Algerian Democratic and Popular Republic
Ministry of Higher Education and Scientific Research

KASDI MERBAH UNIVERSITY OUARGLA
Faculty of New Information and Communication Technologies
Department of Computer Science and Information Technology

Thesis Submitted in Candidacy for a Master Degree in computer science, option

Artificial Intelligence and Data Science

by Salah eddine Addoune & Mohammed Islam Ouahbi

A Hybrid UNet-GNN Architecture for Enhanced Medical Image Segmentation

Jury Members:

Dr.	Ameur Khadidja	Jury Chair	UKM Ouargla
Dr.	Bouanane Khadra	Supervisor	UKM Ouargla
Dr.	Aiadi Oussama	Reviewer	UKM Ouargla

Academic year: 2023/2024

Acknowledgements

Alhamdulillah Rabbil Alameen, and peace and blessings be upon the noblest of creation, our master Muhammad, and upon his family and companions altogether. In these blessed moments, we would like to express our sincere gratitude to everyone who contributed to the completion of this project, and we will never forget the great favor of Allah Almighty, who blessed us with health and intellect so that we could accomplish this work. We begin by thanking our dear parents, who stood by us throughout our academic journey, providing us with all means of comfort, psychological and material support, being the best support for us in our scholarly journey.

We extend our heartfelt thanks to the esteemed supervisor, Dr. Khadra Bouanane, for her patience, sound guidance, and valuable advice that helped us overcome difficulties and achieve the best results. We cannot forget the favor of our respected professors who contributed to the development of our understanding and the enrichment of our knowledge, especially, Professor Mohammed El-Amine Kherfi, Dr. Oussama Aiadi, Dr. Bilal Khaldi , and Dr. Said Bachir.

Finally, we thank everyone who helped and supported us during our academic journey, whether from family, friends, or colleagues.

This project is the result of everyone's efforts and collaboration, and we will never forget your kindness.

With sincere thanks and appreciation.

Ouahbi Mohammed Islam
Addoune Salah Eddine

Abstract

Accurate delineation of brain tumors from magnetic resonance imaging (MRI) data is crucial for effective treatment planning and disease monitoring. This thesis proposes a novel computational approach for brain tumor segmentation that combines the strengths of convolutional neural networks (CNNs) and graph neural networks (GNNs). The proposed hybrid model integrates a U-Net CNN architecture with GNNs to leverage their complementary capabilities, the U-Net's skill in capturing local spatial patterns and the GNNs' ability to model long-range dependencies and relational information. Extensive experiments on the BraTS 2020 brain tumor segmentation challenge datasets demonstrated the potential of the proposed methodology. Compared to the baseline U-Net model, the integration of GNNs yielded substantial improvements in segmentation accuracy, including a 2.63% increase in the Dice similarity coefficient for the whole tumor region, a 2.18% improvement for the tumor core, and an impressive 3.71% enhancement for delineating the enhancing tumor region. These results underscore the efficacy of combining CNN and GNN architectures to capitalize on their complementary strengths for advancing brain tumor image segmentation. The proposed approach provides a framework for precisely defining tumor boundaries and subregions, with respect computational resources enabling more accurate quantification and characterization of tumor morphology. This can ultimately aid clinical decision-making and treatment planning while contributing to ongoing efforts to develop robust, automated brain tumor segmentation techniques.

Key words: *Brain Tumor Segmentation, MRI, Graph Neural Networks (GNN), U-Net.*

ملخص

يعد التحديد الدقيق لأورام المخ من بيانات التصوير بالرنين المغناطيسي (MRI) أمرًا بالغ الأهمية للتخطيط الفعال للعلاج ومراقبة المرض. تقترح هذه الأطروحة نهجًا حسابيًا جديدًا لتجزئة ورم الدماغ الذي يجمع بين نقاط قوة الشبكات العصبية التلافيفية ($CNNs$) والشبكات العصبية الرسومية ($GNNs$) . يدمج النموذج الهجين المقترح بنية $U-Net-CNN$ مع شبكات GNN للاستفادة من قدراتها التكميلية، ومهارة $U-Net$ في التقاط الأنماط المكانية المحلية وقدرة شبكات GNN على نمذجة التبعيات طويلة المدى والمعلومات العلائقية. أظهرت التجارب الموسعة على مجموعات بيانات تحدي تجزئة ورم الدماغ $BraTS2020$ إمكانات المنهجية المقترحة. بالمقارنة مع نموذج $U-Net$ الأساسي، أدى تكامل شبكات GNN إلى تحسينات كبيرة في دقة التجزئة، بما في ذلك زيادة بنسبة 2.63% في معامل تشابه النرد لمنطقة الورم بأكملها، وتحسين بنسبة 2.18% في قلب الورم، وتحسن مثير للإعجاب تحسين بنسبة 3.71% لتحديد منطقة الورم المعززة. تؤكد هذه النتائج على فعالية الجمع بين بنيات CNN و GNN للاستفادة من نقاط قوتها التكميلية لتعزيز تجزئة صورة ورم الدماغ. يوفر النهج المقترح إطارًا لتحديد حدود الورم والمناطق الفرعية بدقة، مع احترام الموارد الحسابية التي تتيح تقديرًا كميًا وتوصيفًا أكثر دقة لمورفولوجيا الورم. وهذا يمكن أن يساعد في نهاية المطاف في اتخاذ القرارات السريرية وتخطيط العلاج مع المساهمة في الجهود المستمرة لتطوير تقنيات قوية وآلية لتجزئة أورام المخ.

الكلمات المفتاحية : تقسيم الأورام الدماغية، الشبكات العصبية الرسومية (GNN) ، هندسة $U-Net$ ، الرنين المغناطيسي (MRI) .

Résumé

La délimitation précise des tumeurs cérébrales à partir des données d'imagerie par résonance magnétique (IRM) est cruciale pour la planification efficace du traitement et le suivi de la maladie. Cette thèse propose une nouvelle approche computationnelle pour la segmentation des tumeurs cérébrales qui combine les forces des réseaux neuronaux convolutionnels (CNN) et des réseaux neuronaux graphiques (GNN). Le modèle hybride proposé intègre une architecture U-Net CNN avec des GNN afin de tirer parti de leurs capacités complémentaires : la capacité du U-Net à capturer les motifs spatiaux locaux et l'aptitude des GNN à modéliser les dépendances à longue portée et les informations relationnelles. Des expériences approfondies sur les ensembles de données du défi de segmentation des tumeurs cérébrales BraTS 2020 ont démontré le potentiel de la méthodologie proposée. Comparé au modèle U-Net de base, l'intégration des GNN a conduit à des améliorations substantielles de la précision de segmentation, incluant une augmentation de 2,63% du coefficient de similarité de Dice pour la région tumorale complète, une amélioration de 2,18% pour le noyau de la tumeur, et une impressionnante amélioration de 3,71% pour la délimitation de la région de la tumeur en rehaussement. Ces résultats soulignent l'efficacité de la combinaison des architectures CNN et GNN pour capitaliser sur leurs forces complémentaires afin de faire progresser la segmentation des images de tumeurs cérébrales. L'approche proposée fournit un cadre pour définir précisément les limites et les sous-régions des tumeurs, en utilisant les ressources computationnelles pour permettre une quantification et une caractérisation plus précises de la morphologie des tumeurs. Cela peut finalement aider à la prise de décisions cliniques et à la planification du traitement tout en contribuant aux efforts continus pour développer des techniques robustes et automatisées de segmentation des tumeurs cérébrales.

Mots clés: *Segmentation des tumeurs cérébrales , IRM, Réseaux neuronaux graphiques (GNN) , U-Net .*

Contents

List of Figures

List of Tables

List of Abbreviations

AI:	Artificial intelligence.
DL:	Deep Learning;
GNNs:	Graph Neural Networks.
CNN:	Convolutional Neural Networks.
DSC:	Dice Similarity coefficient.
RF:	Radio Frequency.

General Introduction

Images have become an integral part of modern life, serving as a potent medium for communication, documentation, and analysis across diverse domains [6]. Within the realm of healthcare, the emergence of medical imaging technologies has brought about a revolution, facilitating non-invasive visualization and exploration of the intricate structures and functions of the human body.

Medical images, such as those obtained from X-rays, computed tomography (CT) scans, magnetic resonance imaging (MRI), and ultrasound, play a crucial role in clinical diagnosis, treatment planning, and monitoring of various diseases and conditions [7]. These imaging modalities provide valuable insights into the internal anatomy and physiological processes, allowing healthcare professionals to make informed decisions and deliver personalized care.

However, the interpretation and analysis of medical images can be a complex and time-consuming task, often requiring extensive training and expertise. Manual delineation and quantification of regions of interest, such as organs, tumors, or lesions, are prone to subjective variability and can be labor-intensive [8]. This has led to an increased demand for automated medical image segmentation techniques, which aim to partition images into multiple segments or objects based on various features, such as intensity, texture, or shape.

Image segmentation has numerous applications in the medical field, from computer-aided diagnosis and disease monitoring to surgical planning and radiation therapy [9]. One particularly critical area where image segmentation plays a vital role is in the detection and analysis of brain tumors using magnetic resonance imaging (MRI) scans.

Brain tumors are abnormal growths of cells that can disrupt normal brain function and pose significant health risks. Early and accurate detection of brain tumors is crucial for effective treatment and improving patient outcomes [10]. MRI provides detailed anatomical images of the brain, making it an invaluable tool for visualizing and characterizing brain tumors. However, manual delineation of tumor boundaries from these complex images can be challenging and subjective, highlighting the need for robust image segmentation algorithms.

By automating the segmentation process, these algorithms can potentially reduce the workload on radiologists, improve diagnostic accuracy, and facilitate personalized treatment planning [11]. Accurate segmentation of brain tumors from MRI scans enables quantitative analysis of tumor characteristics, such as size, location, and extent of infiltration, which can guide clinical decision-making and treatment strategies.

Automated medical image segmentation techniques, particularly those driven by machine learning, offer a promising solution to the challenges posed by manual interpretation. By leveraging algorithms that can partition images based on features like intensity, texture, and shape, these methods streamline the analysis process, reducing the burden on healthcare professionals and enhancing diagnostic accuracy [8].

The advantages of using machine learning tools for segmentation in medical imaging are multifaceted, offering significant improvements in efficiency, accuracy, and scalability. Machine learning algorithms can analyze vast amounts of imaging data quickly and consistently, reducing the time required for image interpretation and minimizing human error. These tools are capable of learning from annotated datasets to recognize complex patterns and features within medical images, which can enhance diagnostic precision and reproducibility [8]. In particular, deep learning, a subset of machine learning, has emerged as a powerful approach for medical image segmentation due to its ability to automatically learn hierarchical features from data without the need for manual feature extraction.

Convolutional Neural Networks (CNNs) have revolutionized medical image segmentation by mimicking the human brain's visual processing capabilities. They excel at handling the complexity of medical images, detecting edges, textures, and intricate structures such as tumors or lesions [11]. The deep architecture of CNNs captures both low-level and high-level features, facilitating precise delineation of anatomical structures and pathological regions. CNNs offer several advantages for medical image segmentation. They efficiently process 3D volumetric data from modalities like MRI and CT scans. End-to-end learning optimizes segmentation processes, ensuring higher accuracy and efficiency. Techniques such as transfer learning and data augmentation further enhance CNN performance, making them robust across different imaging conditions and patient demographics [9]. These capabilities establish CNNs as essential for automated, accurate, and efficient medical image segmentation, ultimately improving patient care and outcomes.

U-Net is a type of convolutional neural network architecture designed for medical image segmentation tasks, featuring an encoder-decoder structure with skip connections for effective feature localization [12]. Since its inception, U-Net has evolved with various modifications to enhance performance across diverse medical imaging applications.

U-Net and its variants have been successfully applied to tasks such as MRI image segmentation [13]. The versatility and effectiveness of U-Net make it a valuable tool for medical image segmentation solutions. The nnU-Net [14] framework automatically configures the U-Net architecture for various segmentation tasks, dynamically adjusting its network depth, feature maps, and other hyperparameters based on the input dataset. This framework achieved respectable results for the BraTS 2020 challenge. However, its complexity in adaptation can make it less transparent, it demands significant computational resources during configuration and training, and its effectiveness relies heavily on the quality and characteristics of the input dataset.

To address the computational efficiency and resource requirements of U-Net and its variants, GNNs offer a promising solution.

Graph Neural Networks (GNNs) are deep learning models specifically designed for graph-structured data like social networks and citation networks [3]. They excel in capturing relational information inherent in graph data, making them suitable for handling complex and large-scale structures within medical images. This capability enhances the accuracy of segmentation tasks by effectively incorporating global context. Moreover, GNN-based models generally require fewer computational resources than CNN-based approaches [15]. Therefore, the integration of GNNs into the U-Net model can potentially bring about improvements in segmentation accuracy and robustness, especially in the context of complex medical images such as MRI scans of brain tumors.
Several studies have explored the integration of GNNs with CNN-based architectures for brain tumor segmentation.

Jiang et al. [16] proposed a joint graph and image convolution network (GCNN), employing a Graph Convolutional Networks (GCN) module to capture long-range dependencies and integrate contextual information from the image, which was then fused with CNN features for improved segmentation performance. Huang et al. [17] introduced a spectral-spatial graph neural network (SSGNN), leveraging both spectral and spatial information by combining graph convolutions with traditional convolutions to capture multi-scale contextual information and spatial dependencies. Wang et al. [18] proposed DUNet-GNNPool architecture combines the strengths of a modified U-Net for feature extraction and a graph neural network (GNNPool) for capturing spatial relationships and clustering nodes based on both features and graph topology. The DUNet initializes the node features, and the GNNPool iteratively refines and optimizes the segmentation by pooling the graph and clustering the nodes.

