


People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

AMO Bouira University

Science and Applied Sciences Faculty

Computer Science department

Master's Degree Thesis

in Computer Science

Major: *Computer Systems Engineering*

Theme

Semantic segmentation of medical images using deep learning

Supervised by

- Chihati Sarah

Realized by

- Abidat Mohamed
- Lachemat Houssam Eddine Othman

2020/2021

Acknowledgments

First and foremost, We give thanks to **Allah**, the almighty who has given us courage and patience during this work.

Our sincerest thanks go to our supervisor *Ms. Sarah Chihati* for her precious advice. Thank you for your trust, your availability and your encouragement.

We also extend our heartfelt thanks to the members of the jury who accepted to evaluate our work. As well as all the profesors who ensured our path in university courses.

Finally, we would also like to thank our families and all those who have encouraged and supported us with the ups and downs throughout this work.

Thank you all.

Dedication

In the name of **ALLAH** the clement and the merciful praise to him the Almighty.

I dedicate this modest work as a sign of respect, gratitude and thanks:

To my dear parents, my dear siblings and my dear family.

To all those who accompanied and supported me and to those who have contributed from near or far to the realization of this work.

To the best of the teammates, for all the good times we spent together.

To all my friends.

may God bless you all.

ABIDAT Mohamed.

Dedication

I have the honor to dedicate this modest work to those who have encouraged and supported me, those who always believed in me.

To my friend *Ferdi Habib Taha*, and for all the dreams we shared together, may Allah have mercy on his soul.

My dear Father, the source of my energy, he has always been and still beside me.

My dear Mother, the reason for my standing here with you, she has always given me the hope of living and all the support I needed.

My brother and sisters, for their unlimited advice, they had a big role in my arrival here.

My teammate for his hard work, it was an honor to work with you.

For all my family and friends.

may God bless you all.

LACHEMAT Houssam Eddine Othman.

ملخص

التجزئة الدلالية هي مهمة رؤية الكمبيوتر التي تعتبر كائنات من نفس الفئة ككيان واحد. لديها مجال كبير من التطبيقات مثل القيادة الذاتية ، والتعرف التلقائي على الأنسجة المرضية والمزيد ... سنركز على مشكلة تجزئة ورم الدماغ والتي تعتبر واحدة من أصعب مشاكل التجزئة والمهام التي تستغرق وقتًا طويلاً في المجال الطبي. حيث نرى أنه ربما يمكن لنظام تجزئة أوتوماتيكي لورم الدماغ أن يغطي بعض هذه الصعوبات. في هذه الدراسة ، نسعى جاهدين لاقتراح نظام BTS تلقائي يعتمد على التصوير بالرنين المغناطيسي (MRI) . حيث نعتمد على الشبكات العصبية التلافيفية (CNN) في بناء أنظمتنا.

اقترحنا ثلاثة مقاربات مختلفة لـ BraTS20 وطريقتين لـ BraTS17 . تم تقييم هذه النماذج باستخدام دالة معامل النرد.

كلمات مفتاحية: التجزئة الدلالية، تجزئة ورم الدماغ، الشبكات العصبية التلافيفية، التصوير بالرنين المغناطيسي.

Abstract

Semantic segmentation is a computer vision task that consider the objects of the same class as one entity. it has a large domain of applications such as autonomous driving, automatic identification of pathological tissues and more.... We will focus on **brain tumor segmentation** (BTS) problem which is considered as one of the most difficult segmentation problems and time consuming tasks in the medical domain, where we see that an automatic brain tumor segmentation system might be able to deal with some of these difficulties. In this study, we strive to propose an automatic BTS system based on Magnetic resonance imaging (MRI). where we rely on convolutional neural networks (CNN) in building our systems.

We proposed three different approaches for BraTS20 and two approaches for BraTS17. These models were evaluated using dice score and yielded encouraging results.

Key words: Semantic segmentation, brain tumor segmentation, Magnetic resonance imaging, convolutional neural networks.

Résumé

La segmentation sémantique est une tâche de vision par ordinateur qui considère les objets de la même classe comme une seule entité. elle a un large domaine d'applications telles que la conduite autonome, l'identification automatique des tissus pathologiques et plus encore . . . Nous nous concentrerons sur le problème de **segmentation des tumeurs cérébrales** qui est considéré comme l'un des problèmes de segmentation les plus difficiles dans le domaine médical. où nous voyons qu'un système de segmentation automatique des tumeurs cérébrales peut aider à surmonter ces difficultés. Dans cette étude, nous nous efforçons de proposer un système de segmentation automatique des tumeurs cérébrales basé sur **l'imagerie par résonance magnétique** (IRM). où nous comptons sur **les réseaux de neurones convolutifs** (CNN) pour construire nos systèmes.
Nous avons proposé trois approches différentes pour BraTS20 et deux approches pour BraTS17. Ces modèles ont été évalués à l'aide de la fonction de score aux dés.

Mots clés: La segmentation sémantique, segmentation des tumeurs cérébrales, imagerie par résonance magnétique, les réseaux de neurones convolutifs.

Contents

List of Figures

List of Tables

Abbreviations list

MRI	Magnetic Resonance Imaging.
CT	Computed Tomography.
GBM	Glioblastoma Multiform
PET	Positron Emission Tomography.
MR	Magnetic Resonance.
3d	3 Dimensions
FDG	FluoroDeoxyGlucose.
DL	Deep Learning
CNN	Convolutional Neural Networks
ML	Machine Learning
AI	Artificial Intelligence
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
CPU	Central Processing Unit
MLP	Multi-layer Perceptron
ANN	Artificial Neural Network
RNN	Recurrent Neural Network
1d	1 Dimension
RGB	Red Green Blue
SGD	Stochastic Gradient Descent
FCN	Fully Connected Network
ReLU	Rectified Linear Unit
CRF	Conditional Random Fields

MICCAI	Medical Image Computing and Computer Assisted Intervention
GT	Ground Truth
OI	Output Image
TN	True Negative
FN	False Negative
TP	True Positive
FP	False Positive
FSPGR	Fast Spoiled Gradient Echo
WT	Whole Tumor
ET	Enhancing Tumor
TC	Tumor Core
RCNN	Regions with Convolutional Neural Networks
YOLO	You Only Look Once
RAM	Random Access Memory
API	Application Programming Interface
2D	2 Dimensions
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
FC	Fully Connected

General introduction

The brain is one of the most important organs in the human body, it controls every action we do with high precision. Any malfunction in the brain negatively affects the body, and sometimes leads to death. This could happen because of many diseases, among them, there is brain tumor disease.
Although brain tumors are less common than other cancers such as lung cancer or prostate cancer, they are among the top 10 leading causes of death worldwide and have a long-term and psychological impact on patients' lives even when they survive.

Magnetic Resonance Image (MRI) is the most effective and widely used method for diagnosing brain tumors. it uses magnetic fields and radio waves to generate internal images with detailed information of the body organs.
Significant advances in medical science have been made in recent years as a result of Artificial Intelligence and Deep Learning in particular, such as the Medical Image Processing technique, which allows doctors to diagnose diseases early and easily, whereas previously it was tedious and time-consuming.

To overcome such limitations, computer-aided technology is critical, because the medical field requires efficient and dependable techniques to diagnose life-threatening diseases including brain tumors. So, our goal is to propose an automatic segmentation approach for brain tumors from MR images using deep learning.

Our dissertation is organized as follows:

Chapter 1: Brain tumors and medical imaging.
In this chapter, we will provide information about the brain tumor disease, its types,

causes, treatment and more. We will also talk about the different used imaging techniques and their importance when it comes to brain tumors.

Chapter 2: Deep Learning

In this chapter, we will talk about Deep learning and its background, the different types of learning in DL and we will explain some of the relevant DL architectures like Convolutional neural network.

Chapter 3: Semantic segmentation of brain tumors using deep learning: Review

In this chapter, we will talk about segmentation and its types, the most common techniques that are used for segmenting brain tumors and the results of some of the recent related works to brain tumor segmentation task.

Chapter 4: Our approaches

This chapter has two sections, In the first section we will talk about the environment of development, the used datasets, their description and preprocessing. The second section will include the different architectures that we chose to work with, and our proposed models.

Chapter 5: Results and discussion

In the last chapter, we will talk about the implementation phase and discuss the obtained results of each model.

Finally, we close our work with a general conclusion.

Chapter 1

Brain tumors and medical imaging

1.1 Introduction:

Cancer is a leading cause of death worldwide, accounting for nearly 10 million deaths in 2020 [1]. In Algeria, 50,000 new cases and 20,000 deaths have been recorded in 2019. The head of the oncology department at Mustapha Pacha Hospital estimated that 70,000 new cases per year will be recorded by 2030. [2]

Brain and other nervous system cancers are among the most dreadful and life-threatening diseases–they are the 10th leading cause of death for men and women. A recent brain tumor study estimated that 18,600 adults (10,500 men and 8,100 women) will die from primary cancerous brain and central nervous system tumors this year in the United States alone. [3]

Medical imaging techniques such as Magnetic resonance imaging (MRI) and computed tomography (CT) scan are routine clinical practices for brain tumor diagnosis and treatment planning, as they provide doctors with valuable information about brain tissues.

In this chapter, we will define brain tumors, their symptoms and possible causes. We will also present the brain imaging tests and treatments commonly used by physicians.

1.2 Definition of a brain tumor

A brain tumor is a collection of abnormal cells in the brain [4]. Brain tumors are categorized as primary or secondary. Primary tumors arise from brain cells, the meninges, which are the membranes that cover the brain, glands, and nerve cells. Gliomas and meningiomas are the two most common forms of adult brain tumors [5, 6]. Secondary brain tumors, also known as metastatic brain tumors, develop when cancer cells from other organs such as the lungs and kidneys migrate to the brain. Both types of tumors are life-threatening and can cause permanent disability. [7]
A Brain tumor can be graded into several stages based on the abnormalities found in its cells and tissues. This grading indicates the likelihood of the tumor growing and spreading. [8]

Grade I - The tumor cells look more like normal cells under a microscope and grow and spread more slowly than grade II, III, and IV tumor cells. They rarely spread into nearby tissues. Grade I brain tumors may be cured if they are completely removed by surgery.[9]

Grade II - The tumor cells grow and spread more slowly than grade III and IV tumor cells. They may spread into nearby tissue and may recur. Some tumors may become a higher-grade tumor.[9]

Grade III - The tumor cells look very different from normal cells under a microscope and grow more quickly than grade I and II tumor cells. They are likely to spread into nearby tissue.[9]

Grade IV - The tumor cells do not look like normal cells under a microscope and grow and spread very quickly. There may be areas of dead cells in the tumor. Grade IV tumors usually cannot be cured.[9]
Brain tumors can also be classified into:

Low-grade tumors (grades I & II) - are not very aggressive and are usually associated with long-term survival.

High-grade tumors (grade III & IV) - grow more quickly, can cause more damage, and are often more difficult to treat. These are considered malignant or cancerous. One of the most known grade IV tumors is the glioblastoma multiform (GBM for short).

1.3 Symptoms of brain tumors

The symptoms of a brain tumor vary depending on the tumor's location and size. Some tumors infiltrate brain tissue and inflict direct damage, while others put pressure on the surrounding brain. The most common symptoms are headaches, loss of balance, weakness or paralysis in one part or one side of the body, personality or behavior changes, vision changes, hearing loss, difficulty in thinking and speaking, and memory loss. [10]

1.4 Causes of brain tumors

Although the exact causes of brain tumors are still unknown, researchers are studying the probability that people with certain risk factors are more likely to develop a brain tumor. Risk factors for brain tumors are discovered by performing analytic epidemiologic studies, which typically compare either brain tumor risk in participants with or without certain characteristics or the history of participants with or without brain tumors.[11]
The following are the major brain tumors risk factors:

Genetics - Many researchers have turned their attention to genetic risk factors, in part due to growing understanding of the molecular pathology of brain tumors, especially glioma, and in part due to new technologies for examining genotype-disease associations. While familial glioma aggregation has been demonstrated, distinguishing shared environmental exposures from inherited characteristics may be difficult. [11]

Age - Brain tumors are most common in older adults. However, a brain tumor can occur at any age.[12]

Radiation exposure - People who have been exposed to ionizing radiation are at a higher risk of developing a brain tumor. Ionizing radiation includes cancer radiation

therapy and atomic bomb radiation exposure. Cell Phone radiofrequency radiation has not been linked to brain tumors. [12]

1.5 Brain tumors diagnosis

1.5.1 physical exam

A thorough medical history is essential for a doctor, as it can help him to determine the duration and severity of an illness. Most brain tumor patients have symptoms that were initially misdiagnosed as other illnesses, depression, or stress. Patients frequently complain of fatigue, lack of concentration, or nausea, for example. Patients and doctors may dismiss these non-localized symptoms as unimportant. Localizing symptoms, such as speech disturbance, weakness on one side of the body, or seizures, are more likely to indicate a neurological problem. Signs of a brain tumor on a physical examination can include weakness as well as a tremor, coordination problems on both sides of the patient's body, or jerking movements of their eyes. [13]
When the physical examination of a patient is positive, doctors perform different medical imaging techniques to confirm the diagnosis.

1.5.2 Medical imaging techniques

Imaging is crucial in the clinical management of human brain tumors. Standard magnetic resonance imaging (MRI) and computed tomography (CT) can easily assess diagnostic anatomical features such as the presence (or absence) of contrast enhancement, hemorrhage, calcification, and/or macroscopic necrosis. CT and MRI are also used to evaluate secondary pathologic conditions such as mass effect, edema, and herniation. Serial imaging with gadolinium-enhanced MRI is routinely used to assess therapy success or failure, as well as for post-therapy surveillance to detect early tumor recurrence. [14]
Advanced magnetic resonance imaging (MRI) and positron emission tomography (PET) imaging provide physiologic, metabolic, and functional information about tumor biology that goes beyond the diagnostic yield of standard MRI and CT imaging. Each advanced MRI modality, including MR perfusion, MR diffusion, and MR spectroscopy techniques, and each PET radiotracer, beginning with FDG (fluorodeoxyglucose) and including mul-

tiple tracers for tumor cellular proliferation, hypoxia, and amino acid transport, reveals valuable information about various aspects of brain tumor metabolism. [14]

Computed Tomography

Tomography is the photography of an object in sections or slices using any type of wave. Computed Tomography (CT) is a computer-assisted imaging process that produces cross-sectional images. Using a software, slices of image sections are superimposed on one another to create a 3D digital image. CT scans are widely used in the detection of abnormal tumor growth, tumor stages, and tumor recurrence. [15]

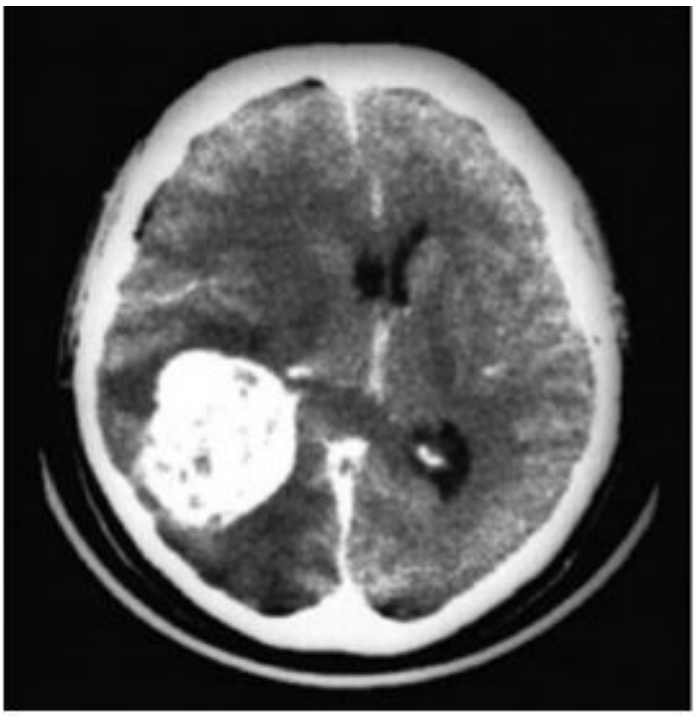

Figure 1.1: CT brain tumor image. [16]

Advantages of Computed Tomography

Noninvasive, quick, and painless.

Good spatial resolution.

Global view of veins.

Distinguished by small differences in physical density.

Avoids invasive insertion of an arterial catheter and guidewire. [17]

Disadvantages of Computed Tomography

Exposure to ionizing radiation, this increases the possibility of developing cancer later in life.

No real-time information.

Cannot detect intra-luminal abnormalities.

Cannot be performed without contrast (allergy, toxicity).

Less contrast resolution where soft tissue contrast is low. [17]

Magnetic Resonance Imaging

MRI is a powerful soft tissue diagnostic technique. A strong and uniform magnetic field, as well as radiofrequency waves, are required for an MRI system. The scanner delivers a suitable resonant radiofrequency to the patient. The waves travel through the tissues or any region of the body that contains hydrogen atoms, i.e., water molecules. The atom is excited and then returns to equilibrium using the energy from the oscillating magnetic field, which is captured by the scanner and digitally processed. As a result, MRI is best suited for visualizing soft tissues, tendons, and ligaments. [15]

MRI can also be used to detect certain brain lesions. Contrast agents, such as gadolinium, are used to distinguish minute differences/changes in body structures. The main advantage of using MRI is that it allows changing the contrast of the image. A small change in the radio wave frequency and magnetic field can change the contrast of the image, highlighting different types of tissues. Another advantage of MRI is that it can create images in any plane (axial/horizontal), which CT cannot do. [15]

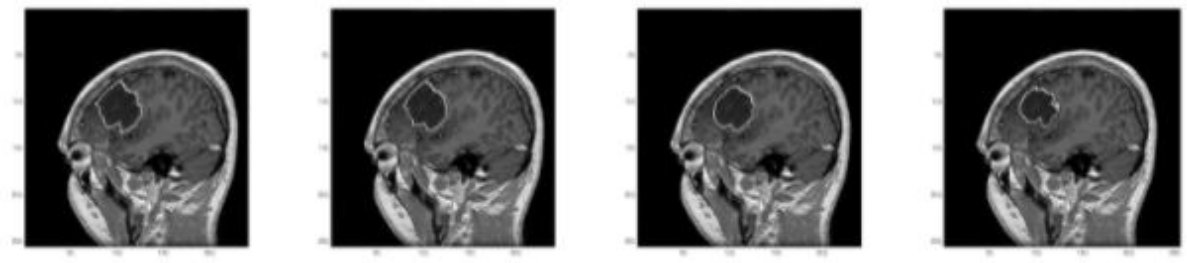

Figure 1.2: Example of a brain tumor in MRI image segmented by different experts.[18]

Advantages of Magnetic Resonance Imaging

Noninvasive and painless.

Without ionizing radiation.

High spatial resolution.

Operator independent.

Easy to blind and ability to measure flow and velocity with advanced technique.

Can be performed without contrast (pregnancy allergy).

Good soft-tissue contrast. [17]

Disadvantages of Magnetic Resonance Imaging

Relatively low sensitivity.

Long scan and post-processing time.

Mass quantity of the probe may be needed.

No real-time information.

Cannot detect intra-luminal abnormalities.

Can make some people feel claustrophobic.

Sedation may be required for young children who can't remain still.

Relatively expensive. [17]

Positron Emission Tomography (PET)

PET scans are not typically used for diagnosis, but they can assist doctors in estimating the grade of a tumor. It may also be used to distinguish between recurrent tumor cells, cells killed by radiation, and scar tissue in some cases. The PET machine resembles a CT scanner. The added diagnostic value of PET imaging over standard MRI and CT remains unknown. PET imaging has the potential to depict metabolic information about brain tumor biology that anatomic techniques (MRI, CT) alone do not provide, but the incremental diagnostic yield from PET imaging of pathophysiologic processes such as glucose metabolism, cellular proliferation, hypoxia, and amino acid transport in brain tumors has not been established.[14]

The complementary value of metabolic information from PET with anatomic and physiologic information from MRI will hopefully become more clearly defined with MRI and PET co-registration and simultaneous image acquisition using hybrid PET-MRI devices.[14]

1.5.3 Histopathological Image Analysis

Unlike the previous medical imaging techniques, histopathology is an invasive test that studies the disease signs using microscopic examination of a brain biopsy. Biopsy is a surgical procedure in which the surgeon removes a small amount of tissue and sends it to the laboratory for analysis [13]. The collected samples are put under a microscope to visualize their different components; the sections are dyed with one or more stains. The aim of staining is to reveal cellular components; counter-stains are used to provide contrast.[19]

1.6 Treatment of brain tumors

Treatment decisions are individualized by an experienced multidisciplinary team consisting of medical oncology, radiation oncology, and neurosurgery. Treatment decisions are based on tumor type and location, malignancy potential, and the patient's age and physical condition.[20] Treatment may require only surveillance but commonly includes surgery, radiotherapy, chemotherapy, or a combination, and enrollment in clinical trials should be offered as an option for some high-grade tumors.[20]

1.6.1 Surgery

The standard treatment for primary brain tumors is the maximal safe surgical removal of the tumor followed by radiotherapy and chemotherapy. Although the extent of resection is a prognostic variable, the extent of safe tumor resection is dependent on tumor location, patient performance status, and, most importantly, patient age. Benefits of maximal resection include relief of mass effect, decreased tumor burden, improved diagnosis, and a trend toward prolonged survival.[20]

1.6.2 Radiotherapy

Radiotherapy can be used as primary treatment or adjunctively following surgical resection. Standard fractionated external beam radiotherapy is the most common approach, although other options include brachytherapy, fractionated stereotactic radiotherapy, and stereotactic radiosurgery. Hypofractionation of radiotherapy may be considered for older

or immunocompromised patients. Radiotherapy can improve progression-free survival and overall survival in patients with high-risk low-grade gliomas, defined as patients younger than 40 years with subtotal resection or biopsy, or patients older than 40 years with any degree of resection.[20]

1.6.3 Radiosurgery

Multiple radiation beams are focused on a very small area. Each radiation beam isn't particularly powerful, but the point where all the beams come together –the tumor– receives a massive dose of radiation. Radiosurgery is typically performed as a single treatment.[12]

1.6.4 Chemotherapy

Chemotherapy given in combination with radiation has been shown to improve survival in selected cases. For example, carmustine wafers (Gliadel), or temozolomide (Temodar) in younger patients, placed during surgery have improved survival in patients with high-grade gliomas. [20]

1.6.5 Targeted drug therapy

Targeted drug therapies target specific abnormalities in cancer cells. One type of targeted therapy prevents the formation of new blood vessels, thereby cutting off the tumor's blood supply. Another type inhibits an enzyme that aids in the growth of cancer cells. [12]

1.7 Conclusion

Medical imaging techniques have made a significant difference in patient care. With increased precision in disease detection and surgical procedures, they quickly became a standard pre-treatment a pre-surgery procedure. Before planning a chemotherapy, a radiotherapy or a surgery, the physician needs to delineate the exact boundaries of the tumor and its sub-regions. In practice, this is a is very difficult task because: tumor tissues exhibit significant shape and appearance variations, the segmentation process is expensive and sensitive to inter and intra-expert variation (fig.I.2), and the quality of

the produced medical images may vary between hospitals. Therefore, the automatic segmentation of pathological tissues has gained significant interest in the medical imaging research community. In the next chapter, we will present and dive into the details of deep learning–a ubiquitous method in the realm of automatic medical image analysis.

Chapter 2

Deep learning

2.1 Introduction:

Nowadays, Deep learning (DL) approaches have become the new state of art in medical image segmentation due to their effectiveness and good results they deliver in this field. In this chapter, we will first define deep learning and talk about the different types of learning in DL. Then, we will talk about the more relevant DL architectures in general and about Convolutional neural networks (CNN) architecture in detail. Last but not least, we will briefly explain the backpropagation algorithm and the role of optimization functions.

2.2 Deep learning

2.2.1 Definition

Deep learning is a subset of machine learning (ML) which itself is a subset of artificial intelligence (AI), it depends on Artificial neural networks. These networks attempt to simulate the behavior of the human brain, where they learn from vast amounts of data to make approximate predictions.

The major difference between DL and ML is that deep learning learns to extract relevant features from raw data, while machine learning methods rely on hand-crafted features.

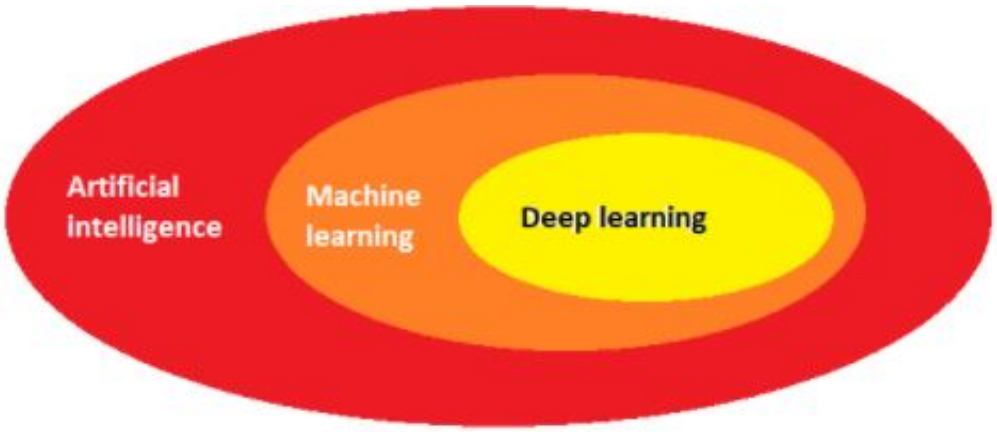

Figure 2.1: The relation between AI, ML and DL.

2.2.2 Background

Looking at the revolution that deep learning is making nowadays, people might think that DL is a recent discovery. It might be surprising to know that DL history dates back to the 1940s. It has been under various name changes, including cybernetics (during 1940-1960), connectionism (during 1980-1990) and deep learning (since 2006). Based on the idea of trying to mimic the behavior of the human brain in a simpler computational model, its goal is to build models that would learn automatically and provide predictions using some input data.

It all started in 1943 with the first artificial neuron that attempted to mimic the biological one. Whalter Pitts and Warren McCulloch introduced the McCulloch-Pitts Neuron, which was based on a linear model that would take various inputs [X1 ,X2 . . . Xn], for each input the model had some weights [W1, W2 . . . Wn] where the output is f(x,w) = X1W1 + X2W2 + . . . + XnWn.This model could only output True or False based on the input and weights. It has very limited capability and has no learning mechanism where the weights are tuned manually.

In 1957, a model called Perceptron, with learning capabilities to do binary classification was innovated by Frank Rosenblatt, Frank built Perceptron for image recognition. The problem with Perceptron is that it was merely a linear classifier unable to solve nonlinear problems which led this model to fall in 1969.[21]

In 1960, Henry J. Kelley introduced the first-ever backpropagation model [22]. Two years later, Stuart Dreyfus came with a backpropagation modal that uses simple derivative chain rule instead of the dynamic programming which was used in Henry's model. [23]

The year of 1965 was the birth of the multilayer perceptron that uses a polynomial activation function, introduced by Alexey Grigoryvich and Valentin Grigoryvich.

In 1970, Seppo Linnainmaa implemented backpropagation in computer code, the research in backpropagation has now come very far.

Ten years later, in 1980, Kunihiko Fukushima came up with Neocognitron [24], the first convolutional neural network architecture which could recognize visual patterns such as handwritten characters. CNN started using backpropagation in 1988 to recognize handwritten digits, it was the foundation of modern Computer vision using deep learning.

In 2014 Ian Goodfellow introduced GANs [25], which opened new doors of application of deep learning in fashion, art, science due to its ability to synthesize real data.

The internet has a principal role in collecting huge data, which has a big impact on the evolution of deep learning algorithms, an example of that is ImageNet. Also advances in hardware have driven deep learning to the next level, where GPUs reduced training time significantly and surpassed the performance of CPUs. Nowadays, the world of deep learning sees an aggressive progress, where it has started being used in mostly all our activities, shopping, watching movies etc...

2.3 Types of learning

2.3.1 Supervised learning

Supervised learning allows the algorithms to learn from labelled data relying on a "supervisor" that provides supervision to the algorithms to predict/classify the desired output correctly. [26] To illustrate this type, we will use the classification task as an example:

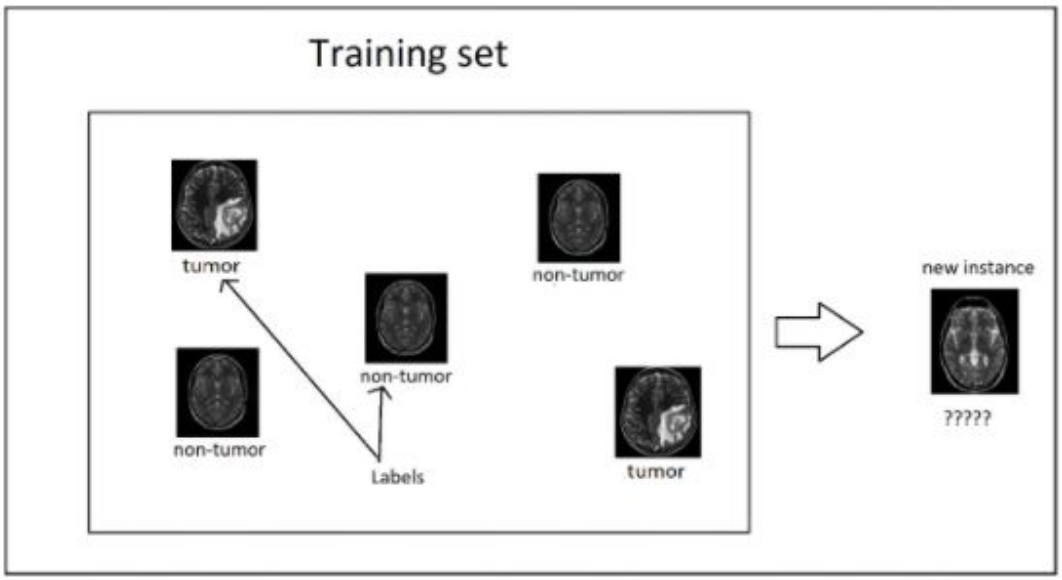

Figure 2.2: Brain tumor classification.

