Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Master in Digital Innovation: Health and Medical Data Analytics

Master Thesis

Comparative Analysis of Deep Learning Models and Strategies for Multiclass Brain MRI Segmentation

Author: Hugo Muñoz Suárez

Madrid, September, 2025

This Master Thesis has been deposited in ETSI Informáticos de la Universidad Politécnica de Madrid.

Master Thesis

Master in Digital Innovation: Health and Medical Data Analytics

Title: Comparative Analysis of Deep Learning Models and Strategies for Multiclass Brain MRI Segmentation

September 2025

Author: Hugo Muñoz Suárez

Tutor
Supervisor:

José Crespo del Arco
- Ingeniero de Telecomunicación (ETS Ingenieros de Telecomunicación, UPM)
- Doctorado Ph.D. (Georgia Institute of Technology, EE.UU.)
- Doctorado (ETS Ingenieros Informáticos, UPM)

ETS de Ingenieros Informáticos
Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software
Universidad Politécnica de Madrid

Co-Tutor
Co-supervisor:

Manuel Villa Romero
- Grado en Ingeniería de Sistemas de Telecomunicación (ETSIST, UPM)
- Master of Internet of Things (ETSIST, UPM)
- Predoctoral (ETSIST, UPM)

ETS de Ingeniería y Sistemas de Telecomunicación
Departamento de Electrónica Física, Ingeniería y Física Aplicada
Universidad Politécnica de Madrid

Abstract

Magnetic Resonance Imaging (MRI) is one of the most advanced techniques for studying brain structure and pathology, and it has become a fundamental tool in the diagnosis and monitoring of neurological disorders. Its clinical utility can be further enhanced by accurate and efficient image segmentation. Accurate segmentation of anatomical and pathological regions is essential for quantitative analysis, treatment planning, and disease monitoring. Although manual segmentation by experts remains the gold standard, it is laborious, error-prone, and impractical in large-scale or time-critical clinical contexts. These limitations have motivated the development of automated segmentation methods based on deep learning. However, despite the outstanding progress achieved in recent years, current solutions still face significant challenges, including limited availability of annotated data, severe class imbalance, and difficulties in generalization across datasets and acquisition protocols.

This Master's Thesis addresses these challenges by performing a comparative analysis of three reference deep learning architectures for semantic segmentation: U-Net, DeepLabV3, and Fully Convolutional Network (FCN). This work was conducted in the context of multiclass brain MRI segmentation, using a synthetic dataset of phantoms. The study is structured around three complementary experimental approaches, designed to assess both architectural performance and improvement strategies. The first approach is based on direct training from scratch and was conducted on a phantom dataset. This dataset was specifically created to simulate realistic brain structures and included multimodal acquisitions (T1, T2, DP), manually and semi-automatically segmented into six classes: background, tumor, white matter, grey matter, blood vessels, and external markers. Masks were generated and standardized using a pipeline combining Label Studio with the Segment Anything Model (SAM), ensuring consistent and reproducible annotations. The second approach explored transfer learning through pretraining on the publicly available BraTS2020 dataset, followed by finetuning on the phantom dataset. This aimed to investigate whether pretrained representations on real patient data could improve performance on a smaller and domain-specific dataset. The third approach introduced advanced data augmentation strategies to address class imbalance and limited sample size. All three architectures were trained under homogeneous configurations to ensure fair comparison, and subsequently with customized hyperparameters optimized per model. Evaluation relied on widely adopted segmentation metrics (Dice coefficient, Intersection over Union, pixel accuracy), with complementary analysis of training dynamics, per-class performance, computational cost, and qualitative visual inspection of representative cases.

Results from the first approach revealed clear architectural differences. Under homogeneous hyperparameters, DeepLabV3 and FCN outperformed U-Net, establishing the initial ranking among the three models. After introducing architecture-specific adjustments, such as tailored learning rates, performance improved substantially in all cases. DeepLabV3 reached the best overall results, and U-Net obtained the lowest performance. These findings highlight the importance of both architectural design and hyperparameter tuning. In terms of computational efficiency, FCN was the fastest and least memory-demanding, while U-Net was the most resource-intensive, requiring significantly higher GPU memory and FLOPs.

In contrast, transfer learning with BraTS2020 did not translate into performance gains under the conditions of this study. While pretrained weights led to smoother training dynamics and mitigated overfitting, the final performance of the three architectures were lower. The third approach, based on augmentation strategies, achieved results comparable to direct training while mitigating overfitting and improving segmentation of certain classes. White and grey matter benefited the most, with Dice scores consistently improving across models. The tumor class also showed slight but consistent gains, demonstrating that targeted augmentation enhanced minority-class learning. However, improvements were less pronounced for highly underrepresented classes, where data scarcity remained a limiting factor. The qualitative visual analysis reinforced these observations, showing that DeepLabV3 and FCN produced cleaner masks with sharper boundaries, whereas U-Net tended to generate blurrier contours and spurious predictions in minority classes. After augmentation, all models produced visually more consistent masks, particularly in complex regions, confirming the benefits of increased data variability.

Taken together, the comparative analysis demonstrates that DeepLabV3 is the most balanced and robust model for multiclass brain MRI segmentation, combining high accuracy with moderate computational cost. Beyond the numerical results, this work highlights the crucial role of dataset quality, annotation consistency, and class balance in training reliable medical segmentation models. The main limitations of the study are the small dataset size and the strong class imbalance, which constrained the ability to evaluate minority structures.

In conclusion, this thesis primarily provides a systematic comparison of three reference deep learning architectures for multiclass brain MRI segmentation, establishing their relative strengths and weaknesses under different conditions. Building on this benchmark, alternative strategies such as transfer learning and data augmentation were explored to address the limitations encountered. The study thus contributes practical insights into the trade-offs between model performance, computational cost, and dataset constraints, offering guidance for future developments in AI-assisted neuroimaging. Among the evaluated models, DeepLabV3 emerges as the most balanced candidate for potential integration into clinical workflows, supporting tasks such as tumor delineation, treatment planning, and disease monitoring.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1 Introduction

Magnetic resonance imaging (MRI) of the brain constitutes one of the most advanced and versatile imaging techniques in the biomedical field. It is fundamental in the diagnosis and monitoring of neurological disorders, as it has a great capacity to obtain high resolution volumetric images and excellent contrast between soft tissues. This technique is essential in both clinical practice and research, as it enables detailed visualisation of intracranial structures such as white matter, grey matter, the ventricular system and pathological regions, including tumours, haemorrhages and ischaemic lesions. Its non-invasive nature and the absence of ionising radiation make it the tool of benchmark for neurological diagnosis and assessment, surgical planning and evaluation of therapeutic response. However, the large amount of information provided by MRI also involves a considerable challenge in terms of analysis and interpretation, given the high interpatient variability and the presence of artefacts and noise resulting from the acquisition process [1]. Brain MRI is particularly relevant in the study of complex pathologies such as brain tumours. These represent approximately 2% of all cancers diagnosed in adults and 15% of those diagnosed in children. In 2020, over 300,000 new cases of primary brain and central nervous system cancers were diagnosed worldwide, with more than 250,000 associated deaths and a five-year survival rate as low as 35% for malignant subtypes [2]. An accurate characterisation of tumour margins and the identification of areas of infiltration are critical for prediction and clinical decision-making.

Medical image segmentation is the process through which are identified and delineated anatomical or pathological regions of interest. It is based on assigning a label to each pixel according to its belonging to a specific class. In the context of brain MRI, accurate segmentation of structures such as tumours, oedema, necrosis, or healthy tissues is essential for objective quantification, treatment planning, and longitudinal disease monitoring. In this context, manual segmentation of brain MRI images, carried out by experts, is still considered the gold standard for obtaining anatomical or pathological masks [3]. However, this is a laborious and time-consuming process that is susceptible to human error and variability among experts. In large patient groups studies or clinical contexts where immediacy is crucial, reliance on manual segmentation becomes unfeasible. This generates the need for the development of robust and accurate automated methods [4].

Historically, automated segmentation has been addressed using classical techniques such as thresholding or clustering. While these approaches allowed for initial advances, they have notable limitations in capturing complex spatial relationships and adapting to the morphological heterogeneity of images. Furthermore, they require considerable manual intervention and case-specific adjustments, which reduces their applicability in real clinical environments [5]. In the last decade, the emergence of artificial intelligence, and in particular deep neural networks, has led to a paradigm shift in the field of medical segmentation. Deep learning models have demonstrated an outstanding ability to learn hierarchical representations from large volumes of data, efficiently integrating global contextual information and local details. These architectures have achieved benchmark results, establishing themselves as the current standard in medical semantic segmentation [6]. Among their main advantages are the automation of the segmentation process, the mitigation of inter-observer variability, and the possibility of accelerating the analysis of large patient groups.

Moreover, the possibility of training models on multimodal images and the flexibility to adapt to different tasks have further expanded their impact on clinical practice and biomedical research.

Despite the progress made, current automated segmentation models deal with some significant challenges [5]. The first one is the scarcity and heterogeneity of annotated data. The collection of large volumes of expert labelled data is costly and complex, and the available public databases are scarce, small, and characterized by heterogeneous annotations. This restricts the generalisation capacity of the models and limits their training and validation. Another constraint is the high inter-institutional variability in acquisition protocols, scanner types and population characteristics. This creates a major problem in terms of the robustness and applicability of the models in real clinical scenarios. When a model is transferred to data from other centres, there is a notable decrease in performance due to this variability. Furthermore, another limitation of current solutions is class imbalance. In the field of brain lesion segmentation, regions of interest (tumours or small lesions) are often underrepresented compared to the background or healthy tissue, occupying a very small percentage of the total image volume. This extreme imbalance between classes complicates model learning and tends to bias predictions towards the majority class, reducing sensitivity for detecting small or atypical lesions. Finally, a frequent consequence in models trained with small datasets is overfitting, where models learn specific patterns from the training set, losing their ability to generalise to new data. Although there are strategies to mitigate this risk, such as data augmentation or regularisation, they often lead to additional computational costs. More advanced architectures are generated, and they require high computational resources, which can limit their adoption in environments with limited infrastructure, such as hospitals [7].

Scope and Contributions of the Master's Thesis

In this context, the present Master's Thesis is focused on the implementation, adaptation, and comparative evaluation of various deep neural network architectures for automated brain MRI segmentation. The main objective is to systematically analyse the performance, robustness, and efficiency of different models, trained and evaluated on the same dataset, under controlled and homogeneous conditions. For this purpose, three reference architectures were selected: U-Net, DeepLabV3, and Fully Convolutional Networks (FCNs). These models were chosen based on a comprehensive review of the literature, as they are among the most frequently employed and consistently achieve state-of-the-art results in medical image segmentation. This work also explores advanced improvement techniques, such as data augmentation or model pretraining, in order to increase the diversity of the training set and mitigate the scarcity of real data.

The development of robust automated segmentation tools is essential for advancing towards the effective integration of artificial intelligence into clinical workflows. Critical comparison of architectures and improvement strategies can provide valuable information on optimal training conditions, current limitations, and future perspective in the field of deep learning-based medical segmentation.

To address the outlined challenges and achieve the main goal of this work, the following specific objectives were defined:

- Implement and train various deep neural network architectures for brain MRI image segmentation.
- Quantitatively evaluate the performance of each model using standard metrics.
- Systematically compare the results obtained, highlighting strengths, weaknesses, and optimal training conditions for each architecture.
- Explore improvement strategies to increase the model's generalisation capacity.

The structure of this thesis is organized as follows. Chapter 2 reviews the state of the art in medical image segmentation. It begins with an introduction to basic neuroanatomy and brain MRI, followed by conventional segmentation approaches, the transition to deep learning, and the challenges currently faced in the field. It then describes the functioning of convolutional neural networks and presents the three selected architectures in detail, concluding with a review of evaluation metrics, current limitations, and complementary techniques to mitigate them. Chapter 3 outlines the methodology and development, including the dataset description, pre-processing steps, model implementation, and experimental design. Chapter 4 presents the results obtained from the different approaches, complemented by both quantitative metrics and qualitative analyses. Chapter 5 summarizes the main conclusions of the work, highlighting its implications, limitations, and potential directions for future research. Finally, Chapters 6 and 7 contain the bibliography and annexes, respectively.

2 State of the art

2.1 Basic anatomy of the nervous system

2.1.1 General brain structure

The human nervous system is a complex network of specialized cells whose main function is to coordinate and regulate the body's activities. Among other functions, it is responsible for controlling communication, responses to external stimuli, and, in general, all internal processes that occur in the body. From an anatomical point of view, it is divided into two major components: the central nervous system (CNS) and the peripheral nervous system (PNS). The first one is formed by the brain and the spinal cord, and the second one consists of the nerves and ganglia that connect the CNS to the rest of the body. The CNS is protected by bones. Specifically, the brain is covered by the skull, while the vertebral column protects the spinal cord. Within the central nervous system, the brain, and in particular the cerebrum, stands as the structure of primary clinical and research interest. The brain is anatomically divided into three regions: the cerebrum or forebrain, the brainstem, and the cerebellum [8].

The cerebrum is the main organ of the CNS and is characterized by its complex structural and functional organization. Its structure can be divided into two hemispheres (left and right), which are practically symmetrical and connected to each other by the corpus callosum. Both have four lobes (parietal, frontal, temporal, and occipital), each of which performs specific functions such as sensory processing, motor control, memory, language, and decision-making. At the same time, the brainstem, which connects the brain to the spinal cord, regulates vital functions such as breathing and heart rate. Finally, the cerebellum is involved in motor coordination and balance [9], [10].

From a medical imaging perspective, the brain is analysed based on its different types of tissue: grey matter, white matter, and cerebrospinal fluid (see Figure 1) [9]. Grey matter is found in the most superficial area, forming what we know as the cerebral cortex. This area contains the cell bodies and dendrites of the brain's neurons. White matter is composed of neuronal axons that connect different regions of the brain to each other and to other areas of the nervous system. Finally, cerebrospinal fluid circulates through the ventricular system and is responsible for maintaining cerebral homeostasis. Detailed knowledge of this organization is essential for clinical interpretation and research using neuroimaging studies [9], [10].

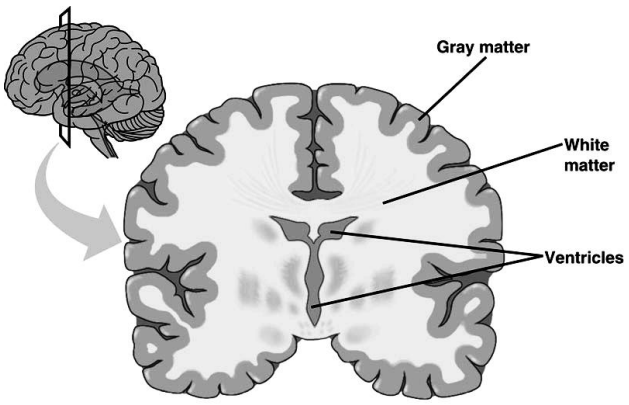

Figure 1. Representation of basic brain structures in a coronal section: grey matter, white matter, and ventricles [11].

2.1.2 Brain Magnetic Resonance Imaging

One of the most widely used techniques for studying the brain is magnetic resonance imaging (MRI). This modality provides high spatial resolution images and excellent contrast between different types of tissue. This enables accurate visualization of the main regions of structural and pathological interest in the brain. Among the most relevant anatomical regions are:

- The cerebral cortex or grey matter, which allows for the identification of cortical patterns in neurodegenerative diseases.
- The white matter, essential for the investigation of demyelinating pathologies such as multiple sclerosis.
- The ventricular system, relevant for the evaluation of dilations or deformations indicative of hydrocephalus or brain atrophy.

In the pathological field, MRI allows the identification and segmentation of brain tumors, ischemic lesions, haemorrhages, and other alterations. It is essential for the diagnosis and evolutionary monitoring of the patient [12] [4].

2.1.3 Clinical relevance

In this context, brain structure segmentation has become a key tool for extracting quantitative anatomical information from images. This segmentation can focus on both healthy regions (brain lobes, ventricles, cortex, etc.) and pathological areas (lesions, tumors, haemorrhagic foci), facilitating longitudinal analysis and clinical decision-making. For example, it allows for the objective volumetric quantification of structures or lesions. This is essential for monitoring tumors, degenerative diseases, and therapeutic responses. Likewise, the identification of critical areas aids in the planning of surgical and radiotherapeutic interventions. As a result, it minimizes damage to essential brain functions and improves patient prognosis.

Segmentation presents major challenges apart from the variety of structures present in the brain. These include morphological variability between patients, possible image artefacts, the presence of noise, and the complexity of distinguishing diffuse boundaries between tissues. In addition, the number of classes to be segmented, their relative proportion, and the similarity of intensity between regions can further complicate the segmentation process. For these reasons, robust and accurate tools are required to automatically identify these structures in MRI images. The automation of these processes through artificial intelligence fosters their integration into the clinical workflow, reduces the workload of specialists, and improves the reproducibility of studies [4].

2.2 Medical image segmentation

2.2.1 Definition and types of segmentation

Medical image segmentation is the process by which specific anatomical, functional, or pathological regions are identified, delimited, and classified. This image can be obtained by different diagnostic imaging techniques, such as magnetic resonance imaging (MRI), computed tomography (CT), or ultrasound. The purpose of this task is to facilitate the extraction of quantitative and structural information that can be used in clinical or research procedures. It allows us to know the precise location of organs, tissues, or lesions [13], [14].

Depending on the objective and type of data, segmentation can be classified into different types. In the first one, the goal is to distinguish a single structure of interest from the rest of the background, for example distinguishing a tumor from healthy tissue. In contrast, in multiclass segmentation, a different label is assigned to each of the multiple regions that appear in the image. This is what

happens when we want to differentiate between grey matter, white matter, and cerebrospinal fluid in a brain image. Another important distinction is between semantic segmentation and instance segmentation. In the former, all pixels belonging to a class are classified without distinguishing individual instances. In the latter, separate objects within the same class are identified, for example, several different tumors within the same image.

2.2.2 Traditional segmentation methods

Before the rise of deep learning, segmentation was approached using classical methods based on image processing techniques and statistical models. These use characteristics such as intensity, texture, or spatial geometry of the image [4] [13]. Among the most commonly used approaches are:

- **Thresholding**: assigns labels to pixels based on whether their intensity falls above or below a predefined threshold. This is the simplest and most computationally efficient method, but it is very sensitive to noise and contrast variation between images [15].
- **Clustering**: techniques such as k-means [16] or fuzzy c-means group pixels according to the similarity of characteristics such as intensity or texture, allowing for unsupervised segmentation. They introduce a certain tolerance to noise and anatomical variability, allowing them to adapt to different structures. However, their performance depends largely on good initialization (the definition of the number of clusters) and data homogeneity [17].
- **Active contours**: these models represent regions of interest using curves that evolve in the image to fit the detected edges. They are highly dependent on the initial estimate and sensitive to noise [18].
- **Region-based methods**: such as region growing or splitting and merging, which group contiguous pixels based on similarity criteria. Their main limitation is their dependence on the starting point and noise [19].

