection of the predicted and ground truth regions divided by the average size of the two regions.

$$\text{DICE} = \frac{2 \times |\text{PREDICTION} \cap \text{GROUND-TRUTH}|}{|\text{PREDICTION}| + |\text{GROUND-TRUTH}|} = \frac{2 \times \text{TP}}{2 \times \text{TP} + \text{FP} + \text{FN}} \tag{4.3}$$

It provides a value between 0 and 1, where 1 indicates a perfect match between the prediction and the ground truth, while lower values represent less accurate segmentations. In essence, it measures the similarity or agreement between the model's output and the expected results, making it a valuable tool for evaluating segmentation performance in machine learning and computer vision applications.

### 4.3.2. Intersection over Union score

Intersection over Union (IoU), also known as the Jaccard Index, is another crucial metric in semantic segmentation tasks. It assesses the quality of the segmentation by measuring the overlap between the predicted and ground truth regions. IoU is computed as the ratio of the intersection of the two regions to their union.

$$\text{IoU} = \frac{|\text{PREDICTION} \cap \text{GROUND-TRUTH}|}{|\text{PREDICTION} \cup \text{GROUND-TRUTH}|} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \tag{4.4}$$

The value of IoU ranges from 0 to 1, where 1 indicates a perfect overlap between the prediction and the ground truth, while lower values suggest less accurate segmentation. In simpler terms, it evaluates how well the predicted segmentation aligns with the actual segmentation, making it a valuable measure for assessing the performance of machine learning models in semantic segmentation tasks. Higher IoU scores indicate more precise and accurate segmentations, while lower scores indicate less accurate results.

# 5 | Experimental design and results

In this chapter, we will present and analyze the design and the results of our work. In section 5.1 we will list the models that we have trained with the two different architectures and provide details about the experimental setup. Then, in section 5.2, we will delve into the quantitative outcomes of the models we trained for both architectures. Following that, in section 5.3, we will showcase several visual examples. Lastly, in section 5.4, we will examine specific edge cases where the model predictions may not be accurate.

## 5.1. Experimental design

### 5.1.1. Segmenter Models

With the Segmenter architecture, we trained a total of ten different models. Unless specified otherwise, the models employed a batch size of 10 (refer to Appendix B for further details on mini-batches). All the models use a Mask Transformer as decoder and were trained with initial learning rate $\gamma_0 = 10^{-3}$ and minimum learning rate $\gamma_{min} = 10^{-5}$.

Specifically, three of these models were trained from scratch using the consecutive-slice merging technique. In the first two models, we merged consecutive T1-weighted slices, while in the third model, we applied this technique to the FLAIR modality.

**Segmenter models - consequent slice merging**

| | Patch | Embedding | Enc. layers | Heads | Dec. layers |
|---|---|---|---|---|---|
| **cons-t1-256-4-4-2** | $6 \times 6$ | 256 | 4 | 4 | 2 |
| **cons-t1-256-8-8-2** | $6 \times 6$ | 256 | 8 | 8 | 2 |
| **cons-flair-256-8-8-2** | $6 \times 6$ | 256 | 8 | 8 | 2 |

Table 5.1: List of models trained with the consequent-slice merging technique.

Furthermore, we trained additional five models using the modality merging technique. Notably, the first three models were trained from scratch, while the last two models underwent fine-tuning on ViT backbones pretrained on ImageNet-21k (see Appendix A for more details on transfer learning). It's worth mentioning that for the third model, referred to as *mod256-16-16-8*, we utilized a batch size of 1 due to its substantial memory demands.

**Segmenter models - modality merging**

| | Patch | Embedding | Enc. layers | Heads | Dec. layers |
|---|---|---|---|---|---|
| **mod256-8-8-2** | $6 \times 6$ | 256 | 8 | 8 | 2 |
| **mod256-8-8-4** | $6 \times 6$ | 256 | 8 | 8 | 4 |
| **mod256-16-16-8** | $6 \times 6$ | 256 | 16 | 16 | 8 |
| **pret192-12-3-8** | $16 \times 16$ | 192 | 12 | 3 | 8 |
| **pret768-12-12-8** | $32 \times 32$ | 768 | 12 | 12 | 8 |

Table 5.2: List of models trained with the modality merging technique.

Lastly, we conducted an additional experiment where we trained two models directly on a binary dataset. This dataset only consisted of two labels: one for the background and another for the tumoral area. The objective was to assess whether this approach would enhance the performance with regard to the WT score. Both models were trained from scratch using a batch size of 1.

**Segmenter models - modality merging - binary segmentation**

| | Patch | Embedding | Enc. layers | Heads | Dec. layers |
|---|---|---|---|---|---|
| **mod-wt256-2-2-8** | $6 \times 6$ | 256 | 2 | 2 | 8 |
| **mod-wt256-8-8-4** | $6 \times 6$ | 256 | 8 | 8 | 4 |

Table 5.3: List of models trained with the modality merging technique on a binary dataset.

## Optimization

During training, the weights of the network are updated with **mini-batch Gradient Descent**, with **polynomial learning rate decay** $\gamma = \gamma_0(1 - \frac{N_{iter}}{N_{total}})^{0.9}$. This technique allows the model to initially learn quickly with a high learning rate and then slow down its learning rate as training progresses. This helps to prevent oscillations and overfitting, ultimately leading to better convergence and improved generalization on unseen data.

## Regularization

To prevent overfitting, we decided to implement the two regularization techniques as they were originally configured in the paper's code by default:

- **Stochastic Depth**: We introduced stochastic depth by randomly skipping a learnable block of the encoder with a probability of 10 percent. This means that during training, some blocks in the model are randomly omitted, helping the model to generalize better and reduce the risk of overfitting. Stochastic depth can be seen as a form of regularization, similar to dropout, but at the block level.
- **Dropout**: We applied dropout with a rate of 10 percent in the decoder of the model. Dropout randomly ignores tokens that are given as input to a block. This technique prevents the model from relying too heavily on specific tokens and encourages it to learn more robust features by dropping out some of them during training. It acts as a regularization method by reducing the interdependence between individual tokens.

Both of these techniques aim to improve the model's generalization and prevent it from memorizing the training data, which can lead to overfitting. These regularization methods help the model perform better on unseen data and make it more robust.

### 5.1.2. SwinUNETR Models

We trained three different models using the SwinUNETR architecture. We used a batch size of 1 for all these models to ensure they fit within the available memory during training. The network structure employed was consistent with the description provided in the research paper:

- **Patch size**: $2 \times 2 \times 2$ voxels
- **Number of Swin-Transformer Blocks**: 4
- **Window size**: $7 \times 7 \times 7$ patches
- **Learning rate**: $10^{-4}$

The trained models differ in both the dimension of the embedding space and the size of the region of interest used for pre-processing/data augmentation techniques.

**SwinUNETR models**

| | Region of Interest | Embedding space dimension |
|---|---|---|
| **SWIN96-48** | $96 \times 96 \times 96$ | 48 |
| **SWIN64-48** | $64 \times 64 \times 64$ | 48 |
| **SWIN96-60** | $96 \times 96 \times 96$ | 60 |

Table 5.4: List of models trained with the SwinUNETR architecture.

The classes taken into account for the soft DICE loss correspond to the subregions defined in the context of the BraTS 2019 challenge, specifically, the Whole Tumor, Tumor Core, and Enhancing Tumor.

#### Optimization and regularization

To minimize the loss function, we employed the **Adam** optimizer with **decoupled weight** decay, often referred to as **AdamW**. This algorithm aims to mitigate overfitting by imposing a penalty on large weights during the training process. The approach is based on the Adam optimizer and incorporates weight decay as an additional regularization technique. You can refer to the paper [18] for more details on AdamW. The weight decay rate is set to $10^{-5}$ for all the models.

### 5.1.3. Training details

Before presenting the results, there are some considerations to do about the training phase:

- All Segmenter models underwent training for 3000 epochs, except for the *mod256-16-16-8* due to its high computational demands. The models were trained on a NVIDIA TITAN V, generously provided by Politecnico di Milano, and on A100 and V100 GPUs rented through Google Colab Pro. Upon analyzing the training logs, which are further detailed in Appendix C, it can be observed that there is weak or no evidence of overfitting in the vast majority of cases. With this consideration, all the test scores are evaluated on the last checkpoint of each model (epoch 1000 for *mod256-16-16-8* and epoch 3000 for all other models).
- Regarding the SwinUNETR models, they were trained for 2000 epochs on the same NVIDIA TITAN V; however, all of them exhibited signs of overfitting before completing the training. Therefore, for each model we have chosen the checkpoint that exhibited the best validation scores and used it to compute the test scores (see Appendix C for more details).
- Segmenter and SwinUNETR adopted divergent training approaches. The former involved computing per-pixel Cross-Entropy loss for classes 0, 1, 2, and 4, with the test scores computed separately for the Whole Tumor, Tumor Core, and Enhancing Tumor subregions. On the contrary, the latter calculated the voxel-wise soft DICE Loss on the three subregions. As a result, Segmenter's results will also incorporate the outcomes for the necrosis and edema labels, while SwinUNETR will present scores directly for the subregions.

## 5.2. Quantitative results

The following statistics are computed on the per-patient test DICE score distribution. For the sake of conciseness, IoU scores are not reported. In the context of the Segmenter, test data undergoes minimal transformations, specifically resizing (preserving the original aspect ratio) and flipping, while SwinUNETR employs sliding window inference, where the window size is determined through ROI parameters, specifically $160 \times 160 \times 128$ voxels.

### 5.2.1. Segmenter

**Segmenter models - Necrotic and Non-Enhancing Tumor core DICE**

| Model | Mean ± Std | Median | Q1 | Q3 | Min | Max |
|---|---|---|---|---|---|---|
| **cons-t1-256-4-4-2** | 30.98 ± 26.87 | 23.82 | 4.80 | 52.01 | 0.0 | 83.04 |
| **cons-t1-256-8-8-2** | 30.09 ± 27.16 | 21.51 | 3.59 | 55.94 | 0.0 | 87.15 |
| **cons-flair-256-8-8-2** | 20.88 ± 18.30 | 15.71 | 6.67 | 37.22 | 0.0 | 68.69 |
| **mod256-16-16-8** | 41.72 ± 31.22 | 46.08 | 9.73 | 70.35 | 0.0 | 91.10 |
| **mod256-8-8-2** | 44.46 ± 30.79 | 50.58 | 10.22 | 68.53 | 0.0 | 91.26 |
| **mod256-8-8-4** | 44.57 ± 30.53 | 53.92 | 15.59 | 69.89 | 0.0 | 88.23 |
| **pret192-12-3-8** | **49.36 ± 31.77** | **64.46** | 20.09 | 76.77 | 0.0 | 91.72 |
| **pret768-12-12-8** | 37.09 ± 32.37 | 31.86 | 4.33 | 65.88 | 0.0 | 90.16 |

Table 5.5: DICE performance on Necrosis [1] class for Segmenter models on the test set

**Segmenter models - Peritumoral edema DICE**

| Model | Mean ± Std | Median | Q1 | Q3 | Min | Max |
|---|---|---|---|---|---|---|
| **cons-t1-256-4-4-2** | 43.13 ± 17.16 | 44.90 | 29.59 | 56.29 | 2.26 | 76.79 |
| **cons-t1-256-8-8-2** | 45.42 ± 17.64 | 45.37 | 37.28 | 59.35 | 3.49 | 76.17 |
| **cons-flair-256-8-8-2** | 58.69 ± 18.62 | 61.79 | 46.62 | 72.35 | 0.00 | 83.01 |
| **mod256-8-8-2** | 68.15 ± 19.88 | 74.23 | 59.57 | 82.45 | 0.00 | 91.28 |
| **mod256-8-8-4** | 68.04 ± 20.00 | 73.70 | 59.74 | 82.59 | 1.43 | 90.16 |
| **mod256-16-16-8** | 66.07 ± 21.19 | 71.89 | 53.99 | 81.13 | 0.69 | 89.04 |
| **pret192-12-3-8** | **70.63 ± 18.42** | **76.55** | 63.70 | 83.36 | 6.24 | 92.36 |
| **pret768-12-12-8** | 65.39 ± 18.52 | 68.87 | 56.24 | 79.66 | 3.55 | 89.24 |

Table 5.6: DICE performance on Edema [2] class for Segmenter models on the test set

**Segmenter models - Enhancing Tumor DICE**

| Model | Mean ± Std | Median | Q1 | Q3 | Min | Max |
|---|---|---|---|---|---|---|
| **cons-t1-256-4-4-2** | 16.85 ± 14.96 | 15.18 | 4.12 | 27.11 | 0.0 | 59.66 |
| **cons-t1-256-8-8-2** | 17.96 ± 14.26 | 17.13 | 6.61 | 26.26 | 0.0 | 50.13 |
| **cons-flair-256-8-8-2** | 22.89 ± 17.82 | 20.25 | 6.86 | 34.96 | 0.0 | 60.57 |
| **mod256-8-8-2** | 62.28 ± 24.97 | 69.01 | 49.90 | 81.13 | 0.0 | 90.03 |
| **mod256-8-8-4** | 62.70 ± 24.58 | 71.10 | 46.98 | 80.85 | 0.0 | 90.00 |
| **mod256-16-16-8** | 61.18 ± 26.07 | 69.06 | 44.12 | 80.66 | 0.0 | 90.11 |
| **pret192-12-3-8** | **64.47 ± 23.80** | **72.26** | 52.34 | 82.07 | 0.0 | 90.73 |
| **pret768-12-12-8** | 55.47 ± 24.08 | 59.41 | 42.47 | 77.29 | 0.0 | 86.19 |