2.3.2 Unsupervised learning

There is no supervisor (teacher) in unsupervised learning, only the input data is known, and no known output data (label) is given to the algorithm. [26] which means that the input data is not explicitly labeled. These algorithms try to use techniques on the input data to mine for rules, detect patterns, and summarize and group the data points which help in deriving meaningful insights and describe the data better to the users. [27] To illustrate this type, we will use this example:
We want to group the patients relying on their points of interest or features, for example, their age, gender, common chronic disease, Tumor progression rate etc... So we will apply a clustering algorithm to our data to divide these patients into groups.

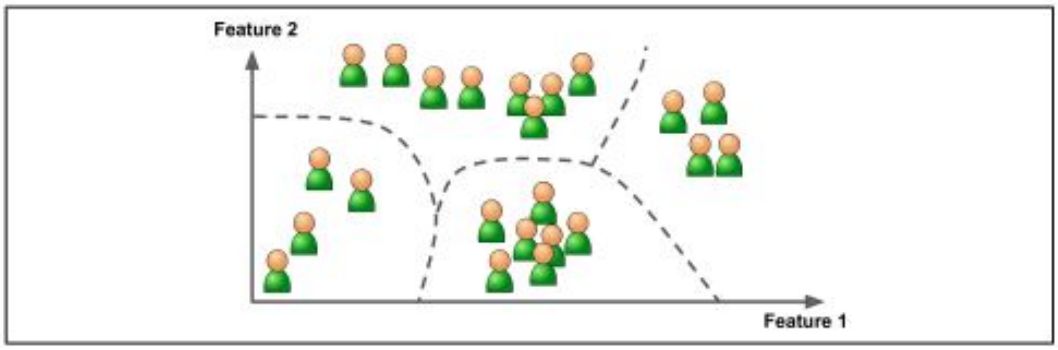

Figure 2.3: Clustering. [28]

2.3.3 Semi-supervised learning

Labelling data is time-consuming and expensive, which requires skilled human experts to do that. Semi-supervised learning algorithms are combinations of unsupervised learning and supervised learning, in other words, labelled data and unlabeled data. These algorithms can deal with data that's partially labelled. [28]

The algorithm will train itself using a limited set of labelled data to get a "partially trained" algorithm, where this algorithm will label the unlabeled data to use it.

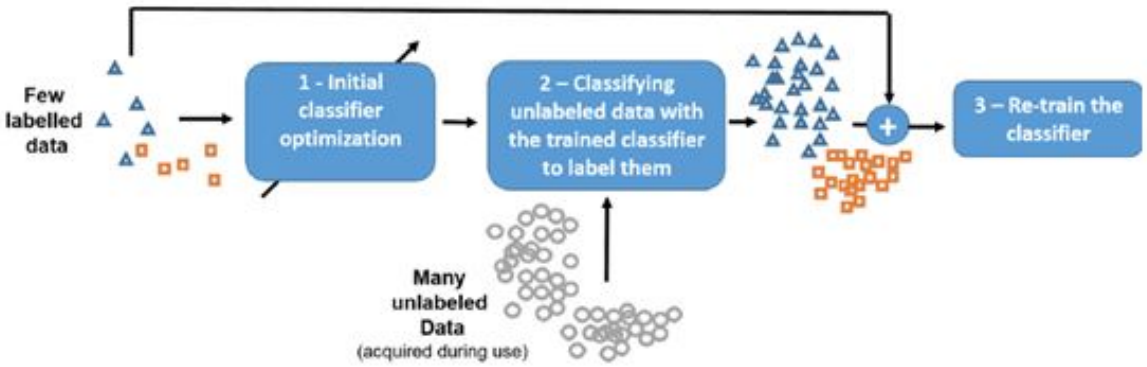

Figure 2.4: Principle of semi-supervised learning [29]

2.3.4 Reinforcement learning

The learning system is called an agent in this context, can observe their dynamic environment, select and perform a sequence of actions a1, a2, . . ., at overtime, and get rewards in return or penalties. [30]

The goal of the algorithm is to learn to act in a way that maximizes the rewards and minimizes the penalties. So, it must then learn by itself to determine the ideal behavior and what is the best strategy (called a policy) within a specific context. [28]

The common algorithms that rely on these techniques are: *Q-learning*, *Temporal difference* and *deep adversarial learning*.

The figure below will illustrate this learning type:

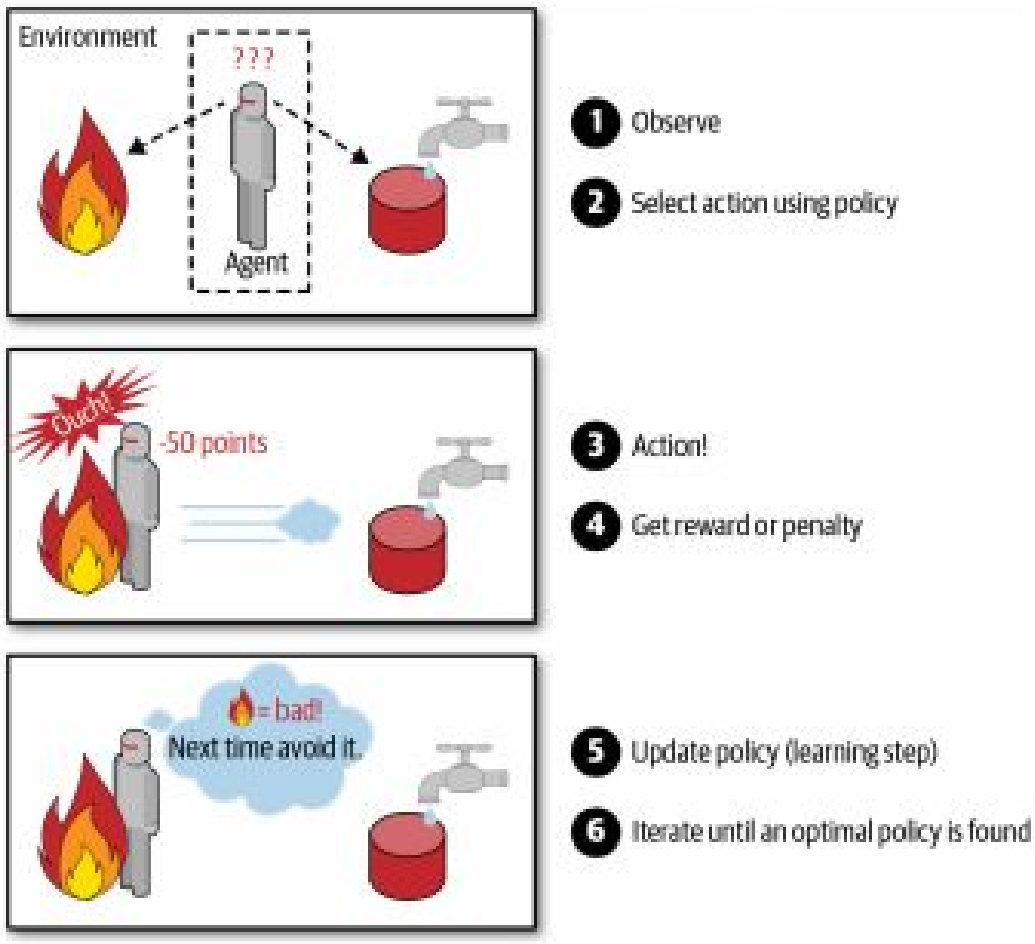

Figure 2.5: Reinforcement learning [28]

2.4 Deep Learning Architectures

Here are some examples of DL architectures:

2.4.1 Multi-layer perceptron (MLP)

MLP is a collection of connected computational units or nodes called neurons arranged in multiple computational layers (one input layer, one or more hidden layers and one output layer). Each neuron linearly combines its inputs and then passes it through an activation function, which can be a linear or nonlinear filter. Linear combination of inputs is performed by summing up the products of weights and inputs. ANN generates the target through feed-forward data flow and then updates the weights of each neuron by backpropagation of errors (it will be discussed later) during the training iterations. [31]

The figure below shows an MLP with one input layer, two hidden layers and an output layer. Each layer contains an infinite number of neurons.

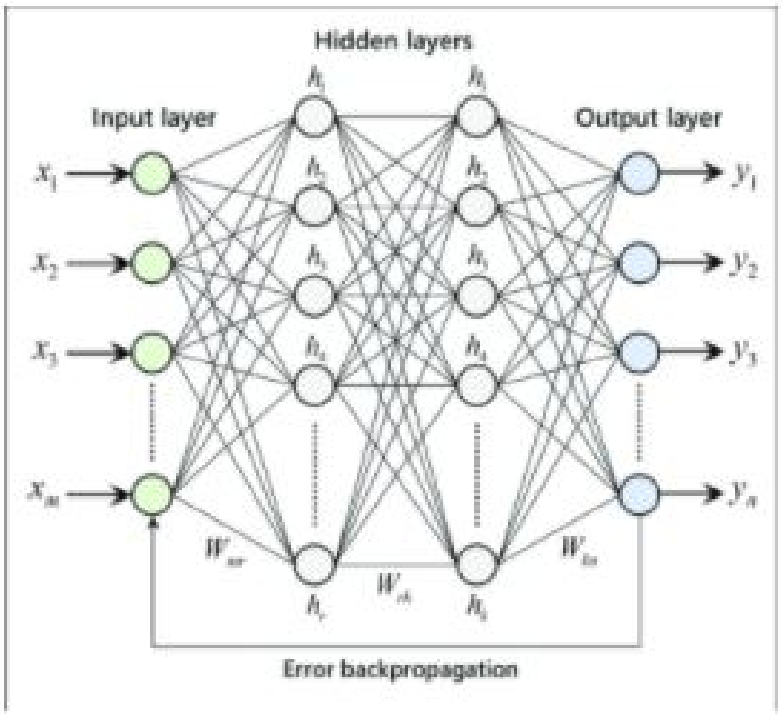

Figure 2.6: Multi-layer perceptron [32]

2.4.2 Recurrent Neural Network (RNN)

RNNs are suitable for datasets that contain sequential data as well as for Natural languages processing tasks, such as language modelling, text generation, or auto-completion of sentences. Recurrent Neural Network remembers the past and its decisions are influenced by what it has learnt from the past. [33]

Feedforward networks such as CNN remember things too, but they remember things they learnt during training and use this knowledge to generate outputs. While RNN remembers things learnt from prior input(s) and uses it while training. [34]

RNNs use other data points in a sequence to make better predictions. They do this by taking in input and reusing the activations of previous nodes or later nodes in the sequence to influence the output. [35]

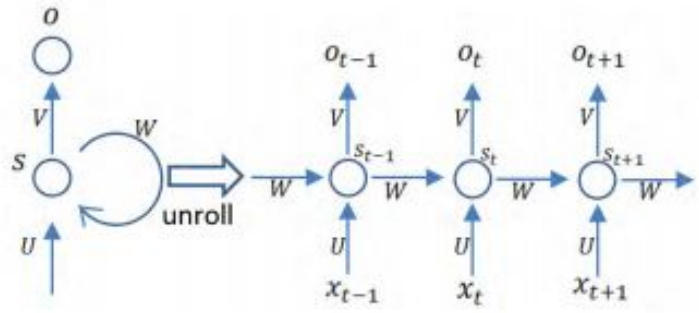

Figure 2.7: The enrolling of RNN in time [36]

2.4.3 Generative Adversarial Network (GAN)

GANs are algorithmic architectures that use two neural networks, One neural network called the generator, generates new data instances (samples) while the second neural network is the discriminator which evaluates them for authenticity with the original data, and predicts a label or category to which that input data belongs. [25]

They are used widely in image generation, semantic image editing, style transfer and classification.

The figure below (Fig.II.9) will illustrate this algorithm:

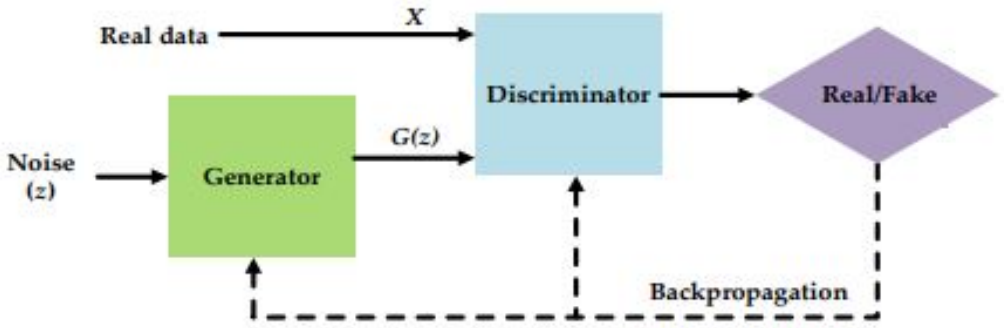

Figure 2.8: Standard GAN architecture [37]

2.4.4 Convolutional Neural Network (CNN)

CNNs are regularized versions of multilayer perceptrons commonly used in computer vision and image recognition. This type of algorithm employs a mathematical operation called convolution (we will discuss it later), and that is where the name CNN came from.

The CNN algorithm takes an input image which is usually represented by a matrix; applying pooling and convolution operations will extract the features of the image that allow the computer to identify and differentiate an object from another [21]. Fig.II.10 depicts a CNN architecture that is used to classify brain MRI images. Each layer of a CNN converts the input volume to an output volume of neuron activation, eventually leading to the final fully connected layers, which result in a mapping of the input data to a 1D feature vector.[38]

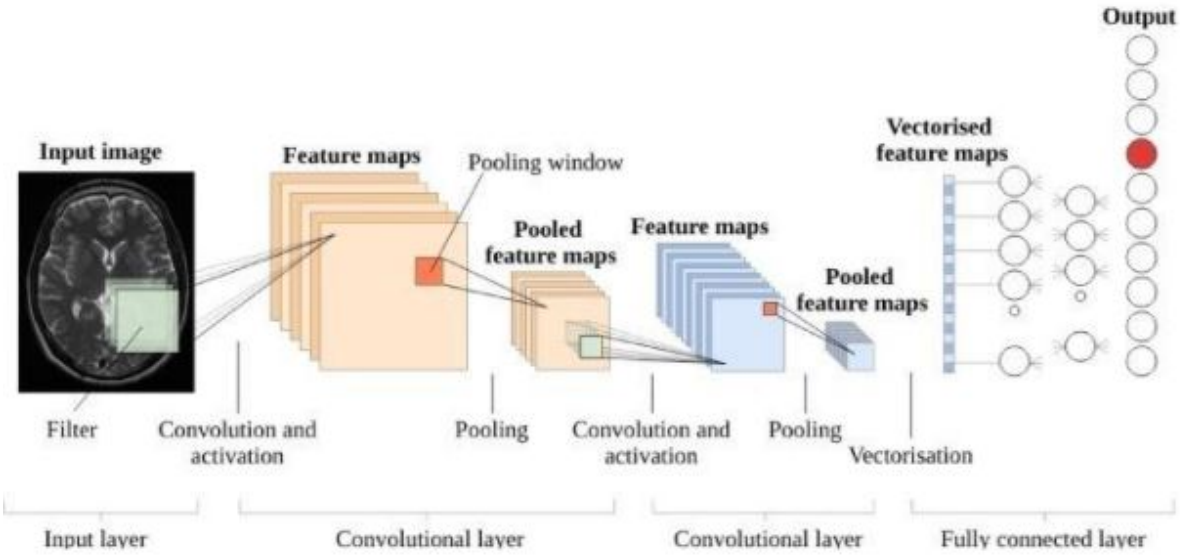

Figure 2.9: Building blocks of a typical CNN [39]

The followings are the different components of CNN:

Kernels (filters)

An image kernel is a small matrix f x f that can be used to apply effects similar to those found in Photoshop or other image editing software, such as blurring, sharpening, outlining, and embossing. Kernels are used in CNN for 'feature extraction', which is a technique for determining the most important features in an image.

Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

Figure 2.10: Kernel examples. [40]

Convolution layer

The role of a convolution layer is to detect local features such as edges, lines, blobs of color, and other visual elements at different positions in the input feature maps by applying convolution operations on learnable kernels. [41] The more kernels that we give to a convolutional layer, the more features it can detect. [21]

Convolution function is the process of flipping both the rows and columns of the kernel and multiplying locally similar entries and summing. As illustrated in the figure below(fig.II.12).

It can be applied by starting the kernel at the far-left top border, which will multiply the elements of this region by the filter weights and summing them. Next, this filter will move to the right advancing by the number of cells specified in the stride which is the size of the steps that the filter moves each time. This process will output the feature map. [21]

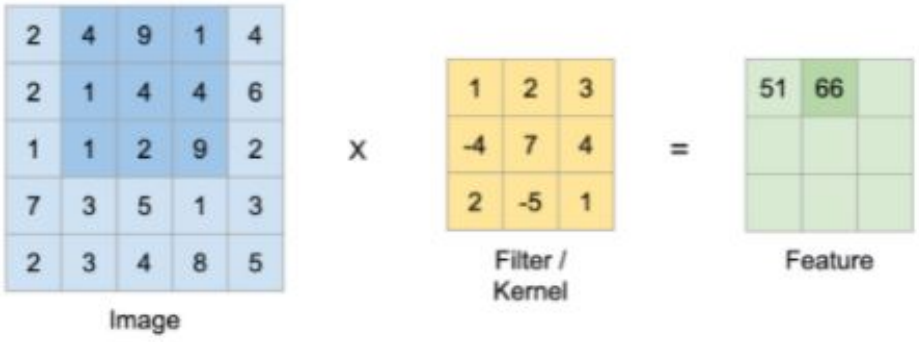

Figure 2.11: Convolution operation using 3x3 filter and stride = 1.

The input and output of a convolutional layer are both 3D boxes. For the input to a convolutional layer, the width and height of the box are equal to the width and height of the input image. The depth of the box is equal to the color depth of the image. For an RGB image, the depth is 3 as you can see in the figure below (2.12). As we said,

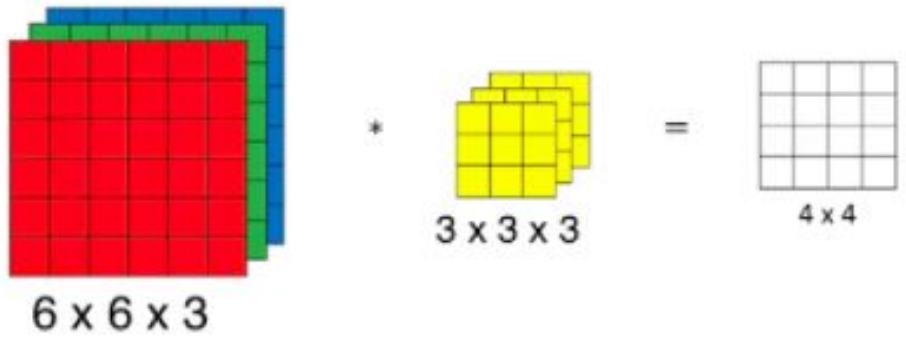

Figure 2.12: RGB color, Depth = 3 with Stride = 1, filter = 3x3.

the output that is fed forward to the next convolution layer will also be a 3D box, so its height and width are both equal to the kernel size but the depth is equal to the number of filters.[21]

Applying convolution operations results in losing pixels from our image, So the size of the feature map is always smaller than the input. The solution for keeping the spatial size constant after performing convolution is the padding technique. By adding extra pixels with zero values around the boundary of our input image the size of our output image will be increased, see the figure below (2.13).

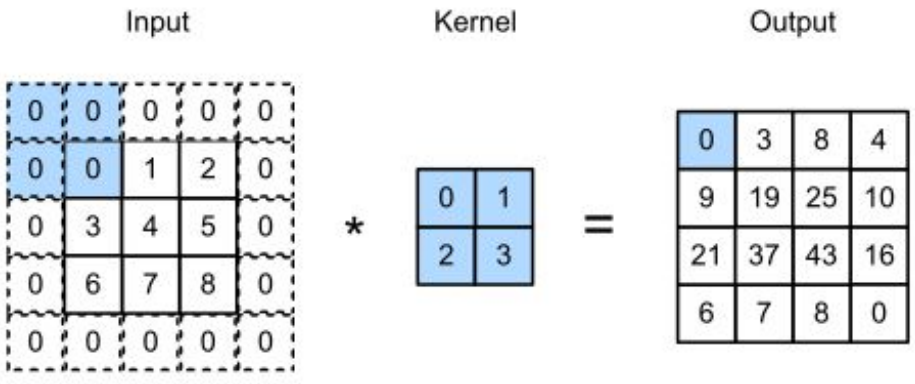

Figure 2.13: The effect of padding.

Pooling layer

The main idea of pooling is down-sampling (reducing the resolution) for the goal of reducing the complexity for further layers. Two of the most common types of pooling methods are Max-pooling and Average pooling. Max-pooling divides the image into sub-region rectangles, and it only returns the maximum value of the inside of that sub-region. As shown in the figure below (2.14). On the other hand, Average pooling takes the average value. It should be considered that down-sampling does not preserve the position of the information.[42]

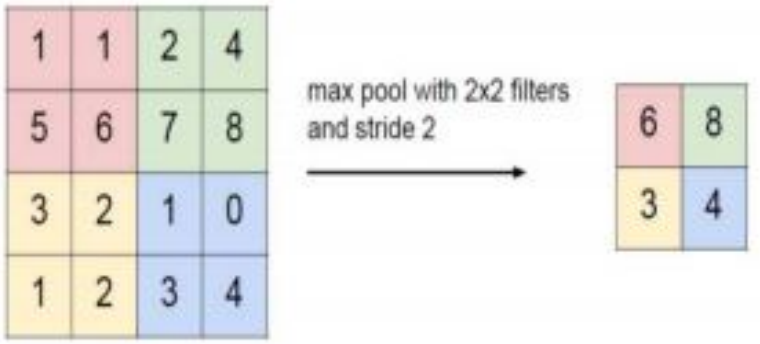

Figure 2.14: Max-pooling with 2x2 filter and stride 2.

Activation layer

The activation layer applies an activation function. An activation function of a node defines the output of that node given an input or set of inputs. *Tanh, ReLU, Leaky, ReLU* and *Softmax* are some of the most common activations for neural networks. In this section, we will briefly introduce them.

Tanh - The hyperbolic tangent function must output values in the range between -1 and 1. [21]

$$\sigma(x) = \tanh(x) \tag{2.1}$$

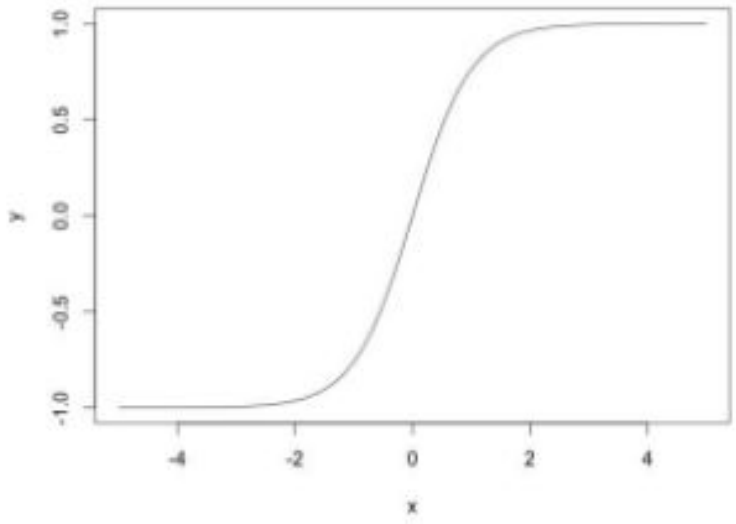

Figure 2.15: The tanh activation function. [21]

ReLU - It is short for the rectified linear unit. It was first introduced in the early 2000s and was later reintroduced in 2010 to great success (Hahnloser et al., 2000; Nair and Hinton, 2010). The rectifier is of the following form:

$$\alpha_{ReLu}(t) = \max(0, t) \tag{2.2}$$

This is not a smooth function. The analytical form of an actual implementation of the rectifier is

$$\alpha'_{ReLu}(t) = \ln(1 + e^t) \tag{2.3}$$

which is a smooth approximation to the rectifier and is often called softplus. The following figure shows the plots of these functions:

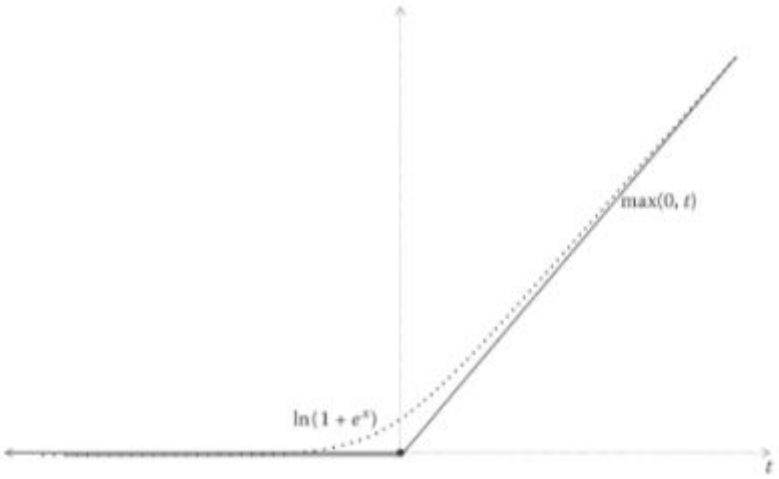

Figure 2.16: The ReLU activation function and its analytical approximation. [43]

Leaky ReLU - Leaky ReLU is an addendum to ReLU. It is of the form:

$$\alpha_{ReLu}(t) = \begin{cases} t, & if \quad t>0 \\ \\ \sigma t, & otherwise \end{cases} \tag{2.4}$$

where the parameter σ is a small constant. This allows the neuron to be very mildly active and produces a small gradient regardless of whether the neuron was meant to be active or not. The parametric leaky ReLU is an extension of this, where σ is considered another parameter of the neuron and is learned alongside the backpropagation of the weights themselves. If $\sigma < 0$, we obtain an interesting activation function shown in the figure below (2.17). In this case, the neuron sends out a negative signal. Even when the neuron is not contributing to the class predictions, it causes the gradients to move much faster.

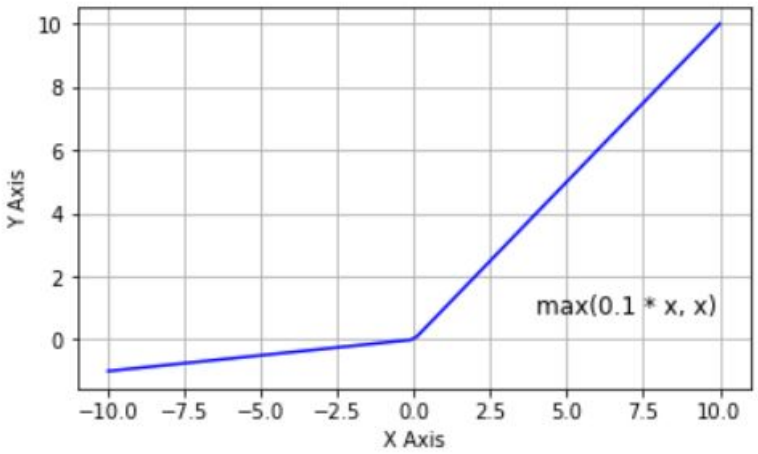

Figure 2.17: Leaky ReLU. [44]

Softmax - The softmax function is usually found in the output layer of a neural network which is used on the classification task. The neuron that has the highest probability claims the input as a member of its class. [21]

$$\sigma_i = \frac{e_i^z}{\sum_{j \in group} e_j^z} \tag{2.5}$$

i represents the index of the output neuron (o) being calculated, j represents the indexes of all neurons in the group/level. The variable z designates the array of output neurons. [21]

Batch Normalization layer

Batch normalization is one of the most powerful and most common techniques of initialization. It is common practice to normalize images before feeding them forward through a network. In a deep network, the input distribution of each layer varies from batch to batch and sample to sample. This is due to the fact that the parameters of the previous layers change with each update. This makes training extremely difficult, especially with saturating activation functions. [43]

Even within the same class, samples differ in their statistical properties across batches of data. This is known as covariate shift. To address the issue of covariate shift, we should normalize the activations coming from each layer. The right variance to normalize with and the mean from which to mean-subtract the data are frequently unknown and can be estimated from the dataset itself. Batch normalization is one such method. If z were the

activations of one layer, we compute

$$\zeta_{bn} = \frac{(\zeta - \mu_z) \times \alpha_z}{\sigma_z} \tag{2.6}$$

where μ_z and σ_z are the activation batch's mean and variance, respectively. α is now one of the network's learnable parameters, and it can be thought of as learning the stretch of the normalization used. α is also learned along with the weights during the same optimization. α can be learned for multiple layers using backpropagation. [43]
Batch normalization is a powerful tool that allows the network to learn much faster, even when the activation functions are non-saturating. Batch normalization is especially popular in visual computing contexts where image data is used.[43]

Transposed convolution layer

Convolution and pooling layers often reduce the dimensions of our input image. Transposed convolutions are usually used in segmentation architectures to generate an output image that has a spatial dimension greater than that of the input, this layer is defined by some learnable parameters, such as padding and stride.
An output image 'B' from a convolutional layer is smaller than its input image 'A', adding stride and padding to the image 'B' will enlarge the image dimensions, so by applying the convolution on the image 'B', we will be able to generate an output image 'C' where its spatial dimensions are equal to that input image 'A'. [45]

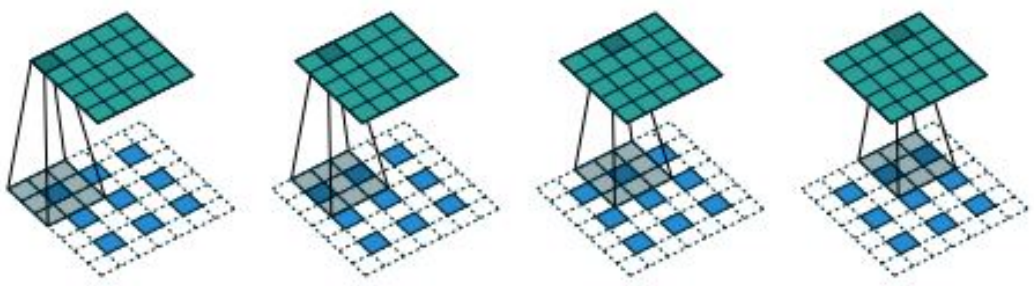

Figure 2.18: Transposed convolution with these parameters: padding=1x1 strides = 2x2 kernel 3x3 [46]

Upsampling layer

Upsampling layers are usually used in segmentation architectures, the commonly used techniques in the upsampling part that we will introduce in this section are *the nearest*

neighbor, Bi-linear interpolation, Bed of nails.

