Université Paris-Est - Ecole Doctorale Mathématiques et Sciences et Technologies de l'Information et de la Communication (MSTIC)

Université Mohamed khider de Biskra

THÈSE EN COTUTELLE

pour l'obtention des diplômes de

Doctorat en Informatique de l'Université Paris-Est
Doctorat en Informatique de l'Université de Biskra
Spécialité: Intelligence Artificielle

Deep Neural Networks for the segmentation and classification in Medical Imaging

---

Par : Mostefa Ben naceur

Soutenue le 17 September 2020 devant le jury composé de :

Mme Carole Lartizien	Directrice de Recherche, CNRS, CREATIS	Rapporteur
M. Mohamed Tayeb Laskri	Professeur à l'Université d'Annaba, LRI	Rapporteur
Mme Rachida Saouli	Professeure à l'Université de Biskra, LINFI	Directeur
M. Mohamed Akil	Professeur émérite, ESIEE Paris, LIGM	Directeur
M. Rostom Kachouri	Professeur associé, ESIEE Paris, LIGM	Invité

In loving memory of my mother Habiba
To my father Mohammed
To my brothers and sisters
To all my family
I dedicate this thesis to them

Acknowledgements

The completion of this dissertation would not have been possible without the encouragement, help and friendship of many individuals. It is my privilege to thank the people who have supported and guided me throughout my pursuit of this thesis.

Foremost, I would like to present my gratitude to Carole Lartizien and Mohamed Tayeb Laskri for doing me the honor to be rapporteurs of my Ph.D thesis.

I would like to express my sincere gratitude to my supervisors Rachida Saouli, Mohamed Akil, and co-supervisor Rostom Kachouri for their continuous support of my Ph.D study and research, for their comments, discussions, and feedbacks. Their guidance helped me throughout my research during all these years of my Ph.D.

I also express my gratitude to the members of the jury who agreed to evaluate the work presented in this manuscript.

Special thanks to my fellow lab mates, friends, and colleagues at the University of Biskra, Computer Science department - Laboratoire d'informatique intelligente (LINFI), University of Paris-Est, and ESIEE Paris - Laboratoire d'Informatique Gaspard-Monge (LIGM) for the motivating discussions during the days we were working together, for their moral support, for their help and friendship, for all the fun we have had during my thesis period.

Finally, I would like to express special thanks to my family, my parents, brothers, and sisters, for supporting me throughout my entire life. Without their unconditional love and encouragement, I would not have finished this thesis. Thank you for everything.

Mostefa

Abstract

Nowadays, getting an efficient segmentation of Glioblastoma Multiforme (GBM) brain tumors in multi-sequence MRI images as soon as possible, gives an early clinical diagnosis, treatment, and follow-up. The MRI technique is designed specifically to provide radiologists with powerful visualization tools to analyze medical images, but the challenge lies more in the information interpretation of radiological images with clinical and pathologies data and their causes in the GBM tumors. This is why quantitative research in neuroimaging often requires anatomical segmentation of the human brain from MRI images for the detection and segmentation of brain tumors. The objective of the thesis is to propose automatic Deep Learning methods for brain tumors segmentation using MRI images.

First, we are mainly interested in the segmentation of patients' MRI images with GBM brain tumors using Deep Learning methods, in particular, Deep Convolutional Neural Networks (DCNN). We propose two end-to-end DCNN-based approaches for fully automatic brain tumor segmentation. The first approach is based on a pixel-wise technique in addition to a new guided optimization algorithm to optimize the suitable hyperparameters while the second one is based on a patch-wise technique. Then, through experiments, we prove that the latter is more efficient in terms of segmentation performance and computational benefits compared to the first approach.

Second, to enhance the segmentation performance of the proposed approaches, we propose new segmentation pipelines of patients' MRI images, where these pipelines are based on deep learned features and two stages of training. We also address problems related to unbalanced data in addition to false positives and false negatives to increase the model segmentation sensitivity towards the tumor regions and specificity towards the healthy regions.

Finally, the segmentation performance and the inference time of the proposed approaches and pipelines are reported along with state-of-the-art methods on a public dataset annotated by radiologists and approved by neuroradiologists.

Keywords— Glioblastomas, MRI images, brain tumor segmentation, Deep Learning, Deep Convolutional Neural Networks, unbalanced data, false positives and false negatives

Résumé

De nos jours, fournir une segmentation précise des tumeurs cérébrales de Glioblastome Multiforme (GBM) à partir d'images IRM multimodales le plus tôt possible, permet de délivrer un diagnostic clinique précoce, pour un traitement et un suivi efficaces. La technique d'imagerie IRM est spécialement conçue pour fournir aux radiologues des outils puissants de visualisation pour analyser des images médicales, mais le challenge réside dans l'interprétation de ces images radiologiques avec les données cliniques et pathologiques, et la cause de ces tumeurs GBM. C'est la raison pour laquelle la recherche quantitative en neuroimagerie nécessite souvent une segmentation anatomique du cerveau humain à partir d'images IRM, afin d'aider à la détection et la segmentation des tumeurs cérébrales. L'objectif de cette thèse est de proposer des méthodes automatisées de Deep Learning pour la segmentation des tumeurs cérébrales à partir d'images IRM.
Dans un premier temps, nous nous intéressons principalement à la segmentation d'images IRM de patients atteints de tumeurs GBM en utilisant le Deep Learning, en particulier, Deep Convolutional Neural Networks (DCNNs). Nous proposons deux approches DCNNs "End-to-End" pour la segmentation automatique des tumeurs cérébrales. La première approche est basée sur la technique pixel-wise et un nouvel algorithme d'optimisation des hypeparamètres tandis que la deuxième approche est basée sur la technique patch-wise. Ensuite, à travers des expérimentations, nous prouvons que la deuxième approche est plus efficace en termes de performance de segmentation et de temps de calcul par rapport à la première approche.
Dans un deuxième temps, pour améliorer les performances de segmentation des approches formulées, nous proposons de nouveaux pipelines de segmentation des images IRM de patients, basés sur des attributs extraits des DCNNs et de deux étapes de training. Nous abordons également les problèmes liés aux données déséquilibrées en plus les faux positifs et les faux négatifs pour augmenter la sensibilité de segmentation vers les régions tumorales et la spécificité de segmentation vers les régions saines.
Finalement, les performances et le temps de segmentation des approches et des pipelines proposés sont rapportés avec les méthodes de l'état de l'art sur une base de données accessible au public, annotée par des radiologues et approuvée par des neuroradiologues.

Mots-clés :—Glioblastome, Images IRM, Segmentation des Tumeurs Cérébrales, Apprentissage Profond, Réseaux de Neurones Convolutifs Profonds, Données Déséquilibrées, Faux Positifs et Faux Négatifs

Contents

Abbreviations

ABTA	American Brain Tumor Association
ANNs	Articial Neural Networks
BRATS	multimodal BRAin Tumor Segmentation challenge
CAD	Computer-Aided Diagnostic
CNNs	Convolutional Neural Networks
CT	Computed Tomography
DCNNs	Deep Convolutional Neural Networks
DNNs	Deep Neural Networks
FC	Fully Connected layer
Flair	T2-weighted FLuid-Attenuated Inversion Recovery
GBM	Glioblastoma Multiform tumors
GCRA	Gadolinium-Containing Radiocontrast Agents
GPU	Graphics Processing Unit
GT	Ground Truth
LR	Logistic Regression
MLP	MultiLayer Perceptron
MRI	Magnetic Resonance Imaging
NBTS	National Brain Tumor Society
OCM	OCcipito Module
OTP	Occipito-Temporal pathway
PCA	Principal Component Analysis
PET	Positron Emission Tomography
ReLU	Rectified Linear Units
RF	Random Forest
SVM	Support Vector Machine
Tanh	Hyperbolic tangent activation function
T1	T1-weighted
T1c	T1-weighted Contrast-enhanced
T2	T2-weighted
WHO	World Health Organization

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 Thesis Context

Cancer cells have a mechanism that does not undergo the growth and proliferation of the human body system. When these cells begin to grow and expand, they infiltrate and invade nearby surrounding tissue. The results of this uncontrolled growth are primary tumors, but the important property of these malignant metastasis cells is that they begin to grow and proliferate outside their primary site to another part of the body. Usually, the cancer cells invasion is toward the brain by blood vessels or the lymphatic system, this process is called Metastasize. Brain tumors are a growing abnormal cell in the brain or central spinal canal [Young and Knopp, 2006]. Moreover, one of the outcomes of cancers in the lung, breast, skin (melanoma), colon or kidney is a brain tumor (ABTA [1]). In the USA, according to the NBTS [2] that estimates every year 13.000 patients die, and 29.000 patients suffer from primary [3] brain tumors [Singh et al., 2012]. In 2007, they expected in the UK, more than 4.200 patients have a brain tumor each year [Logeswari and

1. ABTA: American Brain Tumor Association
2. NBTS: National Brain Tumor Society
3. Primary tumor is a tumor, but it starts in the brain

Karnan, 2009]. These statistical samples are proportional to the population number in many countries in the world. In addition, if we count also the potentiality of medical mistakes besides low survival rates of patients with life-threatening diseases such as Gliomas. Based on the mentioned statistics, we can take a forward-looking snapshot of the global health; the origin of all these diseases in the world in two words: globalization and urbanization; firstly, as the population grows, it is aging. Secondly, according to the WHO[4] classification, the spread of risk factors in the population: tobacco, lifestyle, unhealthy diet and alcohol. Thus, nowadays it is difficult for clinicians and medical practitioners to make a link among these causes and factors with the diagnosed diseases during a session of 30 minutes. To increase survival and diagnosis rates, first we need to increase disease awareness campaigns and intensive prevention programs that represent the first lines of defense against disease, they can also improve early detection and prediction of diseases. Second, we need to get help from technology and machines currently present, especially in the field of artificial intelligence and deep learning in order to assist radiologists and oncologists in the detection and diagnosis of diseases.

1.2 Thesis Motivation and Objectives

In this thesis, we are interested in the segmentation of patients' MRI images with Glioblastoma Multiforme (GBM) brain tumors. GBM are the most frequent primary brain tumors in adults [Holland, 2001]. According to the WHO classification [Louis et al., 2016; Gupta and Dwivedi, 2017], Glioma brain tumors have 4 grades (I, II, III, IV) where the GBM tumors are the grade IV of Glioma tumors. Moreover, GBM affect children between 5 and 10 years old [Goodman and Fuller, 2014], and adults between 30 and 40 years old [Schneider et al., 2010]. What makes the diagnosis of GBM very challenging is that each patient has a different health condition, age, gender, along with the property of the GBM makes these tumors appear anywhere in the brain. With this large number of patients worldwide and a massive medical data produced from different medical imaging techniques such as magnetic resonance imaging (MRI), positron emission tomography (PET), computed tomography (CT), emerged the need to discover new techniques to overcome the limits of conventional methods of diagnosis and treatment of patients. In

4. WHO: World Health Organization

the last decades, we noticed that the number of published papers involving brain tumor segmentation increased exponentially [Menze et al., 2015], this increase indicates that creating a Computer-Aided Diagnosis (CAD) [Akram and Usman, 2011; El-Dahshan et al., 2014] has always been a highly needed option and necessary [Zhang et al., 2001]. In addition, some patients have aggressive tumors (e.g. GBM) that need to be treated in less than two months. This CAD system can decrease the needed time for the diagnosis process [El-Dahshan et al., 2014] and it can give to oncologists more time with their patients in treatment and follow-up procedures during the therapy planning process. This process generally begins with multimodal MRI image generation and acquisition, segmentation, diagnosis, treatment and follow-up. This 5-phase procedure takes at least 2 months and approximately 5% of patients remain under this procedure for 5 years [Ostrom et al., 2014]. Moreover, the manual segmentation is a time-consuming procedure, where the radiologist takes between 3 and 5 hours [Kaus et al., 2001] for each patient. The automation of the radiologist's mission, given the time of manual segmentation, indicates that automatic image interpretation and analysis through a CAD system can have a significant impact on the patient's diagnosis.

MRI is the investigation tool of choice for the segmentation of GBM brain tumors, and for the evaluation of treatment response [Bangiyev et al., 2014]. Usually, an expert radiologist uses MRI as the most effective technique [Akram and Usman, 2011; Bhandarkar and Nammalwar, 2001a] to generate Multi-modal images to identify different tumor regions in the soft tissues. The use of MRI images to visualize the brain, has become the standard protocol and routine in the therapy planning process of patients with brain tumors in recent decades. In general, radiologists generate four standard MRI images modalities for GBM diagnosis (See figure 1.1): T2-weighted fluid-attenuated inversion recovery (Flair), T1-weighted (T1), T1-weighted contrast-enhanced (T1c), and T2-weighted (T2) for each patient. The brain tumor segmentation procedure involves radiologists and oncologists in a process of patients' MRI images interpretation. In this procedure, they need to segment each region of GBM tumors pixel-by-pixel in each slice until the 3D (height, width and depth) brain volume is divided into meaningful regions (See figure 1.1 (GT)). After that, this 3D segmentation image will be used for diagnosis of patients' organs and assess its risk, treatment sessions, surgery planning, and follow-up to see if the tumor is growing or shrinking.

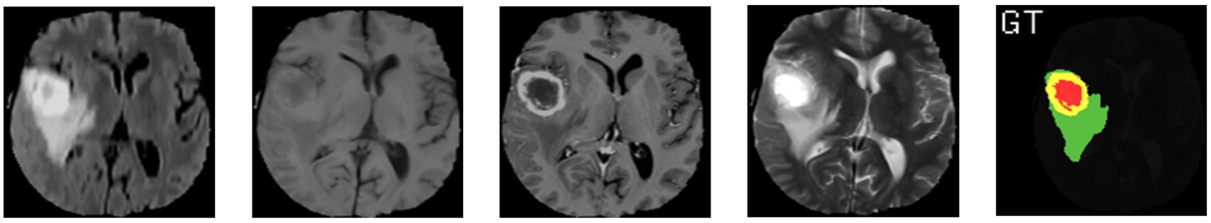


Figure 1.1 – The four multi-sequence MR images of patients with Glioblastomas tumors. From left to right: Flair, T1, T1c, T2, Ground Truth (GT) labels. The color is used to distinguish between the tumor regions: red: Necrotic and Non-Enhancing tumor, green: Peritumoral Edema, yellow: Enhancing tumor, black: Healthy tissue and background.

Medical image analysis is a set of successive operations and computational analysis methods to extract relevant features within the medical images; in this case, segmentation of GBM brain tumors. The segmentation step is a process of dividing a medical image of the brain into significant regions, where each region represents various structures such as organs, tissue classes or pathologies. The idea is to use the extracted diagnostic features from medical images such as shape, texture, size, pixels' intensities or a combination of them to (1) delineating the tumor region and its sub-regions and discriminating them from the background and healthy tissue and to (2) obtaining clinically relevant information and knowledge such as the target volume and the tumor volume (See figure 1.2) that are the most important information to establish treatment that usually include radiotherapy planning of patients with brain tumors [Cheng et al., 2015]. Furthermore, determining (1) and (2) helps clinicians to delimit the radiation doses in the radiotherapy planning process, which is a manual work that depends considerably on their experience in analyzing medical images. Moreover, determining radiation doses is critical and crucial as they can affect adjacent normal tissues, where the use of a higher dose of radiation has aggressive consequences on patients after radiotherapy treatment, among these consequences: higher incidence of fatigue or malaise, insomnia, and poor emotional functioning [DeAngelis, 2001]. For radiologists and oncologists, the radiation dose quantity is very important and it depends significantly, firstly, on the manual segmentation, where in many cases, we obtain a remarkable variability in manual segmentation of brain tumors: intra-rater, $20 \pm 15\%$; inter-rater, $28 \pm 12\%$ [Mazzara et al., 2004]. Secondly, on the information interpretation of radiological images with clinical and pathologies data. Consequently, we can conclude that obtaining an accurate segmentation of GBM brain tumors is crucial for optimal radiotherapy planning and for determining organs at risk, and it makes a difference between life and death for patients under treatment.

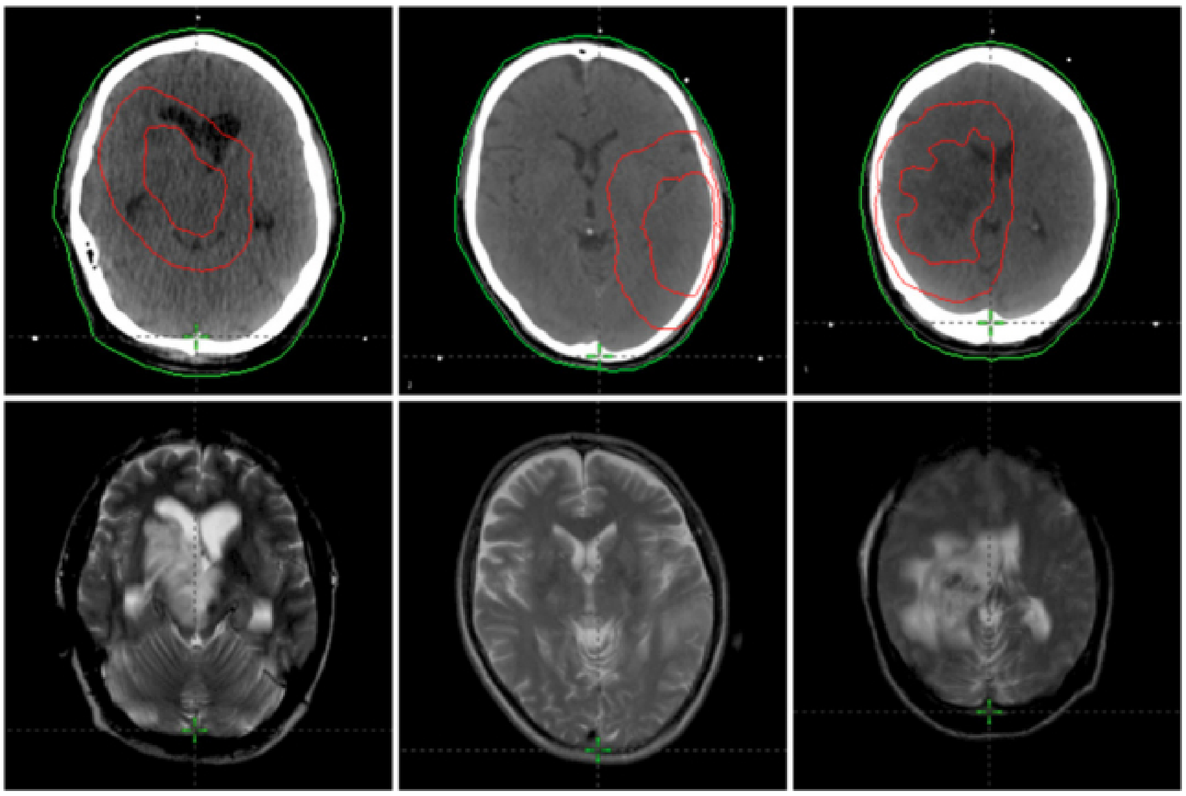


Figure 1.2 – Example of Glioma brain tumors, which are registred on CT (top) images and on T2-weighted axial MRI (Bottom) images. The body contour (green) on CT images is used for radiotherapy planning. The outer red contour is the target volume, and and the inner red contour is the tumor volume [Cheng et al., 2015].