These approaches have been useful for decades. However, despite their simplicity and low computational cost, they have significant limitations in terms of their ability to generalize to images of different origins or quality, their sensitivity to anatomical variations, and their dependence on manual parameters. These restrictions motivated the search for more robust and automated solutions [5].

2.2.3 Transition to deep learning-based methods

The increase in computing power and the availability of annotated databases have enabled a paradigm shift in medical image segmentation. Traditional techniques have been progressively replaced by approaches based on machine learning and, in particular, deep learning. These models are capable of automatically learning discriminative features and hierarchical representations of data directly from images. This largely eliminates the need to manually design specific features or adjustments [20].

Over the last decade, convolutional neural networks (CNNs) have led to a substantial improvement in accuracy and robustness compared to classical methods, even in complex tasks of multiclass segmentation of brain magnetic resonance images. The adoption of these techniques has been accompanied by an increase in the availability of public datasets and GPU-accelerated training platforms. This has encouraged the development of increasingly accurate and generalizable models.

Despite these advances, the clinical application of these techniques still presents significant challenges, including the need for large volumes of

annotated data, the difficulty in explaining the model's decisions, and the variability in the quality of the generated segmentations. Nevertheless, the results obtained so far indicate that deep learning represents the most promising path toward accurate and reproducible automatic segmentation in medical images. Thus, deep learning has become the dominant approach for automated segmentation in clinical neuroimaging and biomedical research [7].

2.3 Foundations of machine learning and neural networks

2.3.1 Machine learning

Machine learning is a branch of artificial intelligence that focuses on designing algorithms capable of learning patterns from data based on statistics. Instead of being explicitly programmed for each task, these algorithms are trained with examples that allow them to adjust their internal parameters [21]. Through a defined cost function and training procedure, the models iteratively update these parameters to approximate an optimal solution. By minimising the error between predictions and expected values, the system progressively improves its performance and gains the ability to generalise to new, unseen data. In this way, machine learning models can perform predictions or decisions autonomously, adapting their behaviour based on prior experience [22].

Machine learning methods can be classified into two main categories. In supervised learning, the model is trained with labelled data, i.e., input examples together with their corresponding labels or expected values. The correct algorithm must be found to associate each piece of data with its corresponding label and to efficiently label new data that may be received. This is the most common approach in tasks such as classification, regression, or semantic segmentation. In unsupervised learning, the algorithm is trained only on input data, which does not contain labels. The goal is usually to learn characteristics or discover hidden patterns based on the properties of the data. This is the case of clustering or dimensionality reduction. Currently, other approaches such as semi-supervised learning or reinforcement learning are beginning to emerge [21] [23].

The applications of machine learning in the medical field are extensive. It is used in diagnostic support systems, disease classification, image anomaly detection, genomic analysis, clinical outcome prediction, and more [23]. When it comes to neuroimaging specifically, machine learning has been revolutionary. It has made it possible to process large volumes of information and find complex patterns that often escape traditional analysis. Supervised learning is dominant in tasks involving the automated segmentation of anatomical structures and lesions in medical images. In this approach, models learn the correspondence between images and their expert-annotated segmentations, optimizing the accuracy of the detection of structures such as grey matter, white matter, or tumor lesions. In the context of this work, machine learning is applied using supervised deep neural networks to segment anatomical structures in brain magnetic resonance images [22].

2.3.2 Biological neural networks

Artificial neural networks are a supervised machine learning technique inspired by the functioning of the biological nervous system. In the human brain, neurons are specialized cells that process information using electrical and chemical signals. A biological neuron consists of dendrites (which receive signals), a cell body (which integrates information), and an axon (which transmits the signal to other neurons) [24].

Communication between neurons occurs at the synapse. Electrical impulses release neurotransmitters that cross the synaptic space (the space between neurons) and activate receptors in the next neuron. The intensity of the signal depends on the number of neurotransmitters released, since as they increase or decrease, the potential in the dendrite of the receiving neuron will vary, producing a different response when the signal from all the dendrites of that receiving cell is added together. Therefore, the learning process of a biological neural network consists of, through the number of neurotransmitters released, modify the strength of the connections between different neurons in order to achieve the most appropriate response [24]. In summary, the functioning of each neuron is based on receiving signals from different neurons to which they are connected and depending on whether the sum of the signals received exceeds a certain activation threshold, emitting an electrical impulse that is transmitted to other neurons. The human brain contains approximately 86 x 10^9 neurons, each connected to a thousand other neurons, forming a massively parallel processing network [25]. This model has served as inspiration for building artificial neural networks. These networks attempt to emulate this learning ability through vastly simplified structures [26].

2.3.3 Basic artificial neural networks

A basic artificial neural network is composed of artificial neurons organized in layers. Each layer is composed of multiple neurons that process information in parallel. The connection between neurons is made through synaptic weights. These determine the relative importance of each input in calculating the neuron's output and are adjusted during training to modulate the influence between neurons [26], [27].

The basic mechanism is as follows: each neuron receives inputs, weights them using synaptic weights, calculates a sum of its inputs, and this is transformed by an activation function to generate an output. This output can be sent to other neurons in subsequent layers or directly to the final result of the model. The output of an artificial neuron can be expressed mathematically as:

$$y = A\left(\sum_{i=1}^{n} x_i w_i + b\right)$$

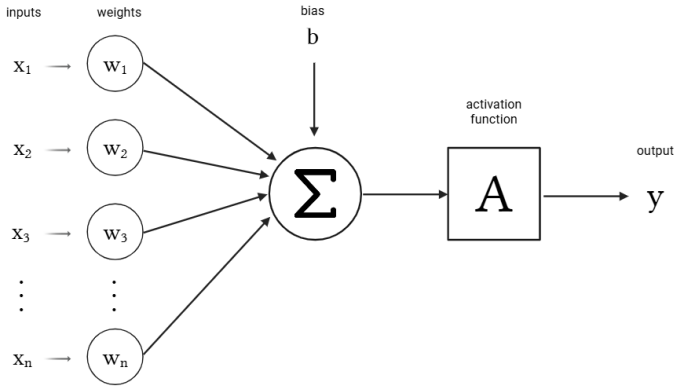

Figure 2. Basic diagram of an artificial neuron.

The operation described is illustrated in Figure 2. Where "x_i" are the inputs, "w_i" are the associated weights, 'b' is a bias, and "A" is the activation function. The equation presented illustrated the basic operation of the neuron shown, where the output will be the activation of the sum of each input multiplied by its respective synaptic weight. The weight of a specific input represents the strength of the connection between the neuron that provided that input and the neuron

we are currently at. The learning process in artificial neural networks consists of adjusting the different weights in all neurons to obtain the final output we want from certain inputs [26], [27]. The function applied to the sum, called the activation function, is responsible for calculating the output based on all the inputs and weights. It simulates the action potential threshold in a biological network, but in reality, in artificial networks, it is not always a threshold; there are other functions that are commonly applied to the neuron's output. Therefore, the type of activation function we choose is a very important decision for the functioning of our network [28], [29]. Among the most commonly used activation functions are:

- ReLU (Rectified Linear Unit)

$$f(x) = \max(0, x)$$

- Softmax

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

- Sigmoide

$$f(x) = \frac{1}{1 + e^{-x}}$$

- Tanh (hyperbolic tangent)

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Another important detail creating a neural network is the design of its architecture. A complete neural network consists of:

- **Input layer**: there is only one, this is where the initial data enters. It will have as many neurons as the amount of input data we want to enter.
- **Hidden layers**: these are located between the input and output layers. They are responsible for processing the data to achieve the desired results. There can be many hidden layers, with as many neurons as we want in each one. Neural networks with more than one hidden layer are called deep neural networks. Hidden layers have the function of extracting increasingly abstract and complex features from the primary information. In the case of medical images, these layers progress from the detection of contours and textures to the representation of complete anatomical structures.
- **Output layer**: there will only be one, and it is responsible for returning the final data. It will have as many neurons as outputs we want. The output layer adapts its format to the task; for segmentation, it usually consists of probability maps that indicate the belonging of each pixel to a given anatomical class.

The number of layers and neurons determines the complexity of the network. The greater the depth (more hidden layers), the greater the capacity to represent the problem to be solved. But there is also a greater risk of overfitting and computational cost. There is no basic rule governing the best architecture for solving a given problem. Design choices are generally guided by existing literature and refined through trial and error evaluation of the architecture´s performance [28].

2.4 Convolutional Neural Networks (CNNs)

2.4.1 Motivation and biological inspiration

The deep neural networks explained so far are only useful for analyzing numerical data. For image processing convolutional neural networks (CNNs) need to be used. These are an extension of artificial neural networks specialized

in processing data with a spatial structure, such as images. They are designed to capture spatial relationships and visual patterns with fewer parameters and better generalization [14], [30]. Their development is inspired by neuroscientific studies on the visual cortex of mammals conducted by Hubel and Wiesel in the 1960s [31]. They observed that certain neurons in the primary visual cortex respond selectively to visual stimuli with specific local characteristics, such as edges or textures. This principle led to the idea of building artificial networks in which certain neurons act as detectors of local patterns in the image, and whose progressive combination allows increasingly complex structures to be captured. CNNs replicate this organization through sequentially arranged layers, each responsible for capturing different spatial aspects of the input data.

Unlike traditional deep neural networks, CNNs take advantage of spatial correlations between nearby pixels. In order to make the most of the information present in all pixels, they take into account not only the value of a specific point but also the value of the nearest ones. This significantly reduces the number of parameters and improves computational efficiency. The hierarchical structure of CNNs allows the first layers to identify simple features (edges, lines), while the deeper layers recognize more complex compositions (shapes, anatomical structures, etc.). This capacity for abstraction has been key to their success in computer vision tasks, including image classification, object detection, and, notably, semantic segmentation [14], [30].

2.4.2 Main layers

A CNN is composed of a sequence of specialized hidden layers that process information hierarchically, allowing for the progressive extraction of features from the input data [14], [27], [30], [32]. The most important types of layers are:

- **Convolution layers**: are the basis of all CNNs. These layers apply filters called kernels to the input image. These filters are small arrays (for example, 3x3 or 5x5) whose values are adjusted during training. The filters slide over the input image to detect specific local patterns (vertical edges, horizontal edges, shapes, textures, etc.). Each filter scans the image applying a convolution operation to the input across all its dimensions, so that the output at a specific point is influenced by that point and the points surrounding it. At each position, the filter is multiplied element by element by the corresponding region of the image and the result is added, thus generating an activation map (as is illustrated in Figure 3). The output of a convolutional layer is a set of activation maps, one per filter. These maps capture different characteristics of the image at different locations. Rather than being manually defined, filters are learned during training: their values (weights) are automatically adjusted through backpropagation and optimization so that they capture the most relevant patterns in the data.

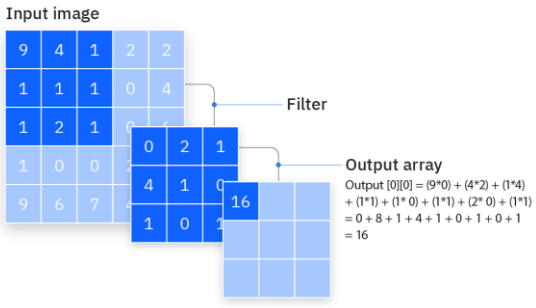

Figure 3. Example of a convolution operation: 3×3 filter slides over the input image, multiplying element by element and summing the result to generate the output activation map [33].

- **Activation layers**: after applying the convolution, an activation function is introduced point-to-point on the resulting activation map. This function introduces non-linearity, to give the model the ability to modulate complex relationships between data. Without non-linear activation functions, the entire network would be equivalent to a single linear operation, severely limiting its learning ability. The most commonly used function today is ReLU, which converts all negative values to zero. This allows for efficient gradient propagation and reduces the gradient vanishing problem observed in functions such as sigmoid or hyperbolic tangent [30].

- **Pooling layers (downsampling)**: their function is to reduce the spatial dimensionality of the activation maps, preserving the most relevant features. This reduces the number of parameters and the computational cost. The most common type of pooling is max pooling. This approach divides the activation map into small regions (e.g., 2x2) and takes the maximum value within this group of pixels, discarding the remainder. This preserves the most important features while reducing the resolution. Other methods that are also used are average pooling, which calculates the average of the region, and min pooling, which retains the minimum value. It has been shown that the use of max pooling leads to faster convergence and allows the selection of more invariant features that have an effect on a larger region of the image due to the reduction in size. In addition, these layers help the model focus on more global and stable features, making it less sensitive to small local variations in the image. On the other hand, reducing the image resolution translates into a decrease in the data to be calculated and, consequently, lower computational memory and time costs. An example of max pooling and average pooling is shown in Figure 4.

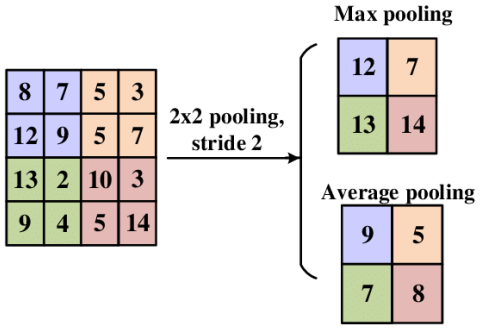

Figure 4. Example of pooling operations in a 2×2 region, showing the different results obtained when applying max pooling and average pooling [34].

- **Interpolation layer (upsampling)**: performs the opposite process to pooling. They are responsible for increasing the spatial resolution of the activation maps, i.e., the number of pixels in the image we are working with. They are particularly important, as they ensure that the network output has the same dimensions as the input image.

- **Fully-connected layer**: are those layers where each neuron is connected to all the neurons in the previous layer. They are used for classification tasks. They are placed at the end of the architecture and their function is to change the dimensionality of the problem. They receive as input the images generated after one or more convolution layers and, depending on the value of each of their pixels, assign a single numerical value per image, which

represents the probability of belonging to a certain class. One or more layers of this type can be placed depending on the level of abstraction required.

In combination, this sequential layer design generates deep architectures that allow to extract first simple and local features, and then, in deeper layers, construct representations of greater complexity and abstraction.

2.4.3 Application of CNNs in computer vision

CNNs have drastically transformed the field of computer vision. Their ability to learn directly from data, without the need for manual feature engineering, has made them the dominant choice for multiple tasks. In medical imaging, CNNs can locate tumors, segment brain tissue, and detect lesions with an accuracy comparable to or superior to that of human experts, even in large datasets and under high anatomical and image quality variability [32]. In image classification, models such as AlexNet, VGG, Inception, and ResNet have achieved accuracies greater than 95%. In the field of semantic segmentation, CNNs are integrated into more complex architectures that combine convolution with upsampling operations to generate pixel-level segmentation masks. Models such as Fully Convolutional Networks (FCNs) and U-Net have demonstrated that it is possible to completely replace dense output layers with transposed convolutional layers, enabling end-to-end segmentation that directly maps an input image to an output mask of the same size [35], [36].

The ability of CNNs to learn spatial representations, their scalability to large volumes of data, and their ease of integration with GPUs has allowed them to become a core technology of most current automated medical image analysis systems. Their versatility and robustness justify their selection as the fundamental models analyzed in this work.

2.5 Neural Network training

Once the architecture of the network we are going to use has been designed, it must be trained. In the case of neural networks, it is a supervised training, in which a training set is used. This training set consists of images with their labels or segmented structures already associated. The training process consists of adjusting the weights of the network connections through an iterative algorithm to minimize the difference or error between the model's predictions and the actual labels. The aim is to achieve the most accurate outputs, i.e., labels on the input data that are as similar as possible to those we have provided in the training set.

2.5.1 Loss function

The loss function calculates the error between the outputs generated by the model and the expected outputs. Choosing the right loss function is especially important in medical segmentation, where minority classes, such as lesions or small tumors, are common. In general, the most commonly used formulas for calculating error are mean square error and mean absolute error [27], [28], [30]. However, in the field of medical segmentation, two other predominant functions emerge. Cross-entropy loss, which measures the discrepancy between the predicted and actual classification for each pixel. And Dice loss, which measures the overlap between the prediction and the actual label, being especially useful in contexts with unbalanced classes [27], [30].

2.5.2 Optimization

Once the error has been calculated using the loss function, it must be reduced as much as possible. To minimize the error, iterative optimization algorithms are used that bring us closer to the ideal solution by progressively adjusting the

weights of the network. The most common and fundamental is gradient descent, which calculates how the loss function (the error) changes with respect to each of the weights and adjusts these weights in the direction that most reduces the error. The gradient is a vector that indicates the direction of greatest growth of a function. Therefore, to minimize loss, we must move in the opposite direction to the gradient [26], [30]. Mathematically, each weight is adjusted according to:

$$w_{t+1} = w_t - \gamma \cdot \frac{\partial L}{\partial w}$$

Where "w" is the weight to be updated, "L" is the loss function, "$\partial L/\partial w$" is the gradient of the loss with respect to that weight, and "γ" is the learning rate, which controls the size of the step in each update [27].

The backpropagation algorithm is used to calculate all these gradients. This algorithm crosses the network from the output layer to the input layer, calculating how each weight contributes to the total error using the chain rule of differential calculus. The corresponding gradient is stored in each layer, which is then used to update the weights. This entire process: forward propagation, error calculation, backpropagation, and weight updating; is repeated many times during training [26], [27], [28].

The learning rate is a parameter of vital importance for achieving minimum error. A very high value can cause the algorithm to fail to converge, rendering training useless. Meanwhile, an excessively low value ensures that the algorithm converges, but not necessarily to the optimal solution. Moreover, since the step at each iteration is very small, the solution will be reached very slowly, requiring a large number of iterations. This can lead to unnecessarily long training times [27], [28], [30].

In addition to classic gradient descent, there are more advanced variants that automatically adapt the learning rate, improving its stability and convergence speed [30]:

- **SGD** (Stochastic Gradient Descent): updates the weights after each sample or mini-batch, introducing a degree of randomness that can help escape local minima.
- **Adam** (Adaptive Moment Estimation): automatically adjusts the learning rate for each weight, based on the mean and variance of the previous gradients.
- **RMSProp**: adapts the learning rate by dividing by a moving average of the gradient square, useful for problems with noisy data.

Choosing a good optimizer and an appropriate learning rate is essential to achieve effective convergence and avoid problems such as overfitting or learning plateau.