Nearest neighbor Briefly, it is the act of copying an input element and putting it to its nearest output elements. As illustrated in the figure below (see figure 2.19).

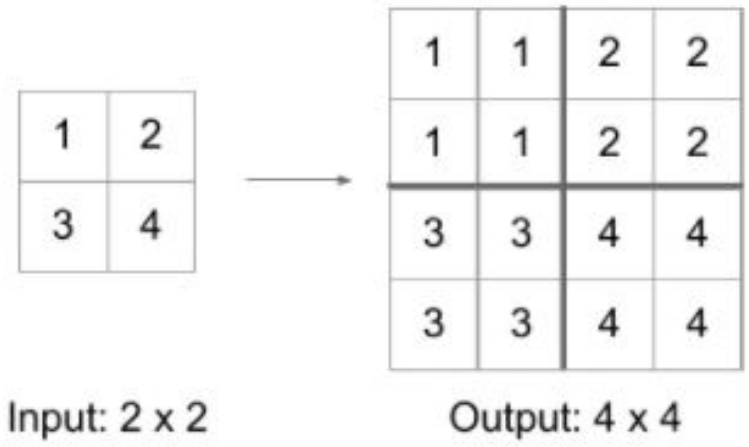

Figure 2.19: Nearest neighbor example.

Bi-linear interpolation it is done by taking an input element and predicting the nearest values for this element. As illustrated in the figure below (see figure 2.20).

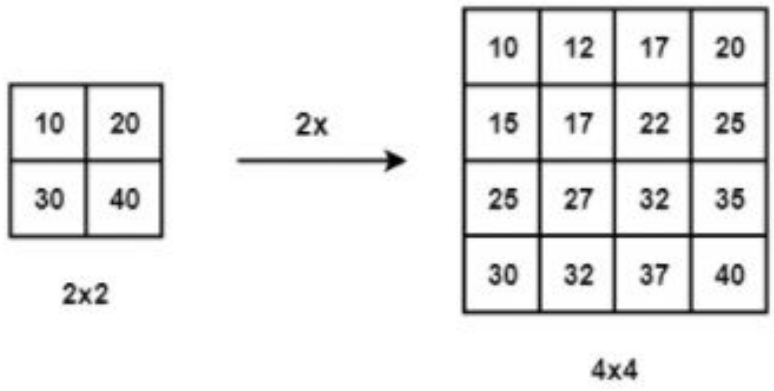

Figure 2.20: Bi-linear interpolation example.

Bed of nails Simply, it is the act of duplicating the value of the input pixel at the corresponding position within the output image and filling the remaining positions by zeros. As illustrated in the figure below (see figure 2.21).

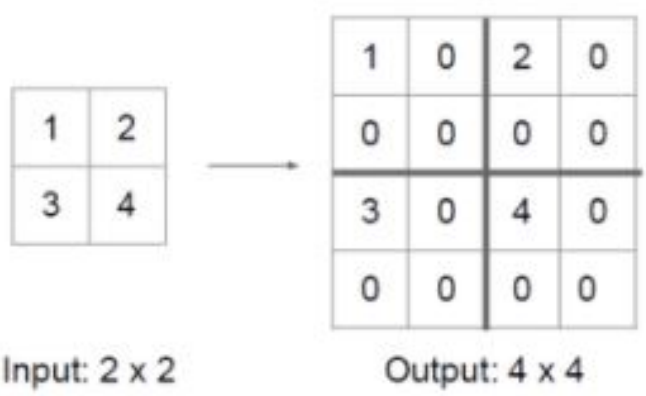

Figure 2.21: Bed of nails example.

2.5 Training

Together, the backpropagation algorithm and optimization algorithms can be used to train neural networks. Optimization requires gradients to be calculated for each variable in the model so that new values for the variables can be calculated. These gradients are calculated by the backpropagation.

2.5.1 Backpropagation

Backpropagation is a common technique used for training feedforward neural networks, where this technique relies on fine-tuning the weights of a neural network based on the error rate obtained in the previous epochs (iterations). Backpropagation is an effective way to calculate the gradients.

2.5.2 Optimization

The goal of optimization algorithms is to minimize the loss value calculated by the loss function. This function is the objective function of an optimization algorithm, where the optimization attempts to reduce the training error. Here is a list of some of the optimization algorithms:

Adam optimizer - an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. [47]

Adagrad optimizer - a subgradient method that dynamically incorporates knowledge of the geometry of the data observed in earlier iterations to perform more informative gradient-based learning. [48]

Stochastic gradient descent (SGD) - an iterative method for optimizing an objective function with suitable smoothness properties. It reduces the computational burden to achieve faster iterations in trade for a lower convergence rate.

Momentum - was invented for reducing high variance in stochastic gradient descent and softens the convergence. It accelerates the convergence towards the relevant direction and reduces the fluctuation to the irrelevant direction.

2.6 Conclusion

In this chapter, we talked about the fundamentals of deep learning. We went from the generalities of learning types in deep learning, to the details of CNN architecture. Even though it has been existing for more than 70 years, deep learning is still in its nascent stage and is still developing with time. Nowadays DL and its techniques and architectures have become an indispensable tool for solving complex problems.

In the next chapter, we will talk about the semantic image segmentation task using deep learning for brain tumors.

Chapter 3

Semantic segmentation of brain tumors using deep learning: Review

3.1 Introduction

Brain tumor segmentation using MRI images is crucial in computer-aided diagnosis. It helps doctors to confirm the size of tumors and evaluate the effects before and after treatment. However, as much as it greatly reduces the workload of doctors, it is a very challenging task.

With the rapid development of DL in computer vision, the semantic segmentation methods for brain tumor MRI images started to see the light and have achieved good results. Moreover, they are in constant improvement.

In this chapter, we will present a review of semantic segmentation methods for brain tumors.

3.2 Image segmentation

Image segmentation is a computer vision task, which refers to the partitioning of an image into several disjoint regions based on features such as grayscale, color, spatial texture and geometric shapes. These features show similarity in the same area, and also show obvious differences between different regions.[49]

In the task of image segmentation, we can find two different board types, semantic segmentation and instance segmentation.

3.2.1 The difference between semantic segmentation and instance segmentation

The semantic image segmentation task consists of classifying each pixel of an image into an instance, where each instance corresponds to a class. This process is similar to the instance segmentation task. If two objects belong to the same category, they will be labeled to the same class; otherwise, for the instance segmentation task, they will be labeled as different classes.
The figure below (3.1) illustrates the difference.

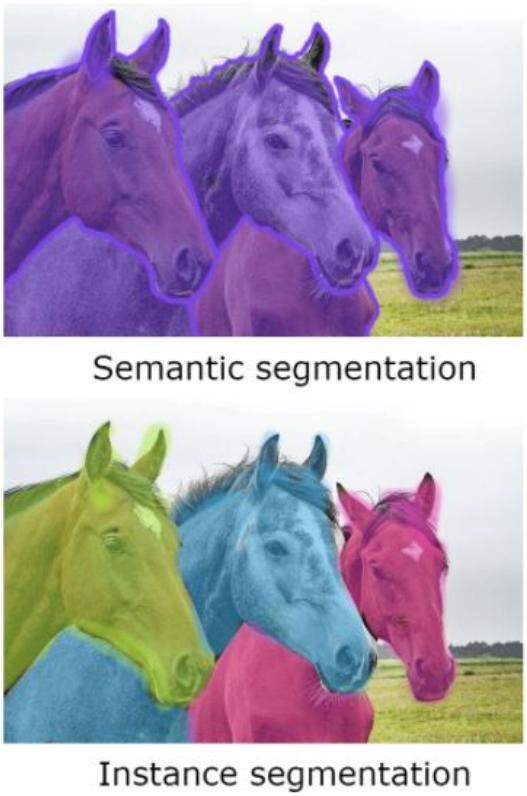

Figure 3.1: The difference between Semantic segmentation and instance segmentation

3.2.2 Medical image segmentation

Medical image segmentation is of great significance in providing non-invasive information about the structure of the human body. This information helps radiologists to visualize

and study the anatomy of the structures, simulate biological processes, localize pathologies, track the progress of diseases, and evaluate radiation treatment or surgery. [50]
Brain tumor segmentation consists of separating the different tumor tissues (advancing or active tumor, edema, and necrotic tumor core) from normal brain tissues. Some tumors, like glioblastomas, are hard to distinguish from normal tissues, because they infiltrate surrounding tissues causing unclear boundaries. As a solution, more than one image modalities with varying contrasts are often employed.
Segmentation of medical images is regarded as a semantic segmentation task. The figure below (3.2) shows brain tumor semantic segmentation.

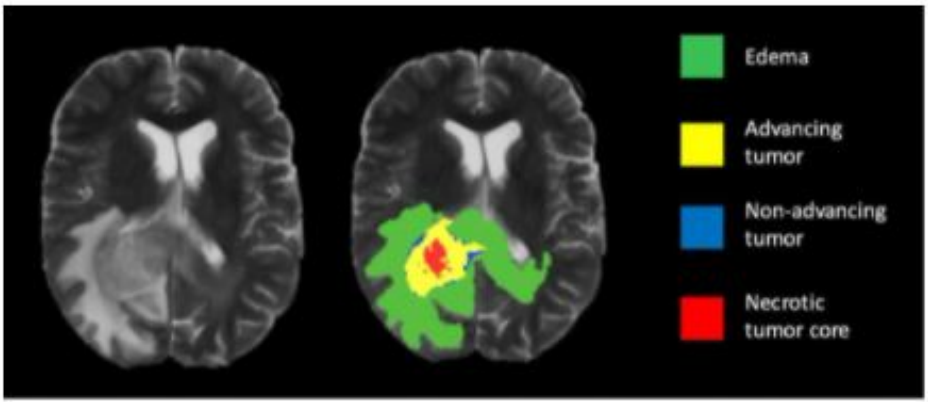

Figure 3.2: Brain tumor semantic segmentation example.[51]

3.2.3 Types of image segmentation

Brain tumor segmentation can be broadly categorized as manual segmentation, semi-automatic segmentation, and fully automatic segmentation, depending on the level of human participation.

Manual segmentation

Through manual segmentation, a human operator uses specialized tools to carefully delineate tumor regions. The accuracy of segmentation results largely depends on the training and experience of the human operator as well as knowledge of brain anatomy. In addition to being tedious and time-consuming, manual segmentation is widely used in approaches based on supervised and semi-supervised learning as a gold standard for semi-automatic and fully automatic segmentation. [52]

Semi-automated segmentation

Semi-automated segmentation combines both computer and human expertise. The initialization of the segmentation process requires user interaction providing feedback and an evaluation of segmentation results [53]. Although semi-automatic segmentation methods are less time consuming than manual segmentation, their results are still dependent on the human operator. [52]

Fully automatic segmentation

In fully automatic brain tumor segmentation, no human interaction is required. Combining artificial intelligence and prior knowledge to solve segmentation problems [53]. Fully automatic segmentation methods are further divided into discriminative and generative methods. Discriminative methods often rely on supervised learning where relationships between the input image and manually annotated data are learnt from a huge dataset. In this group, classical machine learning algorithms, which rely on handcrafted features, have been widely used in the past few years and have achieved great success. However, these methods may not be able to take full advantage of the training data due to the complexity of medical images [54]. More recently, deep learning methods have gained popularity due to their unprecedented performance in computer vision tasks and their ability to learn features directly from data. On the other hand, generative methods use prior knowledge about the appearance and distribution of different tissue types. [52]

3.3 Dataset and evaluation metrics

3.3.1 BraTS dataset

BraTS dataset focuses on the evaluation of state-of-the-art methods for the segmentation of brain tumors in multimodal magnetic resonance imaging (MRI) scans. BraTS utilizes multi-institutional pre-operative MRI scans, which mainly focuses on the segmentation of internal heterogeneity (appearance, shape, and histology) brain tumors (i.e. gliomas). [55]

This dataset is a brain tumor segmentation competition dataset that is used in the MICCAI challenge [55]. This challenge has been held every year since 2012, with the goal of

evaluating the best brain tumor segmentation methods and compare them. The Brats dataset has five labels: healthy brain tissue, necrotic area, edema area, tumor enhancement and non-enhancement area. New training sets are added every year. [49]

3.3.2 Evaluation Metrics

The following metrics are the most commonly used for evaluation.

Dice score (F-measure)

It's the comparison of the ground truth GT (mask created by specialists) and the output image OI (mask created by the model), which can be defined as twice the overlap area of predicted and ground-truth maps, divided by the total number of pixels in both images. [56]

$$\frac{2 * (GT \cap OI)}{(GT \cup OI)} \tag{3.1}$$

Hausdorff

It describes a measure of the degree of similarity between two sets of points, that is, the distance between the two boundaries of ground truth and the segmentation result input to the network. Sensitive to the divided boundary. [49]

$$d(\mathcal{S}eg, \mathcal{G}T) = \left\{ \left(\max_{i \in \mathcal{S}eg} \min_{j \in \mathcal{G}T} d(i,j) \right), \left(\max_{j \in \mathcal{G}T} \min_{i \in \mathcal{S}eg} d(i,j) \right) \right\}. \tag{3.2}$$

where i and j are points belonging to different sets. d represents the distance between i and j.

Specificity

Measures the probability that the model classifies a patient as being normal given that they are normal. [56]

$$\frac{TN}{FP + TN} \tag{3.3}$$

TN: True negative, FP: False positive.

Sensitivity

This indicates the probability that the model classifies a patient as having the disease given that they have the disease. [56]

$$\frac{TP}{TP + FN} \tag{3.4}$$

TP: True positive, FN: False negative.

3.4 Review of semantic segmentation methods for brain tumors

After the success of the classification models in the medical field and brain tumor classification task, such as Inception [57], VGG [58] and AlexNET [59], The researchers have passed to the idea of segmentation medical images in order to help in reducing the time spent in the manual segmentation and help doctors to confirm the size of tumors, quantitatively evaluate the effect before and after treatment and greatly reducing the workload of doctors [49]
In this section we are going to present some of the used architectures for brain tumor segmentation:

3.4.1 Sliding Windows

It is the process of sliding a rectangular region of fixed size across an image. For each region captured by this sliding window, the algorithm will try to determine if the window has an object or region of interest or not. The disadvantages of this technique are that it is very computationally expensive and it does not share features between overlapping patches (local regions).

Yanyun Lian and Zhijian Song [60] proposed an Automated brain tumor segmentation in magnetic resonance imaging based on sliding-window technique and symmetry analysis which was evaluated on 3D FSPGR (fast spoiled gradient echo) brain MR images of 10 patients and the average ratio of correct location was 93.4% for 575 slices containing tumor, the average Dice similarity coefficient was 0.77 for one scan, and the average time spent on one scan was 40 seconds.

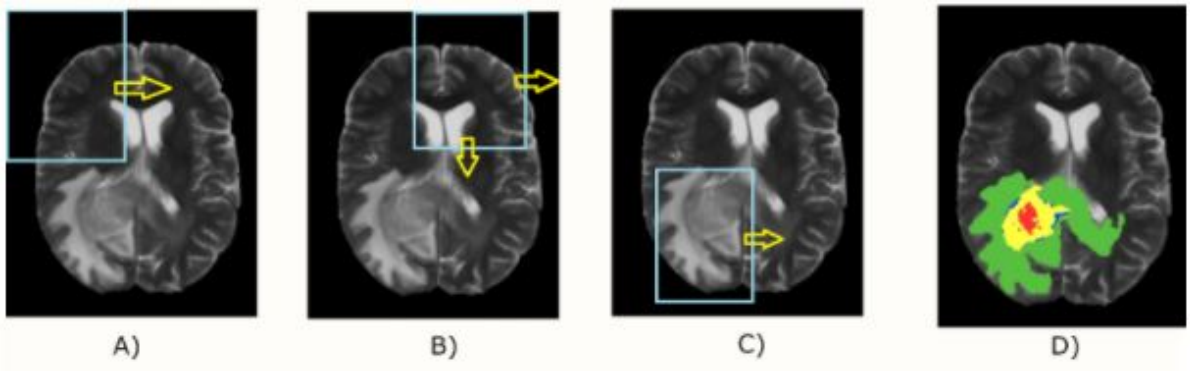

Figure 3.3: Semantic segmentation using a sliding window.

3.4.2 CNNs

The Encoder-Decoder architecture is mainly composed of two parts, in which the encoder captures deep semantic information through several down-sampling processes, which means it extracts image features from the input image and compacts the features by encoding to produce the low-resolution feature map; the decoder part gradually restores the space and detail information from the low-resolution feature maps fed by the encoder through several up-sampling operations.

The U-Net is a popular model in the architecture of encoder/decoder, this semantic segmentation model was proposed by Ronneberger et al in 2015 [61] to handle the lack of training images in biomedical images.

The architecture contains a downsampling path (encoder) and upsampling path (decoder). The encoder is just a traditional stack of convolutional and max-pooling layers which consist in extracting features. The decoder uses transposed convolutions to increase or recover the input image size and enable precise localization for the regions of our image.it does not contain any Dense layer because of which it can accept images of any size.

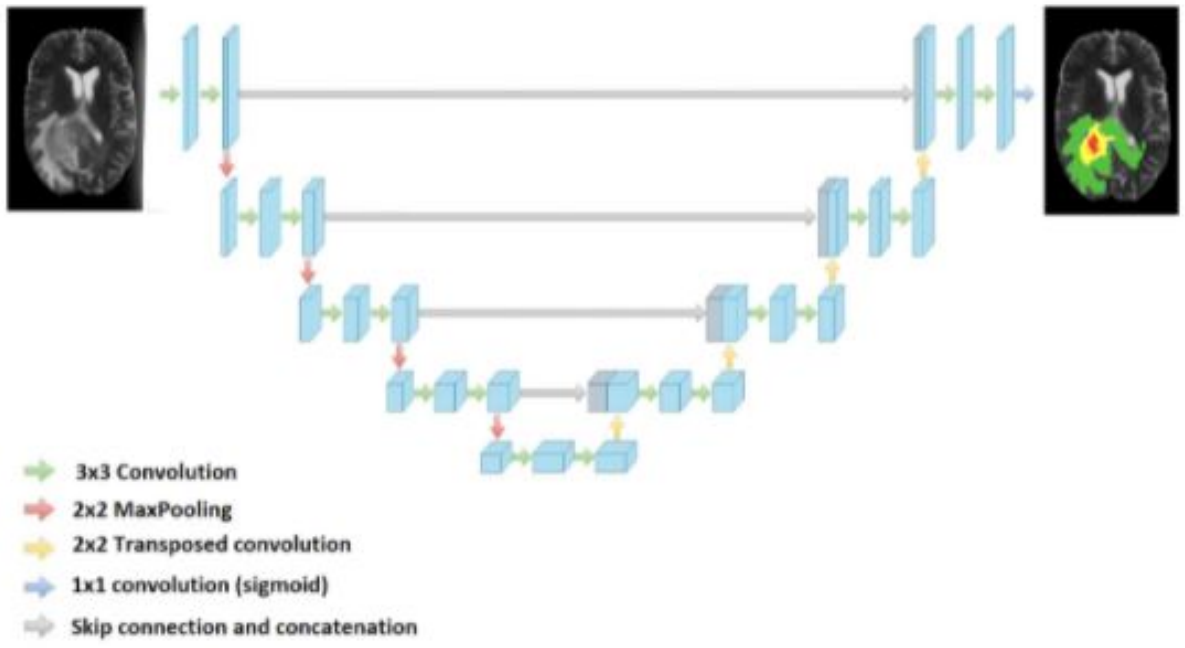

Figure 3.4: U-NET basic architecture.

Borne et al. [62] selected 62 healthy brain images from different heterogeneous databases as the training set and segmented them using 3D U-Net. The result was 85% correct. The use of three-dimensional information in segmentation makes full use of the advantages of spatial information.

3D CNN makes the U-Net structure have richer spatial information, in other words, it can extract more powerful features volume representation on the three axes of X, Y, and Z, the network realizes 3D image segmentation by inputting a continuous 2D slice sequence of 3D images.

Guotai Wang, Wenqi Li, Sébastien Ourselin, and Tom Vercauteren proposed an Automatic Brain Tumor Segmentation using Cascaded Anisotropic Convolutional Neural Networks, The cascaded CNNs separate the complex problem of multiple class segmentation into three simpler binary segmentation problems and take advantage of the hierarchical structure of tumor subregions to reduce false positives. Three networks are proposed to hierarchically segment whole tumor (WNet), tumor core (TNet) and enhancing tumor core (ENet) sequentially.
The figure below (3.5) illustrates this strategy.

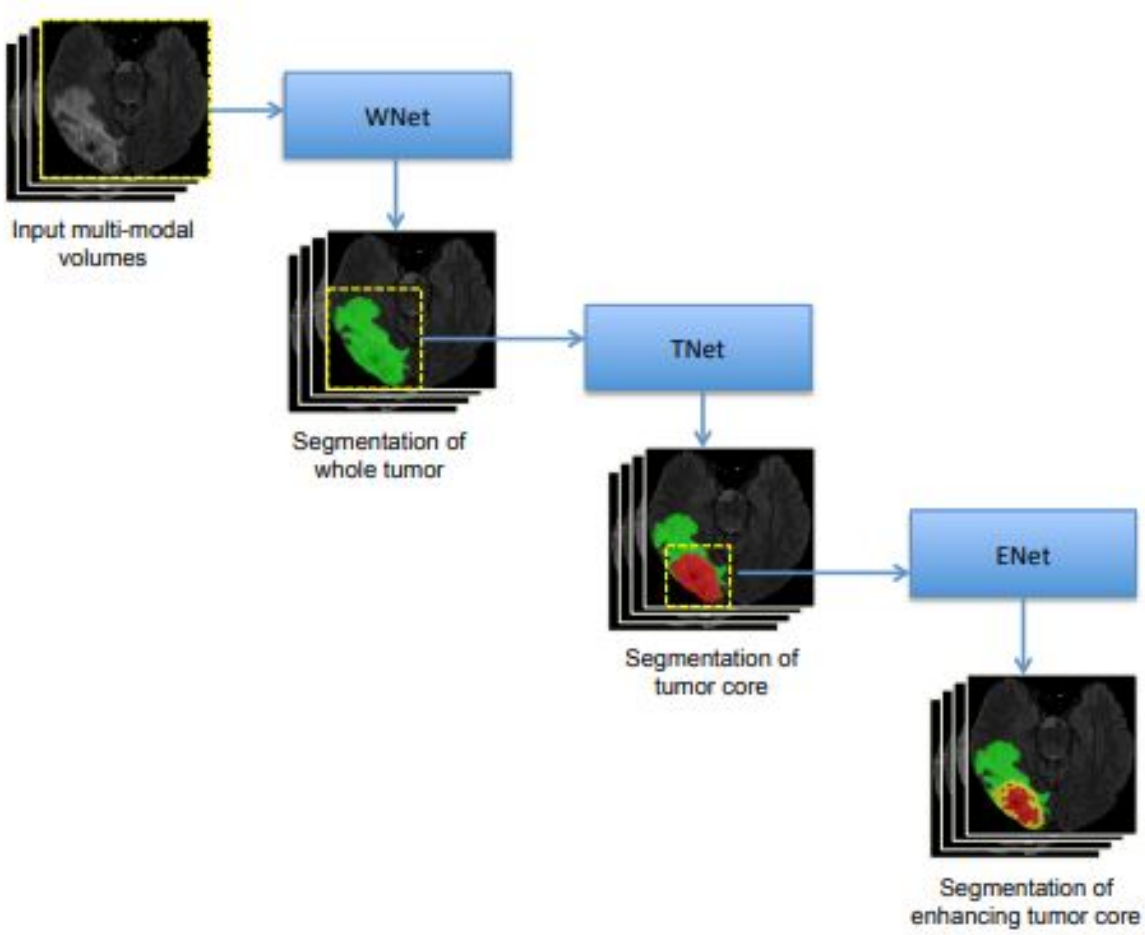

Figure 3.5: The proposed triple cascaded framework for brain tumor segmentation [63]

Casamitjana et al. [64] proposed the cascaded V-Net segmentation of brain tumor, dividing the brain tumor segmentation problem into two simpler tasks, the segmentation of the entire tumor and the division of different tumor regions.

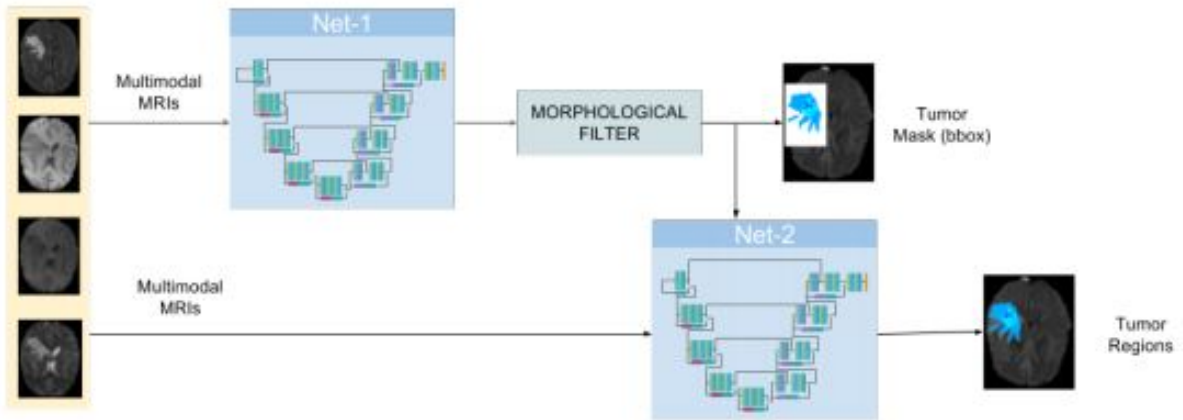

Figure 3.6: Cascaded V-Net segmentation of brain tumor. [64]

The Dice score metric for the whole tumor (WT) region was 0.869 for the training and 0.877 for the validation, but low values for enhancing tumor (ET) with 0.671 for the training and 0.714 for the validation and for tumor core (TC) regions was 0.685 for the training and 0.637 for the validation on the Brats2017 dataset.

Fully convolutional network

A fully convolutional network (FCN) was the first article that applied deep learning to image segmentation and achieved outstanding results, but still not fine enough. Many models of image segmentation have been borrowed from FCN. We will go through some of these models that helped in the progress of this task and had a great performance.

Myronenko et al. [65] proposed a deep learning network 3D MRI brain tumor segmentation that won the first place in the 2018 challenge, this network was based on asymmetric FCN and combined with residual learning. Solving complex problems requires deep neural networks to learn more complex features, which increase performance and accuracy. but there is a maximum threshold for depth with traditional CNN, in other words, going deeper in the network can result in increasing both training and test error.

ResNet [66] solves this problem by using shortcuts, which is composed of many residual blocks. Each module consists of many consecutive layers and a shortcut. This shortcut connects the input and output of the module together, adding them before ReLU (rectified linear unit) activation. The resulting output is then sent to the ReLU activation function to generate the output of this block.
The figure below (3.7) illustrates a residual learning block.

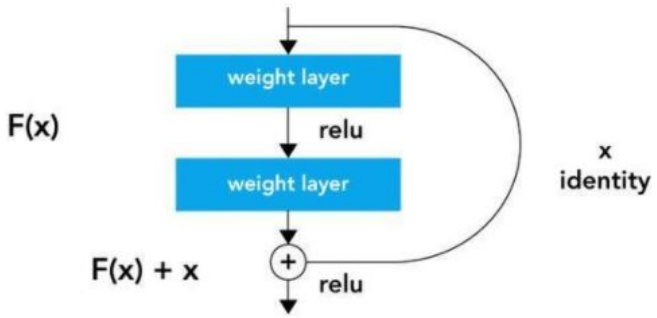

Figure 3.7: Residual learning: a building block.[66]

Kamnitsas et al. [67] combined three different network architectures, namely 3D FCN [68], 3D U-Net [61], and DeepMedic [69] and trained them with different loss functions and different normalization strategies.

Wang et al. [63] employed a FCN architecture enhanced by dilated convolutions [70] and residual connections [66].

Nie et al. [71] implemented network optimization by integrating contextual semantic information and fusing features of different scales, and segmented multimodal brain MRI images using 3D FCN.

Wang et al. [72] proposed a Conditional Random Fields (CRF) based edge-sensing FCN, which achieved more accurate edge segmentation by adding edge information into the loss function. The accuracy of the model was up to 87.31%, far higher than that of FCN-8S and other basic semantic segmentation networks.

Jiang et al. [73] proposed a two-stage cascaded U-Net to segment the brain tumor sub-regions from coarse to fine, where the second-stage model has more channel numbers and uses two decoders so as to boost performance.

Isensee et al. [74] used their recently proposed nnU-NET (no new net) framework for the configuration of segmentation methods. the downsampling is done with strided convolutions, upsampling is implemented as convolution transposed. they replaced the softmax non-linearity with a sigmoid, also replaced the crossentropy loss term with a binary cross-entropy that optimizes each of the tumor class independently. the proposed network uses stochastic gradient descent with an initial learning rate of 0.01 and a Nesterov momentum of 0.99.