In this thesis, we have been motivated by the success of Deep Learning in computer vision and in the context of brain tumor classification and segmentation. After the breakthrough in 2012 in the field of computer vision where a team developed a deep learning model called AlexNet [Krizhevsky et al., 2012], this model outperformed the state-of-the-art methods and it achieved the best results in the field of object recognition. Since 2014, many deep learning-based research [Pereira et al., 2015; Havaei et al., 2017] have been proposed in the field of brain tumor segmentation.

Current state-of-the-art image segmentation in the field of Deep Learning are based on Convolutional Neural Networks (CNNs) [LeCun et al., 1998]. In typical CNNs architectures [Krizhevsky et al., 2012; Simonyan and Zisserman, 2014], we usually find a feature extractor with a bank of convolution filters (i.e., trainable parameters). Then pooling layers to make the images less sensitive and invariant to small translations in addition to reducing the feature maps dimensionality. The last stage in CNNs architectures, is a classifier which classifies each pixel (or voxel) into one of many classes. The success of Deep Learning and CNNs architectures, is due to their operations (e.g., Convolution, Pooling,...etc.) that can be parallelized to take advantage of massive parallel architectures such as GPU and TPU implementations, also due to the large public datasets and to the improvement of learning methods. Moreover, most of the gains in machine learning and deep learning come from extracting great and relevant features, where CNNs algorithm is considered as the best features extraction algorithm in computer vision (See figure 3.1 and figure 5.3)

Finally, despite the success of Deep Learning and CNNs in the field of computer vision and brain tumor segmentation, but they still have limitations, in particular, how to design dedicated and efficient architectures in terms of computational cost and segmentation performance. Another problem known in CNNs is the combinatorial explosion of choices which are due to the large number of hyperparameters (Patches, feature maps, kernels, strides, activation functions, connectivity between layers,...etc), where these choices limit and affect the robustness of CNNs. In addition, the choice of the input shape is important to obtain high results in terms of segmentation performance. The problem of input shape is due to the fact that conventional CNNs do not take into account the global context of the image. Moreover, the most common problem among image segmentation methods is unbalanced data, where we find a class or a label of interest has the minority of data compared to other classes. This kind of problems makes Articial Neural Networks (ANNs), including CNNs, bias toward the more frequent label. Thus, training a CNNs model with such kind of data, will make predictions with low sensitivity, where the most important part in medical applications is to make the model more sensitive toward the lesion-class, i.e. tumoral regions. Another important issue is the performance degradation due to many factores: false positive and false negative regions, vanishing gradient and overfitting. The objective of this thesis is to overcome these limitations: the impact of hyperparameters on the segmentation performance, reducing the computational cost without affecting the performance, the issue of conventional CNNs with the global context, training CNNs models with Unbalanced data, and finally the issue of performance degradation.

1.3 Thesis Contributions

In this thesis, we aim to propose automatic, efficient, and accurate Deep Learning pipelines for brain tumors segmentation using MRI images. The proposed pipelines are used to segment the brain tumors of Glioblastomas with both high- and low-grade. Moreover, these pipelines could assist clinicians and radiologists to provide better therapeutic monitoring of patients affected by Glioblastoma multiforme. For achieving this goal, our contributions are divided into fourfold:

1. We propose two End-to-End DCNN-based approaches for fully automatic segmentation of GBM brain tumors. The first approach is an "Incremental XCNet" algorithm which is a pixel-wise-based technique. The proposed algorithm generates DCNNs architecture, i.e. from a base model (a limited number of layers) we obtain deeper and scalable models. The second approach is based on an OCM module which is an inspired and a patch-wise-based technique, where through this inspired model, we improved the second approach to provide better results in terms of segmentation performance and computational benefits. Then, to enhance the segmentation performance of DCNN architecture, we propose two steps of post-processing based on connected-components and morphological opening operator.
2. We propose a new methodology to train Deep Learning architectures, where the proposed method called "ELOBA" which is a supervised learning algorithm for optimizing an objective function. This method takes into account the most influencing hyperparameters (The number of epochs, learning rate, optimizer, batch size). The algorithm of "ELOBA" bounds and sets a roof to these hyperparameters to fast the training of the generated DCNNs architecture. In particular, we address problems related to vanishing gradient and computational costs.
3. To overcome the issue of unbalanced data, we study the impact of Cross-Entropy loss function on the segmentation results of the DCNNs architectures for the problem of brain tumor segmentation. We then propose a modified, adaptive and more robust version of this loss function called Online Class-Weighting, where our proposed function and after each training epoch provides a set of weights for a multiple number of classes. Moreover, to demonstrate the performance of the Online Class-

Weighting loss function, we evaluate our proposition within a case study applied to a fully automatic brain tumor segmentation of Highly Unbalanced Glioblastoma tumors.

4. To address the limits of Deep Learning architectures, in particular, false positives and false negatives, resulting from the fact that in multi-classification issue, the classifier *Softmax* function is simple and does not take into account any *prior knowledge*. Thus, we propose new pipelines and two phases of training to boost the prediction of GBM tumoral regions. These pipelines are based on 3 stages. In the first stage, we develop DCNNs architecture, then in the second stage we extract multi-dimensional features from the higher-resolution representation of DCNNs, in the third stage, we develop machine learning algorithms. We then feed the extracted features from DCNNs into different algorithms such as Random Forest (RF) and Logistic Regression (LR), and Principal Component Analysis with Support Vector Machine (PCA-SVM).

1.4 Thesis Overview

In this thesis, we are interested in the segmentation of brain tumors using DCNNs architectures trained on MRI images, to assist the radiologists and oncologists in the therapy planning process, in particular, the segmentation of patients' MRI images with GBM brain tumors. The thesis is organized as follows:

Chapter 2: we present a detailed state-of-the-art in the field of MRI image segmentation. Moreover, a description of the most challenging issues in the medical field, in particular, challenges in the brain tumor segmentation, and in the segmetation of MRI images of patients with Glioblastoma multiform tumors. Finally, a description and an illustration of Glioblastoma multiform tumors anatomy and the tumoral regions that need to be segmented. Also, we study BRATS challenge dataset [Menze et al., 2014; Bakas et al., 2017] of patients with Glioblastoma (high- and-low grade), in addition to the evaluation metrics to measure the segmentation performance and the computational benefits of our proposed pipelines.

Chapter 3: is devoted to a brief introduction to Deep Learning, in particular, Convolutional Neural Networks, in addition to the fundamental building blocks of CNNs algorithm, and the operations behind this algorithm such as convolution, non-linear activation functions, pooling, upsampling, fully-connected layers, forward, and backward propagation. Also, we present the difficulties, limitations and the motivation to apply CNNs algorithms for the context of segmentation of patients' MRI images with Glioblastoma multiform brain tumors.

Chapter 4: we present two approaches based on Deep Convolutional Neural Networks (DCNNs) for the issue of MRI image segmentation, where these approaches are dedicated to patients with Glioblastoma brain tumors. In the first approach, we propose a DCNNs-based pixel-wise approach for the segmentation of Glioblastoma brain tumors. The proposed DCNNs model is an End-to-End incremental DCNNs model for fully automatic brain tumor segmentation. Moreover, a description of the challenging issues to apply DCNNs for medical images, in particular, getting the best hyperparameters for each DCNNs model, and the issue of vanishing gradient. Moreover, we adopt an Ensemble learning technique to design a more efficient architecture. And for solving the problem of training CNNs architectures, we propose a new training strategy which takes into account the most influencing hyperparameters by bounding and setting a roof to these hyperparameters to accelerate the training phase. The second approach is dedicated to solve two important problems: reducing the computational complexity and increasing segmentation performance, thus we have introduced some improvements to the first approach by introducing the patch-wise approach instead of pixel-wise approach, also we propose an overlapping patches technique to solve the issue of conventional DCNNs architectures with the global context.

Chapter 5: in order to solve the issue of training DCNNs models with unbalanced data, where this kind of issues makes DCNNs bias toward the more frequent label. Hence, training a DCNNs model with such kind of data (unbalanced data) will make predictions with low sensitivity. Moreover, it is important in medical applications to make the model more sensitive toward the lesion class, i.e. tumoral regions. The second issue is the misclassification regions that are produced by the proposed approaches. To overcome this issue (i.e. misclassified regions), we present novel pipelines specifically to reduce the false positives and false negatives regions. These pipelines are based on Deep learned features and machine learning algorithms; in the first stage, we train DCNNs architecture, then in the second stage we train these algorithms with the trained and extracted features in the first stage. These pipelines show a high segmentation performance, in addition to their ability to reduce the misclassification tumoral regions.

Finally, our conclusions, summary of the contributions and directions for future research are presented in **Chapter 6**.

Chapter 2

Brain Tumor Segmentation

Contents

2.1 Introduction

In this chapter, we present firstly state-of-the-art machine learning and Deep learning methods in the field of MRI image segmentation, in particular, Glioblastoma brain tumors and its properties and the ways of diagnosis and assessment of patients with these diseases. Secondly, we explain the used dataset in this study, which is a Glioblastoma public dataset called BRATS dataset. Thirdly, we discuss the limitation of MRI imaging as a technique to assist radiologists and oncologists to diagnose patients in the process of radiotherapy planning and for determining organs at risk. Finally, we present different evaluation metrics to measure the segmentation performance, the computational benefits and to compare several state-of-the-art methods.

2.2 Survey on MRI image segmentation

Most research on Pattern Recognition, Computer Vision and Machine Learning from 1970s to 1990s, were based on low-level image operations and mathematical modeling [Litjens et al., 2017]. At this time, researchers generally use a few examples data to build a base of knowledge and rules, i.e., a set of if-then-else statements, what was known after that as expert systems such as MYCIN [Shortliffe et al., 1975; Gordon and Shortliffe, 1984]. Then, the research after 1990s jumped to develop systems based on supervised learning algorithms, features extraction (i.e., vector of binary or real values) and statistical models [Litjens et al., 2017]. In this era, researches use features engineering techniques (i.e., hand-designed features) to extract discriminant features from training MRI images to represent and classify each image as for example malignant or benign, or to detect each object or region inside the image. Thus, the use of discriminative models such as SVM [Cortes and Vapnik, 1995] and feature engineering became popular, especially in medical image analysis. Example include the use of large number of training data to extract several features then by using a classifier or a model at the end of the pipeline to distinguish between the different classes or groups. The issue of features engineering is that it requires high domain knowledge of the given task, therefore, for each task, we need people with specialized knowledge, besides, in many cases, the process of gathering data is expensive such as medical data. The annotation of medical data requires collaboration between different people from different backgrounds such as radiologists, oncologists...etc. All these reasons have contributed and pushed the academic research and the industry to explore and discover more practical and less expensive methods, and here the era of machine learning and deep learning comes in. Thus, the research has transitioned from feature engineering to designing networks with many layers to extract features. This perception allows us to see machine learning and deep learning methods as an automation of feature engineering to advance research and to reduce the computational cost. The following sections will give an overview of classical and modern approaches, presenting in the first section five approaches, and discussing their differences and challenges. In the second section, we discuss modern approaches, in particular, methods-based deep learning. In the last section, a summary and the most challenging issues of deep learning methods with GBM brain tumor segmentation.

2.2.1 Classical approaches

Here, we present the most popular classical MRI image segmentation approaches:

— **Threshold-based methods** [Gibbs et al., 1996; Stadlbauer et al., 2004]: these methods use one type of MRI images, usually a MRI image with more contrast (e.g., T_1 or post-contrast T_1 weighted). They rely on a high-intensity signal as a threshold value in MRI images to extract relevant tissue that is classified after that as a tumor class or healthy tissue class. These methods are iterative, in which they apply a global threshold, local or an adaptive threshold to distinguish between all different tissues for many iterations. These methods are simple and computationally efficient but they have a lot of drawbacks and generally, they fail in real applications. Among these drawbacks: they are sensitive to noise, need a user-interaction, applicable more to binary segmentation issues; so they do not scale to more complicated and multi-classification issues.

— **Region-based methods** [Kaus et al., 2001; Letteboer et al., 2004; Cates et al., 2005]: these methods are based on the technique of subdivision or composition of an image into homogenous regions, where each region has a set of connected pixels. These methods deal with pixel-level and they apply two metrics. First, homogeneity criteria-based metrics to connect all candidate pixels. Second, discontinuity criteria-based metrics to find the boundaries among different regions. These methods apply a repetitive algorithm of composition or of split and merge until constituting a uniform region (e.g., tumor region).

— **Edge-based methods** [Caselles et al., 1993; Lefohn et al., 2003; Cates et al., 2004]: edge detection technique is a very important step in image processing and computer vision fields. Moreover, edge detection in a MRI image helps to extract and reduce useful information which in its turn aid to apply image analysis techniques. Edges correspond to object or region boundaries. At the pixel level, the edge is where we can see a significant change between two or more neighboring pixel values. We can group edge detection techniques into two categories: gradient-based methods and laplacian-based methods. The first category relies on the maximum and minimum in the first derivative such as Prewitt, Sobel, Roberts operators while the second category relies on zero-crossings in the second derivative.

— **Atlas-based methods** [Moon et al., 2002; Prastawa et al., 2003; Menze et al., 2010]: these methods deal with more global information extracted from an image; they attempt to segment a MRI image with no well-known relation between regions and pixels' intensities. Usually, the process of Atlas-based segmentation methods involves many stages, the most important ones are image registration and Atlas construction. Atlas refers to a template or a model, where for each different application, we construct a different template. Some simple applications use a single template while in other we need to use multiple templates; for each population of images, they construct an Atlas (e.g., an Atlas for a healthy population and an Atlas for a diseased population).

— **Classification and Clustering methods** [Ozkan et al., 1993; Clark et al., 1998; Bhandarkar and Nammalwar, 2001b; Fletcher-Heath et al., 2001; Geremia et al., 2011]: these methods are a subpart of machine learning methods. Classification is a supervised learning algorithm while clustering is an unsupervised learning algorithm. Classification methods uses training dataset (images and labels) to minimize or maximize an objective function (loss function) such as SVM algorithm with radial basis function kernel which tries through an objective function to maximize the margin between two diffirent classes. Thus, SVM algorithm classifies each pixel to one of the predefined numbers of classes. Clustering methods is an iterative algorithm that tries to subdivide training images into several disjoint clusters such as K-means algorithm that works by partitioning an image into a number of centroids, where each one represents the center of a cluster. Thus, the K-means algorithm tries to label each pixel by assigning it to one centroid among the predefined number of centroids, i.e., the number of centroids is the number of classes.

2.2.2 Modern approaches

In 2006, appeared a new type of learning algorithms called Deep Neural Networks (DNNs) [Hinton and Salakhutdinov, 2006; Hinton et al., 2006; Bengio et al., 2007] which use a large number of data to extract many lower-level features such as lines, edges with different orientations then it combines them in a hierarchical way to obtain higher-level features such as shapes, objects and faces,...etc. In this era, two DNNs algorithms became increasingly popular: Stacked Auto-Encoders and Deep Belief Networks[1] [Hinton et al., 2006; Lee et al., 2009]. Those algorithms have solved the issue of training large and deep architecture of DNNs, but the training is relatively slow [Zeiler et al., 2011]. In 2014, the research started to use different variants of DNNs architectures in medical image analysis, in particular, tumor and lesion segmentation.

[Zikic et al., 2014] proposed a CNNs model for the segmentation of GBM brain tumors using MRI images. The input of their algorithm is a (4*19*19) 2D patch (i.e., four channels: T1, T2, T1c, and FLAIR). Their CNNs network is used to segment the MRI images to five classes: non-tumor, necrosis, edema, non-enhancing tumor and enhancing tumor. Moreover, their CNNs model is a sequential network contains 5 layers.

Another work used CNNs introduced by [Urban et al., 2014] for GBM brain tumor segmentation, where their approach is different. They used a sequential 3D-CNNs model, in which the input data are 3D patches of size (9*9*9) voxels with four channels (i.e., T1, T1c, T2, FLAIR). Moreover, as known in several CNNs models, they did not use the pooling layers to reduce the size of feature maps and therefore, to reduce the computational costs. Moreover, they used an additional post-processing step, where they removed all regions of less than 3000 voxels. In addition, this method takes one minute to segment the whole brain using a GPU implementation.

[Axel et al., 2014] proposed a CNNs architecture for the segmentation of GBM brain images. The input data of their network is 2D patches of size (32*32) pixels. The network is divided into two parallel pathways; the first pathway is a CNNs architecture (i.e., two Maxout convolutional layers -Maxout is the result of merging two or more feature maps-

1. Deep Belief Networks consist of layers of Restricted Boltzmann Machines

), and the second pathway is a fully connected Maxout layer, then these two pathways are concatenated at the end into a softmax layer. Moreover, this architecture takes 20 minutes to segment the whole brain using a GPU implementation.

[Pereira et al., 2015] developed a method based on CNNs for GBM brain tumor segmentation. They used two CNNs architectures for each type of Glioblastomas (i.e., High-Grade HGG and Low-Grade LGG). Their method takes as inputs 2D patches of size (33*33) pixels with four channels (i.e., four MRI sequences: T1, T2, T1c, and FLAIR). The last implemented step in their algorithms is a post-processing method, they applied a morphological filter to delete isolated regions. In addition, this model takes 10 minutes to segment the complete brain using a GPU implementation.

