

UNIVERSITY OF PADUA

DEPARTMENT OF INFORMATION ENGINEERING

Master's Degree in Bioengineering

Development of Encoder-only Neural Network for brain imaging analysis

Supervisor:
Prof. Atzori Manfredo

Candidate:
Francesco Casson

Co-supervisor:
Dr. Tshimanga Louis Fabrice

Student ID:
2149719

ACADEMIC YEAR 2025/2026

Abstract

The increasing availability of heterogeneous neuroimaging data represents an important opportunity for the development of artificial intelligence methods capable of supporting large-scale neurological research. However, the training of deep learning models for medical imaging is often limited by the scarcity of manually annotated datasets and by the substantial variability introduced by different acquisition protocols, scanners, imaging sequences, and patient populations. Self-Supervised Learning (SSL) offers a promising alternative by enabling representation learning directly from unlabelled data, reducing the dependence on manual annotations.

This thesis presents the development and analysis of an anatomical-aware representation learning framework for heterogeneous three-dimensional brain Magnetic Resonance Imaging (MRI), developed within the context of the HEREDITARY project. The proposed architecture combines a three-dimensional convolutional feature extractor initialized from pretrained SynthSeg weights with a Feature Pyramid Network (FPN), a TokenLearner module for converting spatial feature maps into a compact set of anatomical-aware tokens, and a C-JEPA-based encoder architecture for self-supervised representation learning.

Experiments were conducted on a curated subset of the FOMO300K dataset, selected to provide variability across subjects and acquisition sources. SynthSeg+ was employed during preprocessing to obtain anatomical segmentations, cortical parcellations, and quality-control information. Three masking strategies were investigated, operating either on the input image or on the latent token representation, while an ablation study was performed to assess their interaction with the learning objective.

While SynthSeg provides general-purpose anatomical features, the proposed framework is designed to further adapt these representations through self-supervised learning and organize them into spatially structured tokens, with the aim of capturing anatomically meaningful information that can be exploited in subsequent downstream analyses, including clinical phenotype clustering.

The results show that the strictly self-supervised formulation, based on JEPA and VICReg objectives, was able to produce meaningful latent representations, but exhibited limited anatomical localization. The introduction of an auxiliary spatial loss combining Dice and Binary Cross-

Entropy substantially improved the localization of the TokenLearner attention maps and increased the Dice scores obtained for most anatomical regions. However, the learned representation did not exhibit a strict one-to-one correspondence between individual tokens and anatomical regions, indicating that anatomical information remains distributed across multiple tokens.

The study also highlights the impact of computational constraints on the development and evaluation of three-dimensional representation learning models. Limited GPU memory required reductions in spatial resolution, constrained batch sizes, and restricted the size of the experimental cohort, particularly affecting the representation of small anatomical structures. Overall, the results provide a proof of concept for integrating pretrained anatomical representations with self-supervised learning to obtain spatially structured representations from heterogeneous brain MRI. Further work should focus on larger cohorts, independent validation and test sets, higher-resolution representations, and the extension of the framework towards multimodal biomedical data.

Sommario

La crescente disponibilità di dati di neuroimaging eterogenei rappresenta un'importante opportunità per lo sviluppo di metodi di intelligenza artificiale in grado di supportare la ricerca neurologica su larga scala. Tuttavia, l'addestramento di modelli di deep learning per l'imaging medico è spesso limitato dalla scarsità di dataset annotati manualmente e dalla significativa variabilità introdotta da differenti protocolli di acquisizione, scanner, sequenze di imaging e popolazioni di pazienti. Il Self-Supervised Learning (SSL) rappresenta una promettente alternativa, in quanto consente di apprendere rappresentazioni direttamente da dati non annotati, riducendo la dipendenza dalle annotazioni manuali.

Questa tesi presenta lo sviluppo e l'analisi di un framework per l'apprendimento di rappresentazioni anatomically-aware a partire da immagini tridimensionali eterogenee di Risonanza Magnetica (MRI) cerebrale, sviluppato nell'ambito del progetto HEREDITARY. L'architettura proposta combina un estrattore di caratteristiche convoluzionale tridimensionale, inizializzato a partire dai pesi pretrained di SynthSeg, con una Feature Pyramid Network (FPN), un modulo TokenLearner per convertire le mappe di caratteristiche spaziali in un insieme compatto di token anatomically-aware e un'architettura encoder basata su C-JEPA per l'apprendimento self-supervised delle rappresentazioni.

Gli esperimenti sono stati condotti su un sottoinsieme selezionato del dataset FOMO300K, scelto al fine di garantire variabilità tra soggetti e fonti di acquisizione. SynthSeg+ è stato impiegato durante il preprocessing per ottenere segmentazioni anatomiche, parcellazioni corticali e informazioni relative al controllo di qualità. Sono state investigate tre strategie di masking, applicate rispettivamente all'immagine di input oppure alla rappresentazione latente costituita dai token, mentre è stato condotto uno studio di ablazione per valutarne l'interazione con l'obiettivo di apprendimento.

Sebbene SynthSeg fornisca caratteristiche anatomiche general-purpose, il framework proposto è progettato per adattare ulteriormente tali rappresentazioni attraverso il self-supervised learning e organizzarle in token strutturati spazialmente, con l'obiettivo di catturare informazioni anatomicamente significative che possano essere sfruttate nelle successive analisi downstream, incluso il clustering dei fenotipi clinici.

I risultati mostrano che la formulazione strettamente self-supervised, basata sugli obiettivi JEPA e VICReg, è stata in grado di produrre rappresentazioni latenti significative, ma ha mostrato una limitata localizzazione anatomica. L'introduzione di una loss spaziale ausiliaria, combinando Dice e Binary Cross-Entropy, ha migliorato sostanzialmente la localizzazione delle mappe di attenzione del TokenLearner e aumentato i valori di Dice ottenuti per la maggior parte delle regioni anatomiche. Tuttavia, la rappresentazione appresa non ha mostrato una corrispondenza strettamente uno-a-uno tra i singoli token e le regioni anatomiche, indicando che l'informazione anatomica rimane distribuita tra più token.

Lo studio evidenzia inoltre l'impatto dei vincoli computazionali sullo sviluppo e sulla valutazione di modelli di representation learning tridimensionali. La limitata memoria della GPU ha richiesto una riduzione della risoluzione spaziale, ha imposto restrizioni sulle dimensioni dei batch e ha limitato la numerosità della coorte sperimentale, influenzando in particolare la rappresentazione delle strutture anatomiche di piccole dimensioni. Nel complesso, i risultati forniscono una proof of concept per l'integrazione di rappresentazioni anatomiche pretrained con il self-supervised learning, al fine di ottenere rappresentazioni spazialmente strutturate a partire da immagini MRI cerebrali eterogenee. Gli sviluppi futuri dovrebbero concentrarsi sull'utilizzo di coorti più ampie, di set di validazione e test indipendenti, di rappresentazioni a maggiore risoluzione e sull'estensione del framework verso dati biomedici multimodali.

Contents

1 Introduction **1**

2 Background **7**

2.1 Brain Imaging 7

2.1.1 MRI 7

2.1.2 DWI, T1w and T2w 8

2.1.3 Brain MRI Segmentation and Skull Stripping 9

2.2 Deep Learning 11

2.2.1 Convolutional Neural Networks 11

2.2.2 ResNet 16

2.2.3 Feature Pyramidal Network 18

2.2.4 Transformer 20

2.2.5 Tokenization 26

2.3 Self-Supervised Learning 35

2.3.1 BYOL 37

2.3.2 JEPA 39

2.3.3 VICReg 43

2.4 SynthSeg 46

2.4.1 Synthetic Data Generation 47

2.4.2 Segmentation Architecture 48

2.4.3 Training Objective 49

2.4.4 Inference on real MRI scans 49

2.4.5 SynthSeg+ 50

2.4.6 Role of SynthSeg in the proposed architecture 51

3 Objectives **53**

4 Methods **57**

4.1 Dataset 57

4.2 MRI Segmentation and Cortical Parcellation . . . 59
4.3 The Pipeline . . . 61
4.4 CNN + FPN . . . 62
4.4.1 CNN Encoder . . . 62
4.4.2 Feature Pyramid Network . . . 64
4.5 TokenLearner . . . 66
4.6 JEPA . . . 70
4.6.1 Forward Encoder . . . 70
4.6.2 Context and Target Encoders . . . 72
4.7 Masking . . . 74
4.7.1 Anatomical Image . . . 75
4.7.2 Geometrical Image . . . 75
4.7.3 Anatomical Token . . . 76
4.8 Loss . . . 76
4.8.1 VICReg . . . 77
4.8.2 BCE + Dice Auxiliary Loss . . . 79
4.8.3 BCE . . . 80
4.8.4 Dice . . . 80
4.9 Training . . . 81
4.9.1 Hyperparameters and Loss Weights . . . 81
4.9.2 Data Processing and Input Pipeline . . . 82
4.9.3 Optimization and Memory-Efficiency Strategies . . . 83
4.9.4 Parameter Distribution and Computational Efficiency . . . 83
4.10 Hardware and Sustainability . . . 84
4.11 AI Usage . . . 85

5 Results **87**
5.1 Dataset . . . 87
5.2 CNN . . . 89
5.3 Loss . . . 90
5.3.1 VICReg . . . 90
5.3.2 Auxiliary Loss . . . 91
5.4 Ablation Study . . . 92
5.4.1 Masking in the FPN . . . 94
5.4.2 Loss Evolution . . . 96
5.4.3 Dice per ROI . . . 96
5.4.4 Token Similarity . . . 98

5.4.5 Latent Space Organization 103
5.4.6 Information Carried by Tokens 103

6 Conclusions **109**
6.1 Considerations based on the Results 109
6.2 Next Steps 111
6.2.1 Dataset 112
6.2.2 3D CNN and Feature Pyramid Network 113
6.2.3 TokenLearner 114
6.2.4 Loss 115
6.2.5 Final Considerations 116

Bibliography **119**

Chapter 1

Introduction

Neurological disorders represent one of the major public health challenges worldwide. According to the World Health Organization (WHO) Global Status Report on Neurology 2025, an estimated 3.4 billion people were living with a neurological condition in 2021, corresponding to approximately 42% of the global population. Neurological conditions were also responsible for approximately 11.8 million deaths worldwide, making them the leading cause of disability and the second leading cause of death globally. These figures highlight not only the prevalence of neurological disorders, but also the substantial burden they place on individuals and healthcare systems. [1]

The burden of neurological disorders is further amplified by substantial inequalities in access to neurological care. The WHO report highlights a pronounced disparity in the availability of specialized healthcare professionals between countries. High-income countries have, on average, approximately nine neurologists per 100,000 inhabitants, whereas low-income countries have fewer than one neurologist per 100,000 inhabitants. Moreover, only 39% of countries have implemented a national strategy addressing neurological health, while only 15% systematically collect epidemiological data on neurological disorders. These disparities are particularly relevant because the lack of specialized personnel and systematic data collection can limit both the diagnosis and the long-term management of neurological conditions. [1]

The situation is also significant at the European level. Neurological disorders account for more than 90 million disability-adjusted life years (DALYs) in Europe and are associated with an estimated annual societal cost of approximately €900 billion. A DALY represents one lost year of healthy life and combines years of life lost due to premature mortality with years lived with disability. The magnitude of this burden emphasizes the importance of improving not only therapeutic strategies, but also the ability to detect, characterize, and monitor neurological disorders efficiently. [1]

Italy reflects some of the same challenges. Although approximately 7,000 neurologists cur-

rently operate in the country, fewer than 3,000 work within the Italian National Health Service (Servizio Sanitario Nazionale, SSN). This corresponds to an average of approximately five neurologists per 100,000 inhabitants within public healthcare structures. Furthermore, national averages conceal substantial geographical inequalities, with particularly pronounced shortages outside major urban centres and in rural, mountainous, and island areas. In this context, technologies capable of supporting specialists and reducing the time required for the analysis of medical data could represent an important contribution to the sustainability of neurological healthcare. [1]

One of the fundamental tools for the investigation of neurological disorders is brain imaging. Medical imaging provides information about the structure and organization of the brain that cannot be obtained through clinical examination alone. Among the available imaging modalities, Magnetic Resonance Imaging (MRI) plays a particularly important role because of its excellent soft-tissue contrast and its ability to characterize brain anatomy without exposing the patient to ionizing radiation. MRI can provide detailed information about both normal and pathological brain structures and can therefore be used to investigate a broad range of neurological conditions.

The increasing availability of MRI data has also enabled the development of quantitative approaches that go beyond visual inspection of individual scans. Modern image-processing techniques can extract structural and morphological information from MRI volumes and provide quantitative measurements that can be used for diagnosis, prognosis, and longitudinal monitoring. Examples include brain volumetry, shape analysis, voxel-based morphometry (VBM), and voxel-based lesion-symptom mapping (VLSM). These approaches have been investigated in several neurological and psychiatric conditions, including Alzheimer's disease, Parkinson's disease, multiple sclerosis, epilepsy, and major depressive disorder. Quantitative analysis can therefore complement the expertise of clinicians by providing reproducible measurements of subtle structural changes that may be difficult to identify through visual inspection alone. [2]

At the same time, the amount of neuroimaging data produced in clinical practice is continuously increasing. A recent analysis of neuroimaging utilization across 29 European countries reported a 43.5% increase in MRI examination rates between 2015 and 2022, from 5,746 to 8,244 examinations per 100,000 inhabitants per year. During the same period, the number of available MRI scanners also increased, although scan volumes grew faster than scanner availability. The study further highlights substantial regional differences in imaging utilization across Europe. Brain and spinal examinations constitute an important proportion of MRI utilization, further contributing to the growing amount of neurological imaging data that must be interpreted and managed. [3]

This increase creates a clear tension between the growing amount of available imaging data and the limited availability of specialized personnel capable of analysing it. Even when an MRI

examination can be acquired relatively quickly, its interpretation and subsequent quantitative analysis may require substantial time and specialized expertise. This problem becomes particularly relevant when large cohorts are considered, as in longitudinal studies, population-scale research, or the analysis of large collections of heterogeneous clinical scans. [4]

Artificial Intelligence (AI), and in particular Deep Learning, has emerged as a promising approach to address some of these challenges. Deep learning methods have demonstrated state-of-the-art performance in several stages of the MRI processing pipeline, including acquisition acceleration, image reconstruction, segmentation, registration, and quantitative analysis. By automating or assisting parts of the imaging workflow, these methods have the potential to reduce computational and human workload while maintaining or improving the quality of the resulting analyses. [5]

The potential impact of AI is not limited to image reconstruction or preprocessing. AI-assisted interpretation can also reduce the time required by specialists to analyse medical images. For example, a recent multicentre study evaluating AI assistance in knee MRI interpretation reported a 13.4% reduction in reporting time when fully integrated AI-assisted structured reporting was used. The benefit was even larger among general radiologists, who showed a 17.5% reduction in interpretation time. Although these results concern musculoskeletal rather than neurological MRI, they illustrate the broader potential of AI-based tools to reduce the workload associated with medical image interpretation.

However, the development of effective AI systems for medical imaging introduces another fundamental challenge: data. Modern deep learning models generally benefit from large and diverse training datasets, yet medical imaging datasets are considerably more difficult to construct than datasets in many other fields. Medical images are heterogeneous by nature, as they can differ in acquisition protocol, scanner manufacturer, magnetic field strength, spatial resolution, imaging sequence, anatomical coverage, and patient population. In addition, medical data are subject to strict privacy and regulatory requirements, while obtaining reliable expert annotations can be expensive and time-consuming.

These difficulties become particularly important in the context of neurological imaging. A model trained on a limited dataset may learn features that are specific to the acquisition characteristics of that dataset rather than features that generalize across different populations and imaging protocols. Consequently, increasing the quantity of training data alone is not sufficient: datasets must also capture the substantial variability present in real-world medical imaging.

The European Union has recognized the importance of addressing these challenges through initiatives aimed at improving the use of heterogeneous health data. One example is HEREDITARY, a European research programme focused on exploiting multimodal health data to improve disease prevention, diagnosis, and treatment. Such initiatives highlight the growing im-

portance of combining different sources of biomedical information, including genomic data, medical imaging, physiological signals, and clinical text. However, the integration of these heterogeneous data sources remains challenging due to differences in data formats and distributions, technical and legal constraints, privacy requirements, and the difficulty of obtaining sufficiently large and representative datasets.

The availability of large collections of unlabelled medical images therefore represents an important opportunity for the development of modern AI methods. In particular, the need for manually annotated training data can be reduced through Self-Supervised Learning (SSL), in which the learning signal is obtained directly from the input data rather than from manually assigned labels. SSL is especially attractive for medical imaging because large quantities of scans may be available even when only a small fraction of them have been manually annotated by experts. [6]

This distinction is particularly relevant for brain MRI. Producing detailed anatomical annotations for a large number of three-dimensional scans would require substantial expert effort, whereas unlabelled MRI data can be collected at a much larger scale. Self-supervised approaches can exploit these unlabelled volumes to learn general-purpose representations before the resulting models are adapted to specific downstream tasks. Consequently, the availability of large and heterogeneous MRI datasets is becoming an important prerequisite for the development of increasingly capable self-supervised models for neuroimaging.

An important contribution in this direction is FOMO300K, a large-scale heterogeneous dataset specifically designed to support self-supervised learning on three-dimensional brain MRI. The dataset contains 59,969 subjects, 82,678 sessions, and 318,877 scans, collected from hundreds of datasets and covering a wide variety of clinical and research imaging conditions. It includes multiple MRI contrasts and acquisition types, thereby providing substantial variability in both the anatomical characteristics of the subjects and the properties of the acquired images.

This heterogeneity is particularly valuable for self-supervised learning. Rather than learning from a dataset characterized by a single acquisition protocol, a model trained on FOMO300K can be exposed to a much broader distribution of MRI appearances. Such variability can encourage the learned representations to capture information that is more general and less dependent on the characteristics of a specific scanner, sequence, or dataset. The scale of FOMO300K also makes it possible to investigate self-supervised pre-training in a setting that more closely resembles the large-scale training regimes increasingly adopted in modern representation learning.

The combination of large-scale heterogeneous MRI data and self-supervised learning therefore provides a promising framework for developing representations that can be reused across different neuroimaging applications. Rather than training a separate model from scratch for every individual task, a model can first learn general representations from large quantities of

unlabelled data and subsequently use these representations as the basis for more specialized analyses.

In this context, the present work investigates a self-supervised learning framework for the representation of heterogeneous three-dimensional brain MRI. The proposed architecture combines a 3D convolutional feature extractor, initialized using weights derived from SynthSeg, with a C-JEPA-based self-supervised objective and a subsequent tokenization and Transformer-based processing stage. The objective is to exploit the anatomical information encoded by the pretrained convolutional features while learning representations from large-scale heterogeneous MRI data without relying on manual annotations during the self-supervised training stage.

The following sections therefore introduce the theoretical and methodological components underlying this approach. First, the principles of Self-Supervised Learning and the representation-learning methods on which the proposed framework is based are introduced. Subsequently, the SynthSeg model and its role as an anatomical feature extractor are described, followed by the tokenization strategy and the Transformer architecture used to process the resulting representations. Finally, the proposed C-JEPA framework and its application to heterogeneous brain MRI are presented.

Chapter 2

Background

2.1 Brain Imaging

Despite not being the main focus of this work it would be fair to spend few lines on our dataset, especially the modalities with which we acquire brain images, the preprocessing and the segmentation we apply in our workflow.

2.1.1 MRI

Magnetic Resonance Imaging (MRI) is a relatively recent diagnostic imaging technique compared with other modalities such as X-ray imaging, ultrasound, and computed tomography. Its development relies on the principles of nuclear magnetic resonance (NMR), which originated from the discovery of nuclear spin resonance by Rabi in 1938 and was further demonstrated in condensed matter by Bloch and Purcell in 1946.[7]

The fundamental principle behind MRI is the intrinsic angular momentum, or spin, of atomic nuclei. Among biological elements, hydrogen nuclei are particularly relevant for MRI due to their high abundance in the human body, mainly in water molecules, and their non-zero nuclear spin, which generates a magnetic moment.[7]

When exposed to a strong external magnetic field (B0), hydrogen nuclei partially align either parallel or antiparallel to the field direction, corresponding respectively to lower and higher energy states. Due to the interaction between their magnetic moments and B0, nuclei undergo a precessional motion at the Larmor frequency. By applying a radiofrequency (RF) pulse perpendicular to B0 at this frequency, energy is transferred to the nuclei, perturbing their equilibrium state. After the RF pulse is removed, the nuclei return to equilibrium through relaxation processes characterized mainly by longitudinal (T1) and transverse (T2) relaxation times. During this process, electromagnetic signals generated by the precessing nuclei are detected by receiver

coils and used to reconstruct the final MR image.[8]

MRI contrast depends on differences in proton density and relaxation properties among tissues, allowing the visualization of anatomical structures with high soft-tissue contrast. The strength of the main magnetic field directly affects the signal-to-noise ratio (SNR): higher field strengths generally provide increased SNR, enabling higher spatial resolution and/or shorter acquisition times. Consequently, clinical MRI systems commonly operate at 1.5T and 3T, with the latter considered a high-field configuration[8]

In the presented dataset (FOMO-300k), 94% of the acquired images originate from high-field MRI scanners.[9]

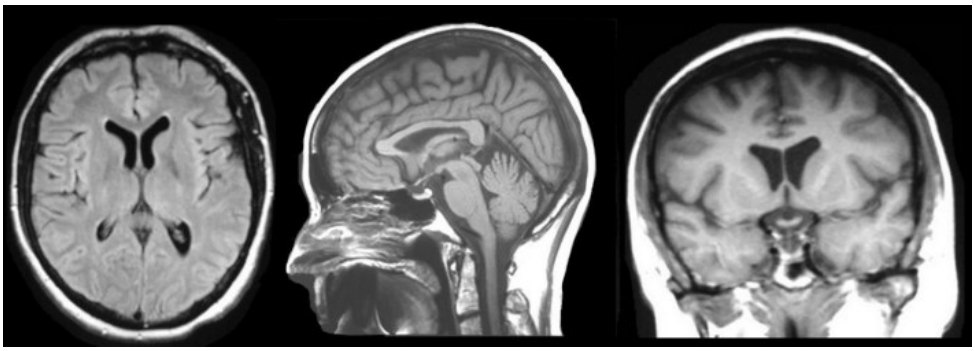


Figure 2.1: Representative MRI scans illustrating brain anatomy in three orthogonal views: axial (left), sagittal (center), and coronal (right).

2.1.2 DWI, T1w and T2w

The FOMO300k dataset is mainly composed of three MRI image modalities: - 39% diffusion-weighted imaging (DWI) - 29% T1-weighted imaging (T1w) - 11% T2-weighted imaging (T2w)[9]

Diffusion-weighted imaging (DWI) is an MRI technique that provides image contrast based on the microscopic motion of water molecules within biological tissues. Unlike conventional anatomical MRI sequences, DWI is sensitive to variations in water diffusivity, which can be quantified through the apparent diffusion coefficient (ADC). Alterations in cellular density and tissue organization affect water mobility, making DWI particularly valuable in detecting acute ischemic stroke, where cytotoxic edema caused by restricted water diffusion leads to characteristic hyperintense signals on diffusion-weighted images and reduced ADC values. Due to its ability to detect microstructural changes occurring before conventional T1- and T2-weighted sequences, DWI has become a fundamental sequence in acute neurological imaging.[10]

T1-weighted imaging (T1w) is an MRI acquisition technique in which image contrast is primarily determined by differences in the longitudinal relaxation time (T1) of tissues. T1 represents the time required for the recovery of the longitudinal component of the net magnetization

after excitation by a radiofrequency pulse. Tissues with shorter T1 relaxation times recover their longitudinal magnetization faster and appear brighter on T1-weighted images, whereas tissues with longer T1 values appear darker. In clinical neuroimaging, T1w images provide excellent anatomical detail and are widely used for visualization of brain structures, cortical anatomy, tissue segmentation, and assessment of structural abnormalities. Typically, white matter appears hyperintense compared with gray matter, while cerebrospinal fluid (CSF) appears hypointense.[11]

T2-weighted imaging (T2w) is based on differences in transverse relaxation times (T2), which describe the decay of transverse magnetization due to loss of phase coherence between spins. Tissues with longer T2 relaxation times maintain transverse magnetization for a longer period and appear brighter on T2-weighted images, whereas tissues with shorter T2 values appear darker. Since pathological processes often increase tissue water content, T2-weighted sequences are particularly sensitive to abnormalities associated with edema, inflammation, tumors, and other lesions. In the brain, gray matter generally appears brighter than white matter, while cerebrospinal fluid exhibits high signal intensity due to its long T2 relaxation time.[12] [2.2]

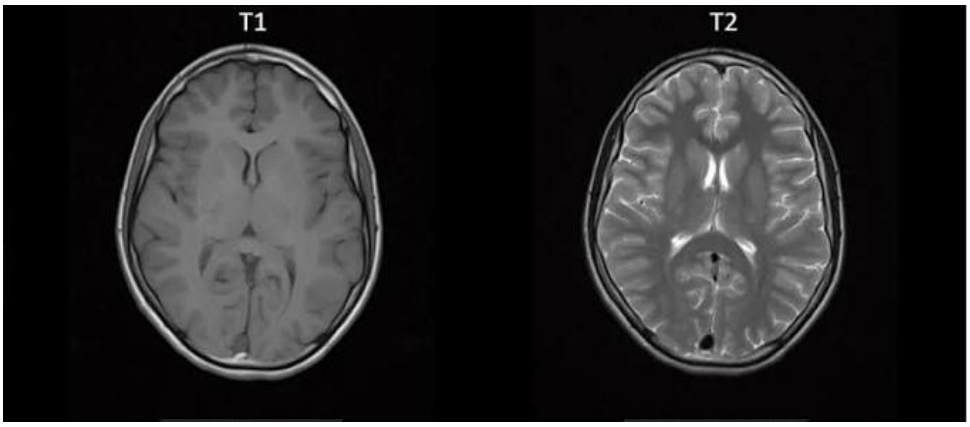


Figure 2.2: Example of a T1-weighted imaging (left) and a T2-weighted imaging(right)

Together, DWI, T1w, and T2w sequences provide complementary information: T1-weighted images offer high-resolution anatomical characterization, T2-weighted images highlight variations in tissue water content and pathological changes, while DWI provides sensitivity to microstructural alterations related to water diffusion.

2.1.3 Brain MRI Segmentation and Skull Stripping

Magnetic Resonance Imaging (MRI) provides detailed anatomical information for the study of brain structure and pathology. However, MRI volumes also contain non-brain structures such as the skull, scalp, and surrounding soft tissues, which are generally irrelevant to brain analysis and may introduce additional variability into subsequent processing and learning algorithms.

For this reason, brain segmentation and skull stripping are commonly used as preprocessing steps in neuroimaging pipelines.

Brain segmentation refers to the identification and delineation of anatomical structures within an MRI volume. Depending on the analysis, segmentation can range from broad tissue classes, such as gray matter, white matter, and cerebrospinal fluid, to fine-grained parcellations of individual brain regions. These segmentations can be used for anatomical characterization, region-based measurements, image alignment, or mask generation.

A related but distinct operation is skull stripping, or brain extraction, whose objective is to separate the brain from non-brain tissue. Given an MRI volume I, a binary brain mask M can be defined as

$$M(x) = \begin{cases} 1, & \text{if } x \text{ belongs to the brain,} \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

The mask can then be applied to the original image through an element-wise multiplication,

$$I_{\text{brain}} = I \odot M, \tag{2.2}$$

resulting in a brain-extracted volume in which voxels outside the brain are removed or set to zero. This reduces irrelevant information and focuses subsequent processing on the anatomical structures of interest.

This preprocessing step is particularly relevant for large and heterogeneous MRI collections, where differences in acquisition protocols, scanners, spatial resolution, and image contrast can introduce substantial variability. Non-brain structures may further contribute to these differences and provide image characteristics unrelated to brain anatomy. Their removal can therefore help focus feature extraction on the brain itself.

Several approaches have been proposed for automated brain extraction, ranging from conventional image-processing techniques to dedicated deep learning models. In this work, SynthSeg is employed as the main tool for obtaining the brain mask. SynthSeg is a deep learning-based brain segmentation method designed to perform anatomical segmentation across MRI scans with different contrasts, resolutions, and acquisition conditions, without requiring a separate model or retraining for each imaging protocol.

Although SynthSeg produces a multi-class anatomical segmentation, its output can also be exploited to obtain a binary brain mask. The anatomical labels corresponding to brain structures are combined into a single foreground class, while voxels outside the brain are assigned to the background. The resulting mask can subsequently be applied to the original MRI volume to perform skull stripping.[13] [**??**]

In this work, SynthSeg therefore plays a dual role in the processing pipeline. First, its

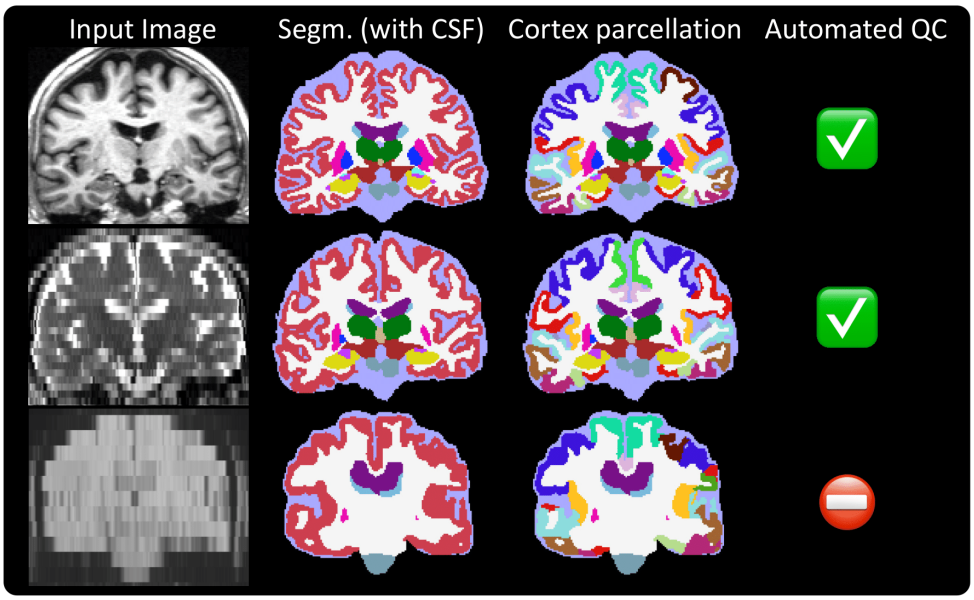


Figure 2.3: SynthSeg's automated brain MRI tissue segmentation, cortical parcellation, and quality control workflow across varying image quality levels. The pipeline processes MRI scans (left column) to perform tissue segmentation including cerebrospinal fluid (second column) and fine-grained cortical parcellation into color-coded anatomical regions (third column). An automated Quality Control (QC) module (right column) evaluates processing fidelity.

anatomical segmentation is used to identify the brain region and remove non-brain tissue from the MRI volumes. Second, the pretrained representation learned by its encoder is used to initialize the three-dimensional feature extraction component of the proposed self-supervised learning architecture. The latter aspect is discussed in greater detail in the dedicated SynthSeg section, while the present section focuses on its role in brain segmentation and skull stripping

2.2 Deep Learning

Deep learning has revolutionized the field of computer vision, enabling models to automatically learn hierarchical feature representations directly from raw data without relying on handcrafted descriptors.

2.2.1 Convolutional Neural Networks

Among the different deep learning architectures, Convolutional Neural Networks (CNNs) have become a standard approach for image analysis due to their ability to efficiently exploit the spatial structure of visual data.

A CNN is composed of a sequence of layers, typically organized in a hierarchical fashion. Each stage generally consists of a convolutional layer [2.5] followed by a non-linear activation

function, such as the Rectified Linear Unit (ReLU) [2.7], and, depending on the architecture, a pooling or downsampling operation. Pooling operations were traditionally used to progressively reduce the spatial resolution of feature maps while increasing the effective receptive field of the network [2.6]. In modern architectures, however, pooling is often replaced or complemented by strided convolutions, which allow the network to learn the downsampling operation directly.

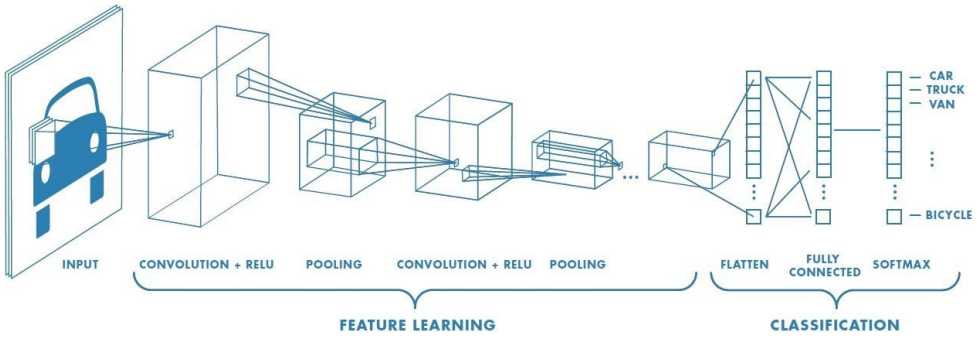


Figure 2.4: Schematic representation of a CNN pipeline divided into feature extraction and classification stages. The input image passes through alternating convolutional layers with ReLU activation for feature extraction and pooling layers for spatial downsampling. The resulting feature maps are flattened and processed by fully connected layers, culminating in a Softmax output layer that assigns class probabilities for object recognition.

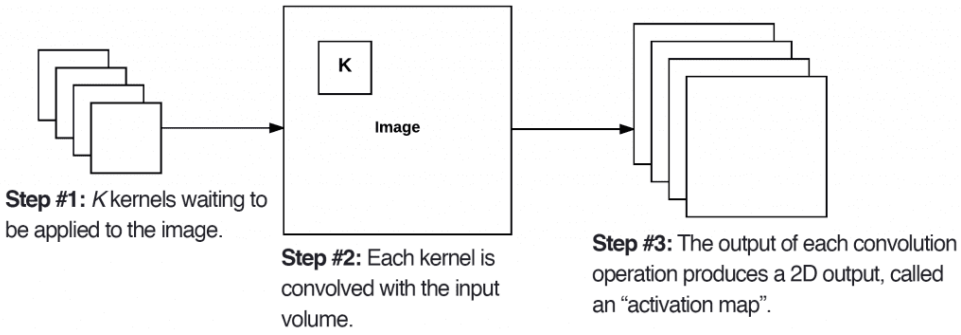


Figure 2.5: Example of the structure of a convolutional block. Left: At each convolutional layer in a CNN, there are K kernels applied to the input volume. Middle: Each of the K kernels is convolved with the input volume. Right: Each kernel produces a 2D output, called an activation map.