R-CNN

R-CNN or Regions with CNN Features is an object detection model that uses high-capacity CNNs to bottom-up region proposals in order to localize and segment objects. It uses selective search to identify several bounding-box object region candidates ("regions of interest") and then extracts features from each region independently for classification.

In other words, R-CNN consists of applying different rectangular shapes to an image, and trying to determine the regions of interest within these rectangular shapes.

The problem with this approach is that the objects of interest might have a different spatial location inside the image, which will lead us to use more regions that could end with expensive computations and time-consuming. This approach has developed in the last few years and went to fast R-CNN, Faster R-CNN and YOLO (You only look once).

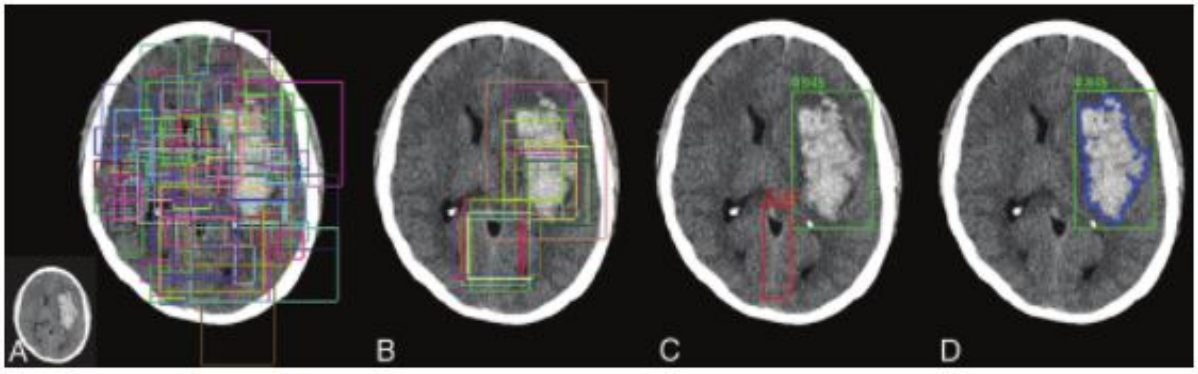

Figure 3.8: Semantic segmentation of brain tumor using R-CNN.[75]

3.4.3 Generative Adversarial Networks

here are a lot of segmentation models using GAN. For instance :

Xue et al. [76] proposed the SegAN model for medical image segmentation, which covers the disadvantage of the U-Net model, which is the problem of unbalanced pixel categories in the image. Based on this problem, the authors designed a new segmentation network based on the ideas of GAN, and proposed a multiscale L1 loss to optimize the segmentation network.

Moeskops et al. [77] used adversarial training to improve the segmentation performance of brain MRI in fully convolutional and a network structure with dilated convolutions.

Rezaei et al. [78] used conditional-GAN to train a semantic segmentation convolutional neural network, which has a superior ability for brain tumor segmentation.

Focusing on the segmentation task of MRI brain tumors, Giacomello et al. [79] proposed SegAN-CAT, a deep learning architecture based on a generative adversarial network. They apply a trained model to different modalities through transfer learning.

SegAN-CAT is different from SegAN in that the loss function is extended, a dice loss term is added. The input of the discriminator network is composed of MRI image stitching and segmentation. By training several brain tumor segmentation models on the BRATS 2015 and BRATS 2019 datasets for testing, SegAN-CAT has better performance than SegAN. The results of these networks are illustrated in the table below (see table 3.1):

	BraTS 2015			BraTS 2019		
Model	Dice Score	Precision	Sensitivity	Dice Score	Precision	Sensitivity
SegAN	0.705	0.759	0.694	0.766	0.745	0.834
SegAN with dice loss	0.825	0.901	0.785	0.814	0.850	0.810
SegAN-CAT	0.859	0.882	0.852	0.825	0.842	0.835

Table 3.1: Comparaison of SegAN, SegAN with dice loss and SegAN-CAT [79]

3.5 Results of recent models for brain tumor segmentation task

The following tables (3.2 and 3.3) shows the results of the winners for BraTS datasets from 2017 to 2020.

Rank	Reference	Architecture	Dice Score			Sensitivity			Specificity			Hausdorff		
			ET	WT	TC	ET	WT	TC	ET	WT	TC	ET	WT	TC
BraTS 2017														
1	[67]	Ensemble	0.738	0.901	0.797	0.783	0.895	0.762	0.998	0.995	0.998	4.499	4.229	6.562
2	[80]	Cascaded	0.786	0.905	0.838	0.771	0.915	0.822	0.999	0.995	0.998	3.282	3.890	6.479
3	[81]	Unet	0.776	0.903	0.819	0.803	0.902	0.786	0.998	0.996	0.999	3.163	6.767	8.642
3	[82]	SegNet	0.706	0.857	0.716	0.687	0.811	0.660	0.99	0.997	0.999	6.853	5.872	10.925
BraTS 2018														
1	[65]	Ensemble	0.825	0.912	0.870	0.845	0.923	0.864	0.998	0.995	0.998	3.997	4.537	6.761
2	[83]	Unet	0.809	0.913	0.863	0.831	0.919	0.844	0.998	0.995	0.999	2.413	4.268	6.518
3	[84]	Ensemble	0.792	0.901	0.847	0.829	0.911	0.836	0.998	0.994	0.998	3.603	4.063	4.988
3	[85]	Ensemble	0.814	0.909	0.865	0.813	0.914	0.868	0.998	0.995	0.997	2.716	4.127	6.545
BraTS 2019														
1	[73]	Two-stage Unet	0.802	0.909	0.865	0.804	0.924	0.862	0.998	0.994	0.997	3.146	4.264	5.439
2	[86]	Unet	0.746	0.904	0.840	0.780	0.901	0.811	0.990	0.987	0.990	27.403	7.485	9.029
3	[87]	Ensemble	0.634	0.790	0.661	0.604	0.727	0.587	0.983	0.980	0.983	47.059	14.256	26.504

Table 3.2: A summary of top-performing methods on BraTS 2017, 2018, and 2019 validation data as reported by the online evaluation platform. ET—Enhancing tumor, WT—Whole tumor, and TC—Tumor core.[52]

Rank	Reference	Architecture	Dice Score			Sensitivity			Specificity			Hausdorff		
			ET	WT	TC	ET	WT	TC	ET	WT	TC	ET	WT	TC
BraTS 2020														
1	[74]	nnU-Net	0.820	0.889	0.850	–	–	–	–	–	–	–	–	–
2	[88]	H 2NF-Net	0.787	0.912	0.854	–	–	–	–	–	–	26.575	4.184	4.971
2	[89]	MPL	0.816	0 .891	0 .842	–	–	–	–	–	–	–	–	–
3	[90]	AutoBTS SAN	0.817	0 .882	0 .843	–	–	–	–	–	–	13.429	5.2176	17.969

Table 3.3: A summary of top-performing methods on BraTS 2020 validation data as reported by the online evaluation platform. ET—Enhancing tumor, WT—Whole tumor, and TC—Tumor core.

3.6 Challenges and discussion

Semantic segmentation of brain tumor MRI images has made great progress, but these recent works still cannot satisfy the needs of practical applications [49]. So, in this section, we will introduce the common challenging problems for this task.

There is a difference between MRI images of brain tumors from hospital to hospital, because of the different hardware resources that are used in this task, also the tumors can vary from a patient to another. This will affect the performance of the models during segmentation. and the noises are also a problem that needs to be handled in the data preprocessing step.

Data preprocessing is a very important step in preparing raw input data to be more amenable to neural networks. because of the artifacts and noises that MRI images contain, the model will not perform well. So, these artifacts need to be corrected before the images are fed into the network for better performance.

The training of deep learning algorithms requires a large amount of data set support, but unfortunately, in the medical field, the existing medical image data sets are small in scale because medical data is protected by data-protection laws that restrict the usage and sharing of this kind of data to other parties [52], which leads to the problem of overfitting in the training process of deep learning models. We can prevent overfitting by simplifying our model complexity by removing some layers or reducing the number of neurons. Also, we can apply the technique of data augmentation, which will produce more images to train our model. Also, by applying the Dropout technique [91], which randomly drops neurons from the neural network during training in each iteration. Another technique to address this limitation is Transfer learning [92], The idea behind it is that you can use a pretrained model that has been trained on large datasets as good start point which has already learned some useful features like lines, edges, curves ... etc and partially retrain the model on a smaller dataset.

The performance of the segmentation task is affected by the class imbalance problem, where there is an unequal distribution of voxel classes in the training dataset. For example,

in brain tumor segmentation, the quantity of healthy voxels is bigger than unhealthy voxels. Which leads the model to be more biased towards the majority class.

Many works use loss-based methods to solve the problem of class imbalance. Lin et al. [93] proposed a loss function to solve the problem by dynamically scaling the loss based on the model's confidence in the classification of samples. When the accuracy of the model classification category increases, the scaling factor decreases. Therefore, the model pays more attention to misclassified samples.[52]

In conclusion, to improve the accuracy and robustness of our segmentation model we have to deal with the class imbalance problem, selection of pre-processing techniques and employing advanced training schemes.

3.7 Conclusion

In this chapter, we reviewed some deep learning-based approaches for semantic segmentation of brain tumors.

The performance of brain tumor segmentation algorithms has continued to increase over the past few years due to the availability of more training data, the use of more sophisticated CNN architectures and training schemes and the recently proposed techniques that can solve some of the challenges mentioned above.

In the next chapter, we will present our contribution for semantic segmentation of brain tumors from MR images.

Chapter 4

Our approaches

4.1 Introduction

This chapter is divided into two parts, in the first part we will talk about everything that was used in the process of development like programming languages, development environment ,the different used libraries and the preprocessing of the data. In the second part, we will explain the approaches that we proposed.

4.2 Development Environment and used materials

In order to create and train our models we used the following configuration:

4.2.1 Programming language

Python


is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major

platforms, and can be freely distributed.[1]

4.2.2 Environment

Training a deep CNN requires powerful computing resources, thus for convenience we have used Google colaboratory. Colaboratory, or "Colab" for short, is a product from Google Research. It allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service (an open-source web application that allows creating and sharing documents that contain live code, equations, visualizations and narrative text) that requires no setup to use, while providing free access to computing resources including GPUs. Colab provides 12GB of RAM and Tesla K80 NVIDIA GPU.[2]

4.2.3 Frameworks and libraries

TensorFlow

TensorFlow is an end-to-end platform that was originally developed by the Google Brain team, written with a Python API over a C/C++ engine for numerical computation using data flow graphs. Multiple APIs have been provided. The complete programming control is provided with the lowest level APIs, called TensorFlow Core. Machine learning researchers and others who need fine levels of control over their models are recommended to use the TensorFlow Core. The higher-level APIs are built on top of TensorFlow Core and they are easier to learn and use, compared to the TensorFlow Core. TensorFlow supports multiple backends, CPU or GPU on desktop, server or mobile platforms. It has well-supported bindings to Python and C++. TensorFlow also has tools to support reinforcement learning. For more details, we suggest the readers to visit the TensorFlow website.[3]

[1] `https://www.python.org/`

[2] `https://colab.research.google.com/`

[3] `https://www.tensorflow.org/`

Keras

Keras is a Python-based open-source library of neural network components. Keras can run on top of TensorFlow, Theano, PlaidML, and other frameworks. Although the library was designed to be modular and user-friendly, it began as part of a research project for the Open-ended Neuro-Electronic Intelligent Operating System, or ONEIROS. Francois Chollet, a Google engineer who also wrote XCeption [94], a deep neural network model, is the primary author of Keras. After Keras was released, it was not integrated into Google's TensorFlow core library until 2017.[4]

Numpy

NumPy (Numerical Python) is a free and open source Python library that is used in nearly every branch of science and engineering. It is the universal standard in Python for working with numerical data, and it is at the heart of the scientific Python and PyData ecosystems.[5]

Pandas

Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool,built on top of the Python programming language.[6]

Nibabel

Nibabel is a library that is used to provide read/write access to some common neuroimaging file formats including GIFTI, NIfTI1, NIfTI2, CIFTI-2. The various image format classes give full or selective access to header (meta) information and access to the image data is made available via NumPy arrays.[7]

H5py

The h5py package is a Pythonic interface to the HDF5 binary data format. It allows storing huge amounts of numerical data, and easily manipulating that

[4] `https://keras.io/`
[5] `https://numpy.org/`
[6] `https://pandas.pydata.org/`
[7] `https://github.com/nipy/nibabel`

data from NumPy. It uses straightforward NumPy and Python metaphors, like dictionary and NumPy array syntax.[8]

Segmentation_models


It is a Python library with Neural Networks for Image Segmentation based on Keras and TensorFlow. It provides multiple models like Unet, with lots of backbones (ResNet, VGG ...) trained on huge data like ImageNet dataset.[95]

Matplotlib


Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.[9]

4.3 Dataset description

In subsection (3.1) of the previous chapter, we defined the BraTS dataset. This part is dedicated to give more details about two versions of this dataset: BraTS2020(The most recent available version) and (BraTS2017).

BraTS2020 contains 369 3D scans for training and 125 for validation. Each training sample is composed of 5 files each with the following shape: height=240, width= 240 and 155 slices.

The first four files are the different multimodal scans (Four channels of information- 4 different volumes of the same region) that come as follows:

T1 T1-weighted, native image, sagittal or axial 2D acquisitions, with 1–6 mm slice thickness.

T1ce T1-weighted, contrast-enhanced (Gadolinium) image, with 3D acquisition and 1 mm isotropic voxel size for most patients.

T2 T2-weighted image, axial 2D acquisition, with 2–6 mm slice thickness.

Flair T2-weighted FLAIR image, axial, coronal, or sagittal 2D acquisitions, 2–6 mm slice thickness.

[8] `https://www.h5py.org/`
[9] `https://matplotlib.org/`

The fifth file is simply the mask (ground truth). The Ground truth images have been segmented manually, by one to four raters, following the same annotation protocol, and their annotations were approved by experienced neuro-radiologists, Where the sub-regions of tumor considered for evaluation are:

- 0 for 'Not tumor'.
- 1 for 'Necrotic or non-enhancing tumor'.
- 2 for 'edema'.
- 3 for 'missing' (no pixels in all the volumes contain label 3).
- 4 for 'enhancing tumor'.

The image below 4.1 shows an example of the different types and the corresponding mask:

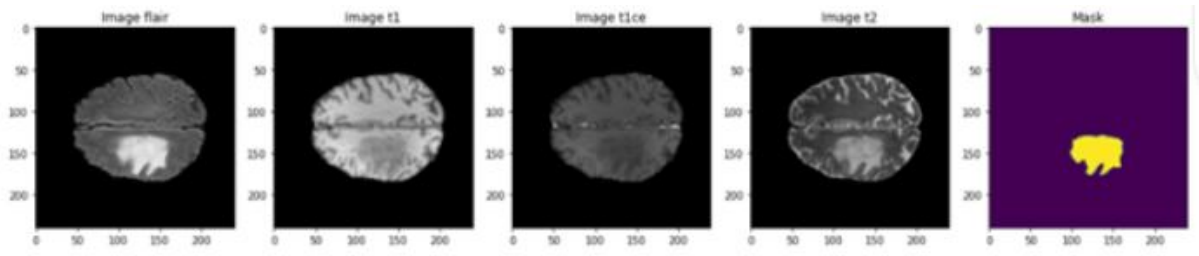

Figure 4.1: demonstration of the BraTS20 dataset.

The validation samples don't contain mask files, so we are not going to use them.
MRI images often come in the DICOM [10] format which is the output format for most commercial MRI scanners.
Our dataset is stored in the NifTI-1 format, for that we will be using the NiBabel library to interact with the files. This data has been mostly pre-processed for the competition participants, so we are not going to need to perform skull striping or image registration.

BraTS2017 is not very different from BraTS2020, the key differences are:
Number of samples: BraTS2017 has 484 samples for training and 266 for validation.
The multimodel scans come in a single file, a 4D array of MR image in the shape of (240, 240, 155, 4).

[10] `https://en.wikipedia.org/w/index.php?title=DICOM`

- "0": "FLAIR",
- "1": "T1w",
- "2": "t1gd",
- "3": "T2w"

The labeling order in the ground truth file is also different and has no missing indexes

- "0": "background",
- "1": "edema",
- "2": "non-enhancing tumor",
- "3": "enhancing tumour"

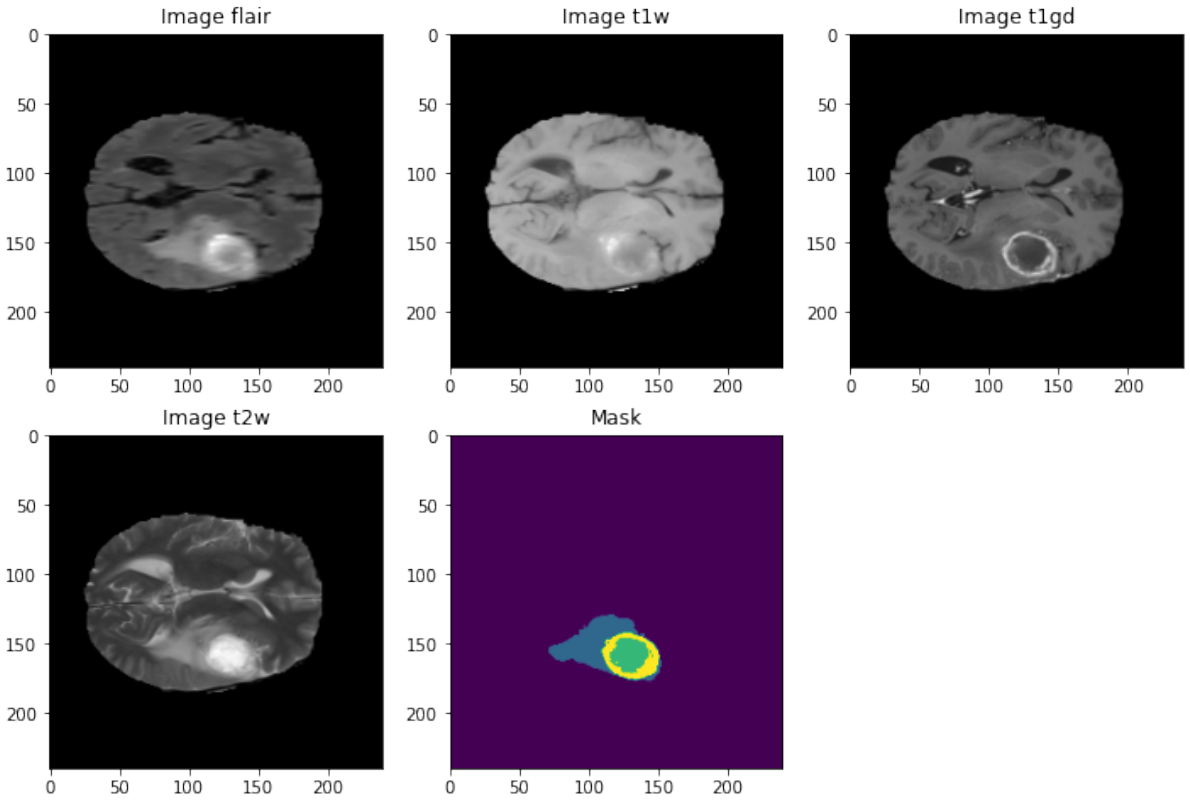

Figure 4.2: BraTS2017 example.

4.4 Data preprocessing

While our dataset is provided to us post-registration and has been mostly pre-processed for the competition participants, we still have to do some minor pre-processing before feeding the data to our model because of the recources limitations.

4.4.1 Preparing the data for 2D UNET

After combining the four types of images (t1, t2, t1ce and flair) into a single multi-channel volume, reassigning pixels of value 4 to value 3 in masks(as 3 is missing from original labels) and converting the mask into **One Hot** format by using **to_categorical** function from Keras utils, we were unable to load our data because 12 GB of RAM could not handle it so we had to resize the data. The first 21 and the last 34 slices of each image we ignored because most of them were black and did not contain any useful information. We fed our model slices of shape [160, 160, 4] but the RAM problem still remained. To tackle this problem we used a data generator, which basically load the data in batches instead of loading it all at once. (see figure 4.3)

Datagen part1

```python
import keras.utils.dataset_creator
IMG_SIZE=160
class DataGenerator(tf.keras.utils.Sequence):
    'Generates data for Keras'
    def __init__(self, list_IDs, dim=(IMG_SIZE,IMG_SIZE), batch_size = 1, n_channels = 4, shuffle=True):
        'Initialization'
        self.dim = dim
        self.batch_size = batch_size
        self.list_IDs = list_IDs
        self.n_channels = n_channels
        self.shuffle = shuffle
        self.on_epoch_end()

    def __len__(self):
        'Denotes the number of batches per epoch'
        return int(np.floor(len(self.list_IDs) / self.batch_size))

    def __getitem__(self, index):
        'Generate one batch of data'
        # Generate indexes of the batch
        indexes = self.indexes[index*self.batch_size:(index+1)*self.batch_size]

        # Find list of IDs
        Batch_ids = [self.list_IDs[k] for k in indexes]

        # Generate data
        X, y = self.__data_generation(Batch_ids)

        return X, y

    def on_epoch_end(self):
        'Updates indexes after each epoch'
        self.indexes = np.arange(len(self.list_IDs))
        if self.shuffle == True:
            np.random.shuffle(self.indexes)
```

Figure 4.3: Data generator for 2D UNET part1.

Datagen part2

```python
def __data_generation(self, Batch_ids):
    'Generates data containing batch_size samples' # X : (n_samples, *dim, n_channels)
    # Initialization
    X = np.zeros((self.batch_size*VOLUME_SLICES, *self.dim, self.n_channels))
    y = np.zeros((self.batch_size*VOLUME_SLICES, 240, 240))
    Y = np.zeros((self.batch_size*VOLUME_SLICES, *self.dim, 4))


     # Generate data
     for c, i in enumerate(Batch_ids):
        case_path = os.path.join(TRAIN_DATASET_PATH, i)

        data_path = os.path.join(case_path, f'{i}_flair.nii.gz');
        flair = nib.load(data_path).get_fdata()

        data_path = os.path.join(case_path, f'{i}_t1ce.nii.gz');
        ce = nib.load(data_path).get_fdata()

        data_path = os.path.join(case_path, f'{i}_t1.nii.gz');
        t1 = nib.load(data_path).get_fdata()

        data_path = os.path.join(case_path, f'{i}_t2.nii.gz');
        t2 = nib.load(data_path).get_fdata()

        data_path = os.path.join(case_path, f'{i}_seg.nii.gz');
        seg = nib.load(data_path).get_fdata()

        for j in range(VOLUME_SLICES):
              X[j +VOLUME_SLICES*c,:,:,0] = cv2.resize(flair[:,:,j+VOLUME_START_AT], (IMG_SIZE, IMG_SIZE));
              X[j +VOLUME_SLICES*c,:,:,1] = cv2.resize(ce[:,:,j+VOLUME_START_AT], (IMG_SIZE, IMG_SIZE));
              X[j +VOLUME_SLICES*c,:,:,2] = cv2.resize(t1[:,:,j+VOLUME_START_AT], (IMG_SIZE, IMG_SIZE));
              X[j +VOLUME_SLICES*c,:,:,3] = cv2.resize(t2[:,:,j+VOLUME_START_AT], (IMG_SIZE, IMG_SIZE));

              y[j +VOLUME_SLICES*c] = seg[:,:,j+VOLUME_START_AT];

        # Generate masks
    y[y==4] = 3
    mask = tf.keras.utils.to_categorical(y, num_classes=4, dtype="float32")
    Y = tf.image.resize(mask, (IMG_SIZE, IMG_SIZE));
    return X/np.max(X), Y
```

Figure 4.4: Data generator for 2D UNET part2.

Note: We used the same process for The 2D UNET with transfer learning approach except for the number of channels, we had to use only 3 of them because the backbones that we chose were trained on imagenet with 3 channels, And in order to start the training with their weights the number of channels had to be 3.

4.4.2 Preparing the data for 3D UNET

As similar to 2D approach, the four types of scans (t1, t1ce, t2 and flair) were combined into one single array and the mask labels were reassigned and then converted into One

Hot format.

combine four images

```
def load_case(img1,img2,img3,img4):

    image1 = np.array(nib.load(img1).get_fdata())
    image2 = np.array(nib.load(img2).get_fdata())
    image3 = np.array(nib.load(img3).get_fdata())
    image4 = np.array(nib.load(img4).get_fdata())
    image = np.array([fname for fname in [image1,image2,image3,image4]])

    return image
```

Figure 4.5: Load images.

Using this data as it is with our environment was impossible, because a network that can process the entire volume at once will simply not fit inside our current environment's memory/GPU. which pushed us to generate patches of our data which can be thought of as sub-volumes of the whole MR images. We used the same size for X and Y dimensions (160), and for Z we tried many sizes until we got the minimum which was 16, So the final size shape was [160, 160, 16, 4]. We randomly generated 30 patches from each image. Furthermore, given that a large portion of the MRI volumes are just brain tissue or black background without any tumors, we wanted to make sure that we pick patches that at least include some amount of tumor data. Therefore, we picked patches that have at least 5% tumor.

Lastly, given that the values in MR images cover a very wide range, we standardized the values to have a mean of zero and standard deviation of 1, which is a common technique in deep image processing since standardization makes it much easier for the network to learn.

One slight change for the masks was that we removed the background class (0). So the corresponding masks were of shape [3, 160, 160, 16] The used function for generating the sub-volumes is in the following figure (see figure 4.6)

Patches generator

```python
def get_sub_volume(image, label,
                   orig_x = 240, orig_y = 240, orig_z = 155,
                   output_x = 160, output_y = 160, output_z = 16,
                   num_classes = 4, max_tries = 200,
                   background_threshold=0.95):
    X = y = None  # Initialize features and labels with `None`
    tries = 0
    while tries < max_tries:
        # randomly sample sub-volume by sampling the corner voxel
        start_x = np.random.randint(0, orig_x - output_x+1)
        start_y = np.random.randint(0, orig_y - output_y+1)
        start_z = np.random.randint(0, orig_z - output_z+1)
        label[label==4]=3 # extract relevant area of label after changing label 4 to 3
        y = label[start_x: start_x + output_x,
                  start_y: start_y + output_y,
                  start_z: start_z + output_z]
        # One-hot encode the categories. (This adds a 4th dimension, 'num_classes')
        y = tf.keras.utils.to_categorical(y, num_classes=num_classes, dtype="float32")
        bgrd_ratio = np.sum(y[:, :, :, 0])/(output_x * output_y * output_z) # compute the background ratio
        tries += 1 # increment tries counter
        # if background ratio is below the desired threshold, use that sub-volume.
        # otherwise continue the loop and try another random sub-volume
        if bgrd_ratio < background_threshold:

            # make copy of the sub-volume
            X = np.copy(image[:,start_x: start_x + output_x,
                              start_y: start_y + output_y,
                              start_z: start_z + output_z])
            y = np.moveaxis(y, 3, 0) # change dimension of y
            # take a subset of y that excludes the background class in the 'num_classes' dimension
            y=y[1:,:,:,:]
            return X, y
    # if we've tried max_tries number of samples return an empty array
    empty_array = np.array([])
    return empty_array,empty_array
```

Figure 4.6: Patches generator.

The figure 4.7 represents an example of Patches:

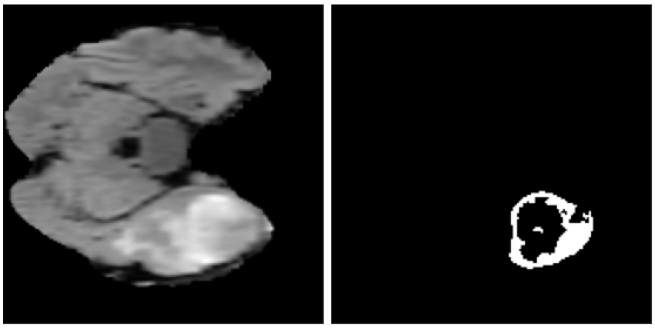

Figure 4.7: Patches Sample.

Finally, We saved the patches data in HDF5 files in Google Drive so it can be loaded during training by our custom data generator (see figure 4.8).

Data_generator

```python
class VolumeDataGenerator(tf.keras.utils.Sequence):
    def __init__(self, sample_list, base_dir,batch_size=1, shuffle=True, dim=(160, 160, 16),
                 num_channels=4, num_classes=3, verbose=1):
        self.batch_size = batch_size
        self.shuffle = shuffle
        self.base_dir = base_dir
        self.dim = dim
        self.num_channels = num_channels
        self.num_classes = num_classes
        self.verbose = verbose
        self.sample_list = sample_list
        self.on_epoch_end()
    def on_epoch_end(self):
        'Updates indexes after each epoch'
        self.indexes = np.arange(len(self.sample_list))
        if self.shuffle == True:
            np.random.shuffle(self.indexes)
    def __len__(self):
        'Denotes the number of batches per epoch'
        return int(np.floor(len(self.sample_list) / self.batch_size))
    def __data_generation(self, list_IDs_temp):
        'Generates data containing batch_size samples'
        # Initialization
        X = np.zeros((self.batch_size, self.num_channels, *self.dim),
                     dtype=np.float64)
        y = np.zeros((self.batch_size, self.num_classes, *self.dim),
                     dtype=np.float64)
        # Generate data
        for i, ID in enumerate(list_IDs_temp):
            if self.verbose == 1:# Store sample
                print("Training on: %s" % self.base_dir + ID)
            with h5py.File(self.base_dir + ID, 'r') as f:
                X[i] = np.array(f.get("x"))
                y[i] = np.array(f.get("y"))
        return X, y

    def __getitem__(self, index):
        'Generate one batch of data'
        # Generate indexes of the batch
        indexes = self.indexes[ index * self.batch_size: (index + 1) * self.batch_size]
        sample_list_temp = [self.sample_list[k] for k in indexes]# Find list of IDs
        X, y = self.__data_generation(sample_list_temp) # Generate data
        return X, y

```

Figure 4.8: Datagenerator for 3D UNET.

4.5 Proposed approaches

In our work we used different approaches: basic 2D and 3D models and other models with transfer learning.