In this thesis, we introduce a novel approach that integrates a Graph Neural Network (GNN) into a CNN-based segmentation model. Initially, the CNN-based model is trained to generate initial segmentation predictions. These predictions guide the GNN model by directing graph operations towards regions where the CNN predictions are uncertain or ambiguous. We employ various training strategies to enhance model performance while ensuring computational efficiency.

This thesis is organized as follows:

- Chapter 1 introduces the fundamentals of digital image processing in medical imaging, with a focus on MRI and brain tumor analysis. It discusses the limitations of traditional methods and explores deep learning techniques, particularly the U-Net architecture. Various approaches addressing these limitations in medical image segmentation are reviewed.
- Chapter 2 provides an overview of Graph Neural Networks (GNNs) with a detailed exploration of the GraphSAGE architecture. It emphasizes its advantages in medical image analysis and explains its operational mechanisms, along with related work in brain tumor segmentation.
- Chapter 3 presents our approach of integrating a CNN-based architecture with a GNN for brain tumor segmentation. Different training strategies for enhancing initial segmentation are described.
- Chapter 4 introduces the BraTS 2020 datasets and outlines our performance metrics for evaluating the model. The chapter describes our experimental setup, including training the U-Net model to improve segmentation. Before utilizing GNN, we validate the mask hypothesis and train GNN using our strategies.
- Chapter 5 discusses the implementation of our methodology, focusing on various employed strategies and includes a detailed discussion of the results.

Chapter 1

Medical Image Segmentation

1.1 Introduction

This chapter provides a brief overview of digital image processing and medical imaging, especially the segmentation task. Explores traditional segmentation methods alongside deep learning techniques. Moreover, it discusses the CNN architecture, explains its importance and acknowledges its limitations including the U-Net architecture in the field of medical image segmentation.

1.2 Digital Image

A digital image [6] is a two-dimensional array of discrete image elements, called pixels, where each pixel is assigned a numerical value representing its intensity. Mathematically, a 2D digital image can be represented as a 2D function $f(x, y)$, where x and y are spatial coordinates, and the amplitude of f at any pair of coordinates (x, y) is the intensity value of the image at that point.

1.2.1 Types of digital Image

There are different types of digital images, based on various characteristics and properties. Here are some common types of digital images [6]:

- **Binary images:** These images contain only two possible pixel values, typically 0 (black) and 1 (white). They are often used in applications like document imaging, character recognition, and simple graphics.
- **Grayscale images:** In these images, each pixel has a single value representing its intensity or grayscale level, typically ranging from 0 (black) to 255 (white). Grayscale

images are commonly used in medical imaging, remote sensing, and photography.

- **Color images:** Color images have multiple color channels (e.g., red, green, blue), and each pixel is represented by a combination of values from these channels. Common color models include RGB (Red, Green, Blue), CMYK (Cyan, Magenta, Yellow, Black), and HSV (Hue, Saturation, Value).
- **Multispectral images:** These images capture information across multiple spectral bands, beyond the visible light spectrum. They are used in remote sensing, astronomy, and other scientific applications.

1.2.2 Medical images

A visual representation or depiction of the internal structures, anatomy, or physiological processes of the human body, obtained through various imaging modalities and techniques such as X-rays, computed tomography(CT), magnetic resonance imaging (MRI), ultrasound, nuclear medicine, and others, used for diagnostic, therapeutic, and research purposes in healthcare [7].It is captured using sophisticated equipment, These equipment use different wavelengths to capture images, with each wavelength showing a different level of tissue or information.Figure 1.1 shows :

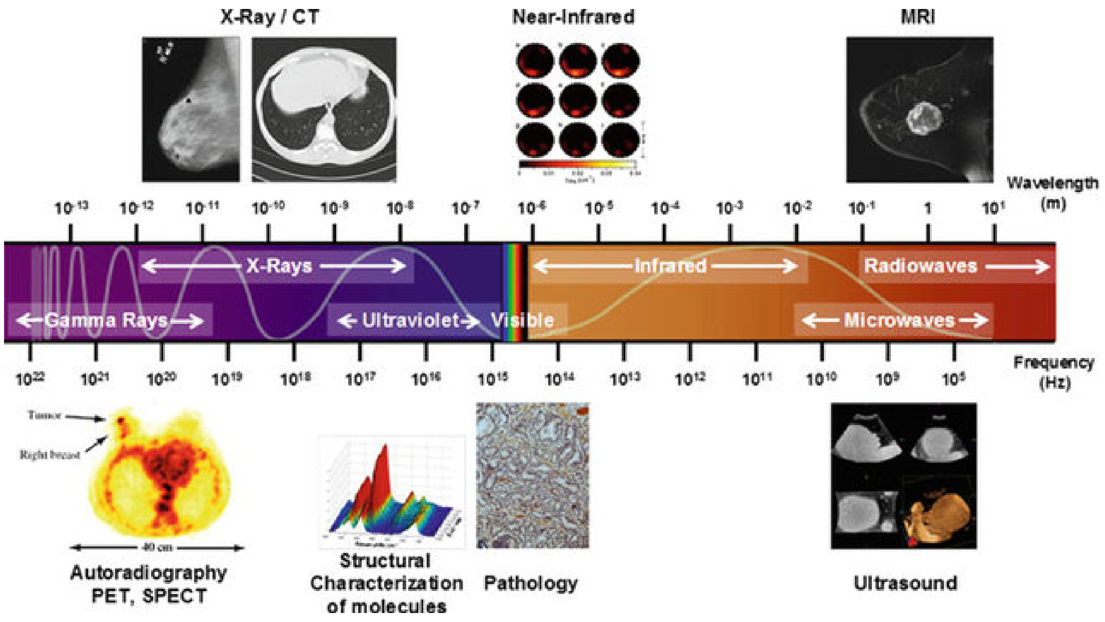


Figure 1.1: Electromagnetic spectrum and associated imaging modalities [1]

1.2.3 Magnetic resonance imaging (MRI)

MRI [7] is a diagnostic technology that uses magnetic and radio frequency fields to image the body tissues and monitor body chemistry. The MRI used for visualizing morphological alterations rests on its ability to detect changes in proton density and magnetic spin relaxation times, which are characteristic of the environment presented by the diseased tissue.

Physical Principles of MRI modalities

Here's the physical basis of each MRI modality [19]:

- **T1-weighted MRI:** Emphasize the differences in T1 relaxation times of tissues, providing good anatomical detail. Structures with short T1 relaxation times appear bright.
- **T2-weighted MRI:** Emphasize the differences in T2 relaxation times of tissues, highlighting fluid-filled structures and pathology. Structures with long T2 relaxation times appear bright.
- **T1 Contrast-Enhanced MRI:** Involves the administration of a contrast agent, typically gadolinium-based, which shortens the T1 relaxation time of tissues, thereby enhancing the contrast between different structures.
- **FLAIR (Fluid-Attenuated Inversion Recovery):** FLAIR is a specialized MRI sequence that suppresses the signal from cerebrospinal fluid (CSF), allowing for better visualization of pathological structures adjacent to CSF-filled spaces.

1.3 Image Segmentation

The process of partitioning a digital image into multiple segments or sets of pixels, with the goal of separating objects or regions of interest from the background [6].

1.3.1 Semantic Segmentation

Semantic segmentation [9] performs pixel-level labeling with a set of object categories (e.g., human, car, tree, sky) for all image pixels, thus it is generally a harder undertaking than image classification, which predicts a single label for the entire image.

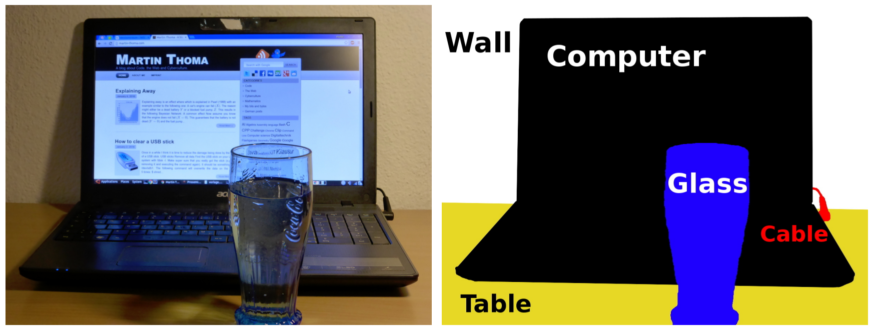


Figure 1.2: Semantic segmentation [2]

1.3.2 Medical Image Segmentation

Medical image segmentation [9]is the process of partitioning a medical image into multiple segments or regions, each representing a different anatomical structure or tissue type. It plays a crucial role in various medical applications, including diagnosis, treatment planning, and quantitative analysis. The goal of medical image segmentation is to accurately identify and delineate regions of interest (ROIs) within the image, enabling radiologists, physicians, and researchers to extract relevant information for further analysis or decision-making.

1.3.3 Importance of Medical Image Segmentation

Medical image segmentation plays a crucial role in various clinical applications, including diagnosis, treatment planning, and image-guided interventions [6, 9]. Accurate segmentation of anatomical structures or pathological regions from medical images such as MRI, CT, and ultrasound scans is essential for quantitative analysis and disease characterization [8, 20, 21]. In the context of brain tumor imaging, segmentation facilitates the delineation of tumor boundaries, which is vital for surgical planning, radiotherapy treatment, and assessing treatment response [12, 22].

1.3.4 Traditional Methods

Medical image segmentation is a crucial step in medical image analysis, where you isolate specific regions of interest (ROIs) within an image. Traditional methods, while not as advanced as deep learning techniques, have laid the groundwork for modern approaches and remain valuable due to their interpretability and efficiency in specific situations[8, 23]. Here's an overview of some prominent traditional methods :

- **Thresholding:** This simple yet effective method separates objects based on intensity values. Pixels exceeding a predefined threshold are classified as foreground (object), while those below are background[24].

- **Region Growing:** This method starts with a seed point within the ROI and iteratively incorporates neighboring pixels with similar intensity or feature values, progressively building the segmented region[25].
- **Edge Detection:** This approach focuses on identifying boundaries between objects by analyzing intensity variations. Edges often correspond to significant changes in intensity and can be used to delineate object borders[26].

1.3.5 Limitation of Traditional Methods

While traditional segmentation techniques have been widely used, they often face several challenges and limitations in medical image analysis [8]:

- **Intensity inhomogeneity**: Medical images frequently exhibit intensity variations, making it difficult to define a single threshold or region criteria for segmentation.
- **Noise and artifacts**: Noise and imaging artifacts can adversely affect the performance of traditional methods, leading to inaccurate segmentation results.
- **Complex anatomical structures**: Traditional methods may struggle with segmenting complex anatomical structures with varying shapes, sizes, and intensities.
- **User interaction**: Many traditional techniques require manual initialization or parameter tuning, which can be time-consuming and subjective.
- **Generalization**: Traditional methods often lack the ability to generalize well to diverse datasets and different imaging modalities.

These limitations have motivated the development of more advanced and robust segmentation techniques, particularly those based on machine learning and deep learning approaches .

1.4 Deep Learning for Image Segmentation

Deep learning, a subfield of machine learning, has revolutionized various domains, including medical image analysis. Convolutional neural networks (CNNs) are a type of deep learning architecture that has been particularly successful in computer vision tasks, such as image classification, object detection, and segmentation [21]. A range of architectures have emerged that are CNN-based, such as U-Net [12] which are effectively the best for medical image segmentation.

1.5 CNN for medical image segmentation

Convolutional Neural Networks (CNNs) [20] are a type of deep learning model designed to process data with a grid-like topology, such as images.They consist of multiple layers, including convolutional layers that apply filters to input data, pooling layers that reduce spatial dimensions, and fully connected layers that perform classification or regression tasks. CNNs excel at learning hierarchical representations of features in images, enabling them to perform tasks like object detection, recognition, and segmentation with remarkable accuracy.

In the field of medical image analysis, CNNs have emerged as powerful tools for image segmentation tasks. By leveraging their ability to learn intricate patterns and features from large-scale datasets, CNNs can accurately segment anatomical structures or abnormalities in medical images. Architectures like U-Net, which utilize encoder-decoder structures with skip connections, are commonly employed for tasks such as tumor segmentation, organ delineation, and lesion detection[12]. The application of CNNs in medical image segmentation facilitates automated and efficient analysis, aiding in diagnosis, treatment planning, and medical research [23].

In this section, we will outline the primary functionalities of CNNs, focusing on three key components:

- convolution operations.
- max pooling.
- activation functions (ReLU, softmax).

1.5.1 Convolution operation

The convolution operation applies a filter (also called a kernel) to an input, such as an image. The filter is a small matrix of parameters that is applied to a local region of the input by computing the dot product between the entries of the filter and the corresponding part of the input. This process is repeated for every spatial location of the input, producing a 2D activation map that gives the responses of that filter at every spatial position [20].

1.5.2 Max Pooling

Max pooling is a form of non-linear down-sampling. A max pooling operation reports the maximum value from a region of the input. This has the effect of progressively reducing the spatial size of the representation, which reduces the number of parameters and computations in the network, and controls overfitting [20].

1.5.3 Activation Functions

An activation function [20] is a mathematical function applied to the weighted sum of inputs in an artificial neuron. The purpose is to introduce non-linearity, enabling neural networks to learn and model complex mappings between inputs and outputs.
The ReLU activation function is widely used in deep neural networks, including those employed for semantic segmentation tasks. It is defined as:

$$ReLu(x) = \max(0, x) \tag{1.1}$$

This non-linear transformation helps introduce sparsity in the activations, which can improve the representational capacity of the neural network and aid in faster convergence during training [27].