The fundamental building block of a CNN is the convolutional layer. Rather than learning a separate weight for every input pixel, CNNs exploit the concept of local connectivity by learning a small set of parameters, commonly referred to as a kernel or filter. The kernel is applied to local neighborhoods of the input while being progressively shifted across the spatial dimensions. At each location, the values within the local neighborhood are multiplied by the corresponding kernel weights and summed to produce an output value. For a two-dimensional input, the resulting feature map can be expressed as

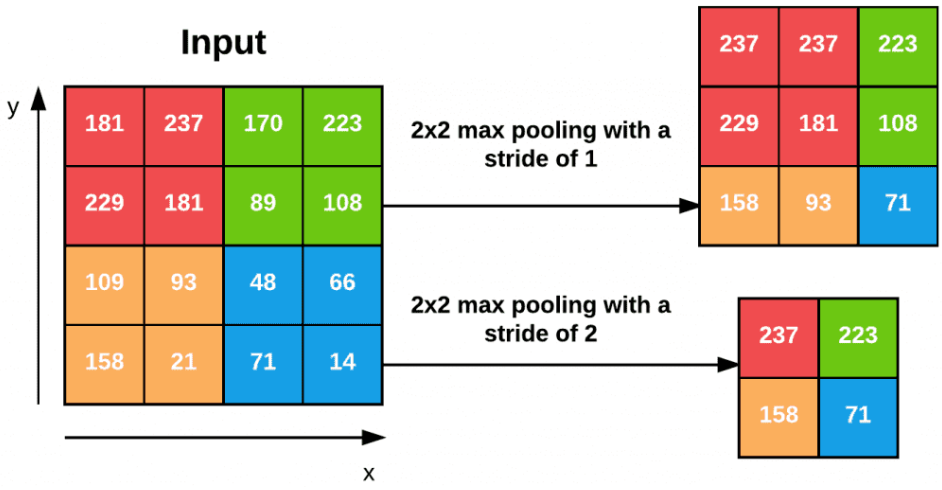


Figure 2.6: The diagram illustrates an example of Pooling: a 2×2 max pooling process applied to a 4×4 input feature map under different stride configurations. When applying a stride of 1 (top), the filter moves incrementally over overlapping sub-regions to select maximum pixel intensities, yielding a 3×3 output map. In contrast, a stride of 2 (bottom) slides across non-overlapping quadrants, extracting the dominant value per region to downsample the input into a 2×2 output matrix. This operation reduces spatial dimensions and computational complexity while granting the network local translation invariance.

$$y(i,j) = b + \sum_{m=0}^{K_h-1} \sum_{n=0}^{K_w-1} w(m,n),x(i+m,j+n), \tag{2.3}$$

where x represents the input feature map, w denotes the convolutional kernel, b is a learnable bias term, and y represents the resulting feature map. The kernel dimensions are denoted by K_h and K_w, while (i,j) identifies the spatial position at which the convolution is evaluated.

The same kernel is applied across the entire spatial extent of the input, a property known as weight sharing. This substantially reduces the number of trainable parameters compared to fully connected networks and allows the same learned pattern to be detected at different spatial locations. As a result, convolutional layers are well suited to learning spatially structured features. The resulting output, commonly referred to as an activation map or feature map, highlights the regions where the patterns encoded by the learned filters are present. [2.8]

As information propagates through successive convolutional layers, the learned representations become increasingly abstract. Early layers typically detect simple low-level features, such as edges, corners, and local intensity or texture patterns. Intermediate layers combine these elementary features into increasingly complex structures, while deeper layers capture higher-level semantic information that is more closely related to the characteristics relevant to the target task. In medical imaging, these representations may correspond to anatomical structures, tissue

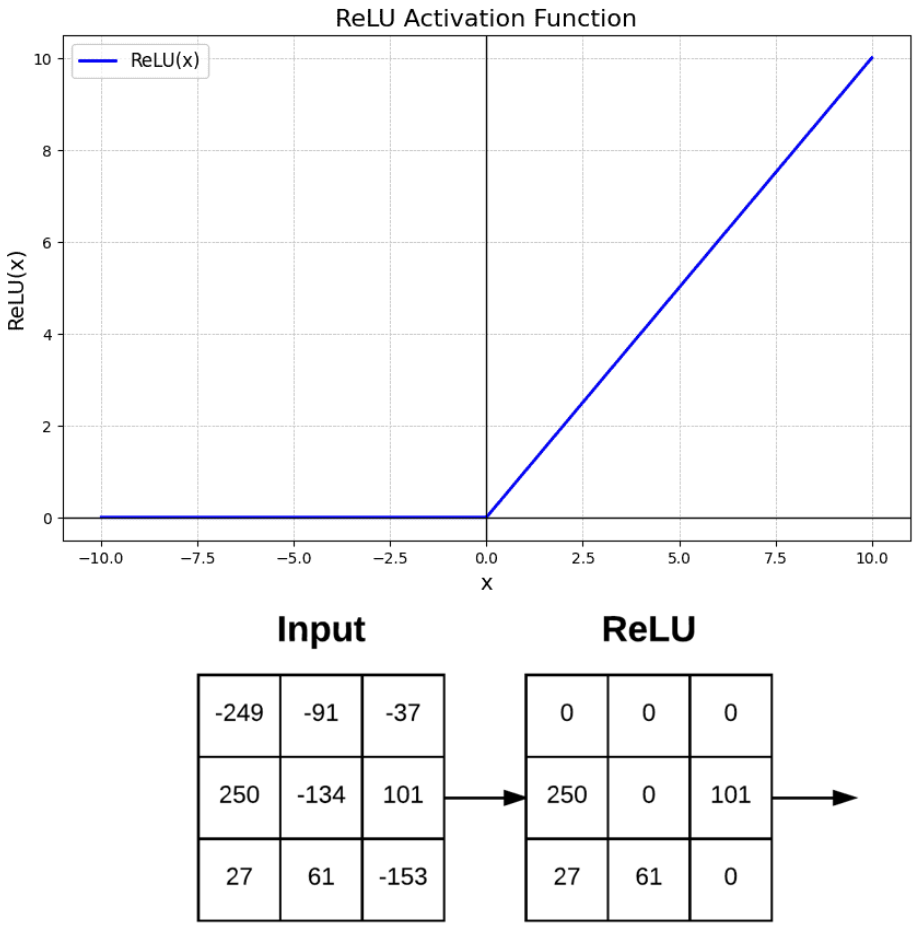


Figure 2.7: The upper plot displays the piecewise activation function defined as $\text{ReLU}(x) = \max(0, x)$, which maps all negative input values to zero while providing an identity mapping for positive inputs. The lower diagram demonstrates this operation applied to a sample 3×3 feature matrix. Each negative entry in the input matrix (such as -249 or -134) is thresholded to zero, whereas all positive values (such as 250 and 101) remain completely unaltered. This element-wise transformation introduces critical non-linearity into the neural network, promoting feature sparsity while helping to mitigate the vanishing gradient problem during training.utional Neural Network (CNN) architecture. n.d.

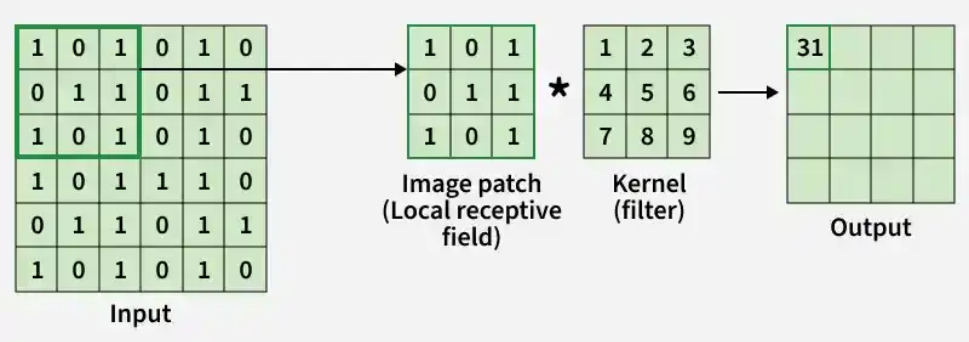


Figure 2.8: The schematic illustrates the extraction of a 3×3 sub-region (local receptive field) from a 6×6 input matrix. An element-wise multiplication is performed between the values of the image patch and the corresponding weights of the 3×3 kernel, followed by a summation of the products to yield a single scalar output. Specifically, the linear combination $(1 \cdot 1 + 0 \cdot 2 + 1 \cdot 3 + 0 \cdot 4 + 1 \cdot 5 + 1 \cdot 6 + 1 \cdot 7 + 0 \cdot 8 + 1 \cdot 9 = 31)$ produces the first entry of the resulting 4×4 feature map. As the filter slides across the input space with a specified stride, it systematically extracts local spatial features across the entire domain.

organization, or pathological patterns.

Although CNNs were originally developed for two-dimensional images, many medical imaging applications involve volumetric acquisitions such as Magnetic Resonance Imaging (MRI) and Computed Tomography (CT), where relevant information is distributed across three spatial dimensions. Processing each slice independently with a conventional 2D CNN neglects the spatial context between adjacent slices and therefore cannot directly model the three-dimensional structure of the acquired volume.

For this reason, many medical imaging applications employ three-dimensional Convolutional Neural Networks (3D CNNs), in which both convolutional kernels and feature maps are extended to three spatial dimensions. Instead of using a two-dimensional kernel of size $K_h \times K_w$, a 3D CNN operates with kernels of size $K_d \times K_h \times K_w$, allowing the network to simultaneously learn spatial features along all three dimensions of the volume.

The three-dimensional convolution can be expressed as

$$y(i,j,k) = b + \sum_{a=0}^{K_d-1} \sum_{m=0}^{K_h-1} \sum_{n=0}^{K_w-1} w(a,m,n),x(i+a,j+m,k+n), \tag{2.4}$$

where x denotes the input volume, w is the three-dimensional convolutional kernel, b is the learnable bias, and y is the resulting three-dimensional feature map. The additional kernel dimension allows the convolution to incorporate information from neighboring voxels along the third spatial axis, thereby preserving the volumetric context of the input.

By exploiting the complete volumetric context, 3D CNNs can learn spatial representations that account for the three-dimensional organization of anatomical structures. This makes them

particularly suitable for medical imaging applications such as organ and brain segmentation, lesion detection, image registration, and representation learning from volumetric MRI. However, this capability comes at a considerably higher computational and memory cost than two-dimensional convolutions. In particular, processing an additional spatial dimension increases the number of operations and the size of the intermediate feature maps, making 3D CNNs substantially more demanding in terms of computational resources.

2.2.2 ResNet

As mentioned earlier, Convolutional Neural Networks learn hierarchical feature representations. The first convolutional layers generally capture low-level features such as edges, corners and simple textures, while deeper layers progressively encode more complex and abstract representations, eventually describing semantic structures and object-level information [14]. Increasing the depth of a network therefore allows the model to learn increasingly sophisticated feature hierarchies, which are often necessary to solve complex visual recognition tasks.

This intuition was confirmed by several studies showing that deeper neural networks consistently outperform shallower ones when sufficient data and computational resources are available[15][16]. In principle, increasing the number of layers enlarges the representational capacity of the model, enabling it to approximate more complex functions.

Nevertheless, simply stacking additional layers does not necessarily lead to better performance. Early attempts to train very deep networks were severely affected by the vanishing and exploding gradient problem, where gradients become either extremely small or excessively large while being propagated backwards through the network [17]. This phenomenon makes optimization unstable and significantly slows down convergence. Several techniques were proposed to alleviate this issue, including carefully designed weight initialization strategies [17] and even intermediate layers of normalization [18], which greatly improved optimization stability.

Surprisingly, even after largely mitigating gradient instability, researchers observed another phenomenon. Beyond a certain depth, training accuracy stopped improving and often became significantly worse. Importantly, this degradation was not caused by overfitting, since the training error itself increased as additional layers were introduced [19]. This observation suggested that the optimization problem of very deep networks remained intrinsically difficult, even when gradients could successfully propagate through the architecture.

To address this limitation, He et al.[20] proposed the Deep Residual Learning Framework, commonly known as ResNet [19]. The key idea is remarkably simple: instead of directly learning the desired mapping $H(x)$, the network learns a residual function $F(x) = H(x) - x$, while the original input is propagated through an identity shortcut connection, also known as a skip connection. Similar shortcut mechanisms had previously appeared in Multi-Layer Perceptrons

and Highway Networks, but ResNet demonstrated that identity shortcuts could successfully enable the training of extremely deep convolutional architectures containing hundreds of layers [19].

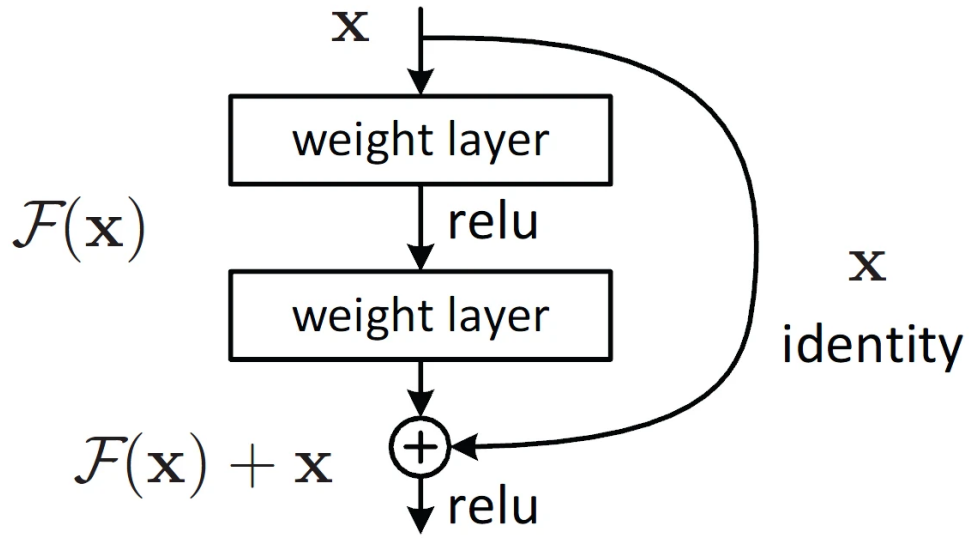


Figure 2.9: Residual building block architecture in Deep Residual Networks (ResNet). The schematic illustrates a basic residual learning unit where an input feature map **x** is transformed by stacked weight layers with an intermediate ReLU activation to fit a residual function $\mathcal{F}(\mathbf{x})$. A shortcut connection performs an identity mapping that bypasses the weight layers, adding **x** directly to the learned residual mapping to yield $\mathcal{F}(\mathbf{x}) + \mathbf{x}$. The combined output is then processed through a final ReLU activation function. This residual formulation facilitates direct gradient flow during backpropagation, effectively mitigating the vanishing gradient and degradation problems in deep neural networks.

More formally, let $H(x)$ denote the target mapping that should be approximated by a sequence of convolutional layers, where x is the input tensor. Instead of directly approximating $H(x)$, the residual block learns

$$F(x) = H(x) - x$$

Consequently, the output of the residual block becomes

$$y = F(x,Wi) + x$$

where $F(x,Wi)$ represents the residual mapping parameterized by the convolutional weights Wi, and y represents the final output of the residual block.

In the simplest case, the shortcut performs an identity mapping and introduces no additional parameters. However, element-wise addition requires both tensors to share identical spatial dimensions and channel size. Whenever this condition is not satisfied, a learnable linear projection

typically implemented as a 1×1 convolution (or 1×1×1 in the 3D case)is applied to the shortcut path to match the dimensions before the addition[20].

By facilitating gradient propagation through the identity connections, residual learning enables the successful optimization of substantially deeper architectures while preserving computational efficiency. For this reason, ResNet has become one of the most influential backbone architectures in modern computer vision and has inspired numerous subsequent architectures for both 2D and 3D medical image analysis

2.2.3 Feature Pyramidal Network

Another fundamental component employed in this project is the Feature Pyramid Network (FPN), a neural architecture specifically designed to improve multi-scale feature representation.

One of the major challenges in computer vision is the large variability in object size. The same object may occupy only a few pixels or almost the entire image depending on the acquisition conditions, making scale invariance a crucial property for visual recognition systems. Before the advent of deep learning, this problem was commonly addressed through image pyramids, where multiple resized versions of the same image were processed independently. Since a change in object scale corresponds to a shift across pyramid levels, image pyramids allow detectors to recognize objects over a broad range of sizes while using the same classifier at every scale [21].

Although highly effective, image pyramids are computationally expensive because every image scale must be processed independently by the complete recognition pipeline. The computational cost and memory consumption therefore grow almost linearly with the number of pyramid levels, making this approach unsuitable for modern deep neural networks and real-time applications [21][22].

Fortunately, Convolutional Neural Networks naturally produce a hierarchical feature representation. As information propagates through the network, spatial resolution progressively decreases while semantic abstraction increases. Early layers preserve fine spatial details but mainly encode low-level features such as edges and textures, whereas deeper layers capture high-level semantic concepts at the expense of localization accuracy. This hierarchical structure suggests that CNNs inherently contain a feature pyramid without requiring multiple forward passes, leading them to be the protagonists in computer vision for many years, outperforming FPNs [23][24].

However, this implicit pyramid presents an important limitation. Although deeper feature maps are semantically rich, their low spatial resolution makes them less suitable for detecting small structures. Conversely, shallower feature maps retain precise localization information but lack sufficient semantic content. Consequently, directly exploiting the CNN hierarchy does not

provide equally informative representations across all spatial scales. Despite the robustness of Deep CNN, pyramids are needed in order to grasp the most accurate results, the main reason for it is that it produces a multi-scale feature representation in which all levels are semantically strong, and not only just a few. However featurizing every level has an obvious limitation: inference time increases considerably [22] thus making this approach unsuitable for real applications. Due to this problematic image pyramids were only used at test time [21], leading to an inconsistency between train/test time inference.

An early attempt to overcome this limitation was the Single Shot MultiBox Detector (SSD) [25], which generated predictions from multiple convolutional layers of different resolutions. SSD efficiently reused intermediate CNN features, avoiding the high computational cost of traditional image pyramids. Nevertheless, because its pyramid was built only from deeper layers, the highest-resolution feature maps remained relatively weak from a semantic perspective, reducing the detector's performance by not properly using feature hierarchy.

To address these limitations, Lin et al. proposed the Feature Pyramid Network (FPN) [21], an architecture capable of constructing semantically strong feature maps at multiple resolutions from a single forward pass through the backbone network. Rather than processing multiple resized images, FPN enriches the hierarchical CNN features through three complementary components: a bottom-up pathway, a top-down pathway, and lateral connections.

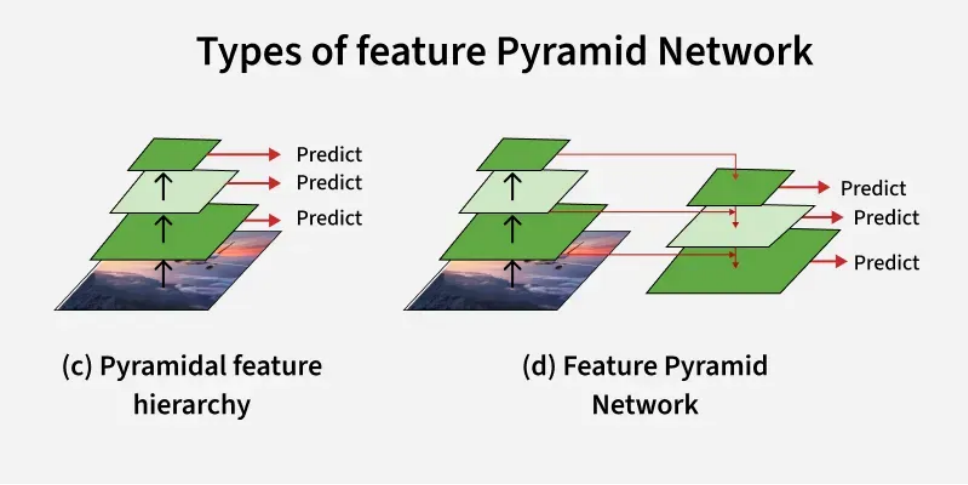


Figure 2.10: Comparison of feature pyramid architectural paradigms for multi-scale object recognition. Diagram (c) illustrates a conventional pyramidal feature hierarchy, where independent predictions are made at each spatial resolution of the bottom-up feature extractor without inter-level feature fusion. Diagram (d) depicts the Feature Pyramid Network (FPN) architecture, which introduces a top-down pathway connected to the bottom-up backbone via lateral connections. This top-down path combines semantically strong high-level features with high-resolution low-level spatial details, enabling highly accurate multi-scale predictions across all pyramid levels.

The bottom-up pathway corresponds to the standard forward propagation through the backbone network, typically a ConvNet or a ResNet. This architecture consists of ”stages”, in each stages many convolutions take place rpducing ano output that halves the spatial resolution and one that increasing the number of feature channels. The final output feature map of every stage is retained, the choice of taking only the final output of the stage is justified by the fact that deeper stages encode increasingly discriminative semantic information.

The top-down pathway starts from the deepest and most semantically meaningful feature map. This representation is progressively upsampled by a factor of two, restoring spatial resolution while preserving its semantic richness. However, upsampling alone cannot recover the fine localization details lost during downsampling.

For this reason, each upsampled feature map is merged with the corresponding feature map produced by the bottom-up pathway through a lateral connection. Before fusion, the bottom-up feature map is projected to the desired number of channels using a 1×1 convolution. The two representations are then combined through element-wise addition, effectively integrating the precise spatial localization of shallow layers with the rich semantic information of deeper layers. This fusion process is repeated recursively until all pyramid levels have been reconstructed.

Finally, a 3×3 convolution is typically applied to each fused feature map in order to reduce aliasing artifacts introduced by the upsampling operation and to generate the final pyramid representations.

The resulting pyramid consists of feature maps at multiple spatial resolutions, all characterized by strong semantic information. Compared with traditional image pyramids, FPN achieves comparable multi-scale representations while requiring only a single forward pass through the backbone network, dramatically reducing computational cost without sacrificing detection performance.[21]

In this work, a 3D adaptation of the Feature Pyramid Network is employed to aggregate volumetric features extracted by the ResNet encoder. Instead of processing two-dimensional feature maps, the network operates on three-dimensional MRI volumes, allowing anatomical information to be fused across multiple spatial resolutions while preserving the contextual relationships between neighboring slices. This multi-scale representation provides richer features for the subsequent representation learning stage.

2.2.4 Transformer

One of the fundamental challenges in sequence modelling is the ability to capture long-range dependencies between distant elements of the input. Ideally, a model should be capable of relating any two positions in a sequence regardless of their distance. Before the introduction of Transformers, this task was mainly addressed using Recurrent Neural Networks (RNNs), Long

Short-Term Memory networks (LSTMs) and Gated Recurrent Units (GRUs), which represented the state of the art for sequence transduction problems such as machine translation and language modelling [26].

These architectures process the input sequentially, updating an internal hidden state after reading each token. Although effective, this sequential computation prevents the parallel processing of sequence elements during training, resulting in long training times and limited computational efficiency. Furthermore, information must propagate through every intermediate hidden state before reaching distant positions, making long-range dependencies increasingly difficult to model despite the introduction of gating mechanisms in LSTMs and GRUs [26] 2.11.

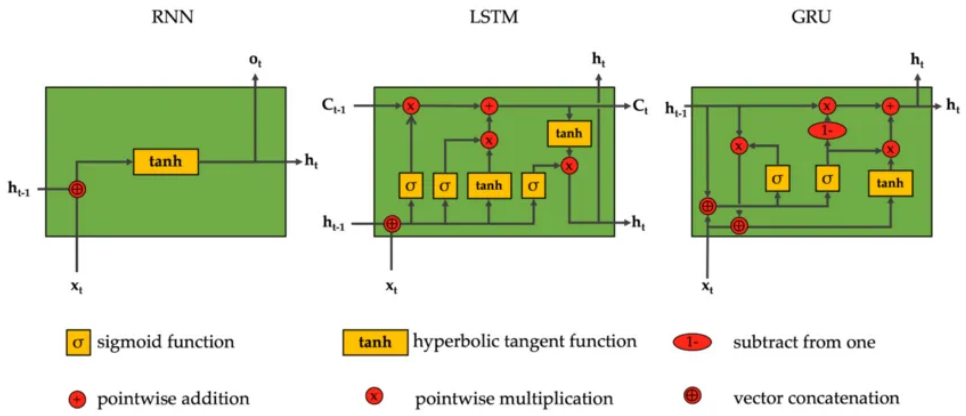


Figure 2.11: Architectural comparison of recurrent neural network (RNN) cell structures. The diagram illustrates the internal gating mechanisms and information flow for standard RNN, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) models. The basic RNN cell (left) employs a single hyperbolic tangent (tanh) layer to process concatenated input (x_t) and hidden state (h_{t-1}). The LSTM cell (middle) manages long-term memory via a separate cell state (C_t) regulated by forget, input, and output gates with sigmoid (σ) activations. The GRU cell (right) streamlines this process by merging the cell and hidden states into a unified hidden vector (h_t), using reset and update gates to control information flow with reduced parameter complexity.

To alleviate these limitations, attention mechanisms were introduced 2.12. Rather than compressing the entire sequence into a single hidden representation, attention allows the model to dynamically determine which parts of the input are the most relevant for computing the current output. In other words, every element of the sequence can selectively focus on the information carried by all the other elements, regardless of their relative position. Early attention mechanisms significantly improved Neural Machine Translation systems, although they were still employed together with recurrent architectures [27], [28].

A major breakthrough came with the work of Vaswani et al. [26], who proposed the Transformer architecture. Instead of relying on recurrence or convolutions, the Transformer is built

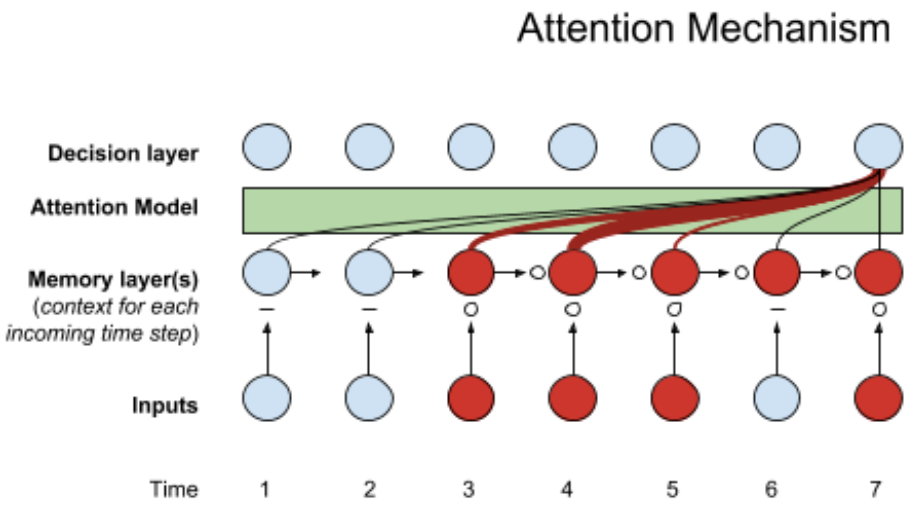


Figure 2.12: General conceptual framework of the attention mechanism in deep learning. The attention mechanism enables neural network models to dynamically focus on specific segments of an input sequence by assigning learnable scalar weights based on contextual relevance. Unlike conventional fixed-size representations, attention computes alignment scores between target query states and source key-value pairs to soft-select pertinent information regardless of temporal or spatial distance. This dynamic weighting resolves long-range dependency limitations, reduces information bottlenecks, and serves as the core foundational building block for modern Transformer-based architectures.

entirely upon attention mechanisms, enabling all sequence elements to be processed simultaneously. This design dramatically improves computational parallelism while allowing the model to capture global dependencies through a single attention operation.

Although previous architectures such as Extended Neural GPU [29], ByteNet [30] and ConvS2S [31] had already attempted to reduce sequential computation using convolutional operations, interactions between distant positions still required multiple computational steps. In contrast, self-attention allows every position to directly interact with every other position within the sequence, considerably shortening the information path between distant elements and facilitating the learning of long-range relationships [26].

The core component of the Transformer is the Self-Attention mechanism, an operation that computes a new representation for every token by considering the entire input sequence. Unlike recurrent architectures, where information flows sequentially, self-attention allows every token to simultaneously exchange information with all other tokens in a single operation.

The original Transformer consists of two main components: an Encoder and a Decoder. The encoder is responsible for generating contextual representations of the input sequence, whereas the decoder uses these representations to autoregressively generate the output sequence. Since this work focuses on Vision Transformers used as feature extractors rather than sequence gen-

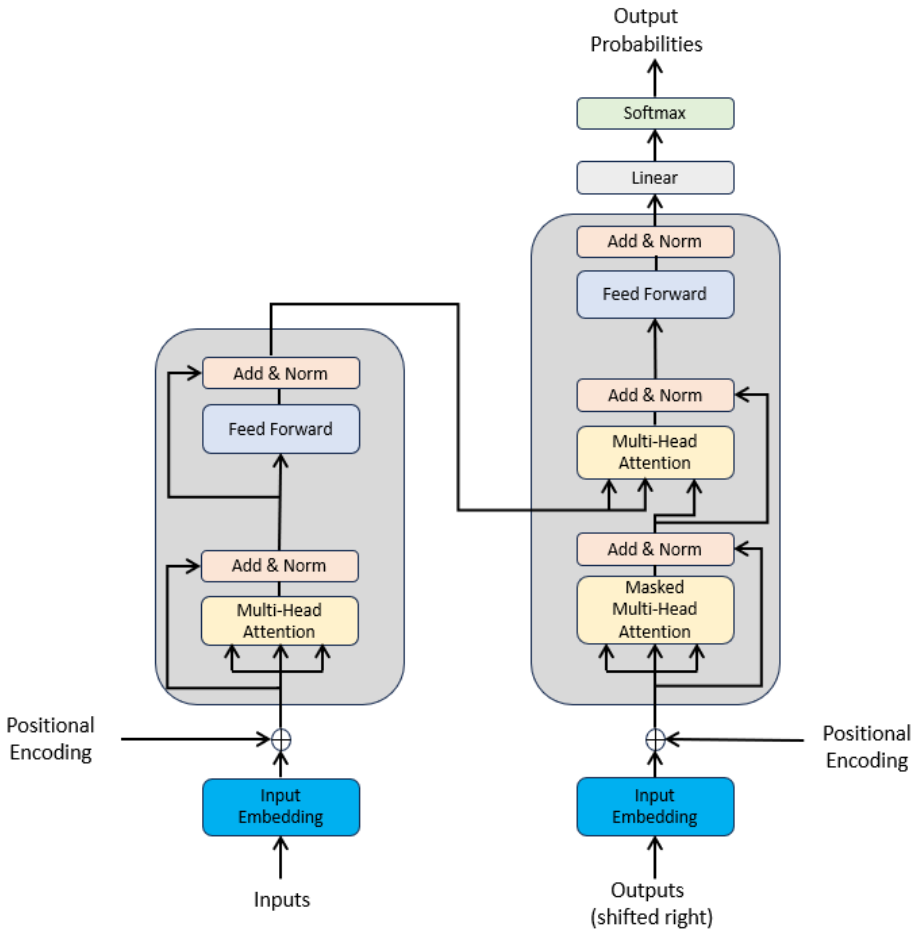


Figure 2.13: Complete encoder-decoder Transformer architecture. The model processes input and target sequences combined with positional encodings to inject token order information into recurrence-free representations. The encoder stack (left) processes the input sequence through multi-head self-attention and position-wise feed-forward networks, both wrapped with residual connections and layer normalization (Add & Norm). The decoder stack (right) employs masked multi-head attention to prevent causal information leakage from future tokens, followed by a cross-attention sub-layer that integrates the contextualized encoder outputs. Finally, a linear projection and Softmax function transform the decoder output into predicted token probability distributions.

erators, only the encoder architecture is directly relevant.

The encoder is composed of a stack of identical blocks. Each block contains two main sub-layers: a Multi-Head Self-Attention module followed by a position-wise Feed-Forward Network (FFN) 2.14. Around each sub-layer, a residual connection is employed, followed by Layer Normalization. Formally, the output of each sub-layer is computed as:

$$\text{LayerNorm}(x + \text{Sublayer}(x))$$

where the residual connection facilitates gradient propagation while Layer Normalization stabilizes training. Importantly, every encoder block preserves the dimensionality of its input, allowing multiple blocks to be stacked without altering the feature size.

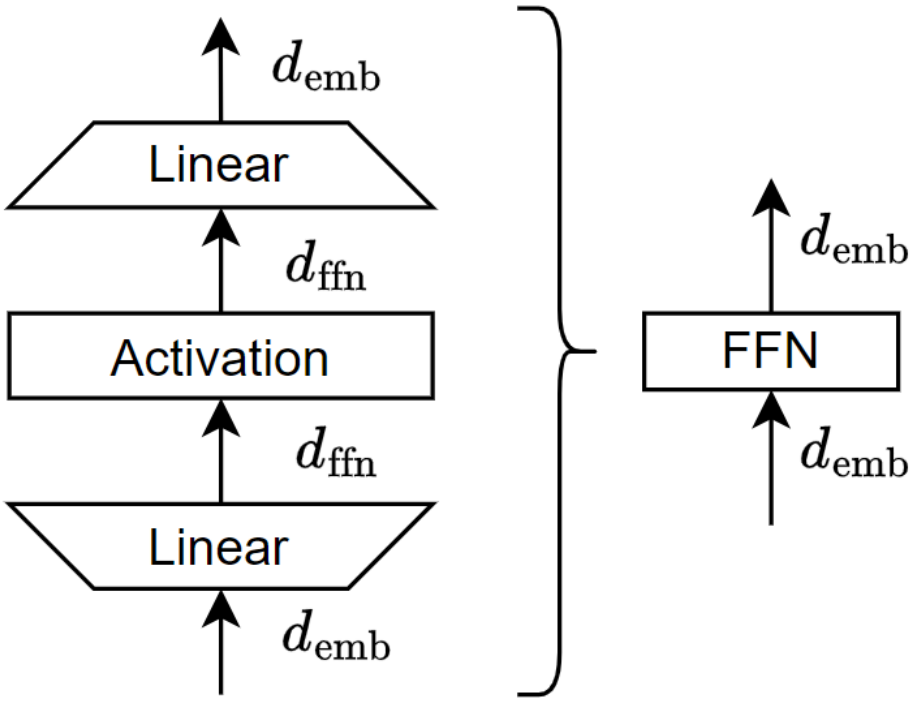


Figure 2.14: Structural breakdown and functional abstraction of the position-wise Feed-Forward Network (FFN) module in a Transformer. The left diagram illustrates the two-stage linear transformation sequence, where an input tensor of embedding dimension d_{emb} is first projected into an expanded inner layer of dimension d_{ffn} via a linear layer. An element-wise non-linear activation function (such as ReLU or GELU) is applied, followed by a second linear layer that projects the features back down to the original embedding dimension d_{emb}. The right diagram shows the compact functional block representation, emphasizing that the FFN operates independently on each token position while preserving input-output dimensionality.

The Feed-Forward Network consists of two fully connected linear transformations separated by a non-linear activation function (originally ReLU). Unlike the attention mechanism, which exchanges information between different sequence elements, the feed-forward network is inde-

pendently applied to every token, increasing the representational capacity of the model.

The key innovation of the Transformer lies in the attention mechanism itself. Attention operates on three different sets of vectors: Queries (Q), Keys (K) and Values (V). Intuitively, each query represents the information a token is searching for, each key describes the information offered by every token, while the values contain the actual features that will contribute to the final representation. The similarity between a query and every key determines the importance assigned to the corresponding value.

Transformer adopts the so-called Scaled Dot-Product Attention, whose output is defined as:

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

where d_k denotes the dimensionality of the keys. First, the dot product between every query and every key is computed, producing a similarity score between all pairs of tokens. These scores are then divided by $\sqrt{d_k}$ before applying the Softmax function. The scaling factor prevents the dot products from becoming excessively large as the feature dimensionality increases, thereby avoiding extremely peaked probability distributions and maintaining stable gradients during optimization [26].

Rather than computing attention only once, the Transformer employs Multi-Head Attention. Queries, Keys and Values are linearly projected multiple times into different lower-dimensional representation subspaces. Independent attention operations are then performed in parallel on each projection, producing multiple attention outputs that are finally concatenated and linearly transformed:

$$\text{Head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^O$$

Where W_i^Q, W_i^K, W_i^V, W^O are learned parametere projection matrixes.

This mechanism allows the model to simultaneously capture different types of relationships within the input. Different attention heads can specialize in different aspects of the data, such as local interactions, long-range dependencies or semantic similarities, resulting in richer and more expressive feature representations than those obtainable through a single attention operation.

Within the Transformer architecture, Multi-Head Attention is employed in three different ways. Encoder Self-Attention allows every input token to attend to all other tokens in the input sequence. Decoder Self-Attention performs the same operation while masking future positions, ensuring that each prediction only depends on previously generated outputs. Finally, Encoder-Decoder Attention enables the decoder to attend to the contextual representations generated by the encoder during sequence generation [26].