[Havaei et al., 2017] proposed a fully automatic brain tumor segmentation method based on Cascaded Convolutional Neural Networks which are an extended version of [Axel et al., 2014]. These Cascaded Networks used two pathways that are trained in different phases to capture local and global features. They also used as inputs 2D Axial patches with four MR sequences as channels (i.e., T1, T2, T1c, and FLAIR) where each pathway has a different input patch size. Moreover, they proposed two stages of training for the problem of class-imbalance to correct in the second training stage the patches that are biased toward the wrong class. After that they applied a threshold technique as a post-processing method to remove the connected-components near to the skull. In addition, this Cascaded architecture takes 180 seconds to segment the complete brain using a GPU implementation.

[Chang et al., 2016] developed a CNNs model that is based on two concepts (1) a fully convolutional architecture that predicts a dense output matrix size as used in the original input [Long et al., 2015], (2) Hyperlocal features concatenation; the input MR images is re-introduced in the concatenation layer before the output. This technique is used first by [Yang and Ramanan, 2015] in their architecture Directed-Acyclic-Graph which is a new variant of standard CNNs [LeCun et al., 1998]. The architecture of [Chang et al., 2016] has 7 convolution layers in addition to upsampling and concatenation layers. In their CNNs architecture, 4 channels of MRI images are used as inputs (i.e., FLAIR, T2, T1c and T1). Moreover, this architecture takes 0.93 seconds to segment the entire brain using a GPU implementation.

[Ellwaa et al., 2016] proposed an iterative method which is based on a random forest with 100 trees, each of which has depth 45. Their method extracts 328 features from MRI images. These features are: gradient features, appearance features, and context aware features. The input to their method is 4 channels MRI images (i.e., FLAIR, T1, T1c and T2). This iterative method works by choosing in each iteration 5 patients' MRI images, then they add these images to the training set, after that they continue the training of this random forest until the training set reaches 50 patients, where at 50 patients their iterative method stops.

[Kamnitsas et al., 2016, 2017] developed a 3D-CNNs model for GBM brain tumor segmentation based on the model's performance of [Urban et al., 2014]. These 3D-CNNs networks composed of dual pathway with 11 layers, the input to this network is 3D MRI images (i.e., FLAIR, T1, T1c and T2). Also, each pathway has a different input patch size (i.e., 4 * 253, 4 * 193). Then, they added conditional random field as a post-processing operation to remove misclassification regions and as a spatial regularization. Moreover, they extended their CNNs network with residual connections [He et al., 2016], in which this new extended network [Kamnitsas et al., 2016] did not obtain a big improvement compared to the original model [Kamnitsas et al., 2017]. Their 3D CNNs model takes 24 hours for training using a GPU implementation, and for testing it takes 35 seconds to segment the entire brain.

[Zhao et al., 2018] developed a GBM brain tumor segmentation method based on the integration of CNNs and conditional random field in one network, as opposed to [Kamnitsas et al., 2017] who used conditional random field as a post-processing step. The authors of [Zhao et al., 2018] developed 3 CNNs networks that take as an input 3 types of MRI images (i.e., FLAIR, T1c and T2). Each of these 3 networks use two pathways similar to [Havaei et al., 2017; Kamnitsas et al., 2017], where these pathways are trained on 2D image patches (i.e., 33 * 33 and 65 * 65) and slices (i.e., 240 * 240) from Axial, Coronal and Sagittal views. At the testing step, the prediction results from 3 views is fused using a voting strategy. Moreover, these 3 networks took 12 days for training using a GPU implementation, and for testing, each model took for each view (i.e., Axial, Sagittal or Coronal) in average 3 minutes to segment the entire brain, i.e., 3 networks * 3 minutes equals to 9 minutes in addition to the fusion time which is not reported in the original paper.

[Mlynarski et al., 2019] developed a fully automatic GBM brain tumor segmentation method based on a combination of six models of 3D CNNs architectures. Each of these architectures is composed of three (or five) CNNs architectures, also, these architectures are trained independently and dedicated for one MRI view (e.g., Axial, Coronal or Sagittal slices of the input image) and one architecture based on 3D-CNNs. The proposed segmentation 3D CNNs model is trained on channels concatenation between extracted feature maps from the Axial, Coronal and Sagittal dedicated architectures and two (T2, T1c) or four (T2, FLAIR, T1, T1c) 3D multisequence MR images. The technique of using feature maps as an additional input into the architecture of another CNNs, is used by [Havaei et al., 2017]. Moreover, [Mlynarski et al., 2019] adressed two issues: the first one is long rang context, to solve this issue, they used 2D CNNs (its input is one of three views) to capture rich information through increasing the size of the receptive field. The second issue is unbalanced data, where they solved this issue by using weighted cross-entropy as a loss function.

2.2.3 Discussion

There are several classical approaches for MRI image segmentation as it is mentioned in section 2.2.1, each approach has its own advantages and disadvantages and therefore, MRI image segmentation methods are application-based, there is no general approach or method for all applications, but it depends significantly on the type of applications and the type of images. These conventional methods have many advantages: they are simple for implementation, they exploit important properties such as the neighboring pixels that have some similarity, they separate groups or classes based on criteria and metrics, they incorporate information and knowledge such as shape, orientation, continuity, elasticity or smoothness within the segmentation model (e.g., Atlas-based methods) [Kalinić, 2009]. However, they have many drawbacks: they generally produce poor results and need a user interaction most of the time, where the choice of the right seed points (e.g., region growing) or the threshold value (e.g., thresholding algorithm) is important. And they are sensitive to noise (e.g., edge detector), to poor contrast and to acquisition artifact. Moreover, they need *prior knowledge* from experts and feature engineering. In general, the most remarkable and shared drawback of the classical approaches lies in the subjectivity of their processing and decision-making when it comes to real applications.

Table 2.1 shows a summary of state-of-the-art modern approaches. These approaches are based on Deep learning algorithms, in particular, Convolutional Neural Networks and its variants. Moreover, these approaches are applied to the Glioblastoma brain tumors segmentation. In this thesis, we have been motivated by the recent deep learning models [Chang et al., 2016; Havaei et al., 2017; Zhao et al., 2018], especially when the performance in the field of brain tumor segmentation methods becomes more accurate and the inference time is decreased compared to the time spent by radiologists to detect all tumoral regions. In addition, these automatic brain tumor segmentation methods provide an objective, fast, and reproducible assessment of patients MRI segmentation compared to radiologists who produce a subjective, slow, and difficult to reproduce even for the same patient MRI images.

Table 2.1 – Summary table of the state-of-the-art approaches, where these approaches are applied GBM brain tumor segmentation. These approaches are trained and tested on BRATS datasets. The performance of the these approaches is evaluated using Dice scores and inference time (Speed).

Reference	**Pre-processing**	**Algorithm**	**Dataset**	**Performance**
[Pereira et al., 2015]	normalization, bias field correction	CNNs	BRATS-2013, BRATS-2015	Dice for complete, core, and enhancing regions: - BRATS-2013: 0.88, 0.83, 0.77. - BRATS-2015: 0.78, 0.65, 0.75. Speed: 8 min
[Havaei et al., 2017]	bias correction	CNNs	BRATS-2013	Dice for complete, core, and enhancing (0.88, 0.79, 0.73), Speed: 25 s to 3 min

Table 2.1 – continued from previous page

Reference	Pre-processing	classification	Dataset	Performance
[Chang et al., 2016]	normalization	CNNs	BRATS-2016	Dice for enhancing, core, and complete (0.72, 0.81, 0.87), Speed: 0.52 s for 64 images
[Kamnitsas et al., 2016, 2017]	skull stripping, normalization, registration	DeepMedic + CRF	BRATS-2015	Dice for whole, core, and enhancing (0.89, 0.75, 0.72)
[Zhao et al., 2018]	bias correction, normalization	FCNN + CRF	BRATS-2013, BRATS-2015	Dice for complete, core, and enhancing: - BRATS-2013: 0.87, 0.82, 0.76. - BRATS 2015: 0.8, 0.68, 0.65. Speed: 8 min
[Hussain et al., 2018]	normalization, bias field correction	Deep CNN	BRATS-2013, BRATS-2015	Dice for complete, core, and enhancing: - BRATS-2013: 0.87, 0.89, 0.92. - BRATS 2015: 0.86, 0.87, 0.9.
[Farahani et al., 2013]	normalization	SDAE	BRATS-2015	Dice for whole, core, and active (0.84, 0.71, 0.81)
[Farahani et al., 2013]	/	SegNet	BRATS-2015	Dice for complete, core, and enhancing (0.75, 0.77, 0.76)
[Farahani et al., 2013]	/	NablaNet	BRATS-2012	Dice for whole, core, and enhancing (0.87, 0.69, 0.56)
[Farahani et al., 2013]	bias correction, normalization	3D CNN	BRATS-2016	Dice for whole, core, and active (0.725, 0.611, 0.572)
[Farahani et al., 2013]	/	DNN	BRATS-2015	Dice for complete, core, and enhancing (0.87, 0.75, 0.71)
[Farahani et al., 2013]	normalization	3D CNN	BRATS-2015	Dice for whole, core, and active (0.89, 0.76, 0.37)

All recent cited papers [Zikic et al., 2014; Urban et al., 2014; Pereira et al., 2015; Havaei et al., 2017; Chang et al., 2016; Ellwaa et al., 2016; Kamnitsas et al., 2016, 2017; Zhao et al., 2018; Mlynarski et al., 2019] address the main problem of automatic GBM brain tumor segmentation in different ways using either 2D, 3D Patches or large number of features. The most common things among these advanced methods are: they use at least 3 MRI image modalities and deal with them as input channels (i.e., FLAIR, T2, T1c), they also use CNNs algorithm and 2D patches. Despite the variety of the proposed deep learning and CNNs architectures, the segmentation of GBM brain tumors with the following subproblems is still a challenging issue: (1) unbalanced data; the class of interest (i.e., tumoral region) has the minority of labels which represent for example in our dataset almost 2% of data, while the healthy class has 98% of data [Havaei et al., 2017]. This subproblem is solved by two training step [Havaei et al., 2017], equal sampling of training patches [Havaei et al., 2017; Zhao et al., 2018], weighted cross-entropy [Mlynarski et al., 2019], (2) computational costs: conventional deep learning architectures, in particular, architectures-based on 3D patches, during training they need to compute the gradient of the loss function for each 3D filter per pass. Consequently, they perform thousands even millions of operations, and therefore, they need a massive computing and large memory in each epoch of training. Moreover, in images segmentation applications, one of the important subproblems in CNNs is (3) global context; the relationship among patches, because the training method typically used in CNN is patch-by-patch [Kamnitsas et al., 2017], we refer to this method as adjacent patches. The limit of this method (adjacent patches) is that it does not take into consideration [Havaei et al., 2017; Zhao et al., 2018] that a set of patches represents the full image, that is why we need a new approach for modeling both the context of MRI image and spatial relationship among the patches. Finally, the (4) subproblem of performance degradation that is due to vanishing gradient, overfitting, false positives- where the model predicts non tumor regions as tumor regions but in fact they are not-, and false negatives -where the model classifies some regions as non-tumor regions but in fact they are. The subproblem of false positives is solved in recent methods of deep learning by the threshold technique [Urban et al., 2014; Havaei et al., 2017]. In this thesis, we focus on four main issues in addition to automatic segmentation of GBM brain tumors: unbalanced data, computational cost, global context, performance degradation.

2.3 Glioblastoma brain tumors

Glioma tumors are the most frequent primary brain tumors in adults [Holland, 2001], these tumors represent 81% of all malignant brain tumors [Ostrom et al., 2014], and 45% of all primary brain tumors [Liu et al., 2016]. Some patients with Glioma tumors have survival rate between 0.05% and 4.7% [Ostrom et al., 2014], representing with that one of the top reason of death. According to the WHO classification [Louis et al., 2016; Gupta and Dwivedi, 2017], Glioma brain tumors have 4 grades (I, II, III, IV) where the GBM tumors are the grade IV of Glioma tumors. Moreover, in the last century, they discovered based on histopathology and markers in the tumors, that the cell of origin of Glioma tumors is glial cells [Lindberg et al., 2009; Jiang and Uhrbom, 2012].

Reports and research indicate that ionizing radiation is the number one risk factor that has been identified for glial tumors [DeAngelis, 2001]. Moreover, they found that the radiation even with low dose (i.e., less than 10 Gy) can increase the incidence of glial by a factor of 3 to 7 [Walter et al., 1998], with the shortest [2] and longest latency at adults brain tumors being 14 and 59 years, respectively [Pollak et al., 1998]. The most observed symptoms of patients exposed to radiation are: headaches, mental impairment, focal neurological findings, late onset seizures, increased intracranial pressure, trophic changes of the scalp consisting of alopecia, atrophy and scarring of the skin. However, most patients with these symptoms have shown improvement after surgery [Pollak et al., 1998].

The treatment of patients with brain tumors, including Glioblastoma brain tumors, requires many sessions of radiotherapy with often a combination of surgery and chemotherapy. Clinicians inject a suitable photosensitizer in the tumor, before surgery, to concentrate all tumoral cells at the tumor site. Then, at the surgery, they use photodynamic therapy and after tumor removal, they irradiate any remaining malignant cells with laser light to destroy them [Gibbs et al., 1996].

The assessment of patients with Glioblastoma brain tumors (or with tumor recurrence)

2. The shortest latency of brain tumors at childhood equals to the patient's age at the time of diagnosis minus 10 [Pollak et al., 1998]

carried out either by clinical indicators which are subjective procedure and expertise-based or by MRI images which are an objective assessment. In addition, MRI has many advantages. First, it does not use ionizing radiation that is one of the causes of brain tumors, especially glioma tumors. Second, the results of the MRI are accurate and reproducible and applicable for all patients with Glioblastoma brain tumors [Gibbs et al., 1996]. Expert raters (radiologists) use MRI [Bhandarkar and Nammalwar, 2001a; Akram and Usman, 2011] as a non-invasive technique to generate multi-sequence images and to identify different regions in soft tissue (central nervous system, muscles, hearts, and tumors ...etc.). Usually, the radiologists generate four types of MRI images for each patient for the diagnosis of Gliomas [Işın et al., 2016]: T2-weighted fluid attenuated inversion recovery (T2-Flair), T1-weighted (T1), T1-weighted contrast-enhanced (T1c), and T2-weighted (T2). Moreover, to generate MRI images of type T1c, radiologists inject Gadolinium-Containing Radiocontrast Agents (GCRA) to follow tumor regions through the contrast of GCRA [3].

In patients' MRI images with GBM tumors, we find 4 sub-regions in addition to background and healthy tissue. Moreover, the challange of Glioma brain tumors, including GBM, is that they invade the surrounding tissue rather than displacing it, causing unclear boundaries. In addition, Glioma tumors in MRI images have the same appearance as Gliosis, stroke, inflammation, blood spots [Goetz et al., 2015]. These challenges that lie to the detection of Glioma brain tumors features make the mission of radiologists and oncologists very difficult to diagnose and to assess the therapeutic responses of these tumors.

To interpret brain MRI images, a specialized radiologist employs a manual segmentation using information from MRI images with anatomical and physiological knowledge [Işın et al., 2016]. Where it's known that the manual segmentation in MRI images is a time-consuming and a tedious procedure [Abd-Ellah et al., 2016]. The key challenge is the interpretation and the segmentation of brain MRI images that depend on the expertise of each radiologist [Akram and Usman, 2011]. Moreover, the mission of radiologist

3. Research and studies [Grobner, 2006] have found that patients injected with GCRA can develop undesirable side effects such as nephrogenic systemic fibrosis in patients with preexisting renal dysfunction. Moreover, other later studies [McDonald et al., 2015] say that the injected GCRA accumulates in neuronal tissue and can stay in the brain for a long time, and they also confirm that the accumulation of GCRA occurs in the absence of renal or hepatobiliary dysfunction. This issue opens the door for discovering an alternative solution in the future for T1c images such as image synthesis using generative adversarial networks.

becomes very difficult in the case if the tumor region has intra-tumoral structures such as the GBM tumors which have four different structures besides healthy tissue inside the tumor region: edema, non-enhancing solid core, necrotic core, enhancing core (See figure 2.1). That's why the Dice score of human rater (i.e. radiologists) is in the range 74% - 85% [Menze et al., 2015]. In addition, the radiologist spends between 3 and 5 hours [Kaus et al., 2001], to extract features manually from MRI images then to label each pixel if it belongs to a tumor region and which region from 4 regions or it belongs to healthy tissue. Thus, the manual segmentation is a time-consuming compared to automated methods which take between few seconds and 20 minutes using a modern GPU implementation. Since the aim is to obtain a segmentation of brain tumors, given the number of patients, obviously automated segmentation methods are practical for day-to-day use in clinical centers and for research.

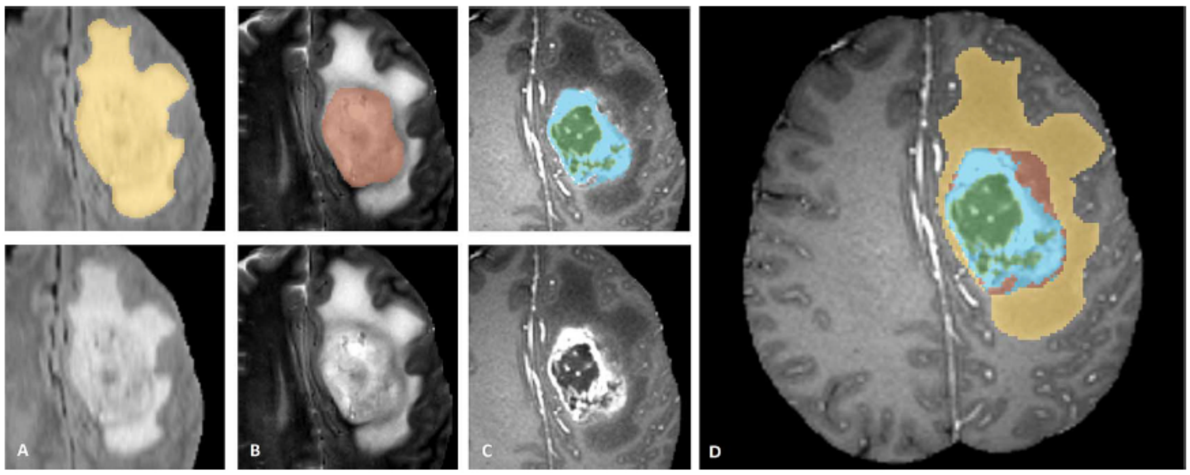


Figure 2.1 – An example of manual annotation (segmentation) by a radiologist (expert rater). The segmentation process passes through two stages: first, each GBM tumor's sub-region is detected in a different MRI modality: they detect the whole tumor in FLAIR (A), the tumor core in T2 (B), the enhancing tumor structures in T1c (blue), the surrounding necrotic components of the core (green) (C). In the second stage, they combine all annotations from different MRI modalities into one template (D): edema (yellow), non-enhancing solid core (red), necrotic core (green), enhancing core(blue) [Menze et al., 2015].