In semantic segmentation tasks, the output of the neural network is typically a set of class probabilities for each pixel or voxel (in the case of 3D medical images). The Softmax function is commonly used to ensure that these class probabilities are well-formed and sum to one. The Softmax function is defined as:

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{1.2}$$

where x_i is the input value for class i, and N is the total number of classes.

The Softmax function takes a vector of arbitrary real-valued scores and squashes them into a vector of values between 0 and 1 that sum to 1, representing the probability distribution over the classes. This function is typically applied to the final layer of the neural network, ensuring that the output predictions are properly normalized and can be interpreted as class probabilities. This is crucial for accurate pixel-wise or voxel-wise classification and segmentation [12].

1.5.4 Fully connected layer

Fully connected layers play a crucial role in neural networks, particularly at the end of architectures like Convolutional Neural Networks (CNNs) [20]. They map the features extracted by earlier layers to the final output, typically for tasks like classification. In a fully connected layer, each neuron connects to every neuron in the previous layer, allowing for a comprehensive integration of features. This connectivity pattern facilitates the computation of activations through matrix multiplication of the previous layer's activations with a weight matrix, followed by the addition of a bias vector [20]. These layers are responsible for making final predictions or classifications based on the learned features.

1.5.5 U-Net for brain tumor segmentation

U-Net is an encoder-decoder structure based on CNN architecture, which has demonstrated remarkable success in brain tumor segmentation tasks [12, 10]. It contains convolutional operations and pooling layers. Skip connections enable the network to incorporate multi-scale features, capturing both coarse-grained contextual information and fine-grained details, which is essential for accurate tumor delineation [13]. The final layer is a fully connected layer that maps the learned features to the desired segmentation classes. This structure ensures that the network can effectively segment complex medical images by preserving spatial information and integrating hierarchical features at multiple scales.

1.5.6 Limitations of U-Net

While the U-Net architecture has demonstrated impressive performance in various medical image segmentation tasks, including brain tumor segmentation [12], it is not without its limitations. Despite its powerful feature extraction capabilities, U-Net may struggle in certain scenarios. The following subsection outlines some key limitations of the U-Net model:

- **Data Requirements and Augmentation :** U-Net, like many deep learning architectures, demands a substantial amount of labeled data for effective training. In medical imaging, obtaining large annotated datasets can be challenging due to privacy concerns, the need for expert annotation, and variability in imaging quality and characteristics [12]

- **Computational Resources and Training Time :** This limitation depends on the previous one. When the U-Net requires more data, it inevitably needs more resources and more time.

- **Lack of Global Context Modeling**: U-Net primarily focuses on capturing local spatial patterns and dependencies within a fixed receptive field. However, it may struggle to effectively model long-range dependencies and global contextual information, which can be crucial for accurate segmentation of complex structures, such as tumors with irregular shapes or diffuse boundaries [28, 29].

- **Limited Structural Reasoning**: U-Net's convolutional operations are designed to extract features from grid-like structures (e.g., images), but they may not fully capture the intricate structural relationships and dependencies present in medical imaging data, particularly in the case of 3D volumes [30].
- **Sensitivity to Input Variations**: Despite its powerful feature extraction capabilities, U-Net can be sensitive to variations in input data, such as changes in intensity distributions, noise levels, or acquisition protocols. This can lead to performance degradation when applied to data that deviates from the training distribution [14].

1.5.7 Addressing U-Net limitations

To overcome the limitations of the U-Net architecture and enhance segmentation performance, various strategies and enhancements have been proposed.

- **Attention Mechanisms:** Incorporating attention modules into U-Net can help the model focus on the most relevant features and suppress irrelevant regions, improving segmentation accuracy, especially for small or occluded objects [31, 32].
- **Ensemble Methods:** Combining predictions from multiple U-Net models trained with different initialization, data augmentation [14], or architectures can help overcome the limitations of a single model and improve overall performance.
- **Multi-Scale Inputs:** Providing the U-Net with multi-scale inputs [33], either by processing different scales separately or using a pyramidal approach, can help capture both local and global contextual information, addressing the fixed receptive field limitation.
- **Hybrid Architectures:** Combining U-Net with other neural network architectures like graph neural networks [16, 18] (as done in this thesis) , recurrent neural networks, or transformers can leverage their respective strengths and potentially improve segmentation performance.

In the upcoming chapter, we will delve into the fundamental concepts of Graph Neural Networks (GNNs) and their functional principles. Our focus will be on exploring their specific architecture and discussing their advantages .Additionally GNN in medical image segmentation.

1.6 Conclusion

In this chapter, we have explored the fundamental concepts and techniques related to digital image processing and medical image segmentation, with a particular emphasis on brain tumor segmentation. We began by discussing the basics of digital images and their different types. We then delved into the realm of medical imaging, highlighting the importance of various modalities like MRI, CT in healthcare diagnosis and treatment planning. Specifically, we provided an overview of magnetic resonance imaging (MRI) and its underlying physical principle.

Next, we introduced the concept of image segmentation and its crucial role in medical applications, including the challenges. Then we discussed CNN architecture, especially the U-Net architecture, which has revolutionized medical image segmentation by effectively segmmode [12]. However, we also acknowledged the limitations of the U-Net, such as its inability to model long-range dependencies, handle input variations, and capture intricate structural relationships [14]. then addressing it's using several methods including hybrid architecture's.

Chapter 2

Graph Neural Networks for Medical Image Segmentation

2.1 introduction

Convolutional Neural Networks (CNNs) have achieved remarkable success in medical image analysis, particularly in tasks such as brain tumor segmentation. demonstrated excellent performance in various biomedical segmentation tasks. However, CNNs primarily focus on local spatial information, which may limit their ability to model long-range dependencies and relationships between pixels or voxels in the image.

In this chapter provides an overview of Graph Neural Networks (GNNs). It delves into the GraphSAGE architecture, highlighting its advantages in medical image analysis and detailing its operational mechanisms.

2.2 Graph Neural Networks (GNNs)

Graph Neural Networks (GNNs) are a class of neural network models that operate directly on graph-structured data. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a set of nodes represent entities and $\mathcal{E}$ i a set of edges represent relationships between entities [3]. Here's an overview of the key concepts and architectures in GNNs:

1. Node Embedding.
2. Message passing.
3. Aggregation Function.

2.2.1 Node Embedding

Node embeddings refer to the low-dimensional vector representations learned for each node in a graph. These node embeddings capture the structural and relational information about the nodes, and they are essential for various graph-based machine learning tasks[3].The node embeddings are typically obtained as the output of the last layer of the GNN model, where the node representations are updated through the message passing process[3]. The node embeddings are generated by an encoder function, which maps nodes to their corresponding vector representations. Formally:

$$\mathbf{z}_v = \text{ENC}(v; \Theta) \tag{2.1}$$

Where:

- $\mathbf{z}_v \in \mathbb{R}^d$ is the d-dimensional node embedding for node v
- $\text{ENC}(\cdot)$ is the encoder function.
- v is the node for which we want to generate the embedding.
- Θ represents the parameters of the encoder function.

This formula indicates that the node embedding $\mathbf{z}_v$ is obtained by applying the encoder function ENC to the node v, with the encoder's parameters Θ. The specific form of the encoder function can vary depending on the embedding method used, but this general formula captures the idea that the node embeddings are generated by an encoder function from the node information.

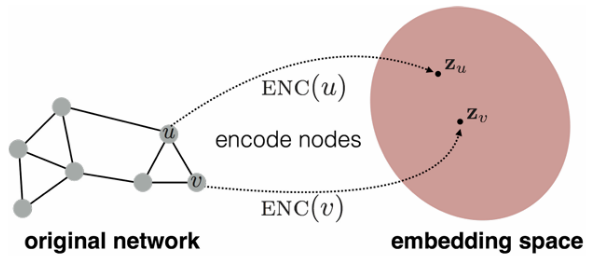


Figure 2.1: Node embedding, encoder (ENC) is a function or model that maps nodes from the original graph into a low-dimensional embedding space [3].

In the context of images, node embeddings can represent different elements or regions within the image. Each node corresponds to a specific region, such as a pixel, a superpixel, or an object proposal. The node embeddings capture the visual features and spatial relationships of these regions, allowing for various image understanding tasks such as image classification, object detection, and image segmentation.

2.2.2 Message Passing

Message passing is the fundamental operation that allows GNNs to learn node representations by aggregating information from a node's local neighborhood. In message passing, each node iteratively aggregates feature information from its neighbors, and then uses this aggregated information to update its own node representation [3, 34]. Formally, the message passing procedure can be expressed as:

$$\mathbf{h}_u^{(k+1)} = \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \text{AGGREGATE}^{(k)}\left(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\}\right)\right) \tag{2.2}$$

$$\mathbf{h}_u^{(k+1)} = \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right), \tag{2.3}$$

Where:

- $\mathbf{h}_u^{(k-1)}$ is the embedding of node u at iteration $k-1$,
- $\text{UPDATE}^{(k)}$ is the update function at iteration k,
- $\text{AGGREGATE}^{(k)}$ is the aggregation function at iteration k,
- $\mathcal{N}(u)$ is the neighborhood of node u,
- $\mathbf{m}_{\mathcal{N}(u)}^{(k)}$ is the message aggregated from the neighborhood of node u at iteration k,

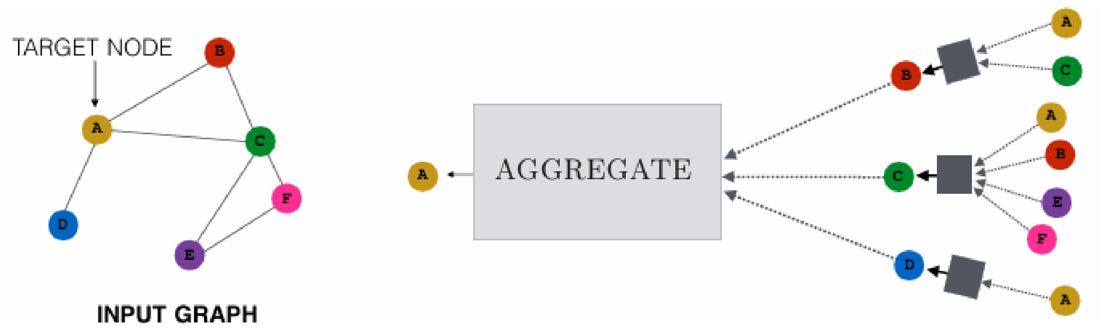


Figure 2.2: Massage passing [3]

Overview of how a single node aggregates messages from its local neighborhood. The model aggregates messages from A s local graph neighbors (i.e., B, C, and D), and in turn, the messages coming from these neighbors are based on information aggregated from their respective neighborhoods, and so on. This visualization shows a two-layer version of a message-passing model. Notice that the computation graph of the GNN forms a tree structure by unfolding the neighborhood around the target node [3]

2.2.3 Aggregation Function

The aggregation function [3], denoted as $\mathcal{A}$, is a key component of the message passing process in GNNs. The aggregation function is responsible for combining the feature information from a node's neighbors to compute the message that will be used to update the node's representation. Formally, the aggregation function is defined as:

$$m_v^{(k+1)} = \mathcal{A}\left(\{h_u^{(k)} | u \in \mathcal{N}(v)\}\right) \tag{2.4}$$

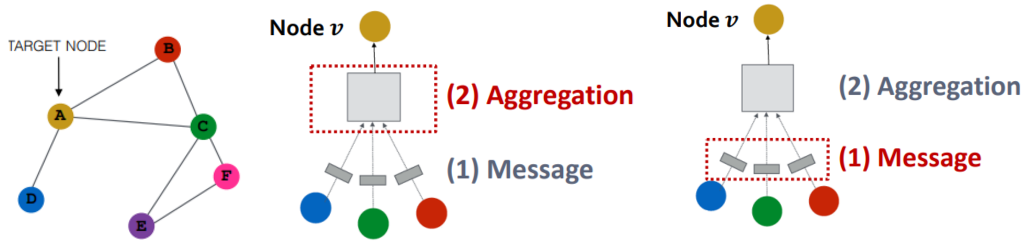
Where:

- $m_v^{(k+1)}$ is the message computed for node v at iteration $k+1$,
- $h_u^{(k)}$ is the representation of node u at iteration k,
- $\mathcal{N}(v)$ is the set of neighbors of node v.

Common aggregation functions include:

- **Mean**:
$$\mathcal{A}(m_j^{(l+1)} | j \in \mathcal{N}(i)) = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} m_j^{(l+1)} \tag{2.5}$$
- **Max-pooling**:
$$\mathcal{A}(m_j^{(l+1)} | j \in \mathcal{N}(i)) = \max_{j \in \mathcal{N}(i)} m_j^{(l+1)} \tag{2.6}$$

The choice of the aggregation function can significantly impact the expressive power and performance of the GNN model [3].



(a) Input Graph (b) Aggregation (c) Message

Figure 2.3: visualize Aggregation function and Message passing [4].

2.2.4 Common Graph Neural Network Architectures

1. **Graph Convolutional Networks (GCNs)**[34] **:** Introduced by Kipf and Welling in 2016, GCNs apply a convolutional operation on graphs by aggregating feature information from a node's local neighbors. The key operation is:

$$H^{(l+1)} = \sigma(D^{-1/2} \cdot A \cdot D^{-1/2} \cdot H^{(l)} \cdot W^{(l)}) \tag{2.7}$$

   Where A is the adjacency matrix, D is the degree matrix, $H^{(l)}$ is the node feature matrix at layer l, and $W^{(l)}$ is the trainable weight matrix.

2. **GraphSAGE** [5]**:** Proposed by Hamilton et al. in 2017, GraphSAGE uses a general inductive framework to generate node embeddings by sampling and aggregating features from a node's local neighborhood. Different aggregation functions.