Since the Transformer contains neither recurrence nor convolution, it possesses no inher-

ent notion of token order. Consequently, positional information must be explicitly incorporated into the input embeddings. This is achieved through **Positional Encoding**, whose dimensionality matches that of the token embeddings (d_{model}). Positional encodings can either be learned during training or computed analytically using sine and cosine functions with different frequencies. By adding these positional vectors to the input embeddings, the model becomes capable of distinguishing between different token positions while preserving the parallel nature of self-attention [26].

Originally developed for Natural Language Processing, the Transformer has subsequently been extended to numerous other domains. In particular, the introduction of the Vision Transformer demonstrated that images can also be represented as sequences of tokens, enabling self-attention to model long-range spatial dependencies within visual data. This adaptation will be discussed in the following section.

2.2.5 Tokenization

The application of Transformer architectures to visual data requires converting the original spatial representation into a sequence of tokens. This process, commonly referred to as tokenization, determines how visual information is presented to the Transformer and therefore has a direct impact on both the computational cost and the quality of the learned representation.

As described in the previous section, self-attention establishes pairwise interactions between all input tokens. Given a sequence containing N tokens, the attention matrix has dimensions $N \times N$, resulting in a computational and memory complexity that grows quadratically with the number of tokens, $O(N^2)$ [1]. Consequently, the choice of tokenization strategy becomes particularly important for high-resolution visual data, where the number of possible tokens can rapidly become very large.

This problem becomes even more pronounced when dealing with volumetric medical images. Consider a cubic 3D image with spatial dimensions $N \times N \times N$. If every voxel were represented as an individual token, the resulting sequence would contain

$$N_{\text{tokens}} = N^3$$

tokens. Self-attention would consequently require an attention matrix containing

$$N^3 \times N^3 = N^6$$

pairwise interactions. Even for relatively moderate spatial dimensions, this rapidly becomes computationally prohibitive. For example, a volume of 128^3 voxels would result in $2{,}097{,}152$ tokens and an attention matrix containing more than 4×10^{12} pairwise elements. This illustrates

why directly applying standard self-attention to high-resolution volumetric data is generally impractical.

Patch-based Tokenization

A first solution to this problem was introduced in the context of the Vision Transformer (ViT) [32]. Instead of treating individual pixels as tokens, ViT divides the input image into a regular grid of non-overlapping patches. Each patch is then flattened and linearly projected into the Transformer embedding space, producing one token per patch.

For a 2D image of spatial dimensions $H \times W$, using patches of size $P \times P$, the number of tokens becomes

$$N_{\text{tokens}} = \frac{H}{P}\frac{W}{P}.$$

The reduction in sequence length is substantial because the number of tokens decreases quadratically with the patch size. For example, an image of 224×224 pixels divided into 16×16 patches produces only $14 \times 14 = 196$ tokens instead of $50{,}176$ individual pixels.

The same principle can be extended to volumetric data. For a volume of dimensions $H \times W \times D$, using cubic patches of size $P \times P \times P$, the resulting sequence contains

$$N_{\text{tokens}} = \frac{H}{P}\frac{W}{P}\frac{D}{P}$$

tokens. The corresponding self-attention matrix therefore contains

$$\left(\frac{HWD}{P^3}\right)^2$$

elements.

This strategy has also been adopted in medical imaging. In UNETR, for example, the input 3D medical image is divided into non-overlapping patches, which are then projected into the Transformer embedding space and processed as a sequence of tokens [33]. This formulation allows Transformers to model long-range dependencies within volumetric medical images while avoiding the prohibitive cost of voxel-level self-attention.

Patch-based tokenization therefore reduces the computational cost of dense visual representations, but introduces a fixed spatial decomposition determined before the image is processed. Each patch becomes a token regardless of whether it contains informative anatomical structures, homogeneous background, or redundant information. Consequently, the Transformer operates on a predefined set of spatial regions rather than dynamically selecting the most informative ones.

While this strategy has proven effective for natural images, particularly with large-scale pre-

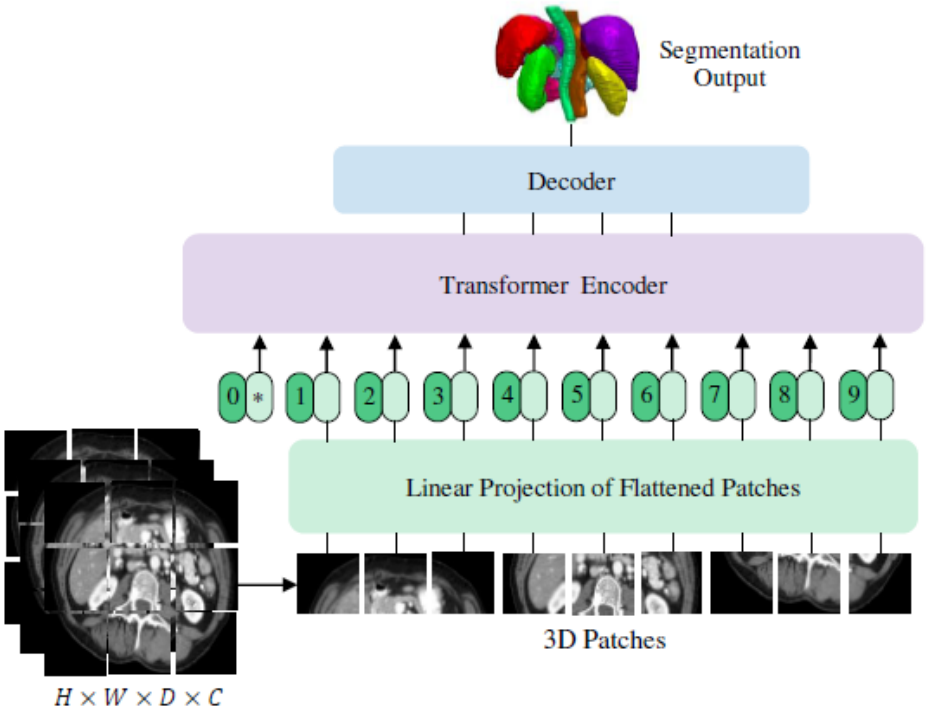


Figure 2.15: Patch-based tokenization and architectural pipeline for 3D medical image segmentation using a Vision Transformer. An input 3D volumetric scan with dimensions $H \times W \times D \times C$ is partitioned into a sequence of non-overlapping 3D spatial patches. These flattened patches are mapped via linear projection into 1D embedding vectors and augmented with positional encodings to form the input sequence of tokens. A Transformer Encoder processes the tokens to model global contextual relationships across spatial regions, after which a Decoder stack upsamples the encoded representations to produce the final 3D multi-organ segmentation mask.

training [34], high-dimensional medical images pose additional challenges. A 3D MRI volume may contain many voxels, while only a subset of its regions may provide highly informative anatomical or pathological features. Processing all regions equally can therefore result in substantial redundant computation.

Feature-based Tokenization

The approach adopted in this work starts from a different assumption. Rather than tokenizing the original MRI volume directly, tokenization is performed on feature maps generated by a preceding 3D convolutional neural network.

The motivation for this choice is that the convolutional encoder has already transformed the raw image into a higher-level latent representation. Instead of asking the Transformer to directly process raw intensity values, the tokenization mechanism operates on features that have already incorporated local spatial context and hierarchical information through the convolutional backbone.

Let the output of the convolutional feature extractor be represented as

$$X \in \mathbb{R}^{H \times W \times D \times C}$$

where H, W and D denote the three spatial dimensions of the feature map and C denotes the number of feature channels. The objective of the tokenization module is then to transform this dense volumetric representation into a much smaller set of informative vectors

$$Z \in \mathbb{R}^{S \times C},$$

where $S \ll HWD$.

This formulation separates two different operations. The convolutional backbone is responsible for extracting spatially structured visual representations, while the tokenization module determines which parts of this representation should be summarized into tokens before global reasoning is performed by the Transformer.

This distinction is particularly relevant for medical imaging, where local spatial structure is highly important but the volume may contain substantial redundancy. Rather than processing every spatial position with the quadratic self-attention mechanism, the objective is to construct a compact representation containing the most relevant information.

TokenLearner

A suitable mechanism for performing this adaptive tokenization is TokenLearner, introduced by Ryoo et al. [34]. TokenLearner was proposed as a visual representation learning module capable of extracting a small number of adaptively learned tokens from images and videos. Unlike conventional ViT tokenization, where tokens are generated according to a fixed spatial partition, TokenLearner learns to identify informative spatial regions directly from the input representation [34].

The fundamental idea is to learn a set of spatial attention maps, each identifying a potentially relevant region of the input. Given an input feature map (2D image for better understanding)

$$X \in \mathbb{R}^{H \times W \times C},$$

TokenLearner generates S spatial attention maps

$$\alpha_i(X) \in \mathbb{R}^{H \times W}, \qquad i = 1, \ldots, S.$$

Each attention map assigns a different importance weight to every spatial location. The corresponding token is then obtained by weighting the input feature map with the learned attention map and aggregating the result spatially:

$$z_i = \text{GAP}\left(\alpha_i(X) \odot X\right),$$

where $\odot$ denotes element-wise multiplication and GAP denotes global average pooling. The resulting token $z_i \in \mathbb{R}^C$ represents a learned summary of the spatial information selected by the corresponding attention map.

The complete token representation can therefore be expressed as

$$Z = [z_1, z_2, \ldots, z_S] \in \mathbb{R}^{S \times C}.$$

Each token does not necessarily correspond to a predefined spatial region. Instead, its spatial support is determined by the learned attention map. Consequently, different tokens can focus on different informative regions of the input representation, potentially overlapping when the input contains multiple relevant structures.

The attention maps themselves are learned functions of the input. In the original TokenLearner formulation, these functions can be implemented using lightweight convolutional layers, allowing the module to infer which spatial regions should receive greater importance [34]. The resulting tokens are therefore input-dependent, in contrast with the fixed tokens generated by conventional patch partitioning.

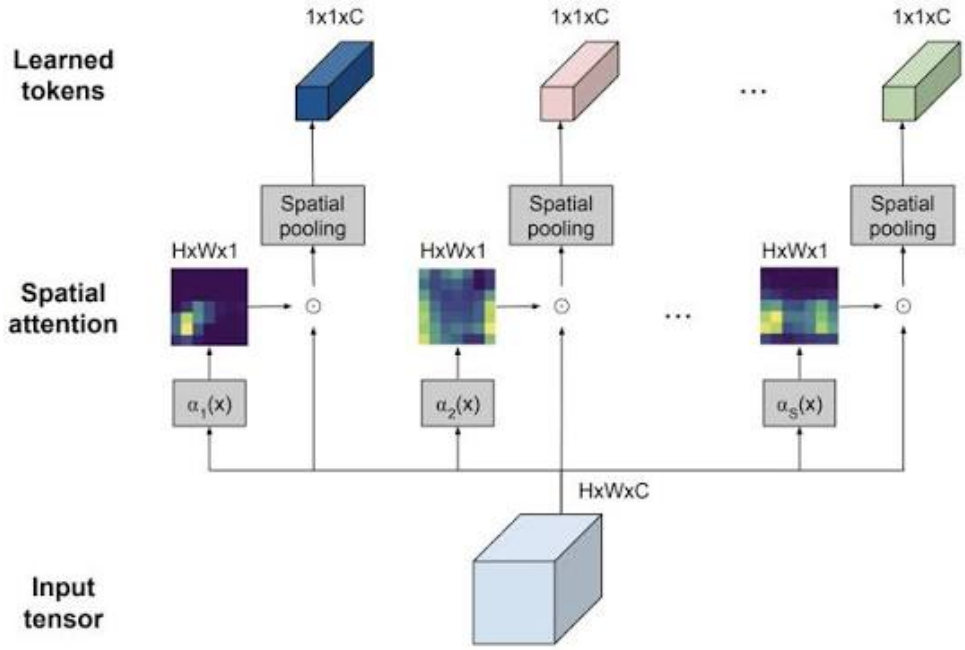


Figure 2.16: TokenLearner architecture for adaptive spatial tokenization. An input feature tensor of shape $H \times W \times C$ is processed by S learned spatial attention functions $\alpha_i(x)$, generating S corresponding weight maps of size $H \times W \times 1$. Each attention map is multiplied element-wise ($\odot$) with the input tensor to highlight salient spatial regions. Spatial pooling then aggregates each weighted feature map, condensing the high-dimensional input volume into a compact set of S learned vector tokens of dimension $1 \times 1 \times C$. This adaptive mechanism enables efficient sequence representation with significantly reduced token counts for downstream Transformer processing.

The distinction between the two approaches can be summarized as follows: In patch-based tokenization

$$X \rightarrow \text{fixed spatial patches} \rightarrow \text{tokens},$$

whereas TokenLearner performs

$$X \rightarrow \text{learned spatial attention maps} \rightarrow \text{weighted spatial representations} \rightarrow \text{tokens}.$$

The latter formulation allows the model to adapt the tokenization process to the content of each input.

Computational advantage

The main computational advantage of TokenLearner arises from the reduction in sequence length. If the original feature map contains N spatial positions, a conventional dense representation would provide approximately N tokens to the Transformer. TokenLearner instead produces only S tokens, where typically $S \ll N$. Since the computational cost of self-attention grows approximately quadratically with the number of tokens, the attention component changes from

$$O(N^2)$$

to

$$O(S^2).$$

The theoretical reduction in the number of pairwise interactions is therefore approximately

$$\left(\frac{N}{S}\right)^2.$$

For example, reducing a representation from 1024 tokens to 16 tokens changes the number of pairwise interactions from approximately one million to only 256. This is the fundamental reason why adaptive tokenization can substantially reduce the computational and memory requirements of subsequent Transformer layers.

Ryoo et al. demonstrated this principle by inserting TokenLearner into visual Transformer architectures and reducing the number of tokens processed by subsequent attention layers while maintaining competitive recognition performance [34]. In their experiments, TokenLearner could reduce representations containing hundreds or thousands of visual tokens to a small set of learned tokens, such as eight tokens per frame, substantially reducing the computational burden of the Transformer.

Importantly, the computational benefit does not come from simply discarding spatial infor-

mation. Instead, TokenLearner attempts to compress the representation according to the information content of the input, assigning greater importance to spatial regions that are considered useful for the downstream task.

TokenFuser

TokenLearner can also be combined with a complementary module called TokenFuser [34]. While TokenLearner transforms a dense spatial representation into a compact set of tokens, TokenFuser performs the inverse type of operation: it takes the processed tokens and remaps their information back into a spatial representation. Given Transformer output

$$Y \in \mathbb{R}^{S \times C},$$

TokenFuser first applies a learned transformation across the token dimension, allowing the model to learn relationships between the tokens themselves. The resulting representation is then projected back onto the original spatial domain using a learned spatial weighting mechanism. Finally, the reconstructed representation can be combined with the original feature map through a residual connection [34].

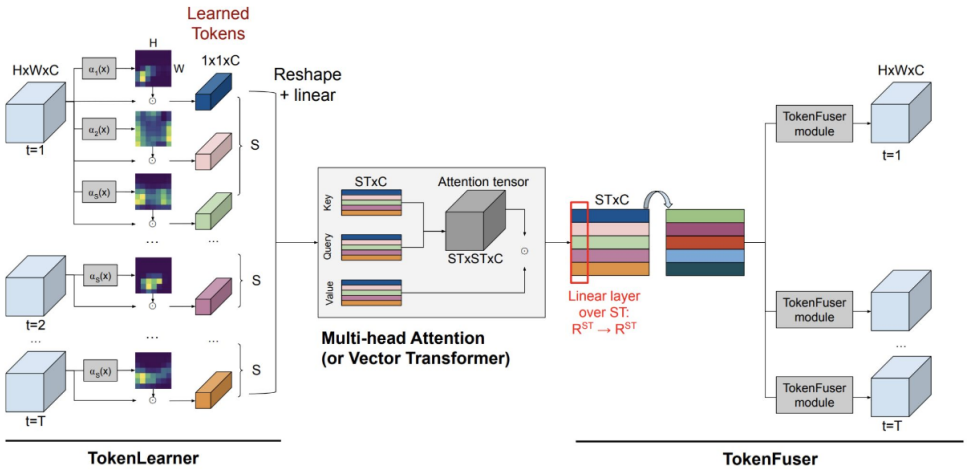


Figure 2.17: Overall architecture of the TokenLearner and TokenFuser framework for spatio-temporal feature interaction. Input tensors of dimensions $H \times W \times C$ across time steps $t = 1, \ldots, T$ are first tokenized by TokenLearner modules, which generate S spatial attention maps per frame to extract a compact set of $ST \times C$ learned tokens. These tokens are reshaped and processed through a Multi-Head Attention block (or Vector Transformer) alongside a linear projection over the temporal-spatial dimension ST to model cross-frame contextual relationships. Finally, the TokenFuser modules remap the contextualized token representations back into the original high-resolution feature spaces ($H \times W \times C$) for each corresponding frame, enabling efficient global token interaction combined with dense spatial feature reconstruction.

The TokenFuser is particularly useful when TokenLearner is inserted inside a larger visual

architecture rather than used only as a final compression stage. In this configuration, TokenLearner compresses the representation before Transformer processing, while TokenFuser restores the representation to its original spatial dimensions so that subsequent convolutional or token-learning operations can continue to operate on spatial feature maps.

Tokenization in the proposed architecture

The tokenization strategy adopted in this work follows the general principle of TokenLearner but operates on three-dimensional feature representations produced by a convolutional backbone.

Specifically, the MRI volume is first processed by a 3D convolutional architecture, which produces multi-scale volumetric feature maps. These representations contain information extracted from the original MRI while preserving their spatial organization. The resulting feature maps are then provided to a 3D adaptation of the TokenLearner module, which learns a set of spatial attention maps over the volumetric domain.

In this setting, the input can be represented as

$$X \in \mathbb{R}^{H \times W \times D \times C},$$

and the tokenization module produces

$$Z \in \mathbb{R}^{S \times C}.$$

The resulting S tokens are subsequently processed by the Transformer, allowing global relationships between the selected volumetric representations to be modeled through self-attention.

This design therefore combines two complementary mechanisms. The 3D convolutional backbone provides strong local and hierarchical spatial representations, while the adaptive tokenization mechanism reduces the dense volumetric representation to a compact set of informative tokens. The Transformer can consequently model long-range relationships without being required to perform self-attention over every spatial position of the original volume.

The distinction from conventional patch-based volumetric tokenization is therefore fundamental. Rather than defining the sequence before feature extraction through a fixed partition of the MRI volume, the proposed approach performs tokenization after feature extraction, allowing the representation itself to influence which spatial information is retained.

This strategy is particularly suitable for the objective of the present work, where the goal is not merely to classify or segment a predefined set of structures, but to learn a compact representation of heterogeneous MRI data. By allowing the tokenization process to depend on the learned feature representation, the model can potentially focus the subsequent global attention mechanism on the most informative anatomical patterns while avoiding unnecessary computa-

tion over large numbers of spatially redundant tokens.

2.3 Self-Supervised Learning

Self-Supervised Learning (SSL) is a learning paradigm that enables models to learn useful representations from unlabeled data by generating the labels from the data itself. Unlike approaches that rely on manually annotated datasets (e.g. Supervised Learning), SSL does not require a human to assign a label to every training sample. Instead, the data are transformed or structured in such a way that part of the available information can be used to predict another part. In this way the labels we'd need are automatically generated from the input data.

The main motivation behind SSL is the large amount of unlabeled data available in many application domains compared with the limited amount of manually annotated data. Creating annotations can be particularly time-consuming and expensive, especially when domain-specific knowledge is required. SSL addresses this problem by exploiting the information already contained in the data to construct an artificial learning objective, allowing models to be pretrained without requiring manual annotations.

The central component of SSL is the pretext task, an auxiliary task specifically designed to generate a supervision signal from unlabeled data. The model is trained to solve this task, and in doing so it is encouraged to learn meaningful features and representations of the input data. For example, an image can be partially hidden and the model can be trained to predict the missing content. Other possible pretext tasks include predicting the rotation applied to an image, restoring corrupted information, or determining the original arrangement of shuffled image patches. Since the correct answer can be automatically derived from the original data and the transformation applied to it, no manual annotation is required.

The objective of the pretext task is not necessarily to obtain a model that performs the auxiliary task itself. Rather, solving the pretext task forces the model to learn representations that capture relevant characteristics of the data. These learned representations can then be transferred to a different downstream task, which represents the actual application for which the model is intended.

The SSL pipeline can therefore be divided into two main stages. During the first stage, commonly referred to as self-supervised pre-training, the model learns representations from a large collection of unlabeled data by solving the chosen pretext task. During the second stage, the learned representations are used for a downstream task. This can be achieved either by fine-tuning the pre-trained model on data associated with the target task or by keeping the learned representations fixed and training a separate classifier on top of them.

A fundamental aspect of SSL is therefore the quality of the representations learned during

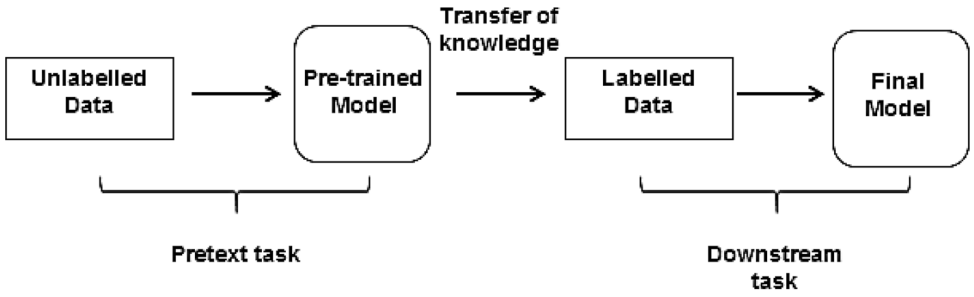


Figure 2.18: General workflow of the Self-Supervised Learning (SSL) paradigm. The pipeline is divided into two main stages: a pretext task and a downstream task. During the pretext phase, a model is pre-trained on abundant unlabelled data using self-supervised objectives to learn meaningful feature representations without manual annotations. Through knowledge transfer, these learned representations are subsequently adapted during the downstream phase, where fine-tuning is performed on a smaller set of labelled data to produce the final model optimized for specific target tasks.

pre-training. The pretext task must provide a sufficiently informative learning signal so that solving it requires the model to capture meaningful properties of the input data. The design of an appropriate pretext task is consequently one of the main challenges of self-supervised learning. If the task can be solved without learning information that is relevant to the target domain, the resulting representations may have limited usefulness for downstream applications.

Beyond pretext-task-based approaches, SSL methods can also be broadly divided into contrastive and non-contrastive approaches. Contrastive learning aims to learn a representation space in which related samples are brought closer together while unrelated samples are pushed apart. A common strategy is to generate different augmented versions of the same input and treat them as positive pairs, while other samples are used as negative examples. The model is then optimized to make the representations of positive pairs similar and those of negative pairs different.

Non-contrastive approaches, on the other hand, learn from positive pairs without explicitly using negative examples. These methods must employ additional mechanisms to prevent the model from converging to uninformative representations, a phenomenon known as representation collapse. Methods such as BYOL[35] and SimSiam[36] demonstrate that meaningful representations can be learned without explicitly contrasting positive samples against negative ones.

Overall, Self-Supervised Learning provides a framework for learning representations from large amounts of unlabeled data by constructing the supervision signal automatically. The learned representations can subsequently be transferred to downstream tasks, reducing the dependence on manually annotated data. This characteristic makes SSL particularly suitable for domains in which large datasets are available but obtaining reliable annotations is costly or dif-

ficult, including medical imaging.[37]

2.3.1 BYOL

Among self-supervised learning methods, Bootstrap Your Own Latent (BYOL) is a non-contrastive approach introduced by Grill et al. [35]. Its main objective is to learn useful representations from unlabeled data without relying on negative examples. This is particularly relevant in representation learning, where the goal of self-supervised pre-training is to obtain representations that can subsequently be exploited by downstream models.

The key idea behind BYOL is to make the representation of one augmented view of an input predictable from another view of the same input. Given an input sample x, two different transformations are applied to obtain two views v and v'. Although they differ at the input level, both views originate from the same sample and are therefore expected to retain common information. BYOL exploits this relationship to encourage consistent representations across different views.

To achieve this, BYOL employs two networks, referred to as the online and target networks. The online network consists of an encoder, a projector, and a predictor. The encoder transforms the input view into a representation, the projector maps it into a projection space used for the self-supervised objective, and the predictor attempts to predict the target projection produced by the other branch.

The target network contains an encoder and a projector, but no predictor. Its parameters are not directly optimized through back-propagation; instead, they are updated as an exponential moving average (EMA) of the online network parameters:

$$\xi \leftarrow \tau\xi + (1-\tau)\theta$$

where θ represents the online parameters, ξ the target parameters, and τ controls how rapidly the target network follows the online network. Consequently, the target network can be interpreted as a slowly evolving version of the online network, providing a more stable target during training.

The training objective is based on the similarity between the prediction generated by the online network and the projection generated by the target network. A stop-gradient operation is applied to the target branch, meaning that the loss is used to update the online network only. The process is then performed symmetrically by exchanging the two augmented views between the branches. In this way, the online network is repeatedly trained to predict the representation produced by the slowly updated target network.

An important characteristic of BYOL is that it performs this learning process without explicitly introducing negative pairs. In contrastive approaches, negative samples are used to prevent

the model from assigning similar representations to different inputs. BYOL instead relies on the combination of the asymmetric online branch, the predictor, and the slowly evolving target network. These components allow the model to bootstrap its representations while avoiding the collapse towards a constant representation, where all inputs would be mapped to the same feature vector.

The resulting representation is obtained from the encoder of the online network. Once pre-training is completed, the projector and predictor are discarded, while the encoder is retained as the learned feature extractor. This distinction between the representation produced by the encoder and the projection used exclusively for the self-supervised objective is particularly important: the projection space facilitates the training objective, whereas the encoder representation is the information that is ultimately transferred to subsequent tasks.

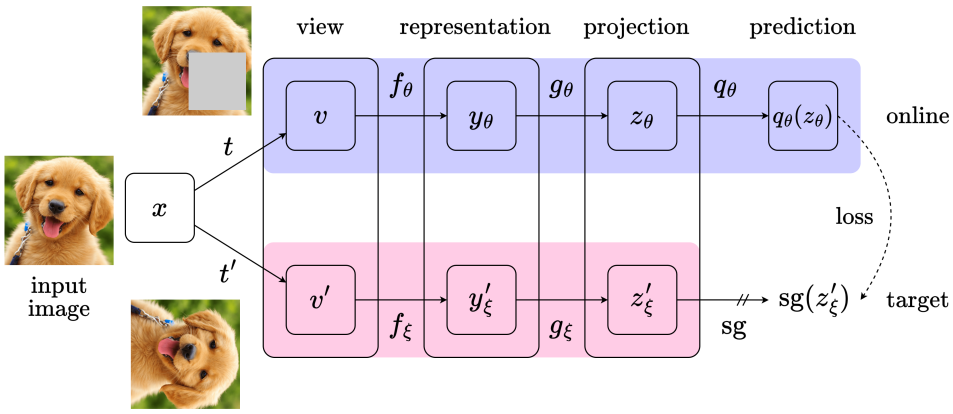


Figure 2.19: Architecture of the Bootstrap Your Own Latent (BYOL) self-supervised learning framework. An input image x is transformed into two augmented views, v and v', using randomized image transformations t and t'. The online network (top) processes view v through an encoder f_θ, projector g_θ, and predictor q_θ to yield a representation y_θ, projection z_θ, and final prediction $q_\theta(z_\theta)$. The target network (bottom) processes view v' through an encoder f_ξ and projector g_ξ to produce target representation z'_ξ. The loss is computed between the online prediction and the target output subjected to a stop-gradient operation $\mathrm{sg}(z'_\xi)$. The online parameters θ are updated via backpropagation, while the target parameters ξ are updated as an exponential moving average of θ, preventing representation collapse without relying on negative sample pairs.

This encoder-based formulation provides a connection with the architecture described in the following sections of this thesis. BYOL separates representation learning from the subsequent processing of learned features. In the present work, this concept is extended to three-dimensional medical images: the encoder transforms the volumetric MRI input into feature representations, which are subsequently processed into a more compact set of tokens for Transformer-based processing.

In the original work, BYOL was evaluated using convolutional residual networks as encoders and demonstrated that the learned representations could be transferred to several down-

stream tasks, including classification, semantic segmentation, object detection, and depth estimation. These results support the central premise of the method: a representation learned through self-supervised pre-training can capture information that remains useful beyond the specific objective used during pre-training.

For the purposes of this thesis, BYOL therefore represents an important step in the transition from self-supervised representation learning to the extraction and subsequent processing of learned features. While the original method operates on 2D natural images, its underlying principle—learning a representation by enforcing consistency between different views of the same input—can be adapted to other data modalities. In the proposed framework, this idea provides the foundation for learning representations from large collections of unlabeled MRI volumes, which can then be further transformed into tokens and processed by the Transformer-based architecture described in the following sections.

2.3.2 JEPA

The Joint-Embedding Predictive Architecture (JEPA) is a self-supervised learning framework proposed by Yann LeCun[38] as a non-generative approach to representation learning. Its central idea is to learn the relationship between two inputs by predicting the representation of one input from the representation of the other, rather than predicting the input itself. In this way, JEPA performs prediction in a learned representation space, allowing the model to focus on relevant information while avoiding the need to reproduce every detail contained in the original data.

A generic JEPA considers two related variables, x and y, which are processed by two encoders. The first encoder produces a representation s_x of x, while the second produces a representation s_y of y:

$$s_x = f_\theta(x), \qquad s_y = f_{\bar{\theta}}(y)$$

A predictor then receives s_x and attempts to predict the representation of y:

$$\hat{s}_y = g_\phi(s_x, z)$$

where z is an optional latent variable that can provide additional information when multiple outcomes are compatible with the same input. The prediction error is computed directly between the predicted and target representations:

$$E(x,y,z) = D(s_y, \hat{s}_y)$$

Thus, unlike generative approaches, the model does not attempt to reconstruct y from x. Instead, it learns whether the representation predicted from x is consistent with the representation

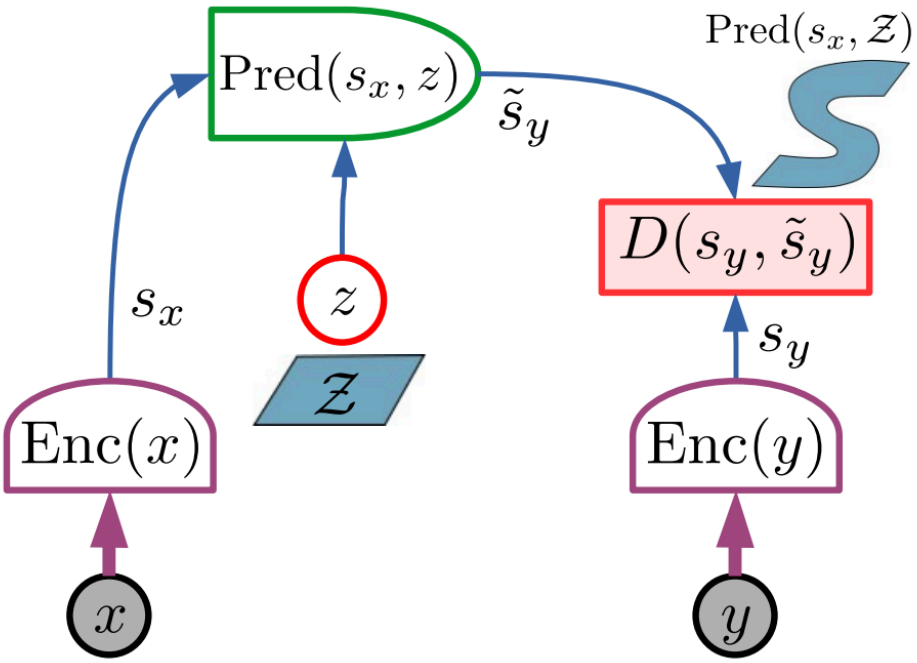


Figure 2.20: General architecture of the Joint-Embedding Predictive Architecture (JEPA) framework. An input context x and target input y are independently mapped into feature representations s_x and s_y using context and target encoders $\mathrm{Enc}(x)$ and $\mathrm{Enc}(y)$. A predictor module $\mathrm{Pred}(s_x, z)$ takes the context representation s_x together with a latent variable $z \in \mathcal{Z}$ specifying positional or structural transformations to estimate the target representation $\tilde{s}_y$. A distance metric $D(s_y, \tilde{s}_y)$ measures the divergence between the predicted representation and the actual target representation s_y. By making predictions directly within abstract latent space rather than generating low-level pixel details, JEPA captures high-level semantic structures while mitigating sensitivity to uninformative visual noise.

obtained from y.

This distinction between input-space prediction and representation-space prediction is one of the defining characteristics of JEPA. In a generative architecture, the model is required to reproduce the target signal itself. This can force the representation to preserve details that are present in the input but are difficult to predict or irrelevant to the task. JEPA takes a different approach: the target encoder is free to produce an abstract representation of y, potentially discarding information that is not useful for predicting the target representation. Consequently, the prediction objective can focus on the predictable aspects of the data rather than on their complete reconstruction.

This property is particularly relevant to the notion of representation learning. The objective of JEPA is not simply to minimize a prediction error, but to learn representations that contain sufficient information about their inputs while also being predictable from related representations. The original formulation describes this in terms of three main requirements: the representations s_x and s_y should remain informative about their respective inputs, while s_y should be predictable from s_x. A fourth criterion can be introduced to limit the information carried by an optional latent variable used by the predictor.

The requirement for informative representations is particularly important because a trivial solution exists: both encoders could map every input to the same constant representation. In that case, the predictor could perfectly predict the target representation, but the resulting representation would contain no useful information about the input. This phenomenon is known as representation collapse. JEPA therefore requires mechanisms that simultaneously encourage informative representations and accurate prediction.

An important aspect of the JEPA framework is that it can be trained using non-contrastive objectives. Rather than relying on negative examples to distinguish compatible from incompatible inputs, regularization can be used to prevent collapse while the prediction objective encourages the representations of related inputs to be predictable. The original proposal discusses criteria such as maximizing the information content of the representations and minimizing the prediction error, with methods such as VICReg[39] and Barlow Twins[40] presented as possible non-contrastive mechanisms for training a JEPA.

The use of a predictor also gives JEPA an important degree of flexibility. The two encoders are not required to have identical architectures or to share their parameters, since the variables x and y may potentially have different characteristics. More importantly, the predictor can model dependencies between the two representations that are not captured simply by forcing them to be identical. In the general formulation, a latent variable z can additionally represent information that is not determined by x, allowing the architecture to account for multiple possible target representations.

The ability to operate in representation space also gives JEPA a natural connection with hierarchical representation learning. The original proposal argues that different levels of abstraction can be useful for different prediction horizons: lower-level representations may retain more detailed information, whereas higher-level representations can discard unpredictable details and capture more abstract properties. Multiple JEPAs can consequently be arranged hierarchically, with one level operating on the representations produced by another.

This concept of progressively learning and predicting abstract representations is particularly relevant to the architecture developed in this thesis. The purpose of the self-supervised stage is not to reconstruct the original MRI volume, but to learn a representation of the volumetric data that retains meaningful information while discarding irrelevant details. The learned spatial features can subsequently be transformed into a more compact representation and provided to the Transformer-based components of the proposed architecture. JEPA therefore provides the conceptual basis for treating the self-supervised objective as a mechanism for learning useful latent features rather than reconstructing the input itself.

The general JEPA framework can be instantiated in different ways depending on the structure of x and y and on how their relationship is defined. For example, I-JEPA extends the framework to images by using a visible context region to predict the representations of masked target regions within the same image. This represents one possible application of the general JEPA principle, rather than defining JEPA itself. In the present work, a different formulation, C-JEPA, is adopted to perform self-supervised representation learning on three-dimensional medical imaging data.