Our systems will go through different steps, from training and validation to tests and evaluating. but before that, our system will pass through an essential step which is Data preprocessing of our dataset. The following image illustrates the global architecture of our systems (see figure 4.9).

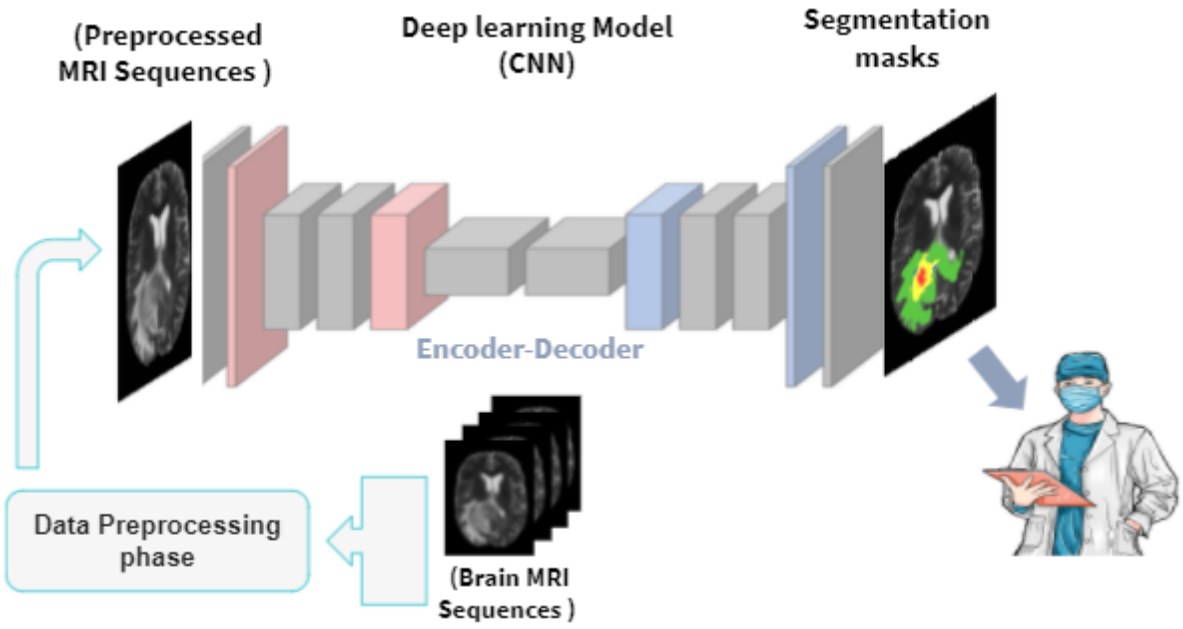

Figure 4.9: System global architecture.

4.5.1 2D approaches

2D Unet

For the 2D approach we used a UNET model developed by Olaf Ronneberger et al [61]. The original network was built for 512x512x3 microscopy images, here it is modified to take in images of shape of 160x160x4. We also chose to start with 32 filters instead of 64. This model will be used as a baseline network (for comparison).

The network architecture is illustrated in (figure 4.10). It consists of a contracting path (left side) and an expansive path (right side). The contracting path follows the typical architecture of a convolutional network. It consists of the repeated application of two 3x3 convolutions (unpadded convolutions), each followed by a rectified linear unit (ReLU) and

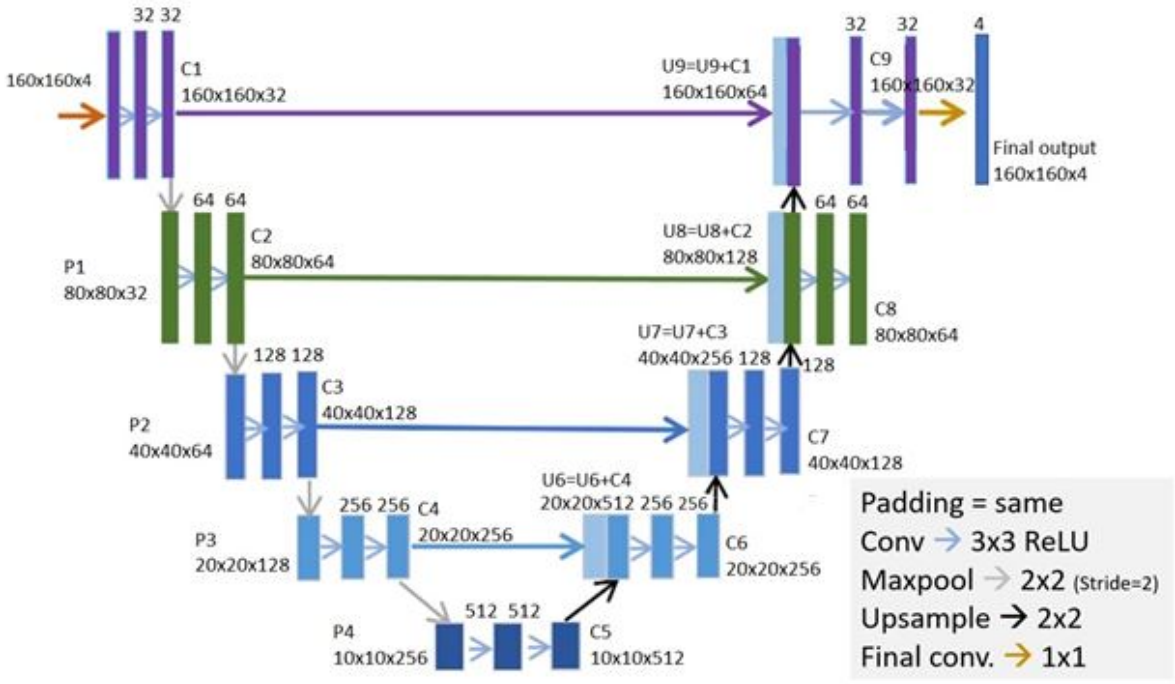

Figure 4.10: 2D UNET architecture.

a 2x2 max pooling operation with stride 2 for downsampling. At each downsampling step we double the number of feature channels. Every step in the expansive path consists of an upsampling of the feature map followed by a 2x2 convolution ("up-convolution") that halves the number of feature channels, a concatenation with the correspondingly feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU. At the final layer a 1x1 convolution is used to map each 32 component feature vector to the desired number of classes. In total, the network has 23 convolutional layers.

2D Unet with different backbones

Transfer learning is the reuse of a model's knowledge that has been trained on a big dataset as the starting point for solving new tasks.

Advantages of Transfer learning

Transfer learning is a technique that addresses some of the challenges mentioned in the previous chapter. Using a small datasets and not having enough labeled data is one of the most common challenges that could be solved using transfer learning.
Using a pre-trained model that has already trained on large datasets such as ImageNet instead of using a CNN model from scratch will give us several benefits, but the main

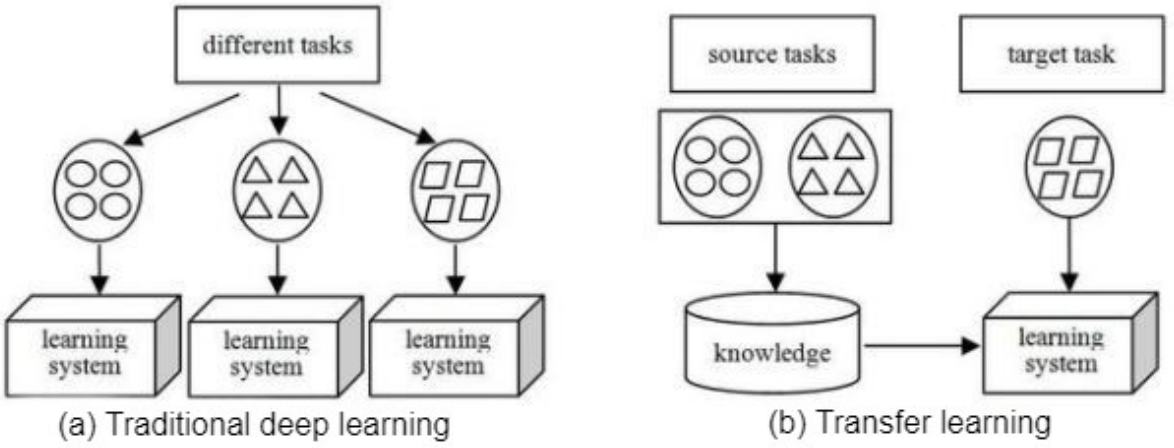

Figure 4.11: Traditional DL vs Transfer learning. [96]

ones are saving training time and getting better performances.

Applying Transfer learning

In a pre-trained model, the first layers learn very useful and a lot of low-level features such as edges, lines, curves ... which we will benefit from in our training.

We will discuss two commonly used methods in applying transfer learning, pre-training and fine-tuning. The pre-training method is the process of freezing all the earlier layers to not retrain their weights and taking off some of the last layers to replace them with one or several new output layers. The weights of these new output layers are initialized randomly and retrained with the dataset that concerns the new task. It is recommended when we have a smaller dataset for the new task. For example, the BraTS dataset that we will use in our implementation.

The Fine-tuning method is the process of using The weights of the pre-trained model as the starting point and retrain all of them, and it could be the same process for the new output layers in the pre-training method. It is recommended when we have a large dataset for the new task.

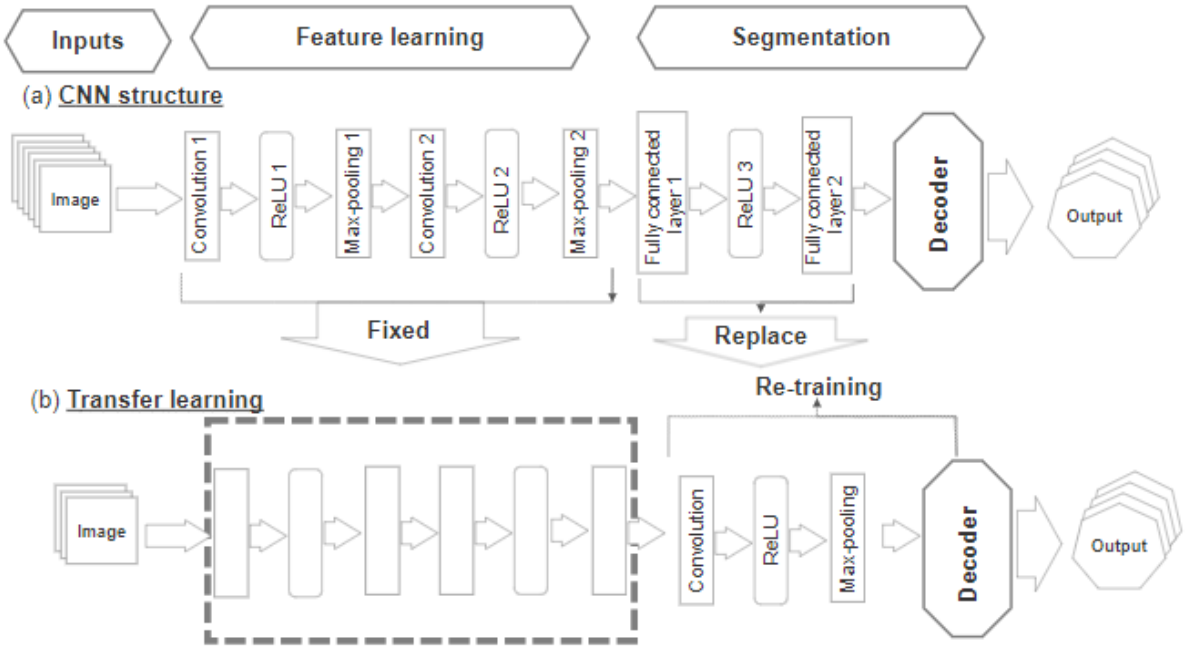

Figure 4.12: How to apply transfer learning. [96]

We are going to replace the encoder of the Unet model with three different networks that were trained on Imagenet. ImageNet is a dataset of over 15 million labeled high-resolution images belonging to roughly 22,000 categories. The images were collected from the web and labeled by human labelers using Amazon's Mechanical Turk crowd-sourcing tool. it contains 1.2 million training images, 50,000 validation images, and 150,000 testing images. [97]

It is very common to use transfer learning in medical images classification(diagnosis) to reduce the effects of overfitting (caused by the small size of medical datasets), which inspired us to explore the possibility of applying this method to the segmentation of medical images, in order to see if it is possible to use the extracted features from a dataset of natural images (Ex. ImageNet) to segment brain tumors.

In this approach, We are going to try three different models 2DUnet_VGG16 (VGG16 as backbone), 2DUnet_Resnet50 (Resnet50 as backbone), 2DUnet_InceptionV3 (Inception V3 as backbone)

Here are more details about each backbone:

VGG16

The VGG16 is a Convolutional Neural Network model that was proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" in the year 2014. [58] It was one of the famous models that won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) Competition that year.

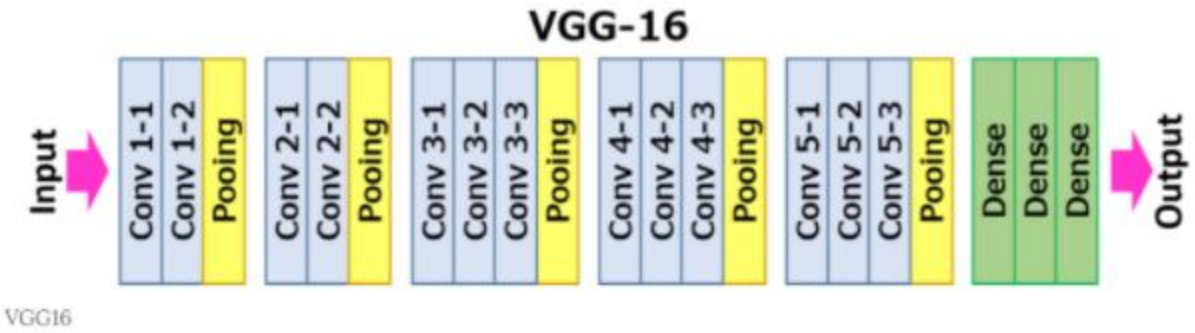

Figure 4.13: VGG-16 illustrated.

The input to the conv1 layer is of fixed size 224 x 224 RGB image. The image is passed through a stack of convolutional (conv.) layers, where the Convolution filters used are only 3x3 and the convolution stride is fixed to 1 pixel.
Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling) . The padding is 1 pixel for 3 x 3 conv. layers.. Max-pooling is performed over a 2 x 2 pixel window with stride 2.
A stack of convolutional layers (which has a different depth in different architectures) is followed by three Fully-Connected (FC) layers: the first two have 4096 channels each, the third performs 1000 way ILSVRC (ImageNet Large Scale Visual Recognition Challenge) classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks. [58]

The architecture of the 2D UNET after using VGG-16 as a backbone will be as follows:

ResNet50

ResNet 50 is a Convolutional Neural Network that was introduced by Microsoft in 2015 [98]. As illustrated in (fig) ResNet50 is composed of an initial 7x7 convolution and a 3x3

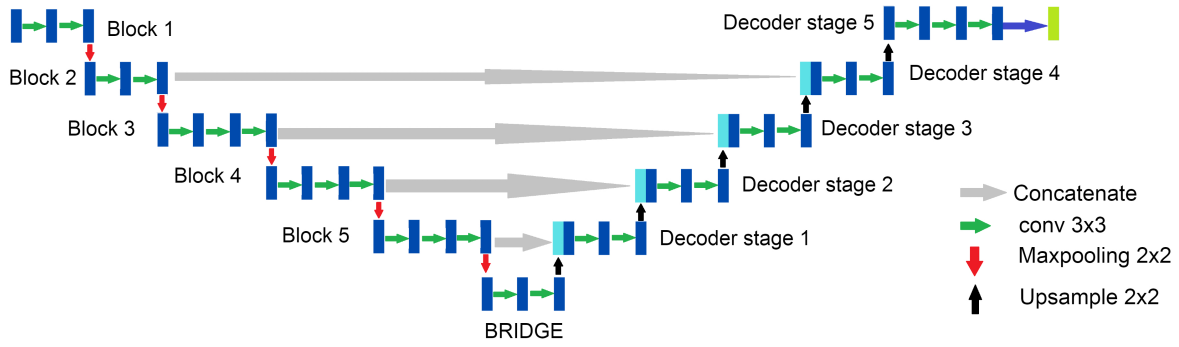

Figure 4.14: 2D UNET with VGG-16 as a backbone.

max-pooling followed by 4 stages. Each stage has a certain number of residual units, stage 1 and stage 4 have 3 units while stage 2 and stage 3 have 4 units and 6 units respectively. Each unit has 3 layers, the first and the third layers are 1x1 convolutions while the second layer is 3x3, each convolution is followed by BN a ReLu activation. The kernel size used to perform the convolutions in each unit are (64,64,256) in stage 1 (128,128,512) in stage 2, (256,256,1024) in stage 3 and (512,512,2048) in stage 4. From one stage to another, the kernel size is doubled and the shape of the image is halved. The 4 stages are followed by an average pooling then a fully connected layer.

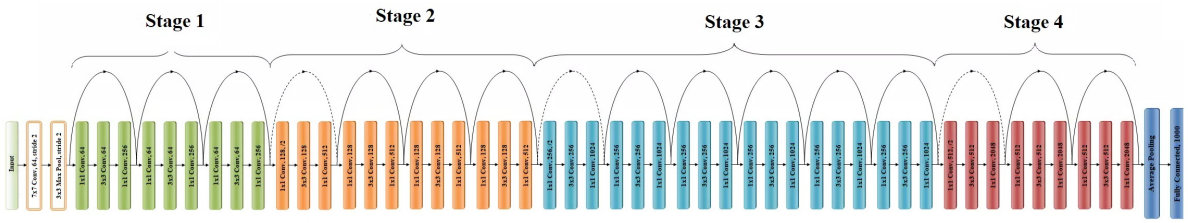

Figure 4.15: ResNet-50 model architecture. [99]

InceptionV3

Inceptionv3 was introduced in 2015 by Google [100] It was a scale up for the 2014 version of inception [57] This network suggested some solutions to improve accuracy and speed without using many layers in depth. As shown in (fig), the inception v3 model is composed by 3 main modules.

- Module A aims for reducing the number of parameters by using smaller convolution layers.

- Module B aims for reducing the complexity of the network by dividing each convolution layer of k x k size to 2 layers of 1 x k dimensions each.
- Module C expands the filters to evade information loss.

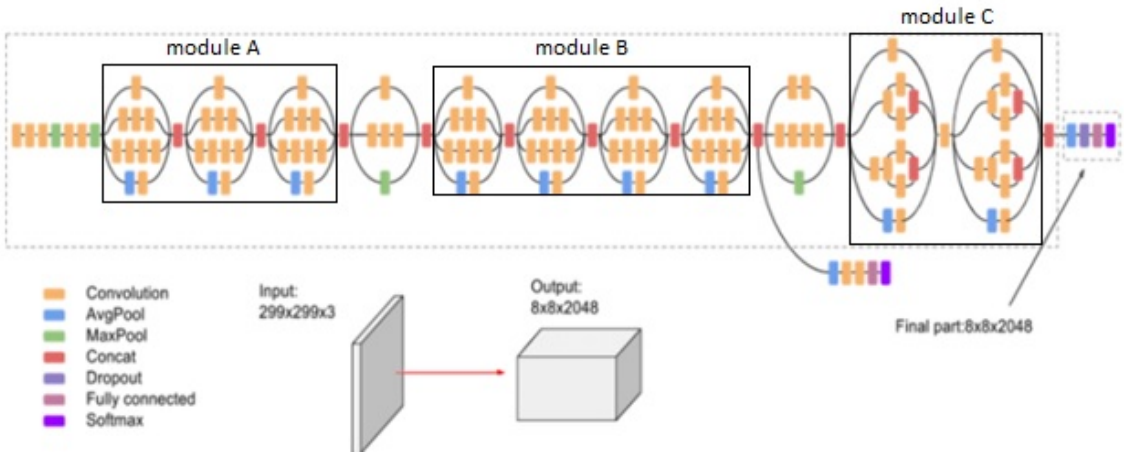

Figure 4.16: InceptionV3 model architecture. [101]

4.5.2 3D approach

We want to try a 3D approach to see the advantages of 3D filters, Also to quantify the impact of information loss when switching from a 3D image to a 2D image. to implement this approach we got inspiration from [102]

The network architecture is illustrated in (figure 4.17). Like the 2D u-net, it consists of a contracting path and an expansive path. In the contracting path, each layer contains two $3 \times 3 \times 3$ convolutions each followed by a rectified linear unit (ReLu), and then a $2 \times 2 \times 2$ max pooling with strides of two in each dimension. In the expansive path, each layer consists of an upconvolution of $2 \times 2 \times 2$ by strides of two in each dimension, followed by two $3 \times 3 \times 3$ convolutions each followed by a ReLu. Shortcut connections from layers of equal resolution in the contracting path provide the essential high-resolution features to the synthesis path. In the last layer a $1 \times 1 \times 1$ convolution reduces the number of output channels to the number of labels which is 4 in our case.

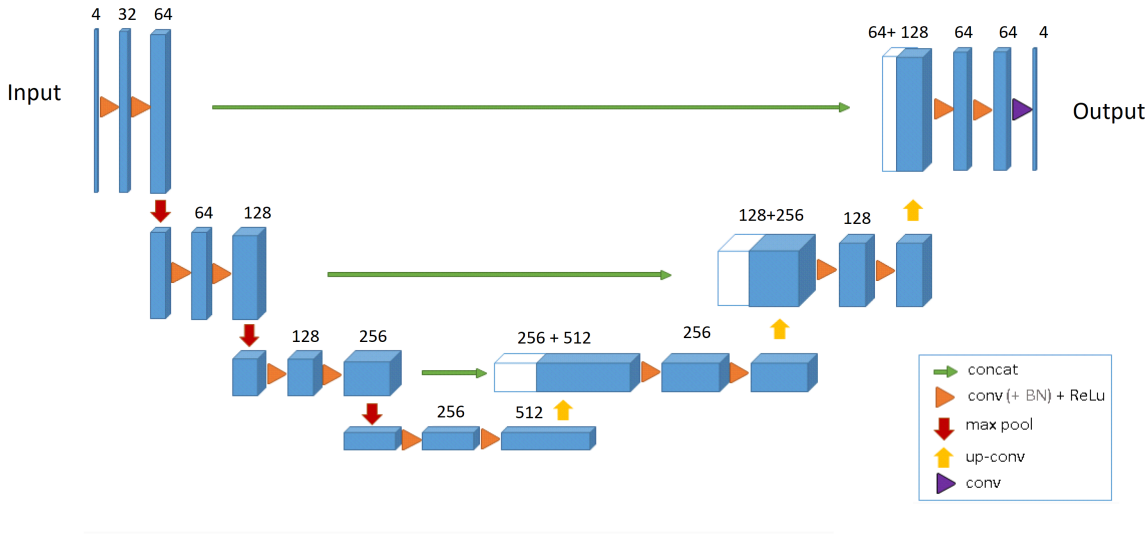

Figure 4.17: 3D UNET architecture.

4.6 Training

4.6.1 2D approaches

For all 2D approaches we used the followings:

- Aside from the architecture, one of the most important elements of any deep learning method is the choice of our loss function. As our task is a multiclass segmentation, a natural choice is the categorical crossentropy loss function. It calculates the loss of an example by computing the following sum:

$$Loss = -\sum_{i=1}^{outputsize} y_i * \log y'_i \tag{4.1}$$

  where y' is the i-th scalar value in the model output, y_i is the corresponding target value, and $outputsize$ is the number of scalar values in the model output.
  This loss is a very good measure of how distinguishable two discrete probability distributions are from each other. In this context, y_i is the probability that event i occurs and the sum of all y_i is 1, meaning that exactly one event may occur.
  The minus sign ensures that the loss gets smaller when the distributions get closer to each other.

- For the optimizer, we used Adam optimizer(introduced in chapter 2 subsection 5.2)

- For metrics, we used Dice Similarity Coefficient (introduced as Dice Score in chapter 3 subsection 3.2)

dice_coef

```
def dice_coefficient(y_true, y_pred, axis=(0, 1, 2),
                     epsilon=0.00001):
    dice_numerator = 2. * K.sum(y_true * y_pred, axis=axis) + epsilon
    dice_denominator = K.sum(y_true, axis=axis) + K.sum(y_pred, axis=axis) + epsilon
    dice_coefficient = K.mean((dice_numerator)/(dice_denominator))
    return dice_coefficient
```

Figure 4.18: Multiclass Dice Similarity Coefficient implementation.

4.6.2 3D UNET

For the 3D unet, we used the same optimizer and the evaluation metric as the 2D UNET approaches, but for the loss function, due to heavy class imbalance we tried something different and went with Soft Dice loss. The formula is:

$$Loss\,(p,q) = 1 - \frac{2 \times \sum_{i,j} p_{ij} q_{ij} + \epsilon}{\left(\sum_{i,j} p_{ij}^2\right) + \left(\sum_{i,j} q_{ij}^2\right) + \epsilon} \tag{4.2}$$

- p is our predictions
- q is the ground truth
- In practice each q_i will either be 0 or 1.
- ϵ is a small number that is added to avoid division by zero

The soft Dice loss ranges between:

- 0: perfectly matching the ground truth distribution q
- 1: complete mismatch with the ground truth.

soft_dice_loss

```
def soft_dice_loss(y_true, y_pred, axis=(0, 1, 2),
                   epsilon=0.00001):
    """ epsilon (float): small constant added to numerator and denominator to
                        avoid divide by 0 errors.   """
    dice_numerator = 2. * K.sum(y_true * y_pred, axis=axis) + epsilon
    dice_denominator = K.sum(y_true**2, axis=axis) + K.sum(y_pred**2, axis=axis) + epsilon
    dice_loss = 1 - K.mean((dice_numerator)/(dice_denominator))
    return dice_loss
```

Figure 4.19: Multiclass Soft Dice Loss implementation.

4.7 Conclusion

In this chapter, we talked about the development environment, the different used libraries and the dataset part, Also we explained our proposed approaches and what we will use as loss function and optimizer.

In the next chapter, we will go through the implementation and the results of the proposed approaches.

Chapter 5

Results and discussion

5.1 Introduction

this chapter is dedicated to the implementation of the different approaches introduced earlier, a discussion of the obtained results for both datasets and the obstacles that we faced.

In this chapter, we will show the implementation of the approaches, from data preprocessing to the obtained results of each approach for both datasets and then discuss them. We will seal this chapter by talking about the obstacles that we faced.

5.2 Brats2020 training and results

The data distribution was 295 images for training, 55 for validation and 19 for testing for all models that were trained using Brats2020

5.2.1 2D Unet

Training phase

The model was trained with 29500 slices for training and 5500 slices for validation for 100 epochs. the results are shown in the graphs below.

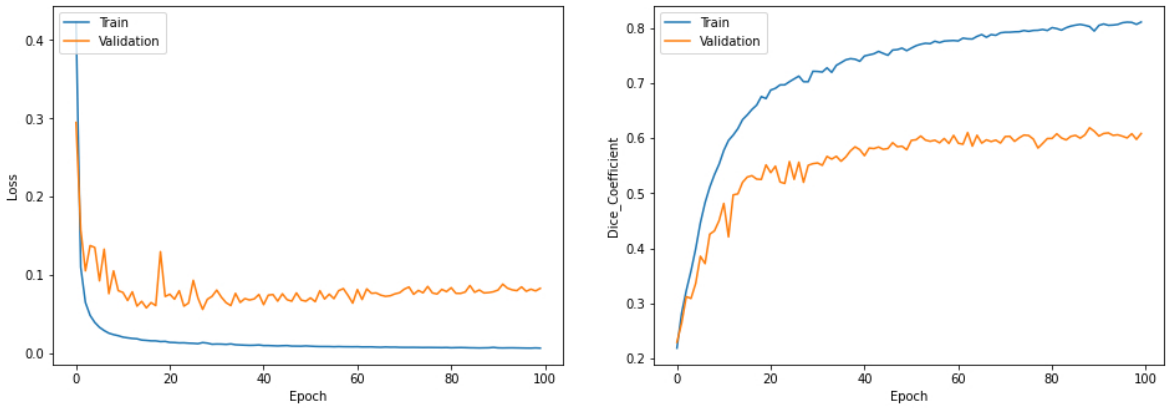

Figure 5.1: Brats20 2D Unet training and validation graphs of loss and dice score functions.

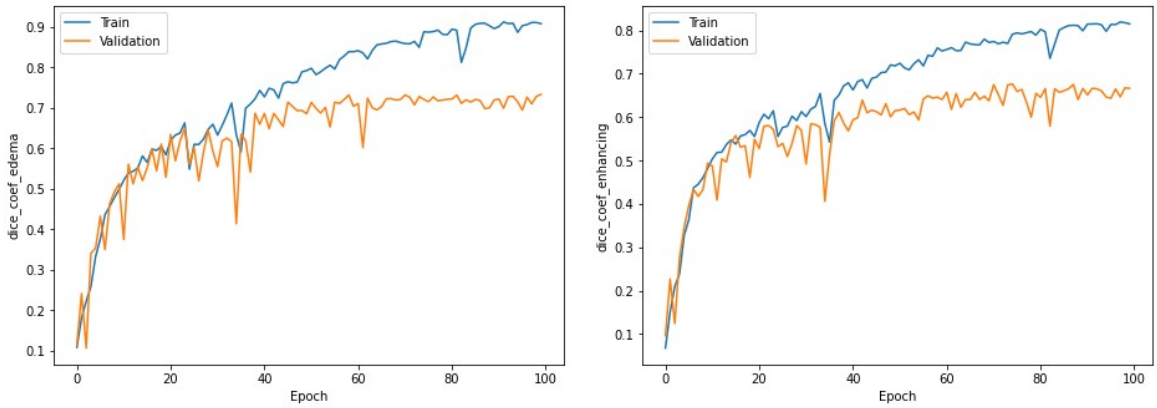

Figure 5.2: Brats20 2D Unet training and validation graphs of dice score for edema and enhancing classes.