3. **Graph Attention Networks (GATs)** [35]**:** Introduced by Velickovic et al. in 2018, GATs use self-attention mechanisms to compute node representations by attending over their neighbors' features, assigning different importance to different neighbors.

2.3 GraphSAGE Architecture

GraphSAGE (SAmple and aggreGatE) is a general inductive framework designed to efficiently generate node embeddings for unseen data by leveraging node feature information, such as text attributes or node degrees [5]. Unlike traditional embedding methods that require all nodes to be present during training, GraphSAGE learns a function that generates embeddings by sampling and aggregating features from a node's local neighborhood. This allows the model to generalize to previously unseen nodes or entirely new graphs, making it highly suitable for dynamic and evolving datasets.

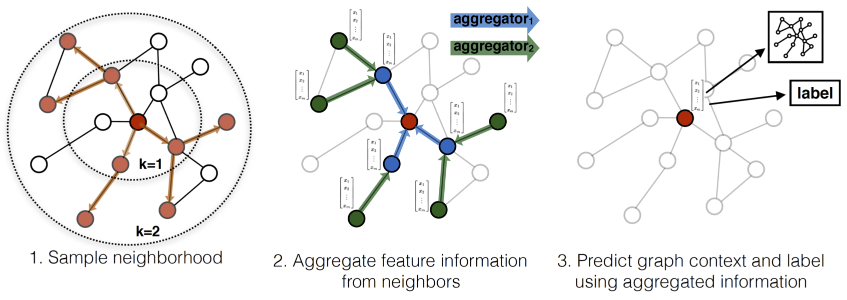

k=1
k=2
1. Sample neighborhood
aggregator1
aggregator2
2. Aggregate feature information from neighbors
label
3. Predict graph context and label using aggregated information

Figure 2.4: Illustration of the GraphSAGE sample and aggregate approach. Each node's feature vector is updated by aggregating features from its local neighborhood [5].

2.3.1 Embedding Generation Process

The GraphSAGE algorithm generates embeddings through a multi-step process, iterating over the local neighborhoods of each node. The key steps are:

1. **Feature Initialization:** Initialize the node features $\mathbf{h}_v^0 = \mathbf{x}_v$, where $\mathbf{x}_v$ represents the input features of node v.

2. **Neighborhood Aggregation:** For each layer $k \in \{1, \dots, K\}$, aggregate the features from the neighbors of node v:

$$\mathbf{h}_{\mathcal{N}(v)}^k = \text{AGG}_k \left(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\} \right), \tag{2.8}$$

   where $\mathcal{N}(v)$ denotes the set of neighbors of node v.

3. **Feature Update:** Update the node features by combining the node's previous layer features with the aggregated neighborhood features:

$$\mathbf{h}_v^k = \sigma \left(\mathbf{W}_k \cdot \text{CONCAT} \left(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k \right) \right), \tag{2.9}$$

   where $\mathbf{W}_k$ is a trainable weight matrix, CONCAT denotes concatenation, and σ is a non-linear activation function.

4. **Normalization:** Optionally normalize the updated features:

$$\mathbf{h}_v^k = \frac{\mathbf{h}_v^k}{\|\mathbf{h}_v^k\|_2}. \tag{2.10}$$

The final node embedding is given by $\mathbf{z}_v = \mathbf{h}_v^K$, which is used for downstream tasks such as node classification or link prediction.

2.3.2 Aggregator Functions

GraphSAGE may use different types of aggregator functions, like the mean aggregator or the max-pooling aggregator, to capture multiple aspects of the neighborhood's information.

2.3.3 SAGEConv Layer

The SAGEConv layer is a specific implementation of the GraphSAGE architecture that uses a specific aggregator function to generate node embeddings. The SAGEConv layer can be defined mathematically as follows:

$$h_v^k = \sigma\left(W^k \cdot \mathrm{AGG}_k\left(\{h_u^{k-1}, \forall u \in N(v) \cup \{v\}\}\right)\right) \tag{2.11}$$

where :

- AGG_k is the aggregation function at layer k,
- W^k is the weight matrix at layer k,
- σ is a non-linear activation function.

The SAGEConv layer thus combines the features of a node with the aggregated features of its neighbors, applies a linear transformation, and passes the result through a non-linear activation function to obtain the updated node features.

2.3.4 Advantages of GraphSAGE in Brain Tumor Segmentation

- **Capturing Spatial Context:** GraphSAGE effectively captures spatial context and relationships between histopathological features within tissue samples, allowing for more accurate and context-aware analysis of cellular structures and tissue morphology [36].

- **Integrating Multi-Modality Data:** GraphSAGE is well-suited for integrating multi-modality medical imaging data, allowing for joint analysis of diverse imaging modalities such as MRI, CT, and PET scans. By modeling relationships between different imaging modalities, GraphSAGE enhances the understanding of complex medical conditions and improves diagnostic accuracy [37].

- **Scalability:** GraphSAGE's ability to generate node embeddings in a scalable manner is advantageous for large-scale datasets, facilitating efficient processing of diverse patient metadata and imaging data [5].

- **Resource Efficiency:** In computational histopathology, where datasets can be enormous, GraphSAGE's sampling and aggregation methods ensure that computations remain feasible and efficient. This is crucial for processing high-resolution histopathological images and associated data[36].

2.4 GNNs for Brain Tumor Segmentation: Related work

In recent years, the application of graph neural networks (GNNs) has emerged as a promising approach for brain tumor segmentation, offering innovative solutions to enhance feature extraction, representation, and integration of multimodal data. In this section, we review a GNNs and Hybrid GNNs architectures designed for medical imaging to integrate multimodal data to improve segmentation accuracy and robustness.

Cui et al. [38] introduced a multi-class brain tumor segmentation approach using a graph attention network (GAT), allowing the model to adaptively focus on relevant contextual information during segmentation. The GAT computations involve computing attention values between neighboring nodes and normalizing them to produce a richer feature representation. This approach enables precise segmentation of brain tumors by effectively combining graph-based methods with feature extraction and attention mechanisms, showcasing a promising methodology for medical image analysis.

Their approach faces challenges. The computational complexity of attention mechanisms could limit scalability, and the dependency on accurately constructed graphs may affect generalizability across diverse datasets. Moreover, interpreting attention values and integrating clinical expertise remain significant hurdles. Overcoming these limitations will be crucial for enhancing the reliability and applicability of GAT-based segmentation methods in clinical settings. Huang et al. [17] introduced a spectral-spatial graph neural network (SSGNN), leveraging both spectral and spatial information by combining graph convolutions with traditional convolutions to capture multi-scale contextual information and spatial dependencies.

The spatial component utilizes a GraphSAGE model to capture local neighborhood features within the graph representation of the MRI data, where nodes correspond to supervoxels. The spectral component employs Chebyshev convolutions to model global structural patterns by analyzing the graph Laplacian in the frequency domain. By integrating these spatial and spectral GNN components into a unified spectral-spatial GNN model.

While the spectral-spatial graph neural network (SSGNN) integrates both spectral and spatial information effectively, it faces challenges in scalability and computational complexity. The use of Chebyshev convolutions for spectral analysis demands significant computational resources, particularly with large-scale datasets. Additionally, the model's performance heavily relies on the accurate construction of the graph representation and the quality of feature extraction from supervoxels, which can vary depending on the dataset quality and preprocessing techniques employed.

Jiang et al. [16] proposed a joint graph and image convolution network (GCNN), Initially, the GNN predicts node labels, providing a coarse understanding of the graph structure. This output is then refined by incorporating CNNs, which leverage the original voxel image data alongside the GNN predictions. By combining these two modalities, the model is empowered to make precise adjustments to its predictions, leveraging fine-grained local voxel information for improved segmentation performance. This novel fusion of GNNs and CNNs represents a significant advancement in the segmentation task.

However, a notable limitation lies in the dependency on initial GNN predictions for subsequent CNN refinement. In scenarios where GNN predictions are inaccurate or incomplete, the overall segmentation performance may suffer. Moreover, the computational overhead of combining these two complex models necessitates efficient hardware and optimization strategies to achieve real-time performance in clinical applications.

Wang et al. [18] proposed DUNet-GNNPool, a deep architecture incorporating graph pooling for 3D image segmentation, is composed of two main levels, a feature extraction with the extend U-Net (DUNet), and a GNN-Graph pooling (GNNPool), In the first level, DUNet is trained to extract the high-level features to initialize the graph nodes. In the second level, GNNPool is built to refine and optimize the segmentation result.

This integration ensures that the model can leverage both the local image features captured by DUNet and the global contextual information provided by GNNPool. Moreover, the use of graph pooling in GNNPool helps to mitigate issues such as over-segmentation and noise, leading to more robust and accurate segmentation results.

Despite its robustness in handling spatial and contextual information, the method may encounter challenges in training convergence and generalization across diverse datasets. The intricate interplay between feature extraction and graph pooling stages requires careful tuning of hyperparameters and regularization techniques to prevent overfitting and ensure stable convergence during training.

2.5 Conclusion

In conclusion, the integration of Graph Neural Networks (GNNs) into brain tumor segmentation represents a significant advancement in medical image analysis. GNNs offer unique capabilities to model complex relationships and dependencies within medical imaging data, particularly valuable for tasks like brain tumor segmentation where spatial context and multi-modal information are crucial.

Studies reviewed here, such as those employing Graph Attention Networks (GATs), Graph Convolutional Networks (GCNs), and hybrid models like Joint Graph and Image Convolution Networks (GCNN), demonstrate the versatility of GNNs in enhancing segmentation accuracy. These models effectively leverage graph-based representations to capture both local and global features from medical images, improving the delineation of tumor boundaries and enhancing diagnostic precision.

However, challenges such as computational complexity, dataset variability, and model interpretability remain significant. Addressing these challenges will be crucial for further adoption of GNNs in clinical settings. Future research should focus on optimizing GNN architectures for scalability, improving integration with traditional image-based methods, and developing robust interpretability frameworks to enhance trust and usability in medical practice.

Overall, GNNs hold promise for revolutionizing brain tumor segmentation by enabling more accurate, context-aware analysis of medical images, ultimately contributing to improved patient care and treatment outcomes.

Chapter 3

Enhancing U-Net Predictions Using Graph Neural Networks

3.1 Introduction

This chapter outlines the proposed approach of integrating a CNN-based architecture with a GNN for brain tumor segmentation. It explains our methodology. This chapter describes the training of the CNN model for initial segmentation and GNN model, and using several strategies to improve the segmentation.

3.2 Methodology

In our methodology for enhancing CNN segmentation models, we first perform two parallel operations: obtaining the model's predictions after training and generating the graph from dataset samples. Using the model's predictions and the graph, we then generate the mask, which targets the weak regions segmented by the CNN model. These operations serve as the baseline for all subsequent enhancement strategies. We distinguish three strategies for enhancement:

1. **Using GNN Alone:** This strategy involves using Graph Neural Networks to enhance the weak regions identified by the CNN.

2. **1D Convolution for Encoding CNN Predictions with GNN:** This approach employs 1D convolution to encode the CNN prediction labels with graph features, which are then used to train Graph Neural Networks to enhance the weak regions.

3. **GNN Encoding CNN Prediction Labels:** This strategy uses Graph Neural Networks to encode the CNN prediction labels with graph features to improve the

weak regions in the CNN output.

Our approach is valid for any segmentation model based on a CNN. However, we used the U-Net model to validate this methodology. 3.1.

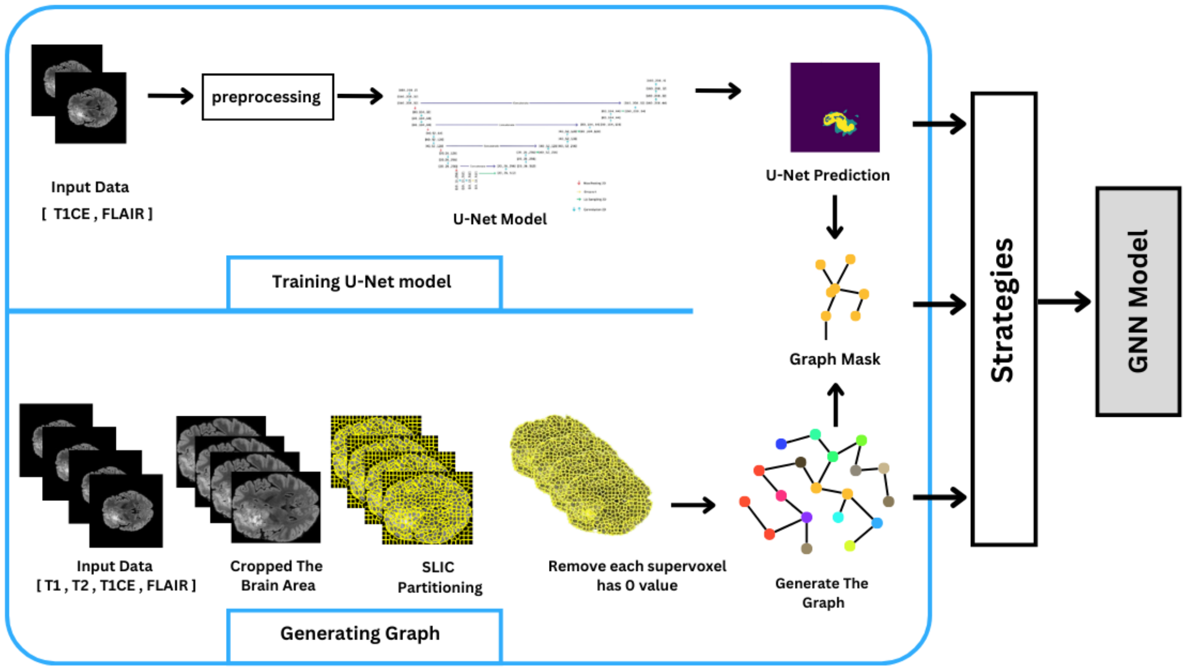


Figure 3.1: Our proposed methodology for enhancing the U-Net model for image segmentation tasks using GNN

3.3 Training U-Net model

In this section, we describe the process of training the U-Net model for multi-model image segmentation tasks. We first explain the U-Net architecture, followed by the preprocessing steps applied to the input data, and finally, we detail the training process.