2.5.3 Stages

The aspect to decide is the type of training to be carried out and the number of iterations. The number of epochs represents the number of times the model processes the entire training set. In contrast, the number of iterations corresponds to the number of times the network weights are updated, which depends on the batch size. The relationship between these concepts is given by:

$$\text{Number of iterations per epoch} = \frac{\text{Total number of samples}}{\text{Batch size}}$$

For example, if you train with a dataset of 1,000 images and use mini-batches of 100 images, one epoch will consist of 10 iterations. If you train for 5 epochs, you will have performed 50 iterations in total.

The importance of defining an appropriate number of epochs and iterations lies, first, in ensuring that the model has enough opportunities to learn the problem correctly. And second, in avoiding overfitting, which occurs when the network memorizes the training data and loses its ability to generalize to new data. To mitigate overfitting, it is useful to employ a validation set. This does not interfere in the adjustment of weights, but allows the performance of the model to be evaluated in each epoch and techniques such as early stopping to be applied [28], [30].

There are different training strategies depending on the batch size:

- **Stochastic Training**: in this mode, the neural network updates its weights after processing a single sample from the training set. This means that, in one epoch, the number of iterations is equal to the number of samples. It has the advantage of being very fast and introducing randomness that can help escape local minima. But it also produces very noisy updates and can hinder the stable convergence of the model.
- **Batch Training**: First, the error is calculated over the entire dataset, and based on this total error, the network weights are updated. So, a single update of the weights is performed per epoch. It provides a very accurate estimate of the gradient, as it reduces the error for all samples. However, it is the most computationally expensive and requires the most memory. Only one iteration is performed per epoch, so many epochs are required to obtain the correct result, which slows down the optimization process.
- **Mini-batch Training**: instead of using all samples or just one, this method divides the training set into small groups (e.g., 32 or 64 samples). For each batch, the error is calculated and the weights are updated. The larger the batch size, the more accurate the gradient estimate, but also the higher the computational cost. It is the most common method as it combines efficiency and stability and allows for better use of hardware resources such as GPUs. It provides an adequate balance between accuracy, speed, and memory usage and contributes to more stable convergence in deep neural networks.

The training of a neural network is composed by several well-defined stages that are repeated over multiple epochs until convergence is reached or a stopping criterion is met. These stages are:

1. Initialization: random initial values are assigned to the network weights.
2. Forward pass: a sample or batch is entered and the model output is calculated.
3. Loss evaluation: The obtained output is compared with the expected output using a loss function.
4. Backward pass: The gradient of the error with respect to the weights is calculated using backpropagation.
5. Weight update: the parameters are adjusted using an optimizer (such as SGD, Adam, etc.).
6. Validation: the model's performance is evaluated on a validation set to monitor for possible signs of overfitting.

This cycle is repeated iteratively for the set number of epochs, with the goal of minimizing error and improving the model's accuracy [28], [30].

2.6 Models of deep segmentation

The evolution of deep neural networks has led to the development of various architectures specialized in semantic segmentation. Among them, models that combine accuracy, efficiency, and versatility stand out and have been widely adopted in clinical and research settings. This section describes three of the most relevant architectures for medical image segmentation: U-Net, DeepLabV3, and FCN.

2.6.1 U-Net

The U-Net architecture was originally proposed by Ronneberger et al. in 2015 for biomedical image analysis [37]. It has become the benchmark model in semantic segmentation of medical images due to its ability to preserve and combine both global contextual information and local details.

Its design is based on a symmetric encoder-decoder structure. The "contraction" branch (encoder) consists of blocks of convolutional layers, ReLU activation functions, and max pooling operations. This combination progressively reduces the spatial resolution of the image while increasing the depth of the activation maps. By increasing the dimensionality of the features, it allows the model to capture high-level contextual information (global features). For its part, the "expansion" branch (decoder) includes upsampling layers that restore the spatial resolution of the image. These are combined with additional convolutions to refine the segmentation. They allow predictions to be generated at the pixel level through pixel-by-pixel segmentation. The main innovation of U-Net lies in the incorporation of skip connections. These directly connect each level of the encoder with its corresponding level in the decoder, transferring detail-rich activation maps. They ensure that fine details and precise edges are not lost during the encoding and decoding process. This compensates for the loss of spatial information caused by pooling and improves segmentation accuracy, especially in small structures or those with diffuse edges. This is particularly relevant in medical images, where the structures of interest can be small, with diffuse edges and high interpatient variability, requiring networks capable of combining local precision with global context, something that U-Net achieves through its modular design based on CNNs [38].

U-Net has demonstrated outstanding performance in medical tasks such as the tumor segmentation, organ delineation and lesion detection. In addition, its simplicity, ability to train with limited data, and modular structure have encouraged its adoption in multiple variants. Some of these, such as U-Net++ and U-Net v2, introduce denser, bidirectional skip connection mechanisms with the aim of greater integration of semantic and detail features. These variants have demonstrated significant improvements in reference datasets and greater computational efficiency [39], [40].

2.6.2 DeepLabV3

DeepLabV3 is a semantic segmentation architecture designed by Chen et al. in 2017 [41] to effectively capture multiscale context and improve the delineation of objects with different sizes and complex shapes. It represents an evolution of the DeepLab family, introducing two core elements: atrous (dilated) convolutions and the Atrous Spatial Pyramid Pooling (ASPP) module. Atrous convolutions modify the way convolutional filters sample the input by inserting spaces between kernel elements. This increases the receptive field without adding parameters or reducing the spatial resolution. The ability to enlarge the field of view while keeping fine details is particularly useful in medical images, where anatomical structures can vary greatly in size and morphology. The ASPP

module applies multiple atrous convolutions in parallel, each with a different dilation rate, and augments them with global image-level pooling. This design allows the network to encode both local and global context efficiently, which is crucial for segmenting structures that appear at different scales or with diffuse boundaries [41]. Structurally, DeepLabV3 is built on top of a convolutional backbone such as ResNet or Xception. The atrous convolutions and ASPP are applied to the final feature maps, and the output is then upsampled through bilinear interpolation to obtain dense, pixel-level predictions. Unlike encoder–decoder models such as U-Net, DeepLabV3 does not rely on skip connections; instead, it achieves accurate segmentation through its multiscale feature encoding.

Thanks to this design, DeepLabV3 achieved state-of-the-art results on benchmarks such as PASCAL VOC 2012 and Cityscapes [42]. In the medical imaging domain, it has been successfully applied to tasks such as brain tissue and organ segmentation, often surpassing U-Net in complex scenarios. However, the model requires considerable computational resources and careful training to avoid overfitting [43], [44].

2.6.3 Fully Convolutional Network (FCN)

The Fully Convolutional Network (FCN), introduced by Long et al. (2015) [42], was the first deep learning architecture specifically designed for semantic segmentation. Unlike conventional CNNs, which end with fully connected layers for classification, FCN replaces these with convolutional layers, allowing the model to produce dense, pixel-wise predictions while preserving spatial information. To recover resolution lost during successive downsampling operations, FCN incorporates upsampling (deconvolution) layers that progressively reconstruct the spatial dimensions of the feature maps. In addition, it employs skip connections, combining coarse, high-level semantic information from deeper layers with fine, low-level details from earlier ones. This mechanism improves boundary delineation and helps the model retain structural details that would otherwise be lost.

Although more recent architectures such as U-Net and DeepLabV3 have surpassed it in performance, FCN remains a fundamental baseline due to its simplicity and efficiency. In medical image segmentation, it provides a solid reference point for evaluating more advanced approaches, especially in contexts where computational resources are limited. It manages to accelerate both training and inference, reducing memory usage and time without significantly sacrificing accuracy. In general benchmarks such as Cityscapes or ADE20K, it has shown results comparable to DeepLabV3, especially in tasks where the global context does not require mechanisms as sophisticated as ASPP [42]. However, its reduced ability to explicitly integrate multiscale context may limit its performance in complex images, for example, in structures with diffuse edges, high morphological variability [45].

2.7 Evaluation of segmentation models

Rigorous evaluation of segmentation models is essential for validating their performance and comparing different architectures. In the context of medical images, the accuracy and fidelity of predictions is critical to ensuring clinical validity and comparability between studies. To this purpose, metrics are used to quantify the degree of coincidence between the segmentation predicted by the model and the reference or ground truth. It is essential to use quantitative metrics that reflect both the overall quality of the model and its specific behaviour in minority classes or difficult contours. Before introducing the main

evaluation metrics used in semantic segmentation, it is necessary to define the basic classification terms, which are graphically summarized in Figure 5.

- True positive (TP): when the value labelled as belonging to the region actually belongs to it.
- False positive (FP): when the value labelled as belonging to the region does not actually belong to it.
- True negative (TN): when the value labelled as not belonging to the region does not actually belong to it.
- False negative (FN): when the value labelled as not belonging to the region actually does belong to it.

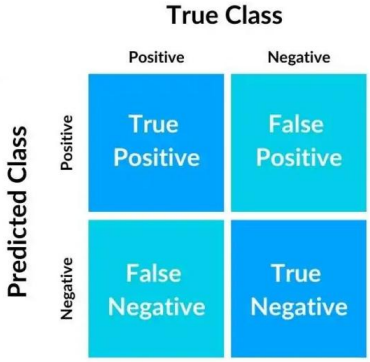

Figure 5. Confusion matrix representation illustrating the four possible outcomes in binary classification: true positive (TP), false positive (FP), true negative (TN), and false negative (FN).

2.7.1 Dice coefficient

The Dice coefficient is one of the most widely used metrics in medical segmentation. It measures the overlap between two sets, in this case between the region segmented by the model and the reference segmentation. It is defined as:

$$\text{Dice} = \frac{2 \cdot |A \cap B|}{|A| + |B|}$$

Where "A" represents pixels predicted as positive (automated segmentation) and "B" represents actual pixels (ground truth). The coefficient value varies between 0 and 1. A value equal to 1 indicates perfect overlap, while 0 indicates a total absence of coincidence. In medical images, values greater than 0.8 are considered indicative of good concordance [46]. This metric is especially useful when working with unbalanced classes, in which the region of interest is much smaller than the background. As this is a typical situation in brain images with lesions or tumors, its use is widespread. In addition, it is highly sensitive to errors, penalizing them heavily.

2.7.2 IoU (Intersection over Union)

The Intersection over Union, also known as the Jaccard Index, measures the ratio between the overlap area and the total combined area between the prediction and the ground truth. Its formula is:

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|}$$

Where the denominator is the total number of pixels belonging to either of the two masks. And the numerator is the total number of pixels belonging to both masks. Its interpretation is similar to the one of the Dice coefficient, but IoU tends to give lower values, as it penalizes prediction errors at the boundary and

overestimation of the segmented area more severely [47]. In summary, it is a robust metric, especially useful in multicategory evaluations.

2.7.3 Pixel accuracy

Pixel accuracy is a simple metric that calculates the ratio of correctly classified pixels to the total number of pixels.:

$$\text{Pixel Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

where "TP" (true positives) and "TN" (true negatives) represent pixels correctly classified as belonging or not belonging to the target class. Meanwhile, 'FP' (false positives) and "FN" (false negatives) represent classification errors. Although easy to interpret and compute, this metric can be misleading in contexts with highly unbalanced classes. For example, if a class represents only 5% of the pixels in an image, a model that simply predicts background for all pixels can achieve 95% accuracy without providing any clinical value [48]. For this reason, it is recommended to use it in conjunction with metrics such as Dice or IoU, which more specifically penalize errors in the region of interest.

2.7.4 Sensitivity and Specificity

In clinical contexts where it is necessary to evaluate a model's ability to detect pathological structures, it is common to use additional metrics such as sensitivity (recall) and specificity:

$$\text{Sensibility} = \frac{\text{TP}}{\text{TP} + \text{FN}} \qquad \text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

Sensitivity measures the model's ability to correctly identify positive pixels (avoiding false negatives). In other words, it is interpreted as the model's ability to not miss any relevant structures or lesions. On the other hand, specificity indicates how well the model avoids classifying pixels that are not positive as positive (avoiding false positives). These metrics are especially relevant in clinical applications where errors can have different clinical consequences. Missing a lesion can have serious consequences, so it is preferable to oversegment a lesion (have more false positives) than to miss it (false negative). A good balance between sensitivity and specificity indicates that the model detects most of the true structures of interest without an excess of false detections.

2.8 Current limitations of deep learning models in medicine

Despite the significant advances that deep neural networks have brought to medical image analysis, their practical application continues to face multiple limitations that condition their clinical adoption. These limitations not only affect the accuracy of the models, but also their scalability, security, interpretability, generalization capacity, and reliability in real-world scenarios. Three of the most relevant challenges are described below.

2.8.1 Lack of annotated data

The performance of deep learning models is closely linked to the quantity and quality of the data used during training. In the medical field, obtaining large volumes of correctly annotated images is particularly complicated. This difficulty stems from the fact that labels must be generated by qualified professionals, often through costly and time-consuming manual procedures. This issue becomes particularly exacerbated in low-prevalence pathologies, as well as in pediatric contexts or minority populations. Unlike other fields such

as general computer vision, where datasets with millions of examples exist, in medicine, datasets are often small, heterogeneous, and fragmented by institution or pathology [7], [38].

This scarcity of high-quality annotated data limits both the training of robust models and their adequate validation in real-world scenarios. The ability of algorithms to learn interpatient variability is reduced, increasing the risk of overfitting to specific characteristics of the training set and hindering the transfer of the model to other populations, centres, or imaging modalities. Ultimately, it prevents the development of highly generalizable solutions. All of this underscores the need to develop learning techniques that work with smaller amounts of data, such as transfer learning or semi-automated and self-supervised annotation strategies, which will be studied in this work.

2.8.2 Class imbalance

In many medical segmentation tasks, regions of interest represent a very small proportion of the total image. For example, a brain tumor may occupy less than 5% of the total volume of an MRI scan, while the rest corresponds to healthy structures or background [38]. This significant class imbalance results in models tending to favor the majority classes, ignoring or underrepresenting smaller or less frequent structures. This reduced sensitivity for detecting small structures or lesions has a negative clinical impact. Furthermore, the lack of representation of minority subtypes, ethnicities, or anatomical patterns in the data can perpetuate biases and reduce the reliability of the model in vulnerable subgroups [49].

As a result, global metrics such as pixel accuracy can be misleading, since a model that predicts “background” for all pixels can obtain a high score, despite not adequately segmenting the relevant class. To mitigate this effect, it is common to use class reweighting, specific metrics such as the Dice coefficient, or synthetic data generation strategies through data augmentation focused on minority classes [50].

2.8.3 Generalization and overfitting

Another important limitation is the lack of generalization ability of trained models, especially when evaluated on data from institutions, devices, or populations other than those in the training set [51]. This problem, known as domain shift, is accentuated in medicine due to the high variability between patients, the diversity of acquisition protocols, and differences in image quality and resolution between centres. Furthermore, when models are trained with limited data or homogeneous datasets, there is a high risk of overfitting, meaning that the model learns specific patterns from the training set but fails to extrapolate them to new cases. This phenomenon compromises clinical applicability and can lead to false confidence in predictions.

Addressing these challenges requires strategies such as collecting multisite datasets (data from different sources), using image normalization and harmonization techniques, incorporating regularization during training, and designing architectures that are more resilient to domain variability.

2.9 Complementary techniques for improving performance

To address these limitations faced by deep segmentation models in medicine, various complementary strategies have been developed that seek to improve their performance without substantially modifying the architecture. These

techniques make it possible to enrich the dataset, facilitate training, or increase accuracy without compromising computational efficiency.

2.9.1 Data augmentation

Data augmentation is a fundamental technique, particularly useful when the volume of labelled biomedical images is limited, as is often the case in medicine. It consists of applying controlled transformations to the images in the training set in order to generate new artificial samples that maintain the original semantics. In the context of medical segmentation, when both the image and its corresponding mask are available, these transformations must be applied jointly to ensure that the spatial correspondence between them is preserved.

The most common techniques include rotations, translations, scaling, horizontal or vertical flips, brightness or contrast changes, Gaussian noise addition, and elastic deformations [50]. These operations increase the diversity and size of the training set without the need to obtain new annotated data. This helps reduce overfitting, improves generalization ability, and allows the model to learn to be more robust to variations not seen in the actual data.

2.9.2 Semiautomatic annotation models

Another strategy to tackle the scarcity of high-quality annotated data is the use of automated segmentation methods. A notable example is Segment Anything Model (SAM), a semi-automatic annotation model developed by Meta AI. SAM is a segmentation model trained on a massive and diverse set of images, capable of generating initial masks from images and reference points [38]. The usefulness of SAM in the biomedical field lies in its ability to accelerate and facilitate the manual annotation process. Although it does not completely replace expert judgment, it does allow preliminary segmentations to be generated quickly, which can then be refined by specialists. This significantly reduces the time and effort required to create labelled datasets, while standardizing the process and reducing inter-annotator variability.

In clinical and research applications, SAM is being used to create segmentation databases for magnetic resonance imaging, tomography, and microscopy, among other modalities. It allows medical teams to generate large volumes of masks with less workload. In this way, SAM is a key complementary technique to the problem of annotated data scarcity, as it combines the clinical accuracy of the expert with the efficiency of a pre-trained automatic system.

3 Methodology and development

3.1 Introduction to the methodological approach

The methodology followed in this work has been designed with the aim of ensuring reproducibility and clarity in the description of each experimental stage. To this end, it is structured as a systematic pipeline designed to ensure the correct preparation of data, the implementation of different segmentation models, and the objective evaluation of their performance on magnetic resonance images. In general terms, the workflow has been structured around three complementary experimental approaches:

- **Approach 1**: direct training on the dataset used in this work, after generating masks using semi-automatic techniques and the corresponding standardization.
- **Approach 2**: use of a reference public dataset for model pretraining and subsequent fine-tuning on our dataset.
- **Approach 3**: application of data augmentation techniques to our dataset in order to increase the variability of the samples and evaluate their impact on the generalization of the models.

The same segmentation architectures have been implemented in all three approaches and evaluated using a common set of metrics, ensuring the comparability of the results obtained. The overall pipeline followed in this study is summarized in Figure 6, highlighting the three experimental approaches considered, and the convergence in common evaluation stage and final comparison of results.

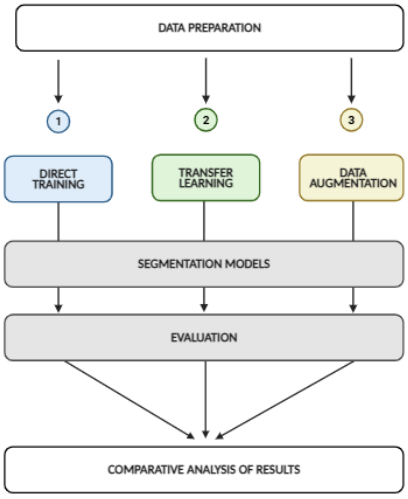

Figure 6. General workflow of the study, illustrating the three experimental approaches.