2.4 BRATS datasets

To develop Deep Learning models for the segmentation of GBM brain tumors, we need to use an annotated dataset of MRI images and validated by radiologists. Thus, to validate these models, we have to evaluate our experiments on real patient's data, in which we use three BRATS[4] datasets: BRATS 2017, BRATS 2018, and BRATS 2019 datasets. The training sets of BRATS 2017 and 2018, each has 210 patient's brain images with high-grade (HGG) and 75 patient's brain with low-grade (LGG) while BRATS 2019 training set has 259 patients' brain images with HGG and 76 patient's brain images with LGG. Each patient's brain image comes with 4 MRI sequences (i.e. T1, T1c, T2, FLAIR) and the ground truth of the 4 segmentation labels which are obtained manually by radiologists experts: Healthy tissue, Necrotic and Non-Enhancing tumor, Peritumoral Edema, Enhancing core. BRATS 2018 validation set contains 66 images of patients while BRATS 2019 validation and testing sets contain 125 and 166 images respectively with unknown grade, i.e. the validation set does not have the ground truth labels. BRATS datasets are collected from different centres–Bern University, Debrecen University, Heidelberg University, Massachusetts and General Hospital– and using different MRI scanners (1.5T, 3T) [Menze et al., 2015].

The initial preprocessing operations performed by the organizer and the provider of the dataset are: first, to homogenize the MRI images, they registered rigidly all images to the T1c MRI in addition to resampling to 1 mm^3 resolution. Second, to guarantee the anonymization of the patients they skull stripped all MRI patients images.

There are four essential points in the BRATS dataset: first, the size of each patient's MRI image is $240 * 240 * 155$ (height*width*number of slices). Second, each voxel is labeled manually by an expert annotation (radiologist), where after that each voxel is assigned to one class: healthy tissue and background (class 0), non-enhancing core (class 1), edema (class 2) contrast-enhancing core (class 3). Third, there are no negative cases, in other words, all patient MRI images contain GBM brain tumors. Fourth, there is in many cases a remarkable variability in manual segmentation of brain tumors: intra-rater, $20 \pm 15\%$; inter-rater, $28 \pm 12\%$ [Mazzara et al., 2004], indicating that the manual segmentation by expert raters is not the same even for the same patient MRI image, thus, to avoid an individual baised segmentation, they fused the decision of many expert

4. BRATS data is the only known, largest and publicly available dataset for patients with Glioblastoma, possibly due to the issue of data privacy and confidentiality, data security, data access rights, and these issues open the door to the exploration of two approaches adapted to these issues: federated learning and reinforcement learning.

raters in one consensus[5] segmentation (See figure 2.2). Fifth, the experts use a specific annotation and clinical protocol to label each voxel [Menze et al., 2015]:

1. They detect the edema region primarily from T_2 images, they also use FLIAR image to check and discriminate edema's extensions from ventricles and fluid-filled structures
2. After that, the two tumors sub-structures, i.e., enhancing and non-enhancing are detected in T_1c images together with T_1 images.
3. They determine visually an intensity threshold to segment the enhancing core in T_1c images within the result of step (3) in addition to using GCRA and excluding the necrotic and vessels.
4. The necrotic core is segmented within the enhancing rim visible in T_1c images.
5. Finally, the remaining part after subtraction the enhancing core and the necrotic core sub-structures, they obtain the non-enhancing (solid) core.

5. Given the consensus segmentation (Ground truth) is done by the fusion of the results of many experts, the number of experts still affects the final result, and this issue opens the door to exploring other segmentation approaches in the future such as unsupervised and reinforcement learning.

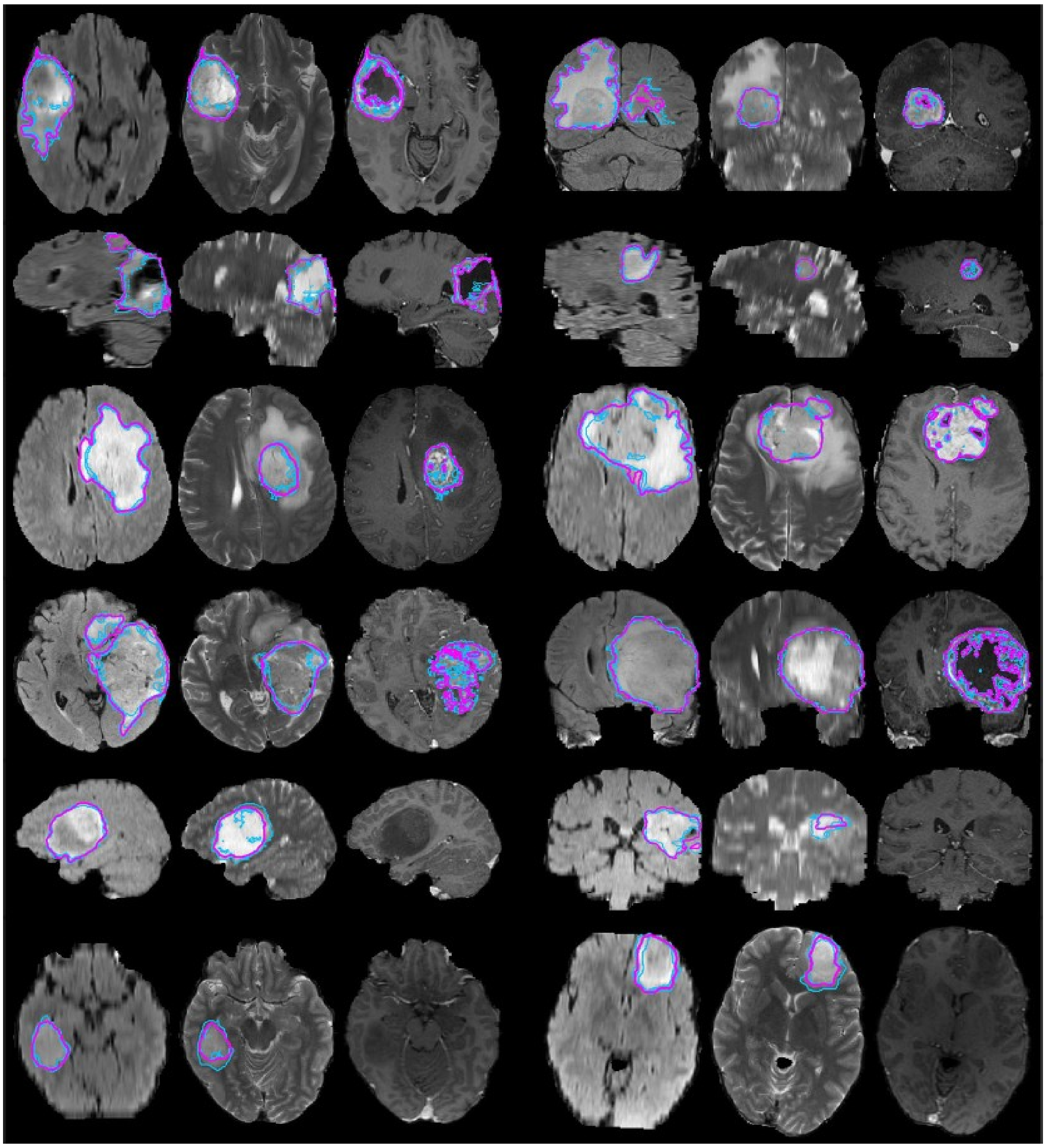


Figure 2.2 – MRI examples from the BRATS dataset. In each row, we have two cases of HGG tumor (rows 1–4), LGG tumor (rows 5–6). Also, for each MRI image, we have different views (i.e., axial, sagittal, and coronal), different types: the whole tumor region in FLAIR (left), the core region in T2 (center), the active tumor region in T1c (right) and two segmentations: individual segmentation is shown in blue lines, while the consensus segmentation is shown in magenta lines. [Menze et al., 2015].

2.5 MRI image quality limitations

In this section, we highlight the most challenges and issues in MRI images, that can affect different image segmentation approaches:

— Noise: MRI images prone to noise especially white noise which can be modeled by Rician distribution, this type of issues affect significantly threshold-based approaches. These approaches do not have the ability to deal with noise which makes the separation of structures or regions very difficult due to the corruption of the MRI image histogram [Pham et al., 2000].

— Intensity inhomogeneity: this type of issues is due to intensity variation for each MRI image, which can lead after that to destroying the image histogram [Yazdani et al., 2015] and to generating a shading effect over the MRI image. Intensity inhomogeneity issue arises from the process of the image acquisition limitation (see figure 2.3), where the sources of this issue came from two reasons: firstly, static field inhomogeneity, bandwidth filtering, radio frequency transmission and reception inhomogeneity. Secondly, the state of the object inside the magnet, the magnetic permeability and dielectric properties of the object [Vovk et al., 2007].

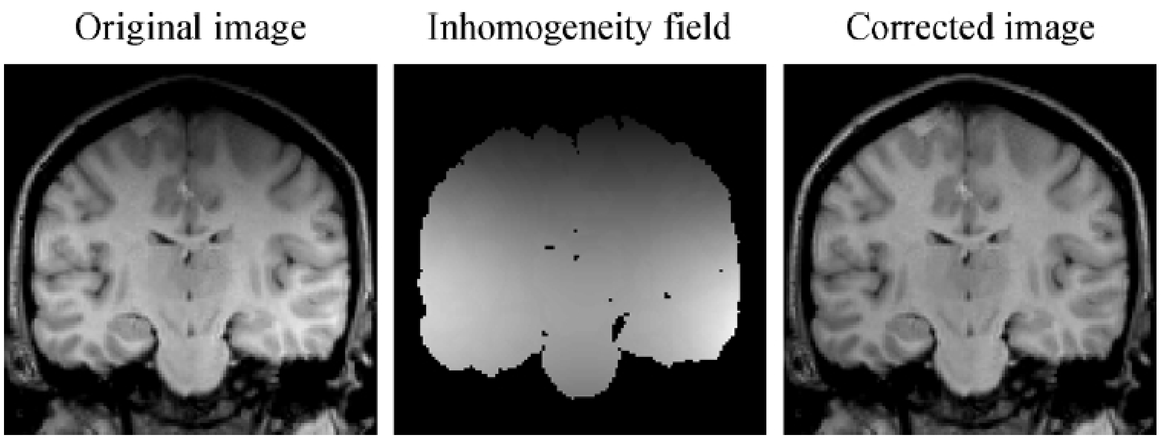


Figure 2.3 – Example of the Intensity inhomogeneity issue in a MRI image [Vovk et al., 2007].

— Intensity non-standardization: the MRI images of the BRATS datasets are taken using different MRI scanners and parameters in different acquisition centers as it is mentioned in section 2.4. MRI scanners manufacturers use different imaging protocols, and therefore, they provide different MRI images even for the same patient [Sugita et al., 2010]. Unlike X-ray and CT scans, the pixel intensities in MRI scans have different values for the same issue [Alegro et al., 2012].

— Patient motion: it is known that the patient moves between image acquisitions after administration of Gd-DTPA [6] contrast agent, especially if we know that the MRI exams take for about 30 minutes [Gibbs et al., 1996]. Moreover, with this issue, we obtain MRI images with contrast and anatomic variations which lead to difficulties and challenges in image registration. The latter is an important step in image segmentation approaches and especially it delimits Atlas-based segmentation methods [Yazdani et al., 2015].

— Partial volume effect: the results of this issue are: getting connected regions despite they are separated and a blurring of intensity at the boundary between regions (see figure 2.4). The latter issue is due to the contribution of different tissues to a single voxel. The partial volume effect is the first reason for the existence of uncertainty in the region boundaries, consequently, the existence of a soft segmentation resulting from the overlapping of the regions. On the other hand, hard segmentation does not allow the voxel to be in two regions at the same time (whether inside or outside the region) [Pham et al., 2000].

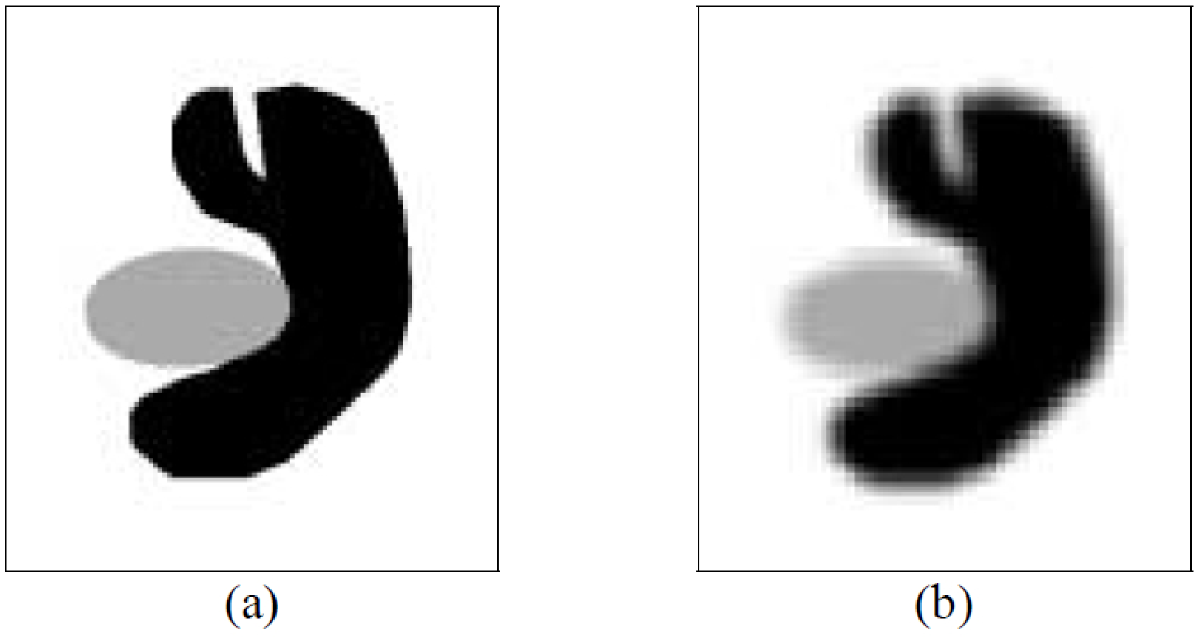


Figure 2.4 – Example of the partial volume effect issue [Pham et al., 2000]: (**a**) ideal image, (**b**) acquired image by a MRI scanner.

6. Gd-DTPA: Gadolinium-diethylenetriamine pentaacetic acid

2.6 Evaluation metrics

To evaluate the performance of Deep Learning architectures, we used BRATS online evaluation system [7], where we upload the segmentation results and the system provides the quantitative evaluations contain Dice score equation 2.1, Sensitivity equation 2.2, Specificity equation 2.3, and Hausdorff distance equation 2.4. For each one of these metrics we have three sub-metrics: Complete (i.e. Necrotic and Non-Enhancing tumor, Peritumoral Edema, Enhancing tumor), Core (i.e. Necrotic and Non-Enhancing tumor, Enhancing tumor), Enhancing (i.e. Enhancing tumor). The following figure 2.5 shows the used regions to compute the evaluation metrics:

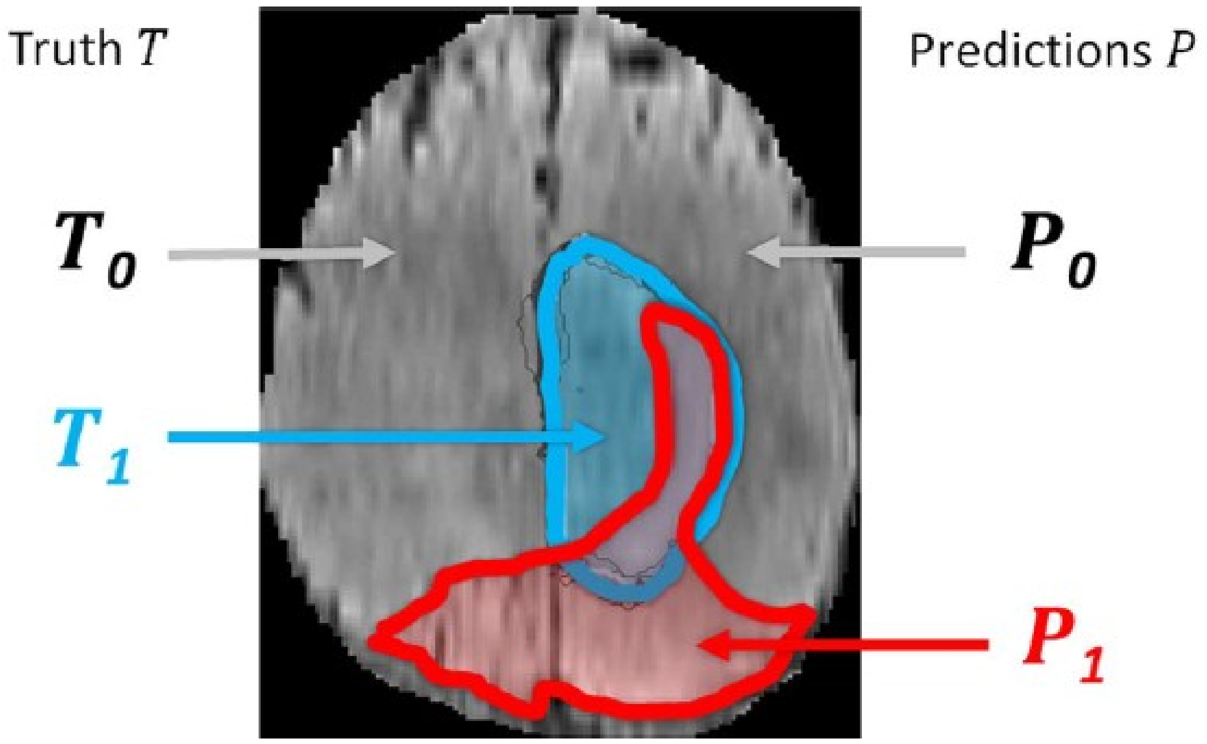


Figure 2.5 – The quantitative evaluations: Dice score, Sensitivity, Specificity, Hausdorff distance are computed from the outlined four regions. T_1 is the true lesion region (blue line), T_0 is the remaining normal region, P_1 is the predicted lesion region (red line) and P_0 is the predicted to be normal region [Menze et al., 2015].