3.3.1 U-Net Architecture

The U-Net architecture used in this work is designed for multi-class image segmentation tasks. The architecture consists of three main parts: the encoder (contracting path), the bottleneck, and the decoder (expanding path). Here's a detailed explanation:

1. **Encoder (Contracting Path) :** The encoder is responsible for capturing the context of the image through a series of convolutional and pooling layers. It consists

of four blocks, each with two convolutional layers followed by a max-pooling layer, which reduces the spatial dimensions.

- Block 1:
  - 2 Convolutional Layers: 32 filters, kernel size 3x3, ReLU activation, ’same’ padding, He-normal initialization.
  - 1 MaxPooling Layer: pool size 2x2.
- Block 2:
  - 2 Convolutional Layers: 64 filters, kernel size 3x3, ReLU activation, ’same’ padding, He-normal initialization.
  - 1 MaxPooling Layer: pool size 2x2.
- Block 3:
  - 2 Convolutional Layers: 128 filters, kernel size 3x3, ReLU activation, ’same’ padding, He-normal initialization.
  - 1 MaxPooling Layer: pool size 2x2.
- Block 4:
  - 2 Convolutional Layers: 256 filters, kernel size 3x3, ReLU activation, ’same’ padding, He-normal initialization.
  - 1 MaxPooling Layer: pool size 2x2.

2. **Bottleneck :** The bottleneck serves as the bridge between the encoder and the decoder, capturing the most compressed representation of the input.

   - Bottleneck Block:
     - 2 Convolutional Layers: 512 filters, kernel size 3x3, ReLU activation, ’same’ padding, He-normal initialization.
     - 1 Dropout Layer: dropout rate 0.2.

3. **Decoder (Expanding Path) :** The decoder reconstructs the spatial dimensions of the image while combining features from the encoder through skip connections. It consists of four blocks, each with an upsampling layer, concatenation with the corresponding encoder block, and two convolutional layers.

   - Block 1:
     - 1 UpSampling Layer: size 2x2.

  - 1 Convolutional Layer: 256 filters, kernel size 2x2, ReLU activation, 'same' padding, He-normal initialization.
  - Concatenation with corresponding encoder block.
  - 2 Convolutional Layers: 256 filters, kernel size 3x3, ReLU activation, 'same' padding, He-normal initialization.

- Block 2:
  - 1 UpSampling Layer: size 2x2.
  - 1 Convolutional Layer: 128 filters, kernel size 2x2, ReLU activation, 'same' padding, He-normal initialization.
  - Concatenation with corresponding encoder block.
  - 2 Convolutional Layers: 128 filters, kernel size 3x3, ReLU activation, 'same' padding, He-normal initialization.

- Block 3:
  - 1 UpSampling Layer: size 2x2.
  - 1 Convolutional Layer: 64 filters, kernel size 2x2, ReLU activation, 'same' padding, He-normal initialization.
  - Concatenation with corresponding encoder block.
  - 2 Convolutional Layers: 64 filters, kernel size 3x3, ReLU activation, 'same' padding, He-normal initialization.

- Block 4:
  - 1 UpSampling Layer: size 2x2.
  - 1 Convolutional Layer: 32 filters, kernel size 2x2, ReLU activation, 'same' padding, He-normal initialization.
  - Concatenation with corresponding encoder block.
  - 2 Convolutional Layers: 32 filters, kernel size 3x3, ReLU activation, 'same' padding, He-normal initialization.

4. **Output Layer :** The output layer maps the final feature maps to the desired number of classes.

   - Output Layer: 1 Convolutional Layer: 4 filters (for 4 classes), kernel size 1x1, softmax activation.

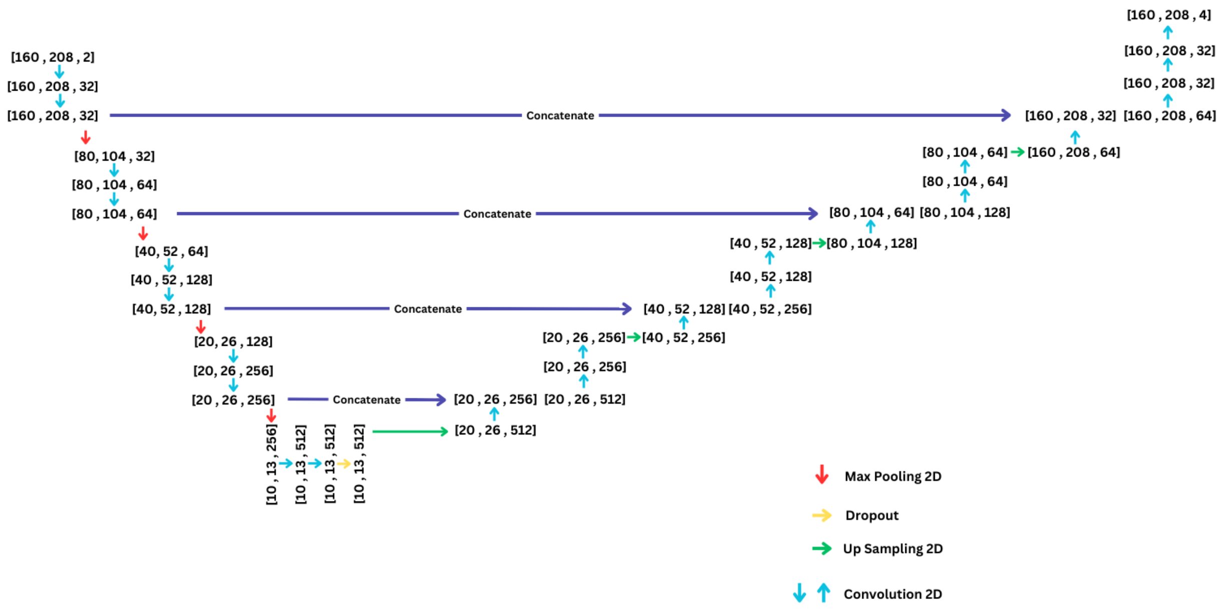


Figure 3.2: U-Net architecture

3.3.2 U-Net Preprocessing

Before training the U-Net model, a series of preprocessing steps were applied to the input data. First, we selected two MRI modalities, T1CE and FLAIR. Each MRI modality was cropped to focus on the brain area, resulting in image with a size of $100 \times 160 \times 208 \times 2$.

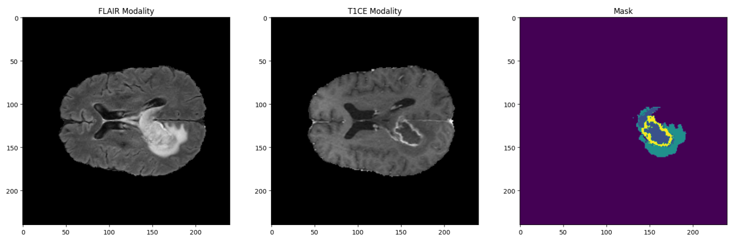


Figure 3.3: Visualize (FLAIR and T1CE) Modalities in (slice = 82) before cropping

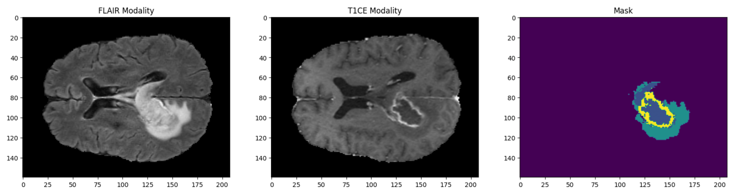


Figure 3.4: Visualize (FLAIR and T1CE) Modalities in (slice = 82) after cropping

3.3.3 U-Net Training

We trained the model using the Adam optimizer [39] with a learning rate of 0.001. The model was trained for 77 epochs with a batch size of $B = 1$.we use categorical cross-entropy loss function [40], given by:

$$L(y, \hat{y}) = -\sum_i y_i \log(\hat{y}_i) \tag{3.1}$$

3.4 Generating graph

The GNN preprocessing aims to generate the graph data structure. we have a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the $\mathcal{V}$ is a set of nodes represent entities and $\mathcal{E}$ i a set of edges. To generate the embedding nodes that represent the image structure , first we cropped the brain area to remove the background, than use SLIC (Simple Linear Iterative Clustering) [41] partitioning to segment the images into supervoxels, which are compact and similar pixel clusters. Any supervoxels containing only zero values are removed to reduce the number of supervoxels. A graph is generated where each supervoxel represents a node, and the connections between them (edges $\mathcal{E}$) represent spatial relationships. The feature matrix $N \times F$ for all supervoxels is represented as an $N \times F$ matrix, where each row corresponds to the feature vector of a supervoxel. This feature matrix is constructed using quintile-based statistics for each supervoxel. For a given supervoxel i with voxel intensity values X_i, the feature vector is computed as $\text{Features}_i = [Q1, Q2, Q3, Q4, Q5]$, where $Q1$ to $Q5$ represent the quintile values of X_i [17]. Thus, each node has 20 features.

supervoxels: are a 3D extension of superpixels, which are groups of pixels in an image that are similar in terms of color, texture, or other features and form contiguous regions [41].

3.5 Graph Mask

U-Net predictions are used to create a mask. Each voxel is assessed, if any voxel within a supervoxel falls below a specified threshold, the entire supervoxel is included in the mask. This step ensures that the mask accurately represents regions of interest as determined by the U-Net model's predictions. The mask can be represented as:

$$M_j = \begin{cases} 1 & \text{if } \min_i P_{ij} < T \\ 0 & \text{otherwise} \end{cases}$$

where P_{ij} as the predicted probability of the pixel i in supervoxel j being part of the region of interest (ROI), as determined by the U-Net model , T as the specified threshold.

3.5.1 Choosing the threshold

Choosing threshold depend of the model itself, Let I be an image in the train or validation set with voxels (x, y, z), and let P be the model's predicted segmentation mask for I, where $P(x, y, z)$ represents the predicted probability of the voxel (x, y, z) . For a given threshold value T, we replace voxel probabilities lower than the threshold with ground truth labels:

$$R(T) = \begin{cases} P(x, y, z), & \text{if } P(x, y, z) \geq T \\ G(x, y, z), & \text{if } P(x, y, z) < T \end{cases}$$

where $G(x, y, z)$ is the ground truth label for the voxel (x, y, z). S_k be the set of voxels belonging to the k-th supervoxel, where $k = \{1, 2, 3, \ldots, K\}$, and K is the total number of supervoxels. For a set of threshold values $\{T_1, T_2, T_3, \ldots, T_n\}$, we construct the modified predicted mask $R(T_i)$. To select the best threshold T_i^* with minimal supervoxel replacements while improving the segmentation mask, we calculated the Dice of each $R(T_i)$, then we choose the best threshold T.

After this process, we obtained a graph containing two types of nodes (supervoxels), those under the threshold and those over the threshold. The graph mask represents the supervoxels under the threshold.

3.6 GNN Model

In this section, we describe the process of training the GNN model for multi-modal image segmentation tasks. We first explain the GNN architecture and then provide a detailed explanation of the training process.

3.6.1 GNN Architecture

The architecture being used is a GraphSAGE, We trained it for a sub-region that U-Net model are ambiguous, The GraphSAGE model consists of the following components:

1. **Input Layer:** This layer takes in the input features and the edge indices.

2. **GraphSAGE Convolutional Layers:** These are a series of convolutional layers that apply a mean aggregation operation over the neighborhood of each node. The number of layers and their respective hidden dimensions are hyperparameters that can be set. SAGConv Layers :
   - (0): SAGEConv(20, 128, aggr=mean)
   - (1-3): 3 x SAGEConv(128, 128, aggr=mean)
3. **Activation Layers:** After each GraphSAGE convolutional layer, a ReLU activation function are applied to introduce non-linearity and regularization.
4. **Fully Connected Layer:** The output of the last GraphSAGE convolutional layer is passed through a fully connected linear layer for the final classification task. (fc): Linear(in-features=128, out-features=4, bias=True)
5. **Output Layer:** A log softmax function is applied to the output of the fully connected layer to obtain the final class probabilities.

3.6.2 GNN Training

We are training a GNN model where a masked region is utilized for backpropagation while the entire graph is used during the forward pass.

The model is trained using the following configurations:

- Number of classes = 4 ,
- Optimizer = AdamW [42] ,
- Learning rate = 0.0001 ,
- Loss function = weighted cross-entropy [43] is :

$$L = -\sum_i w_i \left(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right)$$

  where:

  - w_i is the weight for the i-th sample.
  - y_i is the true label for the i-th sample.
  - $\hat{y}_i$ is the predicted probability for the i-th sample.
  - Class weights w is [0.1900, 0.9970, 0.9100, 0.9930]

3.7 Training Strategies

In this section, we describe our proposed training strategies: using GNN alone, GNN encoding CNN prediction labels, and 1D convolution for encoding CNN predictions with GNN.

3.7.1 Strategy 1: Using GNN Alone

This first strategy (Figure 3.5) serves as the baseline approach, where the Graph directly fed into the GNN model, along with the Mask.