Table 5.7: DICE performance on ET [4] class for Segmenter models on the test set

**Segmenter models - Whole Tumor DICE**

| Model | Mean ± Std | Median | Q1 | Q3 | Min | Max |
|---|---|---|---|---|---|---|
| **cons-t1-256-4-4-2** | 67.47 ± 19.48 | 75.70 | 59.22 | 80.44 | 8.68 | 90.59 |
| **cons-t1-256-8-8-2** | 69.21 ± 18.18 | 75.94 | 62.69 | 82.87 | 17.65 | 90.57 |
| **cons-flair-256-8-8-2** | 81.26 ± 14.65 | 86.45 | 74.83 | 90.18 | 1.12 | 94.38 |
| **mod256-8-8-2** | 84.75 ± 13.96 | 89.66 | 82.76 | 91.73 | 0.85 | 94.50 |
| **mod256-8-8-4** | 84.81 ± 13.12 | 88.58 | 81.12 | 91.42 | 5.77 | 94.50 |
| **mod256-16-16-8** | 83.11 ± 13.96 | 88.48 | 80.50 | 91.39 | 6.84 | 94.78 |
| **pret192-12-3-8** | **86.56 ± 11.14** | **89.79** | 84.11 | 93.27 | 25.54 | 95.61 |
| **pret768-12-12-8** | 83.91 ± 10.80 | 86.81 | 80.63 | 91.20 | 45.38 | 94.23 |
| **mod-wt256-2-2-8** | 85.94 ± 10.07 | 89.51 | 82.09 | 91.79 | 35.96 | 94.82 |
| **mod-wt256-8-8-4** | 85.47 ± 12.43 | 89.14 | 82.04 | 92.14 | 11.51 | 94.65 |

Table 5.8: Whole Tumor DICE performance for Segmenter models on the test set

**Segmenter models - Tumor Core DICE**

| Model | Mean ± Std | Median | Q1 | Q3 | Min | Max |
|---|---|---|---|---|---|---|
| **cons-t1-256-4-4-2** | 43.11 ± 25.41 | 47.38 | 27.99 | 60.08 | 0.00 | 83.99 |
| **cons-t1-256-8-8-2** | 42.63 ± 25.22 | 47.84 | 23.43 | 58.69 | 0.00 | 85.02 |
| **cons-flair-256-8-8-2** | 46.23 ± 21.42 | 50.45 | 34.95 | 61.78 | 0.00 | 85.51 |
| **mod256-8-8-2** | 75.08 ± 21.10 | 83.59 | 65.63 | 90.25 | 0.16 | 95.55 |
| **mod256-8-8-4** | 74.61 ± 21.27 | 82.46 | 65.23 | 90.47 | 8.38 | 96.04 |
| **mod256-16-16-8** | 72.06 ± 24.78 | 83.22 | 61.33 | 89.85 | 0.15 | 94.44 |
| **pret192-12-3-8** | **80.14 ± 19.26** | **88.29** | 75.91 | 91.59 | 0.55 | 96.08 |
| **pret768-12-12-8** | 75.45 ± 20.29 | 84.40 | 71.41 | 89.79 | 7.08 | 93.83 |

Table 5.9: Tumor Core DICE performance for Segmenter models on the test set

**Segmenter models - Summary of Classes**

| Model | Necrosis [1] | Edema [2] | Enhancing Tumor [4] |
|---|---|---|---|
| **cons-t1-256-4-4-2** | 30.98 ± 26.87 | 43.13 ± 17.16 | 16.85 ± 14.96 |
| **cons-t1-256-8-8-2** | 30.09 ± 27.16 | 45.42 ± 17.64 | 17.96 ± 14.26 |
| **cons-flair-256-8-8-2** | 20.88 ± 18.30 | 58.69 ± 18.62 | 22.89 ± 17.82 |
| **mod256-16-16-8** | 41.72 ± 31.22 | 66.07 ± 21.19 | 61.18 ± 26.07 |
| **mod256-8-8-2** | 44.46 ± 30.79 | 68.15 ± 19.88 | 62.28 ± 24.97 |
| **mod256-8-8-4** | 44.57 ± 30.53 | 68.04 ± 20.00 | 62.70 ± 24.58 |
| **pret192-12-3-8** | **49.36 ± 31.77** | **70.63 ± 18.42** | **64.47 ± 23.80** |
| **pret768-12-12-8** | 37.09 ± 32.37 | 65.39 ± 18.52 | 55.47 ± 24.08 |

Table 5.10: Summary of DICE performance for Segmenter models on the test set across different classes

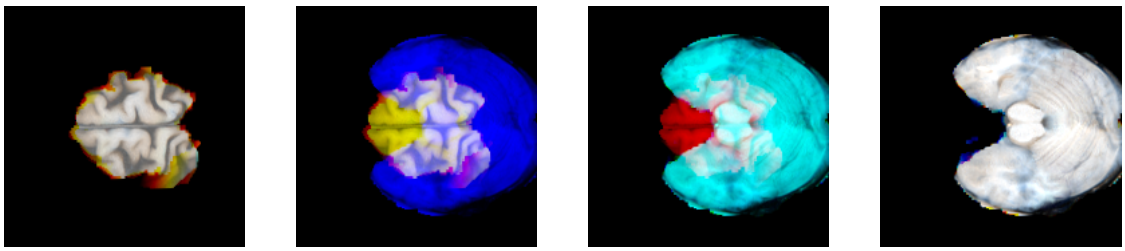
**Segmenter models - Summary of BraTS subregions**

| Model | Whole Tumor | Tumor Core | Enhancing Tumor |
|---|---|---|---|
| **cons-t1-256-4-4-2** | $67.47 \pm 19.48$ | $43.11 \pm 25.41$ | $16.85 \pm 14.96$ |
| **cons-t1-256-8-8-2** | $69.21 \pm 18.18$ | $42.63 \pm 25.22$ | $17.96 \pm 14.26$ |
| **cons-flair-256-8-8-2** | $81.26 \pm 14.65$ | $46.23 \pm 21.42$ | $22.89 \pm 17.82$ |
| **mod256-8-8-2** | $84.75 \pm 13.96$ | $75.08 \pm 21.10$ | $62.28 \pm 24.97$ |
| **mod256-8-8-4** | $84.81 \pm 13.12$ | $74.61 \pm 21.27$ | $62.70 \pm 24.58$ |
| **mod256-16-16-8** | $83.11 \pm 13.96$ | $72.06 \pm 24.78$ | $61.18 \pm 26.07$ |
| **pret192-12-3-8** | $\mathbf{86.56 \pm 11.14}$ | $\mathbf{80.14 \pm 19.26}$ | $\mathbf{64.47 \pm 23.80}$ |
| **pret768-12-12-8** | $83.91 \pm 10.80$ | $75.45 \pm 20.29$ | $55.47 \pm 24.08$ |
| **mod-wt256-2-2-8** | $85.94 \pm 10.07$ | – | – |
| **mod-wt256-8-8-4** | $85.47 \pm 12.43$ | – | – |

Table 5.11: Summary of DICE performance for Segmenter models on the test set across different BraTS subregions

Some interesting considerations regarding these results:

- Models trained using the consecutive slice merging technique appear to yield significantly lower results in all classes and subregions. This could be attributed to the inherent challenge of this technique, where certain images incorporate slices from two different patients.



(a) last correct image of patient $i$ (b) Image with slices from different patients (c) Image with slices from different patients (d) first correct image of patient $i+1$

Figure 5.1: Superposition of patients in consequent slice merging techinique

The problem could be addressed by avoiding the merging of slices from different patients. Unfortunately, in our situation, this solution is not viable due to the variable number of slices for each patient in the training dataset. Conversely, for the valida-

tion set and test set, where each patient consistently has 128 slices, implementing this fix is feasible. Consequently, we have removed from these sets all the images containing slices from different patients.

- The *cons-flair-256-8-8-2* model demonstrates superior performance in segmenting the edema and enhancing tumor areas compared to models trained with the same technique on the T1 modality (*cons-t1-256-8-8-2* and *cons-t1-256-4-4-2*). However, it exhibits lower accuracy in predicting the necrotic region. This discrepancy is likely due to each MRI modality containing distinct features crucial for identifying specific tumor regions. In this context, the T1 modality offers valuable insights for necrosis segmentation, while the FLAIR modality provides additional information about the edematous region.

  These findings align with the visual characteristics of MRI scans: the FLAIR modality appears brighter in the edematous region, whereas the darkest area in the T1 modality better outlines the shape of the necrotic region. On this premise, we can hypothesize that employing the consequent slice merging technique on T1ce scans would yield improved results for the necrosis and enhancing tumor classes.

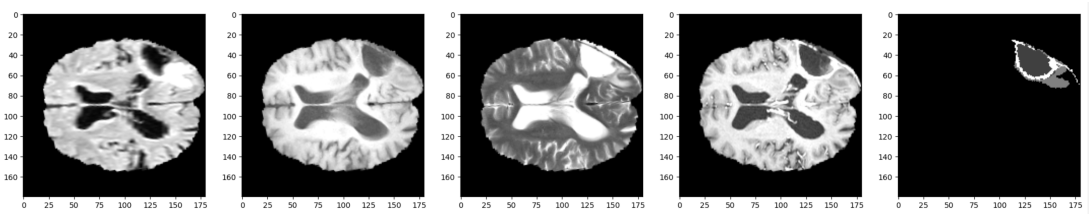


Figure 5.2: FLAIR, T1, T2 and T1ce modalities with segmentation map

  This outcome also elucidates why models trained with the modality-merging technique achieve higher scores: the integration of information from different modalities is crucial for identifying each tumor region.

- The smaller pre-trained model *pret192-12-3-8* achieves better predictions on all subgregions and tumor classes with respect to the other models. The larger pre-trained model *pret768-12-12-8* seems to have performances comparable to the other models for what concerns Whole Tumor and Tumor Core subregions, but yelds lower results on the Enhancing Tumor class.

- The *mod-wt256-2-2-8* and *mod-wt256-8-8-4* models, trained on a binary dataset, demonstrate slightly enhanced performance in addressing the whole tumor subregion compared to most of the models trained for the multi-class problem. However, the improvement is not substantial.

### 5.2.2. SwinUNETR

Now, we present the test scores of the SwinUNETR models. In contrast to the Segmenter models, all these models experienced overfitting before the completion of training. Therefore, in this subsection and the next one, we present the test results based on the best checkpoint available. Details on the selected checkpoints are available in Appendix C.

**SwinUNETR models - Whole Tumor DICE**

| Model | Mean ± Std | Median | Q1 | Q3 | Min | Max |
|---|---|---|---|---|---|---|
| **SWIN64-48** | 87.77 ± 10.19 | 91.23 | 84.63 | 93.92 | 36.99 | 97.22 |
| **SWIN96-48** | 89.07 ± 10.51 | **93.24** | 86.69 | 94.29 | 30.60 | 97.45 |
| **SWIN96-60** | **89.32 ± 11.57** | 92.41 | 87.67 | 94.66 | 16.83 | 97.73 |

Table 5.12: Whole Tumor DICE performance for SwinUNETR models on the test set

**SwinUNETR models - Tumor Core DICE**

| Model | Mean ± Std | Median | Q1 | Q3 | Min | Max |
|---|---|---|---|---|---|---|
| **SWIN64-48** | 84.00 ± 13.49 | 88.46 | 80.29 | 93.57 | 36.09 | 97.39 |
| **SWIN96-48** | **85.24 ± 13.49** | 88.63 | 82.67 | 93.98 | 24.79 | 97.73 |
| **SWIN96-60** | 85.04 ± 15.59 | **89.43** | 81.64 | 94.08 | 5.37 | 97.77 |

Table 5.13: Tumor Core DICE performance for SwinUNETR models on the test set

**SwinUNETR models - Enhancing Tumor DICE**

| Model | Mean ± Std | Median | Q1 | Q3 | Min | Max |
|---|---|---|---|---|---|---|
| **SWIN64-48** | 73.77 ± 20.70 | 80.37 | 64.75 | 86.83 | 9.68 | 94.48 |
| **SWIN96-48** | 75.01 ± 21.08 | 81.80 | 70.24 | 89.76 | 5.37 | 95.14 |
| **SWIN96-60** | **76.16 ± 21.53** | **82.52** | 70.44 | 89.96 | 0.02 | 96.28 |

Table 5.14: Enhancing Tumor DICE performance for SwinUNETR models on the test set

**SwinUNETR models - Summary**

| Model | Whole Tumor | Tumor Core | Enhancing Tumor |
|---|---|---|---|
| **SWIN64-48** | 87.77 ± 10.19 | 84.00 ± 13.49 | 73.77 ± 20.70 |
| **SWIN96-48** | 89.07 ± 10.51 | **85.24 ± 13.49** | 75.01 ± 21.08 |
| **SWIN96-60** | **89.32 ± 11.57** | 85.04 ± 15.59 | **76.16 ± 21.53** |

Table 5.15: Summary of SwinUNETR model performance on the test set

From these scores, we can infer that training the model with a larger region of interest (ROI) may contribute to better generalization to unseen data. Specifically, the model *SWIN64-48*, trained using an ROI of $64 \times 64 \times 64$ voxels, yields lower results in all subregions compared to the other two models, which employ an ROI of $96 \times 96 \times 96$. This effect can be attributed to the inherent nature of Transformers, which leverage information from the entire volume under consideration, enabling more precise predictions when considering larger areas. Simultaneously, we observe that expanding the embedding space does not notably enhance performance for the Whole Tumor subregion. Instead, it slightly worsens the score for the Tumor Core, resulting in a higher median but lower mean and higher variance. However, this expansion proves beneficial in enhancing generalization for the Enhancing Tumor subregion. Now, let's compare our obtained results with the scores reported in the SwinUNETR paper [13]:

**SwinUNETR score - original paper**

| | Whole Tumor | Tumor Core | Enhancing tumor |
|---|---|---|---|
| **SwinUNETR** | 92.7 | 87.6 | 85.3 |

Table 5.16: DICE performance on test set reported in SwinUNETR paper [13]

Our models did not achieve the same level of performance as the model presented in the paper. The results differ slightly for what concerns the WT and TC subregions, but drop sensibly for the Enhancing Tumor subregion.

Considering that the paper's model is trained on the Brats2021 dataset, which includes 1251 cases for training and 219 patients for validation, our models, trained on less than a fifth of the patients, still demonstrate competitive results.

Moreover, our analysis involved a subset of the original test set. In this subset, patients with exceptionally small enhancing tumor regions may introduce outliers in the computation of the DICE metric. The DICE metric, particularly sensitive to classes that present few voxels, can significantly impact the metric even with a minimal error in voxel prediction, thereby influencing a lower overall mean. This phenomenon is attributed to the inherent mathematical characteristics of the DICE score. In instances where a specific subregion has minimal extent, even slight errors can disproportionately impact the score due to their significant influence on the ratio between true positives, false negatives, and false positives.

In particular, when the region is not present in the ground truth, two scenarios may arise:

- If there is at least one pixel/voxel of the class incorrectly predicted, the DICE formula will yield 0 (true positives) in the numerator, resulting in a null DICE score, even with just one false positive.
- if the model accurately predicts the absence of that class, we will have True Positives, False Positives, and False Negatives all set to 0 simultaneously. This condition causes the DICE formula to have 0 in both the numerator and denominator, resulting in a not-a-number value.