The evaluation metrics (i.e., Dice score, Sensitivity, Specificity, Hausdorff distance) are calculated as follows:

$$Dice(P,T) = \frac{|P_1 \wedge T_1|}{(|P_1| + |T_1|)/2} \tag{2.1}$$

7. Center for Biomedical Image Computing and Analytics University of Pennsylvania, Url:https://ipp.cbica.upenn.edu/

$$Sensitivity(P,T)= \frac{|P_1 \ \wedge \ T_1|}{|T_1|} \tag{2.2}$$

$$Specificity(P,T)= \frac{|P_0 \ \wedge \ T_0|}{|T_0|} \tag{2.3}$$

$$Hausdorffdistance(P,T)= max \ \{ \ \underset{p\in\partial P_1}{sup} \ \underset{t\in\partial T_1}{inf} \, d(p \ , \ t) \ , \underset{t\in\partial T_1}{sup} \ \underset{p\in\partial P_1}{inf} \, d(t \ , \ p) \ \} \tag{2.4}$$

Where $\wedge$ is the logical AND operator, P is the model predictions and T is the ground truth labels. T_1 and T_0 represent the true lesion region and the remaining normal region respectively. P_1 and P_0 represent the predicted lesion region and the predicted to be normal respectively. ($|.|$) is the number of pixels. MRI image is a 3D space (i.e. a set of points in R^3): height, width and depth (i.e. a set of slices), each point on the image is a voxel, and a set of voxels represent a volume. To compute the Hausdorff distance, first we differentiate the voxels of each class (i.e. creating a map) then we save only the voxels that belong to the borders of each class, these voxels represent a surface (*Voronoi surface*). 'p' and 't' are two points on the surface ∂P_1 and ∂T_1 respectively, and d(p,t), d(t,p) are the shortest least-squares distance between point 'p' and 't' and vice versa for d(t,p). Finally, instead of calculating the maximum distance, we compute only 95% of the surface distance between each point from the first surface (i.e. the true lesion region) and the other surface (i.e. the predicted lesion region). This fractional distance is a robust version of Hausdorff because the existence of outliers can have a considerable impact on the last one (i.e. Hausdorff), also, it is proven that it gives better results than the maximum distance [Menze et al., 2015].

In addition to the four metrics provided by the online evaluation system (i.e., Dice score, Sensitivity, Specificity, Hausdorff distance), we use another evaluation metrics to compute the computational benefits: the number of parameters (trainable weights), inference time that represents the needed time to make prediction on a new data set, which means the quantity of operations that needed to execute the model on the platform (device). Moreover, to focus on the effectiveness of Deep learning models, we measure the metric of Dice whole density, i.e. Dice whole divided by the number of parameters. The best model is the one that has a high Dice whole density, in other words, the model that uses effectively its parameters. Measuring information density is one of the most used metrics to measure the effectiveness of deep learning models; information density shows us the ability of each model to use its parameters efficiently [Canziani et al., 2016; Bianco et al., 2018].

In this thesis, we propose different contributions with different deep learning architectures to introduce an improvement of seven evaluation metrics of state-of-the-art methods: Dice score, Sensitivity, Specificity, Hausdorff distance, inference time, number of parameters, Dice whole density.

2.7 Discussion and Conclusion

In this chapter, we presented a survey on MRI image segmentation approaches. We also showed the anatomy and the origin of Glioblastoma brain tumors, besides, how radiologists and oncologists diagnose and assess patients' MRI images with these tumors. The limitation and the challenges of MRI image quality are presented in addition to their influence on the segmentation and classification methods. Moreover, we presented the evaluation metrics that will be used to compare the performance of state-of-the-art methods.

Through a variety of applications in medical image analysis, in particular, Glioblastomas brain tumors segmentation, Convolutional neural networks have proven their effectiveness in predicting these tumors. In addition, they provide a promising segmentation performance compared to conventional approaches and state-of-the-art methods in terms of MRI image segmentation and inference time.

In the next chapter, we present Convolutional Neural Networks (CNNs), and how these Networks can be applied to input matrices, and from this principle, how we can apply CNNs to MRI images as input for Glioblastomas image segmentation. Furthermore, we present the fundamental building blocks of CNNs algorithm, and the operations behind this algorithm such as convolution, non-linear activation function, pooling, upsampling, and fully-connected layers. In addition, we present two important algorithms: firstly, the forward algorithm to transform the input and the output of the each layer from a representation space to another non-linear representation space, also how this algorithm is used to compute the inference and loss function in each epoch. Secondly, the backward algorithm which is considered the workhorse for optimizing deep learning networks is used to adjust the parameters (weights, biases) of each layer towards the global minimum. At each epoch, the backward algorithm computes the gradient of the objective function with respect to each parameter before determining the next value of each parameter in the continuous landscape of hills and valleys of the objective function.

Chapter 3

Background Theory: Convolutional Neural Networks

Contents

3.1 Introduction

Deep learning is a subfield of machine learning that allows computers to learn from experience and data, and machine learning, in turn, is a subfield of Artificial Intelligence [1] (AI). AI is a set of fields that study how can machines mimic the congitive functions of the human being such as learning and problem solving. Machine learning is basically about how to program a computer without being explicitly programmed. So in fact, Deep Learning dates back to 1940s with the work of (McCulloch and Pitts, 1943; Hebb, 1949; Rosenblatt, 1958), which consist of a set of algorithms inspired by the brain function

1. At an important conference in 1955 among many experts in information theory and mathematics, they used the word Artificial Intelligence for the first time to present an academic discipline.

and the idea of stacking many layers of neural networks to transform raw data from one representation space to another with more compact and discriminative features. In this chapter, we focus more on deep learning, precisely, on Convolutional Neural Networks (CNNs).

In recent years, CNNs [Fukushima, 1980; LeCun et al., 1989] have proved that they are a powerful tool for feature detection and extraction in many computer vision tasks. One of the first known CNNs architectures is LeNet5, this architecture developed by [LeCun et al., 1998], and it appeared after many previous versions of CNNs since the work of handwritten zip code recognition in 1989 [LeCun et al., 1989].

CNNs networks are usually applied to images or in general to matrices because images have a spatially correlated nature, also, the image's features are distributed across the whole image. So, CNNs networks have exploited these properties and added a new layer called convolution. This convolution layer is applied to the input images (raw pixels). The idea of adding the convolution layer before classifier allowed to extract relevant features at multiple locations. By stacking many convolution layers (pooling layer can be added alternatively with convolution), kernels in each CNNs layer play an important role as a feature detector. So, by choosing a specific kernel we can detect a specific feature that we want. Moreover, the kernels' parameters represent the weights that need to be updated to obtain the most suitable configuration, in another word, the kernels tuned during the training process for pushing it to respond strongly to specific features. This updating operation of weights is often done by the Backpropagation [Rumelhart et al., 1986] algorithm. Where lower-kernels' parameters at lower-layers of CNNs networks are trained to detect features such as edges, corners, and higher-kernels' parameters at higher-layers of CNNs are trained to detect more complex features or high-level features such as shapes and object parts. CNNs algorithm is based on four standard operations: Convolution, Pooling, Non-linear activation function, Multi-Layer Perceptron (MLP) [Rosenblatt, 1961] in addition to two algorithm for computing the loss function and the new parameters: forward and backward algorithms. In this chapter, we present the fundamental building blocks of Convolutional Neural Networks such as convolution, non-linear activation function, pooling, upsampling, fully-connected layers, and Dropout, in addition to, the forward and backward algorithms.

3.2 Convolutional Neural Networks

In this section, we describe the important operations behind the CNNs algorithm: Convolution, Pooling, Upsampling, Non-linear Activation function, Fully-conncected layers. The important property of these operations is that they can be applied to the Input images or to the output of the hidden layers.

3.2.1 CNNs operations

3.2.1.1 Convolutional Layer

To a computer, an image is a grid or matrices of numbers, so, to apply a Convolution[2] operation over the input image, a set of filters (also known as Kernels) is used on raw pixels. Moreover, applying filters[3] on images is known in conventional methods such as Sobel filter, Gaussian filter, high-pass filter, ...etc., the new added property of CNNs compared to conventional methods is that CNNs use many filters with trainable parameters in successive layers. Therefore, the latter property allows CNNs to overcome the conventional methods by extracting hierarchical features (See figure 3.1) across the whole image with the same filters (shared weights) and only some input units contribute to the output unit (sparse connectivity). Furthermore, the advantage of Convolution is that it is applicable to many types of multidimensional data and to many computer vision tasks.

Moreover, the used filters in many stackable convolutional layers (other layers can be added alternatively with convolution) act as a feature detector. So, by choosing a specific filter we can detect a specific feature that we want. The filter parameters represent the weights that need to be updated to get the most suitable configuration, in another word, the filter tuned during the training process for pushing it to respond strongly to a

2. The original idea of Convolution is Cross-correlation which shifts a filter (window) through the input to find the large numbers which are a matching indicator between the filter and the input.

3. In conventional methods, filters are used often to highlight edges, to remove noise or to increase image details,...etc.

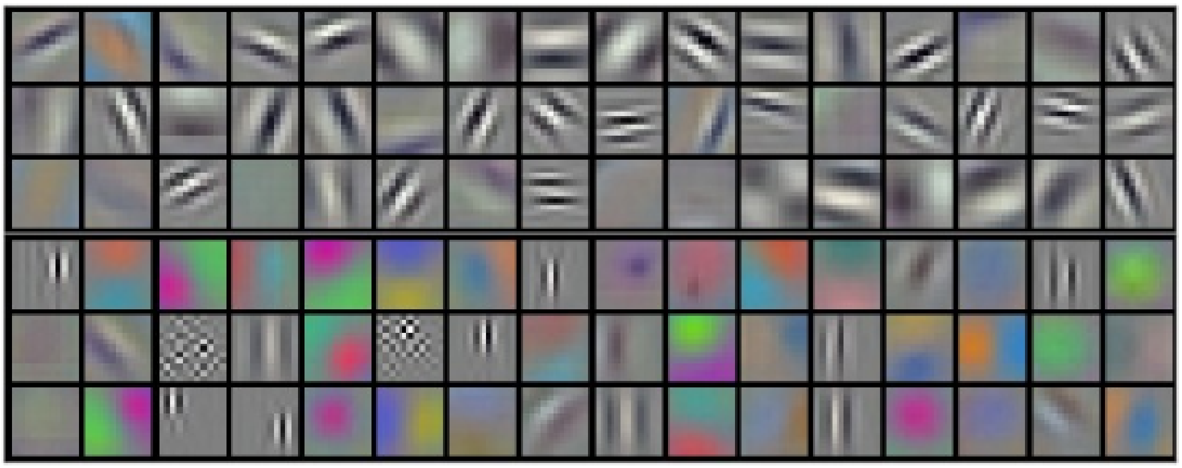


Figure 3.1 – Visualization of the first convolutional layer 's filters of the AlexNet model [Krizhevsky et al., 2012]. Those filters are trained to detect lower-features such as lines and edges in a different directions.

specific feature. This updating operation of weights is often done by Backpropagation [4] [Rumelhart et al., 1986] algorithm, i.e, the goal of the Backpropagation is to minimize the objective function (also called loss function) by adjusting the parameters (weights and biases) towards the global minimum. Where lower-filters parameters are trained to detect features such as edges, corners, and higher-filters parameters are trained to detect more complex features or high-level features such as shapes and object parts. The output of convolution operation called feature map which is calculated as follows:

$$Y_i = b_i + \sum_j W_{ij} * X_j \tag{3.1}$$

For 2D inputs (single-channel image), the convolution operation of input image $X \in \mathbb{R}^{f \times h}$ and kernel matrix W $\in \mathbb{R}^{n \times m}$ is defined as follows:

$$Y_{ab} = \sum_{i=1}^{n}\sum_{j=1}^{m} W_{ij} * X_{a+i-1,b+j-1} \tag{3.2}$$

Where Y is a feature map (the output of convolution operation), b$_i$ is a trainable parameter bias, W$_{ij}$ is trainable parameter weights. In CNNs architectures W$_{ij}$ are the kernel matrices parameters. X is the input, * is the convolution operator. Convolution layer reduces the number of dot-product operations by using the property of sparse interactions [5] which is in contrast with traditional feed forward neural networks that have a

4. See section 3.2.3 for further details
5. Sparse connectivity has fewer connections between two successive layers.

dense connectivity, i.e, all input units are connected to all hidden layer units. The Convolution operation works by convolving the filter with the input where the filter starts with the leftmost part, then it slides to the right by a step called stride after computing the elementwise production between the filter elements and the input elements. Then, the results are summed, plus a bias term, to form an output unit (see figure 3.2). This process is repeated until the filter slides all positions of the input and computing all the output units. Each filter is initialized with different weights to push it to look for a specific and different feature on the input. In standard CNN architecture (e.g. AlexNet [Krizhevsky et al., 2012]), multiple filters are used for each layer, where after calculating the feature map, they are grouped together to form the output.

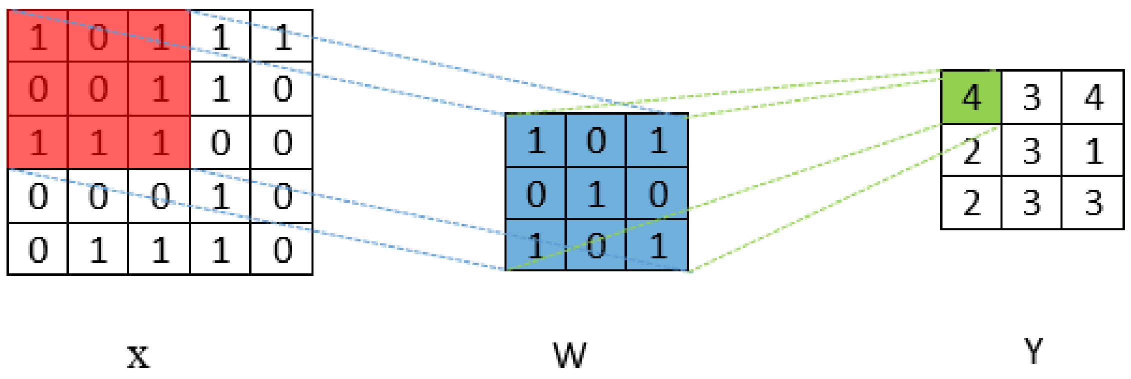


Figure 3.2 – An allustartion of Convolution Operation on a single-channel image (Equation 3.2). Feature maps (Y) is the result of convolving the input (X) with a Filter (W).
.

The size and the number of the feature maps are controlled by three parameters in addition to the size of the Input Image:

— **Depth**: the depth of the output (Convolved feature) is related to the number of filters we use for the Convolution operation. In the standard CNNs architecture, we use many filters in each layer and as a result we obtain many features maps, so, these feature maps will be stacked as 2D matrices.

— **Stride**: is how much a kernel is shifted across the input image with each step

— **Zero-padding**: is the operation of adding zeros around the border (outside the matrix), like this we can control the size of output (feature map), this operation called also *wide convolution*

More formally, in the following section, we define the relationship between these parameters:

Relationship 3.1 *A convolution output O^{L+1} of layer L+1, is described by Input I^L, Kernel K^L, Padding P^L and Stride S^L:*

$$O^{L+1} = \frac{I^L + 2P^L - K^L}{S^L} + 1 \qquad \forall O, S, I, K, P \in \mathbb{N} \tag{3.3}$$

if No zero padding (i.e., $P = 0$) then:

$$O^{L+1} = \frac{I^L - K^L}{S^L} + 1 \qquad \forall O, S, I, K \in \mathbb{N} \tag{3.4}$$

Example: Let's consider an Input $I = 5$ X 5, Kernel $K = 3$ X 3, Padding $P = 1$ and Stride $S = 2$, thus, the output O of the convolution operation in this case: $O = [(5+(2*1)-3)/2]+1 = 3$, see figure 3.3 for more details.

Remark: if the Input I or the Kernel K are not square (e.g., $I = 4$ X 5 or $K = 3$ X 2), in this case, equation 3.3 must be applied independently to each axis (i.e., horizontal and vertical axes).

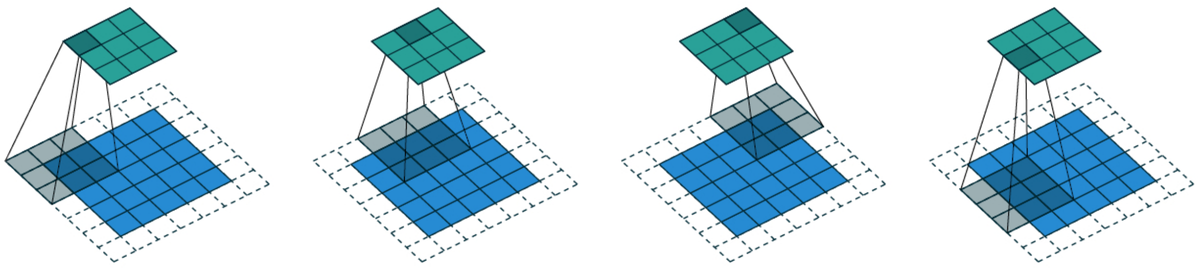


Figure 3.3 – An example of the convolution layer, an Input image I= 5 × 5 (blue), Kernel K= 3 × 3 (gray), Padding P= 1 ,Strides S = 2, the produced output O = 3 × 3 (green). Figure from [Dumoulin and Visin, 2016].