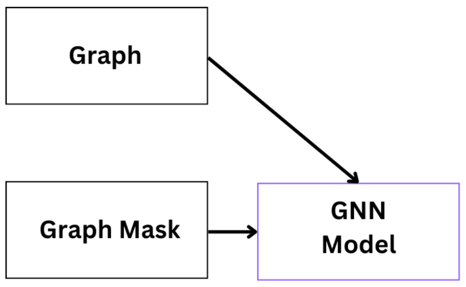


Figure 3.5: Strategy 1: Using GNN Alone

Strategy 1:Intuition

Using this approach to train the GNN allows messages to be passed among all nodes during the message-passing process and requires fewer computational resources.

3.7.2 Strategy 2: 1D CNN for Encoding CNN Predictions labels

In this strategy (see Figure 3.6), we use a pretrained 1D CNN (Figure 3.7), to encode the CNN-predicted labels with the graph features, resulting in new features that contain both the graph feature information and the CNN prediction labels information. These new features are then used to update the graph features. The updated graph, along with the mask, is subsequently fed into the GNN.

Strategy 2 :Intuition

To take advantage of the unmasked regions, where the CNN has high confidence in its segmentation, we pass these labels as additional information along with the graph features to the GNN. We used the CNN to encode these two types of information, as it is known to be powerful in feature extraction [44].

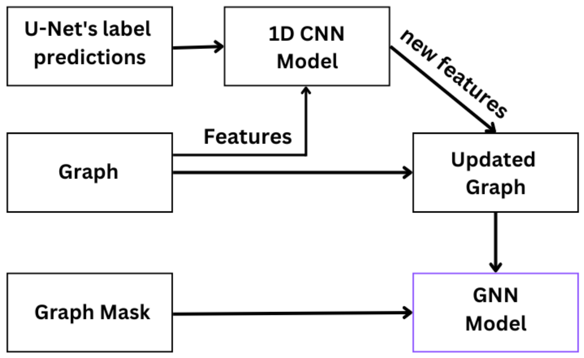


Figure 3.6: Strategy 2: 1D CNN for Encoding CNN Predictions labels

1 D CNN Architecture , Training

This CNN architecture 3.7 is designed for processing 1-dimensional input data. The architecture consists of alternating convolutional layers,activation functions, and max pooling layers, followed by fully connected layers at the end.

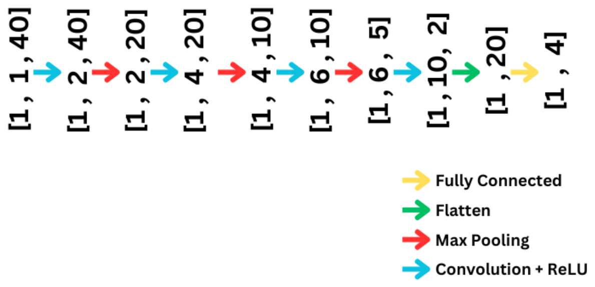


Figure 3.7: 1D CNN Architecture

The 1D CNN initially takes as input a feature vector of a node concatenated with label vector (label occurrences 20 times). During training, the 1D CNN is trained to predict the labels based on this input. After training, The penultimate layer are then utilized for extracting features (20 features for each node).

We use this configuration for training this model :

- Activation function = Relu .
- Loss function = Weighted CrossEntropy.
- Class weights = [0.1900, 0.9970, 0.9100, 0.9930] .

- Optimisation function = Adam [39] .
- Learning rate = 0.001.

3.7.3 Strategy 3: GNN Encoding CNN-Predicted Labels

In this strategy (see Figure 3.8), We concatenate the CNN prediction labels with the graph features, resulting in 21 features for each node. These new features are then used to update the graph features. The updated graph, along with the mask, is subsequently fed into the GNN.

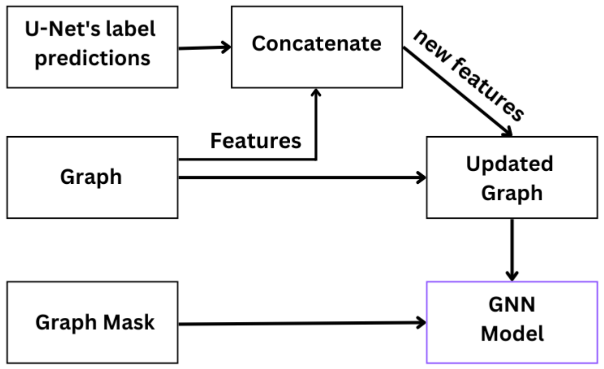


Figure 3.8: Strategy 3: GNN Encoding CNN-Predicted Labels

Strategy 3:Intuition

GNNs are effective for feature embedding [34]. Passing CNN prediction labels will enhance this embedding process and by utilizing the message passing process of the GNN, the nodes will determine their labels based on their neighbors' labels.

3.8 Conclusion

In this chapter, we presented a methodology to enhance CNN-based segmentation models using three distinct strategies that integrate Graph Neural Networks (GNNs) to address weakly segmented regions. The strategies include: (1) Using GNN Alone, (2) employing 1D convolution to encode CNN predictions alongside graph features, and (3) directly encoding CNN prediction labels with GNN. Each strategy aims to boost segmentation accuracy by leveraging GNN's capability in capturing spatial relationships within the data. Our approach is adaptable and applicable to various CNN-based segmentation models, as demonstrated with the U-Net model. Through meticulous preprocessing, model training, and strategic integration of CNN and GNN components, we endeavor to achieve more precise and resilient segmentation results, as discussed in the subsequent chapter.

Chapter 4

Experiments, Results and Discussion

4.1 Introduction

In this chapter, we detail the characteristics of the BraTS 2020 dataset and outline the performance metrics used to evaluate our proposed approaches. We describe our experimental process, which includes training the U-Net model and using it to enhance the GNN model. Before applying the GNN, we validate the mask hypothesis, followed by training the GNN using our developed strategies.

4.2 Dataset

This work utilizes the BraTS 2020 challenge dataset [45], a publicly accessible resource used to train and assess deep learning models for segmenting brain tumors. This dataset consists of multi-institutional, multi-modal brain magnetic resonance imaging (MRI) scans of glioblastoma (GBM) and lower-grade glioma (LGG) patients. The dataset includes pre-operative MRI scans acquired with four different modalities: T1-weighted (T1), T2-weighted (T2), T1-weighted with gadolinium-enhancing contrast (T1ce), and Fluid Attenuated Inversion Recovery (FLAIR) [46]. Each MRI modality contributes to the detection and delineation of different tumor sub-regions [47]:

- **(T1)** useful for detecting the brain area.
- **(T2)** useful for detecting edema (ED).
- **(T1ce)** useful for identifying the enhancing tumor (ET).
- **(FLAIR)** useful for identifying the necrotic tumor core (NET) and edema (ED).

The dataset comprises a training set of 369 cases and a test set of 125 cases. However, the test sets are not segmented. In our work, we split the data into 295 cases for training and 74 cases for test.

4.2.1 Understanding labels

The dataset is carefully annotated by expert radiologists, each voxel must be labeled, we have 4 labels

- **Label 4:** Enhancing tumor (ET).
- **Label 2:** Edema (ED).
- **Label 1:** Necrotic and Non-Enhancing Tumor core (NCR & NET).
- **Label 0:** everything else (Background or brain area).

4.2.2 Class Distribution:

Understanding class distribution is essential for developing machine learning models that can effectively handle for achieving high accuracy in tumor segmentation. Table 4.1 shows the class distribution of BraTS 2020 Datasets.

Table 4.1: Class Distribution in BraTS 2020 Dataset

Class	**Percentage (%)**
Label 0: Background	98.1
Label 1: Non-enhancing Tumor	0.3
Label 2: Edema	0.9
Label 4: Enhancing Tumor	0.7

The dominant class is background (Label 0), constituting 98.1% of the voxels. This is expected as most of the brain is healthy tissue. Tumor voxels (Labels 1, 2, and 4 combined) account for a small fraction (1.9%) of the total volume. Label 1 (0.3%), Label 2 (0.9%), and Label 4 (0.7%) represent the distribution of tumor sub-regions within the entire tumor mask.

4.2.3 Data Visualization

This is a sample of the BraTS 2020 dataset. As shown in Figure 4.1, it includes visualizations of one sample across different slices: 25, 35, and 75.

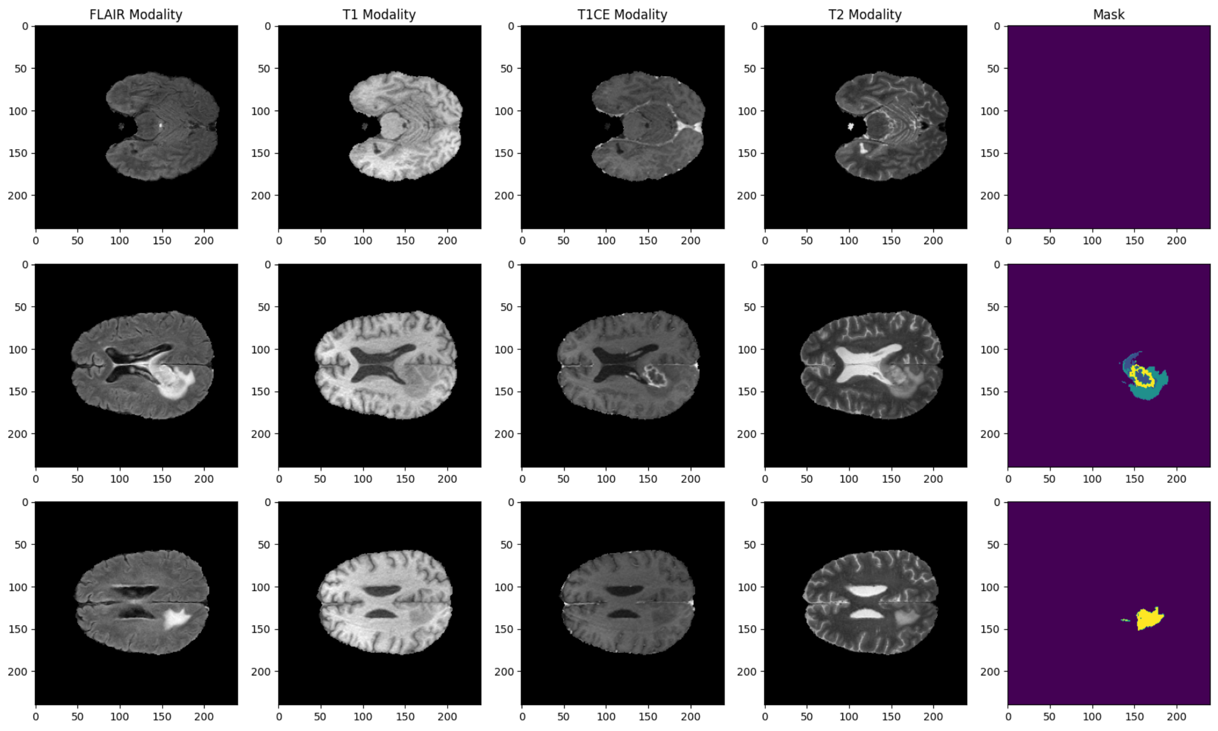


Figure 4.1: Visualize Sample Data Slices (25 , 35 , 75)

4.2.4 Segmentation Target

In the BraTS challenge, the considered sub-regions of the tumor for evaluating models, are shown in Figure 4.2 :

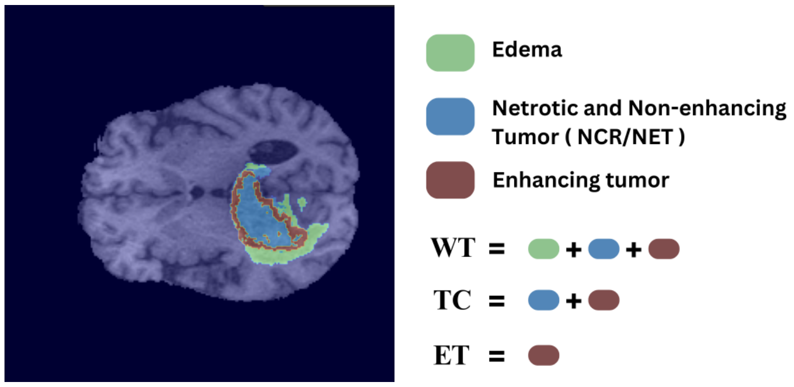


Figure 4.2: Visualize segmentation target (WT , TC , ET).

- **Enhancing Tumor (ET)**: Includes regions labeled with 4.
- **Tumor Core (TC)**: Includes regions labeled with 1 and 4.
- **Whole Tumor (WT)**: Includes all regions labeled with 1, 2, and 4.

4.3 Performance Metrics for Evaluation

In this section, we introduce the performance metrics utilized for evaluating the effectiveness of our proposed brain tumor segmentation model.

4.3.1 Dice coefficient (micro)

The Dice coefficient quantifies the spatial overlap between the predicted and ground truth segmentations. It computes the similarity between the two segmentations, with values ranging from 0 to 1, where 1 indicates perfect overlap. The micro version calculates the Dice coefficient across all classes without considering class-wise variations.

$$\text{Dice(A, B)} = \frac{2 \times |A \cap B|}{|A| + |B|} \tag{4.1}$$

4.3.2 Dice coefficient (macro)

The macro version of the Dice coefficient calculates the average Dice coefficient across multiple classes. It accounts for class-wise variations in segmentation performance and provides insights into the segmentation accuracy for individual classes.

$$\text{Dice(macro)}(A, B) = \frac{1}{N} \sum_{i=1}^{N} \frac{2 \times |A_i \cap B_i|}{|A_i| + |B_i|} \tag{4.2}$$

Where:

- A : is the set of pixels/voxels in the ground truth segmentation, and
- B : is the set of pixels/voxels in the predicted segmentation.
- $|A_i|$ (Intersection size of ground truth)
- $|B_i|$ (Intersection size of predicted segmentation)
- $|A_i \cap B_i|$ represents the intersection of pixels/voxels labeled as class i in both sets.
- $|A_i \cup B_i|$ represents the union of pixels/voxels labeled as class i in both sets.
- N is the total number of classes.