Likewise, the experimental development was carried out on a high-performance computing server equipped with 8 A100 GPUs, belonging to the CITSEM centre (UPM). These computers are equipped with graphics processing units (GPUs) that significantly accelerate the training of deep learning models. In terms of software, work was carried out in an environment based on Python 3.10 [52], mainly using the PyTorch library [53] for the implementation of neural networks, Label Studio software [54] for annotation and mask management, and a set of auxiliary visualization and analysis libraries (Matplotlib, NumPy, Pandas). The workflow was organized through custom training and evaluation scripts, which automatically generated metrics, performance curves, and predictions for further analysis. In addition, BioRender was employed to design the schematic figures included in this thesis.

3.2 Approach 1: Internal Dataset

3.2.1 Structure and composition of the dataset

The first experimental approach in this study was based on the use of a dataset generated in the laboratory of the CITSEM (UPM) research group. This dataset does not derive from human patients, but from laboratory-designed phantoms made with agar-agar gels at different concentrations. They were specifically created to simulate brain structures by reproducing MR tissue response based on distinct T1 and T2 relaxation times. Each compartment (white matter, grey matter, ...) was formulated with a different mixture, reproducing distinct relaxation times and appearing with different contrasts in the MR images, as described in similar phantom studies [55]. This approach enabled the generation of reproducible magnetic resonance data under controlled laboratory conditions. The images (see Figure 7) were obtained using a preclinical 9.4T MRI scanner available in the group's laboratory.

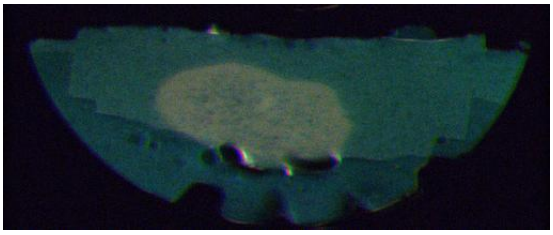

Figure 7. MRI slice of the laboratory-designed phantom used in this study.

Each phantom underwent different acquisition modalities, similar to what occurs in clinical studies, which provided complementary information for the segmentation of structures:

- T1-weighted: highlights the anatomy and provides good contrast between white and grey matter.
- T2-weighted: emphasizes the presence of fluid, which is useful for differentiating pathological structures.
- DP (Proton Density): reflects the density of protons in the tissue, providing complementary information to T1 and T2.

The use of these three modalities provided multimodal information for each image. From this multimodal information, two complementary versions of the dataset were organized: on the one hand, one version with the T1, T2, and DP modalities separated as independent images; and on the other hand, a merged version in RGB format, in which each modality was assigned to one of the three color channels (red, green, blue). In practice, this meant stacking the T1, T2, and DP slices into a single three-channel image, where each channel contained the intensity values of one modality. This fusion strategy allowed the models to process all modalities simultaneously and to exploit the complementary information they provide. The latter dataset, in which the three modalities are combined into a single image, will be the one used in this project.

Furthermore, the study includes images from four different phantoms. Each phantom which will be treated throughout the project as independent patients, so we have four patients. For each phantom, slices were acquired in two main planes, axial and coronal. Specifically, 60 slices were obtained for each phantom, 35 in the axial plane and 25 in the coronal plane. This resulted in a total of 240 multichannel images in the merged dataset. Likewise, the dataset consists of two parts. The first part contains these magnetic resonance images in .png format. While the second part includes the corresponding segmentation masks.

Initially, not all the images in the dataset had a corresponding mask available, so a mask generation and standardization process had to be carried out, which will be explained in the next section. In terms of the classes segmented in each image, the dataset included six relevant categories: tumor, white matter (WM), grey matter (GM), blood vessels (BV), external markers, and background.

Thus, although it was a synthetic dataset, the multiclass composition of the phantoms allowed us to address a realistic segmentation problem, incorporating both large, regular structures (such as white matter) and small regions with greater variability (tumor, vessels). Which ultimately enabled us to achieve the objective of training and evaluating different segmentation models directly on these images.

3.2.2 Generation and standardization of masks

The creation of consistent segmentation masks was an essential step in training and evaluating deep learning models. Originally, not all images in the dataset were labelled, so it was necessary to generate masks for the remaining images. To ensure the quality of the dataset, this process was applied only to those images in which the classes of interest were clearly visible, excluding those without relevant information or those that were confusing. To do this, a strategy based on the combination of Label Studio and the Segment Anything Model (SAM) was used. SAM is a model able of generating mask proposals from simple prompts, such as points or selected areas in the image. This combination allowed annotations to be generated semi-automatically and with a sufficient level of manual control to ensure final quality.

Before starting to generate the masks, it was necessary to prepare the annotation environment. First, Label Studio was installed on the server. To ensure compatibility with the SAM backend, version 1.13.1 had to be used instead of more recent versions. Regarding data access, it was managed using Label Studio's *Local Storage Import* function. This feature allows direct interaction with system files without the need to manually load them one by one. To achieve this, access to local image files was enabled through environment variables, following the configuration guidelines provided in the official Label Studio documentation [56]. This way, images stored in the dataset directory are automatically imported and can be viewed directly from the Label Studio web interface. This maintains complete traceability of the paths and ensures consistency between the original database and the generated annotations.

Subsequently, the Segment Anything Model was installed, which runs as an inference service, i.e., a program that receives input images and automatically returns segmentation masks as output. For this purpose, we used Docker, a technology that allows applications to be encapsulated together with all their dependencies within containers, ensuring portability and reproducibility. The SAM backend was configured from the official GitHub repository *label-studio-ml-backend*, specifically the example available at *label_studio_ml/examples/segment_anything_model* [57]. This backend was run using Docker Compose, a tool that allows the resources, environment variables, and ports necessary to launch the service automatically to be defined in a *docker-compose.yml* file. The following relevant parameters were specified in this file:

- The use of the container was restricted to a specific server GPU to avoid conflicts with other users.

- A unique container name and its own communication port were defined in order to isolate the service from other SAM instances that might be running.
- The access credentials for Label Studio were added using the environment variables LABEL_STUDIO_HOST (the address of the main Label Studio server) and LABEL_STUDIO_ACCESS_TOKEN (personal authentication token).
- Finally, the model variant to be used was specified, prioritizing the full SAM model over reduced versions such as MobileSAM.

Once this file was defined, the service was built and launched using the commands *the docker compose build* and *docker compose up*. As a result, the SAM backend became available as a background service, accessible via a URL.

Once the backend was deployed, Label Studio allowed the registration of SAM as an external machine learning service through its web interface, using the *Add model* option and providing the container URL. From that point on, Label Studio and SAM were fully integrated. Each user click on an image within the Label Studio interface generates a request to the SAM backend, sending the selected image and the coordinates of the click point. SAM then processes the request, produces a segmentation mask, and returns it in real time for visualization and review. In summary, the procedure followed for each image was as follows:

1. Loading the image into Label Studio from the dataset folder.
2. Automatic generation of segmentation proposals using SAM, with reference points placed on the structures of interest (tumor, WM, GM, vessels, etc.).
3. Visual review and adjustment of the proposed masks, correcting inaccuracies in boundaries or ambiguous regions between anatomical classes.
4. Manual validation of the final mask, ensuring that each class was correctly represented.
5. Exporting the masks in PNG format, with a specific color assigned to each class.

The use of SAM provided a clear advantage in terms of speed and consistency. As a result of this workflow, a complete set of multiclass masks was obtained, consistent with the six defined categories.

3.2.3 Dataset Preprocessing

After the generation of the masks, a preprocessing stage was required to consolidate and standardize the dataset before using it in the segmentation models. This workflow aimed to transform the heterogeneous outputs from Label Studio into consistent multiclass masks, aligned in both format and color palette with the existing masks, thus ensuring the overall consistency of the dataset.

First, the annotations obtained from Label Studio were exported in PNG format. Each export generated multiple binary files, one per class and per image, along with a metadata file in JSON format. The first step was to merge all these binary layers into a single multiclass mask, assigning each pixel an integer identifier corresponding to its respective class. To resolve potential overlaps between layers, a priority criterion was established in which certain classes overwrote others in case of conflict (for example, external markers prevailed over the background, or the tumor over grey and white matter). This process resulted in one final mask per image, in greyscale, where the values 0–5 represented the

six categories defined in the project (background, tumor, white matter, grey matter, blood vessels, and markers).

The next step consisted of applying a predefined color palette to these indexed masks. To this end, the actual palette from the existing masks was first extracted and subsequently applied to the newly generated masks. The working palette assigned red to the tumor, blue to white matter, grey to grey matter, pink to blood vessels, and green to markers. This conversion into RGB format proved very useful in the visual inspection and validation stage, as it made it easy to quickly check whether each anatomical structure was labelled with the expected color. With this step, all masks were ensured to share exactly the same palette and encoding mode, thereby guaranteeing homogeneity across the entire dataset.

Subsequently, the pairing between original images and their corresponding masks was addressed. A key decision was to include in the final dataset only those images for which a mask was available, in order to maintain a strict 1:1 relationship between image and annotation. Since the filenames exported from Label Studio did not match those of the original images, it was necessary to implement a renaming and copying procedure, in which correspondence was stablished based on patient identifier, acquisition plane (axial or coronal), and slice number. Once this naming issue was resolved, a final dataset was assembled with the images in one folder and the masks in another, ensuring that the number of files was identical and that each image had its corresponding mask.

Next, a quality check on the assembled dataset was performed. This verification included ensuring that all images and masks had the correct spatial resolution, and that they corresponded to the defined palette. The result of this process was a final dataset consisting of 187 image-mask pairs, corresponding to the four patients (phantoms). In other words, after filtering only the representative images, i.e., those containing of relevant classes, the dataset size was reduced from 240 to 187 images. All images had a uniform resolution of 512×512 pixels, and the masks were encoded in palette mode with a direct correspondence between indices and classes. As a result, the dataset was fully standardized and ready to be used in the training and evaluation experiments of the segmentation models.

3.2.4 Implementation of segmentation models

With the dataset ready, the implementation of the segmentation models was carried out. In this work, three reference architectures for semantic segmentation in the biomedical field were employed: U-Net, DeepLabV3, and FCN. Each presents a characteristic design, but all were adapted to address the multiclass problem defined in this project.

The U-Net network used in this work is a custom implementation based on the encoder-decoder scheme with skip connections. The encoder consists of DoubleConv blocks (conv3$\times$3 $\rightarrow$ BN $\rightarrow$ ReLU, repeated twice) followed by max-pooling for resolution reduction. The decoder performs bilinear upsampling and concatenates the encoder features (skips) before applying another DoubleConv, correcting potential spatial mismatches through padding and ending with a 1×1 layer (OutConv) that projects into the class space. In our configuration, the model was instantiated with three input channels (T1–T2–DP fusion) and six output channels (one map per class), so that the last 1×1 convolution directly produces the per pixel logits for the six categories in the dataset. This adaptation

is reflected during training, where *UNet(n_channels=3, n_classes=6)* is invoked and optimized with *Cross-Entropy Loss* over the indexed label maps.

For DeepLabV3, the variant with ResNet-50 backbone from torchvision was used. It integrates an Atrous Spatial Pyramid Pooling (ASPP) module to capture multiscale context and a lightweight decoder block for boundary refinement. The adaptation to the multiclass problem was performed at model construction by setting *num_classes=6*, which configures the final classification head (1×1 convolution from torchvision) to output six logit maps. During the forward pass, the tensor is retrieved as *model(images)['out']* to compute the loss against the indexed ground truth.

Finally, in the case of FCN, the *fcn_resnet50* implementation from torchvision was used. Its design replaces the fully connected stage with a fully convolutional head and employs skip connections from the backbone to enhance semantic resolution. Similar to DeepLabV3, the adaptation to the six-class set was specified with *num_classes=6*, ensuring that the final head generates a probability map per class at dense resolution. The forward pass outputs a dictionary, from which the key 'out' is used to obtain the logits for Cross-Entropy loss computation.

Overall, the three architectures share the same RGB input dimensionality (3 channels) and an output layer parameterized to six classes, ensuring direct comparability between models and consistency with the preprocessing and labelling scheme of the dataset.

3.2.5 Training and Validation

In this approach, model training was carried out directly on the dataset of the research group. To ensure fair evaluation and avoid data leakage, the dataset was split into three subsets: training, validation, and test. In order to preserve independence between patients and prevent images from the same subject from being distributed across different subsets, all data from patient 3 were allocated exclusively to the test phase. In this way, the final evaluation would more realistically reflect the generalization capacity of each architecture. The remaining patients were distributed between training and validation following an approximate ratio of 80% and 20%, respectively, ensuring an adequate balance between the amount of data for learning and a reliable set for hyperparameter tuning.

Focusing on the hyperparameters, a homogeneous configuration was employed across the three models to guarantee comparability of results. First, multiclass Cross-Entropy Loss was used as the loss function, which is appropriate when segmentation masks are represented as maps of integers indexed by class. For optimization, Adam was used, as it combines adaptive learning rate with implicit regularization, offering a good compromise between convergence speed and stability. The learning rate was set at 1×10^{-4}, chosen after preliminary tests: higher values produced oscillations in validation loss, while lower values slowed convergence excessively. Training was executed for a maximum of 50 epochs, sufficient to allow the model to converge without incurring prolonged overfitting. An early stopping criterion was also applied, halting training if no improvements in validation were observed for five consecutive iterations. In addition, the model with the lowest validation loss throughout the epochs was recorded, so that the final test evaluation would always be performed on the best state achieved. The batch size was set to 4 images, a compromise between having a minimum number of examples per iteration and the GPU memory available on the servers.

A larger batch would have saturated the VRAM memory when handling 512×512 images with three input channels, while a smaller batch (e.g., 1 or 2) would have generated too much noise in the gradient estimation and slowed down learning. Overall, the training configuration was designed to maximize learning stability, mitigate the risk of overfitting, and ensure that the three models could be compared under a consistent experimental protocol.

After this initial training phase using a uniform configuration, a second round of experiments was conducted in which each architecture was individually adjusted based on its observed performance. These model-specific improvements, described in detail in the results section and summarized in Table 2, were aimed at enhancing the performance of each network by addressing their specific limitations.

3.2.6 Evaluation of the approach

The evaluation in this first approach was carried out following the same metrics scheme described in section 2.7. For each model, Pixel Accuracy, Dice coefficient, and Intersection over Union (IoU) were calculated. All reported metrics were computed per class and then averaged using a macro-average strategy, i.e., the arithmetic mean across all classes, giving each class equal weight regardless of its pixel frequency. These evaluations were performed on the test set, always using the best checkpoint according to the validation loss. These metrics were selected for their ability to reflect both overall performance and specific accuracy in minority classes, which are particularly relevant in the medical context.

In addition to numerical metrics, training curves were generated to illustrate the evolution of the loss and the performance of metrics across epochs. This enable the analysis of phenomena such as overfitting or premature convergence. Furthermore, comparative visualizations were produced, showing the original MRI, the reference mask, and the model prediction. All of this allowed to complement the quantitative evaluation with a qualitative analysis of each architecture's performance.

Overall, this approach made it possible to establish a baseline reference for the performance of the selected architectures when applied exclusively to this dataset, serving as a starting point for comparison with the subsequent approaches based on pretraining and data augmentation.

3.3 Approach 2: Transfer Learning

3.3.1 Selection of the public dataset

The second methodological approach was based on exploring transfer learning. It consisted of pretraining the architectures on a public magnetic resonance dataset widely used for tumor segmentation tasks, follow by fine-tuning on our own phantom dataset. This mechanism was expected to improve the initialization of the models before exposure to the specific distribution of our data. This is because it takes advantage of initial weights adapted to the MRI domain (textures, contrasts, and typical morphologies) and mitigates the limited size of our own dataset.

For this purpose, the public dataset Brain Tumor Segmentation Challenge 2020 (BraTS2020) [58] was selected. This dataset is widely recognized within the scientific community and systematically used as a benchmark for semantic segmentation tasks in MRI. BraTS2020 includes MRI studies of a large number

of patients with gliomas of varying grades, providing a much greater variety and volume of data than the available in our dataset. Each patient is represented through four acquisition modalities: T1, T1c, T2, and FLAIR, which together offer a multimodal view of the brain anatomy and tumor regions. The annotations are provided by clinical experts and include different subregions of the tumor, making this dataset a particularly useful resource for pretraining segmentation models aimed at detecting complex structures in MRI. Although the class definitions differ from those used in the present project, it was considered that the features learned from real patient data could be effectively transferred during the finetuning phase on our dataset.

3.3.2 Preprocessing and homogenization

Once BraTS2020 was selected as the pretraining resource, the first step was to download the dataset directly into the server. For this, the Kaggle API was used to automate the download process, avoiding manual file transfer and ensuring immediate availability in the training environment. After downloading, the compressed file was extracted into the corresponding directory, making the files available in .h5 format with two-dimensional MRI slices and their corresponding masks. Since our dataset was also organized into 2D slices in PNG format, it was considered appropriate to adapt BraTS2020 to the same scheme. Therefore, a preprocessing pipeline was implemented to homogenize both datasets, ensuring compatibility with the models and consistency with the workflow used in the first approach, avoiding the need to redesign the training pipeline.

To achieve this, the .h5 files were inspected with an exploration script to verify the image dimensions, number of channels, and mask encoding. Once the presence of image files and their corresponding masks was confirmed, both were systematically converted into PNG format. Of the four modalities included in BraTS2020 (T1, T1Gd, T2, and FLAIR), T1Gd was discarded in order to ensure consistency between pretraining and finetuning. For the images, the T1, T2, and FLAIR modalities were combined into a single PNG file in RGB format, with each modality assigned to one color channel. Since the original slices in BraTS2020 had varying in-plane dimensions depending on the patient, all images were resized to a uniform grid of 512×512 pixels using bilinear interpolation for the MRI channels (*Image.BILINEAR*) and nearest-neighbor interpolation for the masks (*Image.NEAREST*). Pixel intensities were then normalized to the 0–255 range to match the format of our own dataset. This step was essential to ensure that models could be trained identically on both datasets, without requiring changes in their internal architecture.

Regarding segmentation masks, BraTS2020 contains multiclass annotations aimed toward identifying tumor subregions. These masks were converted to indexed format, where each pixel is represented by an integer value corresponding to its class, as was done for our own dataset. Although the class scheme does not match that defined for the phantoms (BraTS employs tumor-specific categories instead of the six classes used in our project), this representation enabled the models to learn general discriminative features in MRI, which could later be transferred during finetuning to our own dataset.