C-JEPA extends the I-JEPA framework by addressing some of its limitations through the integration of the Variance-Invariance-Covariance Regularization (VICReg) strategy. In particular, the authors identify limitations in the ability of the Exponential Moving Average (EMA) mechanism of I-JEPA to prevent complete representation collapse, as well as limitations in accurately learning the mean of patch representations. VICReg is introduced to provide explicit regularization of the learned representations through three complementary components: invariance, variance, and covariance. The invariance term encourages consistent representations across different views of the same input, while the variance and covariance terms promote non-degenerate and non-redundant representations, helping to prevent representational collapse.[41]

By integrating VICReg with I-JEPA, C-JEPA combines the masked predictive learning paradigm of JEPA with regularization principles traditionally associated with contrastive self-supervised learning. This combination is intended to improve the stability of representation learning and to achieve faster convergence compared with the original I-JEPA formulation.

The ”C” in C-JEPA stands for Contrastive, reflecting this connection with contrastive self-supervised learning. However, C-JEPA does not employ a conventional contrastive setup based on explicitly defined positive and negative pairs. Instead, the contrastive aspect is introduced

through the VICReg regularization, which promotes invariance between corresponding representations while simultaneously constraining their variance and covariance. Therefore, C-JEPA can be regarded as contrastive in spirit rather than as a conventional contrastive learning architecture.[41]

In the present framework, this distinction is particularly relevant because the representations are organized as spatially structured tokens rather than as a single embedding for the entire image. The TokenLearner module produces multiple token representations from different spatial regions of the brain volume. The VICReg objective therefore contributes to learning consistent representations for corresponding predicted and target tokens while maintaining sufficient diversity across the latent representation space. The resulting framework combines JEPA's predictive objective with contrastive-style regularization, without requiring the explicit construction of negative pairs.

Therefore, JEPA can be viewed as the conceptual link between the general Self-Supervised Learning framework introduced previously and the specific self-supervised architecture employed in this thesis. Its key principle is the shift from predicting the input to predicting its representation. This allows the model to learn representations that emphasize information that is both informative and predictable, providing the basis for the subsequent feature extraction and tokenization stages described in the following sections.

2.3.3 VICReg

Variance-Invariance-Covariance Regularization (VICReg) is a self-supervised learning method designed to train joint-embedding architectures while avoiding the problem of representation collapse. In this type of approach, different views of the same input are processed by separate branches of an encoder architecture, and the resulting representations are encouraged to contain consistent information. The central challenge is to achieve this consistency without allowing the model to map all inputs to the same or to otherwise uninformative representations.

Representation collapse represents a trivial solution for a self-supervised objective based only on the similarity between representations. If the encoder produces the same constant vector for every input, two representations will always be identical, and a similarity-based objective can therefore be minimized without learning any meaningful information about the data. VICReg addresses this problem by explicitly regularizing the information content of the learned embeddings through three complementary terms: invariance, variance, and covariance.

The method can be understood within a Siamese architecture, in which two different views of the same input are processed by two branches. Given an input x, two transformations t and t' are sampled to produce:

$$x = t(i), \qquad x' = t'(i)$$

The two views are then passed through the encoder and subsequent projection layers, producing two embeddings z and z'. For a batch of N samples, these embeddings form two matrices

$$Z = [z_1, \ldots, z_N], \qquad Z' = [z'_1, \ldots, z'_N]$$

The objective of VICReg is to ensure that the two embeddings corresponding to the same input remain similar while simultaneously preserving sufficient variability and reducing redundancy within each embedding space.

Invariance

The invariance term encourages the representations produced from different views of the same input to remain similar. It is defined as the mean squared Euclidean distance between the corresponding embeddings:

$$\mathcal{L}_{inv} = \frac{1}{N} \sum_{i=1}^{N} \|z_i - z'_i\|_2^2.$$

Minimizing this term encourages the encoder to become invariant to the transformations used to generate the different views. Consequently, information that is preserved across the two views is encouraged to be represented consistently.

However, invariance alone is insufficient. As discussed above, a model could satisfy this objective by producing the same representation for every input. VICReg therefore introduces two additional regularization terms that explicitly constrain the structure of the embeddings.

Variance

The variance term prevents individual dimensions of the representation from becoming constant across the samples of a batch. For each dimension of the embedding, VICReg encourages its standard deviation to remain above a predefined threshold. The variance regularization for a batch Z can be written as:

$$\mathcal{L}_{var}(Z) = \frac{1}{d} \sum_{j=1}^{d} \max(0, \gamma - \text{std}(Z_{:,j}))$$

where γ represents the desired minimum standard deviation and a small constant is generally introduced for numerical stability.

The same regularization is applied independently to the two branches. In this way, the representations are encouraged to remain variable across the samples in the batch. If a particular embedding dimension were to become constant, its standard deviation would approach zero and the corresponding variance penalty would increase. The variance term therefore prevents

the simplest form of collapse by ensuring that the individual representation dimensions retain information.

Covariance

Maintaining variance in each individual dimension does not, however, guarantee that the representation contains non-redundant information. Different dimensions could still encode highly correlated or essentially identical information. VICReg therefore introduces a covariance regularization term.

For a batch of embeddings, the covariance matrix describes the statistical relationship between the different dimensions of the representation. VICReg penalizes the squared off-diagonal elements of this matrix:

$$\mathcal{L}_{cov}(Z) = \frac{1}{d} \sum_{i \neq j} C_{ij}^2$$

where C is the covariance matrix of the embeddings.

Minimizing this term encourages the off-diagonal elements of the covariance matrix to approach zero, thereby reducing redundancy between different embedding dimensions. The covariance term is inspired by the redundancy-reduction principle introduced by Barlow Twins and complements the variance term: while variance prevents individual dimensions from becoming constant, covariance discourages different dimensions from encoding the same information.

The variance and covariance terms are computed separately for each branch. This is important because preventing collapse must apply independently to both embedding spaces rather than only to their relationship.

VICReg's Objective

The three terms are combined into a single objective function:

$$\mathcal{L} = \lambda \mathcal{L}_{inv} + \mu(\mathcal{L}_{var}(Z) + \mathcal{L}_{var}(Z')) + \nu(\mathcal{L}_{cov}(Z) + \mathcal{L}_{cov}(Z'))$$

where λ, μ, and ν control the relative contribution of the three components.

The three terms therefore address complementary aspects of representation learning. Invariance encourages different views of the same input to produce similar representations, variance prevents individual dimensions from becoming constant, and covariance reduces redundancy between dimensions. The combination allows VICReg to learn consistent representations while explicitly maintaining their information content.

An important characteristic of VICReg is that these constraints are imposed directly through the loss function rather than through architectural mechanisms. In particular, VICReg does not

require negative samples, large memory banks, or specific normalization strategies to prevent collapse. It also does not require the two branches to share their weights or to use identical architectures. This flexibility makes the method suitable for joint-embedding architectures in which the two branches can process different types of inputs.

This property is particularly relevant when considering the JEPA framework introduced in the previous section. JEPA requires a mechanism capable of learning informative and predictable representations while avoiding collapse. VICReg can provide precisely this type of regularization: rather than relying on negative examples, the embedding spaces are constrained to remain informative through their variance and covariance structure, while the invariance term encourages the desired agreement between corresponding representations. The original JEPA proposal explicitly identifies VICReg as one possible non-contrastive objective for training joint-embedding architectures.[39]

In the context of this thesis, VICReg is therefore not considered as an independent architecture, but as the self-supervised objective used to train the joint-embedding system. The distinction between the architecture and the objective is important: JEPA defines the general mechanism through which one representation is predicted from another, while VICReg provides the regularization strategy that encourages the resulting embeddings to remain both consistent and informative.

This combination is particularly useful for the proposed C-JEPA framework. The different branches of the architecture can process the corresponding representations of the MRI data, while VICReg constrains their embeddings through invariance, variance, and covariance regularization. The resulting feature representations can then be used by the subsequent components of the architecture, eventually leading to the compact token representation processed by the Transformer. Thus, VICReg provides the training objective that supports the representation-learning stage, while the following sections describe how these learned features are structured and transformed into tokens for the subsequent Transformer-based processing.

2.4 SynthSeg

A major challenge in medical image analysis is the variability of MRI scans acquired using different acquisition protocols, scanners, contrasts, resolutions, and patient populations. Convolutional neural networks trained directly on real MRI datasets can be particularly sensitive to these variations, resulting in a substantial decrease in performance when the characteristics of the input data differ from those observed during training. SynthSeg was introduced to address this problem by developing a brain MRI segmentation network capable of operating on scans with different contrasts and resolutions without retraining or fine-tuning for each target domain.

The central idea behind SynthSeg is to avoid training the segmentation network on real MRI images altogether. Instead, the network is trained exclusively on synthetic images generated from anatomical segmentation maps. The parameters used to generate these images are extensively randomized, producing a very broad distribution of synthetic scans. This strategy, known as domain randomization, exposes the network to variations that are substantially larger than those normally encountered within a single MRI dataset. The objective is to prevent the network from associating anatomical structures with characteristics specific to a particular scanner, acquisition protocol, contrast, or resolution, thereby encouraging it to learn features that are more independent of the imaging domain.

This approach represents a particularly important aspect of SynthSeg. Conventional data augmentation generally introduces relatively small modifications around the distribution of the available training data. Domain randomization instead attempts to make the synthetic training distribution sufficiently broad that different real-world domains fall within it. Consequently, the network is not trained to solve a single domain-specific segmentation problem, but rather to identify anatomical structures across a wide range of possible image appearances.

2.4.1 Synthetic Data Generation

The training process starts from a set of anatomical label maps, rather than from MRI scans. Each label map describes the spatial arrangement of the anatomical structures that the network is expected to segment. At every training iteration, a label map is randomly selected and used as the basis for generating a new synthetic MRI volume. The corresponding segmentation is therefore known by construction, since it is the label map from which the synthetic image was generated.

This has two important consequences. First, the generated image and its ground-truth segmentation are perfectly aligned, eliminating the need for manual annotation of the synthetic scans. Second, the same anatomical label map can be used to generate a very large number of different images by changing the parameters of the generative process. The training set is therefore effectively generated on the fly, with a different combination of image characteristics sampled for each mini-batch.

The generative model is designed to reproduce a broad range of characteristics observed in real MRI acquisitions. Morphological variability is introduced by applying spatial transformations to the anatomical label maps, while image appearance is randomized independently. In particular, the model varies the simulated contrast, intensity distributions, acquisition resolution, orientation, and image artefacts. The official implementation exposes these parameters directly through the training generator, including affine transformations, nonlinear deformations, resolution randomization, and bias-field simulation.

Intensity variability is particularly important because the same anatomical structure can exhibit substantially different intensity values depending on the MRI sequence and acquisition parameters. SynthSeg therefore does not attempt to reproduce a single realistic MRI contrast. Instead, the intensity generation process is deliberately randomized. The resulting images can consequently look very different from one another while retaining the same underlying anatomical structure.

The model also explicitly accounts for resolution variability. Synthetic images can be generated with different voxel sizes and slice thicknesses, including anisotropic resolutions. The repository allows the resolution parameters to be randomized during training, exposing the network to a range of image qualities rather than a single fixed resolution.

This design is particularly relevant for heterogeneous MRI datasets, where volumes can differ considerably in spatial resolution and acquisition orientation. Instead of requiring the segmentation network to be retrained whenever a new resolution is encountered, SynthSeg incorporates resolution variability directly into the training distribution.

The same principle is applied to acquisition artefacts. For example, the generative process can introduce bias fields, which simulate low-frequency intensity inhomogeneities commonly observed in MRI. The implementation generates these fields from a lower-resolution random tensor that is resized to the image dimensions and transformed into a multiplicative intensity field.

More generally, the synthetic generation process introduces variability in morphology, contrast, resolution, orientation, noise and artefacts. The result is a continuously generated training distribution in which the network encounters a different combination of these factors at every iteration.

2.4.2 Segmentation Architecture

Although the main contribution of SynthSeg lies in its training strategy rather than in a novel segmentation architecture, the original model employs a three-dimensional U-Net. The choice of a U-Net is appropriate for volumetric medical image segmentation because it combines a contracting path, which progressively extracts higher-level features at lower spatial resolutions, with an expanding path that restores spatial resolution. Skip connections between the two paths allow spatial information from earlier layers to be propagated to the corresponding stages of the decoder.

The SynthSeg implementation uses five resolution levels. Each level contains two convolutional layers with $3 \times 3 \times 3$ kernels. Batch normalization is used between levels, while max pooling is employed along the contracting path and upsampling along the expanding path. The convolutional layers use Exponential Linear Unit (ELU) activations, with the final layer using

a softmax activation to produce the probability of each anatomical class.

The first convolutional layer produces 24 feature maps. The number of feature maps is then doubled after each downsampling operation and halved after each upsampling operation. Consequently, deeper levels operate on progressively more compact spatial representations while increasing the number of feature channels.

The resulting architecture can therefore be conceptually divided into two functions. The encoder progressively transforms the input volume into increasingly abstract feature representations, while the decoder uses these representations together with information propagated through the skip connections to reconstruct a spatially detailed segmentation. Although this encoder-decoder structure is primarily used by SynthSeg to perform segmentation, the feature representations generated by the encoder are also relevant to the present work, since the encoder weights are reused as the initialization of the 3D CNN employed in the proposed self-supervised framework.

2.4.3 Training Objective

The segmentation network is trained using a soft Dice loss, which measures the overlap between the predicted segmentation and the corresponding ground-truth label map. For each anatomical class, the network produces a soft probability map, which can be directly compared with the one-hot encoded target. The Dice-based objective encourages the predicted probability distribution to overlap with the target anatomical regions.

In the original implementation, the network is optimized using the Adam optimizer. The published model was trained with a learning rate of 10^{-4}, batch size of one, and 300,000 training steps, with training taking approximately seven days on an NVIDIA Quadro RTX 6000 GPU.

The publicly available implementation additionally performs an initial training stage using a weighted L_2 objective followed by training with the Dice metric. The corresponding training script exposes these stages through $wl2_epochs$ and $dice_epochs$, respectively.

The important aspect for the present work, however, is not the specific segmentation optimization procedure, but the representation learned by the convolutional encoder during this training. Since the network has been exposed to synthetic volumes with highly randomized imaging characteristics, the features learned by its convolutional layers are encouraged to represent anatomical structures rather than relying exclusively on domain-specific intensity patterns.

2.4.4 Inference on real MRI scans

At inference time, SynthSeg receives a real MRI scan, which may differ substantially from the synthetic images used during training. The input is resampled to a high-resolution isotropic

space, corresponding to 1 mm isotropic resolution in the distributed model, and its intensities are normalized. The network then produces a probability map for each anatomical label.

SynthSeg also employs test-time augmentation. In addition to segmenting the original scan, a left-right flipped version is processed by the network. The resulting prediction is transformed back to the original orientation and averaged with the prediction obtained from the original image. This provides a small improvement in segmentation robustness.

One of the notable properties of SynthSeg is its computational efficiency compared with traditional Bayesian segmentation approaches. The authors report an inference time of approximately ten seconds per scan on an NVIDIA Titan Xp GPU for the original model, including preprocessing, prediction, and postprocessing. The current repository similarly reports inference on the order of seconds on a GPU and approximately one minute on a CPU, depending on the configuration.

More importantly, however, SynthSeg maintains its segmentation capability across substantial variations in contrast and resolution without requiring domain-specific retraining. The original study evaluated the method across thousands of scans, multiple modalities and resolutions, and also demonstrated generalization beyond brain MRI.

2.4.5 SynthSeg+

The original SynthSeg framework was subsequently extended to SynthSeg+, which was designed to improve robustness on heterogeneous clinical scans and to provide additional anatomical and quality-control functionality. Rather than relying on a single segmentation network, SynthSeg+ employs a hierarchical architecture in which the segmentation problem is decomposed into multiple stages.

The first network, S_1, produces a coarse segmentation of four broad anatomical classes. A denoising network D then corrects possible topological or structural errors in this preliminary segmentation. The resulting information is passed to a second segmentation network, S_2, which produces the target anatomical segmentation. A third network, S_3, further parcellates the cortical region into finer anatomical structures. Finally, a regression network estimates quality-control scores for representative regions, allowing potentially faulty segmentations to be automatically identified.

The architecture of the segmenters remains based on the same general 3D U-Net structure used by SynthSeg, with five levels, $3 \times 3 \times 3$ convolutions, ELU activations, and skip connections. The denoiser uses a lighter architecture with one convolution per level and a constant number of 16 features, while the quality-control network follows the encoder structure with $5 \times 5 \times 5$ convolutions and a final global max-pooling operation.

SynthSeg+ therefore extends the original system in two directions. First, the hierarchical

segmentation architecture improves robustness to difficult clinical scans, including scans with low tissue contrast and low signal-to-noise ratio. Second, the system incorporates automated quality control, allowing segmentation failures to be detected without requiring manual inspection of every output. In addition to whole-brain segmentation, SynthSeg+ provides cortical parcellation and intracranial volume estimation.

The distinction between SynthSeg and SynthSeg+ is relevant to this work because the two models should not be considered interchangeable. The original SynthSeg model provides the encoder architecture and weights used for the feature-extraction component of the proposed framework, whereas SynthSeg+ represents a subsequent extension focused primarily on robustness and additional clinical analysis capabilities.

2.4.6 Role of SynthSeg in the proposed architecture

In this thesis, SynthSeg plays a role that extends beyond conventional preprocessing. Its use can be divided into two interconnected stages.

First, SynthSeg+ is employed to obtain anatomical segmentations of the MRI scans used in the preprocessing pipeline. These segmentations provide spatial information about the organization of the brain and can subsequently be used to define anatomical regions or guide the extraction of spatial features.

Second, and more importantly for the architecture presented in this work, the pre-trained weights of the SynthSeg convolutional encoder are used to initialize the 3D CNN feature extractor. This provides the proposed self-supervised model with an encoder that has already learned to identify meaningful three-dimensional anatomical patterns.

This initialization is particularly relevant because the objective of the subsequent C-JEPA framework is not to learn low-level image processing features from scratch. Instead, the CNN is expected to produce a structured representation of the MRI volume that can subsequently be transformed into a compact set of tokens. Starting from SynthSeg weights therefore provides a form of prior knowledge about brain anatomy before self-supervised training begins.

The relationship can be summarized as:

$$\text{MRI} \rightarrow \boxed{\text{SynthSeg-pretrained CNN}} \rightarrow \text{token extraction} \rightarrow \text{C-JEPA / Transformer processing.}$$

The role of SynthSeg is consequently different from that of the self-supervised methods discussed previously. BYOL, JEPA and VICReg define principles for learning representations from unlabeled data, whereas SynthSeg provides a pretrained convolutional representation of three-dimensional brain anatomy that is subsequently adapted within the proposed architecture. The combination allows the model to exploit both prior anatomical information and self-supervised

learning from the heterogeneous MRI dataset.

This also motivates the use of a 3D CNN rather than directly applying a conventional 2D feature extractor. MRI data are inherently volumetric, and the SynthSeg encoder has been trained to process three-dimensional anatomical structures. Its convolutional filters therefore provide an appropriate starting point for extracting spatial features from volumetric brain MRI. These feature maps can then be further processed by the tokenization mechanism described in the previous sections, reducing the spatial representation to a number of informative tokens suitable for the subsequent Transformer-based architecture.

In this sense, SynthSeg represents the connection between the medical-image-specific prior knowledge and the self-supervised representation-learning framework developed in this thesis. Its domain-randomized training procedure encourages the original encoder to learn features that are less dependent on MRI contrast and acquisition conditions, while the subsequent C-JEPA training further adapts these representations to the objectives of the proposed model.

Chapter 3

Objectives

The goal of this research is to develop a Self-Supervised Learning (SSL) architecture capable of learning meaningful and anatomically informative representations from heterogeneous three-dimensional Magnetic Resonance Imaging (MRI) volumes, without relying on large amounts of manually annotated data. The work is developed within the framework of the HEREDITARY project, where the ability to exploit large-scale and heterogeneous biomedical datasets is a fundamental requirement for the development of subsequent analytical and multimodal applications.

Brain MRI is particularly suitable for this purpose because large collections of scans are available, while detailed manual annotations remain relatively scarce due to the time and specialized expertise required to produce them. Furthermore, MRI volumes can vary substantially in terms of acquisition protocol, imaging contrast, spatial resolution, and other acquisition characteristics. The main objective is therefore to investigate whether self-supervised learning can exploit this variability to learn a representation that captures relevant anatomical information while being sufficiently general to be useful across different MRI scans.

To achieve this, the proposed architecture is designed around three main aspects: anatomically informed feature extraction, efficient representation of three-dimensional spatial information, and self-supervised representation learning.

The first objective is to develop a feature extraction stage capable of processing MRI volumes directly in three dimensions. Unlike two-dimensional approaches, a 3D convolutional network can exploit spatial relationships across all three axes of the volume, allowing anatomical structures to be represented together with their surrounding spatial context. The convolutional feature extractor is therefore responsible for transforming the original MRI volume into a hierarchy of spatial feature maps, progressively moving from low-level image characteristics towards more abstract anatomical features.

An important component of this stage is the use of pre-trained SynthSeg weights to initialize the 3D convolutional network. SynthSeg was originally trained to perform brain MRI seg-

mentation across heterogeneous imaging domains and consequently provides an encoder whose parameters have already been exposed to a wide range of anatomical and imaging variability. In this work, these weights are used not simply to perform segmentation, but as an anatomically informed initialization for the feature extraction stage. The objective is to exploit the prior knowledge acquired by SynthSeg while allowing the features to be further adapted through self-supervised training.

The second objective concerns the efficient representation of the spatial information produced by the convolutional encoder. Three-dimensional feature maps can contain a very large number of spatial elements, making their direct processing computationally expensive, particularly when they are subsequently provided to a Transformer. Processing every spatial location as an individual token would lead to unnecessarily long sequences and increase the computational cost of self-attention.

For this reason, the proposed architecture incorporates TokenLearner to transform the dense spatial feature maps into a smaller set of learned tokens. Rather than treating all spatial locations equally, TokenLearner learns to identify and aggregate the most informative spatial information into a compact representation. The objective is therefore to reduce the dimensionality of the feature maps while retaining the spatial information that is most relevant for the subsequent representation learning process. This component provides the connection between the convolutional feature extractor and the Transformer-based part of the architecture.

The third and central objective is to integrate these representations within a Joint-Embedding Predictive Architecture (JEPA) framework. The model is trained using a self-supervised objective, allowing it to learn from the MRI volumes without requiring explicit anatomical or clinical labels. Instead of directly optimizing the model for a predefined downstream task, the objective is to learn an embedding that captures information that can be exploited by subsequent applications.

The use of a JEPA-based framework is particularly suited to the objectives of this work because it allows representation learning to be formulated around relationships between different representations of the data rather than around manually provided targets. This is relevant in the context of HEREDITARY, where the availability of large amounts of heterogeneous medical data provides an opportunity for learning from unlabelled samples. The resulting representation is intended to retain information that is useful beyond the specific self-supervised objective used during training.

The architecture therefore combines the different components into a single representation learning pipeline:

$$\text{3D MRI} \rightarrow \text{SynthSeg-initialized 3D CNN} \rightarrow \text{TokenLearner} \rightarrow \text{Transformer} \rightarrow \text{JEPA}.$$

Each component addresses a specific limitation of applying self-supervised learning directly to heterogeneous three-dimensional MRI. The 3D CNN provides volumetric feature extraction and incorporates prior anatomical information through SynthSeg initialization. TokenLearner reduces the large spatial representation to a manageable number of informative tokens, while the Transformer provides the mechanism for processing relationships between these tokens. Finally, the JEPA framework supplies the self-supervised learning objective through which the representation is learned.

A further objective is to investigate the suitability of this approach for heterogeneous brain MRI data at scale. The model is not designed around a single MRI contrast or acquisition protocol, but is intended to operate on data exhibiting substantial variability. This is an important requirement for the intended application within HEREDITARY, where biomedical data originate from different sources and therefore cannot necessarily be assumed to follow a common acquisition distribution.

The ultimate aim is consequently not to build a model dedicated to a single segmentation, classification, or diagnostic task. Instead, this work aims to develop a general representation of three-dimensional brain MRI that can provide a foundation for downstream analyses. A useful learned representation could subsequently be adapted or combined with other modalities and task-specific models, reducing the need to repeatedly learn low-level and anatomical features for each individual application.

In summary, the objectives of this research are to:

Exploit unlabelled heterogeneous MRI data through a self-supervised learning framework, reducing the dependence on large-scale manual annotation. Extract three-dimensional anatomical features using a convolutional feature extractor initialized with pre-trained SynthSeg weights. Compress the resulting spatial representations through TokenLearner, retaining informative spatial information while producing a compact token sequence suitable for Transformer processing. Learn general-purpose MRI representations through a JEPA-based self-supervised framework, with the aim of producing embeddings that can subsequently support downstream applications.

By combining these objectives, the work aims to provide a representation learning framework that can contribute to the broader goals of the HEREDITARY project. In particular, the developed model is intended to serve as a component for the analysis of brain MRI within a larger multimodal biomedical framework, where learned imaging representations may eventually be integrated with information originating from other data modalities. The following sections describe the methodological choices adopted to achieve these objectives, including the architecture of the feature extractor, the tokenization mechanism, the Transformer, and the C-JEPA training framework.

Chapter 4

Methods

4.1 Dataset

Before describing the model architecture, it is necessary to introduce the data used for training and the procedure adopted to construct the training set. The characteristics of the available MRI data are particularly relevant in the context of this work, as the objective is to develop a self-supervised representation capable of handling heterogeneous three-dimensional brain volumes. Consequently, the selection of the training data was not considered a purely preliminary operation, but rather an important part of the experimental design.

All the data used in this work were obtained from FOMO300K, a large-scale heterogeneous dataset of brain Magnetic Resonance Imaging (MRI) scans containing both clinical and research-grade acquisitions. The dataset was specifically developed to support large-scale research on three-dimensional brain MRI and self-supervised learning, and includes scans acquired under substantially different imaging conditions. This heterogeneity is particularly relevant to the objectives of the present work, since the model is intended to learn representations that are not excessively dependent on a specific acquisition protocol or imaging domain.

FOMO300K contains a wide variety of MRI sequences and contrasts, including T1-weighted (T1w), MPRAGE, T2-weighted (T2w), T2*, FLAIR, susceptibility-weighted imaging (SWI), contrast-enhanced T1 (T1c), proton density (PD), diffusion-weighted imaging (DWI), apparent diffusion coefficient (ADC), and other acquisition types. These sequences differ substantially in terms of tissue contrast, acquisition parameters, spatial resolution, and the type of information they provide about brain anatomy. As a consequence, the dataset represents a considerably more heterogeneous environment than a collection obtained using a single standardized acquisition protocol.

The full version of FOMO300K comprises 306,207 scans acquired across 81,195 sessions from 58,741 subjects. Despite the considerable scale of the dataset, it was not feasible to use the

entire collection for the experiments presented in this thesis. The complete dataset occupies approximately 2.29 TB of storage, and processing such a large number of three-dimensional MRI volumes would have resulted in substantially longer training times and considerably higher computational and storage requirements. Since the primary objective of this work was to investigate the proposed architecture and its ability to learn representations from heterogeneous MRI data, a considerably smaller subset was selected for the initial experiments.

A total of 550 scans were therefore selected from the complete FOMO300K dataset. This subset was deliberately constructed rather than obtained through a purely random sampling procedure. The reason for this choice was to maximize the diversity of the data while limiting the possibility that the training process would become dominated by a particular population, acquisition site, scanner, or other characteristic of the source dataset.

The importance of this aspect became evident during the initial training experiments. In the first version of the dataset, the scans were selected approximately according to the proportions observed in the complete FOMO300K collection. Although this approach preserves the original distribution of the dataset, it also reproduces its most represented subgroups. In a heterogeneous medical imaging dataset, however, the most frequent subgroup is not necessarily representative of the variability that the model should learn to handle. A highly imbalanced distribution can introduce unwanted correlations between the image content and metadata-related characteristics such as acquisition site or scanner.

In particular, the initial subset contained a disproportionately large contribution from a dataset acquired in Nigeria. This over-representation resulted in a form of scanner or acquisition-site bias during the early training experiments. Rather than being forced to learn features that were broadly related to brain anatomy, the model could exploit characteristics associated with the dominant acquisition source. In other words, the model could partially solve the self-supervised objective by relying on systematic differences between groups of scans rather than learning representations that generalized across the heterogeneous data distribution.

Following these observations, the data selection strategy was revised. Instead of attempting to reproduce the global distribution of FOMO300K, the second selection procedure focused on maximizing diversity across subjects and acquisition sources. Scans were manually selected with the aim of avoiding the formation of a dominant group and ensuring that no single source contributed a disproportionate number of samples. This approach deliberately sacrifices the exact statistical proportions of the original dataset in favour of a more balanced representation of the different conditions encountered within it.

The resulting subset should therefore not be considered a statistically representative sample of FOMO300K in the strict sense. Rather, it represents a curated subset designed for experimental diversity.

A second modification to the selection procedure concerned the type of MRI acquisitions included in the training set. During the initial dataset preparation, Diffusion-Weighted Imaging (DWI) scans were also downloaded and included among the candidate volumes. These scans were subsequently excluded after inspection revealed that the corresponding data did not satisfy the three-dimensional volumetric assumptions required by the proposed architecture.

The distinction between two-dimensional and three-dimensional data is particularly important for the present model. The feature extraction stage is based on a 3D convolutional neural network, whose convolutional operations are designed to process spatial information along the three dimensions of a volumetric image. A volume with a substantially different dimensional structure cannot be treated as an ordinary 3D MRI without introducing an artificial representation of its missing spatial dimension.

Initially, this incompatibility was not immediately detected during data loading. A padding operation was applied during the early preprocessing stage to standardize the dimensions of the input data. Consequently, the DWI samples could pass through the initial data-processing pipeline without immediately generating a dimensionality error. Although this allowed the training procedure to start, the resulting inputs were not equivalent to genuine three-dimensional MRI volumes and were therefore not suitable for the intended feature extraction process.

The problem became apparent through the behavior of the training process and the quality of the resulting representations. The inclusion of these improperly structured inputs contributed to poor training performance, indicating that satisfying the required tensor dimensions through padding was not sufficient to make the data compatible with the assumptions of the model. The DWI acquisitions were consequently removed from the final training subset.

After all the modifications, the final dataset used for the experiments consisted of 550 selected MRI scans, chosen with the primary objective of providing a diverse collection of 3D brain volumes while avoiding excessive representation of individual acquisition sources.

4.2 MRI Segmentation and Cortical Parcellation

After creating the final data pool of 550 subjects from the FOMO300K dataset, the next step was to perform anatomical segmentation in order to obtain a label map for each MRI volume. The label map assigns an anatomical label to each voxel, indicating the brain region or Region of Interest (ROI) to which it belongs. This information is subsequently exploited by the proposed pipeline to introduce anatomical information into the processing of the MRI volumes.

Although the resulting label maps contain anatomical labels, their use does not directly transform the proposed learning framework into a supervised learning approach. In particular, the labels are not used as target classes for the representation learned by the C-JEPA framework.

Instead, they provide anatomical information that is used by subsequent processing stages, while the representation itself is learned through the self-supervised objectives described in the following sections.

The generation of the label maps was performed using SynthSeg+, which provides the required segmentation and cortical parcellation functionality. This allowed the anatomical processing stage to be integrated directly into the existing preprocessing pipeline without requiring the development of a separate segmentation method.

The implemented procedure first generates a list containing all raw MRI files available in the selected dataset. For each subject, the output directory is then inspected to determine whether both the corresponding segmentation and quality control (QC) files are already present. If either of these files is missing, the corresponding subject is added to the list of volumes that need to be processed. This mechanism avoids unnecessarily repeating segmentation for subjects that have already been processed and allows the pipeline to be safely resumed after an interruption.

Once the list of subjects requiring processing has been generated, the corresponding input paths are collected into a single batch. This follows the procedure suggested by the SynthSeg repository and allows multiple volumes to be submitted to the prediction script in a single execution, improving the efficiency of the preprocessing stage.

The segmentation is performed by invoking the SynthSeg prediction script through the following command:

```
cmd = [
    "python", str(SYNTHSEG_SCRIPT),
    "--i", str(input_txt_path),
    "--o", str(output_txt_path),
    "--qc", str(qc_txt_path),
    "--parc",
    "--robust",
    "--crop", "192"
]
```

The first argument `python`, specifies that the prediction script is executed using the Python interpreter. The variable `SYNTHSEG_SCRIPT` contains the path to `SynthSeg_predict.py`, the prediction script provided by SynthSeg for performing inference with the trained model.

The `"--i"` argument specifies the input list containing the MRI volumes to be processed, while `"--o"` specifies the output location where the resulting segmentation files are stored. The `"--qc"` argument specifies that we're interested in calculating the qc scores and save them, thus they will be saved in the output folder together with the segmentation masks.

The `"--parc"` option enables the cortical parcellation functionality of SynthSeg+. In addition to the anatomical segmentation, this produces a more detailed parcellation of the cerebral cortex, providing a larger set of anatomical regions that can subsequently be used as ROIs within the proposed pipeline.

The `"--robust"` option enables the robust processing mode of SynthSeg+. This option was retained throughout the preprocessing procedure because the anatomical parcellation represents an important source of information for the subsequent stages of the proposed architecture. Although robust processing increases the computational cost and inference time, it was therefore preferred in order to obtain more reliable anatomical segmentations across the heterogeneous MRI data used in this study.

Finally, the command includes the `"--crop 192"` option. This parameter was introduced because processing the complete input volumes at their original dimensions exceeded the available computational resources in the initial configuration. The option limits the spatial extent processed by SynthSeg to a volume of 192^3 voxels, reducing memory consumption during inference. The selected crop size was considered sufficient for the dataset used in this work, as the brain was contained within the resulting spatial extent.

The output of this procedure consists of an anatomical label map and its associated QC information for each processed MRI volume. The label maps are subsequently used to provide anatomical structure to the representation-learning pipeline, particularly during the definition of the ROI-based masking strategy described in the following sections.

4.3 The Pipeline

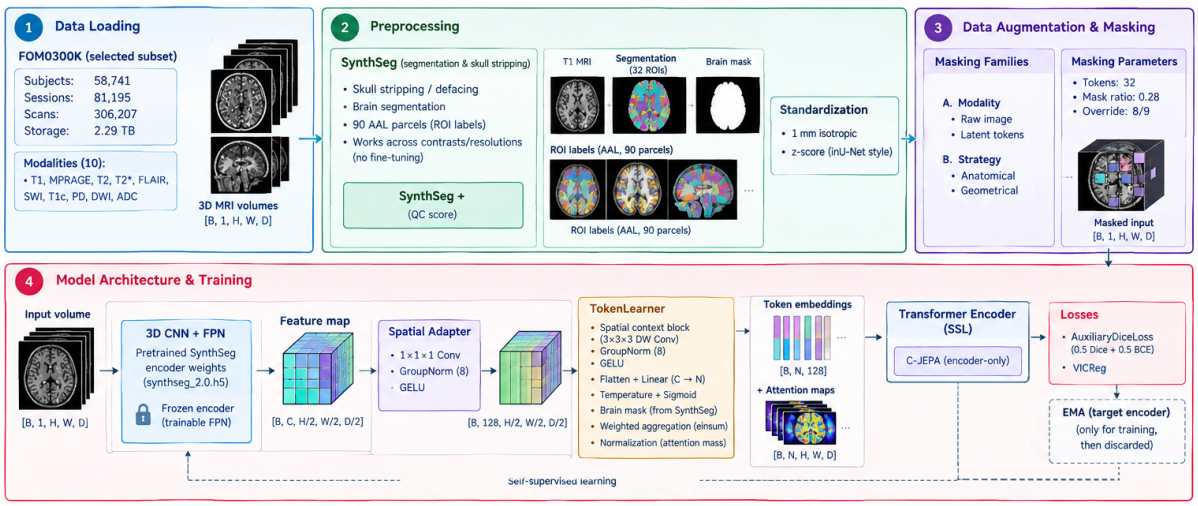


Figure 4.1: Visualization of a summary of the pipeline of the model

4.4 CNN + FPN

Once all subjects had been downloaded, segmented, and associated with their corresponding quality control (QC) scores, the dataset was ready to be processed by the proposed model.