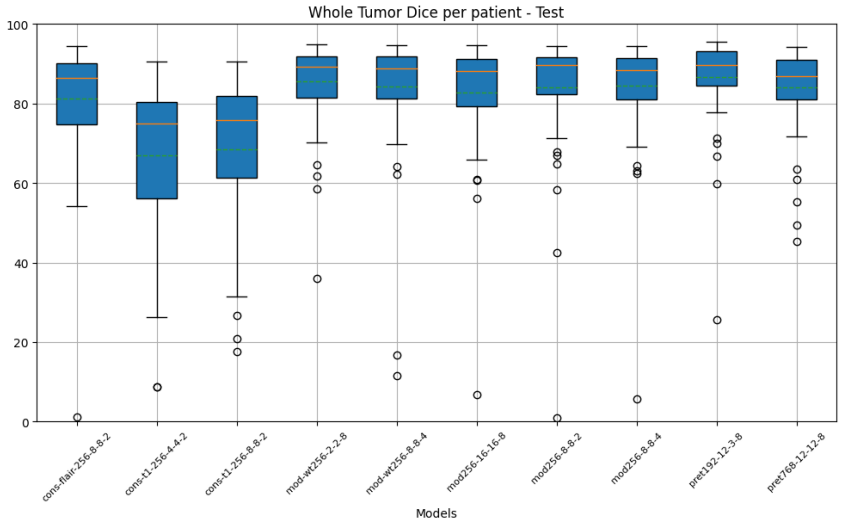
This suggests that regardless of the model's overall ability to predict that subregion, the metric will consistently be 0 or undefined, significantly influencing the average score and making it challenging to interpret. Consequently, the computation of the DICE score in such cases is entirely meaningless.

To address this issue, we opted to exclude from the test set the patients for whom a subregion is not present in the ground truth. Specifically, we removed the $46^{th}$ and $56^{th}$ patients from the test set, which ground truth has no voxels associated to the Enhancing Tumor class, reducing the number of test patients from 59 to 57.

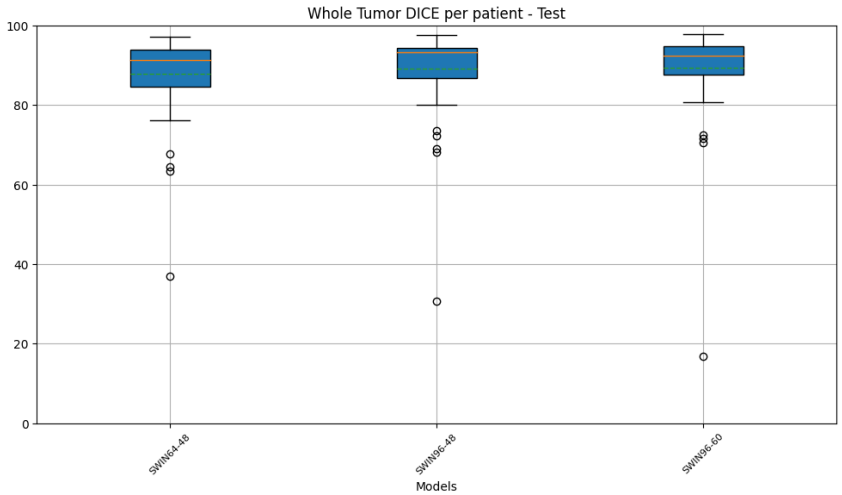
Considering the factors we've discussed, and given the uncertainty surrounding the class composition in the ground truth of each Brats2021 test patient, it's crucial to perceive the comparison between our scores and the one reported in the original paper as indicative rather than conclusive.

### 5.2.3. Segmenter vs SwinUNETR: result comparison

Upon analyzing the outcomes on the test set, it becomes apparent that, in general, the SwinUNETR models surpass the performance of Segmenter models across all subregions. This superiority is particularly pronounced in the case of the Enhancing Tumor subregion, where the enhancement is notably significant. In order to provide a visual comparison, in the upcoming figures we showcase box plots illustrating the distribution of per-patient DICE test scores for the Segmenter and SwinUNETR models:

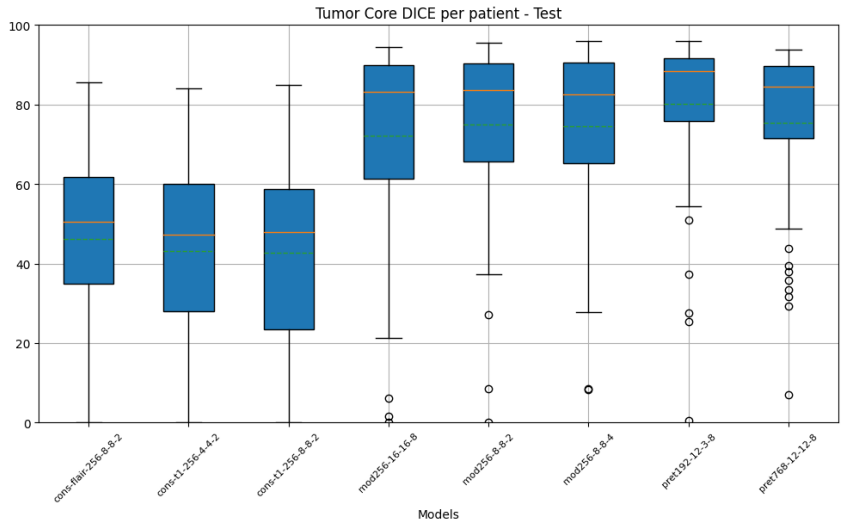



(a) Segmenter models

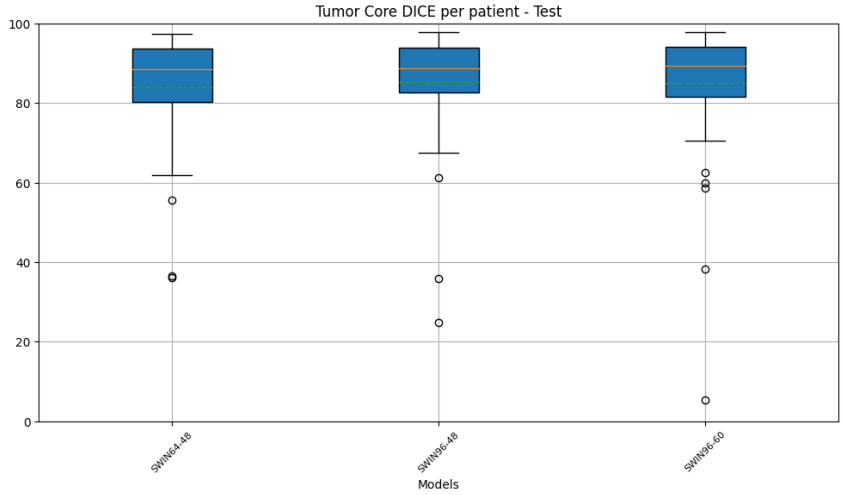



(b) SwinUNETR models

Figure 5.3: Box plot - Whole Tumor DICE metric on test set

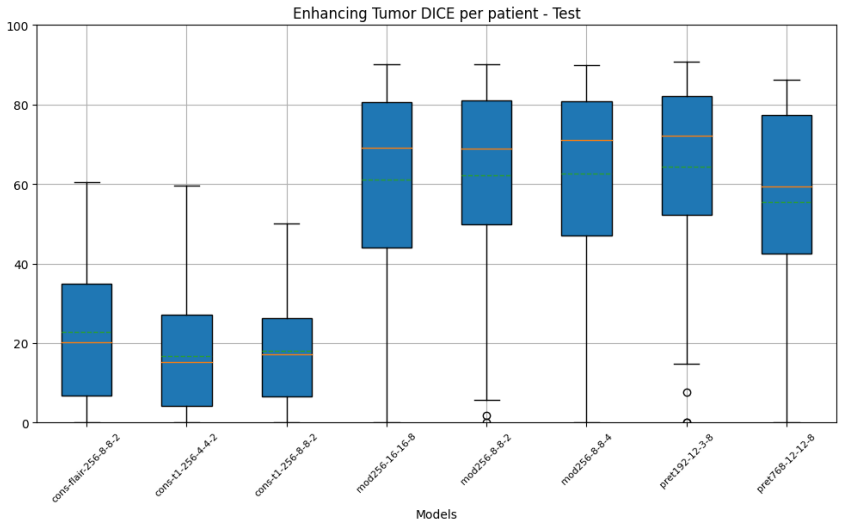



(a) Segmenter models

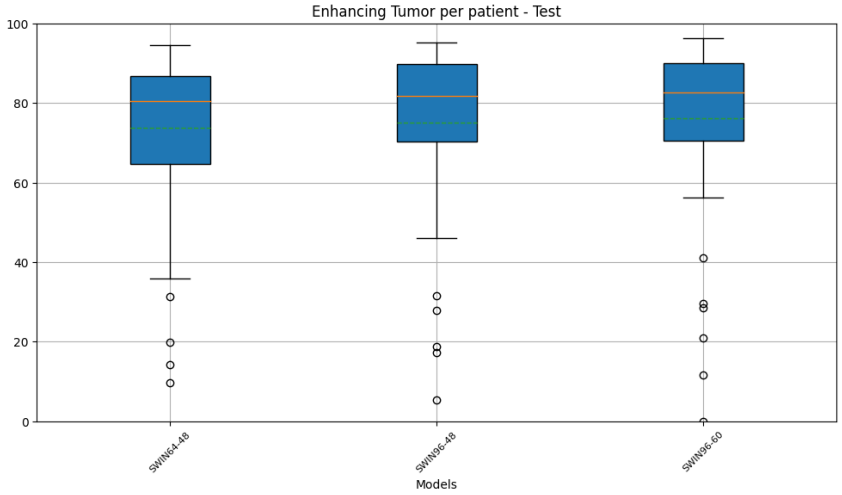



(b) SwinUNETR models

Figure 5.4: Box plot - Tumor Core DICE metric on test set




(a) Segmenter models




(b) SwinUNETR models

Figure 5.5: Box plot - Enhancing Tumor DICE metric on test set

A more detailed examination of the box plots yields additional insightful findings. When comparing the distribution of per-patient scores between SwinUNETR and Segmenter models for the Whole Tumor, Tumor Core, and Enhancing Tumor subregions, a distinct pattern emerges. SwinUNETR models exhibit significantly less variance in their score distribution, coupled with a noticeable reduction in the occurrence of outliers. Interestingly, in cases where outliers are present, they tend to achieve higher scores on average.

This phenomenon is especially prominent in the Tumor Core and Enhancing Tumor subregions. The box plots for SwinUNETR models depict a more tightly clustered distribution with fewer extreme values, indicating a higher consistency in predicting tumor areas within

individual patients compared to Segmenter models.

The comparative analysis leads to a notable conclusion: SwinUNETR models demonstrate an improved capability to predict tumor areas at both global and individual patient levels. The reduced variance, fewer outliers, and consistently higher scores collectively signify a more robust performance in delineating tumor regions for individual patients.

### 5.2.4. Transformers vs CNNs: result comparison

In this subsection, we'll compare the results of our experiments with various state-of-the-art architectures. Our focus is on comparing the Transformer-based models' outcomes with those from related works that adopt diverse strategies within the context of the BraTS 2019 challenge. In particular, while some of these models rely exclusively on CNNs, others employ a hybrid approach.

For this comparison, we will consider the following studies:

- The paper from Andriy Myronenko and Ali Hatamizadeh [23], which introduces a method employing a typical U-Net encoder-decoder structure for analyzing three-dimensional scans.
- The study conducted by Xiangyu Li, Gongning Luo, and Kuanquan Wang [16], which presents an innovative approach using multi-step cascaded networks in sequence. The initial network reconstructs the Whole Tumor segmentation map of a random cropped sub-volume of the input scan, utilizing it as a filter for the input T1ce image. Subsequently, the second segmentation network takes as input the filtered image, which comprises only tumoral tissue, and focuses on the sub-nested Tumor Core region, which, in turn, guides the final segmentation network to identify the Enhancing Tumor subregion. Each segmentation network consists in a classical U-Net based on CNNs.
- The work of Rupal R. Agravat and Mehul S. Raval [2], that proposes a fully convolutional neural network designed for segmenting 2D slices.
- The 3D U-Net architecture [32] proposed by Feifan Wang et Al.
- The TransBTS [33], an approach that adopts a Transformer architecture as bottleneck of a U-Net encoder-decoder structure for the analysis of 3D MRI scans.
- The SegTran [15], which uses a CNN to extract features of MRI scans and feeds them to a series of Transformer layers that implement a peculiar technique called Squeeze-and-Expansion.

**Brats2019 DICE results comparison**

| | WT | TC | ET |
|---|---|---|---|
| cons-t1-256-4-4-2 | 67.47 | 43.11 | 16.85 |
| cons-t1-256-8-8-2 | 69.21 | 42.63 | 17.96 |
| cons-flair-256-8-8-2 | 81.26 | 46.23 | 22.89 |
| mod256-8-8-2 | 84.75 | 75.08 | 62.28 |
| mod256-8-8-4 | 84.81 | 74.61 | 62.70 |
| mod256-16-16-8 | 83.11 | 72.06 | 61.18 |
| pret192-12-3-8 | **86.56** | **80.14** | **64.47** |
| pret768-12-12-8 | 83.91 | 75.45 | 55.47 |
| mod-wt256-2-2-8 | 85.94 | - | - |
| mod-wt256-8-8-4 | 85.47 | - | - |
| SWIN96-48 | 89.07 | **85.24** | 75.01 |
| SWIN64-48 | 87.77 | 84.00 | 73.77 |
| SWIN96-60 | **89.32** | 85.04 | **76.16** |
| 3D - Myron. [23] | 88.20 | **83.70** | **82.60** |
| 3D - Xiangyu* [16] | **88.60** | 81.30 | 77.10 |
| 2D - Rupal* [2] | 73.00 | 65.00 | 59.00 |
| 3D U-Net [32] | 85.20 | 79.80 | 77.80 |
| TransBTS* [33] | **90.00** | **81.94** | **78.93** |
| SegTran* [15] | 89.50 | 81.70 | 74.00 |

Table 5.17: Comparison on DICE performance on Brats2019 challenge

The models highlighted in blue correspond to our Segmenter models, while the green ones represent our SwinUNETR models. Additionally, those marked in orange are purely CNN-based models, and the red ones signify hybrid models that integrate both CNNs and Transformers. For models marked with an asterisk (*), the studies introducing them exclusively reported scores on the validation set, omitting results on the test set. We have included these validation scores in the table.