3.2.1.2 Activation function

After the convolution operation, we apply another important operation called non-linear activation function to separate and to transform the feature maps after each convolution operation from a space representation to another representation with more discriminative feature maps. In literature, there are many activation functions (See figure 3.4): Sigmoid (see equation 3.5) has a function curve in the range [0,1], Hyperbolic tangent (Tanh[6]) (see equation 3.6) has a function curve in the range [-1,1], and Rectified

6. Tanh was first used in the 1760s independently in the work of Vincenzo Riccati and Johann Heinrich Lambert. In addition, Johann Heinrich Lambert proved that Tanh can be written as a series of a continued fraction (known as Lambert's continued fraction).

Linear Units (ReLU) (see equation 3.7) has a function curve in the range [0, ∞[. The cited functions arised naturally in neural networks theory as the activation function of a neural unit [Cybenko, 1989] and they have recently been adopted into almost deep learning networks. The drawback of Sigmoid and Tanh activation functions is that they lead to the problem of vanishing or exploding gradient when they used with gradient-based algorithms to adjust neural networks' parameters [Gao et al., 2019] such as stochastic gradient descent and its variants.

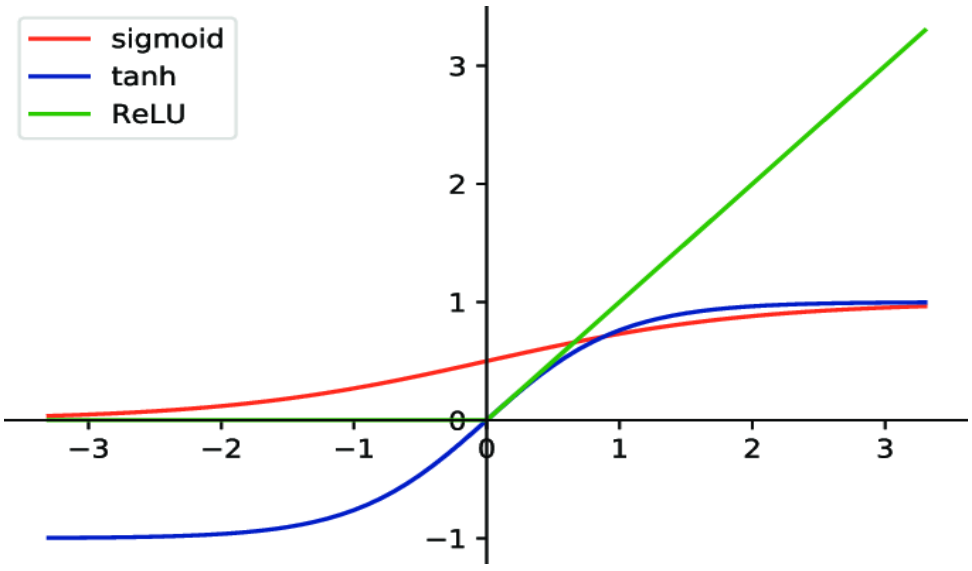


Figure 3.4 – An illustration of Sigmoid, Tanh, and Relu function curves. The principal axes represent the pre-activation and activation function. Figure from [Gao et al., 2019]

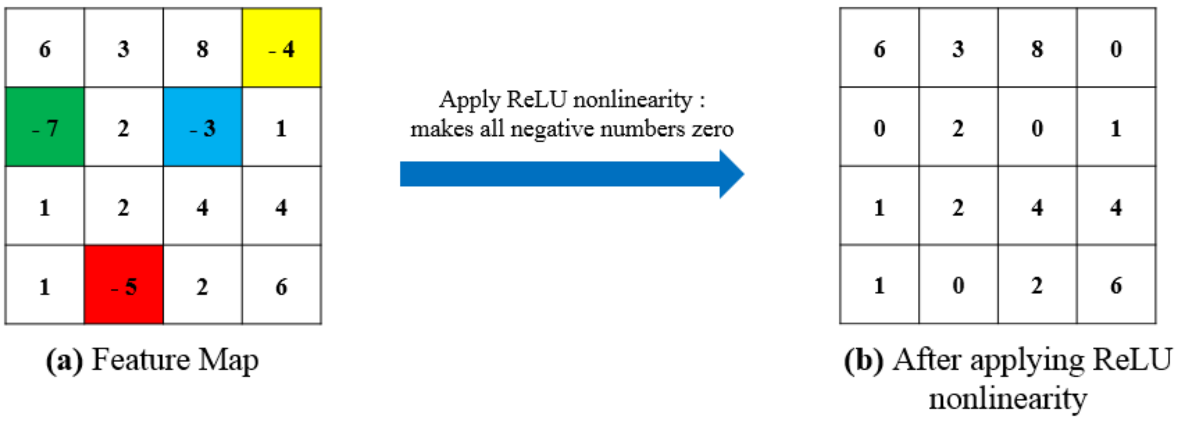
The most used activation function in the field of deep learning is ReLU function [Glorot et al., 2011] due to its properties: it is a non-linear function, differentiable (it has either 0 or 1 slope), and it has a non-saturating property on the positive side [0, +∞[.

$$F(w \times x + b) = Sigmoid(w \times x + b) = \frac{1}{1 + e^{-(w \times x + b)}} \tag{3.5}$$

$$F(w \times x + b) = Tanh(w \times x + b) = \frac{\mathrm{e}^{(\mathrm{w \times x+b})} - e^{-(w \times x + b)}}{\mathrm{e}^{(\mathrm{w \times x+b})} + e^{-(w \times x + b)}} \tag{3.6}$$

$$F(w \times x + b) = Relu(w \times x + b) = \begin{cases} w \times x + b & , if(w \times x + b) > 0 \\ 0 & , otherwise \end{cases} \tag{3.7}$$

Where $(w \times x + b) \in \mathbb{R}$ is the neuron's output, w and b are trainable parameters weight and bias respectively. F is the activation function that transfoms the neurons' output from linear space to non-linear space.



6 3 8 - 4
- 7 2 - 3 1
1 2 4 4
1 - 5 2 6
(a) Feature Map
Apply ReLU nonlinearity : makes all negative numbers zero
6 3 8 0
0 2 0 1
1 2 4 4
1 0 2 6
(b) After applying ReLU nonlinearity

Figure 3.5 – An example of applying ReLU as an activation function. (**a**) is the result of convolution operation called feature maps, (**b**) is the result after applying ReLU nonlinearity. ReLU works by replacing neuron's output (x) by zero if $x < 0$, else Relu acts as a linear function with slope equals to 1.

Moreover, ReLU activation function works by replacing all negative pixel (or feature) values in the convolved feature map by zero, this property allows Relu function to create a sparse representation (see figure 3.5). In addition, the purpose of activation function is to introduce the non-linearity in CNNs networks because the Convolution operation is a linear operation. The empirical experiments in figure 3.6 show that training CNNs networks with Relu activation function will converge six times faster compared to Tanh activation function [Krizhevsky et al., 2012] that is used mostly in shallow neural networks.

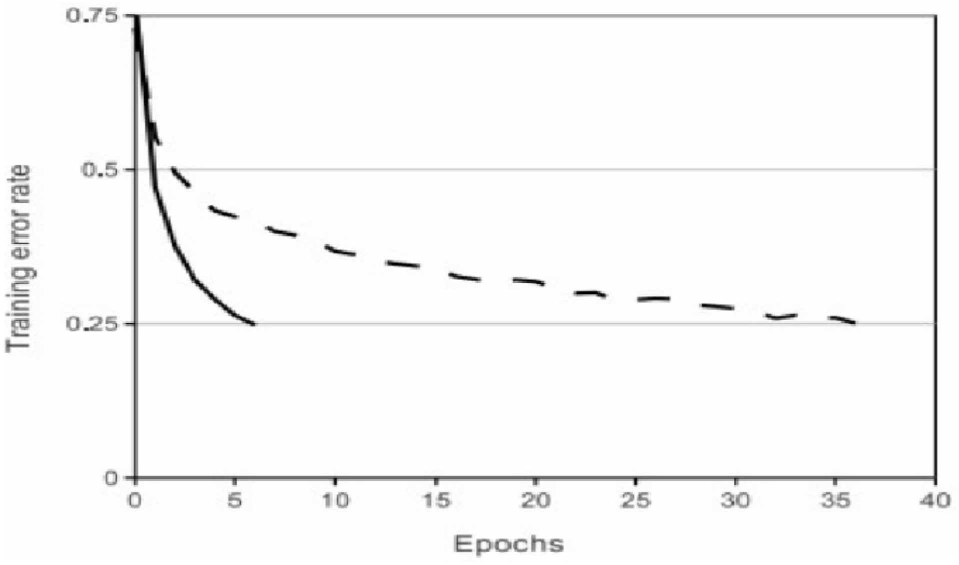


Figure 3.6 – An illustration of a comparison between the training curve of a CNNs model with Tanh (dashed line) function and ReLU (solid line) function. Relu unit shows 6 times faster in convergence in terms of training error rate compared to Tanh unit. Figure from [Krizhevsky et al., 2012].

3.2.1.3 Transposed convolution layer

One of most used technique in Deep Learning architectures to obtain a higher-resolution feature map (higher-dimensional space) from a lower-resolution feature map is Up-sampling technique[7] (also known as Transposed convolution). This technique does not use any trainable parameters, which works by inserting zeros padding internally between pixels, where the number of zeros is determined by the stride parameter. Final step is to convolve the sparse feature maps with trainable parameters (i.e. convolution operation) to obtain dense features maps, this technique helps to avoid the problem of Overfitting[8] [Badrinarayanan et al., 2015]. More formally, the technique of Up-sampling operation works as follows:

$$M_i(n) = Upsampling_{L,n}(G_i) \tag{3.8}$$

7. There is another technique called deconvolution, but this technique uses trainable parameters and it is less used because it is highly prone to overfitting.

8. The model begins to learn patterns unique to the training data but irrelevant when it comes to the test data.

$$M_i = \begin{cases} G_i(n/L) & if(n/L) \in \mathbb{N} \\ 0 & otherwise \end{cases} \tag{3.9}$$

Where G $\in \mathbb{R}^{f\times h}$ is a lower-resolution feature map matrix, and M $\in \mathbb{R}^{(a.f)\times(b.h)}$ is a higher-resolution feature map matrix. Up-sampling operator works by inserting $L-1$ zeros between $G(n)$ and $G(n+1)$ for all input units. The obtained sparse feature map M is convolved with a Convolution Kernel to transform M from sparse to a dense feature map. Transposed convolution technique is used in many deep learning architectures [Zeiler et al., 2011; Zeiler and Fergus, 2014; Long et al., 2015; Radford et al., 2015; Visin et al., 2015; Ronneberger et al., 2015; Im et al., 2016] as a technique to transform the feature map shape back to the original image shape[9]. Formally, the following definition explains the relationship that controls the size of the Transposed convolution output, when it is applied to a 2D input image:

Relationship 3.2 *A Transposed convolution output O^{L+1} of layer L+1, is described by Input I^L, Kernel K^L, Padding P^L, Stride S^L and $a^L \in \{0,...,S^L\text{-}1\}$ to relax the constraint on the input size:*

$$O^{L+1} = S^L(I^L - 1) + K^L + a^L - 2P^L \qquad \forall O, S, I, K, a, P \in \mathbb{N} \tag{3.10}$$

if No zero padding (i.e., P = 0) then:

$$O^{L+1} = S^L(I^L - 1) + K^L + a^L \qquad \forall O, S, I, K, a \in \mathbb{N} \tag{3.11}$$

Example: Let's consider an Input I = 3 X 3, Kernel K = 3 X 3, Padding P = 1, Stride S = 2 and a = 1, thus, the output O of the Transposed convolution operation in this case: O = 2(3 - 1) + 3 + 1 - 2*1 = 6, see figure 3.7 for more details.*

Remark: if the Input I or the Kernel K are not square (e.g., I = 4 X 5 or K = 3 X 2), in this case, equation 3.10 must be applied independently to the horizontal axis and to the vertical axis.

9. Note that the transposed convolution technique is not the inverse of convolution, so it returns a feature map that has the same width and height of the input.

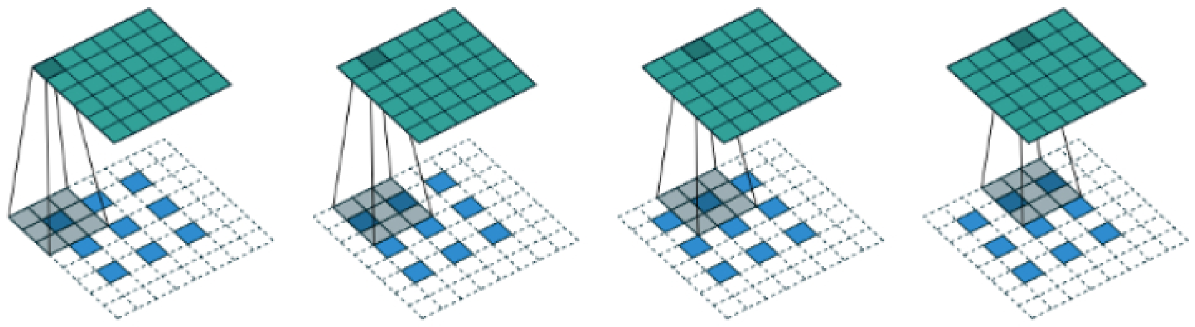


Figure 3.7 – An example of transposed convolution layer, applied on a single-channel image (Equation 3.8 and Equation 3.9). Input image I= 3 × 3 (blue), Kernel K= 3 × 3 (gray), Padding P= 1, a = 1 (an added border of zeros, to the bottom and right edges) , Strides S = 2, the produced output O = 6 × 6 (green). Figure from [Dumoulin and Visin, 2016].

3.2.1.4 Pooling layer

On of the most used technique to reduce the feature maps dimensionality is Pooling technique. Moreover, the Pooling technique is a powerful operation for various types of distortion, it has also a shifting and rotation invariant property. To downsample the resulted feature maps, in overall Max-Pooling operation is applied, which takes the maximum value of each non-overlapping square (i.e. subregion) of feature map. Pooling operation reduces the dimensionality of each Convolved feature maps and in the same time it summarizes the most important information by keeping the maximum activated components. There are many type of Pooling: Max, Average, ...etc. The Max-Pooling operation output K_{ij} is computed as follows:

$$K_{i,j} = \max_{h} \; Z_{i+h,j+h} \tag{3.12}$$

Where $h \in \mathbb{N}$ is the size of sub-region, $i, j \in \mathbb{N}$ are the stride values for the vertical and horizontal axes respectively.

The following definition explains the relationship that controls the size of the Pooling operation output:

Relationship 3.3 *A Pooling output O^{L+1} of layer L+1, is described by Input I^L, Pooling Window W^L, Stride S^L:*

$$O^{L+1} = \frac{I^L - W^L}{S^L} + 1 \qquad \forall O, S, I, W \in \mathbb{N} \tag{3.13}$$

Example: Let's consider an Input I = 4 × 4, Pooling Window W = 2 × 2, Stride S = 2, thus, the output O of the Pooling operation in this case: O = ((4 - 2)/2) + 1 = 2, *so the Pooling ouput size equals to* 2×2*, see figure 3.8 for more details.*

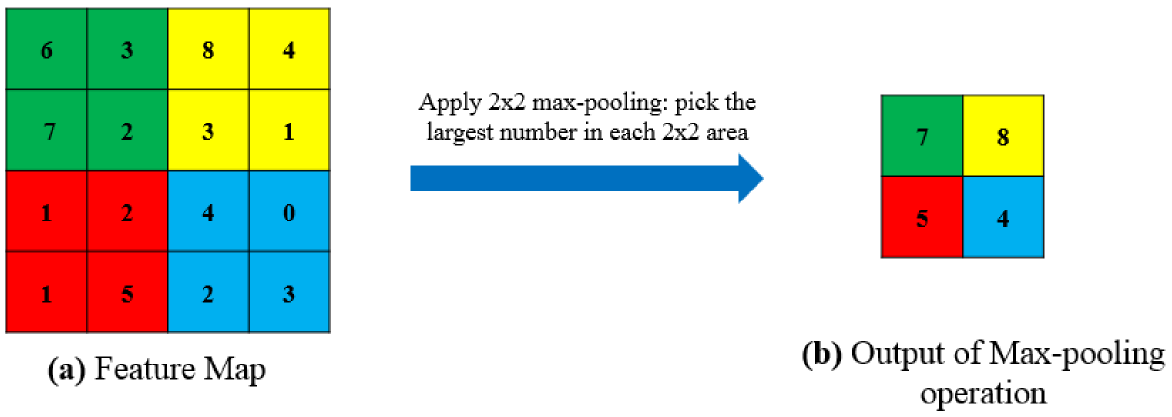

6
3
8
4
7
2
3
1
1
2
4
0
1
5
2
3
(a) Feature Map
Apply 2x2 max-pooling: pick the largest number in each 2x2 area
7
8
5
4
(b) Output of Max-pooling operation

Figure 3.8 – An example of max-pooling operation. (**a**) is the output of convolution operation, (**b**) is the result after applying 2×2 max-pooling operation. In typical CNN architectures, they use a window of 2×2 (which is most common) because it provides the minimum spatial reduction of feature maps.

In this example (see figure 3.8), we shift 2×2 Pooling window by two cells (called stride) over the Rectified Feature map then we take the maximum value in each region then we shift again the window to the right. We repeat this process until the Pooling windows slides all the position of the feature maps. In typical CNNs architectures [Krizhevsky et al., 2012; Simonyan and Zisserman, 2014], they apply Pooling operation to the output of the Convolution operation (Feature maps).