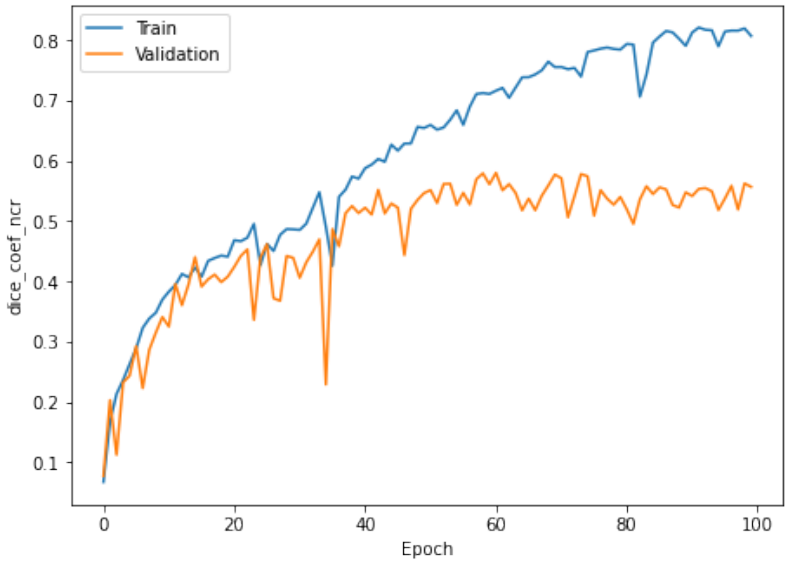

Figure 5.3: Brats20 2D Unet training and validation graph for dice score for necrotic class.

Test phase

The model was tested on 19 images (1900 slices). The results are shown in the tables below (see tables 5.1 and 5.2)

Test Loss	Test dice_coef	Test dice_coef_ncr	Test dice_coef_ed	Test dice_coef_en
0.0174	0.6631	0.5397	0.7407	0.8118

Table 5.1: Test results for 2D UNET with BraTS2020. ncr: necrotic, ed: edema and en: enhancing.

Test precision	Test sensitivity	Test specificity	Test accuracy
0.7961	0.6633	0.9983	0.9949

Table 5.2: Test results for precision, sensitivity, specificity and accuracy.

Prediction phase

Although the values of metrics seemed improving but we did not get what we expected in the predictions. Here are some examples. (see figure 5.4)

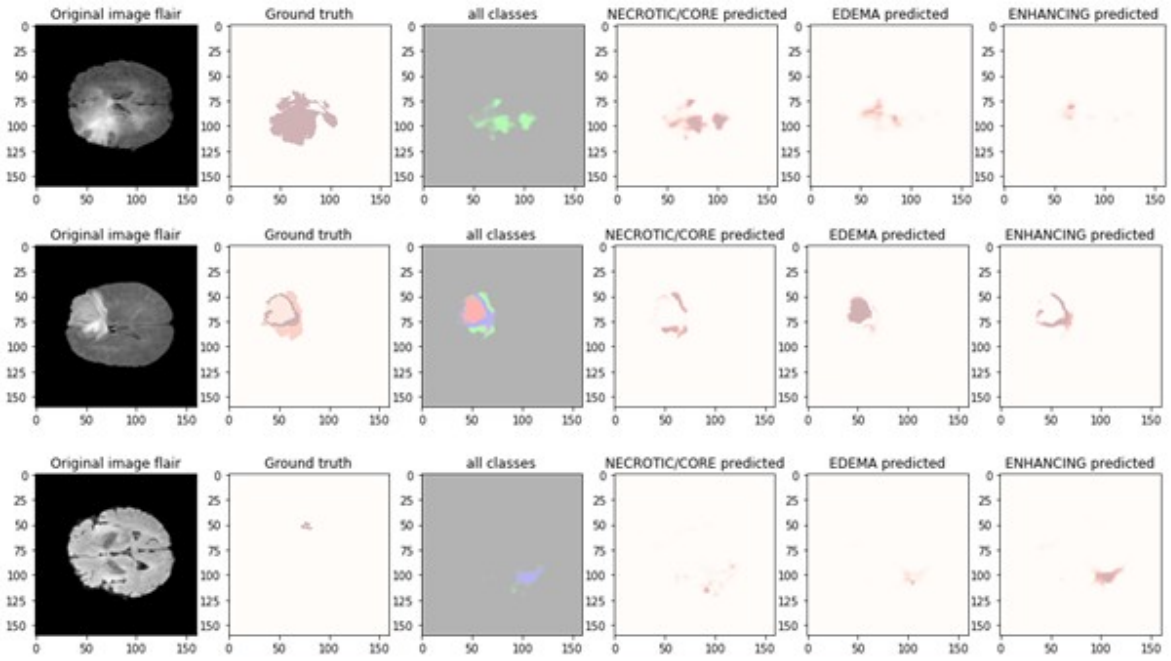

Figure 5.4: Predicted image with 2D UNET model

5.2.2 2D UNET models with transfer learning

These models were trained with 29500 slices for training and 5500 slices for validation for 100 epochs. the results are shown in the graphs below. . the results are shown below.

2D Unet_VGG16

Training phase

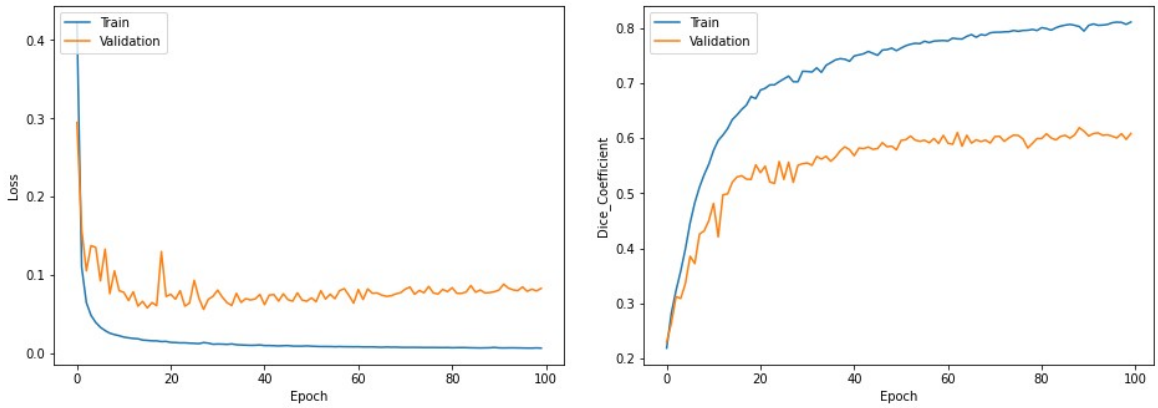

Figure 5.5: 2DUnet_VGG16 training and validation graphs of loss and dice score functions.

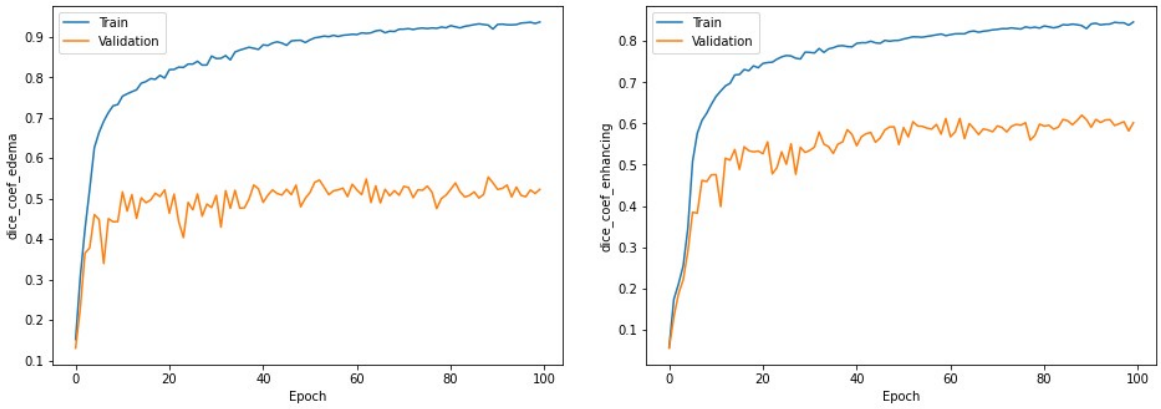

Figure 5.6: 2DUnet_VGG16 training and validation graphs of dice score for edema and enhancing classes.

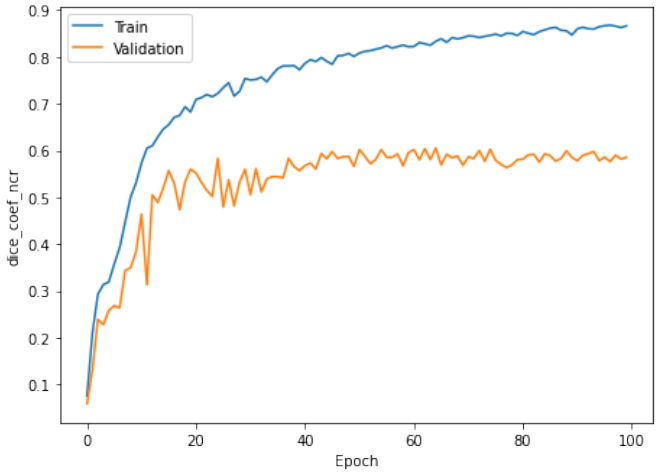

Figure 5.7: 2DUnet_VGG16 training and validation graph of dice score for necrotic class.

Test phase

The results of the test are shown in the tables below

Test Loss	Test dice_coef	Test dice_coef_ncr	Test dice_coef_ed	Test dice_coef_en
0.0504	0.5964	0.5190	0.4709	0.6983

Table 5.3: Test results for 2DUnet_VGG16. ncr: necrotic, ed: edema and en: enhancing.

Test precision	Test sensitivity	Test specificity	Test accuracy
0.6914	0.4988	0.9968	0.9864

Table 5.4: Test results for precision, sensitivity, specificity and accuracy.

Prediction phase

Here are some predictions:

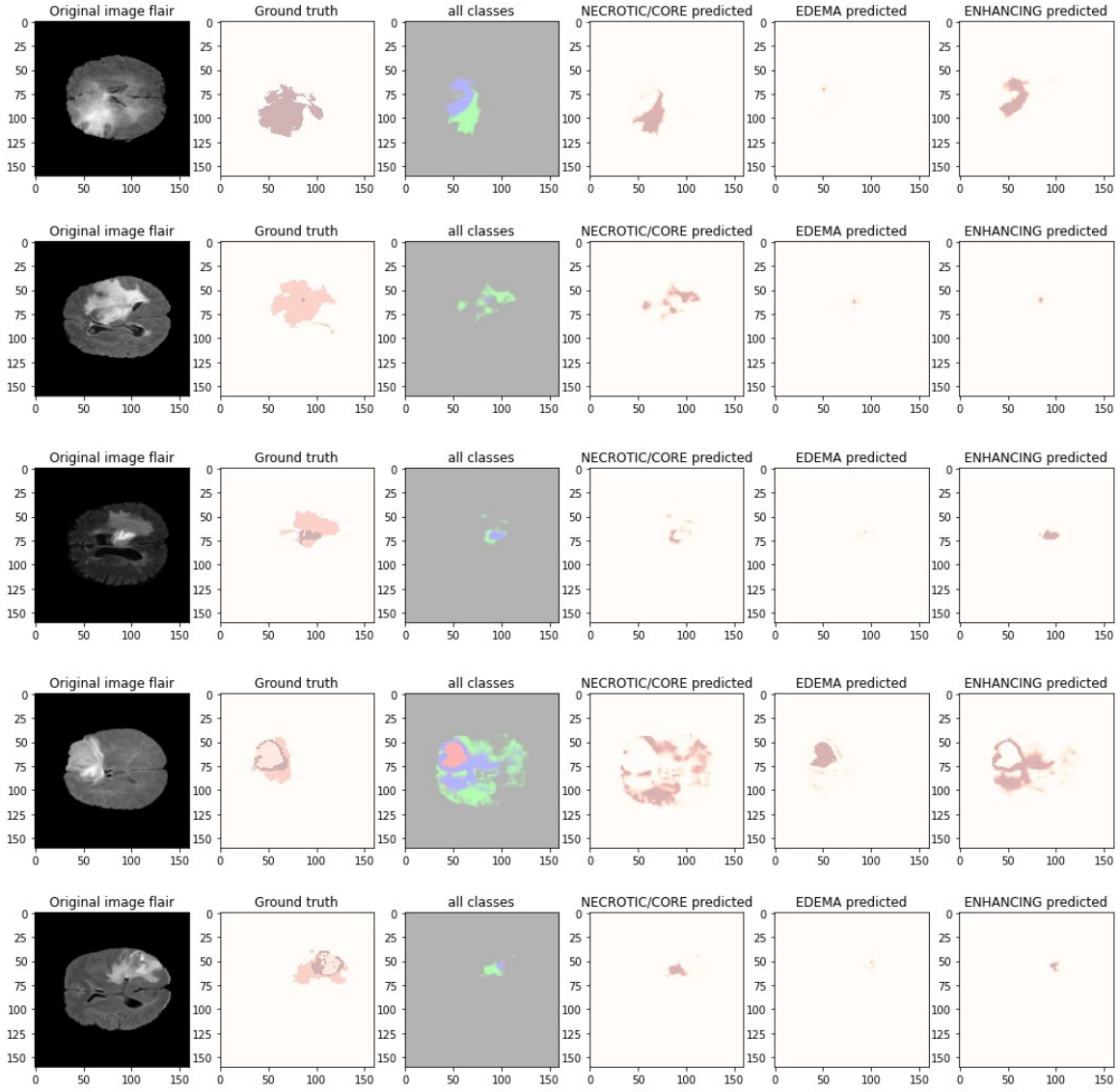

Figure 5.8: Predicted images with 2DUnet_VGG16 model

2D Unet_Rsnet50

Training phase

The results of training are shown in the graphs below.

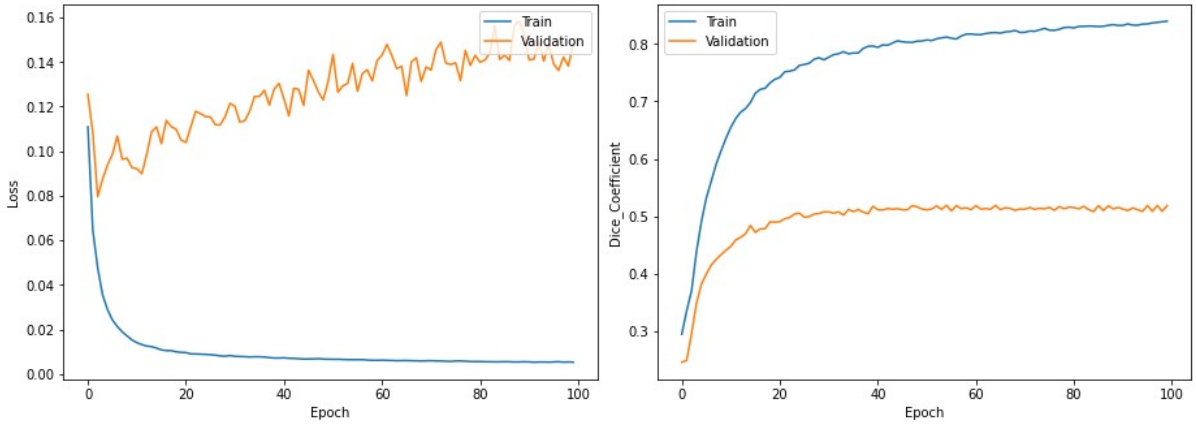

Figure 5.9: 2DUnet_Resnet50 training and validation graphs of loss and dice score functions.

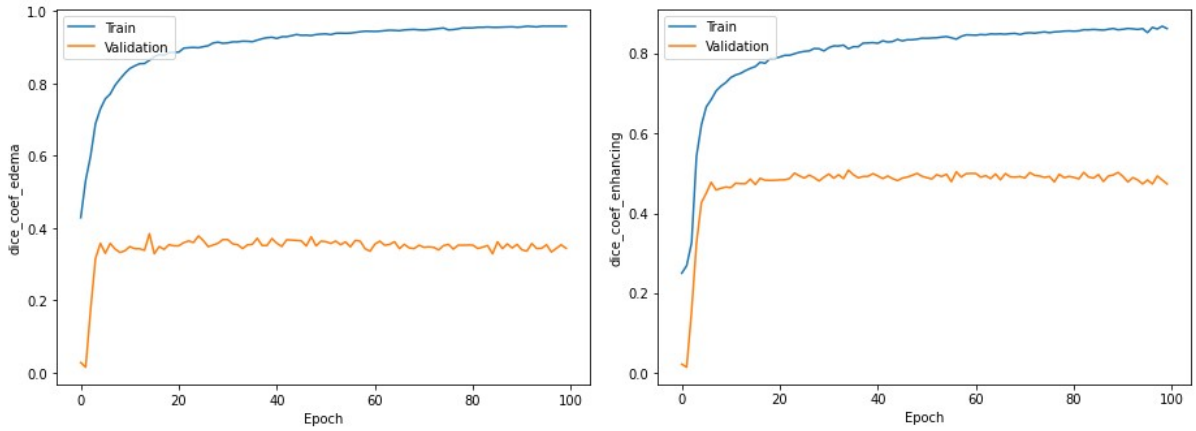

Figure 5.10: 2DUnet_Resnet50 training and validation graphs of dice score for edema and enhancing classes.

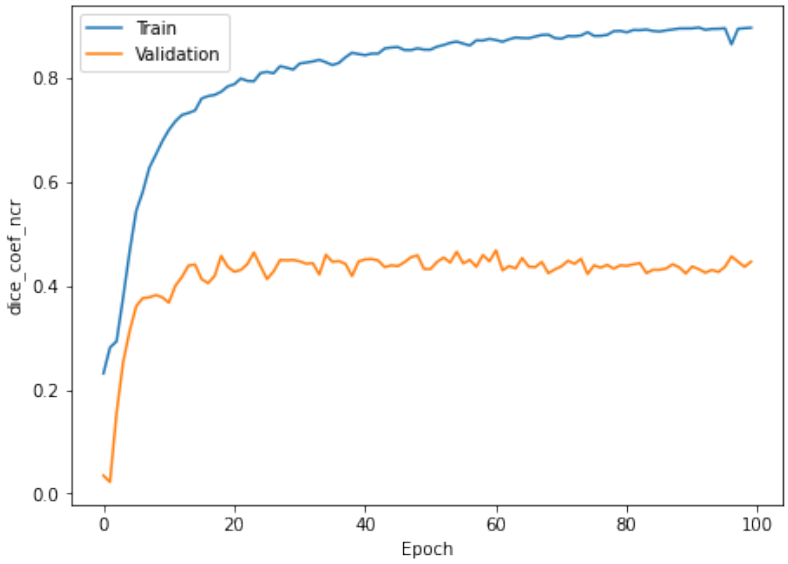

Figure 5.11: 2D Unet_Resnet50 training and validation graph of dice score for necrotic class.

Test phase

The results of the test are shown in the tables below.

Test Loss	Test dice_coef	Test dice_coef_ncr	Test dice_coef_ed	Test dice_coef_en
0.1553	0.4917	0.3427	0.2936	0.5440

Table 5.5: Test results for 2DUnet_Resnet50 . ncr: necrotic, ed: edema and en: enhancing.

Test precision	Test sensitivity	Test specificity	Test accuracy
0.6002	0.3736	0.9920	0.9694

Table 5.6: Test results for precision, sensitivity, specificity and accuracy.

Prediction phase

Here are some predictions:

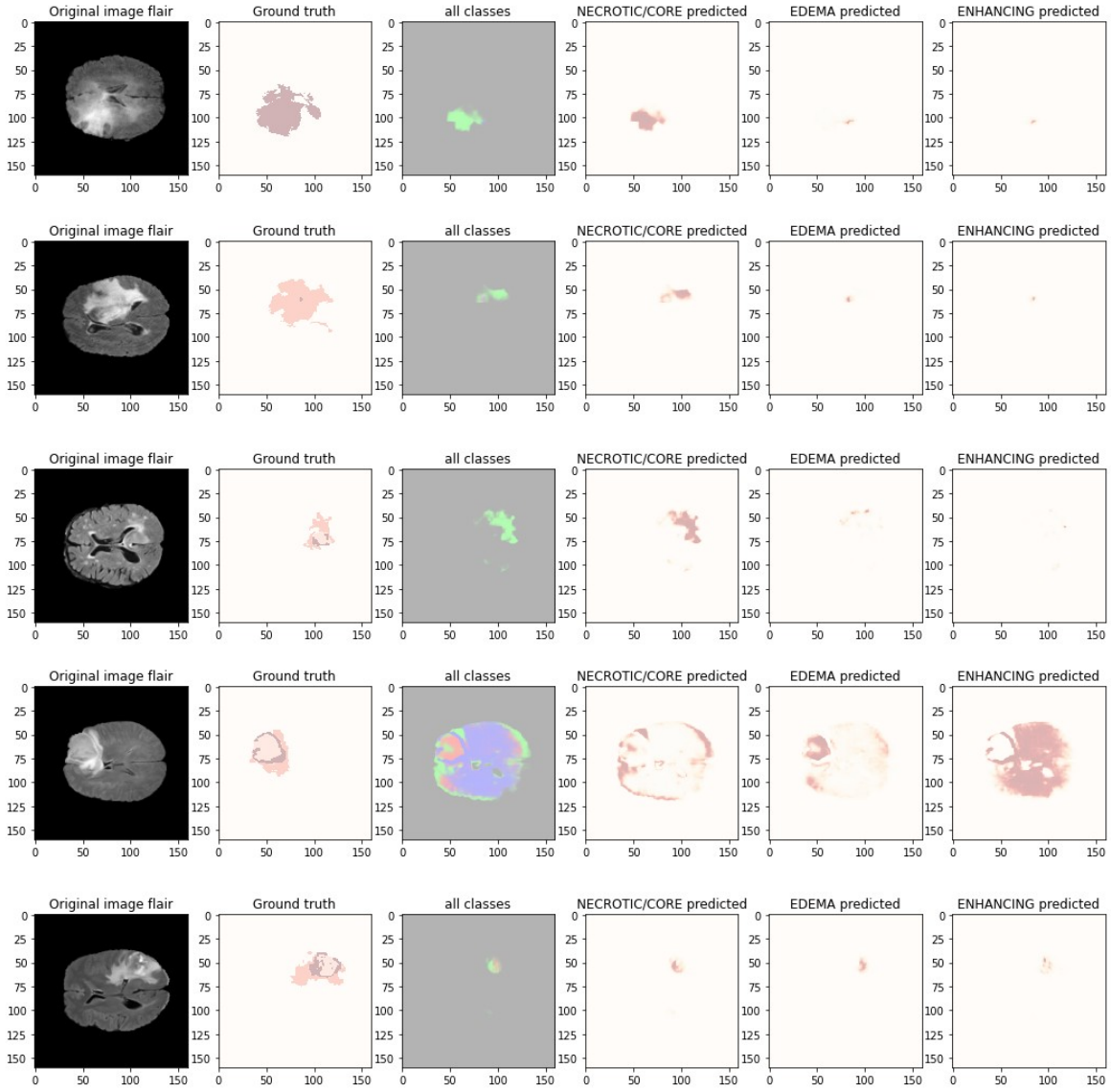

Figure 5.12: Predicted images with 2DUnet_resnet50 model.

2D Unet_InceptionV3

Training phase

These graphs represent the values of both training and validation for 100 epochs.

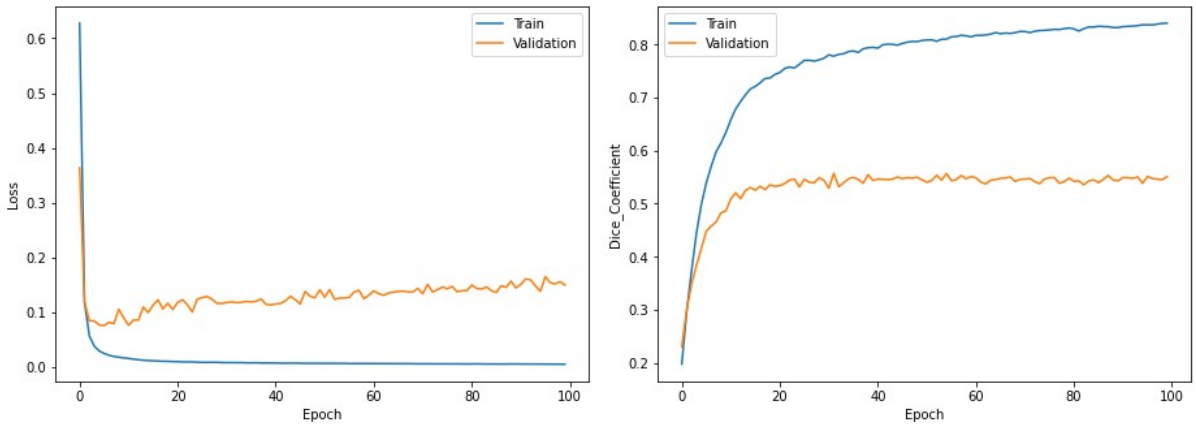

Figure 5.13: 2DUnet_InceptionV3 Training and validation graphs of loss and dice score functions.

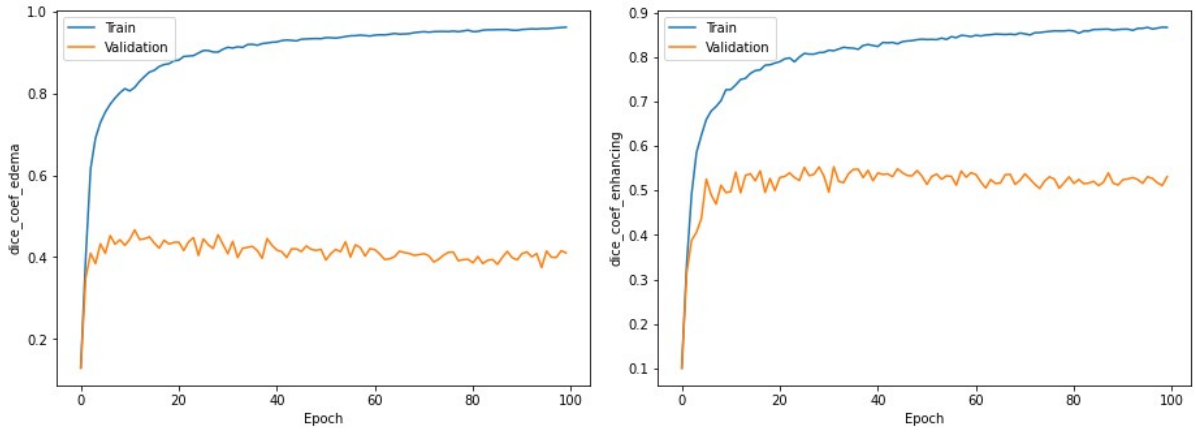

Figure 5.14: 2DUnet_InceptionV3 Training and validation graph of dice score for edema and enhancing classes.

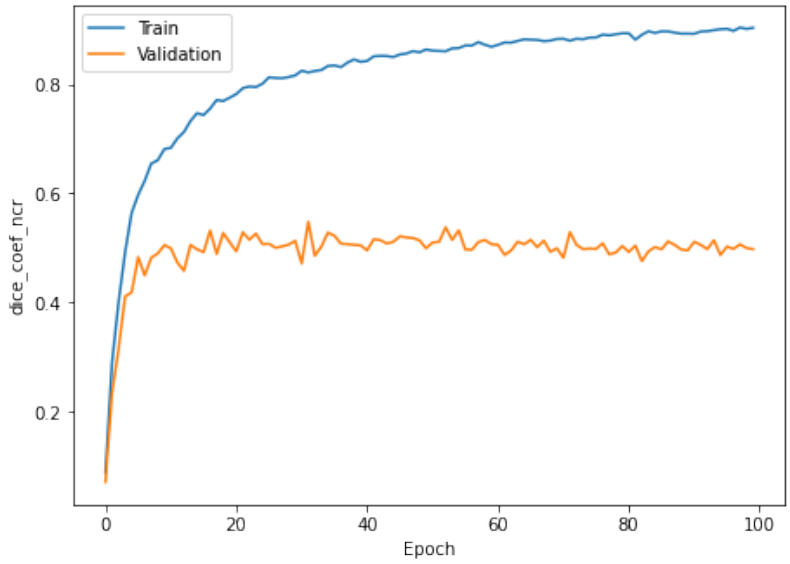

Figure 5.15: 2DUnet_InceptionV3 Training and validation of dice score graph for necrotic class.

Test phase

The results of the test are shown in the tables below.

Test Loss	Test dice_coef	Test dice_coef_ncr	Test dice_coef_ed	Test dice_coef_en
0.1376	0.5101	0.4291	0.2950	0.5197

Table 5.7: Test results for 2DUnet_InceptionV3 model. ncr: necrotic, ed: edema and en: enhancing.

Test precision	Test sensitivity	Test specificity	Test accuracy
0.5482	0.3940	0.9911	0.9670

Table 5.8: Test results for precision, sensitivity, specificity and accuracy.