Before delving into the interpretation of the reported scores, it's crucial to emphasize the complexity of making direct comparisons across models from different papers. Our models were tested on a subset comprising 57 patients from the test set. Importantly, we lack information about the class distribution across the full test set (166 patients) for various patients. Therefore, these values should be considered indicative and may require further investigation. Notably, the reported models were trained with MRIs from 335 patients, in contrast to the 227 patients (approximately two-thirds) used for training SwinUNETR and Segmenter models. Additionally, the reported averages across all patients lack information about the variance of the score distributions, which is crucial for a fair comparison.

Given these considerations, it appears that our SwinUNETR models outperform all Segmenter and CNN-based models in predicting WT and TC subregions. However, 3D CNN-based approaches still seem to outperform this algorithm in the Enhancing Tumor subregion. Segmenter models, utilizing bi-dimensional slices, face challenges in achieving performances comparable to 3D CNN models and hybrid models, especially in the Tumor Core and Enhancing Tumor subregions. Nevertheless, the performance of the *pret192-12-3-8* pre-trained model on the Whole Tumor and Tumor Core is relatively comparable to those of 3D U-Nets. On the other hand, Segmenter models trained with modality merging techniques significantly outperform the U-Net model that uses bi-dimensional slices.

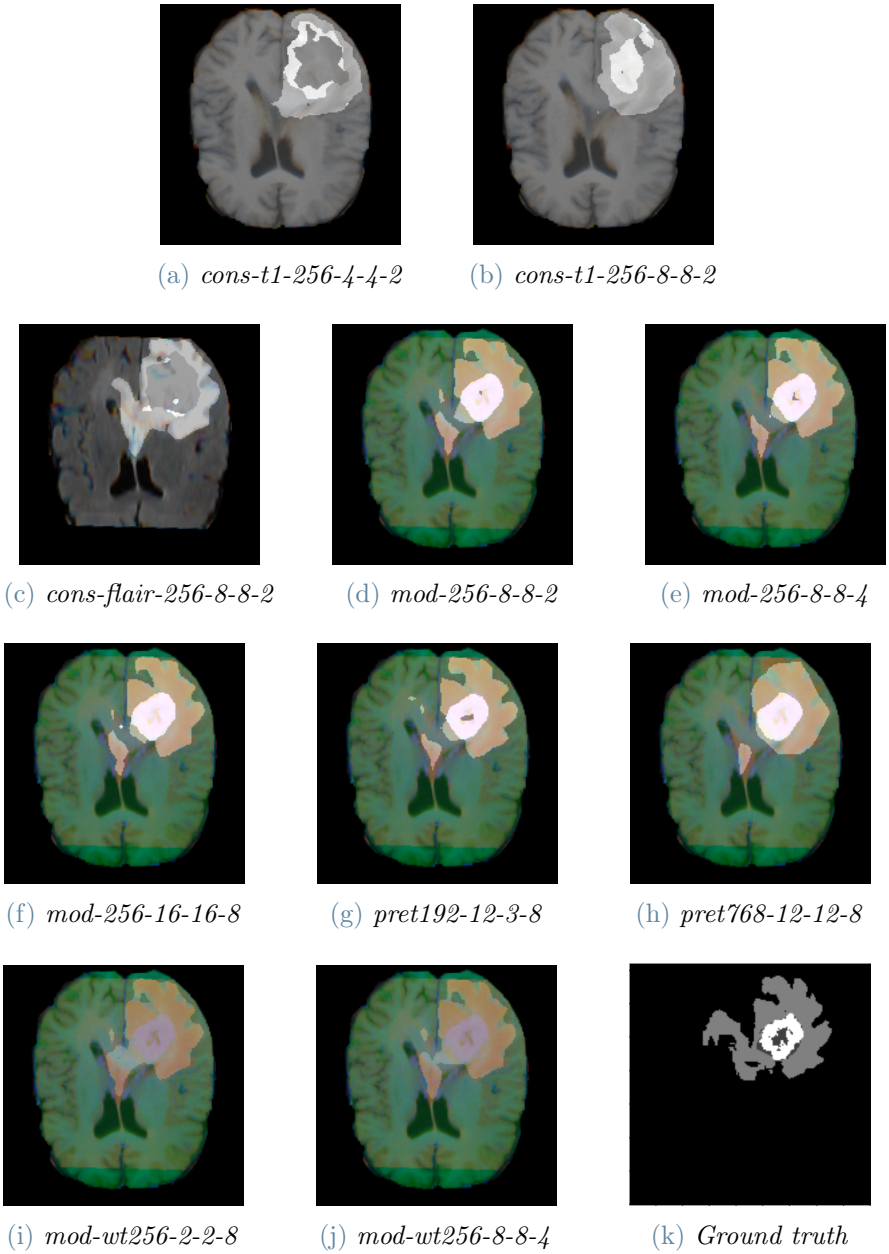
Furthermore, SwinUNETR has demonstrated superior performance on the Tumor Core subregion compared to hybrid models. However, it doesn't seem to clearly outperform these models in the Whole Tumor and Enhancing Tumor subregions. In the case of hybrid models, the Segmenter model *pret192-12-3-8* doesn't have comparable results in the Whole Tumor class but shows a similar performance in the Tumor Core. However, it's important to note that the reported scores for TransBTS and SegTran are computed on the validation set, and therefore, may not be entirely unbiased.

Among purely CNN-based methods, the methodology proposed by Andriy Myronenko and Ali Hatamizadeh emerges as the sole approach surpassing hybrid models in both the Tumor Core and Enhancing Tumor subregions. Furthermore, it demonstrates a close competition with SwinUNETR models, particularly in the Whole Tumor and Tumor Core, ultimately outperforming them drastically in the case of the Enhancing Tumor subregion.

## 5.3. Qualitative results

In this section we explore some visual examples of the performances of the trained models. Now we present the predictions, for every trained model, of the slices 73 and 200:

### 5.3.1. Slice 73, patient 1



(a) *cons-t1-256-4-4-2* (b) *cons-t1-256-8-8-2*

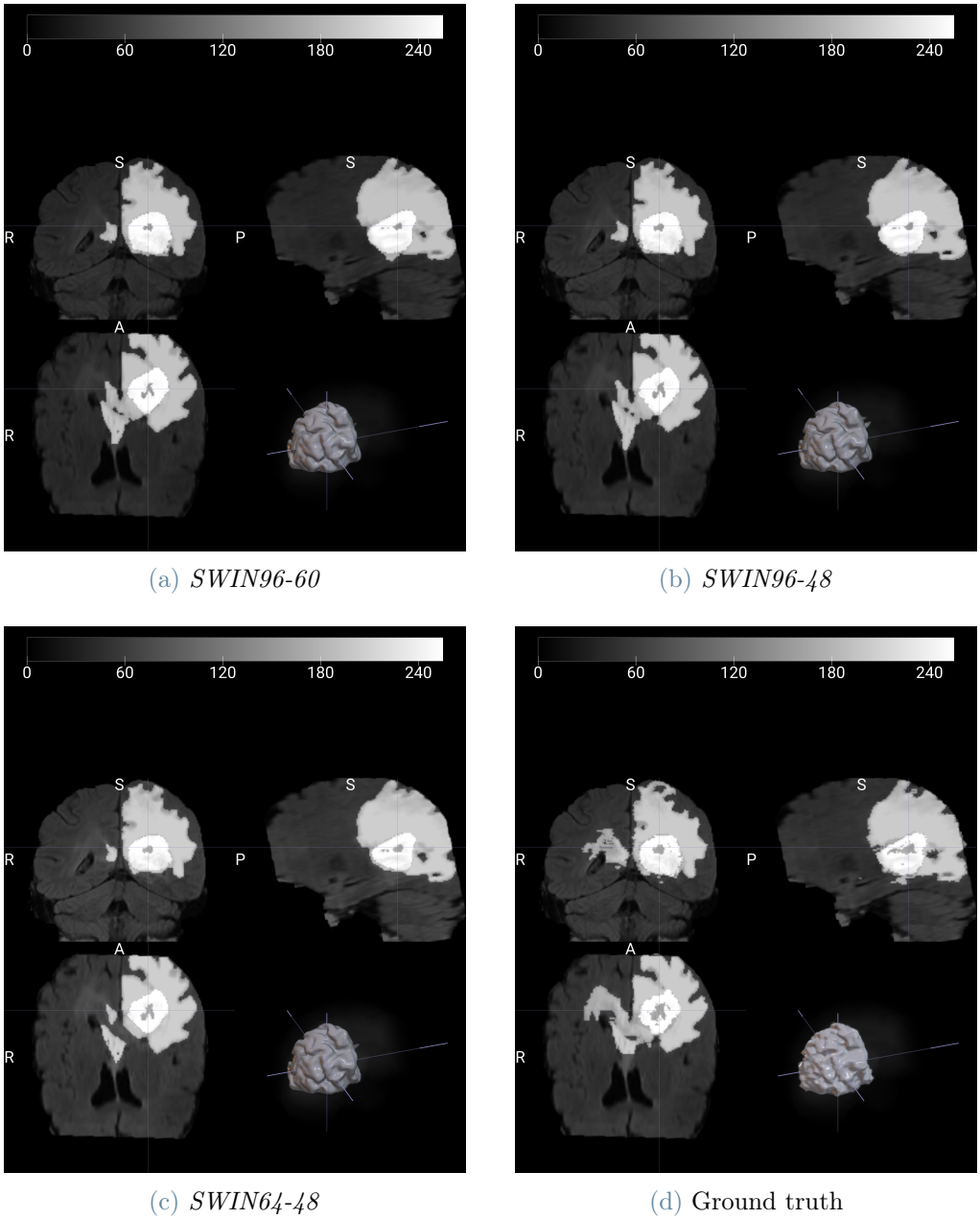
(c) *cons-flair-256-8-8-2* (d) *mod-256-8-8-2* (e) *mod-256-8-8-4*

(f) *mod-256-16-16-8* (g) *pret192-12-3-8* (h) *pret768-12-12-8*

(i) *mod-wt256-2-2-8* (j) *mod-wt256-8-8-4* (k) *Ground truth*

Figure 5.6: Segmenter models prediction on slice 73

In the context of SwinUNETR models, it's crucial to highlight that these models produce comprehensive 3D segmentation maps instead of individual slices. To ensure a meaningful comparison with the Segmenter models, we have selected the $73^{rd}$ slice on axial plane on the first patient's predictions, and adjusted the coronal and sagittal planes to center around the tumor.