4.3.3 Model Assessment Levels

In all our experiments, we evaluate models at two levels,These are specified as follows:

- **Node level:** After converting the image into a graph.
- **Image level:** After converting the graph into an image.

U-Net / U-Net GNN (Node): Evaluates the U-Net / U-Net GNN model at the node level, **U-Net / U-Net GNN (Image):** Evaluates the U-Net / U-Net GNN model at the image level.

4.4 Platform and Environment

The experiments were conducted using Google Colab Pro and Kaggle environments due to their accessibility and computational resources. Both Google Colab Pro and Kaggle offer CPU and GPU acceleration, enabling efficient training of deep learning models.

Training and evaluation were performed on GPU-accelerated instances provided by Google Colab Pro and Kaggle. The hardware specifications included P100, or T4 GPUs, depending on the availability and runtime configuration.

4.5 Experiments

In this section we explore our experiments and discussion results , start with training U-Net then GNN training for each strategies. Our aim is to investigate various methods to enhance GNN performance by leveraging high-confidence segmentation regions identified by the U-Net.

4.5.1 Training U-Net

This is results of training U-Net model,The performance metrics for the test set Table 4.2
.

Table 4.2: U-Net model Performance on the Test Set

Metric	Loss	Dice Macro	Dice Micro	WT Dice	TC Dice	ET Dice
Value	0.0326	0.8294	0.7104	0.7320	0.7037	0.6956

4.5.2 The mask hypothesis validation

In our experiments, we assumed that the mask region represents the weak regions predicted by the CNN model, identified using a specific threshold. This hypothesis can be validated by replacing the mask region with the ground truth labels and observing the resulting improvement. Using U-Net, the mean size of the mask regions represents 14.7% of the whole image. After correcting these regions, the results are as follows : Table 4.3

Table 4.3: The mask hypothesis validation results

	Dice WT	Dice TC	Dice ET
Before correcting	0.6686	0.6429	0.6617
After correcting	0.9987	0.9993	0.9998

These results demonstrate that even though the mask regions are relatively small, correcting them leads to substantial improvements, thereby validating our mask hypothesis.

4.5.3 Simple experiment:(strategy one)

In this experiment, we implemented strategy one, in which we set the degree for each node to be of degree 10. The degree is determined by calculating the spatial distance between the node and other nodes and then selecting the k nearest neighbors. In this case, k is set to 10.

Result Discussion

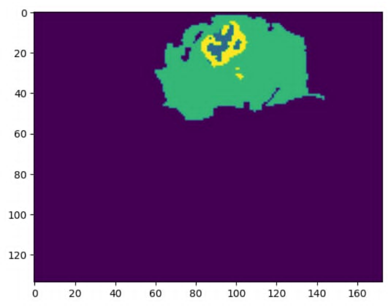
The SLIC partitioning algorithm generates a set of supervoxels $S = \{s_1, s_2, \ldots, s_m\}$, where each supervoxel s_i is a subset of voxels from $V = \{v_1, v_2, \ldots, v_n\}$, and each voxel v_j has one label l_j.

Initially, each supervoxel s_i may contain voxels with different labels. The GNN model enforces the constraint that each supervoxel s_i should have a single label L_i, such that all voxels $v_j \in s_i$ are assigned the same label L_i. $f(v_j)$ is the original label of voxel v_j. The GNN model predicts a unique label L_i for each supervoxel s_i.

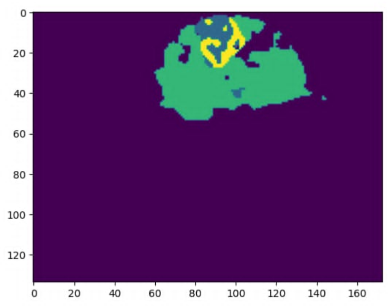
After the GNN prediction, to reconvert the graph representation to image representation, the labels of all voxels $v_j \in s_i$ are updated to be L_i:

$$\forall v_j \in s_i, \quad l_j = L_i$$

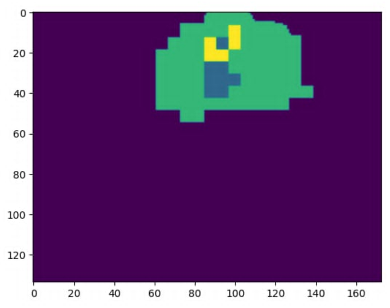
However, this process may lead to a problem where each supervoxel s_i is assigned a single label L_i, but within that supervoxel, there may be voxels with different original labels $f(v_j)$. we train GNN using the same configuration as we can see Table 4.4.



(a) Ground Truth



(b) U-Net prediction



(c) U-Net GNN prediction

Figure 4.3: visualize results before updated SLIC partitioning.

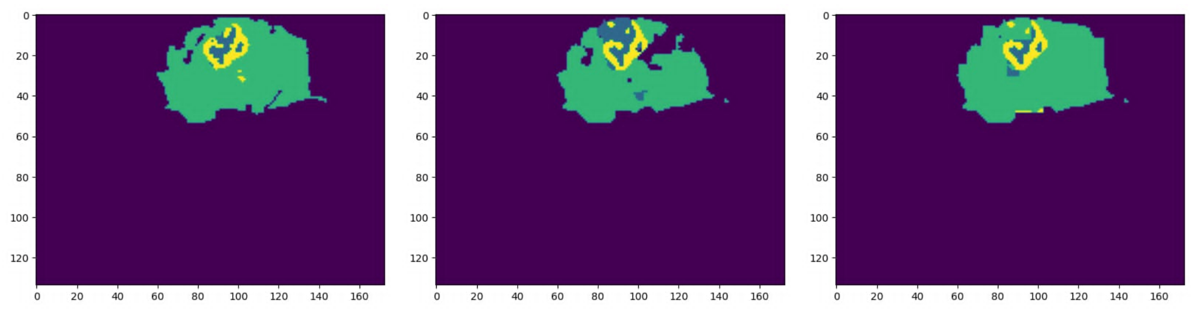
Table 4.4: Model Evaluation results before updated SLIC partitioning.

Model	Dice (Micro)	Dice (Macro)	Dice WT	Dice TC	Dice ET
U-Net (Node)	0.9903	0.7793	0.7557	0.7297	0.7386
U-Net GNN (Node)	0.9904	0.7922	0.7674	0.7464	0.8187
U-Net (Image)	0.8294	0.7104	0.7320	0.7037	0.6956
U-Net GNN (Image)	0.8258	0.6483	0.6592	0.6064	0.6538

Update the SLIC partitioning

Update the SLIC partitioning based on the U-Net predictions. $P(v_j)$ is the U-Net prediction of voxel v_j, For each supervoxel $s_i \in S$,if there exists at least one voxel $v_j \in s_i$ with multiple predicted labels from the U-Net, i.e., $\exists v_j \in s_i$ such that $|P(v_j)| > 1$, then split s_i into individual voxels ,each voxel is supervoxel (node) in the updated partitioning. The GNN model is then retrained on the updated partitioning S, allowing it to predict labels for individual voxels when necessary, instead of enforcing a single label for an entire supervoxel with conflicting predictions.as we can see :

Note: Logically, after the last update, the number of nodes is greater , yet it still requires the same amount of learning time and resources, maximum 2GB GPU RAM.



(a) Ground Truth (b) U-Net prediction (c) U-Net GNN prediction

Figure 4.4: visualize results after updated SLIC partitioning.

Table 4.5: Model Evaluation results after updated SLIC partitioning.

Model	Dice (Micro)	Dice (Macro)	Dice WT	Dice TC	Dice ET
U-Net (Node)	0.9654	0.7153	0.6686	0.6430	0.6617
U-Net GNN (Node)	0.9746	0.7668	0.7515	0.7179	0.7318
U-Net (Image)	0.8294	0.7105	0.7320	0.7037	0.6957
U-Net GNN(Image)	0.8297	0.7202	0.7466	0.7207	0.7271

4.5.4 Selecting the best level of neighborhood for GNN

In our graph, we observed that the structure of the network was relatively simple Figure 4.5. This simplicity may have limited the ability of the nodes to communicate effectively with distant neighbors. As a result, the model might not have fully leveraged the spatial dependencies present in the data, potentially impacting overall performance. We experimented with different levels of neighbor degree in the graph, where the neighbor degree determines how far the connections extend from each node.

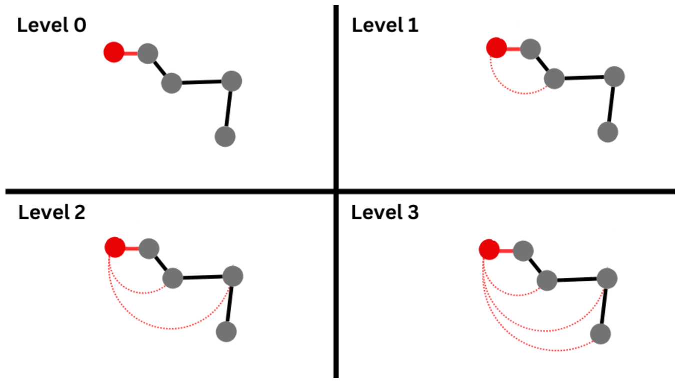


Figure 4.5: Explore four different level of neighbors

A neighbor degree of 0 means that a node only considers its direct neighbors, while higher degrees mean that the node also considers the neighbors of its neighbors, and so on Figure 4.5. In this experiment, we set the neighbor degrees to 0, 1, 2, and 3.

Result and Discussion

Table 4.6: Performance Metrics and Time per Epoch for Each Level degree

Level of Neighbors	Dice WT	Dice ET	Dice TC	Time per Epoch (minutes)
$Level_0$	0.7566	0.7396	0.7171	1.512
$Level_1$	0.7647	0.7468	0.7309	1.948
$Level_2$	0.7717	0.7505	0.7453	2.532
$Level_3$	0.7742	0.7465	0.7416	4.885

After evaluating these results, we discussed the trade-offs between training time and model performance. We chose level 2 as the best option because it strikes a balance between achieving good performance and maintaining an acceptable training time. Specifically, level 2 showed the highest performance metrics (WT, ET, TC) compared to levels 0 and 1, and its training time per epoch is significantly lower than that of level 3, making it a more efficient choice for our needs.

4.5.5 Enhancing GNN Performance through High-Confidence CNN Segmentation Regions

Based on our initial hypothesis, the unmasked regions represent areas where the CNN has high confidence in its segmentation. Therefore, these regions can provide additional information to the GNN based on the CNN predictions. By leveraging these strong regions identified by the CNN, we can enhance the GNN's performance.

Using 1d CNN to encode CNN model prediction with graph features:(strategy two)

We implemented strategy two, in which we used a 1-dimensional CNN to encode CNN model predictions with graph features. In this strategy, we explored two different training approaches for the 1D CNN:

1. **Training on Ground Truth Labels:** The labels used to encode and predict are the ground truth.

2. **Training on CNN Predicted Labels:** The labels used to encode and predict are the labels predicted by the CNN model.

Result and Discussion

This Table 4.7 describe results of training the model using the first approach (Training 1D CNN model on Ground Truth Labels)

Table 4.7: Model Evaluation results using the first approach

Model	**Dice Micro**	**Dice Macro**	**Dice WT**	**Dice TC**	**Dice ET**
U-Net (Node)	0.9655	0.7153	0.6686	0.6430	0.6617
U-Net GNN (Node)	0.9679	0.7393	0.7080	0.6763	0.6714
U-Net (Image)	0.8294	0.7105	0.7320	0.7037	0.6957
U-Net GNN (Image)	0.8284	0.7061	0.7316	0.7046	0.6940

This Table 4.8 describe results of training the model using the second approach (Training 1D CNN model on CNN Predicted Labels)

Table 4.8: Model Evaluation results using the second approach

Model	**Dice Micro**	**Dice Macro**	**Dice WT**	**Dice TC**	**Dice ET**
U-Net (Node)	0.9655	0.7153	0.6686	0.6430	0.6617
U-Net GNN (Node)	0.9715	0.7500	0.7228	0.6852	0.6658
U-Net (Image)	0.8294	0.7105	0.7320	0.7037	0.6957
U-Net GNN (Image)	0.8298	0.7141	0.7413	0.7070	0.6844

Although the use of a 1D CNN to encode features with CNN model predictions is intended to enhance the training of the GNN, training the 1D CNN with ground truth labels yields better results. This improvement is due to the inaccuracies in the CNN model's predictions. When the 1D CNN is trained on ground truth labels, it learns to encode the features accurately in relation to the true labels. In contrast, training on the predicted labels, which may contain errors, leads to inconsistencies between the features and the labels. These inconsistencies hinder the 1D CNN's ability to effectively learn and encode the correct relationships, thereby impacting the overall performance of the model.

GNN Encoding CNN Prediction Labels :(strategy three)

We implemented Strategy 3, in which the GNN encodes the labels along with the features. For the masked regions, where the labels have low confidence in their segmentation, we tried different initializations in these regions before passing them to the GNN to encode them with the features:

1. **Keep the Same Labels:** In this experiment, we concatenated the feature nodes with their associated labels, even in the masked regions.

$$\mathbf{h}_i = [\mathbf{f}_i; y_i]$$

where $\mathbf{f}_i$ represents the feature vector of node i and y_i represents its label.