The first stage of the processing pipeline is the CNN encoder. As discussed in the Background, the CNN represents a fundamental component of the C-JEPA architecture, as it processes the input MRI volumes and extracts spatial features that are subsequently converted into tokens and provided to the Transformer.

Two alternative backbone architectures were initially considered during the development of the project: a Recurrent Neural Network (RNN)-based architecture and a SynthSeg-based 3D CNN combined with a Feature Pyramid Network (FPN). The present work focuses exclusively on the latter solution, and all experiments and results reported in this thesis use the SynthSeg-based 3D CNNFPN backbone. This architectural choice also allowed the two studies conducted within the project to investigate complementary backbone configurations.

4.4.1 CNN Encoder

The first step in the development of the CNN-based backbone was the analysis of the original SynthSeg architecture. An ad hoc function was implemented to inspect the network structure and determine both the number of downsampling levels and the spatial and channel dimensions of the corresponding feature maps.

Starting from an input volume with dimensions $128 \times 128 \times 128$, the encoder produced the following feature maps:

Table 4.1: Feature map dimensions produced by the SynthSeg-based CNN encoder.

Level	Channels	Spatial dimensions
Input	1	$128 \times 128 \times 128$
0	24	$64 \times 64 \times 64$
1	48	$32 \times 32 \times 32$
2	96	$16 \times 16 \times 16$
3	192	$8 \times 8 \times 8$
4	384	$8 \times 8 \times 8$

The spatial dimensions reported for the encoder levels correspond to the feature maps after the associated pooling operation. Based on these observations and on the information available from the original SynthSeg architecture, the fundamental convolutional block was reconstructed as follows:

```
class ConvBlock(nn.Module):
```

```
    def init(self, in_c, out_c):
        super().init()
        self.conv1 = nn.Conv3d(in_c, out_c, kernel_size=3, padding=1)
        self.conv2 = nn.Conv3d(out_c, out_c, kernel_size=3, padding=1)
        self.bn = nn.BatchNorm3d(out_c)

    def forward(self, x):
        x = F.elu(self.conv1(x))
        x = F.elu(self.conv2(x))
        x = self.bn(x)
        return x
```

where in_c denotes the number of input channels and out_c denotes the number of output channels. The block consists of two consecutive three-dimensional convolutional layers. The first convolution transforms the input feature representation from in_c to out_c channels, while the second convolution operates on the resulting representation while preserving the number of channels.

Both convolutional operations use a kernel size of $3 \times 3 \times 3$ with padding equal to one. The padding preserves the spatial dimensions of the feature maps within the convolutional block. Each convolution is followed by an Exponential Linear Unit (ELU) activation function, introducing the non-linearity required to learn complex feature representations. Finally, a three-dimensional Batch Normalization layer is applied to the output of the second convolution.

The convolutional block described above defines the internal structure of each encoder stage, but does not itself perform spatial downsampling. The complete encoder is obtained by arranging multiple convolutional blocks and inserting a $3 \times 3 \times 3$ max-pooling operation between consecutive blocks:

```
class SynthSegEncoder(nn.Module):
    def init(self, channels):
        super().init()
        self.blocks = nn.ModuleList()
        self.pools = nn.ModuleList()
        for i, (in_c, out_c) in enumerate(channels):
            self.blocks.append(ConvBlock(in_c, out_c))
            if i < len(channels) - 1:
                self.pools.append(nn.MaxPool3d(2))
```

```
    def forward(self, x):
        feats = []
        for i, block in enumerate(self.blocks):
            x = block(x)
            feats.append(x)
            if i < len(self.pools):
                x = self.pools[i](x)
        return feats
```

The encoder iteratively applies each convolutional block to the input representation and stores the resulting feature map in the `feats` list. For all but the final block, a $2 \times 2 \times 2$ max-pooling operation is subsequently applied before the output is passed to the next stage. Consequently, the spatial resolution is progressively reduced while the number of feature channels increases.

An important aspect of this implementation is that the feature maps are stored before pooling. This allows the outputs of different encoder levels to be retained and subsequently used by the FPN through lateral connections. In this way, both high-resolution and deep, semantically richer representations can be exploited by the following stage of the architecture.

4.4.2 Feature Pyramid Network

The CNN encoder is followed by a Feature Pyramid Network (FPN), whose purpose in the proposed architecture is to combine feature maps produced at different levels of the encoder and progressively reconstruct a higher-resolution representation.

The FPN used in this work is defined as follows:

```
class FPN3D(nn.Module):
    def init(self, encoder_channels, out_channels=128, stop_idx=1):
        super().init()
        self.stop_idx = stop_idx
        self.lats = nn.ModuleList([
        nn.Conv3d(c, out_channels, kernel_size=1)
        for c in encoder_channels[stop_idx:][::-1]
        ])
        self.smooth = nn.Conv3d(
        out_channels, out_channels,
        kernel_size=3, padding=1
```

```
        )

    def forward(self, feats):
        target_feats = feats[self.stop_idx:][::-1]
        x = self.lats[0](target_feats[0])

        for i in range(1, len(self.lats)):
            x = F.interpolate(
                x,
                scale_factor=2,
                mode='trilinear',
                align_corners=False
            )
            x = x + self.lats[i](target_feats[i])

        return self.smooth(x)
```

Each selected feature map is first passed through a $1 \times 1 \times 1$ convolution, referred to as a lateral projection, which maps its original number of channels to a common dimensionality of 128 channels.

The deepest feature representation is used as the initial FPN representation. At each subsequent level, the current representation is upsampled by a factor of two along each spatial dimension using trilinear interpolation. If the current representation has dimensions

$$[B, C, D, H, W], \tag{4.1}$$

the interpolation produces a representation with dimensions

$$[B, C, 2D, 2H, 2W]. \tag{4.2}$$

The upsampled representation is then summed element-wise with the corresponding lateral feature map from the encoder. This fusion mechanism allows information from deeper, lower-resolution representations to be combined with features extracted at higher spatial resolutions. The process is repeated until the selected encoder levels have been incorporated.

Finally, the resulting representation is processed by a $3 \times 3 \times 3$ convolution with 128 output channels. This final convolution acts as a smoothing and refinement operation on the fused feature representation before it is passed to the subsequent component of the architecture.

The reconstruction of the output is stopped at the level indicated by stop_idx. In this case, being equal to 1, the final output will be at dimension $64 \times 64 \times 64$ while, without this index (or by putting it equal to 0), the final output would be of dimension $128 \times 128 \times 128$. This choice was more a necessity dictated by the limited computational capabilities of the hardware we were working with: an output at full dimension would bring the model Out Of Memory thus generating a critical error.

Overall, the CNN encoder and FPN provide a hierarchical three-dimensional representation of the input MRI volume. The CNN progressively transforms the original volume into feature maps with increasing channel depth and decreasing spatial resolution, while the FPN combines information from different levels and restores a higher-resolution representation. This output constitutes the spatial feature representation subsequently processed by the TokenLearner component of the proposed C-JEPA architecture.

4.5 TokenLearner

Once the 3D features have been extracted by the 3D CNN and FPN, the next stage of the proposed architecture is the TokenLearner. As introduced in the Background, its purpose is to transform the dense spatial feature representation into a smaller set of learned tokens. This reduces the number of elements subsequently processed by the Transformer while allowing the model to assign greater importance to the spatial locations that are considered informative.

Due to the computational constraints of the available hardware, the TokenLearner was implemented using `bfloat16` precision where possible. This numerical format requires 16 bits per value, compared to the 32 bits required by standard single-precision floating-point representation (FP32), thereby reducing memory consumption. However, some operations in the implementation are explicitly performed in FP32 for numerical stability, as described below.

Following the data flow through the TokenLearner, the feature representation produced by the CNN is initially provided as a tensor with dimensions

$$[B, C, H, W, D], \tag{4.3}$$

where B is the batch size, C is the number of feature channels, and H, W, and D represent the three spatial dimensions. The total number of voxels contained in each feature volume is then computed as

$$N_{\text{voxels}} = H \times W \times D. \tag{4.4}$$

In the implementation, this operation corresponds to:

```
B, C, H, W, D = feature_map_bchwd.shape
n_voxels = H * W * D
```

The feature representation is then processed by a spatial context block, defined as

```
self.spatial_context = nn.Sequential(
    nn.Conv3d(in_channels, in_channels, kernel_size=3, padding=1,
    groups=in_channels, bias=False),
    nn.GroupNorm(num_groups=8,num_channels=in_channels),
    nn.GELU()
    )
```

The first operation is a three-dimensional convolution with a kernel size of $3 \times 3 \times 3$. The number of output channels is equal to the number of input channels, and the parameter `groups=in_channels` ,makes this a depthwise convolution. Consequently, each channel is processed independently using its own convolutional filters, without mixing information between different channels. The padding of one preserves the spatial dimensions of the feature map.

The convolution is followed by Group Normalization with eight groups. Unlike Batch Normalization, which computes statistics across samples within a batch, Group Normalization divides the channels into groups and computes the normalization statistics within each group. Finally, a GELU activation function introduces a non-linear transformation.

Overall, this spatial context block allows the feature representation to be further processed according to its local three-dimensional spatial context before the tokenization step, while preserving the number of channels and the spatial dimensions.

The resulting feature map is then reshaped by collapsing the three spatial dimensions into a single voxel dimension:

```
feature_map_bcv = feat_ctx.reshape(B, C, n_voxels)
feature_map_bvc = feature_map_bcv.permute(0, 2, 1)
token_scores_bvn = self.linear(feature_map_bvc)
```

The first operation transforms the feature map from $[B,C,H,W,D]$ into $[B,C,N_{\text{voxels}}]$. A permutation is then applied to move the channel dimension to the last position, obtaining

$$[B,N_{\text{voxels}},C]. \tag{4.5}$$

This arrangement is required because the subsequent linear layer operates on the last dimension of the input tensor.

The linear layer represents the core operation responsible for generating the spatial attention scores used to construct the tokens. For each voxel, it maps the C-dimensional feature representation to N values, where N is the predefined number of tokens. Therefore, the output tensor `token_scores_bvn` has dimensions

$$[B, N_{\text{voxels}}, N], \tag{4.6}$$

where each of the N values associated with a voxel represents the score assigned by one of the learned tokens to that spatial location.

These scores do not yet represent the final token representations. Instead, they constitute spatial attention maps indicating which voxels are considered relevant for each token. Before converting these scores into attention weights, they are scaled by a temperature parameter and passed through a sigmoid activation function.

The use of a sigmoid activation allows each token to independently assign an attention weight to each voxel. In contrast to a softmax activation, which normalizes the scores across tokens and therefore introduces competition between them, sigmoid activation allows multiple tokens to attend to the same spatial location. The choice of sigmoid was made empirically, as it provided better results than the initially adopted softmax formulation.

The anatomical information obtained from SynthSeg is then incorporated into the attention maps. The brain mask is used to suppress voxels outside the brain, setting their corresponding attention values to zero. This constrains the tokenization process to the brain region and prevents non-brain voxels from contributing to the learned tokens.

The attention representation is subsequently reshaped back into a three-dimensional spatial structure, producing a tensor with dimensions

$$[B, N, H, W, D], \tag{4.7}$$

where each of the N channels corresponds to the spatial attention map associated with one learned token.

The final stage of the TokenLearner combines these attention maps with the original feature representation in order to generate the token embeddings:

```
attn_flat = attention_masks_bnhwd.float().view(B, self.n_tokens, n_voxels)
feats_flat = feature_map_bchwd.float().reshape(B, C, n_voxels)
tokens_bnc_sum = torch.einsum('b n v, b c v -> b n c',
                                                attn_flat,feats_flat)
token_mass = attn_flat.sum(dim=-1,keepdim=True)
token_mass_safe = torch.clamp(token_mass,min=1.0)
```

```
tokens_bnc = (tokens_bnc_sum / token_mass_safe).to(feature_map_bchwd.dtype)
```

The attention maps and feature maps are first flattened along their spatial dimensions. Both tensors are explicitly converted to FP32 before the weighted aggregation is performed. The attention tensor has dimensions

$$[B,N,N_{\text{voxels}}], \tag{4.8}$$

while the feature tensor has dimensions

$$[B,C,N_{\text{voxels}}]. \tag{4.9}$$

The two representations are then combined through a tensor contraction implemented using `torch.einsum`. Specifically, the operation

```
torch.einsum('b n v, b c v -> b n c', attn_flat, feats_flat)
```

performs a weighted sum of the feature vectors across all voxels for each token. The resulting tensor has dimensions

$$[B,N,C]. \tag{4.10}$$

Consequently, each token is represented by a C-dimensional feature vector obtained by aggregating the spatial features according to its corresponding attention map.

Since the `einsum` operation performs a weighted sum, tokens attending to a larger number of voxels would naturally tend to produce representations with larger magnitudes than tokens attending to smaller regions. To compensate for this effect, the attention values associated with each token are summed along the voxel dimension to obtain the corresponding attention mass:

$$M_n = \sum_{v=1}^{N_{\text{voxels}}} A_{n,v}, \tag{4.11}$$

where $A_{n,v}$ denotes the attention assigned by token n to voxel v.

The aggregated feature representation is then divided by this mass, effectively converting the weighted sum into a normalized weighted average:

$$T_n = \frac{\sum_{v=1}^{N_{\text{voxels}}} A_{n,v} X_v}{\max(M_n,1)}, \tag{4.12}$$

where X_v represents the feature vector associated with voxel v and T_n is the resulting embedding of token n. The use of the minimum value of one in the denominator prevents numerical problems in cases where the attention mass is very small.

The final output of the TokenLearner therefore consists of two complementary elements: the token embeddings `tokens_bnc`, with dimensions $[B,N,C]$, which are passed to the subsequent Transformer encoder, and the corresponding attention maps `attention_masks_bnhwd`, with dimensions $[B,N,H,W,D]$. While the token embeddings constitute the representation used by the self-supervised learning architecture, the attention maps retain the spatial information describing which regions contributed to each token. This information is subsequently exploited for diagnostic analyses and for the definition of the auxiliary loss.

4.6 JEPA

Before describing the JEPA training procedure, it is useful to summarize the evolution of the data throughout the encoder pipeline. This can be done by following the `forward` function implemented in the `ForwardEncoder` class, which is responsible for encoding both the context and target inputs.

4.6.1 Forward Encoder

The first stage of the encoder is the 3D CNN described in the previous sections. Starting from an input volume with dimensions

$$[B,1,H,W,D], \tag{4.13}$$

the CNN produces a three-dimensional feature representation. In the configuration used in this work, the resulting feature map has dimensions

$$[B,C,H/2,W/2,D/2]. \tag{4.14}$$

This operation can be summarized by the following portion of the forward pass:

```
cnn_features = self.cnn(x_volume.float())
if self.spatial_adapter is not None:
    feature_grid = self.spatial_adapter(cnn_features)
else:
    feature_grid = self.spatial_pool(cnn_features)
```

Since the feature dimensionality produced by the CNN does not directly correspond to the embedding dimensionality required by the subsequent TokenLearner, a spatial adapter is applied. In the configuration adopted in this work, the adapter projects the CNN feature channels to

an embedding dimension of 128. It consists of a $1 \times 1 \times 1$ convolution followed by Group Normalization and a GELU activation function:

```
self.adapter = nn.Sequential(
    nn.Conv3d(cnn_out, embed_dim, kernel_size=1),
    nn.GroupNorm(8, embed_dim),
    nn.GELU(),
    )
```

where `cnn_out` denotes the number of channels produced by the CNN and `embed_dim`=128 is the dimensionality of the feature representation used by the subsequent stages of the architecture. The $1 \times 1 \times 1$ convolution changes only the number of channels while preserving the spatial dimensions of the feature grid. Group Normalization is then applied to stabilize the resulting representation, followed by the GELU non-linear activation.

In parallel with the feature extraction process, an anatomical brain mask is generated from the SynthSeg segmentation. The corresponding operation is:

```
brain_mask_bool = (seg_tensor > 0).float()
target_spatial_size = feature_grid.shape[2:]

if brain_mask_bool.shape[2:] != target_spatial_size:
    brain_mask_bool = F.interpolate(brain_mask_bool,
    size=target_spatial_size, mode="nearest")
```

The expression `(seg_tensor > 0)` identifies the voxels belonging to the segmented brain structures. Although the resulting variable is named `brain_mask_bool`, the subsequent `.float()` operation converts the mask into a floating-point tensor, whose values are equal to one inside the segmented region and zero elsewhere.

Because the original segmentation has a spatial resolution of $128 \times 128 \times 128$, while the feature grid produced by the CNN and spatial adapter has a lower spatial resolution, the mask must be resized accordingly. If the two spatial dimensions do not match, the mask is interpolated to the spatial dimensions of the feature grid using nearest-neighbor interpolation. Nearest-neighbor interpolation is used to preserve the discrete binary nature of the anatomical mask.

Once the feature grid and corresponding brain mask have been obtained, both are passed to the TokenLearner:

```
sequence_tokens, token_attn_maps = self.spatial_tokenizer(feature_grid,
                                    brain_mask_bool)
```

The TokenLearner converts the dense three-dimensional feature grid into a sequence of learned tokens while simultaneously producing the corresponding spatial attention maps. The resulting token sequence is subsequently provided to the Transformer-based components of the JEPA architecture.

The final operation of the encoder is the addition of a learnable identity embedding to each token. This is implemented as follows:

```
roi_encoding = self.roi_token_norm(self.roi_class_tokens)  # [1, N_rois, C]
sequence_tokens = sequence_tokens + roi_encoding
```

The tensor `roi_class_tokens` contains a learnable embedding associated with each ROI token. These embeddings provide each token with a distinct, learnable identity that is associated with its corresponding anatomical ROI. Before being added to the sequence, the embeddings are normalized through `roi_token_norm`. The resulting ROI-specific encodings are then added element-wise to the token representations.

At the end of the forward encoder, the data have therefore been transformed from the original volumetric MRI representation into a structured sequence of anatomically associated token embeddings. These tokens constitute the input to the JEPA predictor.

4.6.2 Context and Target Encoders

The JEPA framework operates through two branches: a context encoder and a target encoder. At initialization, the two encoders share the same architecture and identical parameters. In the implementation, the target encoder is created as a deep copy of the context encoder:

```
self.context_encoder = base_encoder
self.target_encoder = copy.deepcopy(base_encoder)
```

The two branches subsequently play different roles during training. The context encoder acts as the trainable branch and receives the masked input, while the target encoder provides a representation of the corresponding unmasked input. The target representation therefore acts as the prediction target for the context branch.

The parameters of the target encoder are not updated through gradient descent. This is enforced by disabling gradient computation for its parameters:

```
for param in self.target_encoder.parameters():
    param.requires_grad = False
```

The target representation is additionally computed within a `torch.no_grad()` context:

```
with torch.no_grad():
    z_full_target, _ = self.target_encoder(x_volume=x_full,
        seg_tensor=seg_full,
        masking_mode="image", masked_roi_indices=None,)

        ...

    z_tgt = z_full_target[batch_indices, masked_roi_indices]
```

The target encoder receives the complete, unmasked volume x_{full}. Consequently, the resulting representation contains information from the entire input volume. Only the token positions corresponding to the masked ROIs are subsequently selected, producing z_{tgt}. These selected target tokens constitute the representations that the context branch is required to predict.

The context encoder, on the other hand, processes the masked input:

```
z_full_ctx, ctx_attention_masks = self.context_encoder(
        x_volume=x_masked if masking_mode == "image" else x_context_full,
        seg_tensor=seg_full,
        masking_mode=masking_mode,
        masking_family=masking_family,
        percentage_mask_ratio=percentage_mask_ratio,
        masked_roi_indices=masked_roi_indices,
        )
```

Depending on the selected masking mode, the context encoder receives either the masked volume or the corresponding context representation. The masking procedure removes or hides part of the information available to the encoder, forcing the model to infer the missing representations from the remaining context.

The resulting tensor `z_full_ctx` contains the token representations produced by the context encoder. These representations are then passed to the predictor, which is implemented as a Transformer encoder:

```
    z_pred_full = self.predictor(z_full_ctx)
    z_pred = z_pred_full[batch_indices, masked_roi_indices]
```

The predictor architecture is defined as follows:

```
    predictor_layer = nn.TransformerEncoderLayer(
            d_model=cnn_out,
```

```
            nhead=cfg.predictor_nhead,
            dim_feedforward=cnn_out * cfg.dim_feedforward_multiplier,
            batch_first=cfg.batch_first_transformer,
            activation=cfg.predictor_activation,
            )

    self.predictor = nn.TransformerEncoder(predictor_layer,
            num_layers=cfg.predictor_layers)
```

The predictor receives the sequence of context tokens and applies a stack of Transformer encoder layers. Its purpose is to transform the context representation into a prediction of the target representation. As with the target branch, only the token positions corresponding to the masked ROIs are selected for the subsequent loss computation.

The predicted and target representations are finally combined into a single tensor:

```
z_combined = torch.stack([z_pred, z_tgt],dim=1)
```

The resulting representation contains, for each masked ROI, both the prediction generated from the masked context and the corresponding target representation obtained from the unmasked input. These representations are subsequently provided to the loss function, together with the full context and target representations and the attention maps produced by the context encoder:

$$\left\{ z_{\text{pred}}, z_{\text{tgt}}, z_{\text{full}}^{\text{ctx}}, z_{\text{full}}^{\text{tgt}}, A_{\text{ctx}} \right\} \tag{4.15}$$

The predicted and target tokens form the basis of the JEPA objective, while the additional representations and attention maps are used by the auxiliary components of the training objective and by the diagnostic procedures described in the following sections.

4.7 Masking

This section details how the masking strategy is implemented within the encoder branch.

As previously established, masking must be applied within the context encoder to facilitate the model's representation learning process. Consequently, we implemented various masking strategies to evaluate the model's ability to learn under different conditions.

The masking strategies are categorized across two dimensions: by modality (applied either to the raw input image or to the latent tokens) and by family (anatomical or geometrical).

To facilitate the subsequent analysis of the results, each applied masking approach is detailed separately below.

4.7.1 Anatomical Image

As the name suggests, this modality masks specific anatomical regions directly within the input brain image. Masking operates on the 3D MRI volume, leveraging the corresponding SynthSeg segmentation maps to isolate targeted anatomical structures.

The first step involves collecting the segmentation indices generated by SynthSeg and mapping each value to a specific Region of Interest (ROI). SynthSeg natively segments the brain into 32 distinct regions, each assigned an integer label. However, because we utilized cortical parcellation, the `right_cerebral_cortex` and `left_cerebral_cortex` are further subdivided into 35 parcels each. Therefore, it is necessary to aggregate these sub-parcel indices back under their respective parent region names.

Once the index-to-region mapping is established, the masking process begins. First, we select the regions to obscure. Since predicting a single missing anatomical region would yield a trivial task for the model, we chose to mask 8 distinct indices per batch.

After selecting the target indices, a binary mask is applied. This mask systematically sets all voxels belonging to the chosen ROIs to zero, resulting in the final masked T1-weighted input volume.

4.7.2 Geometrical Image

The geometrical image masking strategy also operates on the original 3D MRI volume. However, instead of targeting specific ROIs, a 3D cuboid is used to obscure all voxels within its spatial extent.

The dimensions of this cube are determined by sampling randomly from a predefined size range, which is set as a hyperparameter. Positioning is critical; placing the cube near the image borders might result in masking only empty background space. To prevent this, we first compute the bounding box of the actual brain tissue by identifying its spatial extent (i.e., filtering out background voxels with a value of zero).

Within these brain coordinates, a random anchor point is selected. This anchor point is the central target for the masking operation. We then calculate the valid spatial ranges for the cube to ensure that this anchor point is strictly contained within its boundaries.

The generated cube is then applied to the image, zeroing out all enclosed voxels. Finally, a quality check is performed and logged as metadata to verify that a significant portion of actual brain tissue was successfully obscured during the procedure.

4.7.3 Anatomical Token

This modality operates deeper within the context encoding section. Instead of altering the input image, the masking is applied directly to the latent tokens before they are processed by the Transformer.

Specifically, this masking occurs immediately after the TokenLearner module. Rather than zeroing out the latent vectors of the selected tokens, they are replaced with learnable parameters initialized from a random normal distribution, which are subsequently normalized.

```
        self.roi_class_tokens = nn.Parameter(
            torch.randn(1, self.num_rois, cnn_out)*0.02
        )
        self.roi_token_norm = nn.LayerNorm(cnn_out)
    ...
    def forward:
    ...
    if masking_family == "anatomical" and masked_roi_indices is not None:
            normalized_concepts = self.concept_token_norm(self.roi_concept_tokens)
            sequence_tokens = apply_sample_token_mask(sequence_tokens, masked_roi_indice
```

Here, `self.num_rois` represents the total number of extracted ROIs (32), and `cnn_out` indicates the channel dimension exiting the CNN feature extractor (128). During the forward pass, we identify the list of target ROIs to be masked and replace their corresponding TokenLearner outputs with these randomly initialized, normalized concept tokens.

4.8 Loss

The loss function defines the primary optimization objective of the framework. Training the network inherently involves iteratively adjusting the model's weights to minimize this objective.

Our approach utilizes a multi-objective loss function. We integrate multiple loss components to achieve robust representation learning: the primary objective is VICReg, originally proposed for JEPA architectures. To this, we appended an auxiliary spatial loss (a combination of Binary Cross-Entropy and Dice loss). Incorporating a Dice loss technically transitions the learning paradigm away from strict self-supervision, as it requires labeled data to guide the attention maps. Ultimately, the long-term goal of this framework is to completely remove the auxiliary Dice component, relying purely on self-supervised learning.

4.8.1 VICReg

Having introduced the theoretical foundation of VICReg in the background section, this section details its specific implementation within our model.

The function first casts both the masked and full tokens to `float` and normalizes them using Root Mean Square (RMS) normalization:

```
z_masked = z_masked.float()
z_full = z_full.float()
...
z_m_norm = self._rms_norm(z_masked)  # [B, 2, S_masked, C]
z_f_norm = self._rms_norm(z_full)    # [B, 2, S_full, C]
```

The normalized tokens are then split based on their originating branch:

```
z_pred_m   = z_m_norm[:, 0]
z_target_m = z_m_norm[:, 1]
z_ctx_f    = z_f_norm[:, 0]
z_tgt_f    = z_f_norm[:, 1]
```

Here, `z_pred_m` represents the tokens predicted by the online predictor from the masked input, while `z_target_m` represents the corresponding target tokens from the unmasked branch (the EMA target). `z_ctx_f` contains the full sequence of contextual tokens from the masked branch, and `z_tgt_f` contains the full sequence generated by the target branch.

Quality Control (QC) scores calculated by SynthSeg are also integrated into this loss formulation. Because these scores reflect the reliability of the segmentation, we weight the relevant loss terms by them; this ensures that poorly segmented regions exert less influence on the parameter updates.

The first component of VICReg is the **Invariance** term, which is computed exclusively on the masked tokens and their corresponding targets:

```
inv_loss_per_token = (
    F.mse_loss(z_pred_m, z_target_m,
    reduction='none').mean(dim=-1).mean(dim=0))
inv_loss_weighted = (self.sim_weight * inv_loss_per_token *
                     qc_weights_s).mean()
```

The Mean Squared Error (MSE) is computed between the predicted and target tokens. This error is averaged first across the channel dimension (`dim=-1`), and subsequently across the batch

dimension (`dim=0`), resulting in an array of errors. These values are then weighted by the invariance hyperparameter (`self.sim_weight`) and the SynthSeg QC scores before being reduced to a single scalar.

The second core component is the **Variance**, which comprises two distinct elements: Cross-Subject variance and In-Subject variance. For the Cross-Subject variance, we calculate the variance across tokens within the same batch (i.e., across different subjects encoded under identical masking conditions):

```
std_pred_b = torch.sqrt(z_pred_m.var(dim=0) + self.eps)

var_b_per_token = torch.mean(F.relu(1.0 - std_pred_b), dim=-1)
var_b_weighted = (self.var_weight_b * var_b_per_token *
                  qc_weights_s).mean()
```

We calculate the variance along the batch dimension and apply the square root, effectively yielding the standard deviation. A hinge loss formulation (via a ReLU activation) forces the standard deviation to be at max 1.0. If the standard deviation meets or exceeds this threshold, the penalty drops to zero, preventing negative components. The resulting penalty is averaged across the channel dimension, weighted by the cross-subject variance hyperparameter (`self.var_weight_b`), modulated by the QC scores, and reduced to a scalar.

The In-Subject (spatial) variance requires more careful handling. Because masked tokens are replaced by learnable or random concept tokens during processing, the model could trivially minimize the loss by simply distancing the concept tokens from the real anatomical tokens. To prevent this shortcut, we apply a boolean mask to exclude all masked tokens from the calculation:

```
unmasked = torch.ones(B, S_full, dtype=torch.bool, device=z_ctx_f.device)
bidx = torch.arange(B, device=z_ctx_f.device).unsqueeze(-1)
unmasked[bidx, masked_indices] = False
```

We initialize a boolean tensor matching the size of the entire token sequence `S_full` to `True`. Using batch indices and masked indices, we set the positions of the obscured tokens to `False`. After isolating the unmasked contextual tokens, we compute the true spatial mean and variance. From this point, the hinge loss application is identical to the cross-subject variance, though QC weighting is omitted since the computation assesses spatial variance within the exact same subject.

The final term of the VICReg objective is the **Covariance** penalty, which decorrelates the latent channels and is calculated exclusively on the unmasked features:

```
z_unmasked_pooled = z_ctx_f[unmasked]
cov_global = _cov_loss(z_unmasked_pooled)
cov_weighted = self.cov_weight * cov_global
```

These unmasked features are processed by the `_cov_loss` function, defined as follows:

```
def cov_loss(x: torch.Tensor) -> torch.Tensor:
    if x.shape[0] < 2:
        return torch.tensor(0.0, device=x.device)
    x_flat = x.float() - x.float().mean(dim=0, keepdim=True)
    cov = (x_flat.T @ x_flat) / (x_flat.shape[0] - 1)
    cov.fill_diagonal(0.0)
    return cov.pow(2).sum() / C
```

The input tensor is first centered by subtracting its mean along the sample dimension (guaranteeing a zero mean). We then compute the covariance matrix by multiplying the transposed centered tensor by itself, dividing by the number of elements minus one (Bessel's correction). Because the diagonal represents the variance of individual channels—which is already optimized by the variance terms—it is forced to zero. Finally, the off-diagonal elements are squared (to equally penalize positive and negative correlations), summed, and normalized by the total number of channels `C`.

The total VICReg objective is simply the sum of these weighted components:

```
total_loss = inv_loss_weighted + var_b_weighted + var_s_weighted + cov_weighted
```

4.8.2 BCE + Dice Auxiliary Loss

An auxiliary spatial loss was added to the primary VICReg objective. As previously mentioned, integrating a spatial loss technically shifts the paradigm away from strict self-supervision. However, as demonstrated in the Results section, this addition significantly improves the model's spatial encoding capabilities.

This hybrid loss combines Binary Cross-Entropy (BCE) and Dice loss, equally weighted at 50%:

```
loss_combo = 0.5 * dice_loss + 0.5 * bce_loss
```

Before detailing these components, we define `pred_flat` as the predicted attention masks flattened across the spatial dimensions (D,H,W), resulting in a shape of `[B, N, V]`. The tensor `target_flat` represents the corresponding flattened target masks generated by the EMA branch, and `brain_mask` is a boolean tensor indicating the spatial extent of the brain tissue within the volume.

4.8.3 BCE

The Binary Cross-Entropy (BCE) component is computed as follows:

```
eps = 1e-7
p_clamped = pred_flat.clamp(eps, 1.0 - eps)
bce_voxels = -(target_flat * torch.log(p_clamped) + (1.0 - target_flat) *
            torch.log(1.0 - p_clamped))
if brain_mask is not None:
    mask_v = brain_mask.flatten(start_dim=2).float()
    bce_loss = (bce_voxels * mask_v).sum(dim=-1) / mask_v.sum(dim=-1).clamp_min(1.0)
else:
    bce_loss = bce_voxels.mean(dim=-1)
```

First, the predictions are clamped to a stable interval to prevent logarithmic singularities. The standard BCE formula is then applied at the voxel level:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)\right] \tag{4.16}$$

If a brain mask is provided, the loss calculation is refined by masking out the background, restricting the penalty exclusively to the foreground brain voxels.

4.8.4 Dice

The Dice loss component is similarly implemented:

```
intersection = (pred_flat * target_flat).sum(dim=-1)
union = pred_flat.pow(2).sum(dim=-1) + target_flat.pow(2).sum(dim=-1)
dice_loss = 1.0 - (2.0 * intersection + self.smooth) / (union + self.smooth)
```

The intersection is computed via element-wise multiplication between the prediction and the target, summed over the voxel dimension. The union is calculated by summing the squared values of both voxel maps. The final Dice loss is defined as 1.0 minus the Dice coefficient, utilizing a smoothing factor (`self.smooth = 1e-5`) in both the numerator and denominator to prevent division by zero.

4.9 Training

4.9.1 Hyperparameters and Loss Weights

The training configuration is established through a structured set of hyperparameters defining the optimization dynamics, loss function weighting, data selection criteria, and spatial masking geometries. The most important ones are here listed:

```
number_of_epochs: int = 200
batch_size: int = 3
accumulate: int = 12
learning_rate: float = 1e-4

learning_rate_SynthSeg_cnn: float = 1e-6
learning_rate_token_learner: float = 2e-4
learning_rate_pool: float = 1e-4
learning_rate_roi_class: float = 1e-4
learning_rate_predictor: float = 1e-4
learning_rate_spatial_adapter: float = 1e-4

dice_loss_weight: float = 5.0

roi_class_tokens_weight: float = 0.1
roi_concept_tokens_weight: float = 0.1
ema_momentum: float = 0.996

qc_filter_threshold: float = 0.75

synthseg_vicreg_invariance_weight: float = 8.0
synthseg_vicreg_var_b_weight: float = 3.0
synthseg_vicreg_var_s_weight: float = 3.0
synthseg_vicreg_cov_weight: float = 0.8

geometric_cube_size_range: tuple[int, int] = (30, 60)
```

During exploratory phases and architecture validation, the model was trained for 80 epochs to achieve a favorable trade-off between convergence monitoring and computational efficiency. The complete training schedule of 200 epochs was deployed for the final experimental evaluation.

To ensure statistical significance during the calculation of variance and covariance terms within the VICReg loss, a local batch size $B_{\text{local}} \geq 2$ is strictly required. While initial experiments utilized $B_{\text{local}} = 2$, final runs expanded this to $B_{\text{local}} = 3$ based on VRAM capacity.

To overcome hardware memory constraints while maintaining a large effective batch size, gradient accumulation was implemented over $N_{\text{acc}} = 12$ steps, resulting in an effective batch size of:

$$B_{\text{effective}} = B_{\text{local}} \times N_{\text{acc}} = 3 \times 12 = 36 \tag{4.17}$$

Learning rates of the single modules were assigned heterogeneously. A reduced rate ($\eta = 10^{-6}$) was allocated to the pre-trained SynthSeg-FPN backbone to prevent catastrophic forgetting of anatomical features, whereas a higher rate ($\eta = 2 \times 10^{-4}$) was assigned to the `TokenLearner`. Based on our architectural hypothesis, the `TokenLearner` plays a critical role in spatial token extraction, requiring more aggressive gradient updates.

Rather than relying on an expensive grid search, loss scaling factors were fine-tuned via post-training empirical analysis of the validation Dice score evolution across epochs. Data curation was enforced using a quality control threshold (QC > 0.75), discarding corrupted raw MRIs or failed initial segmentations prior to training.