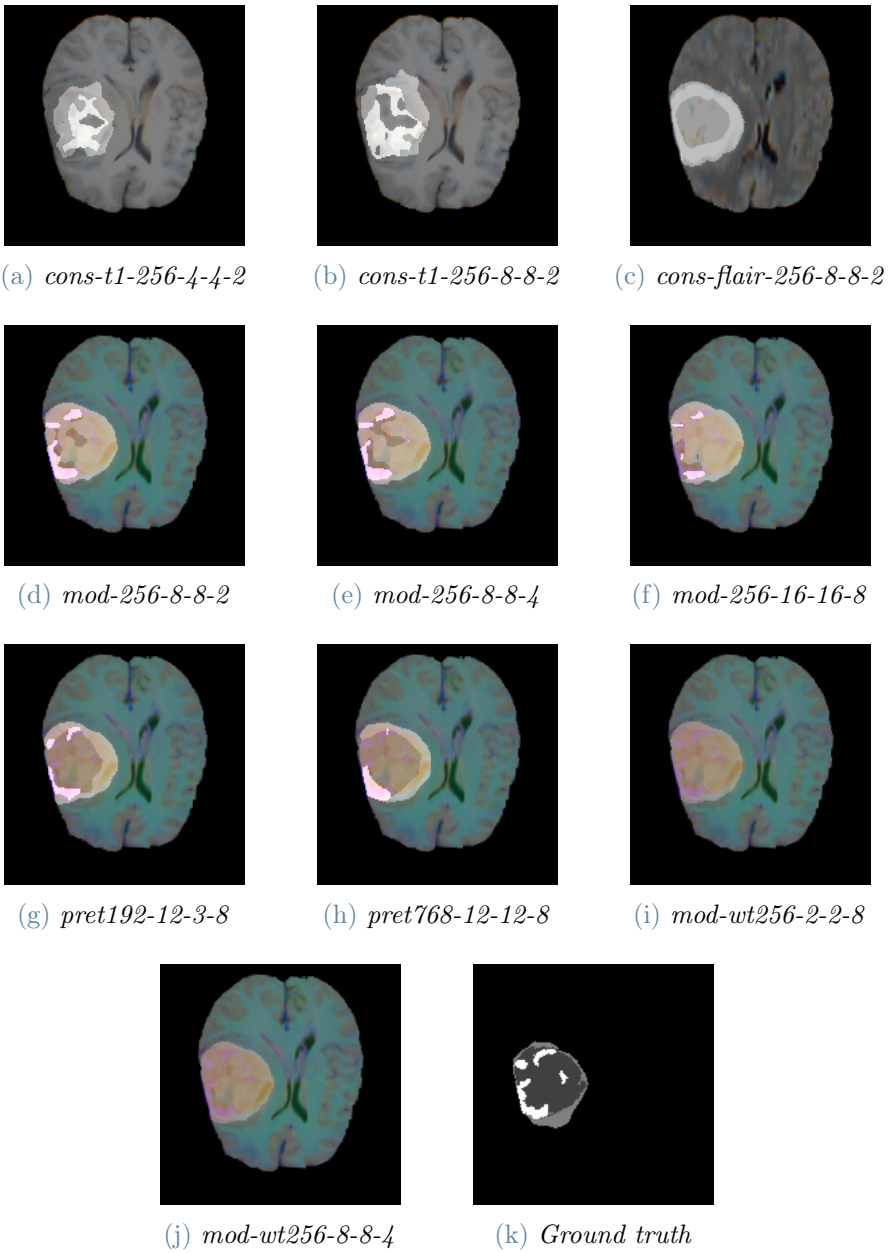



(a) *SWIN96-60* (b) *SWIN96-48*

(c) *SWIN64-48* (d) Ground truth

Figure 5.7: SwinUNETR models prediction on slice $67 \times 126 \times 73$

### 5.3.2. Slice 200, patient 2



(a) *cons-t1-256-4-4-2* (b) *cons-t1-256-8-8-2* (c) *cons-flair-256-8-8-2*

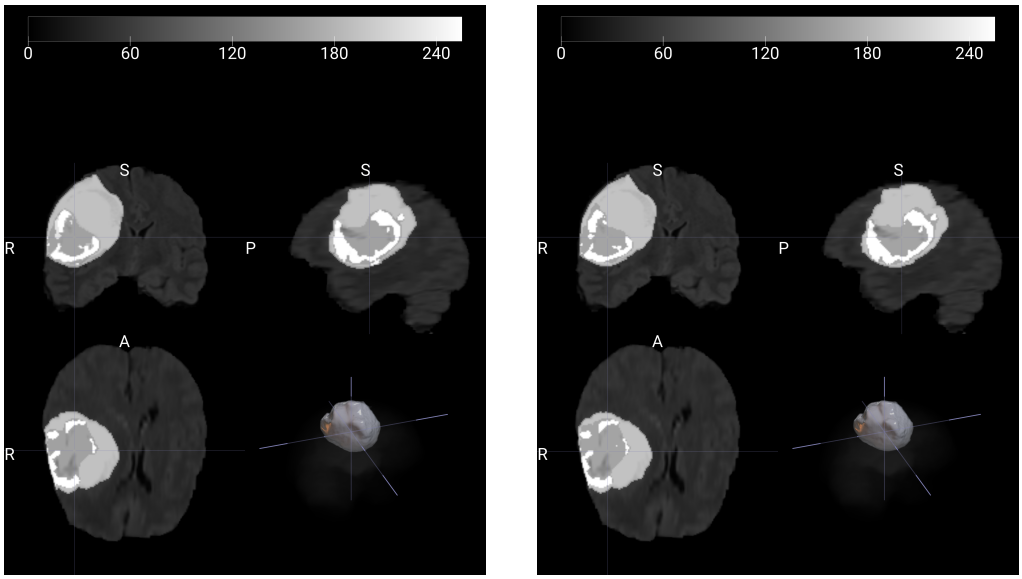
(d) *mod-256-8-8-2* (e) *mod-256-8-8-4* (f) *mod-256-16-16-8*

(g) *pret192-12-3-8* (h) *pret768-12-12-8* (i) *mod-wt256-2-2-8*

(j) *mod-wt256-8-8-4* (k) *Ground truth*

Figure 5.8: Segmenter models predictions on slice 200

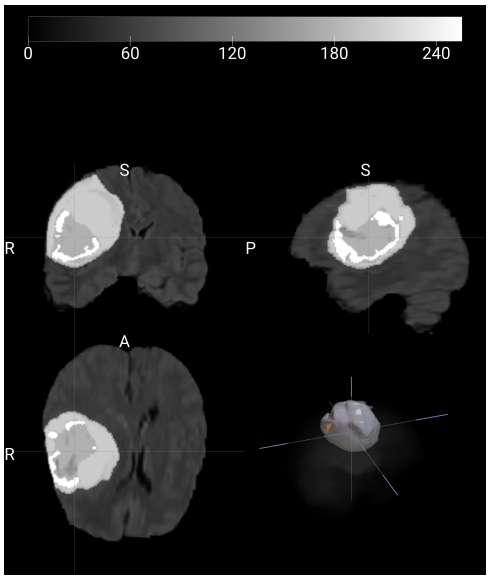
In the 3D dataset, slice 200 corresponds to the $72^{nd}$ slice of the second patient ($72 + 128 = 200$). Therefore, we sliced the predictions at $z = 72$ on the axial plane and aligned the sagittal and coronal planes to the center of the tumor.



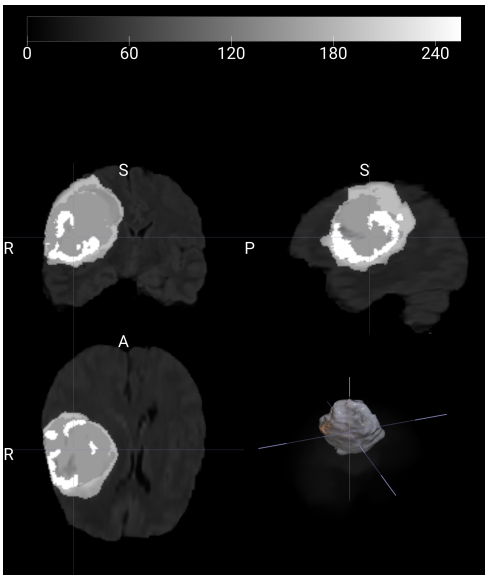

(a) *SWIN96-60*




(b) *SWIN96-48*




(c) *SWIN64-48*



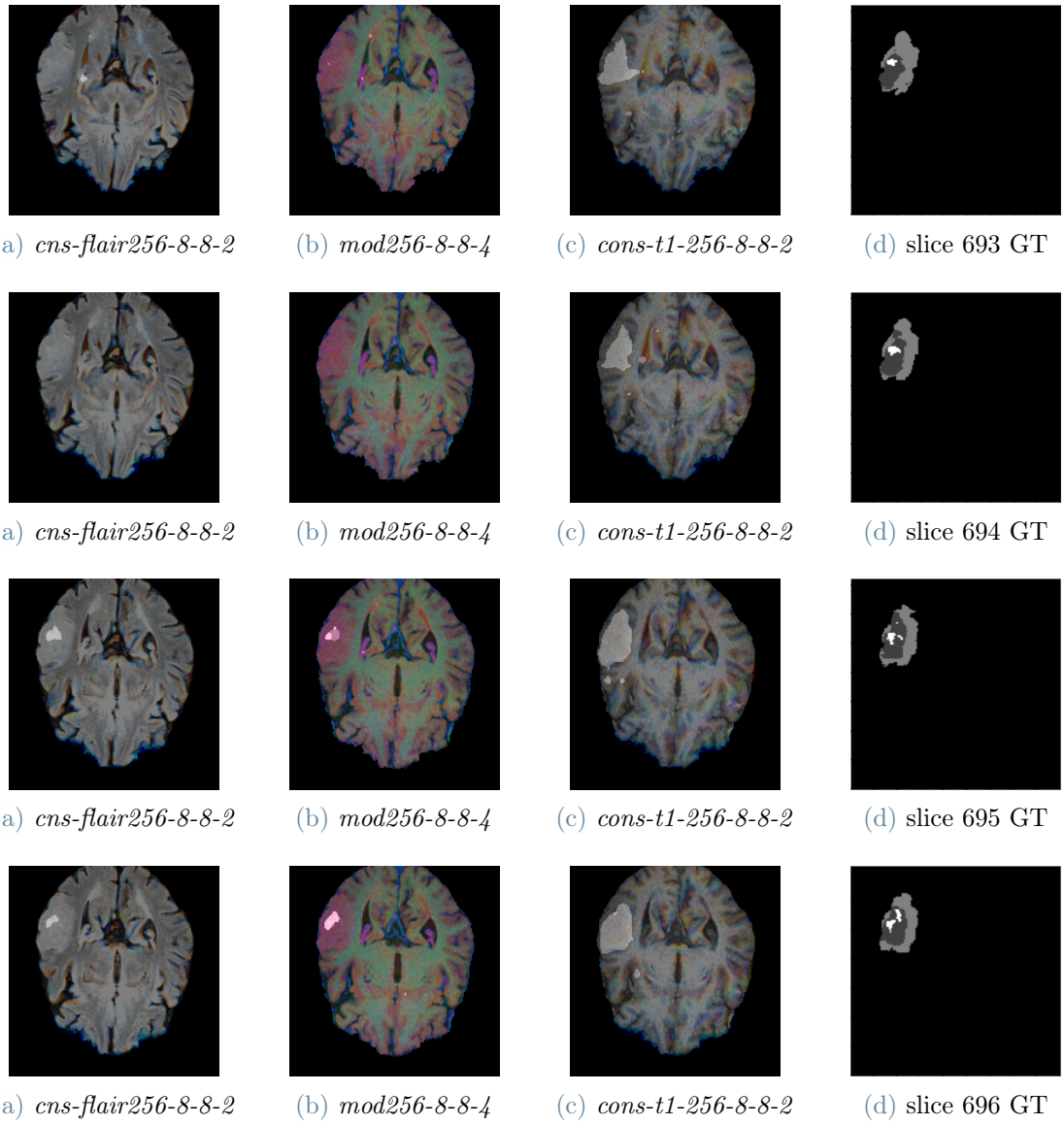

(d) Ground truth

Figure 5.9: SwinUNETR models prediction on slice $127 \times 92 \times 72$

## 5.4. Edge cases

Upon examining the scores per patient in detail, it is evident that some patients present some challenges, consistently yielding lower scores across various models.

As an example, let's consider some slices of the sixth patient in the test set:



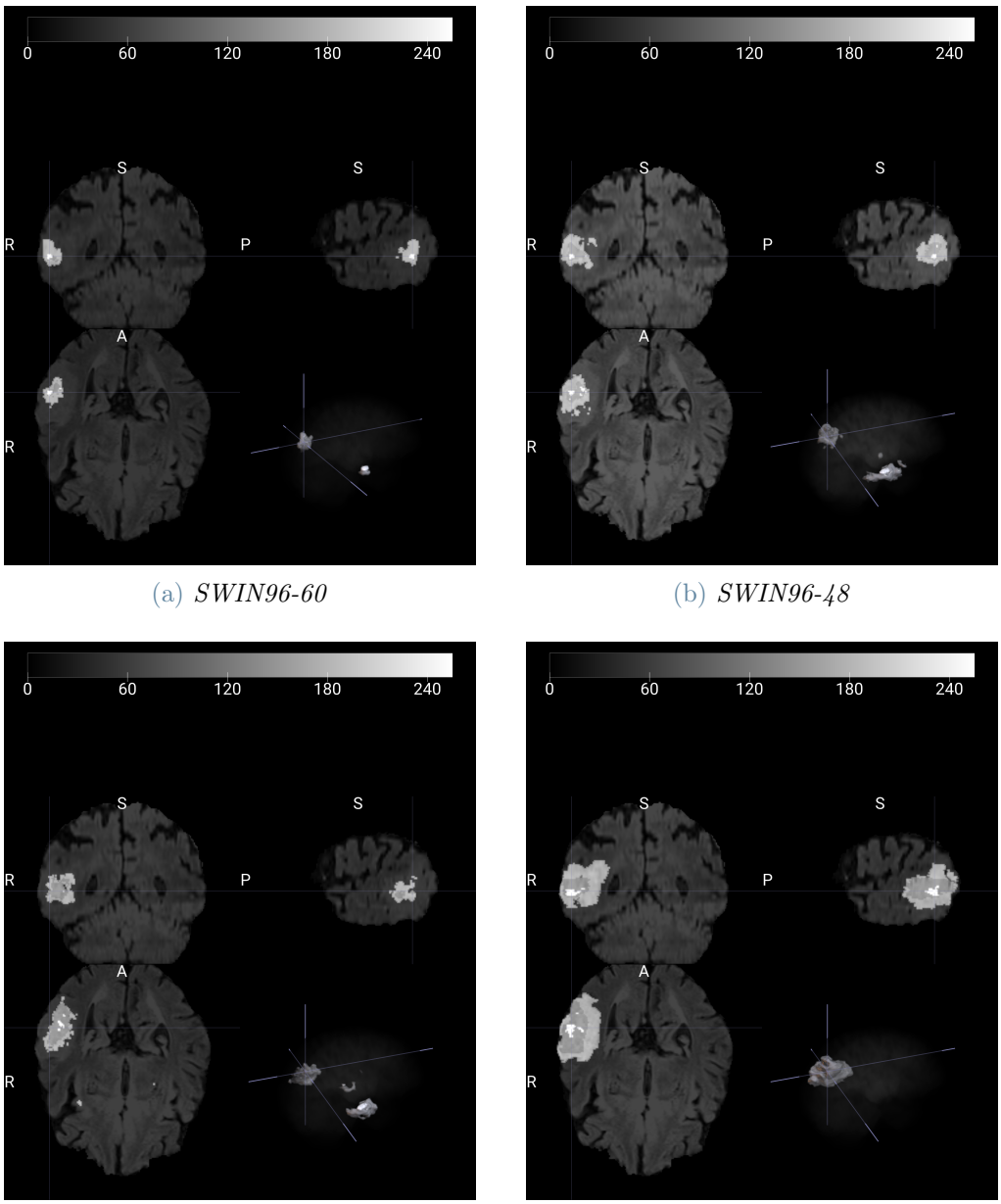
(a) *cns-flair256-8-8-2* (b) *mod256-8-8-4* (c) *cons-t1-256-8-8-2* (d) slice 693 GT

(a) *cns-flair256-8-8-2* (b) *mod256-8-8-4* (c) *cons-t1-256-8-8-2* (d) slice 694 GT

(a) *cns-flair256-8-8-2* (b) *mod256-8-8-4* (c) *cons-t1-256-8-8-2* (d) slice 695 GT

(a) *cns-flair256-8-8-2* (b) *mod256-8-8-4* (c) *cons-t1-256-8-8-2* (d) slice 696 GT

Figure 5.10: Predictions of the Segmenter models for slices 693-696, pertaining to the sixth test patient




(a) *SWIN96-60* (b) *SWIN96-48*

(c) *SWIN64-48* (d) Ground truth

Figure 5.11: SwinUNETR models predictions on slice 145 × 131 × 55 of sixth patient

Regarding the Segmenter models, for this comparison we chose the models *cons-flair-256-8-8-2*, *mod256-8-8-4*, and *cons-t1-256-8-8-2*. However, all the other models exhibited either similar or inferior performances. At the end of each row, the ground truth segmentation is presented. Notably, none of the three models under examination successfully predicts the necrotic region. However, *cons-t1-256-8-8-2* demonstrates a relatively better performance in capturing the edematous region, albeit with occasional inaccuracies.

This clarifies why the T1 model achieves superior performance in the Whole Tumor category (see Table 5.18): despite occasional inaccuracies in predicting the edematous region, where the ground truth shows instead a necrotic area, the model is still capable of distinguishing the tumor area from the background. In the central slices, which feature a necrotic region, *mod256-8-8-4* demonstrates the capability to predict a small part of the enhancing tumor region, consequently leading to a marginally better Tumor Core score as well.