Prediction phase

Here are some predictions from the test set:

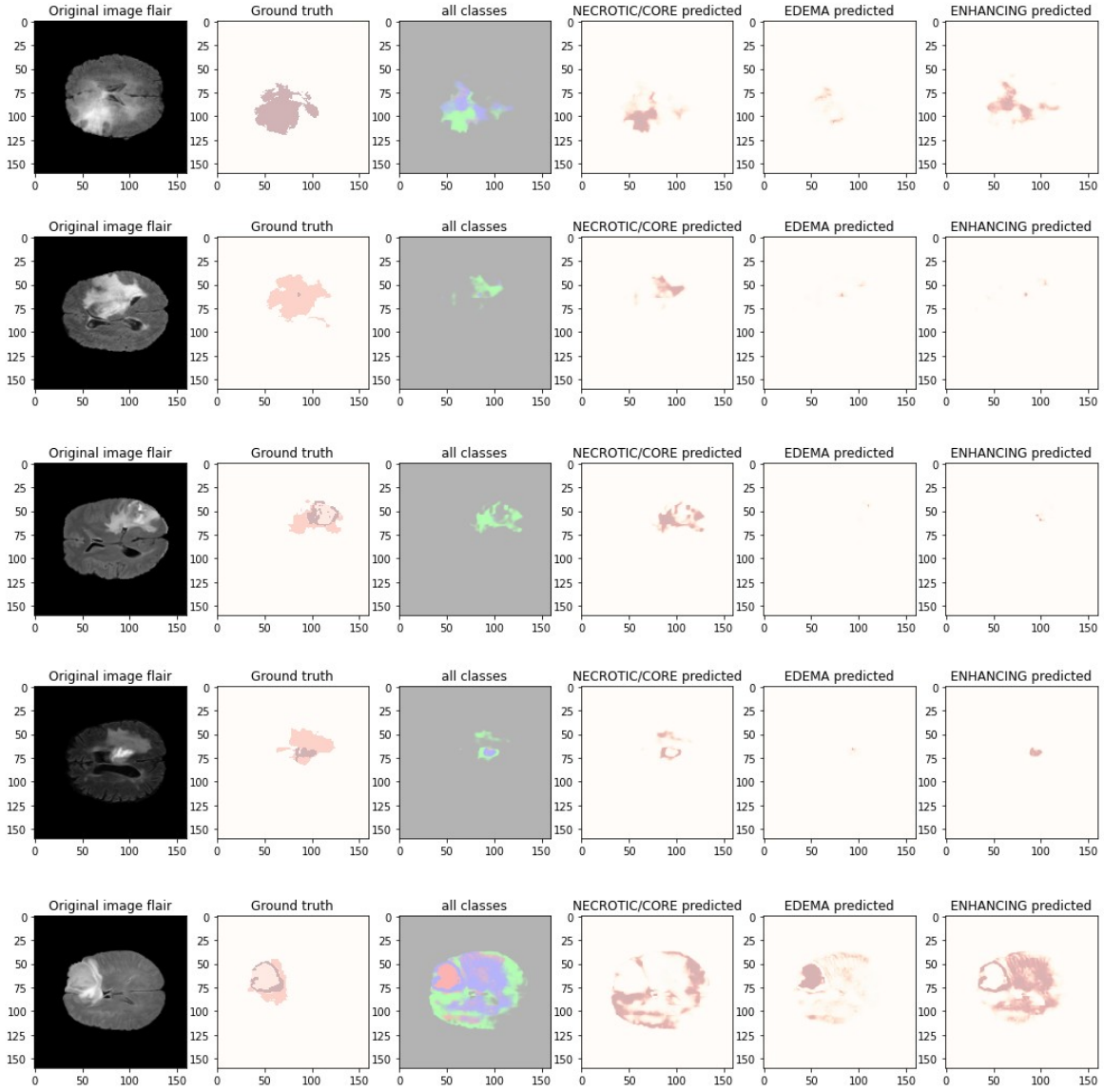

Figure 5.16: Predicted images with 2DUnet_InceptionV3 model.

5.2.3 3D UNET

This part was the most difficult part of the work due to resource limitations and Colab's new policies (discussed later in Obstacles section).

Training phase

Using 8850 patches(295 images) for training and 1650 patches(55 images) for validation, the model was trained for 36 epochs(140 hours approximately) the results of training are shown in the graphs below.

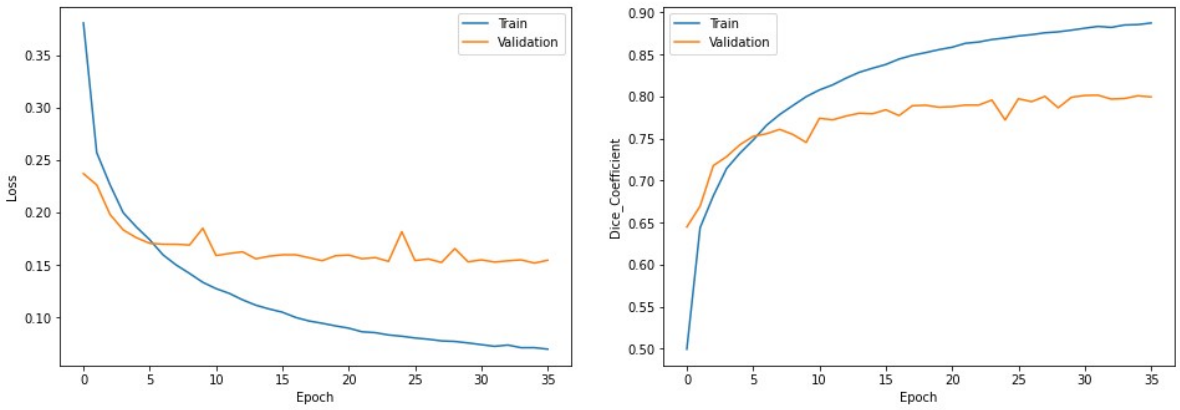

Figure 5.17: 3D Unet training and validation graphs of loss and dice score functions.

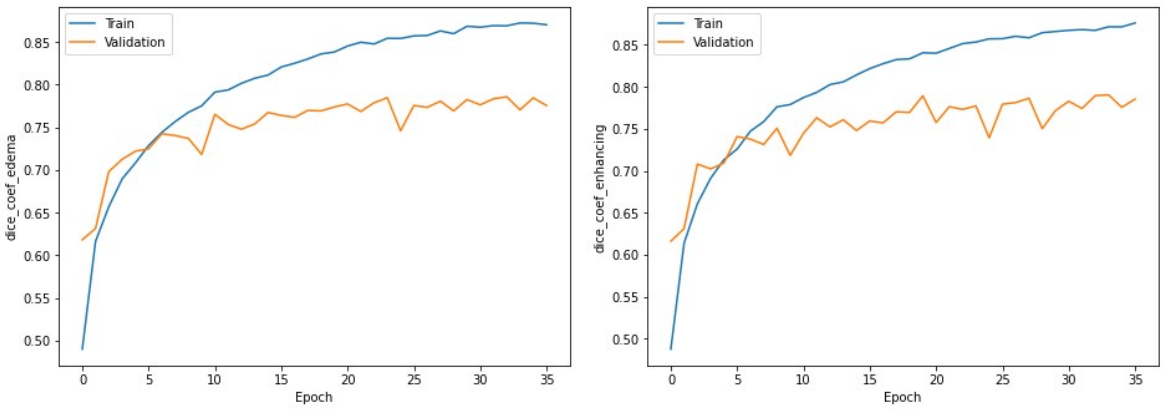

Figure 5.18: 3D Unet training and validation graphs of dice score for edema and enhancing classes .

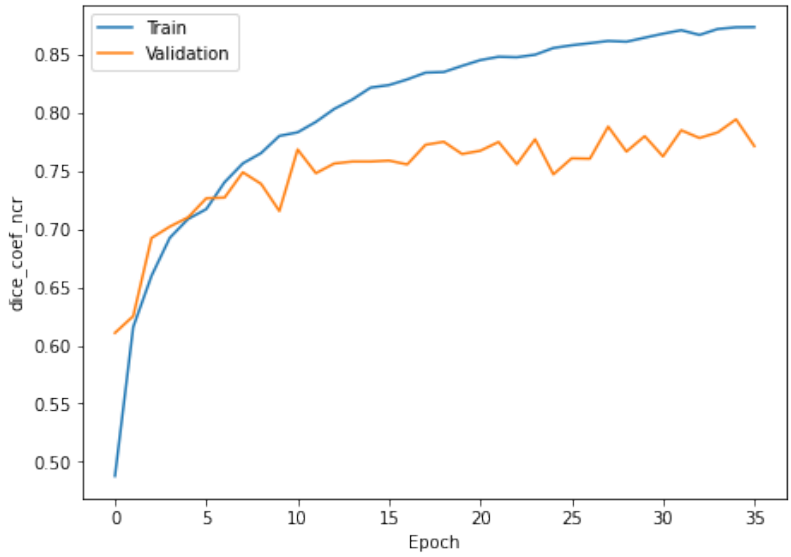

Figure 5.19: 3D Unet training and validation graphs of dice score for necrotic class.

Test phase

The model was tested on 570 patches (19 images). The results are shown in the table below (see tables 5.1 and 5.2)

Test Loss	Test dice_coef	Test dice_coef_ncr	Test dice_coef_ed	Test dice_coef_en
0.1938	0.7626	0.7360	0.7335	0.7679

Table 5.9: Test results for 3D UNET on brats2020. ncr: necrotic, ed: edema and en: enhancing.

Test precision	Test sensitivity	Test specificity	Test accuracy
0.7809	0.7822	0.9983	0.6505

Table 5.10: Test results for precision, sensitivity, specificity and accuracy.

Prediction phase

Here are some predictions from the test set:

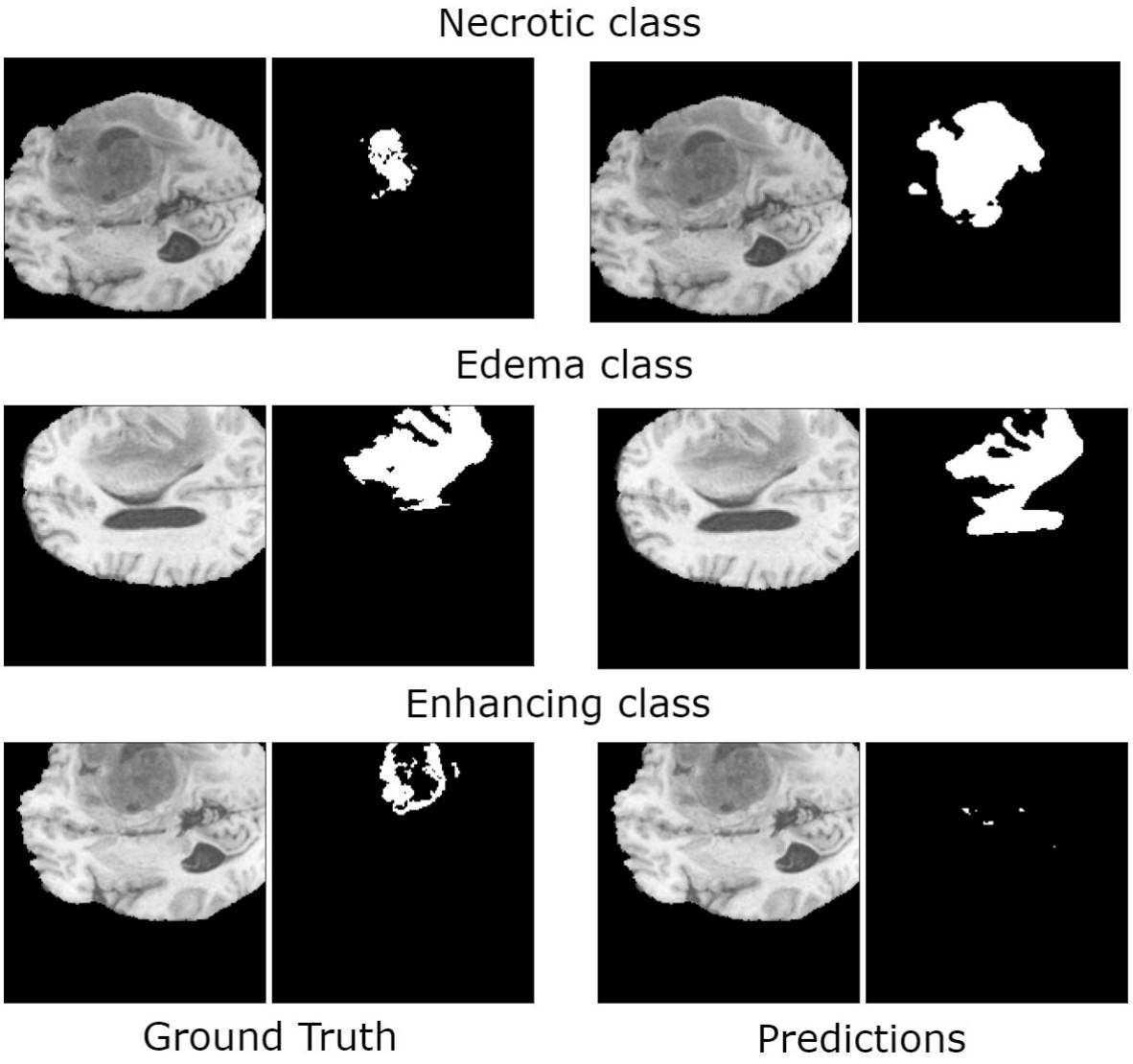

Figure 5.20: Prediction for 3D UNET model BraTS20

5.3 Brats2017 training and results

5.3.1 2D Unet

Training phase

This model was trained with 387 images for training and 73 images for validation (100 slices were took from each image) for 104 epochs, the results are shown in the graphs below.

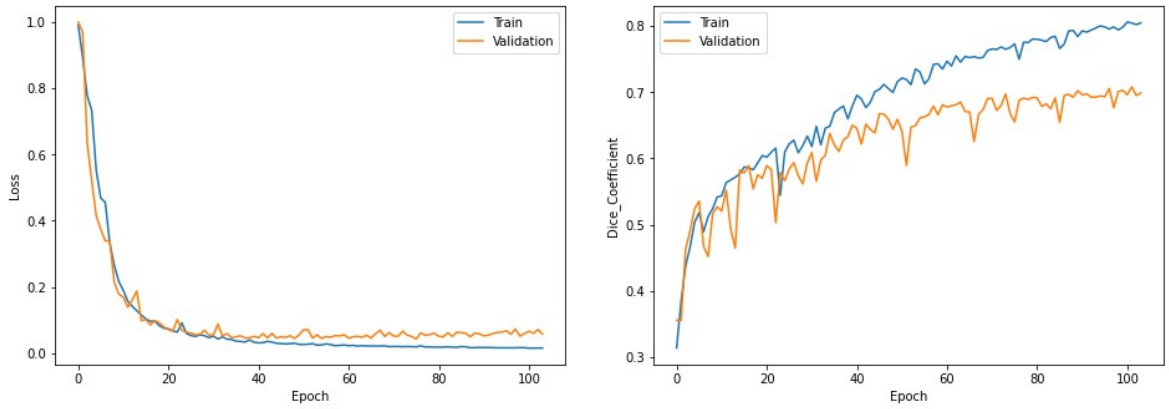

Figure 5.21: Brats17 2D Unet training and validation graphs of loss and dice score functions.

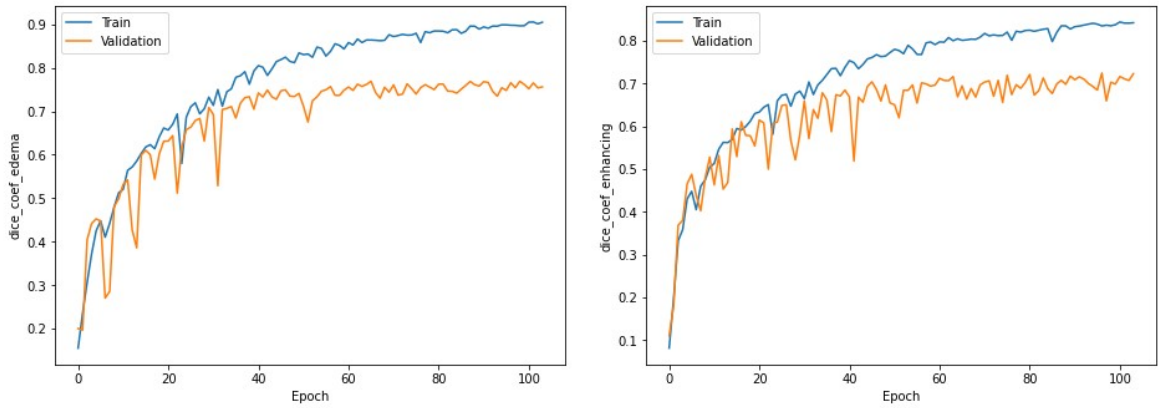

Figure 5.22: Brats17 2D Unet training and validation graphs of dice score for edema and enhancing classes.

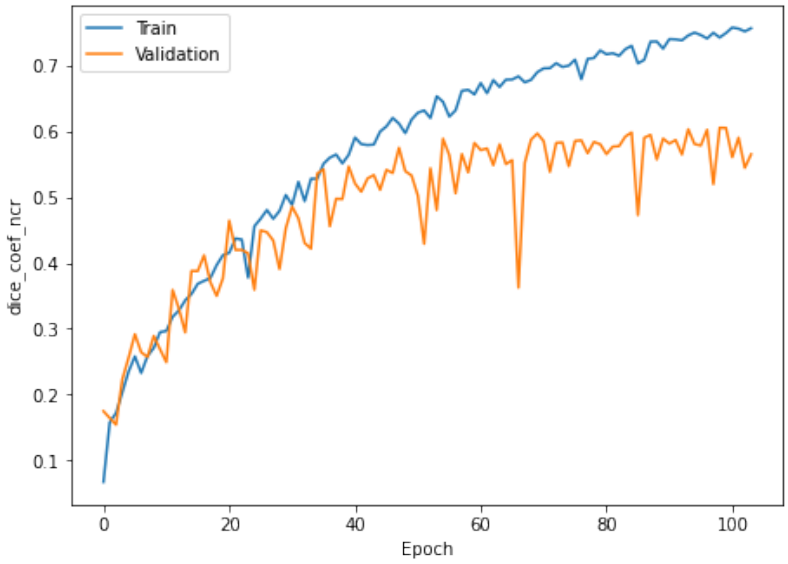

Figure 5.23: Brats17 2D Unet training and validation graphs of dice score for necrotic class.

Test phase

The model was tested on 24 MRI images (100 slices were took from each image). The results are shown in the table below (see tables 5.11 and 5.12)

Test Loss	Test dice_coef	Test dice_coef_ncr	Test dice_coef_ed	Test dice_coef_en
0.0581	0.6987	0.5653	0.7563	0.7228

Table 5.11: Test results for 2D UNET with BraTS2017. ncr: necrotic, ed: edema and en: enhancing.

Test precision	Test sensitivity	Test specificity	Test accuracy
0.7752	0.5756	0.9962	0.9837

Table 5.12: Test results for precision, sensitivity, specificity and accuracy.

Prediction phase

Here are some predictions for different slices for an image from the test set (see figure 5.24)

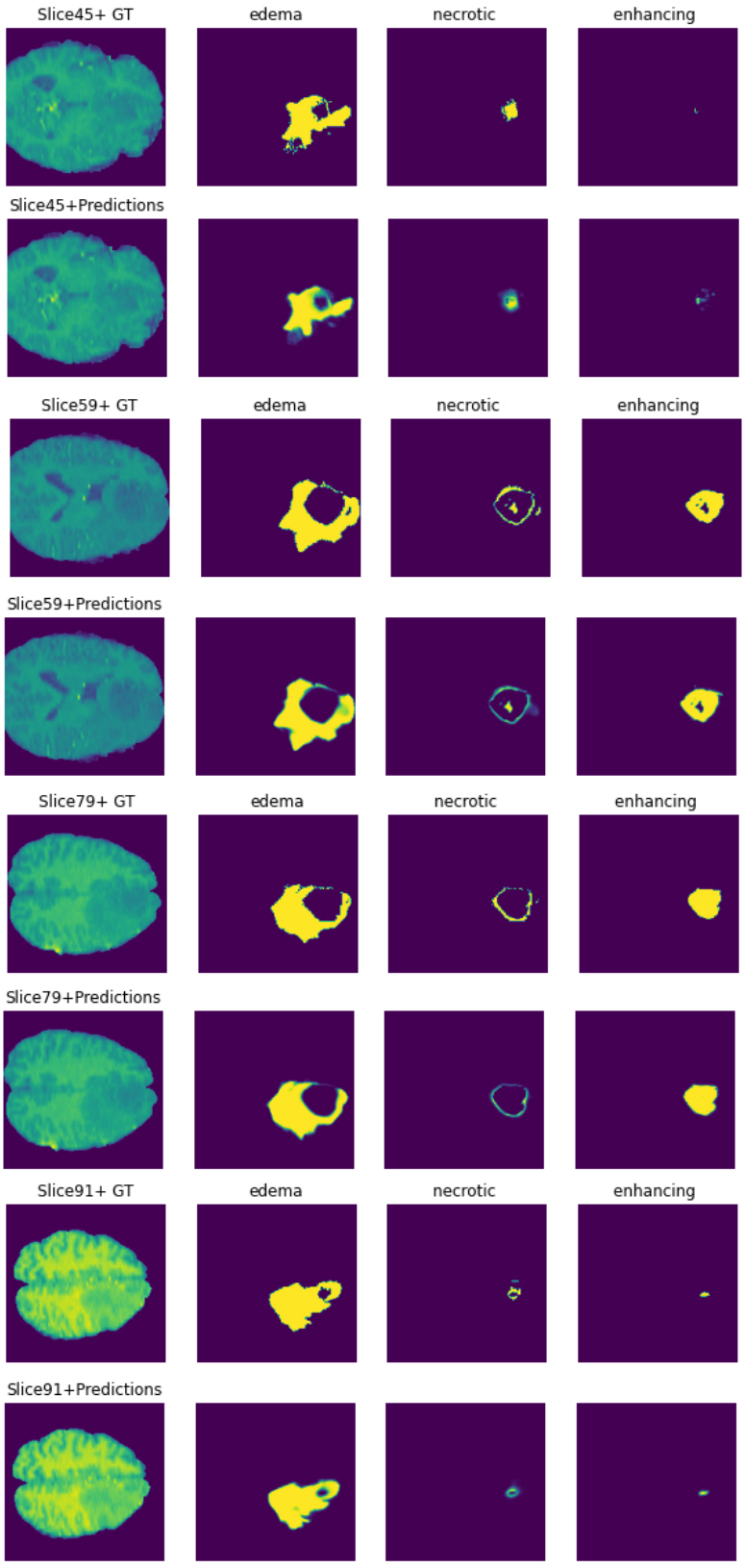

Figure 5.24: Predictions for 2D UNET model brats17

5.3.2 3D Unet

Training phase

For the 3d approach using BraTS 17 the model was traind with 387 images for training and 73 images for validation, from each image 20 sub-volumes were generated ,which gave 7740 sub-volumes for training and 1460 sub-volumes for validation. 24 images were left for testing. The model was trained in total for 13 epochs (40 hours approximately)The results are shown below.

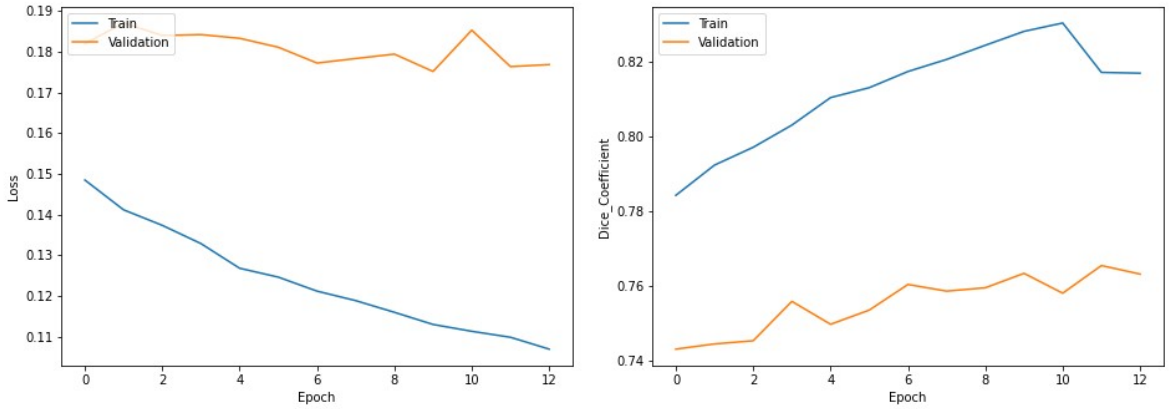

Figure 5.25: Brats17 3D Unet training and validation graphs of loss and dice score functions.

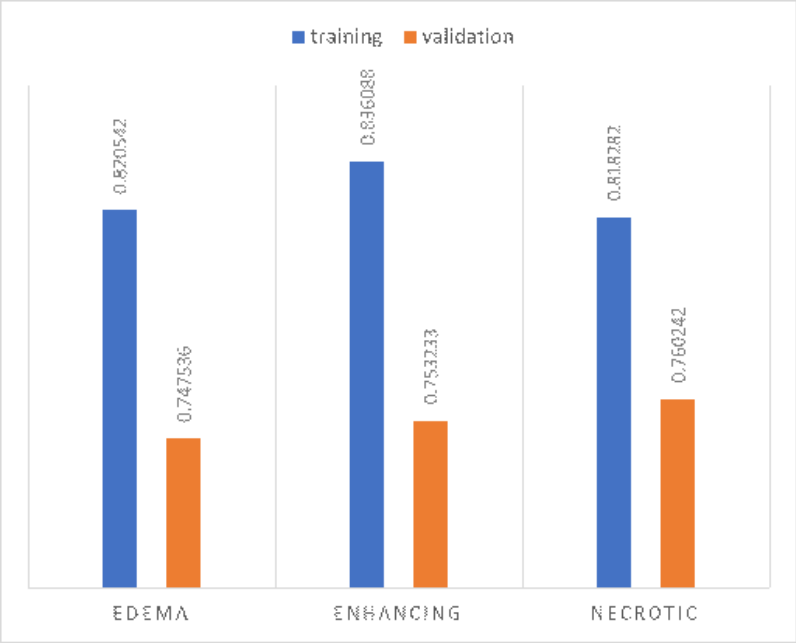

Figure 5.26: Brats17 3D Unet training and validation values of dice score for each class in the 13th epoch.

Test phase

The model was tested on 480 sub-volumes generated from the 24 images left for testing(20 per each). The results are shown in the tables below (see tables 5.13 and 5.14)

Loss	Dice coefficient	Dice for necrotic	Dice for edema	dice for enhancing
0.1767	0.7632	0.7657	0.7547	0.7575

Table 5.13: Test results for 3D UNET using BraTS2017. ncr: necrotic, ed: edema and en: enhancing.

Precision	Sensitivity	Specificity	Accuracy
0.7703	0.8131	0.9927	0.8064

Table 5.14: Test results for precision, sensitivity, specificity and accuracy.

Prediction phase

Here are some predictions from the test set

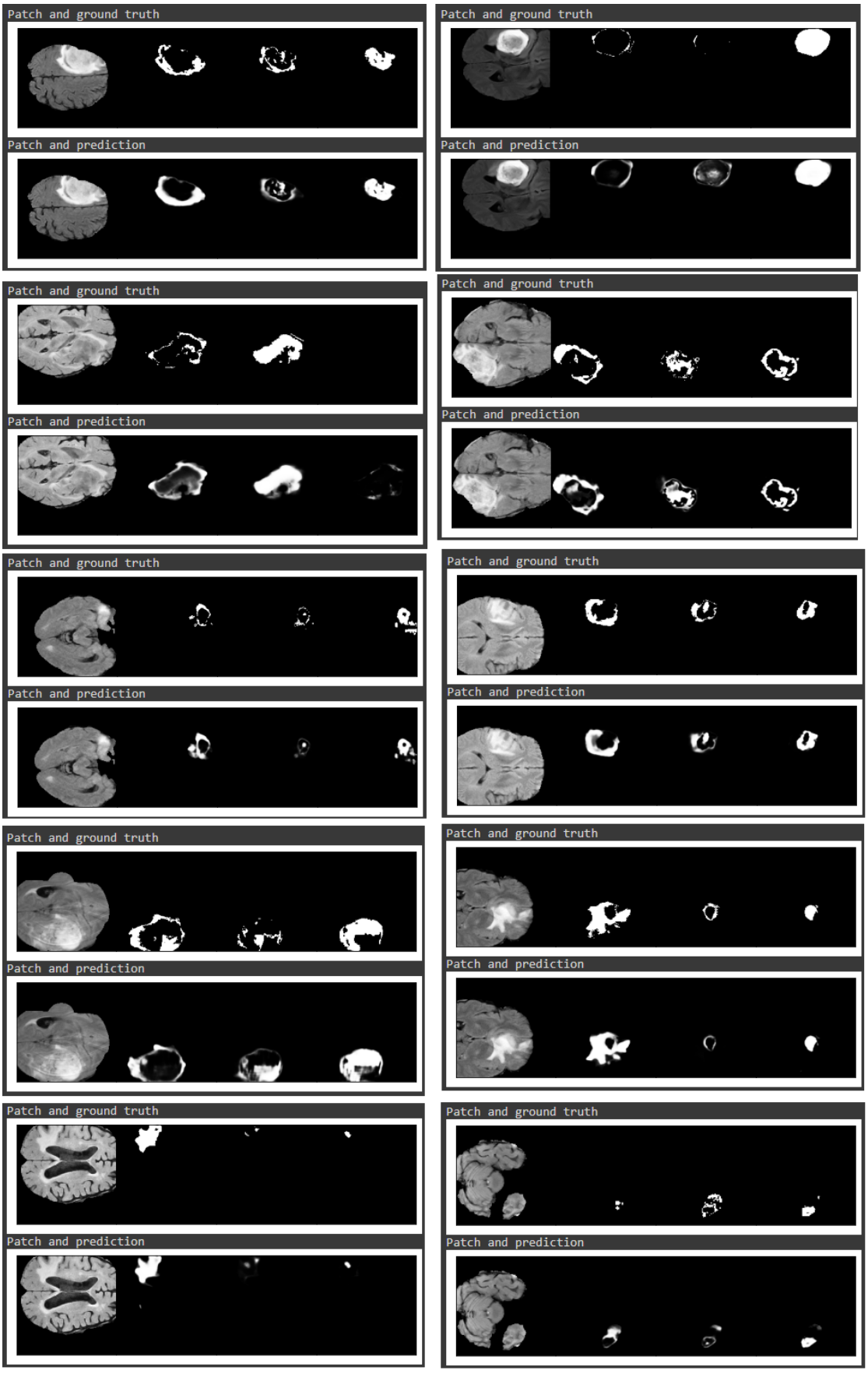

Figure 5.27: Patch predictions for brats17 3d model.