4.9.2 Data Processing and Input Pipeline

Subject identifiers are retrieved from a master metadata file. The dual-branch framework—comprising the Context Encoder and the Target Encoder updated via Exponential Moving Average (EMA)—is initialized and transferred to the GPU device.

Data transformations are organized using a MONAI-based pipeline divided into two main processing stages:

1. **Deterministic Resampling:** Input volumes are standardized to a uniform isotropic resolution of $128 \times 128 \times 128$ voxels using `LoadImaged`, `EnsureChannelFirstd`, and `Resized`.

2. **Stochastic Domain Augmentation:** To force the Context Encoder to learn scanner-invariant representations, independent spatial and intensity transformations are applied to the context branch. These include magnetic field bias non-uniformities (`RandBiasFieldd`), T_1 relaxation variability (`RandAdjustContrastd`), intensity distribution shifts (`RandHistogramShiftd`), and additive Gaussian noise (`RandGaussianNoised`).

To eliminate redundant disk I/O and NIfTI decoding overhead across epochs, pre-processed volumes meeting the quality control criteria (QC > 0.75) are loaded directly into GPU VRAM

using MONAI's `CacheDataset`. Using `DuplicateImaged`, each cached item yields two independent volume views: an augmented view routed to the Context Encoder and an uncorrupted target view passed to the Target Encoder.

The Target Encoder parameters θ_{target} are not updated via backpropagation. Instead, they track the Context Encoder parameters θ_{context} using an EMA momentum $\tau = 0.996$:

$$\theta_{\text{target}} \leftarrow \tau\theta_{\text{target}} + (1-\tau)\theta_{\text{context}} \tag{4.18}$$

4.9.3 Optimization and Memory-Efficiency Strategies

Parameter optimization is managed via the Adam algorithm. To ensure smooth convergence toward global minima, a Cosine Annealing learning rate scheduler decays the learning rate across epochs according to:

$$\eta_t = \eta_{\text{min}} + \frac{1}{2}(\eta_{\text{base}} - \eta_{\text{min}})\left(1 + \cos\left(\frac{t}{T_{\text{max}}}\pi\right)\right) \tag{4.19}$$

where $\eta_{\text{min}} = 10^{-5}$. This schedule provides slow initial decay, rapid reduction in intermediate epochs, and fine-grained updates near termination.

Given the low memory available four computational optimization strategies were integrated:

- **Automatic Mixed Precision (AMP):** Operates forward and backward passes dynamically in `bfloat16`/`float32` precision, significantly reducing VRAM consumption while preserving numerical stability.
- **Gradient Accumulation:** Accumulates gradients across 12 micro-batches before executing `optimizer.step()`, enabling an effective batch size of 36 without exceeding single-GPU memory limits.
- **Gradient Clipping:** Normalizes total parameter gradient norms to a threshold of max_norm $= 1.0$ prior to optimizer steps, preventing gradient explosion in the Transformer Predictor layers.
- **Gradient Checkpointing:** Recomputes intermediate activations on-the-fly during the backward pass rather than storing all intermediate feature maps, preventing peak VRAM Out-Of-Memory (OOM) errors at the cost of a minor compute overhead.

4.9.4 Parameter Distribution and Computational Efficiency

Table 4.2 details the total and trainable parameter counts across the C-JEPA sub-modules.

Table 4.2: Parameter distribution across C-JEPA architectural modules.

Module	Total Parameters	Trainable Parameters
`cnn` (Backbone + FPN)	8,485,864	535,168
`spatial_tokenizer`	7,841	7,840
`spatial_adapter`	16,768	16,768
`roi_concept_tokens`	4,096	4,096
`predictor`	396,544	396,544
`target_encoder` (EMA, Frozen)	8,519,177	0
Total	**17,430,290**	**960,416**

To preserve domain-specific volumetric representations learned during SynthSeg pre-training, the convolutional feature extractor within the 3D CNN backbone was frozen. Only the Feature Pyramid Network (FPN) projection layers were kept trainable ($535{,}168$ parameters out of 8.48M). This reduced the total trainable parameter count to $960{,}416$ (5.51% of total parameters), accelerating convergence and training times.

Computational throughput remained consistent across various spatial masking strategies (Anatomical, Geometric, and Random), averaging approximately 2 minutes per epoch (6 hours per 200-epoch training run). This represents a $> 5\times$ speedup in training compared to a full-volume ResNet3D baseline under identical hardware conditions.

4.10 Hardware and Sustainability

The training experiments were performed using a single NVIDIA A30 GPU, providing 24,576 MB of available VRAM. Approximately 30 training sessions were conducted to evaluate the different model configurations and architectural choices. Depending on the experiment, individual training sessions lasted between approximately 2 and 6 hours.

During training, GPU memory utilization typically ranged between 80% and 90% of the available memory. The maximum power consumption of the NVIDIA A30 is reported as 165 W [42]. To provide a conservative estimate of the energy consumption, all training sessions were assumed to have lasted 6 hours while operating continuously at the maximum rated power.

Under these assumptions, the estimated energy consumption is:

$$E = N \times t \times P \tag{4.20}$$

where $N = 30$ is the number of training sessions, $t = 6$ h is the assumed duration of each session, and $P = 165$ W is the maximum GPU power consumption. This results in an estimated total consumption of approximately 29.7 kWh.

This estimate represents an upper-bound approximation of the GPU energy consumption during the reported experiments, as the actual training durations and instantaneous power consumption were generally lower than the assumed maximum values. It therefore provides a conservative reference for the computational cost of the experiments, rather than a direct measurement of their total energy or environmental impact.

4.11 AI Usage

Large Language Models (LLMs) were used as supporting tools during the development of this thesis and the associated research activities.

ChatGPT was primarily used during the writing phase to improve the readability and linguistic quality of the thesis. More specifically, after drafting individual sections, it was used to: generating the pipeline image(manually edited later)4.1, identify grammatical errors, improve sentence structure, and adapt the text to a more formal and academic style. The scientific content and the underlying work were produced by the author, with the LLM being used as a revision and language-support tool.

Gemini was also used during the development of the computational framework, primarily as a supporting tool during debugging and for minor code adjustments.

The use of these tools was conducted in accordance with the applicable University of Padua guidelines concerning the use of Artificial Intelligence in thesis and research activities.[43]

Chapter 5

Results

In this section, the results obtained from the most informative training sessions are presented and discussed. The analysis follows the overall structure of the Methods section rather than the chronological order in which the experiments were conducted. When substantial changes were introduced during the development of the framework, their effects are therefore analyzed by methodological component, comparing the configurations before and after the corresponding change. After analyzing singular changes, an ablation study will be presented comparing three different type of masking and their relative performances with the presence or absence of the auxiliary DICE + BCE loss in the loss function.

5.1 Dataset

As described in the Methods section, the initial subset was selected with the aim of approximately preserving the proportions of the original FOMO300K dataset. However, further analysis revealed that this selection resulted in an uneven distribution of acquisition sources. A subsequent reselection of subjects was therefore performed to increase the diversity of the final subset and reduce the influence of overrepresented acquisition sources.

The initial imbalance was investigated through Principal Component Analysis (PCA). The first principal component (PC1) accounted for 48.2% of the total variance, indicating that a substantial portion of the variability in the selected dataset could be described along a single principal direction. To investigate the origin of this variability, the corresponding PC1 score distribution was further analyzed using a score-density plot. This analysis revealed a marked contribution from subjects originating from the Nigerian acquisition source, which was overrepresented in the initial subset.

The observed distribution suggested the presence of an acquisition-related bias. In particular, the Nigerian subjects exhibited a different distribution in the feature space compared with

the other acquisition sources. Consequently, the model could potentially exploit acquisition-specific characteristics rather than learning representations primarily related to the anatomical or clinical properties of the MRI volumes. Although diversity across acquisition sources is generally desirable, the excessive representation of a single source can introduce a disproportionate contribution to the learned feature distribution.

An additional issue was identified in the DWI acquisitions. Some of the selected DWI images were effectively two-dimensional rather than suitable three-dimensional volumetric inputs. Although this issue was not identified as the primary source of the observed distributional imbalance, these acquisitions were incompatible with the intended 3D processing pipeline and were therefore excluded during the subsequent dataset reselection.

The effect of the dataset modification is illustrated in Figure 5.1. The upper row shows the PCA distributions obtained from the initial dataset, whereas the lower row shows the corresponding distributions after the reselection. Following the rebalancing of the acquisition sources and the exclusion of incompatible DWI acquisitions, the dominant structure previously observed in PC1 was substantially reduced. This indicates that the contribution of the overrepresented acquisition source to the overall feature distribution was reduced, resulting in a more heterogeneous representation of the selected dataset.

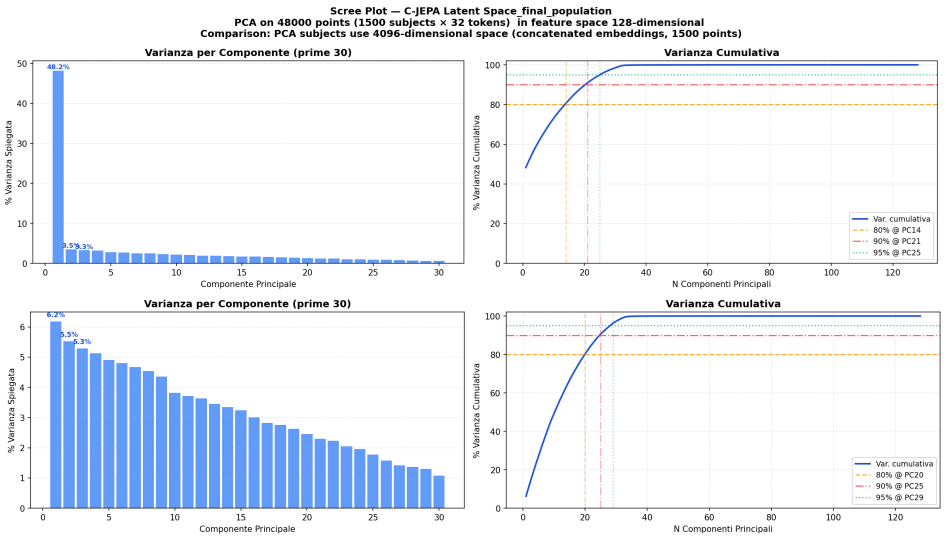


Figure 5.1: Comparison of Principal Component Analysis (PCA) in the latent space before and after dataset modification. The top row displays the explained variance distribution before the intervention, characterized by a strong dominance of the first principal component (48.2%). The bottom row illustrates the analysis following the dataset modification, where the variance is more evenly distributed across multiple components (the first component explains only 6.2%), indicating a balanced latent representation and increased informational diversity.

5.2 CNN

The SynthSeg-based CNN and FPN architecture did not undergo any major modifications during the experiments presented in this thesis. However, a minor implementation issue identified during the development of the framework is worth reporting.

In the initial implementation of the SynthSeg-based CNN, the activation function used within the convolutional blocks was incorrectly set to ReLU instead of the ELU activation function adopted by the original SynthSeg architecture. This discrepancy was identified and corrected before the training sessions analyzed in this section; therefore, it did not affect the results reported here.

Nevertheless, the difference between the two activation functions is relevant when considering the behaviour of the CNN feature extractor. To provide a visual comparison, Figure 5.2 illustrates a portion of the output produced by the different activation functions, highlighting their different responses to the same input.

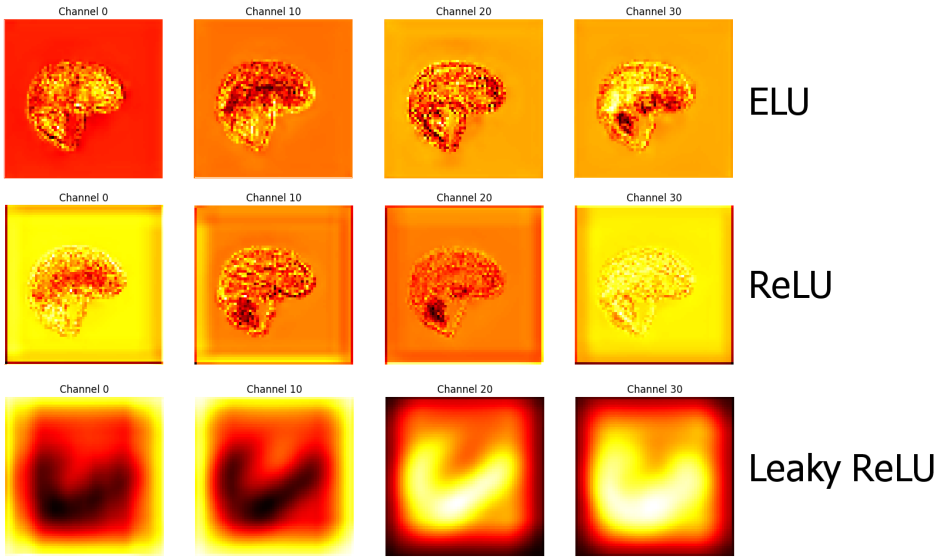


Figure 5.2: Comparison of FPN feature maps using different activation functions in the SynthSeg-based encoder. Each row displays the Feature Pyramid Network output for selected channels (Channel 0, 10, 20, and 30) utilizing ELU (top), ReLU (middle), and Leaky ReLU (bottom) activation functions within the encoder's convolutional blocks. The comparison highlights how the choice of activation directly influences the contrast, the sharpness of anatomical details, and the overall spatial response of the extracted features.

5.3 Loss

Among the components of the proposed framework, the loss function underwent the changes that had the most noticeable impact on the model's performance. While the VICReg formulation remained substantially unchanged, apart from a rebalancing of its individual loss weights, the most significant improvement was obtained through a modification of the auxiliary spatial loss.

5.3.1 VICReg

To investigate the effect of the VICReg weight rebalancing, a visual comparison of the training results was first performed. The VICReg coefficients were adjusted to obtain a more balanced contribution of its individual components, and the resulting evolution of the model was compared with the previous configuration (Figure 5.3).

Based on the diagnostic visualizations developed during the rebalancing process, the differences between the two configurations were most apparent in the evolution of the overall loss. The changes introduced by the VICReg weighting primarily affect the learned token representations, whereas their effect on the spatial attention maps is less pronounced. Similarly, no substantial difference was observed in the organization of the latent space, as illustrated by the corresponding t-SNE representations.

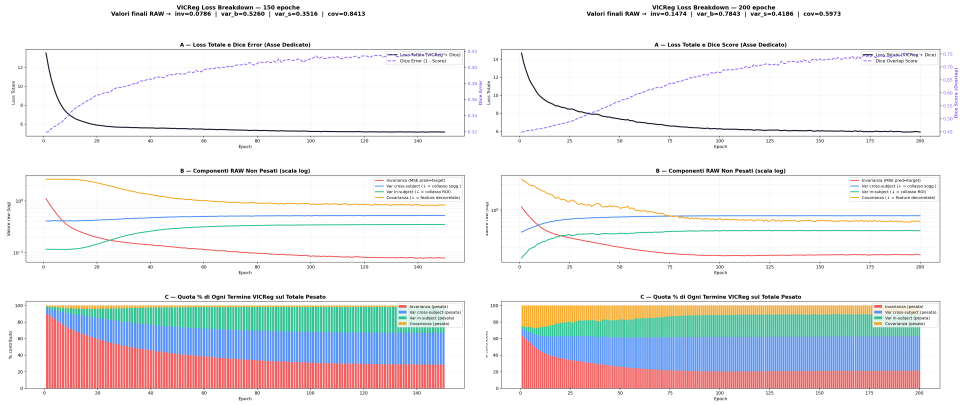


Figure 5.3: Comparison of VICReg loss term balancing before and after weight recalibration. The left column illustrates the initial configuration at 150 epochs, while the right column shows the reweighted configuration at 200 epochs. The panels display the total loss and Dice score trends (A), the unweighted raw components in logarithmic scale (B), and the percentage share of each term on the weighted total (C). Although the temporal trend of individual components is correct in both experiments, the initial weighting almost entirely obscured the covariance contribution, skewing the loss towards invariance; the subsequent rebalancing restores a proportionate and effective distribution among all regularization terms.

Overall, the VICReg rebalancing therefore modified the optimization dynamics without producing a comparable change in the spatial behaviour of the model. This observation mo-

tivated further investigation of the auxiliary spatial loss, which had a more direct influence on the anatomical attention maps.

5.3.2 Auxiliary Loss

The most substantial improvement in model performance was obtained by modifying the auxiliary spatial loss. In the original configuration, the spatial objective relied exclusively on the Dice loss. This formulation was subsequently extended by introducing a Binary Cross-Entropy (BCE) component, resulting in the combined Dice+BCE objective described in the Methods section.

The effect of this modification is particularly evident in the ROI-specific Dice scores and in the corresponding attention maps. With the original Dice-only formulation, several ROIs showed limited improvement throughout training, with some regions remaining difficult to learn even after a substantial number of epochs. Following the introduction of the BCE component, the Dice scores increased for most ROIs, indicating a more accurate correspondence between the predicted and target spatial representations.

A similar improvement can be observed in the attention maps. In the earlier configuration, the attention associated with several ROIs was distributed over relatively large and diffuse regions, producing broad ”blobs” rather than localized anatomical patterns. One possible explanation is that this behaviour allowed the model to obtain a relatively favourable spatial overlap while maintaining broad activation regions. As training progressed and the learning rate decreased according to the adopted CosineAnnealing schedule, the model may have had less capacity to substantially modify this behaviour.

After introducing the BCE component, the attention maps became considerably more localized. Instead of producing broad activation regions, the model was able to focus its attention on anatomically relevant areas, with substantially better-defined boundaries. This behaviour is consistent with the improvement observed in the ROI-specific Dice scores and indicates that the combined Dice+BCE objective provides a more effective spatial constraint for the TokenLearner attention maps.

Despite the overall improvement, the ventricle regions did not show a comparable increase in performance. This behaviour can be related to the spatial resolution of the feature representation used by the model. Due to the memory constraints associated with processing the original $128 \times 128 \times 128$ volumes, the FPN output is produced at a reduced spatial resolution of $64 \times 64 \times 64$. Consequently, small anatomical structures occupy fewer voxels in the resulting feature grid and are therefore more difficult to represent accurately. This limitation is particularly relevant for small structures such as the inferior lateral ventricles, which can consequently result in lower Dice+BCE performance and a less evident improvement compared with larger ROIs.

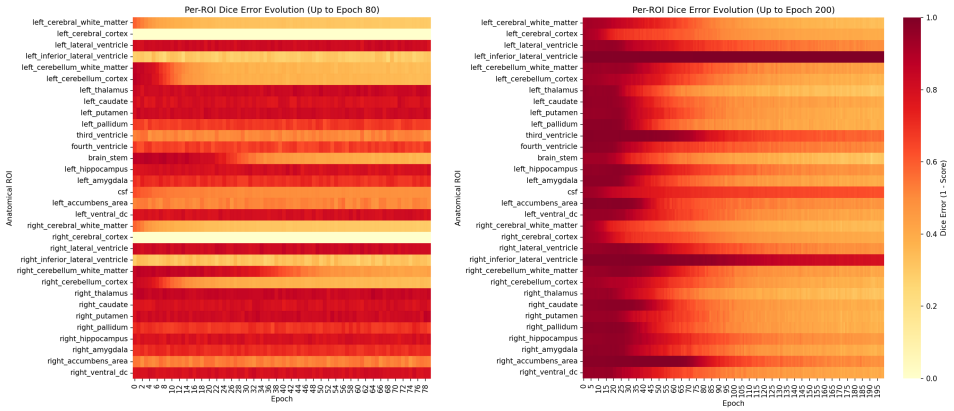


Figure 5.4: Per-ROI Dice error evolution comparing different auxiliary loss functions. The left panel shows the training progression using exclusively the Dice loss, while the right panel illustrates the results obtained by adopting a hybrid BCE+Dice function. The BCE+Dice configuration (right) clearly optimizes global learning, allowing the model to converge and correctly identify almost all anatomical regions. The apparent perfect score (zero error) for the cerebral cortex in the left graph is strictly an artifact caused by a hard-coding error in the original label declaration and does not reflect actual segmentation performance.

5.4 Ablation Study

A final ablation study was performed to investigate how the different masking strategies and the auxiliary spatial loss affected the behaviour of the model. Rather than evaluating a single configuration, six combinations were considered, obtained by crossing the three implemented masking strategies with the presence or absence of the auxiliary Dice+BCE loss:

- Anatomical Image Masking with Dice+BCE Auxiliary Loss;
- Anatomical Image Masking without Dice+BCE Auxiliary Loss;
- Geometrical Image Masking with Dice+BCE Auxiliary Loss;
- Geometrical Image Masking without Dice+BCE Auxiliary Loss;
- Anatomical Token Masking with Dice+BCE Auxiliary Loss;
- Anatomical Token Masking without Dice+BCE Auxiliary Loss.

These configurations were selected because they provide a direct comparison between the three masking strategies while simultaneously evaluating the contribution of the auxiliary spatial loss. The ablation study was performed using the following common training configuration:

```
memory_mode: str = "low" # Option: "low" (A30 24GB, Checkpointing ON)
number_of_epochs: int = 200
```

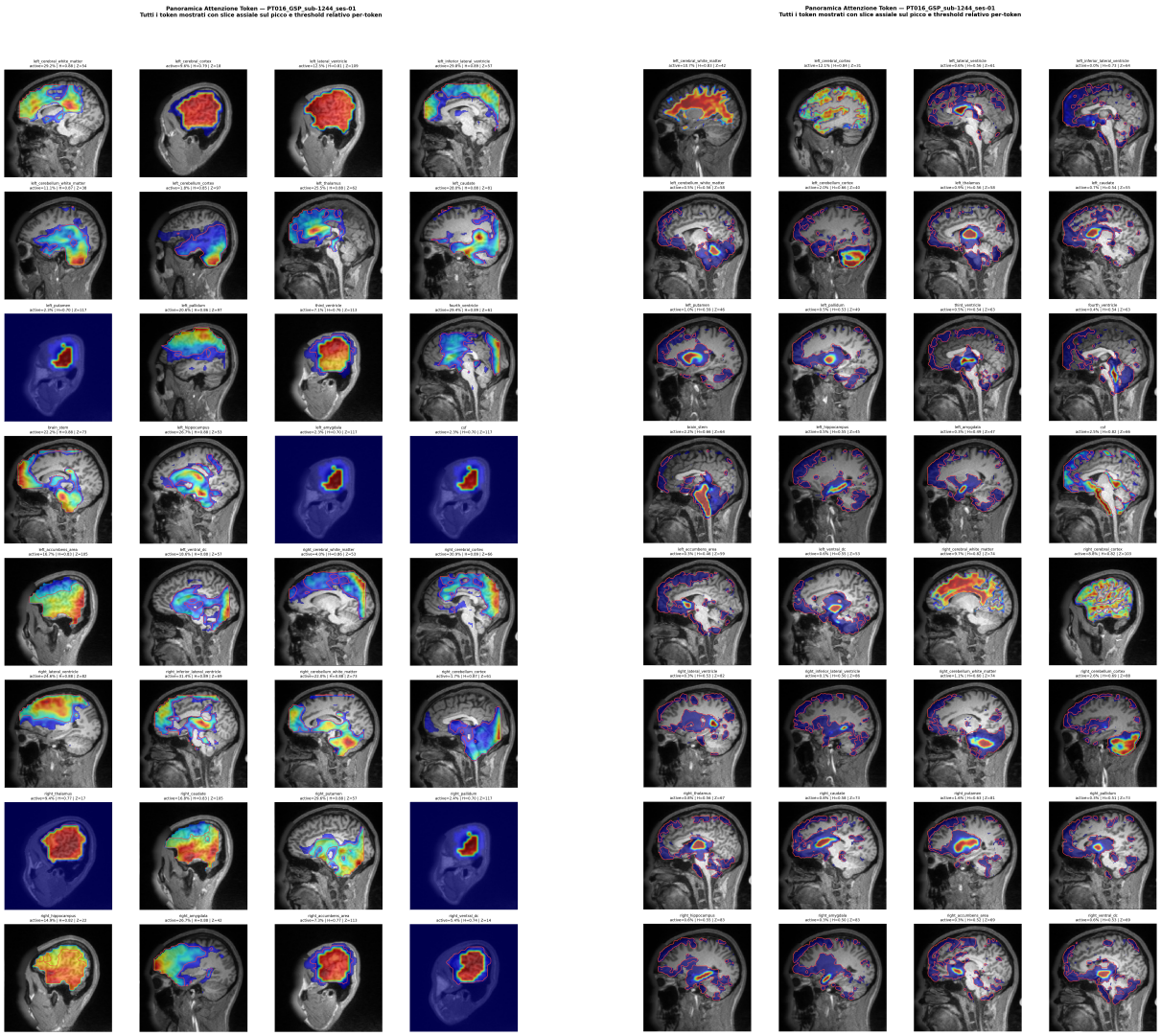


Figure 5.5: Comparison of attention maps before and after loss function optimization. The image illustrates the visual difference in activations generated using exclusively the Dice loss (left) versus the hybrid Dice+BCE combination (right). The displayed slice for each panel is automatically selected by the system based on where the peak attention is detected. The transition to the combined loss highlights a clear qualitative improvement, as the attention maps shift from blurred, dispersed regions (appearing as ”blobs”) to well-defined and anatomically significant areas.

```
batch_size: int = 3
accumulate: int = 12
learning_rate: float = 1e-4

learning_rate_SynthSeg_cnn: float = 1e-6
learning_rate_ResNet_cnn: float = 1e-4
learning_rate_token_learner: float = 2e-4
learning_rate_pool: float = 1e-4
learning_rate_roi_class: float = 1e-4
learning_rate_predictor: float = 1e-4
learning_rate_spatial_adapter: float = 1e-4

auxiliary_loss_mode: str = "bce_dice"
dice_loss_weight: float = 5.0

synthseg_vicreg_invariance_weight: float = 8.0
synthseg_vicreg_var_b_weight: float = 3.0
synthseg_vicreg_var_s_weight: float = 3.0
synthseg_vicreg_cov_weight: float = 0.8

geometric_cube_size_range: tuple[int, int] = (30, 60)
```

The experiments were initially based on the 550-subject subset described in the Dataset section. Ten subjects were subsequently excluded because of insufficient SynthSeg QC scores (below 75%), resulting in a final cohort of 540 subjects.

The purpose of this ablation study was to investigate training dynamics and the internal behaviour of the learned representations rather than to provide an unbiased estimate of generalization performance. For this reason, the experiments were conducted without a separate validation or test set. The reported results should therefore be interpreted as observations of the behaviour of the models on the training cohort and should not be considered direct measurements of performance on unseen data.

5.4.1 Masking in the FPN

The first analysis investigates whether the different masking strategies produce substantially different feature representations at the output of the CNN-FPN module. To visualize these representations, Principal Component Analysis (PCA) was applied to the FPN output. The first

three principal components were then mapped to the three RGB channels, with PC1, PC2, and PC3 corresponding to red, green, and blue, respectively.

For consistency, the coronal slice located at exactly half of the spatial dimension was selected for visualization, corresponding to slice 32 in the $64 \times 64 \times 64$ FPN output. 5.6

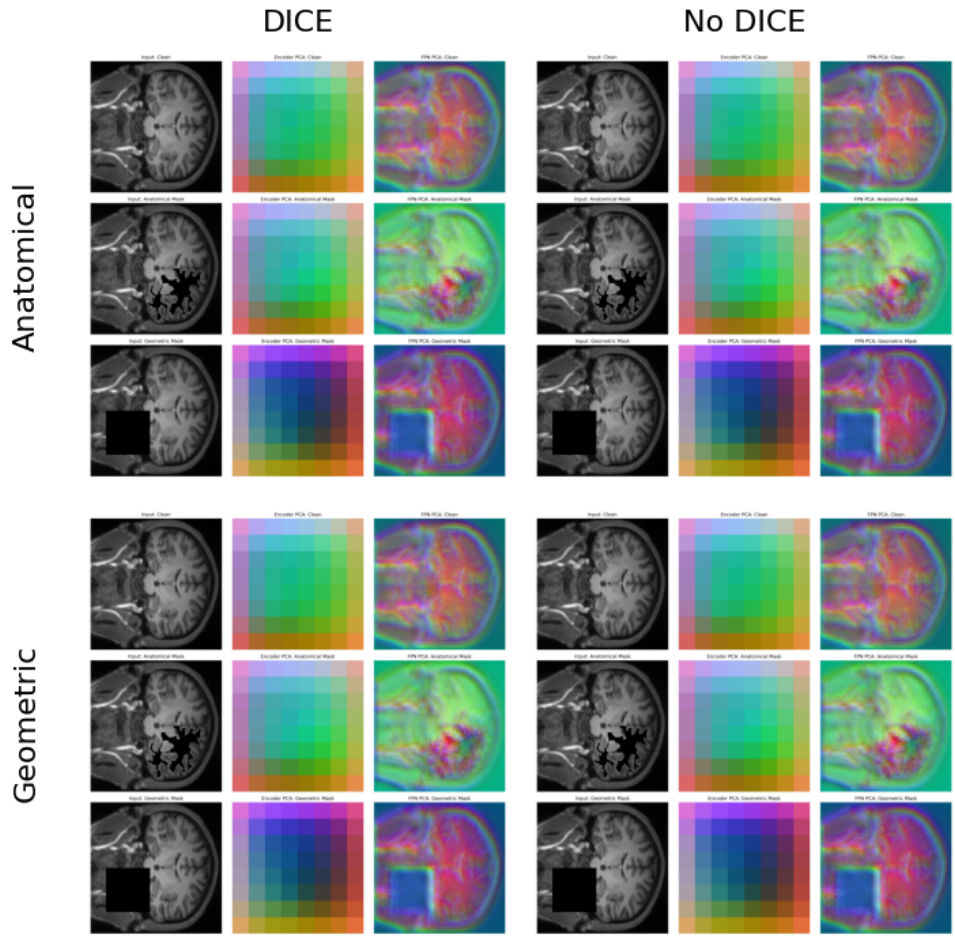


Figure 5.6: PCA visualization of the FPN output for the different masking configurations. The first three principal components are mapped to the RGB channels.

The resulting representations are highly similar across the six configurations. No substantial qualitative difference can be observed in the first three principal components of the FPN output, despite the different masking strategies and the presence or absence of the auxiliary loss.

This result is not unexpected. The comparison is performed at an intermediate stage of the architecture, before the Transformer and predictor, and only the first three PCA components are visualized. Moreover, the CNN backbone is not completely frozen, allowing its parameters to adapt during training, although the SynthSeg-based initialization strongly constrains the initial representation. Therefore, the similarity observed in the PCA visualization should not be interpreted as evidence that the six models learn identical representations, but rather that their

large-scale feature structure remains comparable at the FPN level.

An additional observation is the attenuation of the masking pattern in the resulting feature representation. In particular, small masked anatomical regions are not always clearly distinguishable in the FPN output. This behaviour suggests that the spatial downsampling introduced by the CNN-FPN pipeline can reduce the representation of small structures, making their contribution less evident at the reduced $64 \times 64 \times 64$ resolution.

5.4.2 Loss Evolution

The evolution of the loss components provides a clearer distinction between configurations trained with and without the auxiliary spatial loss5.7. As expected, the VICReg components show broadly similar trends across the two groups, since the VICReg objective and its weighting remain unchanged between the corresponding experiments.

The main difference is observed in the Dice-related metric. When the auxiliary Dice+BCE loss is included, the Dice score generally improves during training, indicating an increasing spatial correspondence between the predicted attention maps and the target anatomical maps. In contrast, when the auxiliary loss is removed, the Dice score does not exhibit the same systematic improvement instead decreases during training. This behaviour is expected because the Dice score is then used only as an evaluation metric and is not directly included in the optimization objective.

The masking strategy produces comparatively smaller differences in the overall loss evolution. The most visible variations are observed between the geometrical masking configurations and the anatomical masking configurations, particularly in the evolution of the Dice score. However, these differences are substantially smaller than those introduced by the presence or absence of the auxiliary spatial loss.

Overall, the loss evolution indicates that the auxiliary objective has a considerably stronger influence on the spatial behaviour of the model than the choice between the different image-level masking strategies.

5.4.3 Dice per ROI

The ROI-specific Dice scores provide a more detailed view of the spatial behaviour observed in the previous analysis5.8. The main distinction again arises from the presence or absence of the auxiliary spatial loss.

When the auxiliary loss is enabled, the Dice score generally increases across the different ROIs, although the magnitude of the improvement varies between anatomical structures. When the auxiliary loss is removed, the Dice score remains substantially lower for most ROIs and, in

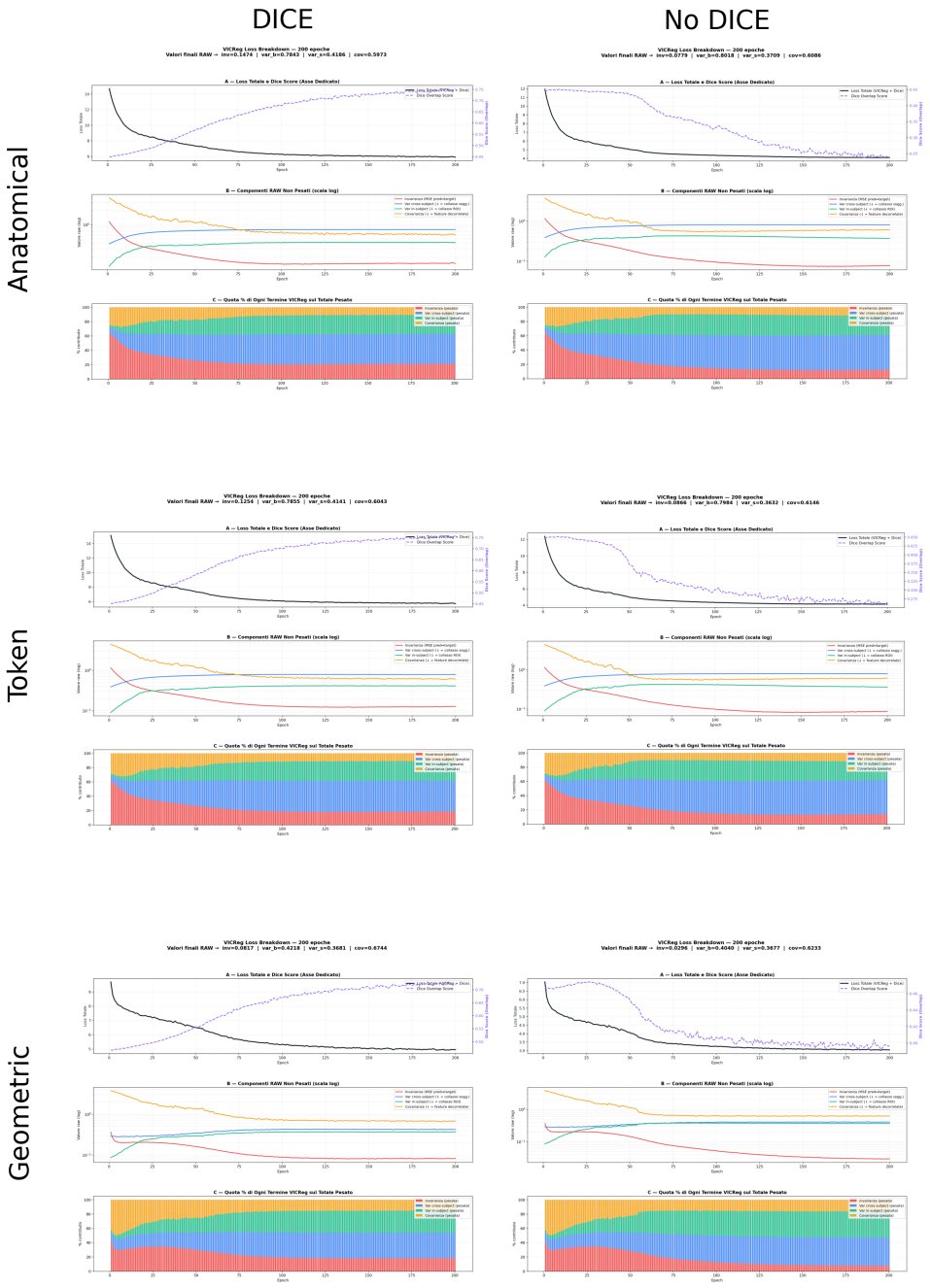


Figure 5.7: Comparison of loss evolution for anatomical image and geometrical image masking with and without the auxiliary Dice+BCE loss.

some cases, decreases during training. This behaviour is consistent with the optimization objective: without the Dice+BCE component, there is no direct loss term encouraging the attention maps to reproduce the anatomical target maps.