The suboptimal performance of the models on this particular patient can be attributed to distinctive features present in its slices, such as a significantly small enhancing tumor region with respect to other patients. This assertion is further supported by the observation that SwinUNETR models also exhibited poor performance when predicting tumor regions for this patient.

The first two 3D models encounter challenges in accurately identifying the tumor cluster. In fact, they even identify two separate tumor areas across the brain, a distinction that would have been evident only through an analysis of a much larger number of slices in the case of the slice dataset. The third model, *SWIN64-48*, goes a step further by erroneously identifying three distinct tumor regions. Of course, these predictions are incorrect, as the ground truth clearly indicates that the tumor is not divided into multiple regions. Nevertheless, SwinUNETR models, while not performing as well as *cons-t1-256-8-8-2* in the Whole Tumor subregion, achieve better results in identifying the Tumor Core.

**Sixth test patient DICE score**

| | WT | TC | ET | Necrosis | Edema |
|---|---|---|---|---|---|
| **cons-flair-256-8-8-2** | 1.12 | 0.0 | 0.0 | 0.0 | 0.0 |
| **mod256-256-8-8-4** | 5.77 | 8.38 | 28.63 | 0.0 | 1.43 |
| **cons-t1-256-8-8-2** | 65.89 | 5.65 | 0.0 | 0.0 | 41.60 |
| **SWIN96-60** | 16.83 | 5.36 | 11.71 | - | - |
| **SWIN96-48** | 30.59 | 24.78 | 17.30 | - | - |
| **SWIN64-48** | 36.99 | 36.53 | 9.68 | - | - |

Table 5.18: DICE performances on sixth test patient

# 6 | Conclusions and future developments

In this concluding chapter, we draw insightful considerations from the results discussed in this thesis.

A crucial insight emerges: SwinUNETR models, harnessing the potential of 3D MRI scans for brain analysis, exhibit a heightened level of robustness and accuracy in predicting tumor subregions compared to the Segmenter models, which focus on bi-dimensional slices.

This improved performance can be attributed to several key factors. Firstly, 3D scans provide a comprehensive overview of the entire brain. The Transformer encoder's proficiency in capturing long-range distance dependencies emerges as a pivotal element contributing to the superior prediction of tumor areas.

Furthermore, SwinUNETR models employ all four modalities of MRI scans – FLAIR, T1, T1ce, and T2. In contrast, the Segmenter models make use of only one modality in the case of consequent slice merging techniques or just three modalities in the case of modality merging.

This aspect gains even more significance when considering that two Segmenter models employed pre-trained backbones, while the SwinUNETR models were trained entirely from scratch. This underscores the robustness and adaptability inherent in the SwinUNETR architecture. Interestingly, despite the modest performance of the consequent slice merging technique on the test set, it provides a valuable insight. Depending on the modality under consideration, the performance on each class exhibits significant variations, offering a nuanced perspective on the impact of modality selection in the segmentation process.

Moreover, in the segmentation of Tumor Core and Enhancing Tumor subregions, it becomes evident that the key to achieving favorable outcomes lies in the concurrent analysis of multiple modalities.

Another noteworthy point is that, as discussed in subsection 2.2.4, effective regularization

and data augmentation play a crucial role in enhancing model performance.

Regarding the comparison between Transformer-based models and CNN-based models, additional considerations come into play.

Firstly, it's evident that the primary determinant for achieving successful results in brain tumor segmentation is the utilization of three-dimensional scans, as opposed to single-slice images. On average, 3D CNN models outperform the Segmenter, while the considered 2D U-Net yields considerably lower results.

Secondly, when comparing various 3D architectures, models based on Transformers outperform CNN-based models on the Whole Tumor and Tumor Core subregions, still facing challenges in the accuracy of the prediction of the smaller Enhancing Tumor subregions. In this context, hybrid models exhibit a somewhat intermediate performance.

## 6.1. Final conclusions

From these observations, we can distill the findings of this thesis into the following conclusions:

- The simultaneous analysis of different MRI modalities serves as the foundational element for achieving favorable results in all subregions.
- Models employing three-dimensional input generally outperform those using single slices. On the other hand, the use of transfer learning on bi-dimensional models can help to bridge the gap between these approaches.
- In the medical image field, which is characterized by a limited data availability, we observe that when using the same approach (slice or volumetric dataset) Transformer models challenge CNNs, outperforming them in the prediction of Whole Tumor and Tumor Core subregions.

On these premises, we can discuss some future developments for this work:

- Consequent-slice merging technique across all modalities: investigating the consequent-slice merging technique across all MRI modalities could be a compelling area of research. Training models specifically for the segmentation of distinct classes or subregions based on the unique features inherent in each modality might yield valuable insights.
- Exploration of modality combinations: while this thesis utilized FLAIR, T1, and T1ce modalities for their informative content, further analysis could delve into ex-

ploring other combinations of modalities. Understanding which modality combinations are more effective for the segmentation task could enhance our comprehension.

- Refinement of loss computation in Segmenter models: the Segmenter models employed in this work computed the loss using scores for all classes, including the background. A refinement could involve experimenting with loss computation excluding the background to evaluate its impact. Alternatively, exploring loss computation on subregions rather than individual classes could provide valuable insights.

- Exploration of various patch sizes in Segmenter models: in the course of this thesis, the majority of Segmenter models were configured with a patch size of $6 \times 6$ pixels. This choice was influenced by memory constraints that prevented the utilization of smaller patch sizes. However, a noteworthy avenue for exploration lies in assessing the implications of employing smaller patch sizes during both training and prediction phases. This investigation could shed light on potential improvements or trade-offs associated with different patch sizes, providing valuable insights into model performance and generalization capabilities.

- Hyperparameter exploration for SwinUNETR: the SwinUNETR models in this study adhered to the architecture proposed in the original paper without variations in the number of layers, patch size, or window size. Future exploration could involve varying these hyperparameters to identify optimal combinations for the specific segmentation task at hand.

- Pre-trained backbones for SwinUNETR models: while our SwinUNETR models were trained from scratch, an intriguing avenue involves incorporating pre-trained backbones. Evaluating the performance enhancement achieved through the utilization of transfer learning could be an interesting direction.

- Optimization of regularization parameters: a potential avenue for improvement involves systematically adjusting regularization parameters in both SwinUNETR and Segmenter models. Identifying an optimal combination of regularization parameters could contribute to maximizing models performance.

These proposed directions offer a comprehensive roadmap for future research, aiming to refine and advance the capabilities of the models presented in this thesis.

# Bibliography

[1] K. Academy. Magnetic resonance imaging (mri). `https://www.khanacademy.org/test-prep/mcat/physical-processes/proton-nuclear-magnetic-resonance/a/magnetic-resonance-imaging-mri`, n.d.

[2] R. R. Agravat and M. S. Raval. Brain tumor segmentation and survival prediction. In *International MICCAI Brainlesion Workshop*, pages 338–348. Springer, 2019. doi: arXiv:1909.09399v1.

[3] U. Baid, S. Ghodasara, S. Mohan, M. Bilello, E. Calabrese, E. Colak, K. Farahani, J. Kalpathy-Cramer, F. C. Kitamura, S. Pati, et al. The rsna-asnr-miccai brats 2021 benchmark on brain tumor segmentation and radiogenomic classification. *arXiv preprint*, 2021. doi: arXiv:2107.02314v1.

[4] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, and J. K. et al. Segmentation labels and radiomic features for the pre-operative scans of the tcga-gbm collection. *The Cancer Imaging Archive*, 2017. doi: 10.7937/K9/TCIA.2017.KLXWJJ1Q.

[5] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, and J. K. et al. Segmentation labels and radiomic features for the pre-operative scans of the tcga-lgg collection. *The Cancer Imaging Archive*, 2017. doi: 10.7937/K9/TCIA.2017.GJQ7R0EF.

[6] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J. S. Kirby, J. B. Freymann, K. Farahani, and C. Davatzikos. Advancing the cancer genome atlas glioma mri collections with expert segmentation labels and radiomic features. *Nature Scientific data*, 4(1):1–13, 2017. doi: 10.1038/sdata.2017.117.

[7] S. Bakas, M. Reyes, A. Jakab, S. Bauer, M. Rempfler, A. Crimi, R. T. Shinohara, C. Berger, S. M. Ha, M. Rozycki, et al. Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the brats challenge. *arXiv preprint*, 2018. doi: arXiv:1811.02629.

[8] A. Berger. Magnetic resonance imaging. *BMJ*, 2002. doi: 10.1136/bmj.324.7328.35.

[9] Y. L. Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, and L. D. J.

W. Hubbard. *Handwritten Digit Recognition with a Back-Propagation Network*, page 396–404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990. ISBN 1558601007.

[10] M. David C. Preston. Magnetic resonance imaging (mri) of the brain and spine: Basics. `https://case.edu/med/neurology/NR/MRI%20Basics.htm`, 2016.

[11] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: transformers for image recognition at scale. *arXiv preprint*, 2021. doi: arXiv:2010.11929v2.

[12] F. Gaillard. Case courtesy of frank gaillard, radiopaedia.org, rid: 27812. `https://radiopaedia.org/articles/glioblastoma-idh-wildtype`, 2023.

[13] A. Hatamizadeh, V. Nath, Y. Tang, D. Yang, H. R. Roth, and D. Xu. Swin unetr: Swin transformers for semantic segmentation of brain tumors in mri images. In *International MICCAI Brainlesion Workshop*, pages 272–284. Springer, 2021. doi: arXiv:2201.01266v1.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] S. Li, X. Sui, X. Luo, X. Xu, Y. Liu, and R. Goh. Medical image segmentation using squeeze-and-expansion transformers. *arXiv preprint*, 2021. doi: arXiv:2105.09511v3.

[16] X. Li, G. Luo, and K. Wang. Multi-step cascaded networks for brain tumor segmentation. In *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: 5th International Workshop, BrainLes 2019, Held in Conjunction with MICCAI 2019, Shenzhen, China, October 17, 2019, Revised Selected Papers, Part I 5*, pages 163–173. Springer, 2020. doi: arXiv:1908.05887v3.

[17] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[18] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2015.

[19] Mathworks. Overfitting, n.d. URL `https://it.mathworks.com/discovery/overfitting.html`.

[20] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest, et al. The multimodal brain tumor image segmentation benchmark (brats). *IEEE transactions on medical imaging*, 34(10): 1993–2024, 2014. doi: 10.1109/TMI.2014.2377694.

[21] T. M. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997. ISBN 0070428077.

[22] MONAI. Monai transforms, n.d. URL `https://docs.monai.io/en/latest/transforms.html`.

[23] A. Myronenko and A. Hatamizadeh. Robust semantic segmentation of brain tumor regions from 3d mris. In *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: 5th International Workshop, BrainLes 2019, Held in Conjunction with MICCAI 2019, Shenzhen, China, October 17, 2019, Revised Selected Papers, Part II 5*, pages 82–89. Springer, 2020. doi: arXiv:2001.02040v1.

[24] n.d. Flair mri. `https://mrimaster.com/characterise-image-flair/`, n.d..

[25] n.d. T2 weighted mri. `https://mrimaster.com/characterise-image-t2/`, n.d..

[26] A. A. of Neurological Surgeons. Brain tumors. `https://www.aans.org/en/Patients/Neurosurgical-Conditions-and-Treatments/Brain-Tumors`, 2023.

[27] Radiopaedia.org. T1 weighted image. `https://radiopaedia.org/articles/t1-weighted-image`, 2016.

[28] M. Raghu, T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in Neural Information Processing Systems*, 34:12116–12128, 2021.

[29] A. Steiner and A. Kolesnikov. How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv:2106.10270v1*, 2021. doi: arXiv:2106.10270v1.

[30] R. Strudel, R. Garcia, I. Laptev, and C. Schmid. Segmenter: Transformer for semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7262–7272, 2021. doi: arXiv:2105.05633.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[32] F. Wang, R. Jiang, L. Zheng, C. Meng, and B. Biswal. 3d u-net based brain tumor

segmentation and survival days prediction. In *International MICCAI Brainlesion Workshop*, pages 131–141. Springer, 2019. doi: aarXiv:1909.12901v2.

[33] W. Wang, C. Chen, M. Ding, H. Yu, S. Zha, and J. Li. Transbts: Multimodal brain tumor segmentation using transformer. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part I 24*, pages 109–119. Springer, 2021. doi: arXiv:2103.04430v2.

# A | Appendix A

## A.1. Transfer Learning

In subsection 2.1.3, we discussed the issue of overfitting, specifically the challenges of generalization when models are trained on limited datasets.

To illustrate this concept in a real-world context, let's consider an example. Imagine asking an average person with no musical background to play Beethoven's "Für Elise" on a piano by ear, after practicing with a limited number of basic melodies for just one week. The likelihood of them delivering a flawless performance is quite low. They lack familiarity with hand placement, key locations, and struggle to maintain tempo – they have no musical experience to draw from.

Now, imagine posing the same challenge to a professional guitarist. While they may not have piano-playing expertise, they possess a deep understanding of music theory. They can read musical notations, manage tempo, and are skilled in various aspects of music. In this context, their experience significantly aids them in completing the task, even if it's somewhat different from their primary field of expertise. This example underscores the value of prior knowledge and its impact on tackling new, related challenges.

The same principle holds true for neural networks and machine learning models. When we aim to achieve strong performance on a task, but we face limitations in the form of a small dataset, computational constraints, or time constraints, we can employ a pre-trained model that has been trained on a larger dataset for a related task. This pre-trained model serves as a valuable starting point, allowing us to leverage its prior knowledge and adapt it to our specific goal using our limited dataset. This approach not only saves computational resources and time but also often leads to improved performance by transferring the learned knowledge from the broader task to our specific problem.

Furthermore, a model pre-trained on a large dataset is less prone to overfitting, thanks to the larger number of samples from which it learned to extract features to solve the task.

In particular, this kind of knowledge transfer can be achieved in two ways:

- **Transfer learning**: in this technique, a larger previously trained model is used to extract features from the smaller dataset, which are then used to train a second, simpler model to perform a specific task. In computer vision, this second model could be the decoder (for semantic segmentation) or the classifier (for image classification).
- **Fine tuning**: in contrast to transfer learning, both the feature extractor (the larger pre-trained model) and the second model undergo an additional training phase with the smaller dataset. In this way, the feature extraction process is optimized for the target dataset. However, it's worth noting that this method is more computationally expensive.

# B | Appendix B

In this appendix we will explore some useful mathematical tools that help the understanding of the machine learning and deep learning framework.