A major difficulty in BraTS2020 is the strong class imbalance, with an almost absolute dominance of the background (≈98.7%) compared to the low representation of tumor regions (classes 1, 2, and 4). Several strategies were considered to mitigate this issue: defining class weights in the Cross-Entropy Loss so that rare regions have a greater influence in loss computation; employing targeted sampling to ensure that batches always included images

with tumor tissue; or monitoring segmentation quality with class-specific metrics such as the Dice coefficient, beyond global pixel accuracy. After testing, it was finally decided to select the most informative images for the pretraining experiments. Specifically, slices were filtered to retain only those with at least two different tumor classes. This procedure reduced the total number of images but improved the richness of the dataset by discarding trivial cases dominated by the background or with only one small tumor region. After filtering, more than 75% of the images in each split contained the three main tumor classes (necrosis, edema, and enhancing tumor), ensuring a more stable and representative training process.

Finally, the data was organized into a folder structure analogous to that of our dataset, with directories for images, masks, and partition files in training, validation, and test. Thanks to this homogenization, the training pipeline could be applied uniformly to both datasets, thereby facilitating comparability between approaches.

3.3.3 Pretraining with BraTS2020

Once the preprocessing of the BraTS2020 dataset was complete, the pretraining phase of the segmentation models was carried out on this dataset. The main objective of this stage was to allow the networks to learn general representations of anatomical structures and tumor patterns specific of MRI, in order to subsequently improve their performance when applying finetuning to our phantom dataset.

Given that BraTS2020 presents particular characteristics, such as a large amount of data, a strong class imbalance, and a multiclass encoding specific to the tumor domain, a different hyperparameter configuration was chosen compared to that used for training in our dataset. Specifically, a larger batch size (16) was employed to better exploit the computational resources of the available GPU, and a higher initial learning rate (1×10^{-3}) was set to promote faster convergence during the early stages of training. In addition, the number of epochs was limited to 10, considering that the purpose of pretraining was to provide an informed initialization rather than achieve full convergence. The loss function used was class-weighted Cross-Entropy Loss, adjusted according to the actual distribution of BraTS2020, in order to compensate for class imbalance. The optimizer selected was AdamW, known for its improved regularization in computer vision tasks, and a CosineAnnealingLR learning rate scheduler was employed to gradually adjust the learning rate across epochs. Finally, a WeightedRandomSampler was incorporated to prioritized the inclusion of images with tumor regions in each batch, thus ensuring balanced coverage during training.

It is important to note that pretraining was performed independently for each architecture (U-Net, DeepLabV3, and FCN), replicating their respective training styles and adapting the internal structure of each network. This is necessary because the weights learned in one architecture are not directly transferable to another due to their profound structural differences.

At the end of each training session, the model that achieved the lowest validation loss was selected as the best model, storing its weights as a reference checkpoint. This strategy made it possible to obtain models with initial weights adapted to the brain MRI domain, which could later be used to assess the impact of informed initialization on the final performance of the models, in comparison to training from scratch.

3.3.4 Finetuning on our dataset

Once pretraining with BraTS2020 was complete, the finetuning stage of the models was carried out on the phantom dataset. The objective of this phase was to finely adjust the pretrained weights to the specific domain of the new dataset, characterized by a different number of classes and a more diverse anatomical structure. To ensure a fair comparison with the models trained from scratch, each architecture was finetuned using its own optimized hyperparameter configuration from Approach 1. These architecture-specific settings are summarized in Table 2.

In addition, for all models, the output layer was modified to match the new number of classes (6), corresponding to the categories defined in our dataset (background, tumor, white matter, grey matter, vessels, and markers). The remaining model weights were initialized with the values obtained in the pretraining, thus allowing training to start from representations already optimized on MRI. During training and evaluation, in order to ensure direct comparability between the two approaches, the same metrics defined in approach 1 were employed: pixel accuracy, Intersection over Union (IoU), and Dice.

In conclusion, this approach provided the three architectures with more informed representations, whose impact on model performance will be assessed in the following chapter.

3.4 Approach 3: Data Augmentation Strategies

The third experimental approach explored the effect of data augmentation strategies as a means to enhance the generalization capacity of segmentation models. Given the limited size of the dataset and the high-class imbalance, particularly for clinically relevant structures such as the tumor, the aim was to artificially increase the variability and representativeness of the training set by introducing synthetic variations of the original data. This augmentation strategy was applied in two complementary ways:

3.4.1 Real-Time General Augmentation

First, a real-time augmentation scheme was implemented across the entire training set. This consisted of applying random transformations in real time during each training epoch. The augmentations were implemented using the Albumentations library, and its scheme was common to the three architectures evaluated (U-Net, DeepLabV3 and FCN). This strategy included both geometric and photometric modifications designed to simulate plausible variations in MRI acquisition and improve model robustness. Specifically, the applied geometric transformations included random affine transformations (scaling, rotation, translation), elastic deformations, flips (horizontal and vertical), and non-linear grid distortions. Moreover, the photometric augmentations consisted in variations in gamma, brightness, and contrast, as well as Gaussian noise, blur, and coarse dropout. These augmentations were applied probabilistically and combined in different compositions per sample, preserving the anatomical plausibility of the images while significantly increasing data variability. This dynamic augmentation strategy enabled the dataset to be enriched without the need to store additional images, with the aim of promoting better model generalization and potentially acting as a regularizer against overfitting.

3.4.2 Class-Focused Offline Augmentation

In parallel, a second strategy was developed to specifically address the underrepresentation of the tumor class, the most clinically important structure in the dataset. This method was based on offline augmentation, generating augmented images before training and saving them to disk as additional training examples. Unlike real-time augmentation, this technique enabled more control over which samples were generated and allowed targeted reinforcement of particular classes. To implement this strategy, only training images that contained tumor pixels (class 3) were selected. For each selected image, firstly, a focus crop was performed. In it a fix-size square region was made centered around a random tumor pixel, applying local zoom and preserving the original image resolution (512×512) by subsequent resizing. Then, advanced transformations, more aggressive and varied than those in the gerenal scheme, were applied to theses crops. The augmentation pipeline included *ShiftScaleRotate*, *ElasticTransform*, *GridDistortion* y *RandomRotate90*, as well as brightness, contrast, gamma, and local noise adjustments. This procedure was performed five times per image, generating a new set of synthetic images with an intensified representation of the tumor class. These augmented images were stored in a separate dataset folder along with their corresponding masks and integrated into the training split alongside the original data. The validation and test sets remained unchanged to ensure an unbiased evaluation of the augmentation impact.

3.4.3 Integration into the Training Pipeline

All three segmentation architectures (U-Net, DeepLabV3, and FCN) were trained using this augmented dataset. For consistency and comparability, each architecture reused the optimized hyperparameters identified in Approach 1, including learning rate, batch size, and stopping criteria (Table 2). The model code was adapted to load the new dataset structure, which included both original and offline-augmented images. During training, models benefited from both augmentation sources: the real-time augmentations added stochastic variability at each epoch, while the offline augmentations increased the absolute presence of tumor structures in the dataset. This dual strategy aimed to improve the model's ability to segment underrepresented structures without compromising the learning of dominant classes such as white and grey matter. The effectiveness of this approach is evaluated in Chapter 4, where both global and per-class results are analyzed in comparison to the baseline training from scratch.

Once each experimental stage has been described, Figure 8 provides a detailed representation of the complete workflow, including the specific procedures applied in each approach.

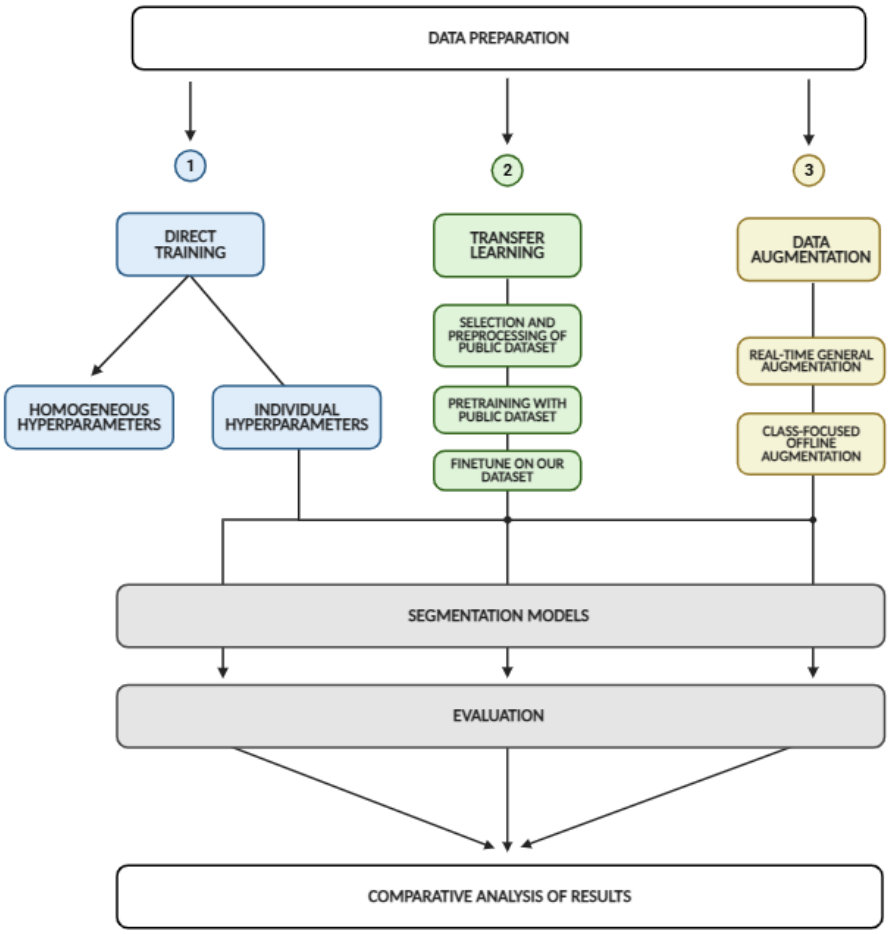

Figure 8. Detailed workflow of the experimental methodology, showing the specific steps within each approach

4 Results and Discussion

4.1 Approach 1

4.1.1 Initial comparison with homogeneous hyperparameters

As a starting point, a comparative evaluation was conducted among the three selected architectures (U-Net, DeepLabV3, and FCN) using a homogeneous hyperparameter configuration. The objective of this first approach was to ensure a fair comparison across models, thus isolating the effect of the different architectures without introducing biases derived model-specific training settings. Table 1 reports the quantitative results obtained after training each model under the same conditions. These conditions were described in detail Section 3.2.5 of the methodology and are also summarized in Table 2. The metrics included are loss, accuracy, IoU, and Dice for training, validation and test. In addition, Figure 8 shows the evolution curves of accuracy, loss, and Dice score over the epochs for each architecture.

Table 1. Comparative results of U-Net, DeepLabV3 and FCN with homogeneous hyperparameters.

Architecture	Best Epoch	LOSS		TRAIN			VALIDATION			TEST		
		Train Loss	Validation Loss	Train Accuracy	Train IoU	Train Dice	Validation Accuracy	Validation IoU	Validation Dice	Test Accuracy	Test IoU	Test Dice
U-NET	46	0.13	0.12	0.97	0.59	0.63	0.97	0.60	0.63	0.93	0.45	0.53
DeepLabV3	19	0.05	0.12	0.99	0.7	0.74	0.97	0.65	0.71	0.94	0.56	0.66
FCN	13	0.06	0.093	0.98	0.69	0.75	0.97	0.63	0.69	0.93	0.52	0.63

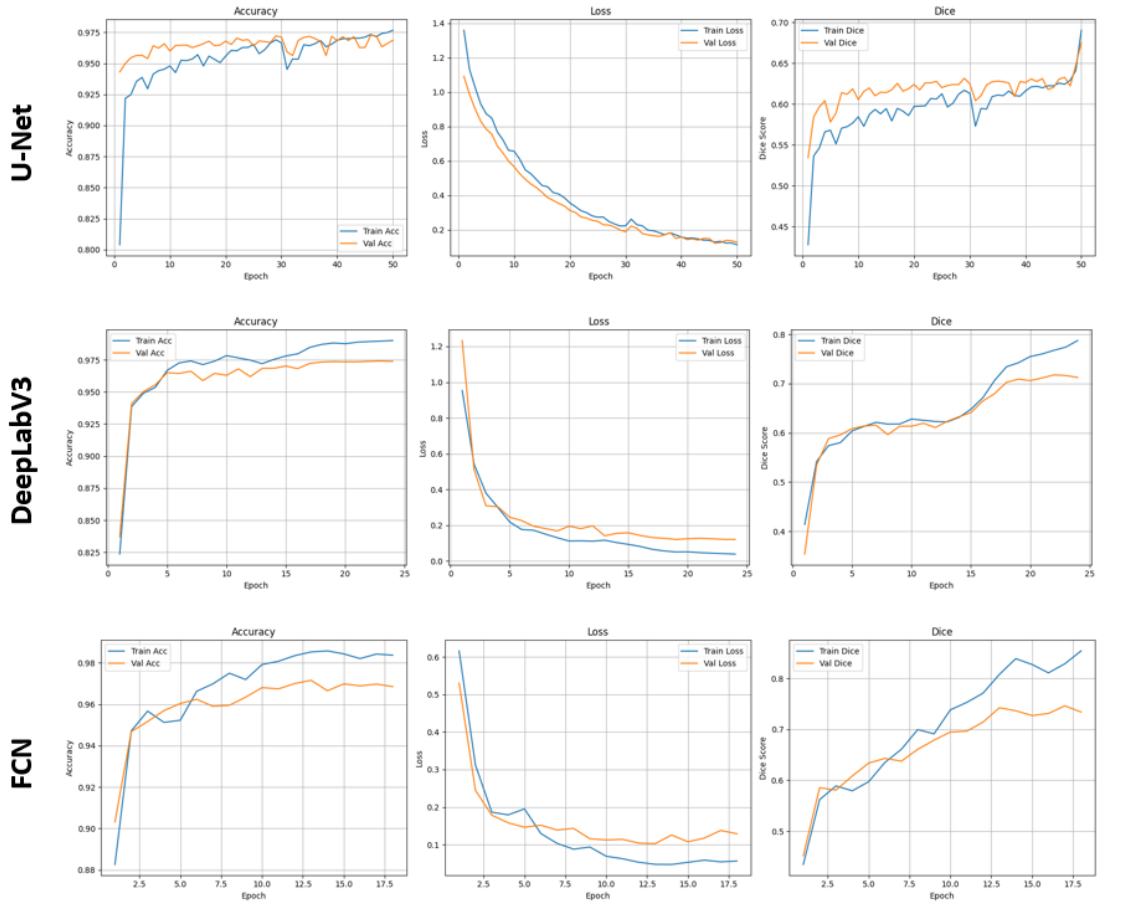

Figure 9. Training and validation curves (accuracy, loss, Dice) per architecture after using homogeneous hyperparameters.

Based on the results in Table 1, the following preliminary conclusions can be drawn. The best overall performance was achieved by DeepLabV3, with superior metrics across all dataset splits. It particularly stands out on the test set with Dice = 0.66 and IoU = 0.56, suggesting a higher learning capacity within the proposed framework. Likewise, FCN model shows a similar behaviour, reaching Dice and IoU values only slightly lower (0.63 and 0.52, respectively). Finally, U-Net is the lowest performing acrros all splits, with test values of Dice = 0.53 and IoU = 0.45, i.e., more than 0.1 below the results of DeepLabV3. These observations are reinforced by the visual analysis of the curves. In Figure 9, we can see how both FCN and DeepLabV3, to a lesser extent, show a clear separation between training and validation in the last epochs, a clear sign of overfitting. This is further supported by the metrics in Table 1, where both models show a large gap between train and validation Dice (0.74/0.71 and 0.75/0.69). In contrast, U-Net exhibits the most stable loss evolution with no signs of overfitting, since the training and validation curves remain close, and the Dice value is equal in the train and validation sets. Furthermore, in the case of U-Net, the loss continues to decrease at the end of training, suggesting the model may not have reached a clear convergence within the assigned number of epochs. These results are consistent with the state-of-the-art descriptions for each model (section 2.6): DeepLabV3 and FCN tend to yield better results due to their higher architectural complexity, but in exchange are more prone to overfitting on small datasets like ours.

4.1.2 Architecture specific adjustments

After analyzing the results obtained with a homogeneous hyperparameter setup, several architecture-specific weaknesses were identified. Consequently, individualized adjustments were made with the aim of maximizing the performance of each network. Each of these changes will be applied only to its corresponding network and are summarized in Table 2.

Table 2. Hyperparameter configurations used in Approach 1: homogeneous setup vs. architecture-specific adjustments.

Hyperparameter	Homogeneous configuration	U-Net	DeepLabV3	FCN
Loss function	CrossEntropyLoss	CrossEntropy + Dice	CrossEntropy + Dice	CrossEntropy + Dice
Optimizer	Adam	Adam	AdamW	AdamW
Learning rate	1e-4	1e-4	3e-5	5e-5
Weight decay	None	None	3e-4	1e-4
Max epochs	50	100	50	50
Early stopping	Patience = 5 (val loss)	Patience = 5 (val loss)	Patience = 5 (val loss)	Patience = 5 (val loss)
Batch size	4	4	4	4

For U-Net, the loss and metric curves indicated that the model had not yet reached clear convergence under the homogeneous configuration. The loss continued to decrease at the end of training and the validation metrics remained stable, indicating that the model could benefit from more epochs. The maximum

number of epochs was therefore increased to enabled full convergence. In addition, the loss function was modified to directly incorporate the Dice metric by combining it with Cross-Entropy, aligning the optimization process with the main evaluation metric. These changes led to a very substantial performance improvement: test Dice increased from 0.53 to 0.70, and IoU from 0.45 to 0.57. Validation also improved significantly, maintaining good curve stability without severe symptoms of overfitting. This confirms that U-Net required longer training and a loss function better aligned with the final objective.

DeepLabV3 already offered good results under the homogeneous configuration (Test Dice = 0.66), but it showed a clear tendency toward overfitting, with a significant gap between training and validation. To mitigate this, the Adam optimizer was replaced with AdamW, including L2 regularization with a weight_decay of 3e-4. In addition, the learning rate was slightly reduced to 3e-5. These changes aim to slow down the model's excessive specialization to the training set and stabilize validation. The loss function was also modified to include the Dice component, seeking better alignment with the target metric. The result was very positive. The outcome was very positive: test Dice increased to 0.73, and IoU reached 0.61, which was the best performance among the three architectures. Although the gap between training and validation remained (Train Dice = 0.92, Val Dice = 0.78), overall performance improved. Additional attempts to introduce further regularization through dropout were tried to reduce this gap, but they were detrimental for performance and therefore discarded.

For the FCN architecture, a strategy similar to DeepLabV3 was applied, using AdamW with weight_decay = 1e-4 and a learning rate of 5e-5. The loss function was also modified to include the Dice component. This adjustment delivered a considerable improvement over the baseline configuration, achieving a Test Dice of 0.72 (versus the original 0.63). Although the train–validation gap remained, the model was able to generalize better to the test set. Subsequent attempts to introduce additional dropout did not offer benefits and were likewise discarded.