Consequently, the low Dice values observed in these configurations should not be interpreted as a failure of the representation-learning objective itself. Rather, they indicate that the learned representations are not explicitly constrained to maintain spatial overlap with the anatomical targets.

The differences between masking strategies are comparatively limited. No consistent pattern indicating that one masking strategy systematically produces substantially higher ROI-specific Dice scores can be identified from these experiments. The presence of the auxiliary spatial objective therefore appears to be the dominant factor affecting this metric.

The same behaviour can also be observed directly in the attention maps5.9. Four representative ROIs were selected to compare the spatial activation produced by the different configurations.

The configurations trained with the auxiliary spatial loss generally produce more localized attention patterns, whereas the configurations without this objective tend to exhibit more diffuse activation. This observation is consistent with the ROI-specific Dice analysis and further indicates that the auxiliary loss directly affects the spatial organization of the TokenLearner attention maps.

5.4.4 Token Similarity

The similarity between the learned ROI token representations was further investigated using cosine similarity matrices5.10. Cosine similarity measures the angular similarity between two vectors in the embedding space and is defined as:

$$\cos(x,y) = \frac{x \cdot y}{|x||y|}. \tag{5.1}$$

A value of 1 indicates that two vectors point in the same direction, whereas a value of 0 indicates orthogonality. A value of -1 indicates opposite directions in the embedding space. Therefore, values close to zero indicate low cosine similarity, but should not be interpreted directly as statistical independence or lack of correlation.

The similarity matrices show that the different training configurations lead to visibly different organizations of the token representations. The most notable deviation from the other configurations is observed for the geometrical masking model trained without the auxiliary spatial loss.

In this configuration, several groups of tokens exhibit pronounced positive or negative co-

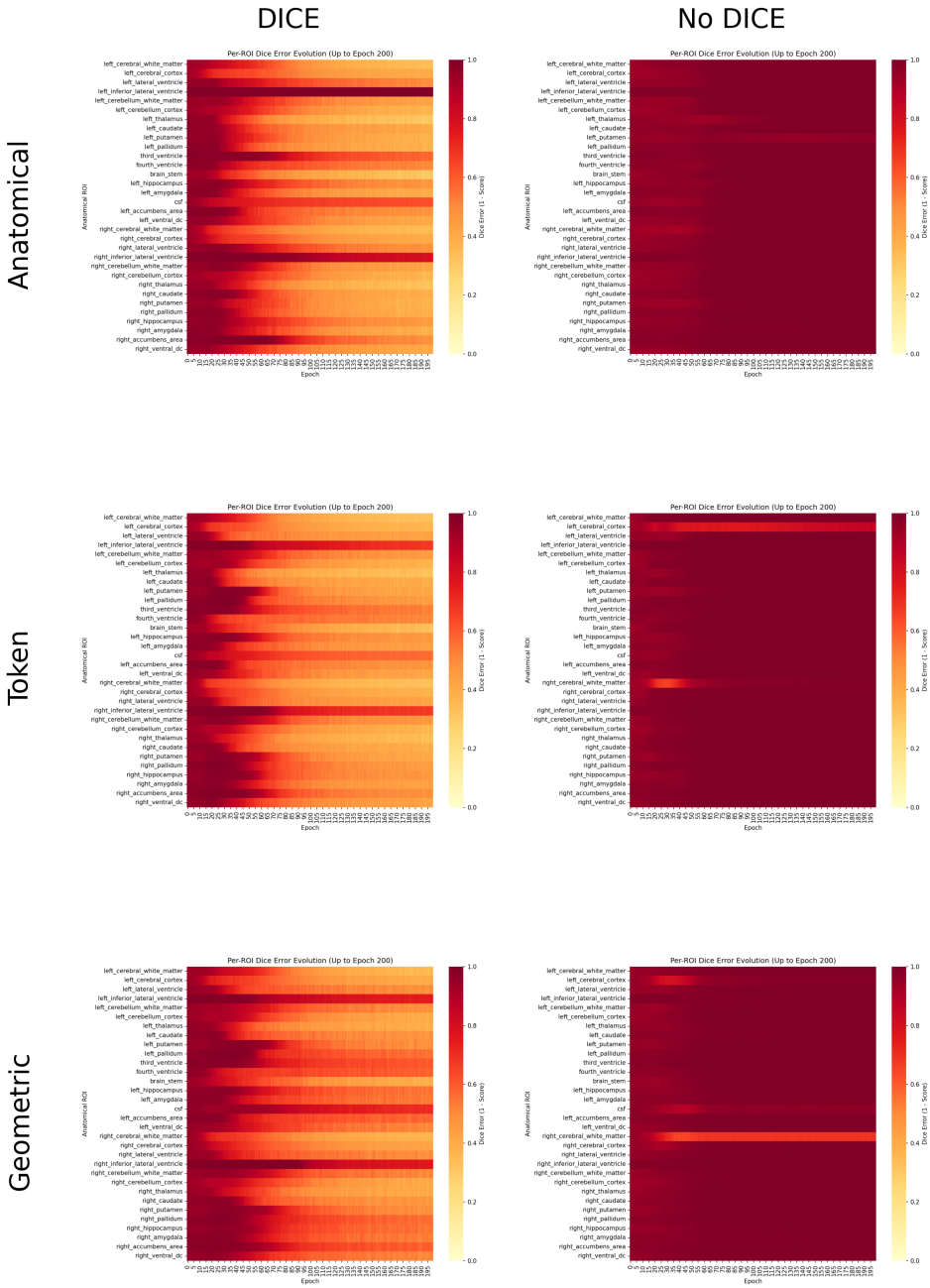


Figure 5.8: ROI-specific Dice score evolution for anatomical image, anatomical token and geometrical image masking with and without the auxiliary Dice+BCE loss.

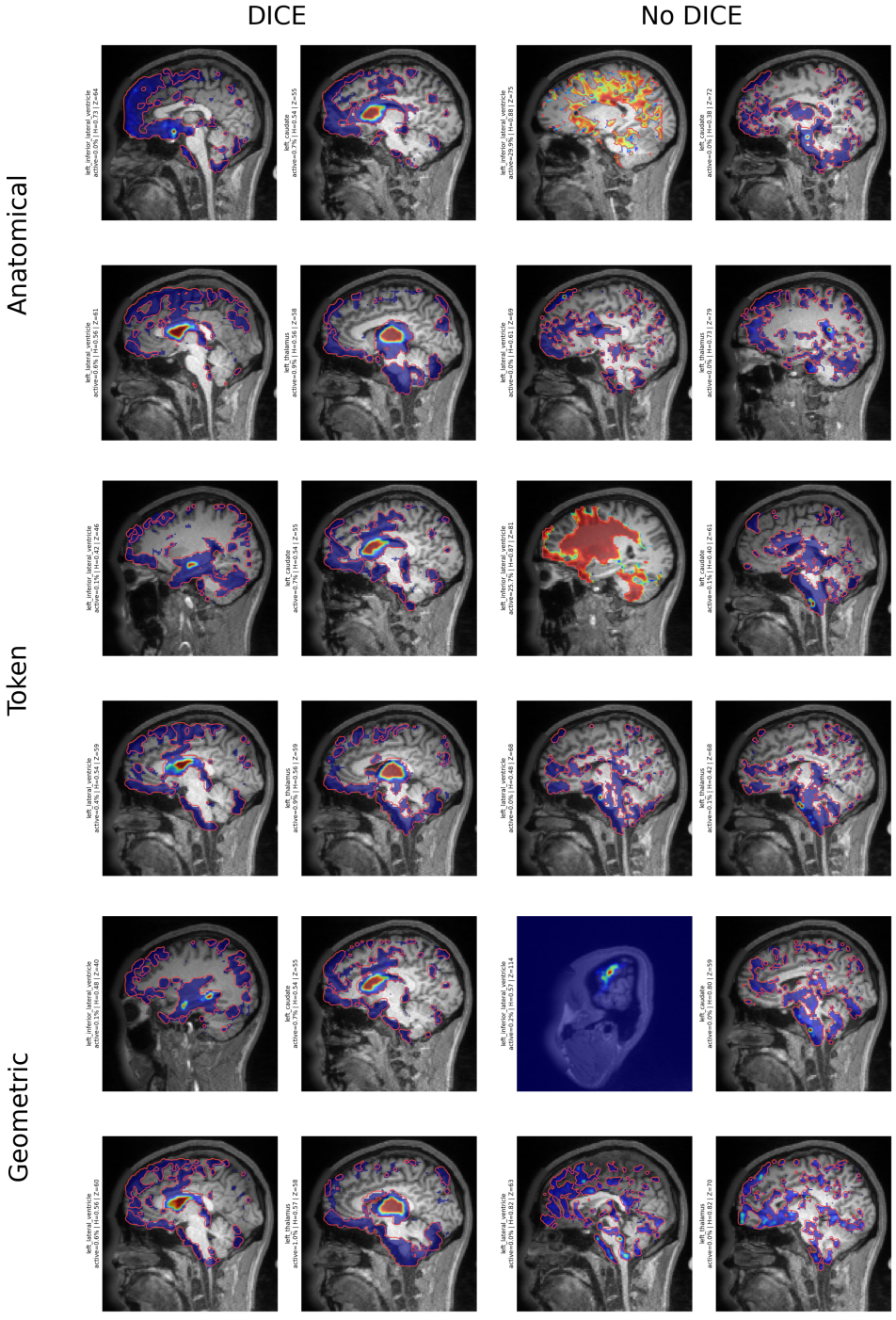


Figure 5.9: Comparison of attention maps for four representative ROIs across different masking and loss configurations.

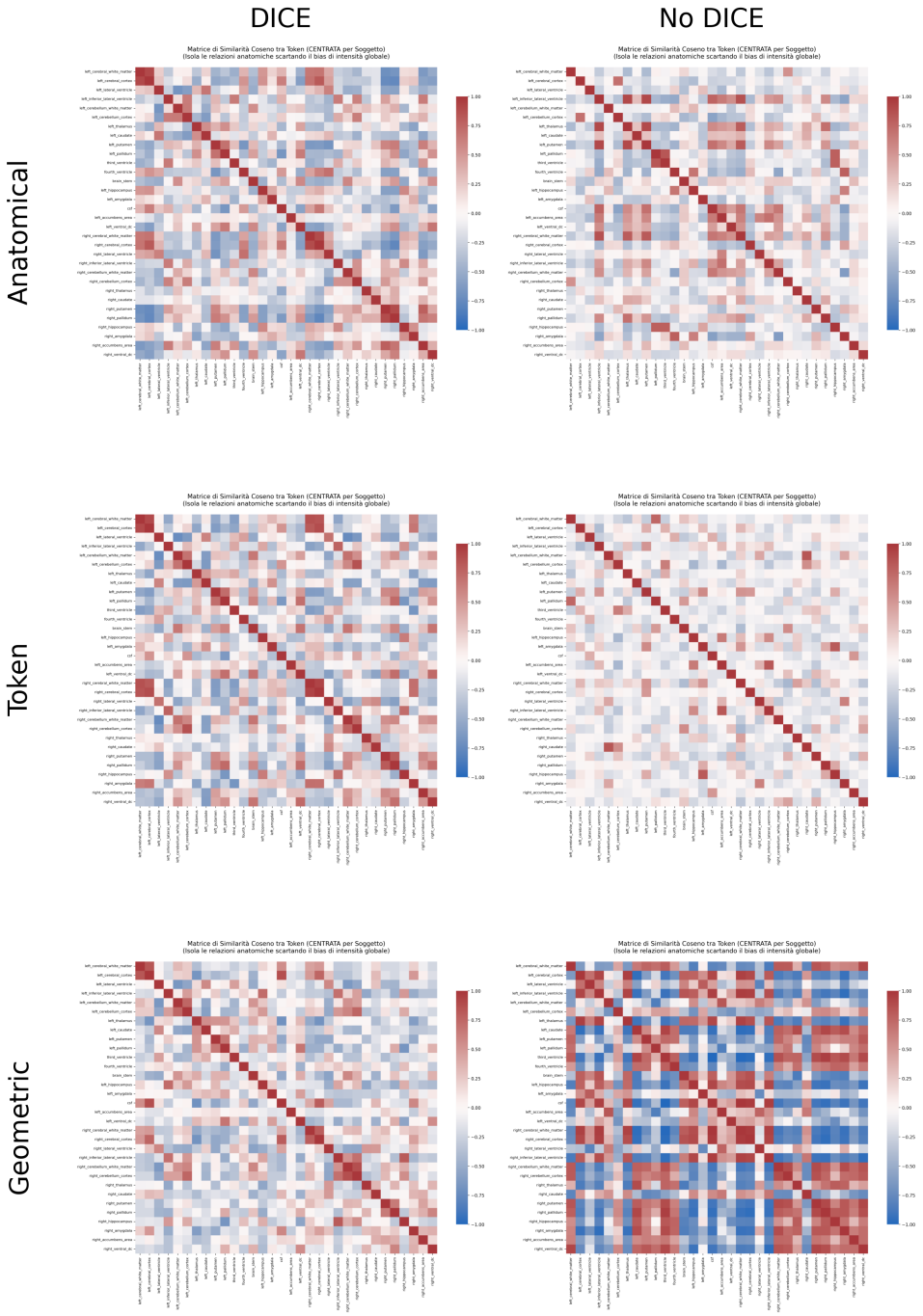


Figure 5.10: Cosine similarity matrices of the learned ROI token representations for the six ablation configurations.

sine similarity. In particular, some large anatomical regions show opposite directions in the embedding space depending on their hemispheric location. A similar tendency can be observed between some ipsilateral regions with substantially different spatial scales. However, these patterns are not sufficiently consistent across all ROIs to support a clear anatomical interpretation. The presence of several exceptions, together with differences between the left and right hemispheres, makes it difficult to associate these similarity patterns with a specific underlying anatomical principle.

For the remaining configurations, the similarity matrices show a more distributed pattern of pairwise relationships. In particular, many token pairs exhibit relatively low cosine similarity, suggesting that the representations occupy different directions in the embedding space. This behaviour is consistent with the role of the VICReg covariance and variance terms, which are designed to prevent redundant or collapsed representations.

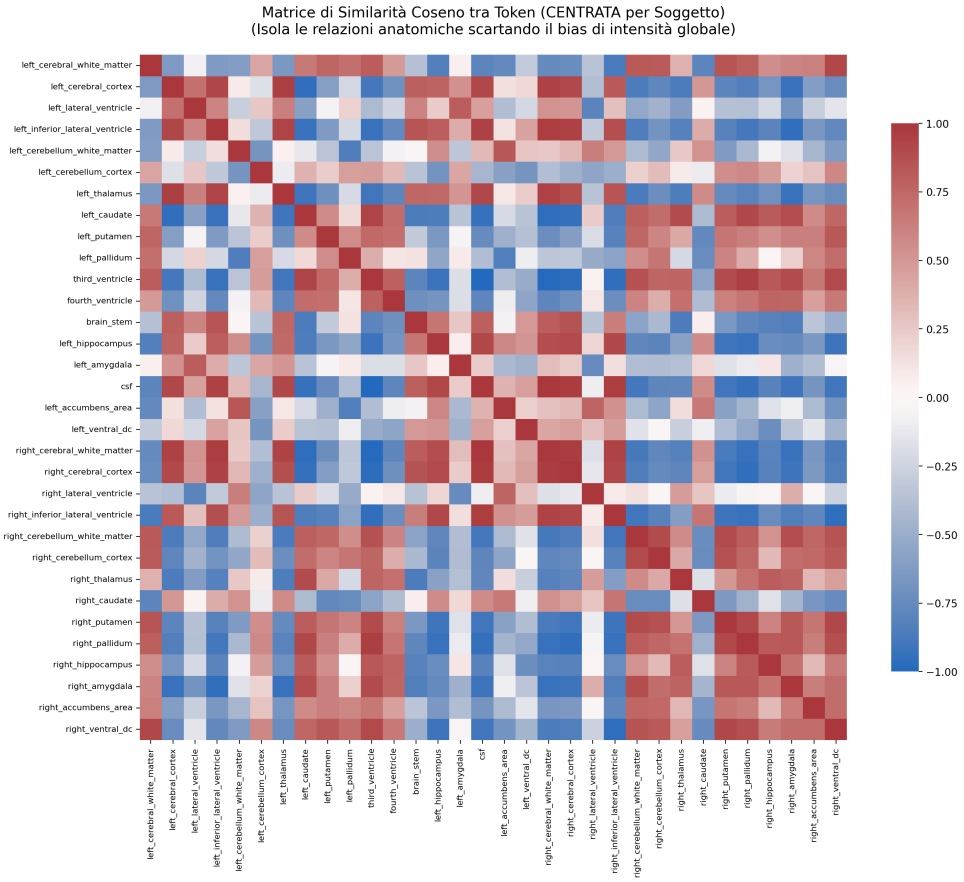


Figure 5.11: Detailed view of the cosine similarity matrix obtained with geometrical masking without the auxiliary spatial loss.

The anomalous structure observed in the geometrical masking configuration without the auxiliary loss therefore represents an interesting difference between the models, but its precise interpretation remains uncertain. In the absence of additional experiments, it cannot be conclu-

sively attributed to a specific anatomical or optimization mechanism.

5.4.5 Latent Space Organization

The organization of the learned token representations was further investigated using t-SNE (t-distributed Stochastic Neighbor Embedding)5.12. t-SNE is a dimensionality-reduction technique that maps high-dimensional representations into a low-dimensional space while attempting to preserve local neighbourhood relationships. The resulting visualization can therefore be used to qualitatively inspect whether groups of representations occupy distinct regions of the embedding space.

Among the configurations, geometrical masking without the auxiliary spatial loss produces the most visibly separated groups, with relatively few representations concentrated in the central region of the embedding. The remaining configurations exhibit more dispersed distributions and a greater degree of overlap between groups.

This visualization suggests that the different training configurations organize the token representations differently in the latent space. However, because t-SNE is primarily a visualization method and its output depends on the dimensionality-reduction procedure and its hyperparameters, the apparent separation of groups should not be interpreted directly as evidence of superior clustering performance. In particular, no quantitative clustering metric was computed as part of this analysis.

5.4.6 Information Carried by Tokens

The final analysis investigates how strongly individual tokens are associated with the different anatomical ROIs. Ideally, if each token learned a highly specific anatomical role, the resulting token-to-ROI representation would exhibit a predominantly diagonal structure, or clearly defined groups, against a relatively uniform background.

The obtained representations instead show a substantial degree of overlap between tokens and ROIs 5.13. No configuration produces a strictly one-to-one correspondence between a token and an individual anatomical region. This indicates that the learned representation remains distributed: a given ROI can contribute to multiple tokens, while individual tokens can simultaneously encode information associated with several ROIs.

To further investigate this distribution, the token-to-ROI matrix was normalized first along the rows 5.14 and then along the columns5.15. The two normalizations provide complementary views of the same relationship. Row normalization emphasizes how the information associated with an individual ROI is distributed across the available tokens, whereas column normalization highlights which ROIs contribute most strongly to each token.

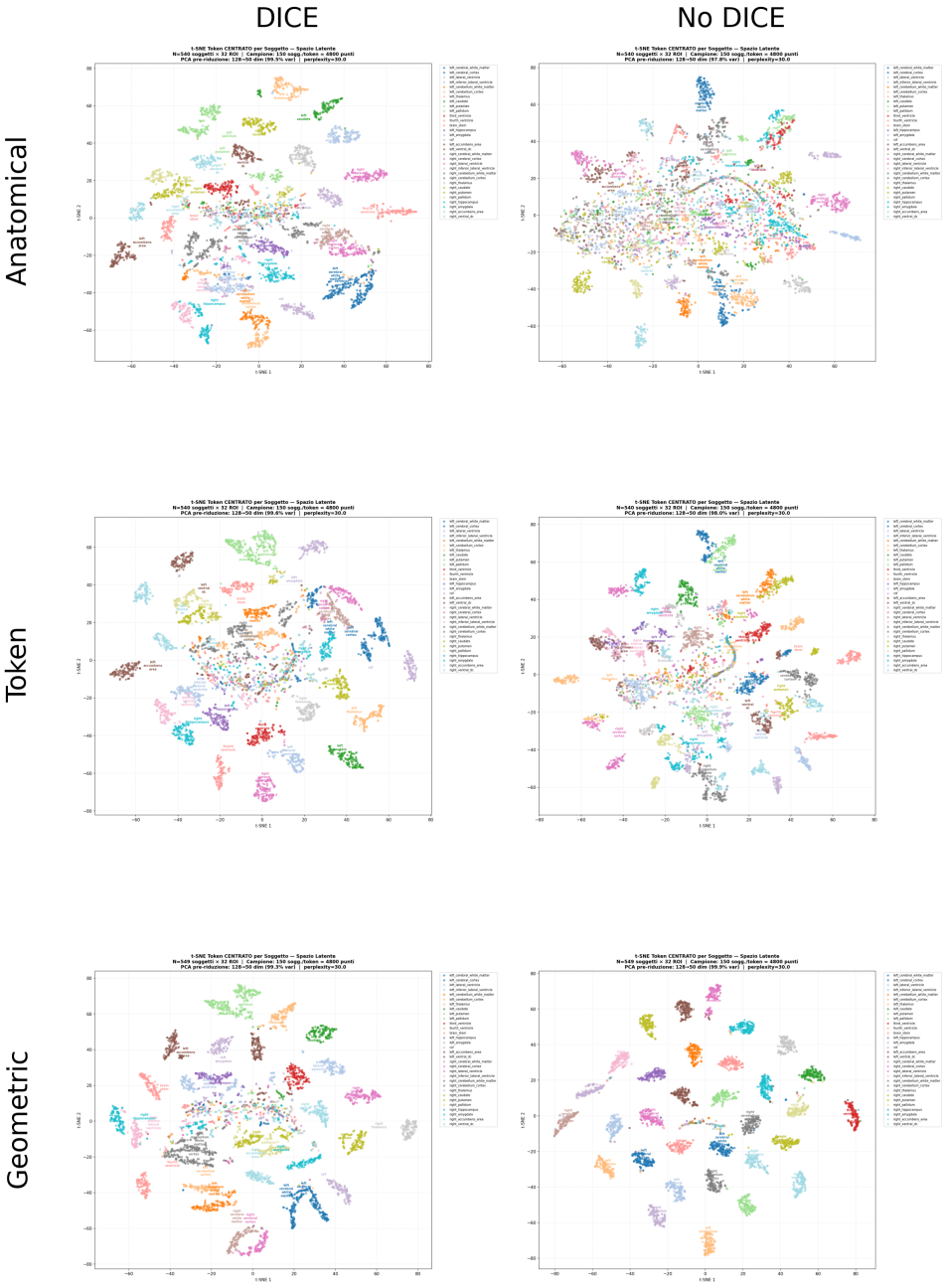


Figure 5.12: t-SNE visualization of the token representations for the different masking and auxiliary-loss configurations.

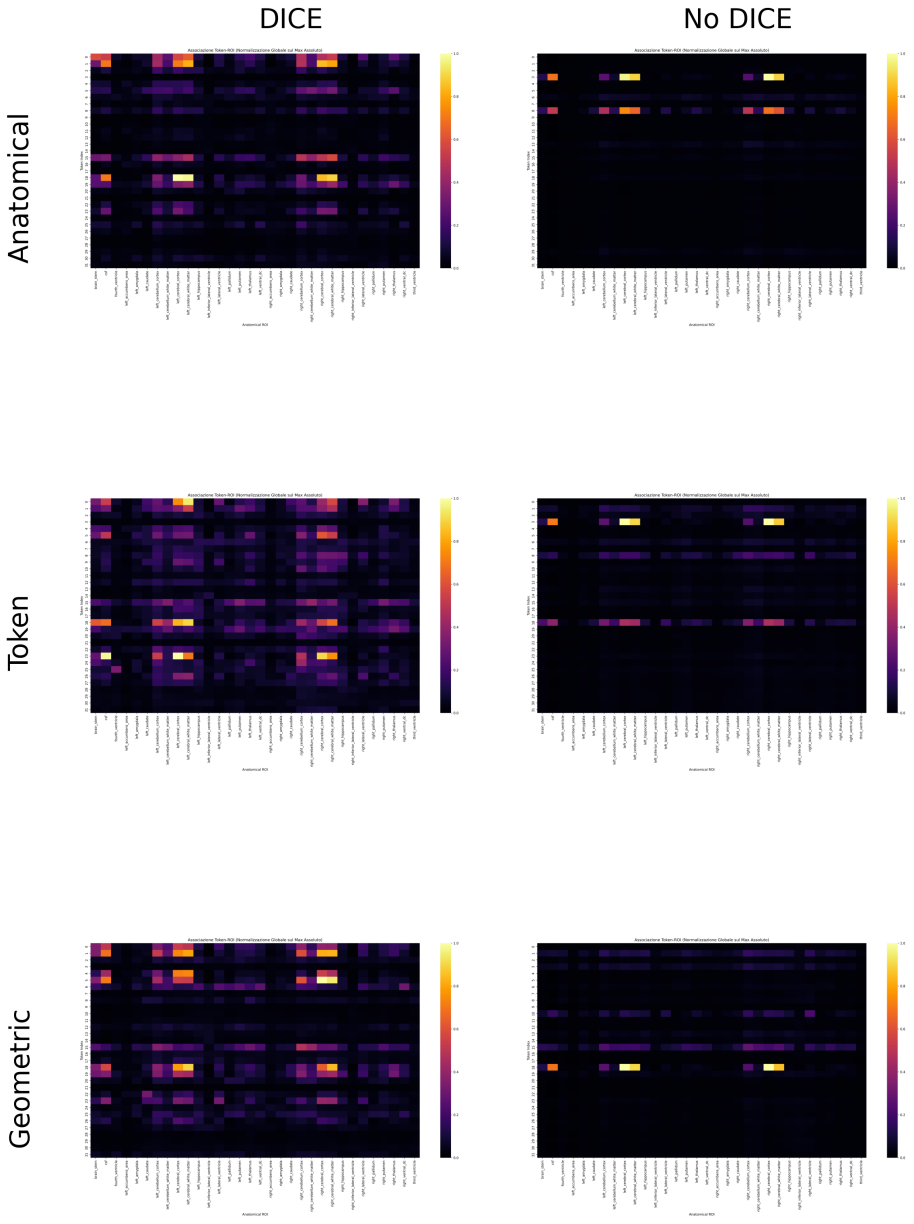


Figure 5.13: Visualization of token-to-ROI information for the different masking and auxiliary-loss configurations, normalized on the maximum value of the graph, for better visualization.

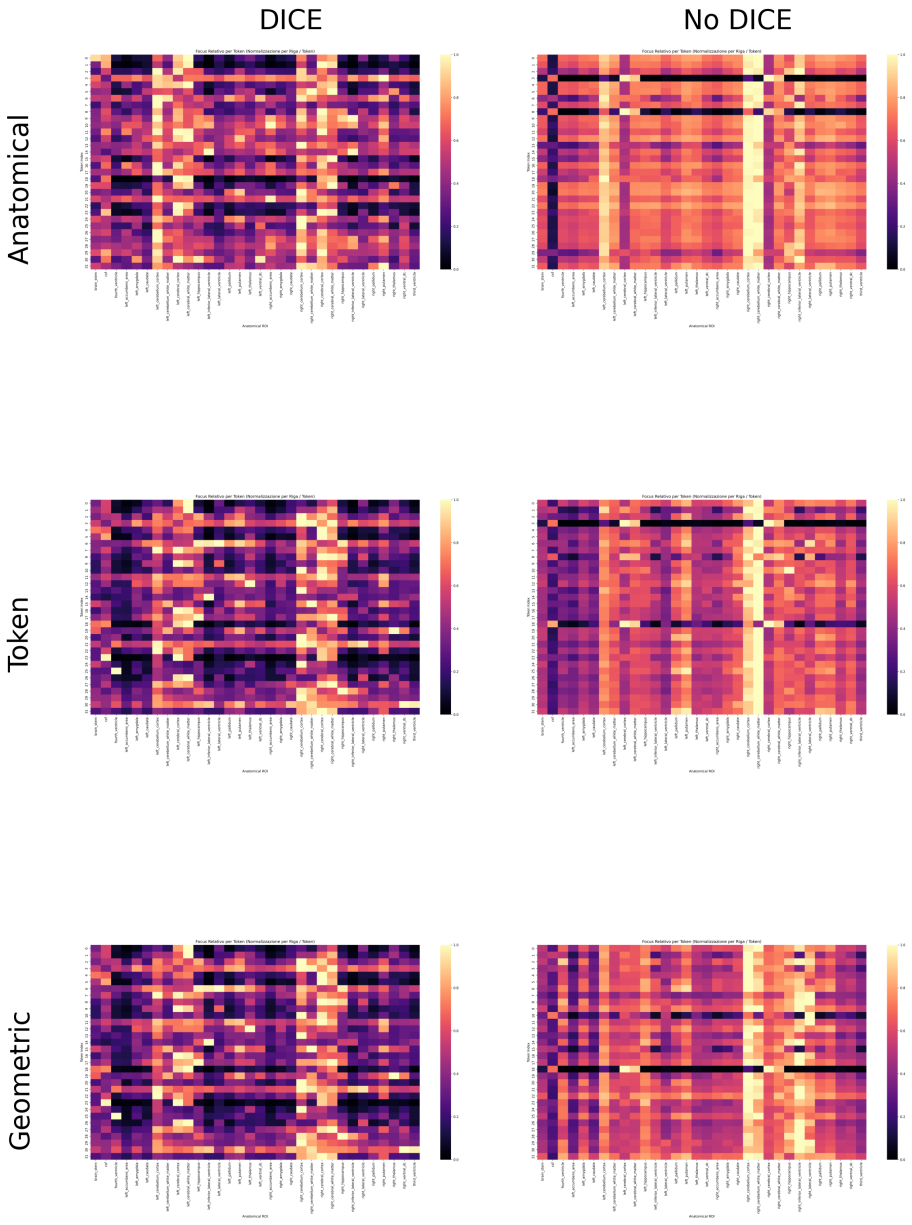


Figure 5.14: Visualization of token-to-ROI information for the different masking and auxiliary-loss configurations, normalized by rows therefore representing which ROI are more represented in the Token.

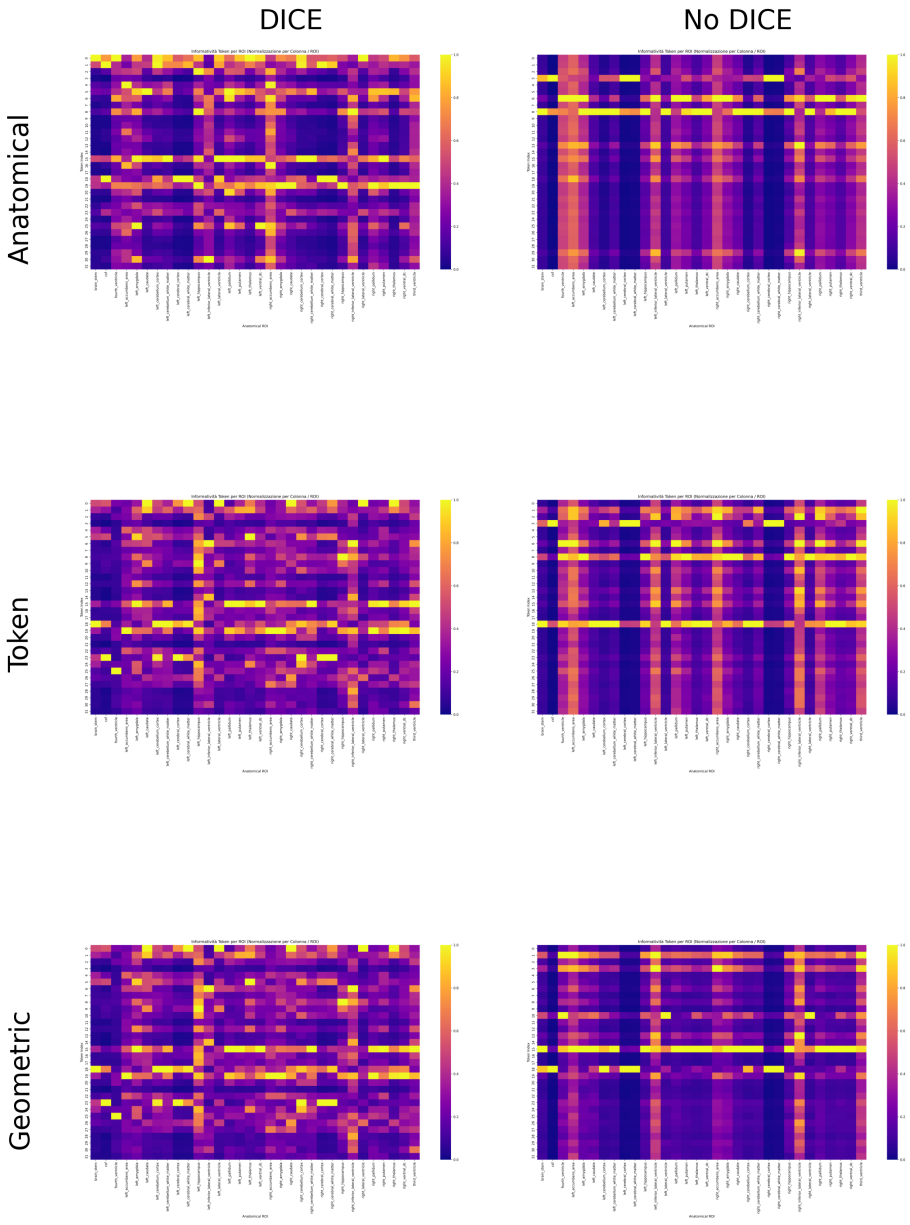


Figure 5.15: Visualization of token-to-ROI information for the different masking and auxiliary-loss configurations, normalized by columns therefore representing which tokens contribute the most to the representetion of the ROI.

The resulting matrices show a recurring difference between configurations trained with and without the auxiliary spatial loss. In the absence of the auxiliary objective, some tokens appear to accumulate a disproportionately large contribution from several ROIs. This suggests that part of the anatomical information may become concentrated in a smaller subset of the available tokens.

The row-normalized representations provide a complementary observation. Larger anatomical regions tend to contribute to multiple tokens, while the information associated with smaller regions can become less prominent. This effect is less pronounced in the configurations trained with the auxiliary Dice+BCE loss, where the contribution of the anatomical regions appears to be distributed more evenly across the token sequence.

These observations suggest that the auxiliary spatial loss influences not only the spatial localization of the attention maps but also the distribution of anatomical information across the learned token representations. Nevertheless, the results do not demonstrate a strict one-token-per-ROI correspondence. Instead, the learned representation remains distributed across multiple tokens, with the auxiliary objective encouraging a more balanced spatial allocation of the available token capacity.

Chapter 6

Conclusions

6.1 Considerations based on the Results

This thesis presented the development and analysis of an anatomical-aware self-supervised learning framework for the extraction of structured representations from three-dimensional brain MRI. The proposed architecture combines a SynthSeg-based 3D convolutional backbone and Feature Pyramid Network (FPN), TokenLearner-based spatial tokenization, and a JEPA-inspired encoder architecture. The main objective was to investigate whether meaningful anatomical representations could be learned from volumetric MRI data without relying on manually annotated training datasets, while retaining an explicit correspondence between the spatial information contained in the input volumes and the resulting token representations.