2. **Initialization with a Non-Existent Label:** In this experiment, we concatenated the feature nodes with their associated labels, but in the masked regions, we used a non-existent label, specifically label 4.

$$\mathbf{h}_i = \begin{cases} [\mathbf{f}_i; y_i] & \text{if } i \text{ is not masked} \\ [\mathbf{f}_i; 4] & \text{if } i \text{ is masked} \end{cases}$$

3. **Initialize the Nodes with the Mean of Their Neighbors' Labels:** In this experiment, we concatenated the feature nodes with their associated labels, but in the masked regions, each node takes the mean of its neighbors' labels.

$$\mathbf{h}_i = \begin{cases} [\mathbf{f}_i; y_i] & \text{if } i \text{ is not masked} \\ \left[\mathbf{f}_i; \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} y_j\right] & \text{if } i \text{ is masked} \end{cases}$$

where $\mathcal{N}(i)$ represents the set of neighbors of node i.

4. **Initialize the Nodes with the Mean of Their Neighbors' Labels with Respect to the Labels' Distributions:** In this experiment, we concatenated the feature nodes with their associated labels, but in the masked regions, each node takes the weighted mean of its neighbors' labels. The weights are relative to the labels' distribution in the dataset.

$$\mathbf{h}_i = \begin{cases} [\mathbf{f}_i; y_i] & \text{if } i \text{ is not masked} \\ \left[\mathbf{f}_i; \frac{\sum_{j \in N(i)} w_m \cdot y_j}{\sum_{j \in L} |N(i)| \cdot w_j}\right] & \text{if } i \text{ is masked} \end{cases}$$

where w_m is the associated weight for the label of y_j, and L is the label. and w_j is the weighted mean of neighbors.

Result and Discussion

Table 4.9: Model Evaluation Results of 4 GNN Encoding CNN Prediction Labels Experiment

Model	Dice Micro	Dice Macro	Dice WT	Dice TC	Dice ET
Keep the Same Labels					
U-Net (Node)	0.9655	0.7153	0.6686	0.6430	0.6617
U-Net GNN (Node)	0.9766	0.7929	0.7731	0.7427	0.7483
U-Net (Image)	0.8294	0.7105	0.7320	0.7037	0.6957
U-Net GNN (Image)	0.8305	0.7410	0.7654	0.7443	0.7420
Initialization with a Non-Existent Label					
U-Net (Node)	0.9655	0.7153	0.6686	0.6430	0.6617
U-Net GNN (Node)	0.9780	0.7844	0.7728	0.7340	0.7466
U-Net (Image)	0.8294	0.7105	0.7320	0.7037	0.6957
U-Net GNN (Image)	0.8310	0.7328	0.7640	0.7353	0.7369
Initialize the Nodes with the Mean of Their Neighbors' Labels					
U-Net (Node)	0.9655	0.7153	0.6686	0.6430	0.6617
U-Net GNN (Node)	0.9770	0.7892	0.7685	0.7454	0.7484
U-Net (Image)	0.8294	0.7105	0.7320	0.7037	0.6957
U-Net GNN (Image)	0.8308	0.7405	0.7621	0.7430	0.7451
Initialize the Nodes with the Mean of Their Neighbors' Labels with Respect to the Labels' Distributions					
U-Net (Node)	0.9655	0.7153	0.6686	0.6430	0.6617
U-Net GNN (Node)	0.9776	0.7905	0.7749	0.7489	0.7553
U-Net (Image)	0.8294	0.7105	0.7320	0.7037	0.6957
U-Net GNN (Image)	0.8311	0.7419	0.7676	0.7477	0.7525

The results are quite close, but we observed that the best performance was achieved when we initialized the nodes with the weighted mean of their neighbors' labels. This approach prevents bias in the node labels and effectively incorporates the neighbors' labels into consideration. This strategy is superior to other strategies, demonstrating that GNN excels in utilizing the UNet prediction labels information and as a classifier.

4.5.6 conclusion

The experiments conducted in this chapter represent a significant step forward in leveraging the strengths of both convolutional neural networks (CNNs) and graph neural networks (GNNs) for the challenging task of brain tumor segmentation. By combining the high-confidence segmentation regions identified by a pre-trained U-Net model with the contextual modeling capabilities of GNNs, we were able to achieve improved segmentation accuracy across multiple evaluation metrics.

Among the various strategies explored, the most promising approach involved encoding the U-Net prediction labels directly into the GNN node features. Specifically, initializing the masked nodes with the weighted mean of their neighbors' labels, while considering the relative distributions of different labels, proved to be the most effective initialization strategy. leading to more accurate and robust segmentation results.

Notably, the integration of the U-Net predictions into the GNN framework consistently outperformed the standalone U-Net and GNN models, highlighting the synergistic benefits of combining these complementary architectures. The GNN demonstrated its strength as a classifier by effectively utilizing the U-Net prediction labels, while the U-Net provided reliable initial segmentations, particularly in high-confidence regions.

While the improvements in segmentation accuracy are encouraging, it is important to note that the experiments also revealed trade-offs between model complexity, computational efficiency, and performance. Strategies that involved more sophisticated neighborhood connectivity or feature encoding techniques often came at the cost of increased training times and computational requirements.

Overall, the findings presented in this chapter underscore the potential of combining deep learning architectures like CNNs and GNNs for medical image analysis tasks, paving the way for more advanced and robust models that can potentially improve patient care and treatment outcomes.

General Conclusion

This thesis presents an in-depth investigation into enhancing Graph Neural Networks (GNNs) for image segmentation tasks. The primary aim was to explore the integration of CNN and GNN methodologies to achieve improved segmentation performance, particularly in complex tasks such as medical image analysis.

Key findings include the validation of the mask hypothesis, where unmasked regions in CNN predictions corresponded to areas of high confidence, and replacing these regions with ground truth labels significantly improved performance. Two encoding strategies were explored, and it was found that training with ground truth labels yielded better results due to the elimination of inaccuracies present in CNN predictions. Additionally, an optimal neighbor degree of 2 was identified, providing a balance between performance and computational efficiency.

The study demonstrated that integrating high-confidence CNN segmentation regions with GNN features can significantly enhance performance by leveraging the strengths of both CNNs and GNNs. Importantly, the addition of the GNN did not significantly affect the overall complexity of the approach, as the GNN is a lightweight model.

However, it is important to note the limitations of our approach:

- The performance is highly dependent on the quality of the CNN model. Better CNN performance leads to better subsequent GNN results.
- The approach depends on the selection of a threshold for masking. A high threshold may result in the entire graph being included in the mask, while a low threshold may include only a few nodes, which does not enhance the CNN model effectively.

Overall, this research contributes to medical image segmentation by introducing a novel approach that combines the predictive strengths of CNNs with the structural modeling capabilities of GNNs. The findings suggest that hybrid models can potentially lead to better clinical outcomes and more efficient workflows in medical practice.

Bibliography

[1] Xavier Intes. *Time-Domain Optical Imaging*. Springer, 2012.

[2] Martin Thoma. A survey of semantic segmentation, 2016.

[3] Will L. Hamilton. *Graph Representation Learning*, volume 14 of *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool Publishers, 2020.

[4] Jure Leskovec. Cs224w: Machine learning with graphs. https://web.stanford.edu/class/cs224w/, 2024. Accessed: 2024-06-13.

[5] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2017.

[6] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Pearson, 4th edition, 2018.

[7] Hisham Kasban, Peter L. Munk, and Bing Xiang. *Image-Guided Interventions*. Expert Radiology Series. Elsevier, 2015.

[8] Gaurav Sharma and S. Srinivasan. *Digital Signal Processing: Principles and Applications*. Tata McGraw-Hill Education, 2010.

[9] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, Demetri Terzopoulos, and Matias Valdenegro-Toro. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[10] Fabian Isensee, Philipp Kickingereder, Wolfgang Wick, Martin Bendszus, and Klaus H. Maier-Hein. Brain tumor segmentation and radiomics survival prediction: Contribution to the BRATS 2017 challenge. In *International MICCAI Brainlesion Workshop*, pages 287–297. Springer, 2018.

[11] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen van der Laak, Bram Ginneken, and Clara

Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42, 02 2017.

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. 2015.

[13] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-Net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571. IEEE, 2016.

[14] Fabian Isensee, Paul F. Jaeger, Simon A. A. Kohl, Jens Petersen, and Klaus H. Maier-Hein. nnu-net: The no-new-net universal for semi-supervised medical image segmentation. In *International Workshop on Machine Learning in Medical Imaging*, pages 63–72. Springer, 2021.

[15] Oliver Saueressig, Sebastian Höfener, Thomas Hänsel, Patrick Schubert, Gregor Weber, Michael Günther, Franz-Josef Schmitt, and Volker Lohweg. Exploring graph networks for 3d medical image segmentation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 449–450. IEEE, 2020.

[16] Huaizu Jiang, Qi Dou, Varut Vardhanabhuti, and Pheng-Ann Heng. Automatic brain tumor segmentation using cascaded anisotropic convolutional neural networks. In *International MICCAI Brainlesion Workshop*, pages 42–52. Springer, 2020.

[17] G. Huang, Z. Liu, L. Maaten, and K. Q. Weinberger. Advancing self-supervised and semi-supervised learning with simclr. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[18] Shuo Wang, Wenqing Sun, and Xiaohong Wang. DUNet: Towards deeper u-nets. *Medical Image Analysis*, 75:102263, 2022.

[19] D. W. McRobbie, E. A. Moore, M. J. Graves, and M. R. Prince. *MRI from Picture to Proton.* Cambridge University Press, 3rd edition, 2017.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016.

[21] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[22] Artem Rozantsev, Vincent Lepetit, and Pascal Fua. Sagnet: Learning attention from human for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[23] Geert Litjens, Thijs Kooi, Babak E. Bejnordi, Arnaud A. A. Setio, Francesco Ciompi, Mohsen Ghafoorian, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, 2017.

[24] Andrew P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 2007.

[25] Richard Adams and Leonard Bischof. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):641–647, 1994.

[26] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[27] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

[28] M. R. Islam, Y. Zhang, S. Alelyani, and E. Keogh. Gated recurrent neural networks for electricity price forecasting. In *Proceedings of the 2017 IEEE International Conference on Big Data (Big Data)*, pages 2273–2282. IEEE, 2017.

[29] Yijun Yuan, Meina Chao, and Chi Lo. Hierarchical attention networks for document classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1482–1492, 2017.

[30] Zhi Chen, Kamran Sedig, and Shuiwang Ji. Graph neural networks: A review of methods and applications. *AI Open*, 2(1):15–52, 2021.

[31] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias P. Heinrich, Kazunari Misawa, and Daniel Rueckert. Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*, 2018.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2019.

[33] Zongwei Zhou, Md. Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. UNet++: A nested U-Net architecture for medical image segmentation. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 3–11. Springer, 2018.

[34] Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. 09 2016.

[35] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[36] David Ahmedt-Aristizabal, Mohammad Ali Armin, Simon Denman, Clinton Fookes, and Lars Petersson. A survey on graph-based deep learning for computational histopathology. *Computerized medical imaging and graphics : the official journal of the Computerized Medical Imaging Society*, 95:102027, 2021.

[37] Kexin Ding, Mu Zhou, Zichen Wang, Qiao Liu, Corey W. Arnold, Shaoting Zhang, and Dimitris N. Metaxas. Graph convolutional networks for multi-modality medical imaging: Methods, architectures, and clinical applications. 2022.

[38] Xuanao Cui, Yuhang Zheng, Xuewen Wang, Shaohua Li, Bin Zhang, and Yao Yao. Multi-class brain tumor segmentation using graph attention network. *Medical Image Analysis*, 75:102265, 2022.

[39] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[40] Zhilu Zhang and Mert R. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels, 2018.

[41] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurélien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels. *Technical report, EPFL*, 06 2010.

[42] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[43] Anqi Mao, Mehryar Mohri, and Yutao Zhong. Cross-entropy loss functions: Theoretical analysis and applications, 2023.

[44] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

[45] Spyridon Bakas, Mauricio Reyes, Andras Jakab, Stefan Bauer, Markus Rempfler, Alessandro Crimi, Russell T. Shinohara, Caroline Berger, Sung Min Ha, Martin Rozycki, et al. Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the BraTS challenge. In *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, pages 20–38. Springer, 2020.

[46] Bjoern H. Menze, Andras Jakab, Stefan Bauer, Jayashree Kalpathy-Cramer, Keyvan Farahani, Justin Kirby, Yuliya Burren, Nicole Porz, Johannes Slotboom, Roland Wiest, et al. The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). In *IEEE Transactions on Medical Imaging*, volume 34, pages 1993–2024. IEEE, 2015.

[47] Bjoern H. Menze, Andras Jakab, Stefan Bauer, Jayashree Kalpathy-Cramer, Keyvan Farahani, Justin Kirby, Yuliya Burren, Nicole Porz, Johannes Slotboom, Roland Wiest, et al. The multimodal brain tumor image segmentation benchmark (brats). *IEEE Transactions on Medical Imaging*, 34(10):1993–2024, 2015.

[48] Petar Velickovic, Aleksandar Fedorov, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. Graph attention networks. In *International Conference on Learning Representations*, 2019.

[49] Hengshi Dong, Guang Yang, Fang Liu, Yilong Mo, Yurun Guo, Liang Xie, and Dinggang Shen. Automatic brain tumor detection and segmentation using u-net based fully convolutional networks. *arXiv preprint arXiv:1705.03820*, 2017.

[50] Risheng Wang, Tao Lei, Ruixia Cui, Bingtao Zhang, Hongying Meng, and Asoke K. Nandi. Medical image segmentation using deep learning: A survey. *IET Image Processing*, 16(5):1243–1267, January 2022.

[51] Jie Zhou, Guodong Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lihui Wang, Chao Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[52] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[53] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.