3.2.1.5 Fully connected layers

In typical CNN architectures [Krizhevsky et al., 2012; Simonyan and Zisserman, 2014], the output of the pooling layer will be flattened to act afterward as an input for the Fully Connected layers (FC). The FC layers are just a Multi-Layer Perceptron (MLP) with three types of layers (see figure 3.9): input, hidden, and output layers. The input layer (blue) is the flattened vector of features which can be integer or float vector. The hidden layers (green) extract more representative features by transforming them from non-linear space representation into another space representation with more compact and discriminative features. The first hidden layers extract lower-level features such as lines, edges while the last hidden layers extract higher-level representative features such as objects, shapes, faces,...etc. The output layer (purple) embeds a classifier at the end to classify the extracted features from the input to one of the predefined classes, in our case for the issue of Glioblastoma brain tumors segmentation, there are 4 classes: edema, non-enhancing core and necrotic core, enhancing core, and healthy tissue. The classifier can be Softmax (see equation 3.16), Sigmoid[10] (see equation 3.17) or a traditional machine learning classifier such as Support Vector Machine (SVM). The embedded functions in the classifier should have an important property that squashes the outputs to be between 0 and 1 for classification issue, i.e., the neural networks output the posterior probabilities $z = p(y = i|x; \theta)$ with z ∈ [0,...,1], where z is the output of the classifier. After computing the vector of probabilities of the output layer, we assign the winning or the most probably true class C with the one with the highest probability. For a j^{th} neuron in MLP network, the neuron's output is computed as follows:

$$s = \sigma(Wx + b) \tag{3.14}$$

Where $s \in \mathbb{R}$ is the neuron's output, x is the input, W $\in \mathbb{R}^{f\times h}$ is the *weight matrix*, b $\in \mathbb{R}^{h}$ is the *bias* vector. The latter (bias) is considered as the intercept added to the activation function. σ is the activation function (see section 3.2.1.2) for further details. Usually, in deep learning networks, Relu function and its variants such as leaky Relu, ELU,...etc, are used as an activation function of hidden layers. After computing the outputs of one layer, we can now increase the *depth*[11] by stacking more hidden layers as a composition of functions in order to extract more discriminative and complex features as follows:

$$\begin{aligned} \sigma_3 \circ \sigma_2 \circ \sigma_1(Wx + b) &= \sigma_3(\sigma_2(\sigma_1(Wx + b))) \\ &= \sigma_3(W_3\sigma_2(W_2\sigma_1(W_1x + b_1) + b_2) + b_3) \end{aligned} \tag{3.15}$$

Where $\circ$ is a composition of functions σ_d. W_d is the *weight matrix* of the layer *d*. b_d is the *bias* vector of the layer *d*. According to *universal approximation theorem*, [Cybenko, 1989] proved that any neural network with one hidden layer and *Sigmoidal*[12] activation function can *approximate any continuous function*. Therefore, the composition of many non-linear activation (i.e., many hidden layers) functions has not an impact on neural network theory and it provides only a desirable response to each type of feature.

10. One neuron with Sigmoid function at the end of the classifier is used in traditional applications for binary classification, and therefore, it deals with linearly separable issues but it is not capable to work with non-linear separable issues.
11. Deep neural network is any neural network with more than one hidden layer.
12. Any function with curve similar to Sigmoid and Tanh activation functions.

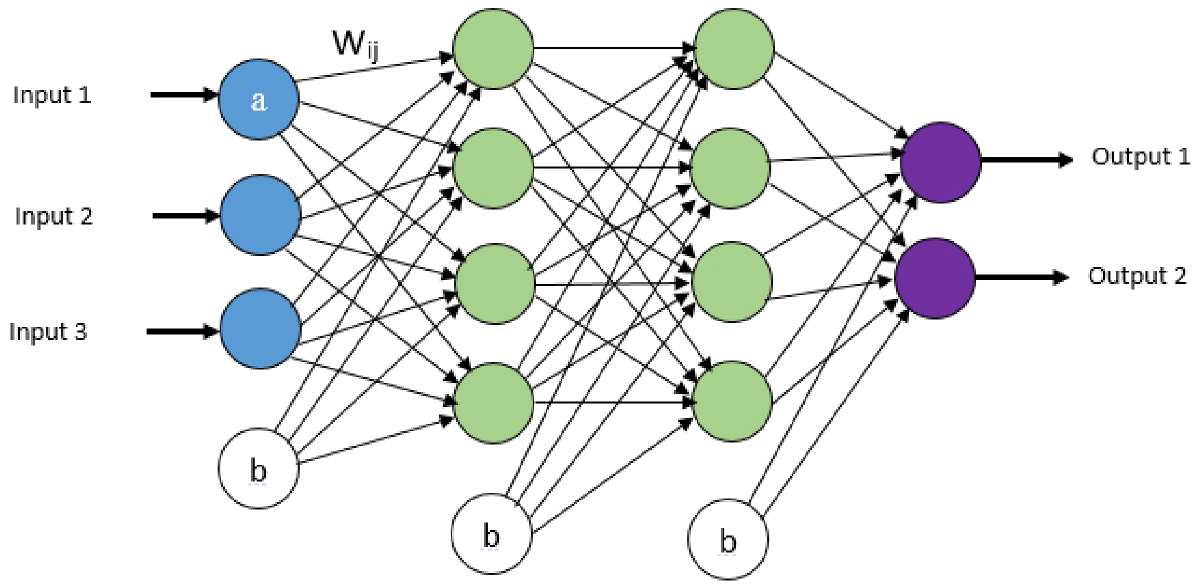


Figure 3.9 – Multilayer perceptron architecture (MLP). MLP has three types of layers: input (blue), hidder (green), output (purple). ”a” is a neuron, ”w” is an weight between input layer and first hidden layer, and ”b” is a bias parameter.

$$Softmax(x)_i = p(y=i|x;\theta) = \frac{e^{x_i}}{\sum\limits_{j=1}^{N} e^{x_j}} \qquad i \in [C_0, ..., C_N] \tag{3.16}$$

$$Sigmoid(x) = p(y=i|x;\theta) = \frac{1}{1+e^{-x}} \qquad i \in [C_0, C_1] \tag{3.17}$$

Where $x \in \mathbb{R}$ is the input to the classifier. i represents one of the N classes C_i in the dataset, and θ is the parameters of the neural networks. The condition $(y|x;\theta)$ of the probability $p(y|x;\theta)$ that the prediction y is i, given x and θ, in the case of Sigmoid function is subject to Bernoulli distribution:

$$\begin{aligned} p(y=C_0|x;\theta) &= g(x) \\ p(y=C_1|x;\theta) &= 1-g(x) \end{aligned} \tag{3.18}$$

Where $g(x) \in \mathbb{R}$ is the the posterior probability. The two equations of 3.18 can be written as follows:

$$p(y|x;\theta) = g(x)^y (1-g(x))^{1-y} \tag{3.19}$$

The equation 3.19 is able to estimate the contribution of neural networks’ parameters on the final results. We can estimate the value of the parameters (weights and biases) by maximizing the likelihood of the neural networks' parameters, and therefore, it is equivalent to minimizing the loss [13] function such as *cross-entropy*, *hinge loss* [Rosasco et al., 2004],...etc, which is done usually in deep learning networks by gradient-based algorithms such as *stochastic gradient descent* and its variants.

3.2.2 Forward Propagation

In Deep Neural Networks family, including Convolutional Neural Networks, there are two important algorithms, i.e., forward propagation and backward propagartion [14]. The first algorithm (Forward Propagation [15]) allows deep learning architectures to compute and predict the probability $p(y = C|x)$ that a class "C" is true. At the phase of the training, a training set contains a pair of input observation $X_{1:m}$ and corresponding target $Y_{1:n}$ of a dataset $(X_{1:m}, Y_{1:n})$, where the class "C" $\in [C_1, ..., C_n]$. The objective of Forward algorithm at the trainin phase is propagate the data X from the input layer to the hidden layers then to the output layer and to calculte the error "E" relative to the desired output Y using a differentiable Loss function L (also known as cost, or objective function). For example for a regression problem, the Mean Squared Error (MSE) function is computed as follows (see equation 3.20):

$$L_{MSE}(y, o)_i = \frac{1}{n}\sum_{i=1}^{n} \parallel y_i - o_i \parallel^2 \tag{3.20}$$

In a binary classification problem, where the number of classes is 2 (i.e., $C = 2$), we use a binary cross-entropy [16] (CE) (also CE is known as Logistic Loss, log-loss, and Multinomial Logistic Loss.) loss function as follows:

$$\begin{aligned} L_{CE}(y, o) &= -\sum_{i=1}^{C=2} y_i \log(o_i) \\ L_{CE}(y, o) &= -(y_1 \log(o_1) + y_2 \log(o_2)) \\ L_{CE}(y, o) &= -(y_1 \log(o_1) + (1 - y_1)\log(1 - o_1)) \end{aligned} \tag{3.21}$$

Where "o" is deep learning model prediction, "y" is the ground truth for each class i in

13. The loss function is an function with one or more variables to measure the efficiency of a model.
14. See section 3.2.3 for further details.
15. Forward propagation is used in many methods such as hidden Markov model to compute the joint probability $p(y_t, x_{1:t})$ at each time step, given the history of evidence, where y_t is the hidden state, and $x_{1:t}$ are the observations 1 to t.
16. Cross-entropy is a function that computes the difference between true and estimated distributions.

C, i.e, "i" $\in [1, ..., C]$, "o_1" and "y_1" are the prediction and ground truth of C_1, therefore, "o_2" $= (1 - o_1)$ and "y_2" $= (1 - y_1)$ for C_2. In a binary classification problem, "o" is the output of either softmax or sigmoid functions (see equation 3.16 or see equation 3.17 respectively).

In a multi-class classification problem, where the number of classes could be more than two classes (i.e., $C > 2$):

$$L_{CE}(y, o) = -\sum_{i=1}^{C} y_i \log(o_i) \tag{3.22}$$

Usualy, in a multi-class classification problem, one-hot encoding is used to encode the labels (Ground truth) over the C classes, i.e., this encoding gives all probabilities of the ground truth to one class (i.e., the correct class), and the other classes become zero, e.g. q = [1,0,0,0], this vector q indicates that the first class is the correct class, in this case, the equation 3.22 can be written as follows (see equation 3.23):

$$\begin{aligned} L_{CE}(y, o) &= -\log\left(\frac{e^{o_p}}{\sum_j^C e^{x_j}}\right) \\ L_{CE}(y, o) &= -\log(e^{o_p}) + \log\sum_j^C e^{x_j} \end{aligned} \tag{3.23}$$

Where L_{CE} is cross-entropy loss functions of the output "i" $\in [1, ..., C]$, "y" is the real value (Ground truth), "o" is the prediction/approximation of the Neural Networks (o_p is the prediction for the positive class). Moreover, the role of the loss function is to calculate the distance between the Ground truth and the prediction value over the overall training set, i.e., how much the prediction gives value simulates the Ground truth for all images data.

If Neural Networks have more than one output, in this case, the total loss function is the sum of all error values of all outputs (see equation 3.24):

$$L_{total} = \sum_{i=1}^{C} E_i = E_1 + E_2 + ... + E_C \tag{3.24}$$

Where E_i is the error for the i^{th} output, with "i" $\in [1, ..., C]$. For the Neural Networks in figure 3.9, the total loss function of the equation 3.20 for a regression problem can be written as follows (see equation 3.25):

$$L_{total} = \sum_{i=1}^{2} E_i = E_1 + E_2 = \frac{1}{2}(y_1 - o_1)^2 + \frac{1}{2}(y_2 - o_2)^2 \tag{3.25}$$

In Neural Networks, if the input data is propagated in batches[17] b with $b \in [1, ..., n]$, in this case, the total loss function L_{total} is the sum of losses of all batches (see equation 3.26):

$$L_{total,b_{1:n}} = \sum_{i=1}^{n} L_{bi} = L_{b1} + L_{b2} + ... + L_{bn} \tag{3.26}$$

The forward algorithm is able to estimate the value of the error in each iteration through a loss function dedicated to each task (e.g, regression, classification), in addition, to producing the inference of input to obtain an output.

3.2.3 Backward propagation

The second important algorithm in Deep Neural Networks is Backward algorithm (also known as Backpropagation). The objective of Backward algorithm is to compute the new parameters (i.e., weights, and biases) associated with each layer and connection between every two successive neurons, i.e, the objective is the find the best weights ”W” and biases ”b” that give the best prediction ”o” over all images.
The algorithm of Backpropagation applies chain rule[18] to calculate the gradient of the error from the output layer to the input layer. The chain rule is computed as follows:

$$\begin{aligned} &if z = f(x) \;\; and \;\; x = g(t) \;\; and \;\; y = h(t) \;\; then \\ &\frac{dz}{dt} = \frac{dz}{dx}.\frac{dx}{dt} + \frac{dz}{dy}.\frac{dy}{dt} \end{aligned} \tag{3.27}$$

Where $\frac{dz}{dt}$ is the derivative of z with respect to t. In Neural Networks, each layer is a function of its previous layer, thus, to compute the derivative of a parameter ”W”, we need to compute the derivative of all parameters before ”W”.

Updating the weights and biases for many epochs leads to minimizing th error between

17. In programming frameworks such as Tensorflow and Keras,...etc, the data is propagated through Networks using a variable called batch size.
18. Chain rule is a technique to compute the derivative of a composite function

the predicted output and real value, thus, to compute the new parameters, we need to propagate the gradient of the error using the chain rule method. To obtain the best parameters of W_{31}, W_{32}, b (see figure 3.10), we need first to compute the gradient of the loss function, second we need to propagate the gradient back from the output layer to

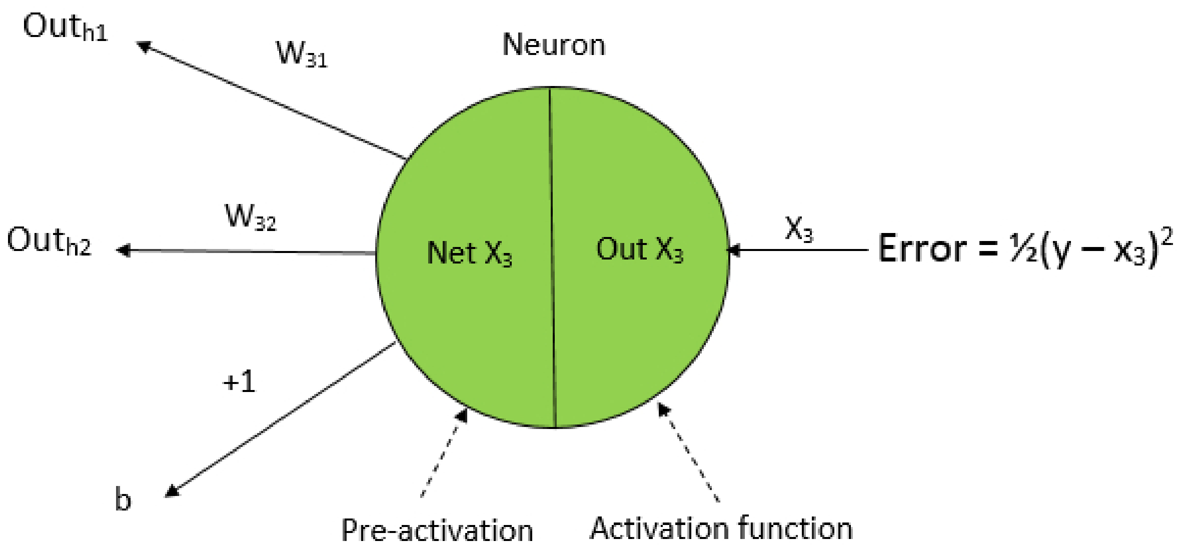


Figure 3.10 – Example of propagating the gradient of the error through a network of one neuron. *W*, *b* are the the weight and bias, *Net X* is the pre-activation function, *Out X* is the output of the neuron, *Out h* is the input.

the input layer. The partial derivative $\frac{\partial L_{total}}{\partial W_{31}}$ (also known as the gradient) of the total error L_{total} (See section 3.2.2) with respect to weight W_{31} is computed as follows:

$$\frac{\partial L_{total}}{\partial W_{31}} = \frac{\partial L_{total}}{\partial X_3} * \frac{\partial X_3}{\partial NetX_3} * \frac{\partial NetX_3}{\partial W_{31}} \tag{3.28}$$

Where X_3 is the neuron's output, $NetX_3$ is the pre-activation function. We have $X_3 = OutX_3$ the predicted output, therefore:

$$NetX_3 = W_{31} \times Out_{h1} + W_{32} \times Out_{h2} + b \tag{3.29}$$

$$OutX_3 = Sigmoid(NetX_3) = \frac{1}{1 + e^{-NetX_3}} \tag{3.30}$$

Backpropagation algorithm uses gradient-based algorithms to adjust the parameters (i.e., weights and biases), where it works basically by moving the parameters in the opposite direction [19] of the gradient at learning rate η.

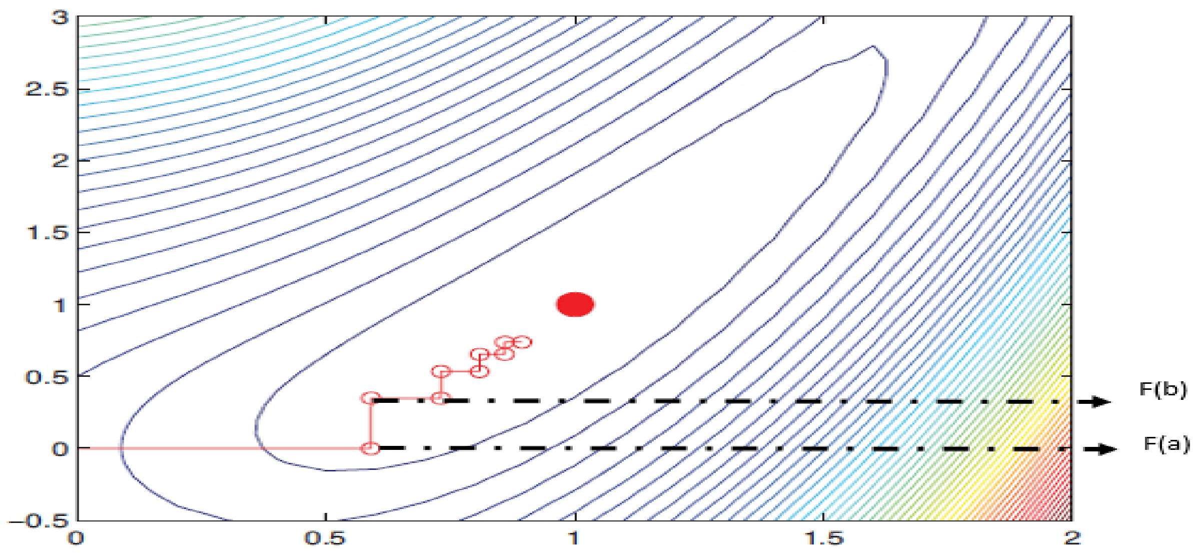


Figure 3.11 – Visualization of the gradient descent trajectory towards the global minimum (the red point). The principal axes are the parameters (e.g. weight 1 and weight 2). F(a), F(b) are the starting point and the second best parameters, respectvely. At the global minimum point, the model gets the best parameters among all iterations. Based on figure 8.3 [Murphy, 2012].