5.4 Discussion

5.4.1 Discussion of the obtained results

Brats2020: We were primarily planing to work with brats20 dataset. But as the training of the basic 2D model reached the 100th epoch and we performed some predictions, we were surprised by the results, because it is true that the metrics values were not very high but they were not that low either. We expected that the shape of the predicted tumor will at least look a little similar to the ones in the ground truth. We concluded that the data is very small and it is not enough to train the model.

Next, we tried transfer learning, where we used 3 different backbones (VGG16, RESNET50 and INCEPTIONV3). These 3 models were pre-trained on Imagenet dataset, and we thought that using them as feature extractors in the encoder of the Unet model would pay off, but we did not get good results. It is probably because we did not train the last layers for the 3 networks. VGG16 gave better results because it is a simple network, so it allows extracting simple characteristics, this is not the case for Resnet50 and InceptionV3 (the last layers contain more complex characteristics and optimized for the Imagenet base which is not a medical base) in the future we will try to retrain the last layers instead of freezing them.

At this point we supposed that 2D approaches are not effective because we had been breaking up the 3D MRI volumes into many 2D slices then each one of these slices was passed into the model which outputs the segmentation for that slice. One by one, each slice was passed through the segmentation model in this manner to generate a segmentation for every slice. The 2D slices would then be combined once again to form the 3D output volume of the segmentation. We figured that the drawback with this 2D approach is that we might lose important 3D context when using this approach. For instance, if there is a tumor in one slice, there is likely to be a tumor in the slices right adjacent to it. Since we had been passing in slices one at a time into the models, the model would not be able to learn this useful context.

For that we decided to try the 3D approach, which is definitely more efficient but has the drawback of consuming a lot of computation resources. This model took so much

time for only training for 36 epochs as it took approximately 140 non-continuous hours of training. It scored an average of 0.74% for all classes for the dice coefficient on the test set(see table5.1), which looked very promising. But the predictions were very poor especially for Necrotic and Enhancing classes (see figure 5.21). At this point we considered two candidates reasons for this failure, it was either a mistake we had made during the data preprocessing or the Brats Challenge for 2020 was very difficult as the first place winner for that year challenge had to train their model for 1000 epochs. [74]

We rechecked the scripts but we could not find what was wrong, we so we thought of using the same scripts and the same models with another dataset and see if there was any difference. The choice fell on Brats17 because it was the easiest to get.

Brats2017: We got this dataset from the Medical Segmentation Decathlon (MSD) challenge.[1] Unfortunately we did not have the chance to work on this data as much as we did with brats2020 for the reasons that we will explain in the next section (Obstacles). We only tried basic Unet models (2D and 3D), this time the 3D model was trained for only 13 epochs. But the good news in all of this were the results that we got using this dataset.

The predictions were acceptable (see figures 5.24 and 5.27) The 2D model scored 0.80% Dice coefficient on the training set and 0.69% on the test set, while the 3D model scored 0.81% on the training set and 0.76% on the test set.

The results of our models

All the results obtained from the proposed approaches are summarized in the following table (see *Table 5.15*).

[1] https://decathlon-10.grand-challenge.org/

Rank	Architecture	Dice Score				Sensitivity	Specificity	Loss
		NCR	EN	ED	WT	WT	WT	WT
BraTS 2017								
1	3D Unet	0.7657	0.7575	0.7547	0.7632	0.8131	0.9927	0.1767
2	2D Unet	0.5653	0.7228	0.7563	0.6987	0.5756	0.9962	0.0581
BraTS 2020								
1	3D Unet	0.7360	0.7679	0.7335	0.7626	0.7822	0.9983	0.1938
2	2D Unet	0.5397	0.8118	0.7407	0.6631	0.6633	0.9983	0.0174
3	2DUnet_VGG16	0.5190	0.6983	0.4709	0.5964	0.4988	0.9968	0.0504
4	2DUnet_InceptionV3	0.4291	0.5197	0.2950	0.5101	0.3940	0.9911	0.1376
5	2DUnet_Resnet50	0.3427	0.5440	0.2936	0.4917	0.3736	0.9920	0.1553

Table 5.15: A summary of all models on BraTS17 and BraTS20 test results. EN—Enhancing class,NCR-Necrotic class,ED-Edema class and WT—Whole tumor.

5.4.2 Comparaison with the winners of BraTS Challenge

The table below shows a comparaison between the winners of both Datasets (BraTS17 and BraTS20) and the model that achieved the best results in our implementation (3D Unet).

Rank	Reference	Architecture	Dice Score			Sensitivity	Specificity
			ET	WT	TC	WT	WT
BraTS 2017							
1	[67]	Ensemble	0.738	0.901	0.797	0.895	0.995
2	[80]	Cascaded	0.786	0.905	0.838	0.915	0.995
#	3D Unet	3D Unet	0.757	0.763	-	0.813	0.992
BraTS 2020							
1	[74]	nnU-Net	0.820	0.889	0.850	-	-
2	[88]	H2NF-Net	0.787	0.912	0.842	-	-
#	3D Unet	3D Unet	0.767	0.762	-	0.782	0.998

Table 5.16: Comparaison of our best model with the winners of the challenge.

5.5 Obstacles

Implementing a deep learning model requires strong computing resources, our laptops didn't have those requirements nor they were available at our university. We saw that using cloud services was the best thing we could do. There are plenty of cloud services out there but just few of them are free and offer descent services, Google Colab was the appropriate choice because it offers 12 GB of RAM and an Nvidia K80 GPU.

Colab says that one session could last for 12 continuous hours only, well that was a problem because training our models was going to take way more than that especially 3D models. The solution for this was simple, we trained our models for the allowed time, and used callbacks to save their weights at the end of each epoch, and by that they were able to continue training in the next sessions from where they have stopped .

Unfortunately, the time limit was lower than that, it was only 8 hours ,but the worst part was that we got 8 hours only when the used account is new and has not used GPU before, if the same account was used multiple times the time limit dropped down to 3-4 hours, When training our 3D models the average time per epoch was 4h and this really was a problem because more than 70% of sessions disconnected before finishing even one epoch. Most of disconnections were due to time limit and the rest were because of the instability of internet, we had to use tricks so used more than 30 Gmail accounts. We were not able to create more accounts because they require to be linked to a phone number and there is a limit to how many could be linked to the same number.

Not only the time was a problem but resources were too because Colab shares them between users and we didn't get the full promised resources. One more problem was that Colab imposes an inactivity timeout to discourage users from using it for long-running tasks, sometimes sessions disconnected for no reason. Moreover, there was the Captcha inactivity test that if you don't check the box in a certain time the session disconnects and all the process is lost, and it would appear randomly so we had to keep an active eye on the browser during the entire training time.

To deal with all the problems mentioned before, google proposed colab pro which is a paid version with more advantages but unfortunately, there are only nine countries allowed to pay for it and Algeria wasn't included.

One more thing that Deep learning requires is data, medical data is known of being small because of patients' privacy, we were planning to use data augmentation to solve this problem, but with all the problems we mentioned above we were barely able to work the small raw data that we had and for this we were not able to try transfer learning for 3D models. Furthermore the datasets we worked with had a huge class imbalance, which made the task more difficult.

5.6 Conclusion

In this chapter, we covered the implementation of our approaches and discussed the obtained results, where we showed that we had some struggles in working with brats20

dataset, while things went smoother with brats17 and we got encouraging results although we did not use data augmentation. Because of resources limitations and all the troubles we faced during the implementation process that we mentioned in the previous section, we did not have the chance to give the models that worked with brats17 the training time they needed, and if only we had spent most of our time working on brats17 instead of brats20 we probably would have gotten better results. It is true that we did not get the results we were hoping for, but this is our first try in this field and we are looking forward to improve the performance of our models and make them more accurate in the future.

General conclusion

Brain tumor segmentation is a difficult task that entails delimiting cancerous tissues in medical images of the brain. This procedure is a touchstone of medical image analysis research and it is unquestionably an important step in computer-aided diagnosis systems. Manual segmentation requires experience and it is a time-consuming process. Recently, great process has been made in automatizing this task using deep learning(DL) in order to mainly use it to support the diagnosis process. In this work, we had the objective of proposing an automatic brain tumor segmentation approach automatic brain tumor segmentation system using convolutional neural networks for MRI images. We tried improving the performance of Unet architecture in segmenting brain tumor images by using transfer learning, which is a common idea in image classification tasks. We used three different backbones that were trained on a non medical dataset(ImageNet) as feature extractors in the encoder part, which are VGG16, ResNet50 and InceptionV3. The results were encouraging looking at the small data that we had and the resources limitations we faced.

This was our first experience in dealing with real world projects in the field of deep learning, this allowed us to improve our knowledge and skills in this field. We have more ideas we plan to try in the future in order to improve the performance of the models that we worked with, including trying different levels of fine tuning when using transfer learning, working more on the preprocessing step, trying other backbones and using data augmentation.

We would like to conclude by pointing out one of the main challenges with applying AI algorithms in hospitals, which is achieving reliable generalization, because in the last recent years there were many attempts to implement systems that achieved good results in brain tumor segmentation, but we do not see them in use today at hospitals or clinics around us. Generalization can be hard due to a variety of reasons, one of them is that these systems were trained on a data collected from a few countries over a few years, but MRI technology is not standard across the globe and across time. The latest scanners have much higher resolution than older scanners. So, before we apply the segmentation models in hospitals, we have to make sure that the model is able to generalize to the resolution of the scanner at that hospital.

Bibliography

[1] "Cancer today." http://gco.iarc.fr/today/home. ONLINE; accessed 2021-06-02.

[2] "World Cancer Day: The situation is getting worse in Algeria , Algerian Radio." https://www.radioalgerie.dz/news/fr/article/20210204/206616.html. ONLINE; accessed 2021-06-02.

[3] "Brain Tumor - Statistics." https://www.cancer.net/cancer-types/brain-tumor/statistics, June 2012. ONLINE; accessed 2021-06-02.

[4] N. Butowski, *Epidemiology and diagnosis of brain tumors*. 2015.

[5] Q. T. Ostrom, H. Gittleman, J. Xu, C. Kromer, Y. Wolinsky, C. Kruchko, and J. S. Barnholtz-Sloan, "CBTRUS Statistical Report: Primary Brain and Other Central Nervous System Tumors Diagnosed in the United States in 2009–2013," *Neuro-Oncology*, vol. 18, pp. v1–v75, Oct. 2016.

[6] "Epidemiology and molecular pathology of glioma Nature Reviews Neurology." https://www.nature.com/articles/ncpneuro0289. ONLINE; accessed 2021-06-02.

[7] L. M. DeAngelis, "Brain Tumors," *New England Journal of Medicine*, vol. 344, pp. 114–123, Jan. 2001.

[8] S. Roy, S. Sadhu, S. Bandyopadhyay, D. Bhattacharyya, and T.-H. Kim, "Brain Tumor Classification using Adaptive Neuro-Fuzzy Inference System from MRI," *International Journal of Bio-Science and Bio-Technology*, vol. 8, pp. 203–218, June 2016.

[9] D. N. Louis, H. Ohgaki, O. D. Wiestler, W. K. Cavenee, P. C. Burger, A. Jouvet, B. W. Scheithauer, and P. Kleihues, "The 2007 WHO classification of tumours of the central nervous system," *Acta Neuropathol*, vol. 114, pp. 97–109, Aug. 2007.

[10] A. K. Sangaiah, *Deep Learning and Parallel Computing Environment for Bioengineering Systems*. Elsevier, 2019.

[11] J. L. Fisher, J. A. Schwartzbaum, M. Wrensch, and J. L. Wiemels, "Epidemiology of Brain Tumors," *Neurologic Clinics*, vol. 25, pp. 867–890, Nov. 2007.

[12] T. M. Clinic, *Mayo Clinic A to Z Health Guide: Everything You Need to Know About Signs, Symptoms, Diagnosis, Treatment and Prevention*. Time Home Entertainment, May 2015.

[13] V. Dubay and M. Louise, *100 Questions & Answers About Brain Tumors*.

[14] J. R. Fink, M. Muzi, M. Peck, and K. A. Krohn, "Multimodality Brain Tumor Imaging: MR Imaging, PET, and PET/MR Imaging," *J Nucl Med*, vol. 56, pp. 1554–1561, Oct. 2015.

[15] S. Sundari Ilangovan, B. Mahanty, and S. Sen, "Biomedical Imaging Techniques," pp. 401–421, Jan. 2016.

[16] M. Angulakshmi and G. G. L. Priya, "Automated brain tumour segmentation techniques— A review," *International Journal of Imaging Systems and Technology*, vol. 27, no. 1, pp. 66–77, 2017.

[17] H. Kasban, M. El-bendary, and D. Salama, "A Comparative Study of Medical Imaging Techniques," *International Journal of Information Science and Intelligent System*, vol. 4, pp. 37–58, Apr. 2015.

[18] S. Luo, R. Li, and S. Ourselin, "A new deformable model using dynamic gradient vector flow and adaptive balloon forces," *APRS Workshop on Digital Image Computing*, Mar. 2003.

[19] M. N. Gurcan, L. E. Boucheron, A. Can, A. Madabhushi, N. M. Rajpoot, and B. Yener, "Histopathological Image Analysis: A Review," *IEEE Reviews in Biomedical Engineering*, vol. 2, pp. 147–171, 2009.

[20] A. Perkins and G. Liu, "Primary Brain Tumors in Adults: Diagnosis and Treatment," *Am Fam Physician*, vol. 93, pp. 211–217, Feb. 2016.

[21] J. Heaton, *Artificial Intelligence for Humans, Volume 3: Deep Learning and Neural Networks*. 2015.

[22] H. J. KELLEY, "Gradient Theory of Optimal Flight Paths," *ARS Journal*, vol. 30, no. 10, pp. 947–954, 1960. Publisher: American Institute of Aeronautics and Astronautics _eprint: https://doi.org/10.2514/8.5282.

[23] "The numerical solution of variational problems," *Journal of Mathematical Analysis and Applications*, vol. 5, pp. 30–45, Aug. 1962. Publisher: Academic Press.

[24] K. Fukushima and S. Miyake, "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Visual Pattern Recognition," in *Competition and Cooperation in Neural Nets* (S. Levin, S.-i. Amari, and M. A. Arbib, eds.), vol. 45, pp. 267–285, Berlin, Heidelberg: Springer Berlin Heidelberg, 1982. Series Title: Lecture Notes in Biomathematics.

[25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," *Advances in Neural Information Processing Systems*, vol. 3, June 2014.

[26] "Introduction to Machine Learning with Python [Book]." ISBN: 9781449369415.

[27] D. Fumo, "Types of Machine Learning Algorithms You Should Know." `https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861`, Aug. 2017. ONLINE; accessed 2021-06-03.

[28] "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition [Book]." ISBN: 9781492032649.

[29] F. Lotte, "Signal Processing Approaches to Minimize or Suppress Calibration Time in Oscillatory Activity-Based Brain–Computer Interfaces," *Proceedings of the IEEE*, vol. 103, pp. 871–890, June 2015.

[30] X. J. Richard E. Neapolitan, "Artificial Intelligence: With an Introduction to Machine Learning."

[31] S. Misra, O. Osogba, and M. Powers, "Unsupervised outlier detection techniques for well logs and geophysical data," in *Machine Learning for Subsurface Characterization*, pp. 1–37, Elsevier, 2020.

[32] "Predicting Roof Pressures on a Low-Rise Structure From Freestream Turbulence Using Artificial Neural Networks | Fernández-Cabán, Pedro L.; Masters, Forrest J.; Phillips, Brian M. | download."

[33] "Artificial Intelligence Machine Learning and Deep Learning | Oswald Campesato."

[34] M. Venkatachalam, "Recurrent Neural Networks." `https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce`, June 2019. ONLINE; accessed 2021-06-03.

[35] "What is the Difference Between CNN and RNN?." `https://lionbridge.ai/articles/difference-between-cnn-and-rnn`, Mar. 2020. ONLINE; accessed 2021-06-03.

[36] A. Shrestha and A. Mahmood, "Review of Deep Learning Algorithms and Architectures," *IEEE Access*, vol. PP, pp. 1–1, Apr. 2019.

[37] Feng, Jie, Feng, Xueliang, Chen, Jiantong, Cao, Xianghai, Zhang, Xiangrong, Jiao, Licheng, and Yu, Tao, "Generative Adversarial Networks Based on Collaborative Learning and Attention Mechanism for Hyperspectral Image Classification," vol. 12, Apr. 2020.

[38] "CS 230 - Deep Learning Tips and Tricks Cheatsheet." ONLINE; accessed 2021-06-03.

[39] A. S. Lundervold and A. Lundervold, "An overview of deep learning in medical imaging focusing on MRI," *Zeitschrift für Medizinische Physik*, vol. 29, pp. 102–127, May 2019.

[40] S. Albawi, S. ALZAWI, O. Ucan, and O. Bayat, *SOCIAL TOUCH GESTURE RECOGNITION USING DEEP NEURAL NETWORK*. PhD thesis, July 2018.

[41] D. S. Kevin Zhou, Hayit Greenspan, "Deep Learning for Medical Image Analysis - 1st Edition."

[42] S. Albawi, T. Abed Mohammed, and S. ALZAWI, *Understanding of a Convolutional Neural Network*. Aug. 2017.

[43] *Ragav Venkatesan, Baoxin Li | Convolutional Neural Networks in Visual Computing: A Concise Guide*.

[44] S. Romanazzi, *BolognaNN - Photo Geolocation in Bologna with Convolutional Neural Networks*. May 2018.

[45] S. Khan, H. Rahmani, S. A. A. Shah, and M. Bennamoun, "A Guide to Convolutional Neural Networks for Computer Vision," *Synthesis Lectures on Computer Vision*, vol. 8, pp. 1–207, Feb. 2018.

[46] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv:1603.07285 [cs, stat]*, Jan. 2018. arXiv: 1603.07285.

[47] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Jan. 2017. arXiv: 1412.6980.

[48] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, July 2011.

[49] X. Liu, L. Song, S. Liu, and Y. Zhang, "A Review of Deep-Learning-Based Medical Image Segmentation Methods," *Sustainability*, vol. 13, p. 1224, Jan. 2021.

[50] A. Elnakib, G. Gimel'farb, J. Suri, and A. El-Baz, "Medical Image Segmentation: A Brief Survey," in *Multi Modality State-of-the-Art Medical Image Segmentation and Registration Methodologies*, pp. 1–39, Apr. 2011. Journal Abbreviation: Multi Modality State-of-the-Art Medical Image Segmentation and Registration Methodologies.

[51] N. Aldeborgh, "Automatic Brain Tumor Segmentation," July 2021. original-date: 2016-04-06T20:41:19Z.

[52] T. Magadza and S. Viriri, "Deep Learning for Brain Tumor Segmentation: A Survey of State-of-the-Art," *Journal of Imaging*, vol. 7, p. 19, Jan. 2021.

[53] A. Işın, C. Direkoglu, and M. Sah, "Review of MRI-based Brain Tumor Image Segmentation Using Deep Learning Methods," *Procedia Computer Science*, vol. 102, pp. 317–324, Dec. 2016.

[54] L. Chen, P. Bentley, K. Mori, K. Misawa, M. Fujiwara, and D. Rueckert, "DRINet for Medical Image Segmentation," *IEEE Transactions on Medical Imaging*, vol. PP, pp. 1–1, May 2018.

[55] B. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahaniy, J. Kirby, Y. Burren, N. Porz, J. Slotboomy, R. Wiest, L. Lancziy, E. Gerstnery, M.-A. Webery, T. Arbel, B. Avants, N. Ayache, P. Buendia, L. Collins, N. Cordier, and K. Van Leemput, "The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)," *IEEE Transactions on Medical Imaging*, vol. 99, Dec. 2014.

[56] S. Asgari Taghanaki, K. Abhishek, J. P. Cohen, J. Cohen-Adad, and G. Hamarneh, "Deep semantic segmentation of natural and medical images: a review," *Artificial Intelligence Review*, vol. 54, pp. 137–178, Jan. 2021.

[57] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," *arXiv:1409.4842 [cs]*, Sept. 2014. arXiv: 1409.4842.

[58] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv:1409.1556 [cs]*, Apr. 2015. arXiv: 1409.1556.

[59] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Neural Information Processing Systems*, vol. 25, Jan. 2012.

[60] Y. Lian and Z. Song, "Automated brain tumor segmentation in magnetic resonance imaging based on sliding-window technique and symmetry analysis," *Chinese Medical Journal*, vol. 127, no. 3, pp. 462–468, 2014.

[61] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *arXiv:1505.04597 [cs]*, May 2015. arXiv: 1505.04597.

[62] L. Borne, D. Rivière, M. Mancip, and J.-F. Mangin, "Automatic labeling of cortical sulci using patch- or CNN-based segmentation techniques combined with bottom-up geometric constraints," *Medical Image Analysis*, vol. 62, p. 101651, Feb. 2020.

[63] G. Wang, W. Li, S. Ourselin, and T. Vercauteren, "Automatic Brain Tumor Segmentation using Cascaded Anisotropic Convolutional Neural Networks," *arXiv:1709.00382 [cs]*, vol. 10670, pp. 178–190, 2018. arXiv: 1709.00382.

[64] A. Casamitjana, M. Catà, I. Sanchez, M. Combalia, and V. Vilaplana, *Cascaded V-Net using ROI masks for brain tumor segmentation.* Dec. 2018.

[65] A. Myronenko, "3D MRI brain tumor segmentation using autoencoder regularization," *arXiv:1810.11654 [cs, q-bio]*, Nov. 2018. arXiv: 1810.11654.

[66] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385 [cs]*, Dec. 2015. arXiv: 1512.03385.

[67] K. Kamnitsas, W. Bai, E. Ferrante, S. McDonagh, M. Sinclair, N. Pawlowski, M. Rajchl, M. Lee, B. Kainz, D. Rueckert, and B. Glocker, "Ensembles of Multiple Models and Architectures for Robust Brain Tumour Segmentation," *arXiv:1711.01468 [cs]*, Nov. 2017. arXiv: 1711.01468.

[68] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440, June 2015. ISSN: 1063-6919.

[69] K. Kamnitsas, C. Ledig, V. F. J. Newcombe, J. P. Simpson, A. D. Kane, D. K. Menon, D. Rueckert, and B. Glocker, "Efficient Multi-Scale 3D CNN with Fully Connected CRF for Accurate Brain Lesion Segmentation," *Medical Image Analysis*, vol. 36, pp. 61–78, Feb. 2017. arXiv: 1603.05959.

[70] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution,

and Fully Connected CRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, June 2016.

[71] D. Nie, L. Wang, E. Adeli, C. Lao, W. Lin, and D. Shen, "3-D Fully Convolutional Networks for Multimodal Isointense Infant Brain Image Segmentation," *IEEE Transactions on Cybernetics*, vol. PP, pp. 1–14, Feb. 2018.

[72] S. Wang, L. Yi, Q. Chen, Z. Meng, H. Dong, and Z. He, "Edge-aware Fully Convolutional Network with CRF-RNN Layer for Hippocampus Segmentation," pp. 803–806, May 2019.

[73] Z. Jiang, C. Ding, M. Liu, and D. Tao, "Two-Stage Cascaded U-Net: 1st Place Solution to BraTS Challenge 2019 Segmentation Task," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries* (A. Crimi and S. Bakas, eds.), Lecture Notes in Computer Science, (Cham), pp. 231–241, Springer International Publishing, 2020.

[74] F. Isensee, P. F. Jaeger, P. M. Full, P. Vollmuth, and K. H. Maier-Hein, "nnU-Net for Brain Tumor Segmentation," *arXiv:2011.00848 [cs, eess]*, Nov. 2020. arXiv: 2011.00848.

[75] P. Chang, E. Kuoy, J. Grinband, B. Weinberg, M. Thompson, R. Homo, J. Chen, H. Abcede, M. Shafie, L. Sugrue, C. Filippi, M.-Y. Su, W. Yu, C. Hess, and D. Chow, "Hybrid 3D/2D Convolutional Neural Network for Hemorrhage Evaluation on Head CT," *American Journal of Neuroradiology*, vol. 39, July 2018.

[76] Y. Xue, T. Xu, H. Zhang, R. Long, and X. Huang, "SegAN: Adversarial Network with Multi-scale L_1 Loss for Medical Image Segmentation," *Neuroinformatics*, vol. 16, Oct. 2018.

[77] P. Moeskops, M. Veta, M. Lafarge, K. Eppenhof, and J. Pluim, "Adversarial Training and Dilated Convolutions for Brain MRI Segmentation," pp. 56–64, Sept. 2017.

[78] M. Rezaei, K. Harmuth, W. Gierke, T. Kellermeier, M. Fischer, H. Yang, and C. Meinel, "A Conditional Adversarial Network for Semantic Segmentation of Brain Tumor," pp. 241–252, Sept. 2018.

[79] E. Giacomello, D. Loiacono, and L. Mainardi, "Brain MRI Tumor Segmentation with Adversarial Networks," *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, July 2020. arXiv: 1910.02717 version: 2.

[80] G. Wang, W. Li, S. Ourselin, and T. Vercauteren, "Automatic Brain Tumor Segmentation Based on Cascaded Convolutional Neural Networks With Uncertainty Estimation," *Frontiers in Computational Neuroscience*, vol. 13, 2019. Publisher: Frontiers.

[81] F. Isensee, P. Kickingereder, W. Wick, M. Bendszus, and K. H. Maier-Hein, "Brain Tumor Segmentation and Radiomics Survival Prediction: Contribution to the BRATS 2017 Challenge," *arXiv:1802.10508 [cs]*, Feb. 2018. arXiv: 1802.10508.

[82] Yang, T., Ou, Y., and Huang, T., "Automatic Segmentation of Brain Tumor from MR Images Using SegNet: Selection of Training Data Sets.," in *the 6th MICCAI BraTS Challenge*, (, Quebec City, QC, Canada), pp. 309–312, Sept. 2017.

[83] F. Isensee, P. Kickingereder, W. Wick, M. Bendszus, and K. H. Maier-Hein, "No New-Net," *arXiv:1809.10483 [cs]*, Jan. 2019. arXiv: 1809.10483.

[84] R. McKinley, R. Meier, and R. Wiest, "Ensembles of Densely-Connected CNNs with Label-Uncertainty for Brain Tumor Segmentation," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries* (A. Crimi, S. Bakas, H. Kuijf, F. Keyvan, M. Reyes, and T. van Walsum, eds.), Lecture Notes in Computer Science, (Cham), pp. 456–465, Springer International Publishing, 2019.

[85] C. Zhou, S. Chen, C. Ding, and D. Tao, "Learning Contextual and Attentive Information for Brain Tumor Segmentation," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries* (A. Crimi, S. Bakas, H. Kuijf, F. Keyvan, M. Reyes, and T. van Walsum, eds.), Lecture Notes in Computer Science, (Cham), pp. 497–507, Springer International Publishing, 2019.

[86] Y.-X. Zhao, Y.-M. Zhang, and C.-L. Liu, "Bag of Tricks for 3D MRI Brain Tumor Segmentation," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries* (A. Crimi and S. Bakas, eds.), Lecture Notes in Computer Science, (Cham), pp. 210–220, Springer International Publishing, 2020.

[87] R. McKinley, M. Rebsamen, R. Meier, and R. Wiest, "Triplanar Ensemble of 3D-to-2D CNNs with Label-Uncertainty for Brain Tumor Segmentation," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries* (A. Crimi and S. Bakas, eds.), Lecture Notes in Computer Science, (Cham), pp. 379–387, Springer International Publishing, 2020.

[88] H. Jia, W. Cai, H. Huang, and Y. Xia, "H2NF-Net for Brain Tumor Segmentation using Multimodal MR Imaging: 2nd Place Solution to BraTS Challenge 2020 Segmentation Task," *arXiv:2012.15318 [cs, eess]*, Dec. 2020. arXiv: 2012.15318.

[89] Y. Wang, Y. Zhang, F. Hou, Y. Liu, J. Tian, C. Zhong, Y. Zhang, and Z. He, "Modality-Pairing Learning for Brain Tumor Segmentation," *arXiv:2010.09277 [cs, eess]*, Dec. 2020. arXiv: 2010.09277.

[90] Y. Yuan, "Automatic Brain Tumor Segmentation with Scale Attention Network," *arXiv:2011.03188 [cs, eess]*, Nov. 2020. arXiv: 2011.03188.

[91] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[92] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," vol. 27, 2014.

[93] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," *arXiv:1708.02002 [cs]*, Feb. 2018. arXiv: 1708.02002.

[94] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," *arXiv:1610.02357 [cs]*, Apr. 2017. arXiv: 1610.02357.

[95] P. Yakubovskiy, "Segmentation models." `https://github.com/qubvel/segmentation_models`, 2019.

[96] N. Kimura, I. Yoshinaga, K. Sekijima, I. Azechi, and D. Baba, "Convolutional Neural Network Coupled with a Transfer-Learning Approach for Time-Series Flood Predictions," *Water*, vol. 12, p. 96, Jan. 2020. Number: 1 Publisher: Multidisciplinary Digital Publishing Institute.

[97] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[98] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[99] "Detailed Guide to Understand and Implement ResNets," Sept. 2019. ONLINE; accessed 2021-08-21.

[100] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *arXiv:1512.00567 [cs]*, Dec. 2015. arXiv: 1512.00567.

[101] H. A. Khan, W. Jue, M. Mushtaq, M. U. Mushtaq, H. A. Khan, W. Jue, M. Mushtaq, and M. U. Mushtaq, "Brain tumor classification in MRI image using convolutional neural network," *Mathematical Biosciences and Engineering*, vol. 17, no. 5, pp. 6203–6216, 2020. Cc_license_type: cc_by Primary_atype: Mathematical Biosciences and Engineering Subject_term: Research article Subject_term_id: Research article.

[102] Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation," *arXiv:1606.06650 [cs]*, June 2016. arXiv: 1606.06650.