## B.1. Loss optimization techniques

Machine learning and deep learning algorithms construct models that are able to achieve specific goals by solving an optimization problem: minimize the error on the training set, also called Loss.

In order to do this, there are various methods.

### B.1.1. Stochastic gradient descent

In **Stochastic gradient descent** (or **SGD**), the model's parameters are updated after processing each individual data point from the training dataset. It is computationally efficient but can have high variance in parameter updates due to the randomness of the selected data point. This randomness can sometimes lead to slower convergence.

Formally, consider $\mathbf{w}$ the parameter vector of our model. For each sample $x_n$ with $n \in [1, N]$ (where $N$ is the dimension of the dataset) we apply a correction on each weight that is proportional to its contribution in the error of the prediction for that sample, that is computed with the gradient of the loss function with respect to the weight vector. Hence, at each step $k$, we compute the updated parameter vector with the following formula:

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \alpha \cdot \nabla L(\mathbf{x_n}, \mathbf{w}) \tag{B.1}$$

The term $\alpha$ represents a rational value that defines the ratio between the error and the applied correction during the training process. This parameter is commonly referred to as **learning rate**.

### B.1.2. Batch gradient descent

In **Batch Gradient Descent**, or **BGD**, the update of the model's parameters is performed at the end of each iteration on the entire training dataset. The correction to the parameters follows the direction of the negative gradient averaged over all data points in the dataset. It provides more stable and less noisy updates but can be computationally expensive, especially for large datasets.

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \alpha \cdot \frac{1}{N} \sum_{i=n}^{N} \nabla L(\mathbf{x_n}, \mathbf{w}) \tag{B.2}$$

### B.1.3. Mini-batch gradient descent

**Mini-batch gradient descent** strikes a balance between the extreme approaches of Stochastic Gradient Descent (SGD) and Batch Gradient Descent (BGD). In this method, the entire dataset is divided into smaller subsets called mini-batches. The weight vector update is executed once for each mini-batch by computing the gradient of the loss on that specific batch of data.

The mini-batch approach offers several advantages. First, it leverages the computational efficiency of SGD since updates are more frequent, yet it is less noisy than pure SGD due to the use of mini-batches. This reduced noise can result in more stable convergence during training. Additionally, mini-batch gradient descent is often more computationally efficient than BGD, making it suitable for training deep learning models on large datasets. Researchers can choose the mini-batch size to balance the trade-off between computational efficiency and noise reduction, with larger mini-batches reducing noise and smaller ones offering faster updates.

In this case, the parameter update is computed with the formula

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \alpha \cdot \frac{1}{M} \sum_{n=1}^{M<N} \nabla L(\mathbf{x_n}, \mathbf{w}) \tag{B.3}$$

where $N$ is the size of the dataset and $M$ is the batch size.

## B.2. Argmax and Softmax functions

### B.2.1. Argmax function

The **argmax** function returns the argument that maximizes the output of a function, typically an array or a sequence of data. Applied to arrays, it returns the index or value that corresponds to the maximum value in the sequence. For example, if we have an array of logits [0.2, 0.6, 0.1, 0.1] that represent the probability of a pixel of belonging respectively to the classes 1, 2, 3 and 4, the argmax would return "2", because the maximum value (0.6) is located at index 2 in the array.

### B.2.2. Softmax

The **softmax** function is a mathematical function often used in machine learning and deep learning to transform an array of numbers into a probability distribution. It is often applied to the output of a neural network's final layer for tasks like classification. If we consider an array $\mathbf{z} = [z_1, z_2, ..., z_K]$ of length $K$, the softmax function, computed as

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{B.4}$$

returns an array $\hat{\mathbf{z}} = [\hat{z}_1, \hat{z}_2, ..., \hat{z}_K]$ such that $\sum_{i=1}^{K} \hat{z}_i = 1$.

E.g., if our input array is

$$[30, 45, 48, 50, 61, 60]$$

the output of the softmax function is

$$[0,\ 0,\ 0.000002,\ 0.000012,\ 0.731048,\ 0.268938]$$

It's worth noting that while the values 61 and 60 are relatively close, the softmax function assigns a significantly higher probability to the first value. This property illustrates why the function is named "softmax", as it offers a more gradual or "soft" transition in assigning probabilities, in contrast with the sharp selection made by the argmax function.

# C | Appendix C

## C.1. Training logs

In this appendix, we provide an overview of the logs for the trained models and delve into a discussion regarding the design choices we've made. To begin, we present the logs specifically for the Segmenter models, visualized using the Matplotlib Python library:

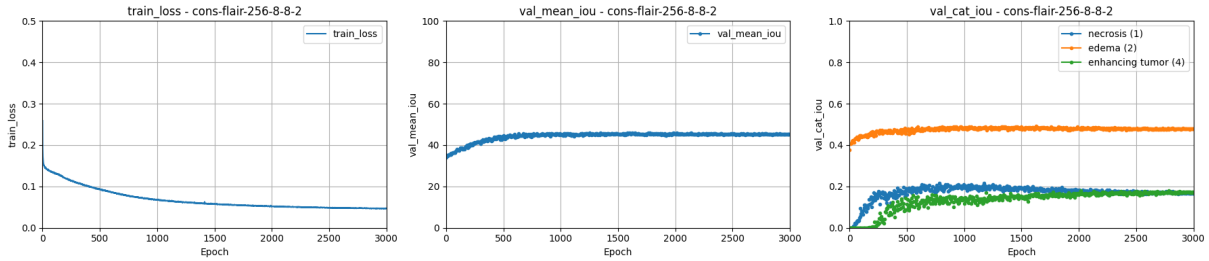



Figure C.1: Training logs for *cons-flair-256-8-8-2*

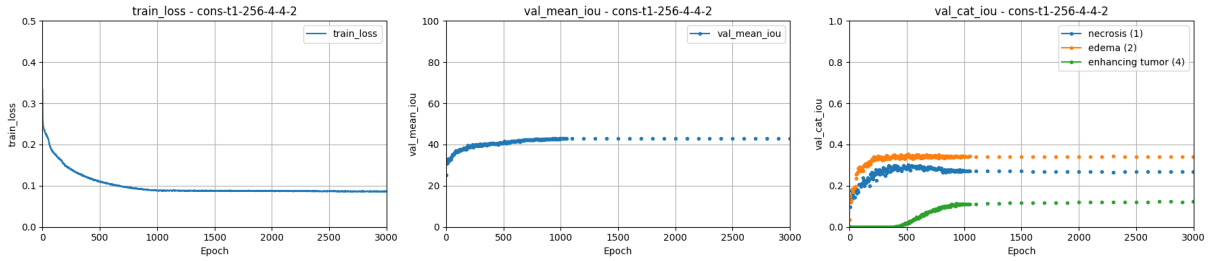



Figure C.2: Training logs for *cons-t1-256-4-4-2*

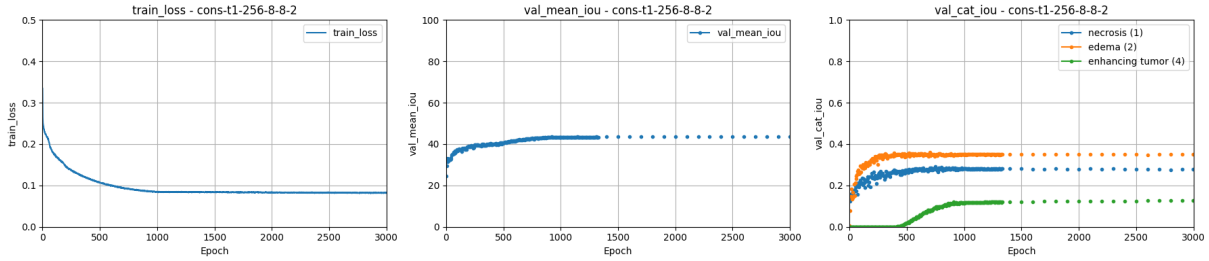



Figure C.3: Training logs for *cons-t1-256-8-8-2*

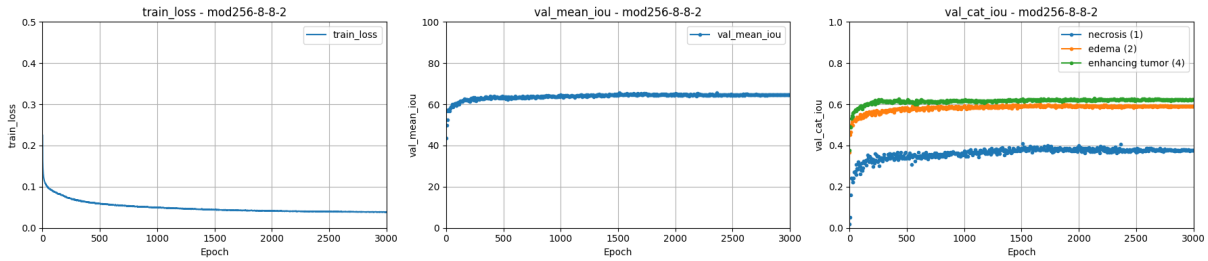



Figure C.4: Training logs for *mod256-8-8-2*

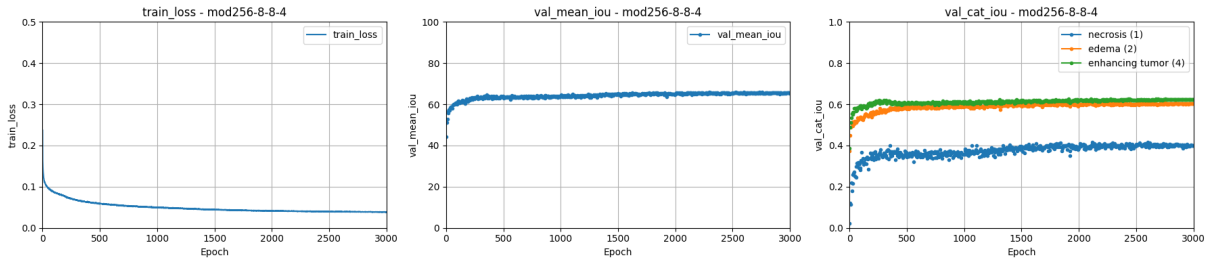



Figure C.5: Training logs for *mod256-8-8-4*

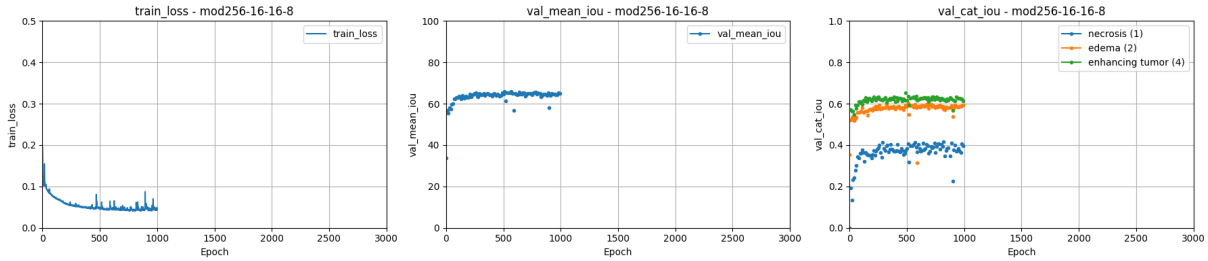



Figure C.6: Training logs for *mod256-16-16-8*

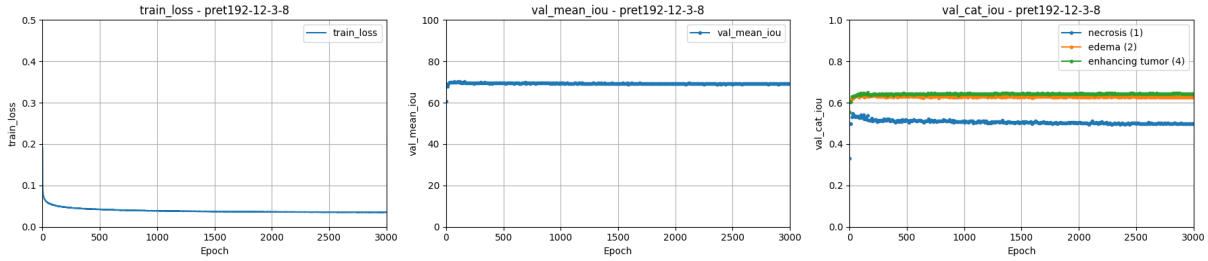



Figure C.7: Training logs for *pret192-12-8-3*

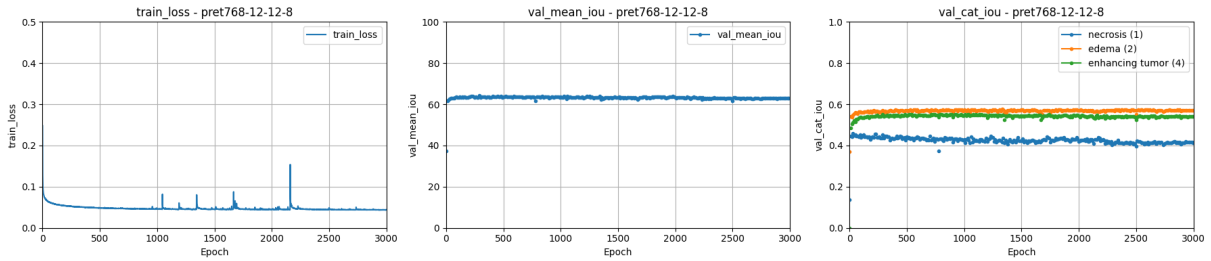



Figure C.8: Training logs for *pret786-12-12-8*

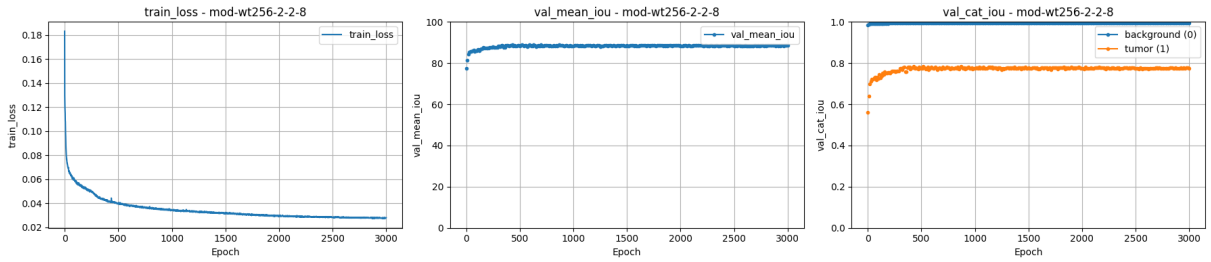



Figure C.9: Training logs for *mod-wt256-2-2-8*

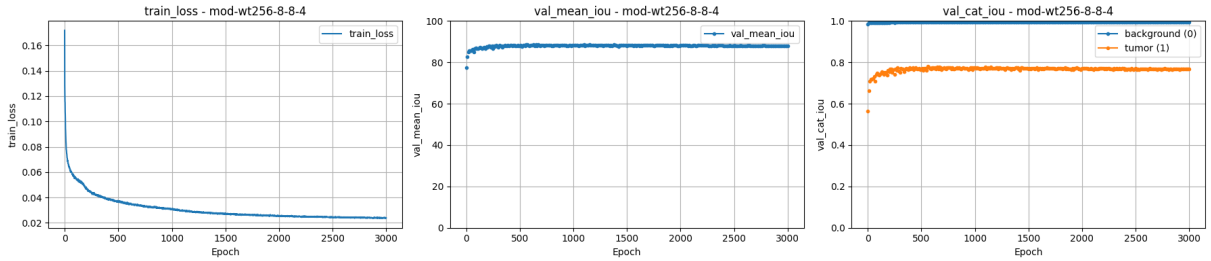



Figure C.10: Training logs for *mod-wt256-8-8-4*

Each figure in this set depicts the training loss, mean IoU, and IoU scores for individual classes across the training epochs for each model. Notably, there is no discernible evidence of overfitting; the IoU score on the validation set consistently shows improvement throughout the training process. The only exception lies with the two pre-trained models, where a gradual decline in performance on the necrosis class becomes apparent after approximately 100 epochs. Furthermore, it is observed that, except for models trained on a binary dataset, the loss tends to reach a plateau before reaching 1500 epochs. This suggests that, for the majority of these models, extensive training beyond this point may not be necessary. Despite our training duration choice, the impact on the validation set's performance was marginal. Consequently, we opted to utilize the last checkpoint for each Segmenter model to evaluate scores on the test set.

Now, we will present the logs for the SwinUNETR models, visualized using the Tensorboard tool. In this context, each plot illustrates the metric values for the three SwinUNETR models. Specifically, the green line corresponds to the *SWIN96-60* model, the blue line represents the *SWIN96-48* model, and the magenta line denotes the *SWIN64-48* model.

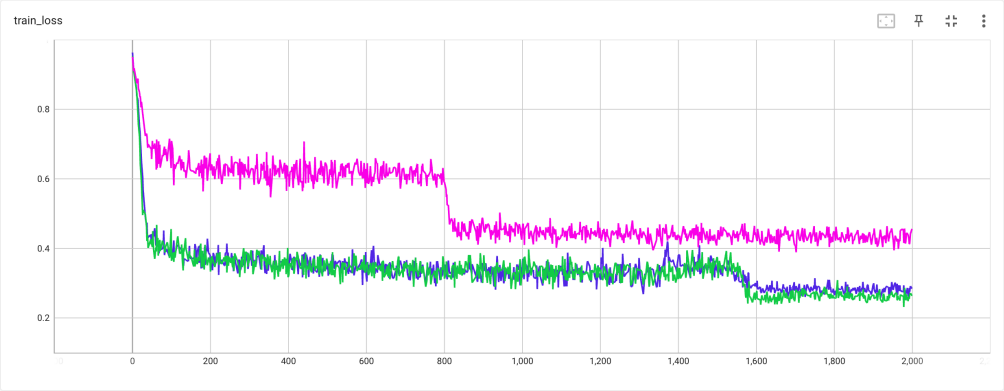



Figure C.11: Training loss for *SWIN96-60*, *SWIN96-48* and *SWIN64-48*

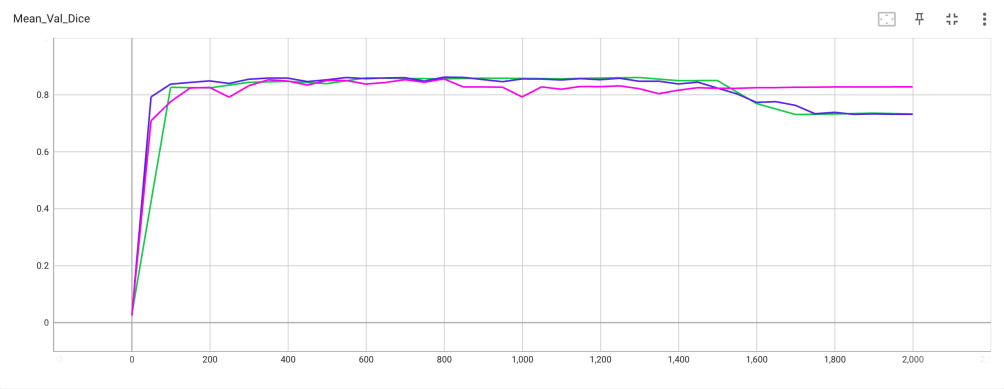



Figure C.12: Mean validation DICE for *SWIN96-60*, *SWIN96-48* and *SWIN64-48*

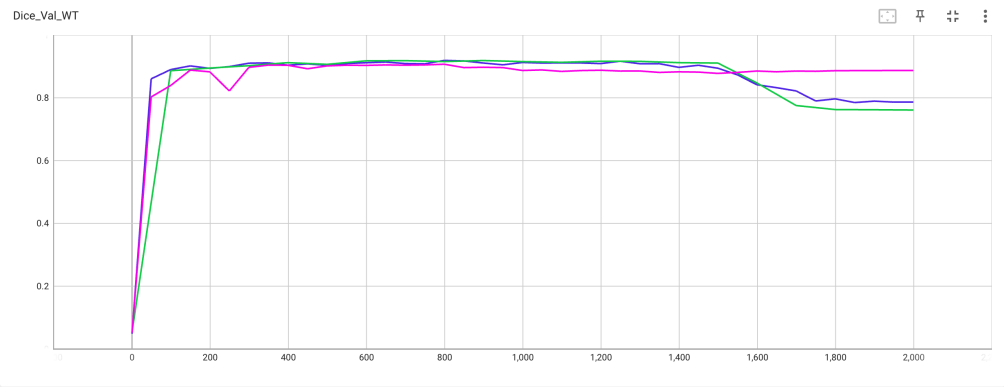



Figure C.13: Whole Tumor DICE for *SWIN96-60*, *SWIN96-48* and *SWIN64-48*

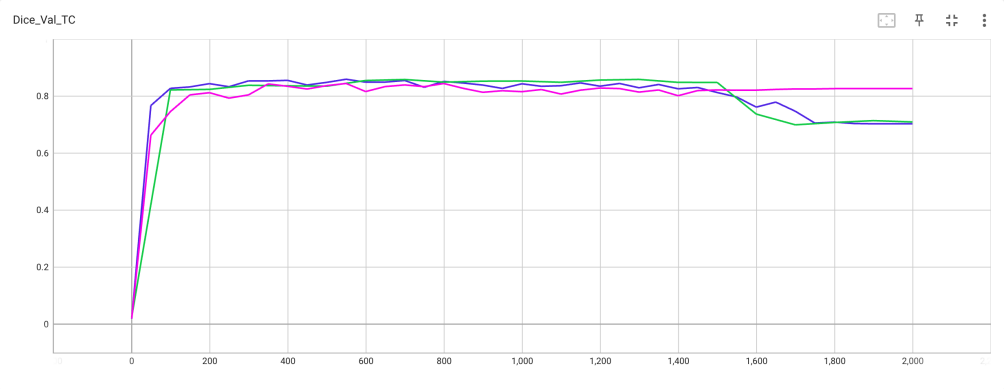



Figure C.14: Tumor Core DICE for *SWIN96-60*, *SWIN96-48* and *SWIN64-48*

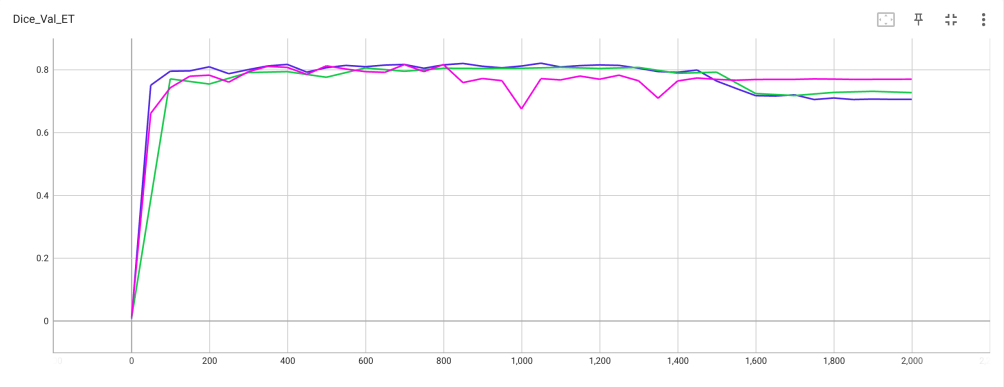



Figure C.15: Enhancing Tumor DICE for *SWIN96-60*, *SWIN96-48* and *SWIN64-48*

In Figure C.12, the trajectory of mean validation DICE scores for the *SWIN96-48* and *SWIN96-60* models reveals a decline post the 1400th epoch, while the *SWIN64-48* model appears to stabilize on a plateau. It's also noteworthy that for the first two models, approximately when the training phase reached a point of declining validation score, there was a simultaneous decrease in the loss. This phenomenon suggests a potential issue of overadaptation, where the models began memorizing the training set, enhancing performance on it but compromising their generalization ability. This is a clear example of overfitting.

Nonetheless, it's crucial to highlight that, across all models, the validation scores at the conclusion of training exhibited a degradation, with some instances displaying a notable downturn. As a result, we decided to utilize the most optimal checkpoints for the test set evaluation. Specifically, for the *SWIN96-60* model, we chose the checkpoint from the $1300^{th}$ epoch, while for the other two models, we opted for the checkpoint from the $800^{th}$ epoch.

# List of Figures

# List of Tables

# List of Acronyms

| Acronym | Description |
|---|---|
| **MICCAI** | Medical Image Computing and Computer Assisted Intervention |
| **CNN** | Convolutional Neural Network |
| **MRI** | Magnetic Resonance Imaging |
| **ViT** | Vision Transformer |
| **RF** | Radio Frequency |
| **FLAIR** | Fluid Attenuated Inversion Recovery |
| **T1ce** | T1 contrast enhanced |
| **BraTS** | Brain Tumor Segmentation |
| **NCR** | Necrosis |
| **ED** | Edema |
| **WT** | Whole Tumor |
| **TC** | Tumor Core |
| **ET** | Enhancing Tumor |
| **IoU** | Intersection over Union |
| **MLP** | Multi layer perceptron |
| **LN** | Layer Normalization |
| **MSA** | Multi-head Self Attention |
| **W – MSA** | Window Multi-head Self Attention |
| **SW – MSA** | Sliding Window Multi-head Self Attention |
| **TP** | True positive |
| **FP** | False positive |
| **TN** | True negative |
| **FN** | False negative |
| **NIFTI** | Neuroimaging Informatics Technology Initiative |
| **SGD** | Stochastic Gradient Descent |
| **BGD** | Batch Gradient Descent |

# Acknowledgements

Personalmente, vedo la tesi di Laurea Magistrale come la destinazione di un percorso ben più lungo della mia carriera accademica, e in quanto tale mi sento in dovere di ringraziare tutte le persone che ne hanno fatto parte. Nella primavera del 2014, i miei professori di informatica delle scuole superiori portarono la mia classe a visitare il Politecnico, e nonostante mancasse ancora più di un anno alla maturità ne rimasi stregato, e, impaziente di cominciare, iniziai il mio conto alla rovescia verso l'immatricolazione.

È al Massimo di nove anni fa, quindi, che devo il primo ringraziamento. La sua curiosità e la sua determinazione hanno compiuto il primo passo verso questo traguardo, e, nonostante le infinite difficoltà, rifarei la sua scelta mille volte.

Ringrazio i miei relatori Daniele Loiacono e Leonardo Crespi, per avermi guidato in questo progetto e avermi permesso di lavorare ad un argomento tanto interessante quanto delicato.

Ringrazio i miei genitori, Olimpia e Rossano, per non aver mai dubitato di me e delle mie capacità e per avermi sempre supportato in ogni mia decisione, e il mio gemello Luca, che nonostante la distanza rimane per me un punto di riferimento.

Ringrazio i miei stupendi amici Andrea, Francesco, Manuela, Marco I., Marco S., Nicolò, i tre Riccardo e Viviana, per la loro vicinanza in questi anni, e per avermi ispirato, ognuno a proprio modo, con il proprio operato, a dare sempre il meglio e ad essere ambizioso.

Ringrazio Federica, compagna di studi dal 2016, per avermi accompagnato in questo percorso fino all'ultimo miglio.

Ringrazio Kineton S.r.l., la società per cui lavoro, in particolare i miei manager Pasquale e Gianfranco, per avermi concesso la flessibilità necessaria a portare fino in fondo questo progetto, e tutti i miei colleghi.

Ringrazio tutti i professori che mi hanno avuto come studente, in particolare la professoressa Catino e il professor Pitorri, che durante gli anni delle superiori mi hanno formato, mi hanno fatto appassionare alla matematica e all'informatica e mi hanno reso consapevole

delle mie potenzialità. Ringrazio la professoressa Croci, che voglio onorare mantenendo la promessa che mi ha chiesto di farle.

Ringrazio Michele, per avermi sostenuto nei periodi più bui di questi ultimi anni e per esserci sempre stato per me nel momento del bisogno.

Ringrazio Beatrice, per essere stata al mio fianco durante il periodo più intenso di questo progetto con tanta pazienza e dolcezza.

Gli ultimi anni sono stati delle montagne russe di momenti di lavoro intenso e di noia, di dolori e di gioie, di fallimenti e di soddisfazioni. Per questo motivo, l'ultimo ringraziamento va al Massimo adulto, per essere stato tenace e non essersi mai arreso.

Un pensiero va a mio nonno Armando, che per uno scherzo del destino non ha fatto in tempo a vedermi concludere questo percorso. Spero tu sia comunque orgoglioso di me.