Table 3. Comparative results of U-Net, DeepLabV3 and FCN with customized hyperparameters.

Architecture	Best Epoch	LOSS		TRAIN			VALIDATION			TEST		
		Train Loss	Validation Loss	Train Accuracy	Train IoU	Train Dice	Validation Accuracy	Validation IoU	Validation Dice	Test Accuracy	Test IoU	Test Dice
U-NET	75	0.32	0.52	0.98	0.80	0.88	0.98	0.71	0.78	0.93	0.57	0.70
DeepLabV3	45	0.33	0.62	0.99	0.86	0.92	0.97	0.70	0.78	0.94	0.61	0.73
FCN	34	0.17	0.55	0.99	0.92	0.95	0.97	0.71	0.80	0.94	0.60	0.72

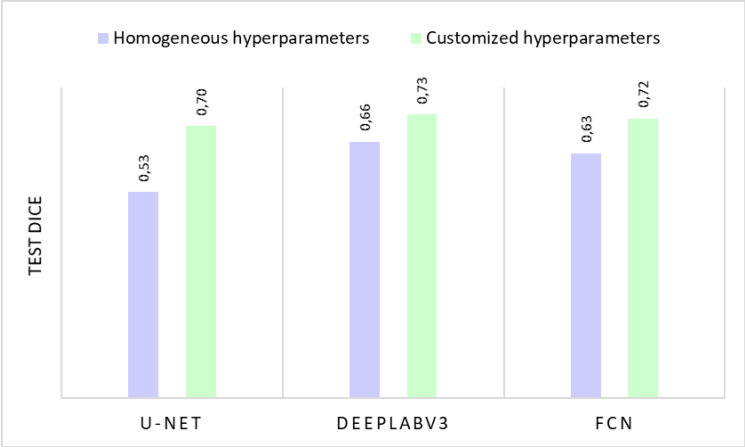

Figure 10. Effect of hyperparameter configuration (homogeneous vs. customized) on the performance of U-Net, DeepLabV3+, and FCN.

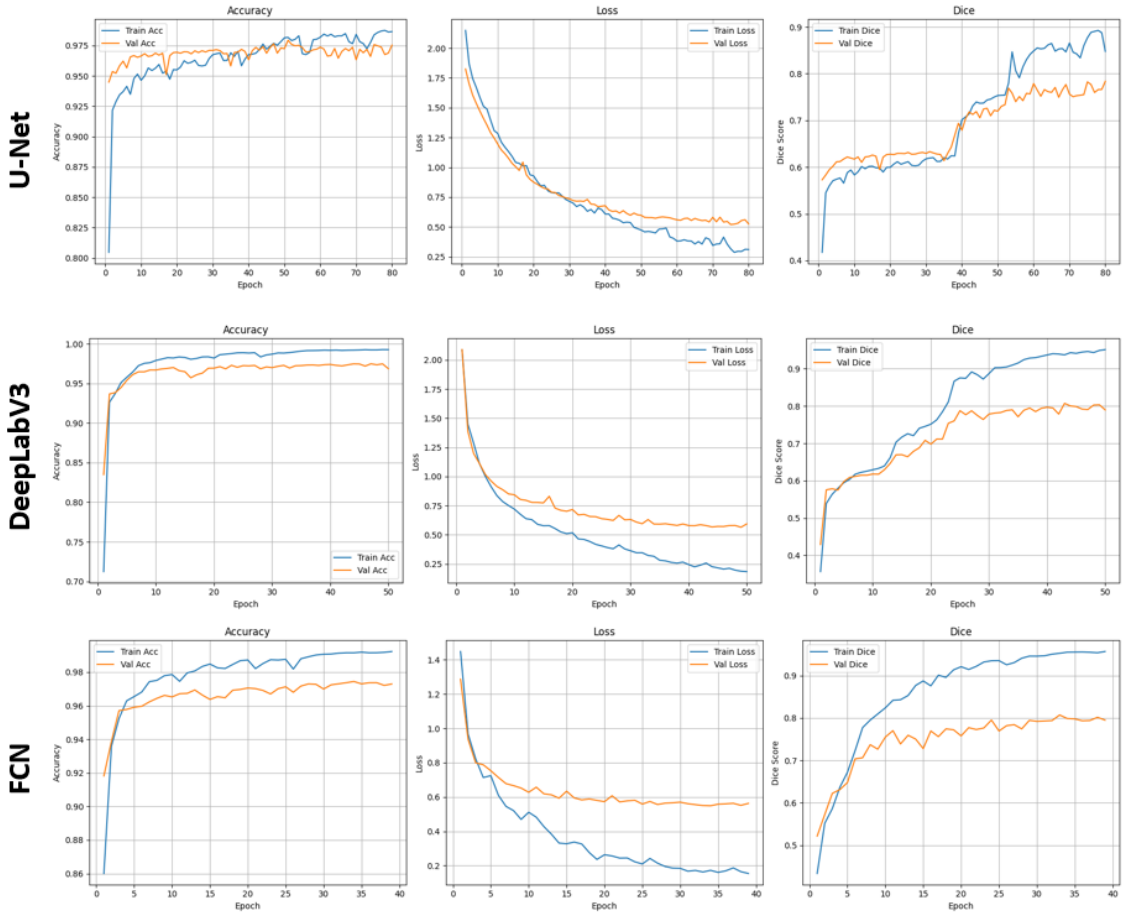

Figure 11. Training and validation curves (accuracy, loss, Dice) per architecture after using customized hyperparameters.

It should be noted that, given the limited dataset size and the high complexity of architectures such as DeepLabV3 and FCN, it is difficult to completely avoid overfitting. In the following approaches, new strategies, such as pretraining and dataset expansion, will be explored to determine whether they can mitigate this limitation. In summary, DeepLabV3 achieved the best overall performance (Test Dice = 0.73), closely followed by FCN (0.72). These results show that, while a homogeneous configuration enables architectural comparisons under equal conditions, model-specific tuning is necessary to fully exploit each model's potential.

Computational cost of each architecture

To complement the analysis of model performance, their computational cost during inference was also evaluated. Specifically, five key metrics were computed using inputs of size 3×512×512 pixels and a batch size of 4 images:

- **Parameters**: total number of trainable weights in the model, measured in millions (M). It reflects the model's learning capacity. More parameters tipically imply larger memory is needed to store the model and, often, greater accuracy (though also a higher risk of overfitting).
- **FLOPs**: estimated number of floating-point operations required to process a single image (in GigaFLOPs, G). Higher values demand more computing power, which can affect speed.
- **Latency (ms/img)**: average time the model takes to segment a single image, useful for clinical environments where response time is critical.

- **Throughput (img/s)**: average processing speed, expressed in images per second.
- **VRAM consumption (MB)**: maximum GPU memory used during inference, relevant in contexts with limited resources.

Table 4. Computational cost comparison of the implemented segmentation models.

Architecture	Parameters (M)	FLOPs (G)	Latency (ms/img)	Throughput (img/s)	VRAM peak (MB)
UNet	31.39	223.71	6.73	148.61	4334
DeepLabV3	39.64	163.86	3.34	299.44	924
FCN	32.95	138.61	2.88	346.94	1072

Table 4 summarizes the results. These show that FCN is the most computationally efficient model, with the lowest latency (2.88 ms per image) and the highest throughput (346.94 images/second), making it the fastest model. It is followed by DeepLabV3, which has the largest number of parameters in the model (39.69M) and therefore the greatest learning capacity, as well as the lowest VRAM consumption (924 MB). By contrast, U-Net is clearly the most computationally expensive: it has the highest number of operations (223.71 GFLOPs), the lowest throughput (148.61 images/second, half that of FCN), and a significantly higher memory consumption (4334 MB), which may limit its applicability in environments with more restricted hardware. This higher computational cost may be a consequence from its more symmetric and heavy structure, in addition to the intensive use of skip connections. This analysis allows us to assess not only model's accuracy, but also their deployment viability in different clinical or industrial scenarios where computational resources, response time, and available memory are decisive factors.

Furthermore, these findings are consistent with previous observations regarding model accuracy. DeepLabV3 achieved the best segmentation metrics, followed by FCN. These architectures have the highest number of parameters and, consequently, the greatest learning capacity. Meanwhile, U-Net required more training epochs to converge properly, which may be related to its higher number of operations. However, both DeepLabV3 and FCN showed a greater tendency to overfit, which is also reflected in their lower number of parameters, suggesting less effective internal complexity compared to the heavier architecture of U-Net. This relationship between computational complexity and training behaviour reinforces the idea that model selection must consider both the available resources and the dataset characteristics.

4.2 Approach 2

In this second approach, we explored the possibility of improving the performance of the three architectures through pretraining on the BraTS2020 dataset. The goal was for the models to learn general representations of brain structures from an external dataset, and then be finetuned to the specific domain of our phantom dataset. Since Approach 1 identified specific optimal configurations for each architecture, those individualized hyperparameters were also used here. This allows for a fairer comparison of the effect of pretraining,

ensuring that each network starts from its best baseline configuration. Figure 12 illustrates the comparison of Dice scores obtained by each architecture when using pretraining on BraTS2020 (Approach 2) versus when trained from scratch (Approach 1).

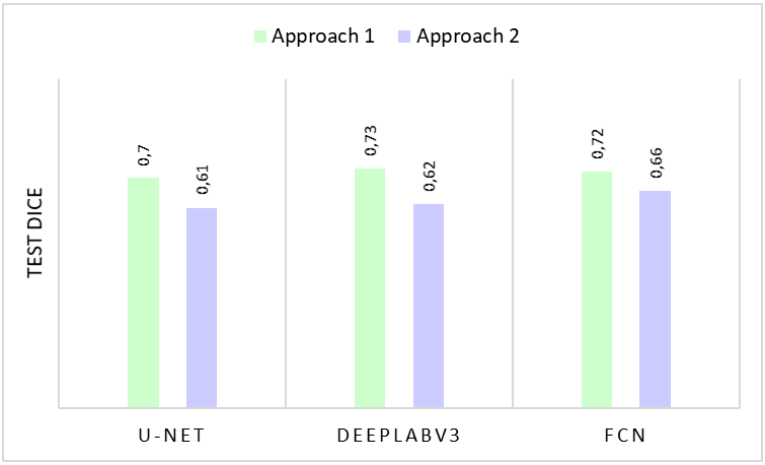

Figure 12. Comparison of Dice scores between Approach 1 (training from scratch) and Approach 2 (with pretraining) across the three architectures.

At first glance, in all three cases the test performance worsens notably after applying pretraining. For U-Net, test Dice drops from 0.70 (without pretraining) to 0.61 (with pretraining), while for DeepLabV3 it goes from 0.73 to 0.62, and for FCN it decreases from 0.72 to 0.66. This indicates that, despite starting from a pretrained base, the models have not been able to adapt effectively to the new multiclass anatomical segmentation task.

Additionally, Figure 13 presents the corresponding training curves for each model, showing the evolution of accuracy, loss, and Dice on the training and validation subsets. Paradoxically, the curves in Approach 2 show more stable and less noisy behaviour. Starting from pretrained weights leads to smoother parameter updates and fewer oscillations. However, this stability does not imply that the model is learning better, rather, it suggests that the model is more constrained by the prior knowledge acquired during pretraining.

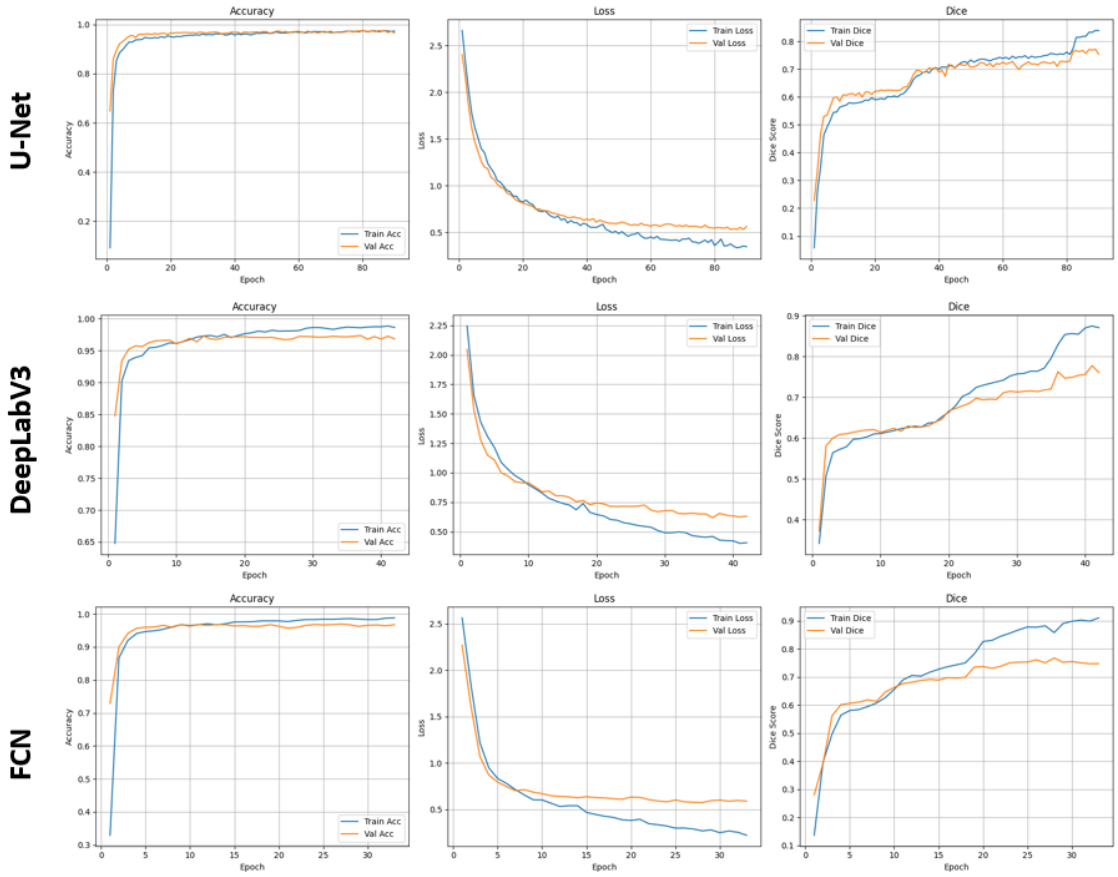

Figure 13. Training and validation curves (accuracy, loss, Dice) per architecture in the second approach.

On the other hand, in this approach we also observe a lower propensity for overfitting. This can be explained by the fact that pretraining has acted as a form of implicit regularization, mitigating the impact of the small dataset size on more complex models such as DeepLabV3 or FCN. Although pretraining has visually reduced overfitting in the three architectures (smaller separation between training and validation metrics), it did not translate into better generalization, as test results were systematically worse. This outcome is not necessarily due to domain mismatch between the BraTS dataset and the phantom dataset, since transfer learning has often proven effective even across very different domains. More likely, the lack of improvement reflects a combination of factors such as the limited size of the target dataset, suboptimal pretraining configuration, or insufficient fine-tuning capacity to fully adapt the pretrained weights to the new task. In practice, training from scratch may have allowed the models to converge more effectively under these constraints.

In summary, although pretraining helped reduce the overfitting observed in the previous approach, it did not improve overall performance. In all cases, training from scratch turned out to be more beneficial, highlighting the importance of both dataset size and training configuration when applying transfer learning in multiclass medical segmentation contexts.

4.3 Approach 3

In this third approach, advanced data augmentation techniques were applied with the goal of increasing the variability of the training set to improve the segmentation performance. First, a general real-time augmentation scheme was applied to the entire training set to enhance model generalization. Additionally,

we sought to specifically improve performance on the “Tumor” class, due to its higher clinically relevance and underrepresentation in the dataset. To this end, an offline augmentation strategy was generated, targeting exclusively images containing the “tumor” class. In this approach, all models were trained from scratch, i.e., without using pretrained weights (unlike Approach 2). Moreover, each architecture reused the hyperparameters that achieved the best performance in Approach 1. This allowed us to evaluate the exclusive impact of the augmentation strategies by comparing against the results from the first approach.

Global results

Figure 14 shows the overall test performance of each architecture in Approach 3, directly compared with the results from Approach 1. When analyzing how model performance changes after applying the augmentation strategies, we observe that segmentation metrics remain very similar across the two approaches. For U-Net, there is a slight improvement in the IoU value (+0.01) and a small decrease (-0.01) in Dice, with accuracy remaining unchanged. For DeepLabV3, the values of the three metrics remained exactly the same, indicating that the data augmentation strategies did not alter the overall performance. Finally, FCN shows a slight decrease in both IoU and Dice (-0.01 in both), with accuracy remaining constant.

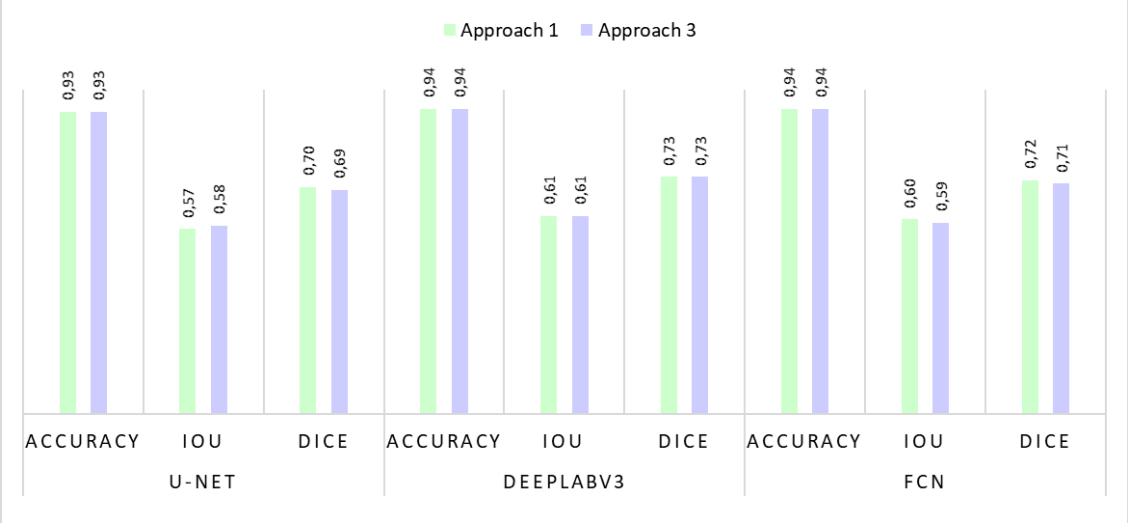

Figure 14. Comparison of test metrics (Accuracy, IoU, Dice) between Approach 1 and Approach 3 across the three architectures.