Two main training configurations were investigated. The first followed the original objective of the project and relied exclusively on the self-supervised JEPA/VICReg objectives, without an auxiliary anatomical loss. The second introduced an additional spatial loss based on Dice and Binary Cross-Entropy (BCE), using the anatomical information provided by SynthSeg to guide the spatial attention maps produced by TokenLearner. Although the latter configuration introduces an anatomically supervised component and therefore departs from a strictly self-supervised formulation, its inclusion produced a substantial improvement in the spatial localization of the learned representations. This result highlights an important trade-off between the degree of supervision and the anatomical interpretability of the resulting representations.

The use of an automatically generated anatomical segmentation represents an interesting intermediate solution between fully supervised and fully self-supervised learning. In the proposed pipeline, anatomical labels do not need to be manually annotated by experts, but can instead be generated automatically through SynthSeg. Consequently, the additional supervision does not require the creation of a dedicated manually labelled training cohort. The main cost is shifted from manual annotation to preprocessing time, since each MRI volume must undergo

segmentation before being used by the model. In the experiments performed in this thesis, this preprocessing step required approximately 1/2 minutes per image. While this computational cost is not negligible, it may be considerably more manageable than the manual annotation of large-scale neuroimaging datasets.

One of the main objectives of the proposed architecture was to establish a clear correspondence between individual tokens and anatomical regions. This objective was not completely achieved. The analysis of the token-to-ROI relationships showed that the learned representations do not exhibit a strict one-to-one mapping between tokens and anatomical structures. Instead, anatomical information is distributed across multiple tokens, with individual tokens often incorporating information originating from several ROIs. Therefore, the results do not support the interpretation of each token as an independent anatomical entity.

Nevertheless, the results obtained with the auxiliary spatial loss are encouraging. In particular, the attention maps and the ROI-level Dice analysis demonstrate that the model is capable of concentrating its spatial attention on specific anatomical structures when explicitly guided by anatomical information. This is an important property for the intended use of the representation, even in the absence of a perfect token-to-ROI correspondence. Rather than producing a rigid anatomical decomposition, the current model appears capable of learning distributed representations in which anatomical information is encoded across a set of spatially meaningful tokens.

The ROI-level Dice analysis further supports this interpretation. Most anatomical regions showed improved localization when the auxiliary Dice+BCE loss was introduced, while the two inferior lateral ventricles represented the most challenging structures. These two regions are also the smallest ROIs considered in the adopted segmentation scheme. Their lower performance is therefore consistent with the limitations imposed by the spatial resolution of the feature representation. The analysis also showed that other relatively small structures, such as the accumbens area, could eventually be learned by the model. This suggests that the difficulty observed for the inferior lateral ventricles may be related at least in part to their limited spatial extent rather than to a general inability of the architecture to represent small anatomical structures.

The computational constraints encountered throughout the project played a substantial role in shaping both the architecture and the experimental protocol. The use of a single NVIDIA A30 GPU with 24 GB of VRAM required several compromises, including reduced spatial resolution, constrained batch sizes, gradient accumulation, selective checkpointing, and the restriction of the experimental cohort. These limitations were particularly relevant when processing three-dimensional volumes, where memory requirements increase rapidly with spatial resolution and feature dimensionality. As a consequence, some architectural choices that would ideally have been investigated more extensively could only be explored under constrained conditions.

The hardware limitations also affected the breadth of the experimental analysis. The training sessions required several hours each, making extensive hyperparameter searches and repeated experiments impractical. The ablation study was therefore primarily designed to investigate the internal behaviour of the architecture and the effect of its principal components rather than to provide a definitive benchmark of generalization performance. In particular, the absence of independent validation and test sets means that the reported results should be interpreted as an analysis of the current training configuration and learned representations rather than as evidence of performance on unseen data.

Despite these limitations, the experiments provided useful evidence regarding the behaviour of the individual components of the architecture. The CNN/FPN component maintained a stable representation of the input volumes, while TokenLearner represented the most important stage for converting spatial feature maps into the structured token representation used by the subsequent Transformer-based components. The loss formulation, in turn, had a particularly strong influence on the spatial properties of the learned representations. The combination of VICReg with the auxiliary Dice+BCE loss produced more localized anatomical attention than the strictly self-supervised configuration, highlighting the importance of the learning objective in determining what information is retained by the representation.

Overall, the work can be regarded as a proof of concept for an anatomical-aware representation learning framework for three-dimensional brain MRI. The results indicate that the proposed architecture can learn spatially meaningful representations and can exploit automatically generated anatomical information to improve the localization of its learned tokens. At the same time, the experiments also show that the current implementation remains far from a final formulation suitable for large-scale deployment. The lack of a strict token-to-ROI correspondence, the limited cohort size, the absence of independent evaluation sets, and the computational constraints all represent important limitations that should be addressed in future work.

6.2 Next Steps

The next stage of development should focus on evaluating the architecture under experimental conditions that are closer to those required for its eventual use within the HEREDITARY project. In particular, future experiments should involve a substantially larger cohort, dedicated validation and test sets, higher spatial resolutions, and more extensive computational resources. These changes would make it possible to distinguish improvements that are specific to the current training configuration from properties that generalize to unseen subjects and acquisition conditions.

Furthermore, future work should investigate the representation at multiple levels of the ar-

chitecture rather than evaluating only the final token space. The current results suggest that the CNN/FPN, TokenLearner, and loss components play different roles in determining the final representation. A more systematic analysis of these components could therefore help identify which architectural properties are essential for anatomical localization and which are mainly consequences of the current training configuration.

6.2.1 Dataset

The cohort used in the experiments consisted of 550 selected subjects, with 540 subjects retained after quality-control filtering. This cohort was intentionally curated to provide acquisition diversity while remaining computationally manageable, but it is considerably smaller than the scale of the original FOMO300K dataset. Increasing the number of subjects should therefore be one of the first priorities of future experiments. A cohort on the order of 10^4 subjects would provide a substantially broader range of anatomical variability and acquisition conditions, allowing the model to learn from a much richer distribution of brain morphology.

Increasing the dataset size would also make it possible to investigate whether the representations learned in the current experiments remain stable when exposed to previously unseen acquisition sources. This is particularly relevant because the original dataset exhibited differences in acquisition-source distribution, and the preliminary analysis showed that such differences can contribute substantially to the overall variance of the data. A larger and more diverse cohort could therefore be combined with a more systematic strategy for controlling acquisition-related variability.

Another important extension concerns the multimodal nature originally envisioned for the project. The current implementation operates exclusively on MRI volumes, whereas the broader objective of the HEREDITARY project involves integrating heterogeneous biomedical information. Extending the representation learning framework beyond a single imaging modality would therefore be a natural direction for future development.

However, multimodal integration cannot be achieved simply by adding additional input channels. The current anatomical component of the pipeline relies heavily on SynthSeg, which is specifically designed for brain MRI segmentation and cannot be directly applied to arbitrary biomedical modalities. For example, modalities such as near-infrared spectroscopy (NIRS) do not provide a three-dimensional anatomical volume that can be processed by SynthSeg in the same manner as MRI. Consequently, extending the framework to heterogeneous modalities would require a more general strategy for representing modality-specific information while preserving a shared anatomical or semantic representation space.

One possible direction would therefore be to separate modality-specific feature extraction from the subsequent shared representation-learning stages. In such a configuration, each modal-

ity could be processed by an appropriate encoder, while the resulting representations could then be projected into a common latent space. This would reduce the dependence of the overall architecture on MRI-specific preprocessing tools and could provide a more flexible basis for multimodal integration.

Finally, the introduction of dedicated training, validation, and test sets represents a necessary methodological improvement. The current experiments were primarily designed to study training dynamics and internal representations, rather than to estimate generalization to unseen data. A future experimental protocol should therefore reserve independent subjects for validation and testing, with the split performed at the subject level to avoid information leakage between different acquisitions from the same individual. This would provide a more reliable assessment of whether the learned representations transfer beyond the subjects used during training.

6.2.2 3D CNN and Feature Pyramid Network

The 3D CNN/FPN component represents one of the more stable parts of the current architecture. The convolutional backbone is based on the SynthSeg encoder and was initialized using pretrained weights, allowing the model to exploit representations learned from a substantially broader segmentation task. The encoder was subsequently kept frozen in the main experiments. This choice was motivated both by the availability of pretrained representations and by computational considerations: unfreezing the backbone would introduce millions of additional trainable parameters and would substantially increase both memory consumption and training time.

Under more favorable computational conditions, a direct comparison between a frozen and an unfrozen SynthSeg backbone would nevertheless be valuable. Such an experiment could determine whether the pretrained anatomical representations are sufficiently general for the proposed representation-learning task or whether further adaptation to the specific distribution of the FOMO300K data provides measurable benefits. This comparison would be particularly informative if performed together with an analysis of the learned feature maps at different depths of the CNN.

The FPN itself introduces a relatively limited number of additional trainable parameters compared with the convolutional backbone, as much of its operation consists of lateral projections, interpolation, and feature-map fusion. Its main role is to combine information from different spatial resolutions and produce a feature representation that retains both semantic information from deeper layers and spatial information from shallower layers.

Nevertheless, the behaviour of the FPN and its interaction with the backbone deserve further investigation. As observed in the Results section, the feature representations extracted at the FPN level showed highly similar large-scale patterns across the different training configurations. This behaviour is not necessarily problematic, since the FPN precedes the components

that are most directly affected by the different masking and loss configurations. However, the consistency of these representations raises the question on how impactful could it actually be to unfreeze the 3DCNN.

A useful future experiment would therefore be to systematically compare representations extracted before and after the FPN under different training conditions. Quantitative similarity measures, feature-space analyses, and controlled freezing/unfreezing experiments could help determine whether the observed stability reflects useful preservation of anatomical information or whether the feature extractor is insufficiently adapting to the representation-learning objective.

Another important direction would be to investigate higher-resolution feature maps. The current experiments required a reduction of the spatial resolution of the representation because of memory constraints. Returning to the full $128 \times 128 \times 128$ resolution would increase the number of spatial locations available to TokenLearner and could be particularly beneficial for small anatomical structures. This change, however, would also substantially increase memory requirements and would therefore need to be evaluated together with appropriate optimization strategies or more powerful hardware.

6.2.3 TokenLearner

TokenLearner represents one of the most important components of the proposed architecture because it performs the transition from dense three-dimensional feature maps to the compact token representation processed by the subsequent Transformer. Its behaviour therefore has a direct influence on both the information retained by the model and the degree to which the resulting tokens can be interpreted anatomically.

The experiments showed that the current TokenLearner configuration is capable of learning spatially localized attention when guided by the auxiliary anatomical loss. However, the absence of a strict one-to-one mapping between tokens and ROIs indicates that the current tokenization mechanism does not naturally produce explicitly interpretable anatomical tokens. Instead, the learned representation is distributed, with multiple tokens contributing to the representation of individual ROIs and individual tokens potentially incorporating information from multiple anatomical structures.

This behaviour is not necessarily inconsistent with the objective of representation learning, since distributed representations can encode information more flexibly than a rigid one-token-per-ROI formulation. Nevertheless, it complicates the interpretation of individual tokens, particularly in a biomedical context where understanding which anatomical structures contribute to a representation can be important.

A major direction for future work should therefore be a more systematic investigation of

TokenLearner itself. Different numbers of tokens, attention-map parameterizations, temperature values, and masking strategies could be evaluated to determine how these parameters affect spatial specialization. In addition, alternative tokenization mechanisms could be considered to establish whether the observed distributed representation is intrinsic to the task or mainly a consequence of the current TokenLearner formulation.

Explainability is particularly relevant in this context. The current attention maps provide a useful visualization of where the model is focusing, but an attention map alone does not necessarily establish that the corresponding region is the causal source of a learned representation. Future experiments could therefore combine attention visualization with perturbation-based analyses, in which individual ROIs or tokens are selectively removed and the resulting changes in the latent representation are measured. Such analyses could provide stronger evidence about the relationship between anatomical structures and learned tokens.

The TokenLearner could consequently become one of the main subjects of future architectural studies, both as a component to optimize and as a component to replace. Comparing it with alternative spatial pooling or tokenization mechanisms would help determine whether the current formulation provides a specific advantage for three-dimensional anatomical representation learning.

6.2.4 Loss

The loss function represents another major area for future investigation because it directly determines the properties that the model is encouraged to preserve in its latent representation. The experiments showed that the introduction of the auxiliary Dice+BCE loss had a substantially stronger effect on anatomical localization than changes in the VICReg weighting alone. This indicates that the learning objective is closely connected to the spatial organization of the resulting representation.

Further incremental modifications to the current loss formulation should nevertheless be approached carefully. Excessive tuning of the individual loss weights could lead to an increasingly specialized configuration that performs well under the specific experimental conditions used in this thesis but is less informative about the general properties of the architecture. Rather than continuously optimizing the existing combination of terms, a more informative direction would therefore be to compare alternative loss formulations under the same experimental protocol.

The current VICReg + Dice+BCE configuration can be considered the reference configuration obtained in this study, against which alternative objectives could be compared. Possible future experiments could investigate different combinations of invariance, variance, and covariance regularization, as well as alternative spatial objectives. The goal would not simply be to maximize an individual metric, but to determine how different objectives affect the balance

between representation diversity, anatomical localization, and spatial interpretability.

An especially relevant question is whether the auxiliary anatomical loss can eventually be reduced or removed once the model has learned a sufficiently strong anatomical representation. For example, future experiments could investigate curriculum-based strategies in which anatomical supervision is used only during an initial stage of training and progressively reduced thereafter. Such approaches could potentially retain some of the localization benefits observed in the current experiments while moving the final representation closer to the original self-supervised objective.

6.2.5 Final Considerations

The work presented in this thesis provides an initial framework for anatomical-aware representation learning from three-dimensional brain MRI and demonstrates that the proposed architecture can learn spatially meaningful representations under constrained experimental conditions. The results also highlight the importance of the training objective: while the strictly self-supervised formulation represents the original methodological goal, the addition of an auxiliary anatomical loss substantially improved the localization of the learned representations.

At the same time, several aspects remain open. The current tokens do not correspond uniquely to individual anatomical structures, the cohort is limited compared with the scale of the source dataset, and the absence of independent validation and test sets prevents a complete assessment of generalization. Moreover, the computational limitations imposed restrictions on spatial resolution, batch size, training duration, and the number of experiments that could be performed.

These limitations also define the most direct path for future development. More powerful computational resources would allow larger cohorts, higher-resolution feature maps, longer training schedules, and more extensive ablation studies to be investigated simultaneously. In particular, increasing the spatial resolution of the feature representation and expanding the number of training subjects could provide additional information for the learning of small anatomical structures, which represented one of the main challenges observed in the current experiments.

From a broader perspective, the most important future step is to move from the current proof of concept toward a systematic evaluation of the learned representations. This requires testing the model on unseen subjects, evaluating the stability of the learned features across acquisition conditions, and determining whether the resulting latent representation can support downstream clinical or research tasks. Such evaluation would provide the necessary evidence to establish how useful the architecture can be within the wider HEREDITARY framework.

The model developed in this thesis should therefore be considered not as a final solution, but as a foundation for further investigation. Its current results demonstrate that automatically

derived anatomical information can be integrated with self-supervised representation learning to obtain spatially structured representations from volumetric MRI. Further work will be required to determine how robustly these representations generalize, how interpretable individual tokens can become, and how the framework can ultimately be extended to the heterogeneous multimodal data envisioned by the HEREDITARY project.

Bibliography

[1] WorldHealthOrganization, "Global status report on neurology," 2025.

[2] T. W. H. B. et al., "Current clinical applications of structural mri in neurological disorders," 2025. eprint: https://doi.org/10.3988/jcn.2025.0185.

[3] B. C. R. A. et al., "Choosing wisely in neuroradiology: Evaluating ct and mr utilization trends in europe," 2026. eprint: doi:10.1007/s00330-026-12528-1.

[4] S. E. P. O, "Ai in mri: Computational frameworks for a faster, optimized, and automated imaging workflow," 2023. eprint: doi:10.3390/bioengineering10040492.

[5] H. G. V. T. et al., "Impact of ai assistance on radiologist interpretation of knee mri," 2026. eprint: doi:10.1007/s00330-025-11820-w.

[6] C. S. M. A. et al., "A large-scale heterogeneous 3d magnetic resonance brain imaging dataset for self-supervised learning," 2026. eprint: doi:10.1038/s41597-026-07688-0.

[7] V. A. E. F. S. S., "History of magnetic resonance imaging: A trip down memory lane," 2021. eprint: doi:10.1016/j.neuroscience.2021.06.038.

[8] G. V. T. J. et al., "Magnetic resonance imaging: Principles and techniques: Lessons for clinicians," 2015. eprint: doi:10.1016/j.jceh.2015.08.001.

[9] S. C. et al., "A large-scale heterogeneous 3d magnetic resonance brain imaging dataset for self-supervised learning," 2026. eprint: doi:10.48550/arXiv.2506.14432.

[10] A. A. D. V. Sawlani, "Diffusion-weighted imaging of the brain: Beyond stroke," 2017. eprint: doi:10.1016/j.carj.2016.10.001.

[11] radiopaedia, "T1 weighted image," 2024. eprint: https://radiopaedia.org/articles/t1-weighted-image.

[12] radiopaedia, "T2-weighted image," 2025. eprint: https://radiopaedia.org/articles/t2-weighted-image-1.

[13] B. B. G. D. et al., "Synthseg: Segmentation of brain mri scans of any contrast and resolution without retraining," 2023. eprint: doi:10.1016/j.media.2023.102789.

[14] M. D. Z. R. Fergus, “Visualizing and understanding convolutional networks,” 2013. eprint: doi:10.48550/arXiv.1311.2901.

[15] K. S. A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2015. eprint: doi:10.48550/arXiv.1409.1556.

[16] C. S. W. L. et al., “Going deeper with convolutions,” 2015. eprint: doi:10.1109/CVPR.2015.7298594.

[17] X. G. Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” 2010. eprint: pp.249-256.

[18] S. I. C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” 2015. eprint: doi:10.48550/arXiv.1502.03167.

[19] S. G. N. Komodakis, “Object detection via a multi-region & semantic segmentation-aware cnn model,” 2015. eprint: doi:10.48550/arXiv.1505.01749.

[20] K. H. X. Z. et. al., “Deep residual learning for image recognition,” 2015. eprint: doi:10.48550/arXiv.1512.03385.

[21] T.-Y. L. P. Dollár; “Feature pyramid networks for object detection,” 2016. eprint: doi:10.48550/arXiv.1612.03144.

[22] R. Girshick, “Fast r-cnn,” 2015. eprint: doi:10.48550/arXiv.1504.08083.

[23] G. E. H. et.al., “Imagenet classification with deep convolutional neural networks,” 2012. eprint: doi:10.1145/3065386.

[24] Y. L. et al., “Backpropagation applied to handwritten zip code recognition,” 1989. eprint: doi:10.1162/neco.1989.1.4.541.

[25] W. L. D. A. et al., “Ssd: Single shot multibox detector,” 2016. eprint: doi:10.48550/arXiv.1512.02325.

[26] A. V. et al., “Attention is all you need,” 2017. eprint: doi:10.48550/arXiv.1706.03762.

[27] D. B. et al., “Neural machine translation by jointly learning to align and translate,” 2014. eprint: doi:10.48550/arXiv.1409.0473.

[28] Y. K. et al., “Structured attention networks,” 2017. eprint: doi:10.48550/arXiv.1702.00887.

[29] Łukasz Kaiser; Samy Bengio, “Can active memory replace attention?,” 2017. eprint: doi:10.48550/arXiv.1610.08613.

[30] N. K. et al., “Neural machine translation in linear time,” 2016. eprint: doi:10.48550/arXiv.1610.10099.

[31] J. G. et al., “Convolutional sequence to sequence learning,” 2017. eprint: doi:10.48550/arXiv:1705.03122.

[32] M. C. et al., “Emerging properties in self-supervised vision transformers,” 2021. eprint: doi:10.48550/arXiv:2104.14294.

[33] A. H. et al., “Unetr: Transformers for 3d medical image segmentation,” 2021. eprint: doi:10.48550/arXiv:2103.10504.

[34] M. S. R. et al., “Tokenlearner: What can 8 learned tokens do for images and videos?,” 2021. eprint: doi:10.48550/arXiv.2106.11297.

[35] J.-B. G. et al., “Bootstrap your own latent: A new approach to self-supervised learning,” 2020. eprint: doi:10.48550/arXiv:2006.07733.

[36] X. C. et al., “Exploring simple siamese representation learning,” 2020. eprint: doi:10.48550/arXiv:2011.10566.

[37] V. R. et al., “Self-supervised learning: A succinct review,” 2023. eprint: doi:10.1007/s11831-023-09884-2.

[38] Y. L. et al., “A path towards autonomous machine intelligence,” 2022. eprint: https://openreview.net/pdf?id=BZ5a1r-kVsf.

[39] A. B. et al., “Vicreg: Variance-invariance-covariance regularization for self-supervised learning,” 2021. eprint: doi:10.48550/arXiv:2105.04906.

[40] J. Z. et al., “Barlow twins: Self-supervised learning via redundancy reduction,” 2021. eprint: doi:10.48550/arXiv:2103.03230.

[41] S. M. S. Tong, “Connecting joint-embedding predictive architecture with contrastive self-supervised learning,” 2024. eprint: doi:10.48550/arXiv:2410.19560.

[42] T. R. Team, “Rtx 5080 vs nvidia a30: An in-depth analysis,” 2025. eprint: https://www.runpod.io/articles/comparison/rtx-5080-vs-nvidia-a30-an-in-depth-analysis.

[43] UniPD, “Linee guida per l’impiego degli strumenti di ia generativa nella didattica e nella redazione delle tesi di laurea e di dottorato,” 2025. eprint: https://wwwassets.unipd.it/sites/default/files/2026-05/LineeGuidaImpiegoAI_didattica-dottorato-Unipd_ITA_rev20251128.pdf.

List of Figures

2.1 Representative MRI scans illustrating brain anatomy in three orthogonal views: axial (left), sagittal (center), and coronal (right). 8

2.2 Example of a T1-weighted imaging (left) and a T2-weighted imaging(right) . . 9

2.3 SynthSeg's automated brain MRI tissue segmentation, cortical parcellation, and quality control workflow across varying image quality levels. The pipeline processes MRI scans (left column) to perform tissue segmentation including cerebrospinal fluid (second column) and fine-grained cortical parcellation into color-coded anatomical regions (third column). An automated Quality Control (QC) module (right column) evaluates processing fidelity. 11

2.4 Schematic representation of a CNN pipeline divided into feature extraction and classification stages. The input image passes through alternating convolutional layers with ReLU activation for feature extraction and pooling layers for spatial downsampling. The resulting feature maps are flattened and processed by fully connected layers, culminating in a Softmax output layer that assigns class probabilities for object recognition. 12

2.5 Example of the structure of a convolutional block. Left: At each convolutional layer in a CNN, there are K kernels applied to the input volume. Middle: Each of the K kernels is convolved with the input volume. Right: Each kernel produces a 2D output, called an activation map. 12

2.6 The diagram illustrates an example of Pooling: a 2×2 max pooling process applied to a 4×4 input feature map under different stride configurations. When applying a stride of 1 (top), the filter moves incrementally over overlapping sub-regions to select maximum pixel intensities, yielding a 3×3 output map. In contrast, a stride of 2 (bottom) slides across non-overlapping quadrants, extracting the dominant value per region to downsample the input into a 2×2 output matrix. This operation reduces spatial dimensions and computational complexity while granting the network local translation invariance. 13

2.7 The upper plot displays the piecewise activation function defined as $\text{ReLU}(x) = \max(0, x)$, which maps all negative input values to zero while providing an identity mapping for positive inputs. The lower diagram demonstrates this operation applied to a sample 3×3 feature matrix. Each negative entry in the input matrix (such as -249 or -134) is thresholded to zero, whereas all positive values (such as 250 and 101) remain completely unaltered. This element-wise transformation introduces critical non-linearity into the neural network, promoting feature sparsity while helping to mitigate the vanishing gradient problem during training.utional Neural Network (CNN) architecture. n.d. 14

2.8 The schematic illustrates the extraction of a 3×3 sub-region (local receptive field) from a 6×6 input matrix. An element-wise multiplication is performed between the values of the image patch and the corresponding weights of the 3×3 kernel, followed by a summation of the products to yield a single scalar output. Specifically, the linear combination $(1 \cdot 1 + 0 \cdot 2 + 1 \cdot 3 + 0 \cdot 4 + 1 \cdot 5 + 1 \cdot 6 + 1 \cdot 7 + 0 \cdot 8 + 1 \cdot 9 = 31)$ produces the first entry of the resulting 4×4 feature map. As the filter slides across the input space with a specified stride, it systematically extracts local spatial features across the entire domain. 15

2.9 Residual building block architecture in Deep Residual Networks (ResNet). The schematic illustrates a basic residual learning unit where an input feature map $\mathbf{x}$ is transformed by stacked weight layers with an intermediate ReLU activation to fit a residual function $\mathcal{F}(\mathbf{x})$. A shortcut connection performs an identity mapping that bypasses the weight layers, adding $\mathbf{x}$ directly to the learned residual mapping to yield $\mathcal{F}(\mathbf{x}) + \mathbf{x}$. The combined output is then processed through a final ReLU activation function. This residual formulation facilitates direct gradient flow during backpropagation, effectively mitigating the vanishing gradient and degradation problems in deep neural networks. 17

2.10 Comparison of feature pyramid architectural paradigms for multi-scale object recognition. Diagram (c) illustrates a conventional pyramidal feature hierarchy, where independent predictions are made at each spatial resolution of the bottom-up feature extractor without inter-level feature fusion. Diagram (d) depicts the Feature Pyramid Network (FPN) architecture, which introduces a top-down pathway connected to the bottom-up backbone via lateral connections. This top-down path combines semantically strong high-level features with high-resolution low-level spatial details, enabling highly accurate multi-scale predictions across all pyramid levels. 19

2.11 Architectural comparison of recurrent neural network (RNN) cell structures. The diagram illustrates the internal gating mechanisms and information flow for standard RNN, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) models. The basic RNN cell (left) employs a single hyperbolic tangent (tanh) layer to process concatenated input (x_t) and hidden state (h_{t-1}). The LSTM cell (middle) manages long-term memory via a separate cell state (C_t) regulated by forget, input, and output gates with sigmoid (σ) activations. The GRU cell (right) streamlines this process by merging the cell and hidden states into a unified hidden vector (h_t), using reset and update gates to control information flow with reduced parameter complexity. 21

2.12 General conceptual framework of the attention mechanism in deep learning. The attention mechanism enables neural network models to dynamically focus on specific segments of an input sequence by assigning learnable scalar weights based on contextual relevance. Unlike conventional fixed-size representations, attention computes alignment scores between target query states and source key-value pairs to soft-select pertinent information regardless of temporal or spatial distance. This dynamic weighting resolves long-range dependency limitations, reduces information bottlenecks, and serves as the core foundational building block for modern Transformer-based architectures. 22

2.13 Complete encoder-decoder Transformer architecture. The model processes input and target sequences combined with positional encodings to inject token order information into recurrence-free representations. The encoder stack (left) processes the input sequence through multi-head self-attention and position-wise feed-forward networks, both wrapped with residual connections and layer normalization (Add & Norm). The decoder stack (right) employs masked multi-head attention to prevent causal information leakage from future tokens, followed by a cross-attention sub-layer that integrates the contextualized encoder outputs. Finally, a linear projection and Softmax function transform the decoder output into predicted token probability distributions. 23

2.14 Structural breakdown and functional abstraction of the position-wise Feed-Forward Network (FFN) module in a Transformer. The left diagram illustrates the two-stage linear transformation sequence, where an input tensor of embedding dimension d_{emb} is first projected into an expanded inner layer of dimension d_{ffn} via a linear layer. An element-wise non-linear activation function (such as ReLU or GELU) is applied, followed by a second linear layer that projects the features back down to the original embedding dimension d_{emb}. The right diagram shows the compact functional block representation, emphasizing that the FFN operates independently on each token position while preserving input-output dimensionality. 24

2.15 Patch-based tokenization and architectural pipeline for 3D medical image segmentation using a Vision Transformer. An input 3D volumetric scan with dimensions $H \times W \times D \times C$ is partitioned into a sequence of non-overlapping 3D spatial patches. These flattened patches are mapped via linear projection into 1D embedding vectors and augmented with positional encodings to form the input sequence of tokens. A Transformer Encoder processes the tokens to model global contextual relationships across spatial regions, after which a Decoder stack upsamples the encoded representations to produce the final 3D multi-organ segmentation mask. 28

2.16 TokenLearner architecture for adaptive spatial tokenization. An input feature tensor of shape $H \times W \times C$ is processed by S learned spatial attention functions $\alpha_i(x)$, generating S corresponding weight maps of size $H \times W \times 1$. Each attention map is multiplied element-wise ($\odot$) with the input tensor to highlight salient spatial regions. Spatial pooling then aggregates each weighted feature map, condensing the high-dimensional input volume into a compact set of S learned vector tokens of dimension $1 \times 1 \times C$. This adaptive mechanism enables efficient sequence representation with significantly reduced token counts for downstream Transformer processing. 31

2.17 Overall architecture of the TokenLearner and TokenFuser framework for spatio-temporal feature interaction. Input tensors of dimensions $H \times W \times C$ across time steps $t = 1, \ldots, T$ are first tokenized by TokenLearner modules, which generate S spatial attention maps per frame to extract a compact set of $ST \times C$ learned tokens. These tokens are reshaped and processed through a Multi-Head Attention block (or Vector Transformer) alongside a linear projection over the temporal-spatial dimension ST to model cross-frame contextual relationships. Finally, the TokenFuser modules remap the contextualized token representations back into the original high-resolution feature spaces ($H \times W \times C$) for each corresponding frame, enabling efficient global token interaction combined with dense spatial feature reconstruction. 33

2.18 General workflow of the Self-Supervised Learning (SSL) paradigm. The pipeline is divided into two main stages: a pretext task and a downstream task. During the pretext phase, a model is pre-trained on abundant unlabelled data using self-supervised objectives to learn meaningful feature representations without manual annotations. Through knowledge transfer, these learned representations are subsequently adapted during the downstream phase, where fine-tuning is performed on a smaller set of labelled data to produce the final model optimized for specific target tasks. 36

2.19 Architecture of the Bootstrap Your Own Latent (BYOL) self-supervised learning framework. An input image x is transformed into two augmented views, v and v', using randomized image transformations t and t'. The online network (top) processes view v through an encoder f_θ, projector g_θ, and predictor q_θ to yield a representation y_θ, projection z_θ, and final prediction $q_\theta(z_\theta)$. The target network (bottom) processes view v' through an encoder f_ξ and projector g_ξ to produce target representation z'_ξ. The loss is computed between the online prediction and the target output subjected to a stop-gradient operation $\text{sg}(z'_\xi)$. The online parameters θ are updated via backpropagation, while the target parameters ξ are updated as an exponential moving average of θ, preventing representation collapse without relying on negative sample pairs. 38

2.20 General architecture of the Joint-Embedding Predictive Architecture (JEPA) framework. An input context x and target input y are independently mapped into feature representations s_x and s_y using context and target encoders Enc(x) and Enc(y). A predictor module Pred(s_x, z) takes the context representation s_x together with a latent variable $z \in \mathcal{Z}$ specifying positional or structural transformations to estimate the target representation $\tilde{s}_y$. A distance metric $D(s_y, \tilde{s}_y)$ measures the divergence between the predicted representation and the actual target representation s_y. By making predictions directly within abstract latent space rather than generating low-level pixel details, JEPA captures high-level semantic structures while mitigating sensitivity to uninformative visual noise. . 40

4.1 Visualization of a summary of the pipeline of the model 61

5.1 Comparison of Principal Component Analysis (PCA) in the latent space before and after dataset modification. The top row displays the explained variance distribution before the intervention, characterized by a strong dominance of the first principal component (48.2%). The bottom row illustrates the analysis following the dataset modification, where the variance is more evenly distributed across multiple components (the first component explains only 6.2%), indicating a balanced latent representation and increased informational diversity. . . . 88

5.2 Comparison of FPN feature maps using different activation functions in the SynthSeg-based encoder. Each row displays the Feature Pyramid Network output for selected channels (Channel 0, 10, 20, and 30) utilizing ELU (top), ReLU (middle), and Leaky ReLU (bottom) activation functions within the encoder's convolutional blocks. The comparison highlights how the choice of activation directly influences the contrast, the sharpness of anatomical details, and the overall spatial response of the extracted features. 89

5.3 Comparison of VICReg loss term balancing before and after weight recalibration. The left column illustrates the initial configuration at 150 epochs, while the right column shows the reweighted configuration at 200 epochs. The panels display the total loss and Dice score trends (A), the unweighted raw components in logarithmic scale (B), and the percentage share of each term on the weighted total (C). Although the temporal trend of individual components is correct in both experiments, the initial weighting almost entirely obscured the covariance contribution, skewing the loss towards invariance; the subsequent rebalancing restores a proportionate and effective distribution among all regularization terms. 90

5.4 Per-ROI Dice error evolution comparing different auxiliary loss functions. The left panel shows the training progression using exclusively the Dice loss, while the right panel illustrates the results obtained by adopting a hybrid BCE+Dice function. The BCE+Dice configuration (right) clearly optimizes global learning, allowing the model to converge and correctly identify almost all anatomical regions. The apparent perfect score (zero error) for the cerebral cortex in the left graph is strictly an artifact caused by a hard-coding error in the original label declaration and does not reflect actual segmentation performance. 92
5.5 Comparison of attention maps before and after loss function optimization. The image illustrates the visual difference in activations generated using exclusively the Dice loss (left) versus the hybrid Dice+BCE combination (right). The displayed slice for each panel is automatically selected by the system based on where the peak attention is detected. The transition to the combined loss highlights a clear qualitative improvement, as the attention maps shift from blurred, dispersed regions (appearing as ”blobs”) to well-defined and anatomically significant areas. 93
5.6 PCA visualization of the FPN output for the different masking configurations. The first three principal components are mapped to the RGB channels. 95
5.7 Comparison of loss evolution for anatomical image and geometrical image masking with and without the auxiliary Dice+BCE loss. 97
5.8 ROI-specific Dice score evolution for anatomical image, anatomical token and geometrical image masking with and without the auxiliary Dice+BCE loss. . . 99
5.9 Comparison of attention maps for four representative ROIs across different masking and loss configurations. 100
5.10 Cosine similarity matrices of the learned ROI token representations for the six ablation configurations. 101
5.11 Detailed view of the cosine similarity matrix obtained with geometrical masking without the auxiliary spatial loss. 102
5.12 t-SNE visualization of the token representations for the different masking and auxiliary-loss configurations. 104
5.13 Visualization of token-to-ROI information for the different masking and auxiliary-loss configurations, normalized on the maximum value of the graph, for better visualization. 105
5.14 Visualization of token-to-ROI information for the different masking and auxiliary-loss configurations, normalized by rows therefore representing which ROI are more represented in the Token. 106

5.15 Visualization of token-to-ROI information for the different masking and auxiliary-loss configurations, normalized by columns therefore representing which tokens contribute the most to the representetion of the ROI. 107

Aknowledgements

I would like to sincerely thank Professor Manfredo Atzori for giving us the opportunity to work on such a prestigious and stimulating project, and for allowing us to gain valuable experience in the field of research. The opportunity to contribute to a project of this scope has been an important part of my academic journey, and I am truly grateful for the trust and opportunities that came with it.

I would also like to express my deepest gratitude to my co-supervisor, Louis Fabrice Tshimanga, for his constant guidance and support throughout the entire duration of the project. His availability, patience, and willingness to help whenever needed made the development of this work a much more positive and rewarding experience.

I am grateful to both of them not only for their dedication and professionalism, but also for the kindness and human support they have shown throughout this experience. Their willingness to listen, provide advice, and follow our progress with genuine attention has been greatly appreciated and has contributed significantly to the development of this project and to my growth as a student and researcher.