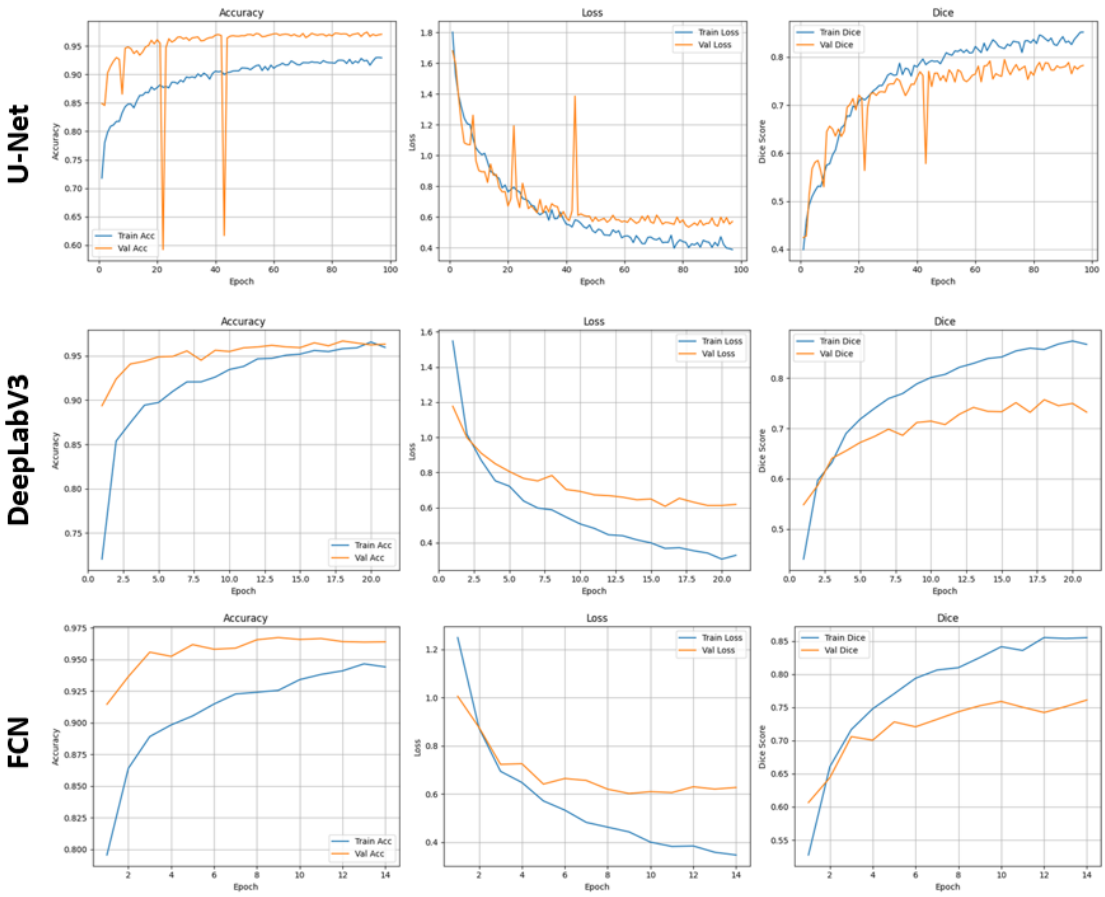

Figure 15. Training and validation curves (accuracy, loss, Dice) per architecture in the third approach.

Looking at the learning curves obtained in this third approach (Figure 15), training dynamics appear somewhat less stable than in Approach 1. In the case of U-Net, the validation metrics show abrupt oscillations, possibly indicating a certain sensitivity to the new images introduced through augmentation techniques. However, the overall trends remain positive, with a progressive improvement in training metrics and acceptable stabilization in validation. For DeepLabV3 and FCN, behaviour is more stable, and fewer epochs are needed to reach the peak performance (as evidenced by earlier early stopping in both cases). This can be interpreted as a sign of greater learning efficiency thanks to the increased data diversity. In addition, there is a reduction in the overfitting previously seen in these two models in Approach 1, as the gap between training and validation metrics is smaller. In general, there curves show smooth progression and clear convergence, very similar to those observed in Approach 1, suggesting that these architectures successfully adapted to the increased data without compromising training stability.

In summary, the augmentation strategy applied in this approach has not produced significant improvements in the overall metrics of the test set compared to training from scratch without augmentation (Approach 1). However, it maintained competitive performance and even matched prior best results in some cases, such as in DeepLabV3. Additionally, the inclusion of these techniques had a clear regularizing effect, reducing the overfitting previously observed in DeepLabV3 and FCN. This resulted in more balanced learning, preventing the models from excessively memorizing the training set and producing closer training and validation curves. It also reduced the number of

epochs required to reach the best performance, implying greater efficiency without compromising training stability. Although the benefits are not strongly reflected in the overall metrics, the increase diversity of examples has enriched the training set without degrading performance, which may be valuable from a generalization perspective. A detailed per-class analysis is needed to determine whether this strategy achieved its main objective: improving segmentation of minority classes, especially the tumor.

4.4 Per-class comparison

The dataset's original class distribution shows a marked imbalance, with a strong predominance of background (82.74%) compared to minority classes such as tumor (1.61%), vessels (0.04%), and markers (0.28%). This disproportion represents a significant challenge for segmentation architectures, which tend to favor majority classes during learning, making it difficult to correctly segment structures of clinical interest such as tumors. Table 5 shows the percentage distribution of pixels per class before and after applying data augmentation strategies.

Table 5. Change in the percentage of pixels per class in the dataset images after data augmentation.

Approach	Background	White matter	Grey matter	Tumor	Vessels	Markers
Approach 1	82.74%	6.43%	8.91%	1.61%	0.04%	0.28%
Approach 3	68.78%	10.38%	15.10%	5.18%	0.05%	0.52%

After applying these strategies, a more balanced distribution was achieved. In particular, the proportion of pixels corresponding to the tumor increased substantially, from 1.61% to 5.18%. The presence of the WM (6.43% → 10.38%) and GM (8.91% → 15.10%) also increased, expanding the representation of key brain structures in the training set. By contrast, the most underrepresented classes (vessels and markers) remained below 1% despite the increase, due to their scarce presence in the original images. This redistribution exposed the model to a larger number of relevant examples, improving its ability to learn their distinctive features. Even so, the natural predominance of background was not completely eliminated and remains the majority. Overall, these changes, especially the substantial increase in the tumor class, justify the combined use of targeted offline augmentation and general real-time augmentation to mitigate bias toward the majority classes and strengthen learning of minority classes. Table 6 shows the performance per segmented class for each architecture for Approach 1 (without augmentation) and Approach 3 (with augmentation strategies). Values that improve over the previous approach are highlighted in green, those that worsen are shown in red, while unchanged values are highlighted in yellow.

Table 6. Per-class Dice scores across the evaluated architectures in Approach 1 and 3.

Architecture	Approach	Background	White matter	Grey matter	Tumor	Vessels	Markers
U-Net	Approach 1	0.969	0.523	0.480	0.335	0.042	0.316
	Approach 3	0.968	0.620	0.520	0.355	0.010	0.331
DeepLabV3	Approach 1	0.971	0.578	0.533	0.345	0.0	0.345
	Approach 3	0.970	0.583	0.552	0.376	0.039	0.353
FCN	Approach 1	0.969	0.555	0.506	0.345	0.021	0.345
	Approach 3	0,969	0.568	0.516	0.367	0.025	0.334

- **Background**: A slight decrease in performance is observed in both U-Net and DeepLabV3, while FCN remains stable. This could be because augmentation introduced more images centered on regions with relevant structures (tumor, WM, GM, etc.), reducing the relative proportion of background pixels. With less background present, the model's background segmentation accuracy declines slightly.
- **White matter (WM)**: A clear improvement is observed in all three models, especially in U-Net, where Dice increases from 0.52 to 0.62. This suggests that the variability generated by the transformations helped the model to better learn the boundaries and shapes of this structure.
- **Gray matter (GM)**: Also shows a consistent improvement across all architectures. Although not as minority as other classes, GM tends to have diffuse boundaries with surrounding tissue. The added variability may have helped its delineation.
- **Tumor**: Dice values have improved slightly in all three models (U-Net: 0.335 → 0.355, DeepLab: 0.345 → 0.376, FCN: 0.345 → 0.367 This indicates that both the real-time and offline tumor-focused strategies were helpful. However, the improvement is not as pronounced as might be expected, suggesting that even with targeted techniques, the poor representation of tumor pixels in the dataset limits effective learning.
- **Vessels**: This is the class with the worst overall performance and the one that has benefited least from data augmentation. In the case of U-Net, its segmentation has worsened, while in FCN it has improved minimally. Notably, in Approach 1 DeepLabV3 failed to segment vessels (null values), while in Approach 3 it obtained a small Dice value, reflecting some learning thanks to the increased examples. In general, the very limited presence of vessels makes effective segmentation difficult.
- **Markers**: Slight improvement in U-Net and DeepLabV3, while FCN shows a small decrease. Since markers are small and infrequent structures, performance differences may be due to factors such as the number of examples present after augmentation and the model's sensitivity to detecting rare patterns.

Overall, the comparison by class reveals that data augmentation strategies have been more effective in improving the segmentation of moderately represented structures such as white and grey matter, while their impact on very minority classes such as vessels, tumors, and markers has been limited. This reinforces

the idea that, although data augmentation can enrich the training set, there are limits when the representation of certain classes is extremely low.

4.5 Qualitative visual analysis

In this section, a qualitative visual analysis of representative MRI slices is presented, comparing the segmentation masks produced by U-Net, DeepLabV3, and FCN across different experimental approaches against the ground truth. The aim is to complement the quantitative evaluation with visual evidence that highlights strengths, weaknesses, and class-specific behaviours of each architecture.

Example 1 – Visual comparison on image P3_axial_19

Focusing on the results from Approach 1, DeepLab and FCN segment the tumor (red) and white matter (WM) (blue) more accurately, producing sharper boundaries closer to the ground truth. In contrast, U-Net presents a smaller tumor region and more diffuse contours. This is consistent with the quantitative results by class, where U-Net performed worst on these categories. In addition, this image shows the detection of the marker class (green). While DeepLab and FCN correctly locate this class in the corresponding region, U-Net predicts multiple additional areas with this label. This suggests possible confusion with artifacts or noise present in the original image, especially at the periphery of the structure. This also agrees with the results analyzed by class, where DeepLab and FCN outperformed U-Net for the marker class. In Approach 3, after applying data augmentation, a general visual improvement is observed across all models: the masks are cleaner, the edges are more precise, and the overall segmentation is closer to the ground truth. This reinforces the increase observed in the Dice metric for all architectures. However, a specific error appears in FCN, which introduces an incorrect marker region, consistent with the slight drop in that class performance after augmentation.

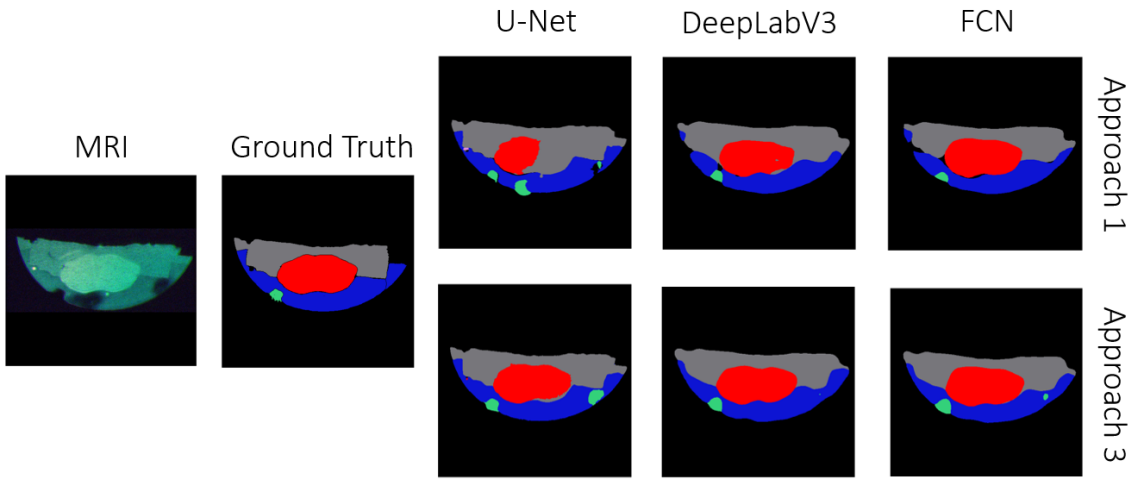

Figure 16. Qualitative visual comparison on sample P3_axial_19: MRI image, ground truth, and predictions from U-Net, DeepLabV3, and FCN for Approaches 1 and 3.

Example 2 – Visual comparison on image P3_axial_29

In Approach 1, clear differences emerge in the segmentation of two key classes. On the one hand, for white matter (WM) (blue) U-Net shows poor segmentation, with diffuse borders and visible mixing with grey matter (GM). However, DeepLab and FCN present more precise contours and segmentations closer to the ground truth. This is consistent with the quantitative results obtained,

where U-Net performed the worst in both classes (WM and GM). On the other hand, the behaviour for blood vessel (pink) is particularly revealing. In the results by class, U-Net was the model that best detected them, followed by FCN, while DeepLab was practically unable to segment them. This trend is reflected in the image: U-Net predicts both vessel regions quite accurately, FCN detects only one of them, and DeepLab predicts neither. When analyzing Approach 3, notable changes appear: DeepLabV3 manages to segment both vessel regions for the first time, consistent with the observed increase in its Dice for this class. In contrast, U-Net, initially the best in this category, now shows very slight detection, which also matches its performance drop for this class after augmentation.

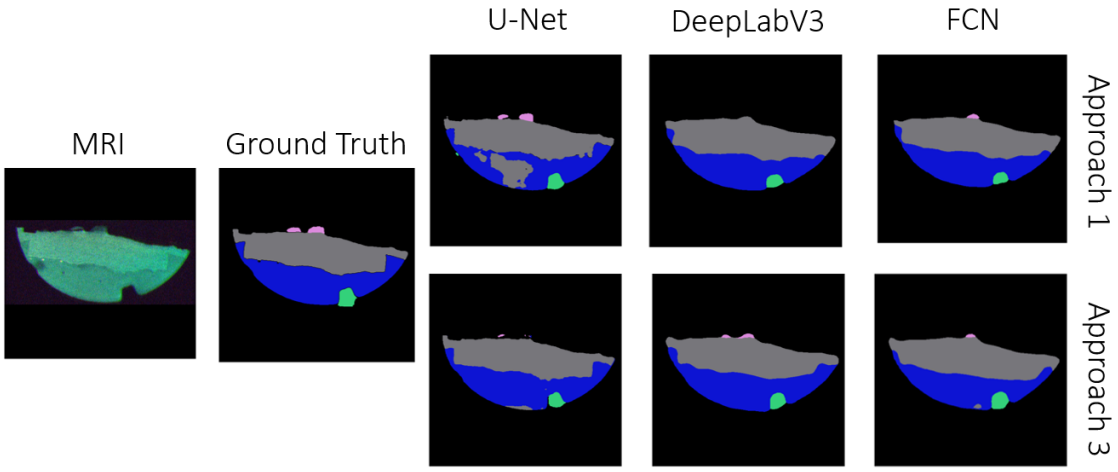

Figure 17. Qualitative visual comparison on sample P3_axial_29: MRI image, ground truth, and predictions from U-Net, DeepLabV3, and FCN for Approaches 1 and 3.

In conclusion, the qualitative visual analysis conducted on selected images reinforces and contextualizes the quantitative results obtained earlier. The differences observed among architectures and approaches show clear coherence with the per-class metrics and confirm several previously identified patterns. While U-Net tends to produce softer boundaries and lower accuracy, DeepLabV3 and FCN generate cleaner and more precise masks, showing better overall performance in structural classes. Finally, Approach 3 shows a visual improvement across all models, demonstrating the positive effect of increased synthetic data on segmentation quality, nevertheless with specific variations by class and architecture. Taken together, these visual observations provide qualitative validation of the metric analysis and allow for an intuitive illustration of the strengths and limitations of each model in complex clinical contexts such as the present one.

5 Conclusions

This work has addressed the complex task of multiclass brain MRI segmentation, a problem particularly challenging due to general limitations, such as the scarcity of annotated datasets, the strong imbalance between classes, and the variability introduced by different acquisition protocols. These issues hinder model training and reduce generalization capacity. In this study, these difficulties were further accentuated by the specific characteristics of the available dataset: a reduced number of phantom images, extreme imbalance of minority classes such as tumor and vessels (≈1% of the pixels), and the morphological heterogeneity of brain structures. These constraints posed significant obstacles for achieving robust training and reliable predictions. To address them, a set of specific objectives was defined: implementing and training different architectures, quantitatively evaluating their performance, conducting a systematic comparison, and exploring complementary strategies to improve generalization. These objectives were satisfactorily achieved, providing the basis for the analysis of the results presented below.

The comparative analysis carried out in this work has enabled a systematic evaluation of three reference deep learning architectures for multiclass brain MRI segmentation (U-Net, DeepLabV3, and FCN), under different experimental conditions that included direct training, transfer learning, and data augmentation strategies. Considering the set of metrics (Dice, IoU, pixel accuracy), the training curves, the per-class analysis, and the computational cost, DeepLabV3 emerges as the most balanced and robust model. It consistently achieved the best performance metrics in the test set (Dice = 0.73 after hyperparameter tuning), with stable performance across major anatomical classes such as tumor, white matter, and grey matter. Although FCN achieved very similar results (Dice = 0.72), DeepLabV3 demonstrated a greater capacity to capture complex and diffuse boundaries, which is essential in the medical domain. U-Net, despite its popularity and lower computational requirements, exhibited the lowest performance overall, and required longer training to converge. These findings are consistent with reports in the literature, where DeepLab-based models often outperform U-Net in tasks requiring accurate delineation of structures with complex boundaries [41]. In terms of computational efficiency, FCN was the fastest and least memory demanding, whereas U-Net was clearly the most resource-intensive. Similarly, FCN has repeatedly been validated as a simple yet efficient baseline, although it rarely surpasses more recent architectures in accuracy [42]. Altogether, the results obtained here strengthen the evidence that architecture choice must balance accuracy, robustness, and computational feasibility depending on the specific clinical context. The qualitative comparison of selected cases provided valuable insights complementary to the quantitative evaluation. U-Net tended to produce blurrier contours and unstable segmentations, often over-predicting markers or missing tumor regions. In contrast, DeepLabV3 and FCN generated cleaner and more faithful masks, confirming their superior performance in structural classes. These observations confirm the coherence between the visual inspection and the class-wise metrics. Regarding the complementary strategies tested, transfer learning with BraTS2020 did not lead to measurable performance improvements. In contrast, data augmentation achieved results comparable to direct training while reducing overfitting and alleviating class imbalance, leading to improved segmentation of minority structures.

For future work, several directions could be explored to further improve performance. First, generating additional phantoms in the laboratory would

provide a larger and more diverse dataset for training. Second, transfer learning could be revisited with alternative configurations or combined with data augmentation, so that fine-tuning benefits from a greater number of images and better adapts the pretrained features to the target classes. Finally, more sophisticated augmentation techniques, such as generative adversarial networks (GANs), could be investigated to further enrich data variability and improve the detection of underrepresented structures.

The results demonstrate the feasibility of applying deep learning architectures to multiclass segmentation of brain MRI. A robust model such as DeepLabV3 could support multiple clinical tasks: assisting in tumor detection and delineation, providing quantitative maps for treatment planning, or enabling longitudinal monitoring of disease progression. Automated segmentation has the potential to reduce the workload of specialists, minimize inter-observer variability, and accelerate the integration of AI-assisted tools into clinical workflows. This work highlights the importance of conducting rigorous comparative studies that consider not only performance metrics but also qualitative inspection, computational cost, and class-level analysis. While DeepLabV3 stands out as the best option in this study, the results underline that no architecture is universally optimal: performance depends on the dataset, the classes of interest, and the computational context. Ultimately, the thesis contributes to advancing the understanding of how different deep learning strategies behave in the complex task of multiclass brain MRI segmentation, providing useful insights for both future research and the translation of AI models into clinical practice.

6 Bibliography

[1] M. Martucci *et al.*, 'Magnetic Resonance Imaging of Primary Adult Brain Tumors: State of the Art and Future Perspectives', *Biomedicines*, vol. 11, no. 2, p. 364, Jan. 2023, doi: 10.3390/biomedicines11020364.

[2] H. Sung *et al.*, 'Global Cancer Statistics 2020: GLOBOCAN Estimates of Incidence and Mortality Worldwide for 36 Cancers in 185 Countries', *CA Cancer J Clin*, vol. 71, no. 3, pp. 209–249, May 2021, doi: 10.3322/caac.21660.

[3] M. K. Singh and K. K. Singh, 'A Review of Publicly Available Automatic Brain Segmentation Methodologies, Machine Learning Models, Recent Advancements, and Their Comparison', *Ann Neurosci*, vol. 28, no. 1–2, pp. 82–93, Jan. 2021, doi: 10.1177/0972753121990175.

[4] I. Despotović, B. Goossens, and W. Philips, 'MRI Segmentation of the Human Brain: Challenges, Methods, and Applications', *Comput Math Methods Med*, vol. 2015, p. 450341, 2015, doi: 10.1155/2015/450341.

[5] Z. Teng *et al.*, 'A literature review of artificial intelligence (AI) for medical image segmentation: from AI and explainable AI to trustworthy AI', *Quantitative Imaging in Medicine and Surgery*, vol. 14, no. 12, pp. 9620652–9629652, Dec. 2024, doi: 10.21037/qims-24-723.

[6] F. J. Dorfner, J. B. Patel, J. Kalpathy-Cramer, E. R. Gerstner, and C. P. Bridge, 'A review of deep learning for brain tumor analysis in MRI', *npj Precis. Onc.*, vol. 9, no. 1, p. 2, Jan. 2025, doi: 10.1038/s41698-024-00789-2.

[7] Mohammad Mojtaba Rohani, Mohammad Mojtaba Rohani, and Soheil Durson, 'Deep learning in medical imaging for disease diagnosis', *World J. Adv. Res. Rev.*, vol. 25, no. 2, pp. 2522–2526, Feb. 2025, doi: 10.30574/wjarr.2025.25.2.0558.

[8] P. J. Bazira, 'An overview of the nervous system', *Surgery (Oxford)*, vol. 39, no. 8, pp. 451–462, Aug. 2021, doi: 10.1016/j.mpsur.2021.06.012.

[9] L. Thau, V. Reddy, and P. Singh, 'Anatomy, Central Nervous System', in *StatPearls*, Treasure Island (FL): StatPearls Publishing, 2025. Accessed: Sep. 08, 2025. [Online]. Available: http://www.ncbi.nlm.nih.gov/books/NBK542179/

[10] P. E. Ludwig, V. Reddy, and M. A. Varacallo, 'Neuroanatomy, Central Nervous System (CNS)', in *StatPearls*, Treasure Island (FL): StatPearls Publishing, 2025. Accessed: Sep. 08, 2025. [Online]. Available: http://www.ncbi.nlm.nih.gov/books/NBK442010/

[11] T. Dhollander, 'Accounting for Complex Structure in Diffusion Weighted Imaging Data using Volume Fraction Representations'.

[12] Kavitha Soppari, Sravya Putnala, Sai Srujana Borala, and Sai Kumar Gondhi, 'A survey on brain MRI segmentation', *World J. Adv. Res. Rev.*, vol. 21, no. 3, pp. 1702–1710, Mar. 2024, doi: 10.30574/wjarr.2024.21.3.0813.

[13] A. A. Mahmoud, E.-S. M. El-Rabaie, T. E. Taha, A. Elfishawy, O. Zahran, and F. E. Abd El-Samie, 'Medical Image Segmentation Techniques, a Literature Review, and Some Novel Trends', *Menoufia Journal of Electronic Engineering Research*, vol. 27, no. 2, pp. 23–58, Jul. 2018, doi: 10.21608/mjeer.2018.63179.

[14] X. Liu, L. Song, S. Liu, and Y. Zhang, 'A Review of Deep-Learning-Based Medical Image Segmentation Methods', *Sustainability*, vol. 13, no. 3, p. 1224, Jan. 2021, doi: 10.3390/su13031224.

[15] M. Sezgin and B. Sankur, 'Survey over image thresholding techniques and quantitative performance evaluation', *JEI*, vol. 13, no. 1, pp. 146–165, Jan. 2004, doi: 10.1117/1.1631315.

[16] G. B. Coleman and H. C. Andrews, 'Image segmentation by clustering', *Proceedings of the IEEE*, vol. 67, no. 5, pp. 773–785, May 1979, doi: 10.1109/PROC.1979.11327.
[17] J.-H. Xue, A. Pizurica, W. Philips, E. Kerre, R. Van De Walle, and I. Lemahieu, 'An integrated method of adaptive enhancement for unsupervised segmentation of MRI brain images', *Pattern Recognition Letters*, vol. 24, no. 15, pp. 2549–2560, Nov. 2003, doi: 10.1016/S0167-8655(03)00100-4.
[18] M. Kass, A. Witkin, and D. Terzopoulos, 'Snakes: Active contour models', *Int J Comput Vision*, vol. 1, no. 4, pp. 321–331, Jan. 1988, doi: 10.1007/BF00133570.
[19] T. Węgliński and A. Fabijańska, 'Brain tumor segmentation from MRI data sets using region growing approach', in *Perspective Technologies and Methods in MEMS Design*, May 2011, pp. 185–188. Accessed: Sep. 08, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/5960339
[20] K.-H. Nenning and G. Langs, 'Machine learning in neuroimaging: from research to clinical practice', *Radiologie (Heidelb)*, vol. 62, no. Suppl 1, pp. 1–10, 2022, doi: 10.1007/s00117-022-01051-1.
[21] '(PDF) Unsupervised and Supervised Models in Machine Learning', ResearchGate. Accessed: Sep. 08, 2025. [Online]. Available: https://www.researchgate.net/publication/387136555_Unsupervised_and_Supervised_Models_in_Machine_Learning
[22] I. H. Sarker, 'Machine Learning: Algorithms, Real-World Applications and Research Directions', *SN Comput Sci*, vol. 2, no. 3, p. 160, 2021, doi: 10.1007/s42979-021-00592-x.
[23] M. Fatima and M. Pasha, 'Survey of Machine Learning Algorithms for Disease Diagnostic', *Journal of Intelligent Learning Systems and Applications*, vol. 09, no. 01, pp. 1–16, Jan. 2017, doi: 10.4236/jilsa.2017.91001.
[24] '[1906.01703] Basic Neural Units of the Brain: Neurons, Synapses and Action Potential'. Accessed: Sep. 08, 2025. [Online]. Available: https://arxiv.org/abs/1906.01703?utm_source=chatgpt.com
[25] S. Herculano-Houzel, 'The Human Brain in Numbers: A Linearly Scaled-up Primate Brain', *Front Hum Neurosci*, vol. 3, p. 31, Nov. 2009, doi: 10.3389/neuro.09.031.2009.
[26] S.-H. Han, K. W. Kim, S. Kim, and Y. C. Youn, 'Artificial Neural Network: Understanding the Basic Concepts without Mathematics', *Dement Neurocogn Disord*, vol. 17, no. 3, pp. 83–89, Sep. 2018, doi: 10.12779/dnd.2018.17.3.83.
[27] L. K. Avberšek and G. Repovš, 'Deep learning in neuroimaging data analysis: Applications, challenges, and solutions', *Front. Neuroimaging*, vol. 1, Oct. 2022, doi: 10.3389/fnimg.2022.981642.
[28] O. A. M. López, A. M. López, and D. J. Crossa, 'Fundamentals of Artificial Neural Networks and Deep Learning', in *Multivariate Statistical Machine Learning Methods for Genomic Prediction [Internet]*, Springer, 2022. doi: 10.1007/978-3-030-89010-0_10.
[29] J. Lederer, 'Activation Functions in Artificial Neural Networks: A Systematic Overview', Jan. 25, 2021, *arXiv*: arXiv:2101.09957. doi: 10.48550/arXiv.2101.09957.
[30] J. Bernal *et al.*, 'Deep convolutional neural networks for brain image analysis on magnetic resonance imaging: a review', *Artificial Intelligence in Medicine*, vol. 95, pp. 64–81, Apr. 2019, doi: 10.1016/j.artmed.2018.08.008.
[31] D. H. Hubel, 'Single unit activity in striate cortex of unrestrained cats', *J Physiol*, vol. 147, no. 2, pp. 226-238.2, Sep. 1959, doi: 10.1113/jphysiol.1959.sp006238.

[32] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, 'Convolutional neural networks: an overview and application in radiology', *Insights Imaging*, vol. 9, no. 4, pp. 611–629, Aug. 2018, doi: 10.1007/s13244-018-0639-9.
[33] 'What are Convolutional Neural Networks? | IBM'. Accessed: Sep. 05, 2025. [Online]. Available: https://www.ibm.com/think/topics/convolutional-neural-networks
[34] H. Yingge, I. Ali, and K.-Y. Lee, 'Deep Neural Networks on Chip - A Survey', in *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Busan, Korea (South): IEEE, Feb. 2020, pp. 589–592. doi: 10.1109/BigComp48618.2020.00016.
[35] I. Naseer, S. Akram, T. Masood, A. Jaffar, M. A. Khan, and A. Mosavi, 'Performance Analysis of State-of-the-Art CNN Architectures for LUNA16', *Sensors (Basel)*, vol. 22, no. 12, p. 4426, Jun. 2022, doi: 10.3390/s22124426.
[36] I. D. Mienye, T. G. Swart, G. Obaido, M. Jordan, and P. Ilono, 'Deep Convolutional Neural Networks in Medical Image Analysis: A Review', *Information*, vol. 16, no. 3, p. 195, Mar. 2025, doi: 10.3390/info16030195.
[37] O. Ronneberger, P. Fischer, and T. Brox, 'U-Net: Convolutional Networks for Biomedical Image Segmentation', May 18, 2015, *arXiv*: arXiv:1505.04597. doi: 10.48550/arXiv.1505.04597.
[38] W. Jiangtao, N. I. R. Ruhaiyem, and F. Panpan, 'A Comprehensive Review of U-Net and Its Variants: Advances and Applications in Medical Image Segmentation'.
[39] Y. Peng, M. Sonka, and D. Z. Chen, 'U-Net v2: Rethinking the Skip Connections of U-Net for Medical Image Segmentation', Mar. 30, 2024, *arXiv*: arXiv:2311.17791. doi: 10.48550/arXiv.2311.17791.
[40] Z. Zhou, M. M. Rahman Siddiquee, N. Tajbakhsh, and J. Liang, 'Unet++: 4th International Workshop on Deep Learning in Medical Image Analysis, DLMIA 2018 and 8th International Workshop on Multimodal Learning for Clinical Decision Support, ML-CDS 2018 Held in Conjunction with MICCAI 2018', *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support - 4th International Workshop, DLMIA 2018 and 8th International Workshop, ML-CDS 2018 Held in Conjunction with MICCAI 2018*, pp. 3–11, 2018, doi: 10.1007/978-3-030-00889-5_1.
[41] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, 'Rethinking Atrous Convolution for Semantic Image Segmentation', Dec. 05, 2017, *arXiv*: arXiv:1706.05587. doi: 10.48550/arXiv.1706.05587.
[42] J. Long, E. Shelhamer, and T. Darrell, 'Fully Convolutional Networks for Semantic Segmentation', Mar. 08, 2015, *arXiv*: arXiv:1411.4038. doi: 10.48550/arXiv.1411.4038.
[43] I. R. Vega, L. I. P. Urías, A. E. R. Mata, C. L. G. Mata, and A. P. González, 'EVALUACIÓN DE LOS MODELO U-NET TIPO XCEPTION Y DEEPLABV3+ EN LA DETECCIÓN DE NÓDULOS PULMONARES EN IMÁGENES DE TOMOGRAFÍA COMPUTARIZADA', vol. 45, 2023.
[44] 'The Comparison of U-net and Deeplab V3 as Semantic Segmentation Models for Food Images | Irwanto | Brahmana : Jurnal Penerapan Kecerdasan Buatan'. Accessed: Aug. 31, 2025. [Online]. Available: https://tunasbangsa.ac.id/pkm/index.php/brahmana/article/view/281/278
[45] B. Dhiyanesh, M. Vijayalakshmi, P. Saranya, and D. Viji, 'EnsembleEdgeFusion: advancing semantic segmentation in microvascular decompression imaging with innovative ensemble techniques', *Sci Rep*, vol. 15, no. 1, p. 17892, May 2025, doi: 10.1038/s41598-025-02470-5.

[46] K. H. Zou *et al.*, 'Statistical Validation of Image Segmentation Quality Based on a Spatial Overlap Index', *Acad Radiol*, vol. 11, no. 2, pp. 178–189, Feb. 2004, doi: 10.1016/S1076-6332(03)00671-8.
[47] T. Eelbode *et al.*, 'Optimization for Medical Image Segmentation: Theory and Practice when evaluating with Dice Score or Jaccard Index', *IEEE Trans. Med. Imaging*, vol. 39, no. 11, pp. 3679–3690, Nov. 2020, doi: 10.1109/TMI.2020.3002417.
[48] D. Müller, I. Soto-Rey, and F. Kramer, 'Towards a guideline for evaluation metrics in medical image segmentation', *BMC Res Notes*, vol. 15, no. 1, p. 210, Dec. 2022, doi: 10.1186/s13104-022-06096-y.
[49] L. Gao, L. Zhang, C. Liu, and S. Wu, 'Handling Imbalanced Medical Image Data: A Deep-Learning-Based One-Class Classification Approach', *Artif Intell Med*, vol. 108, p. 101935, Aug. 2020, doi: 10.1016/j.artmed.2020.101935.
[50] L. Perez and J. Wang, 'The Effectiveness of Data Augmentation in Image Classification using Deep Learning', Dec. 13, 2017, *arXiv*: arXiv:1712.04621. doi: 10.48550/arXiv.1712.04621.
[51] A. Chaddad, Y. Hu, Y. Wu, B. Wen, and R. Kateb, 'Generalizable and explainable deep learning for medical image computing: An overview', *Current Opinion in Biomedical Engineering*, vol. 33, p. 100567, Mar. 2025, doi: 10.1016/j.cobme.2024.100567.
[52] 'What's New In Python 3.10 — Python 3.10.18 documentation'. Accessed: Aug. 31, 2025. [Online]. Available: https://docs.python.org/3.10/whatsnew/3.10.html
[53] A. I. A. Eid, S. A. Mjlae, S. Y. Rababa'h, A. Hammad, and M. A. I. Rababah, 'Comparative Analysis of Machine Learning Libraries for Neural Networks: A Benchmarking Study', Feb. 04, 2025, *bioRxiv*. doi: 10.1101/2025.02.02.635632.
[54] 'Label Studio Documentation'. Accessed: Aug. 31, 2025. [Online]. Available: https://labelstud.io/guide
[55] A. Antoniou *et al.*, 'MR relaxation times of agar-based tissue-mimicking phantoms', *J Appl Clin Med Phys*, vol. 23, no. 5, p. e13533, Apr. 2022, doi: 10.1002/acm2.13533.
[56] 'Label Studio Documentation — Interactive annotation in Label Studio with Segment Anything Model (SAM)'. Accessed: Sep. 07, 2025. [Online]. Available: https://labelstud.io/tutorials/segment_anything_model
[57] *HumanSignal/label-studio-ml-backend*. (Aug. 29, 2025). Python. HumanSignal. Accessed: Aug. 31, 2025. [Online]. Available: https://github.com/HumanSignal/label-studio-ml-backend
[58] 'Brain Tumor Segmentation(BraTS2020)'. Accessed: Aug. 31, 2025. [Online]. Available: https://www.kaggle.com/datasets/awsaf49/brats2020-training-data

7 Annexs

Supplementary table 1. Comprehensive summary of training, validation, and test results for all architectures and approaches.

Architecture	Approach	Best Epoch	LOSS		TRAIN			VALIDATION			TEST		
			Train Loss	Validation Loss	Train Accuracy	Train IoU	Train Dice	Validation Accuracy	Validation IoU	Validation Dice	Test Accuracy	Test IoU	Test Dice
U-Net	1	75	0.32	0.52	0.98	0.80	0.88	0.98	0.71	0.78	0.93	0.57	0.70
	2	85	0.38	0.53	0.97	0.73	0.82	0.97	0.69	0.77	0.91	0.48	0.61
	3	82	0.40	0.53	0.92	0.75	0.85	0.97	0.71	0.79	0.93	0.58	0.69
DeepLabV3	1	45	0.33	0.62	0.99	0.86	0.92	0.97	0.70	0.78	0.94	0.61	0.73
	2	37	0.46	0.62	0.99	0.78	0.85	0.97	0.68	0.75	0.93	0.52	0.62
	3	16	0.37	0.61	0.96	0.77	0.85	0.96	0.67	0.75	0.94	0.61	0.73
FCN	1	34	0.17	0.55	0.99	0.92	0.95	0.97	0.71	0.80	0.94	0.60	0.72
	2	28	0.27	0.57	0.99	0.79	0.86	0.97	0.68	0.77	0.92	0.53	0.66
	3	9	0.44	0.60	0.93	0.72	0.83	0.97	0.67	0.75	0.94	0.59	0.71