From the figure 3.11, computing the second best parameters of F(a) is done as follows (see equation 3.31):

$$F(b) = a - \eta \nabla F(a) \tag{3.31}$$

Where ∇ F(a) is the derivative of F(a) with respect to "a", and η is the learning rate which represents the step size between two successive updates.

For the example in figure 3.10, the new weight W_{31}^{+} is computed as follows:

$$\begin{aligned} W_{31}^{+} &= W_{31} - \eta(\frac{\partial L_{total}}{\partial W_{31}}) \\ W_{31}^{+} &= W_{31} - \eta(\frac{\partial(\frac{1}{2}(y_i - o_i)^2)}{\partial W_{31}}) \quad , o_i = X_3 \\ W_{31}^{+} &= W_{31} - \eta(\frac{\partial(\frac{1}{2}(y - X_3)^2)}{\partial W_{31}}) \end{aligned} \tag{3.32}$$

Where $\frac{\partial L_{total}}{\partial W_{31}}$ is the partial derivative of the total error L_{total} (See section 3.2.2) with respect to weight W_{31}, η is the learning rate, X_3 is the output of the neuron, y is the

19. In practice, we do not expect to reach a global minimum.

ground truth (also is known as the desired output). In a multi-class classification problem, i.e. we have many output neurons, therefore, computing the gradient with respect to the positive class (i.e., the correct class) is done as follows:

$$\begin{aligned}
\frac{\partial L_{total}}{\partial o_p} &= \frac{\partial(-\log(\frac{e^{o_p}}{\sum_j^C e^{x_j}}))}{\partial o_p} \\
\frac{\partial L_{total}}{\partial o_p} &= \frac{\partial(\log\sum_j^C e^{x_j} - \log(e^{o_p}))}{\partial o_p} \\
\frac{\partial L_{total}}{\partial o_p} &= \frac{\partial \log\sum_j^C e^{x_j} - \partial\log(e^{o_p})}{\partial o_p} \\
\frac{\partial L_{total}}{\partial o_p} &= \frac{1}{\sum_j^C e^{x_j}}\frac{\partial\sum_j^C e^{x_j}}{\partial o_p} - \frac{\partial\log(e^{o_p})}{\partial o_p} \\
\frac{\partial L_{total}}{\partial o_p} &= \frac{e^{o_p}}{\sum_j^C e^{x_j}} - \frac{\partial\log(e^{o_p})}{\partial o_p}
\end{aligned} \tag{3.33}$$

Where $x_j \in \mathbb{R}$ is the output of the j^{th} neuron with $j \in [1, ..., C]$, $\frac{\partial L_{total}}{\partial o_p}$ is the partial derivative of the total error with respect to the positive class, o_p is the CNNs prediction for the positive class. In a multi-class classification problem, computing the gradient with respect to the other classes o_n (negative classes) is done as follows:

$$\begin{aligned}
\frac{\partial L_{total}}{\partial o_n} &= \frac{\partial(-\log(\frac{e^{o_p}}{\sum_j^C e^{x_j}}))}{\partial o_n} \\
\frac{\partial L_{total}}{\partial o_n} &= \frac{e^{o_n}}{\sum_j^C e^{x_j}}
\end{aligned} \tag{3.34}$$

Where $x_j \in \mathbb{R}$ is the output of the j^{th} neuron with $j \in [1, ..., C]$, $\frac{\partial L_{total}}{\partial o_n}$ is the partial derivative of the total error with respect to the negative class, o_n is the CNNs prediction for the negative class.

For the example in figure 3.12, where Sigmoid (See equation 3.17) is applied as an activation function [20], the derivative of the error E with respect to weight W_3 is computed as follows:

$$\begin{aligned}
\frac{\partial L_{total}}{\partial W_{31}} &= \frac{\partial L_{total}}{\partial X_3} * \frac{\partial X_3}{Net_3} * \frac{\partial Net_3}{\partial W_{31}} \\
\frac{\partial L_{total}}{\partial W_{31}} &= (y - X_3) * X_3(1 - X_3) * X_2
\end{aligned} \tag{3.35}$$

20. The important property of the activation function is to be differentiable, therefore, the backpropagation algorithm can flow the gradient of the loss function back to update and correct the parameters.

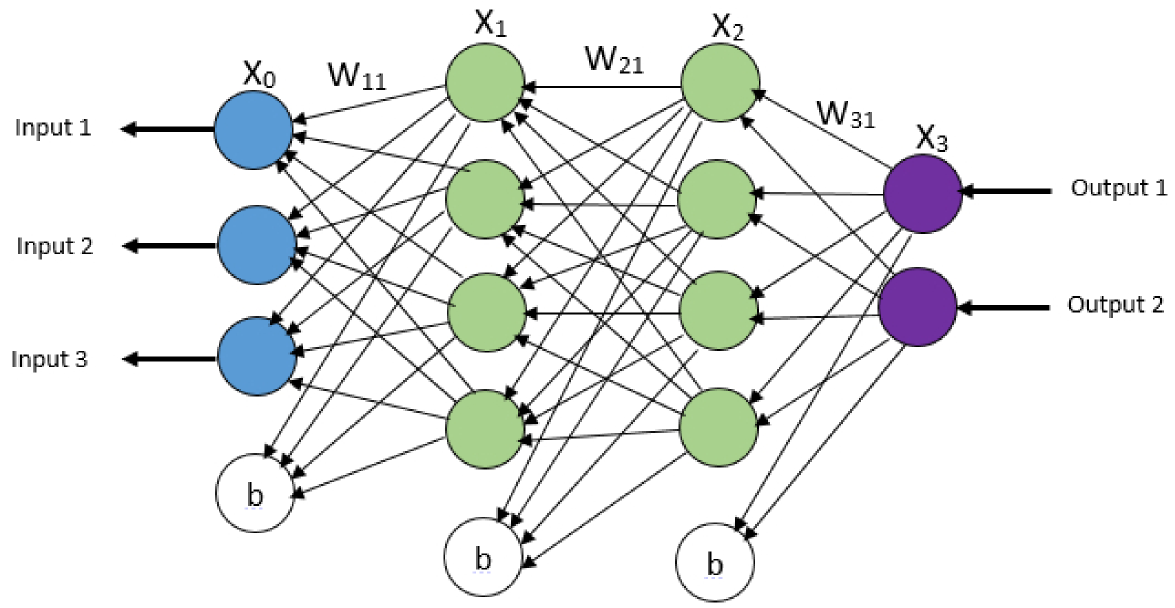


Figure 3.12 – Example of the backpropagation algorithm. Backpropagation computes the gradient of the objective function with respect to the parameters, then it backpropagates this gradient from the last layer to the first layer.

The derivative of the error E with respect to weight W_2 is computed as follows:

$$\begin{aligned}\frac{\partial L_{total}}{\partial W_{21}} &= \frac{\partial L_{total}}{\partial X_2} * \frac{\partial X_2}{Net_2} * \frac{\partial Net_2}{\partial W_{21}} \\ \frac{\partial L_{total}}{\partial W_{21}} &= (\frac{\partial L_{total}}{\partial X_3} * \frac{\partial X_3}{\partial Net_3} * \frac{Net_3}{\partial X_2}) * X_2(1-X_2) * X_1 \\ \frac{\partial L_{total}}{\partial W_{21}} &= (\frac{\partial L_{total}}{\partial X_3} * \frac{\partial X_3}{\partial Net_3} * W_{31}) * X_2(1-X_2) * X_1\end{aligned} \tag{3.36}$$

The derivative of the error E with respect to weight W_1 is computed as follows:

$$\begin{aligned}\frac{\partial L_{total}}{\partial W_{11}} &= \frac{\partial L_{total}}{\partial X_1} * \frac{\partial X_1}{\partial Net_1} * \frac{\partial Net_1}{\partial W_{11}} \\ \frac{\partial L_{total}}{\partial W_{11}} &= (\frac{\partial L_{total}}{\partial X_2} * \frac{\partial X_2}{\partial Net_2} * \frac{Net_2}{\partial X_1}) * X_1(1-X_1) * X_0 \\ \frac{\partial L_{total}}{\partial W_{11}} &= (\frac{\partial L_{total}}{\partial X_2} * \frac{\partial X_2}{\partial Net_2} * W_{21}) * X_1(1-X_1) * X_0\end{aligned} \tag{3.37}$$

Where $\frac{\partial L_{total}}{\partial Z}$ is the partial derivative of the total error with respect to Z, Net_j is the pre-activation function of the layer j, X_j is the output of neuron of the layer j. Therefore, the new weights $W_{31}^{+}, W_{21}^{+}, W_{11}^{+}$ are computed as follows:

$$\begin{aligned}W_{31}^{+} &= W_{31} - \eta(\frac{\partial L_{total}}{\partial W_{31}}) \\ W_{31}^{+} &= W_{31} - \eta((y - X_3) * X_3(1-X_3) * X_2)\end{aligned} \tag{3.38}$$

$$\begin{aligned} W_{21}^{+} &= W_{21} - \eta(\frac{\partial L_{total}}{\partial W_{21}}) \\ W_{21}^{+} &= W_{21} - \eta((\frac{\partial L_{total}}{\partial X_3} * \frac{\partial X_3}{\partial Net_3} * W_{31}) * X_2(1 - X_2) * X_1) \end{aligned} \tag{3.39}$$

$$\begin{aligned} W_{11}^{+} &= W_{11} - \eta(\frac{\partial L_{total}}{\partial W_{11}}) \\ W_{11}^{+} &= W_{11} - \eta((\frac{\partial L_{total}}{\partial X_2} * \frac{\partial X_2}{\partial Net_2} * W_{21}) * X_1(1 - X_1) * X_0) \end{aligned} \tag{3.40}$$

Where $\frac{\partial L_{total}}{\partial Z}$ is the partial derivative of the total error with respect to Z, W_{ij}, W_{ij}^{+} are the old weight and new weight of connection between layer i and layer j, respectively.

The key operation of the backward[21] algorithm is to compute the gradient of the objective function with respect to some parameters, which is obtained by an application of the chain rule technique. Moreover, the backward algorithm has the ability to estimate the next best parameters of Deep Neural Networks based on the previous parameters.

3.2.4 Regularization Dropout

One of the known issues in Deep learning models is *overfitting*, where the model gives a much better prediction on training dataset compared to test dataset, this issue is known as *data fitting-generalization tradeoff*, in other words, the model has learned patterns specific to training dataset but are irrelevant to test dataset. Generally, the issue of *overfitting* is due to either using a very complex model with a large number of parameters (i.e., weights and biases) or training a model with a small dataset. To prevent Deep learning models from *overfitting*, the best solution after getting more training data is using the technique of *Dropout* [Srivastava et al., 2014]. This technique is based on "dropping out" randomly some neurons (see figure 3.13), therefore, this technique eliminats the *dependence* among neurons and encourags the creation of *specialised neurons*, e.g., for a layer's output vector q = [0.4, 0.5, 0.2, 0.1, 0.6, 0.3]. And after applying *Dropout* technique with a "dropout rate" R = 0.5, i.e., we randomly omit each element in the vector q with probability R = 0.5, so the vector q becomes *q'* = [0.4, 0, 0, 0.1, 0, 0.3]. Therefore, the use of *Dropout* technique is equivalent to random sampling from 2^N different Neural Networks that share weights. *Dropout* is a much better *regularization* technique than the penalties L_1 or L_2

21. The most interesting thing about backward is its ability to be used in an End-to-End fashion to train large and Deep Neural Networks.

that drag the weights towards zero. At test phase, the values of the neurons of each layer will be scaled down by a factor equals $\frac{1}{1-R}$.

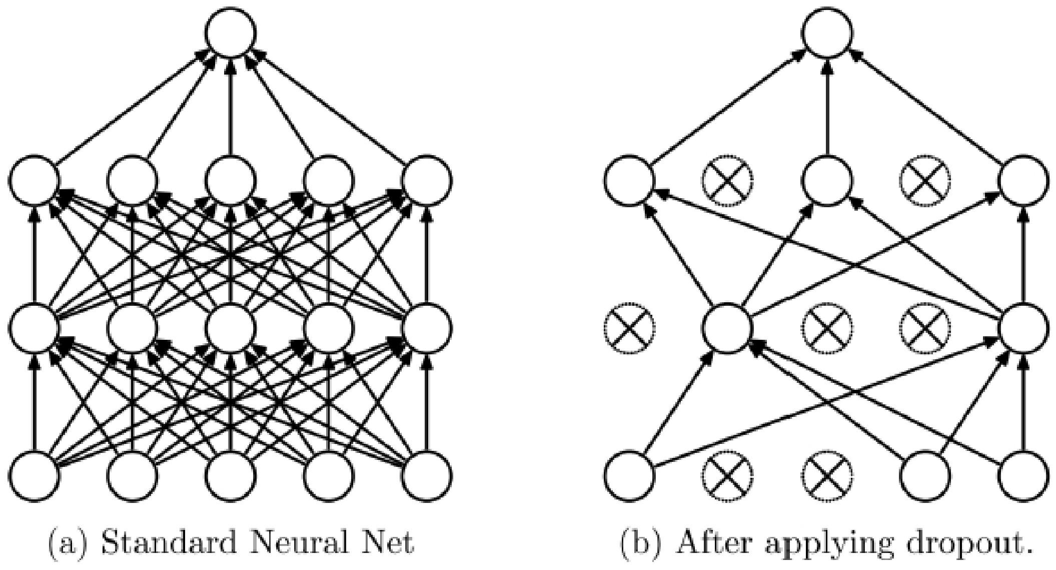


Figure 3.13 – Example of Dropout technique with different "dropout rates" for each layer. **Left**: is a fully-connected neural network. **Right**: is the same fully-connected neural network, where we randomly eliminate each hidden unit with a predefined probability. Figure from [Srivastava et al., 2014].

3.3 Discussion and Conclusion

In this chapter, we presented Deep Neural Networks, in particular, Convolutional Neural Networks, in addition to the fundamental building blocks of CNNs algorithm, and the operations behind this algorithm such as convolution, non-linear activation function, pooling, upsampling, and fully-connected layers.

The formalism of adding the convolution stage with a bank of filters to extract hierarchical features instead of using many blocks of features engineering as in classical machine learning methods in addition to transforming lower-level feature maps into higher-level feature maps and verse versa and reducing feature maps dimensionality using pooling operations represents the important property behind CNNs. Dropout regularization technique helps Deep learning, in particular, CNNs to prevent from the issue of overfitting. Consequently, the use of the Dropout regularization technique allows CNNs architectures to incorporate a large number of filters and neurons without raising the issue of Overfitting. Additionally, Dropout is considered as the easiest way to learn many different Deep Neural Networks in parallel, therefore it helps to extract a lot of knowledge from the training data.

Finally, the success of Convolutional Neural Networks, is due to 3 important properties: the availability of large and public datasets, learning methods, and frameworks that take advantage to parallelize CNNs operations. The fundamental building blocks such as convolution for features extraction over the input data, pooling to reduce the dimensionality of feature maps, Upsampling to transform features maps from lower-resolution to higher-resolution representation allow CNNs architectures to develop a feature extractor of hierarchical representation with more compact and discriminative features without the need of features engineering as in classical machine learning methods. However, despite the advantage of CNNs architectures, many drawbacks have appeared with these architectures such as, firstly, the structure of the architecture, the hyperparameters and parameters (weight and bias). It is interesting to note that Cybenko's theorem proof [Cybenko, 1989] does not define how many neurons can be included in the hidden layer or how to obtain the hyperparameters. In the next chapter, we propose two algorithms to solve and answer questions related to the structure of the architecture, hyperparameters and parameters. Secondly, the development of very deep architectures leads in most cases to the issue of vanishing gradient. To solve this issue, we propose a new methodology to train a Deep Learning architecture, which involves hyperparameters in addition to learning methods in one learning graph.

Chapter 4

Brain Tumor Segmentation with Deep Neural Networks

Contents

4.1 Introduction

In this chapter, we present the issue of developing an efficient brain tumor segmentation model using Multi-sequence MRI images, where getting an efficient and accurate prediction of Glioblastoma brain tumors as soon as possible gives an early clinical diagnosis, treatment, and follow-up. Glioblastoma tumors have four main properties: different sizes, shapes, contrasts, besides, these tumors appear anywhere in the brain. Thus, detecting Glioblastoma brain tumors in MRI images is a challenge and needs to take into account all the properties of Glioblastoma tumors. To study Glioblastoma tumors, in this chapter, we used BRATS dataset which has patients' MRI images with Glioblastoma tumors. Moreover, BRATS dataset has the largest public MRI data for Glioblastoma tumors. Each patient's brain MRI image comes with 4 MRI sequences (i.e. T1, T1c, T2, FLAIR) and the ground truth labels of 4 segmentation labels which are obtained manually by radiologists: healthy tissue and background (class 0), non-enhancing core (class 1), edema (class 2) contrast-enhancing core (class 3). Furthermore, ground truth labels are created by radiologists and approved by expert board-certified neuroradiologists. Although a large number of MRI data in this dataset, few medical image analysis works have been carried out to study these tumors. This chapter aims to develop new Convolutional Neural Networks architectures for the segmentation of Glioblastoma brain tumors. The proposed architectures are used to segment the brain tumors of Glioblastoma with both high- and low-grade. We propose in section 4.2 the first approach for brain tumor segmentation of Glioblastoma, which takes into account 2 main issues: how to develop efficient Convolutional Neural Networks architectures, in addition to a new methodology of training to solve the issue of vanishing gradient. In section 4.3, we propose a second approach that is an inspired architecture by the function and structure of the visual cortex to improve the first approach and to provide better results in terms of segmentation performance and computational benefits, in addition to proposing two steps of post-processing.

4.2 End-to-End incremental Deep learning

4.2.1 Problem statement

During the last three decades, the problem of brain tumor segmentation has attracted many researchers, in which the number of huge published papers as noted by [Menze et al., 2014] involving brain tumor segmentation increased exponentially. These statistics indicate that the creation of a smart brain tumor segmentation system has always been a highly needed option. The motivation behind the need to create a smart system powered by Deep Learning and CNNs is that instead of wasting time that takes between 3 and 5 hours per patient [Kaus et al., 2001]. With manual diagnosis by radiologists, this smart segmentation system decreases the needed time for the diagnosis process. This way gives radiologists and oncologists more time with their patients in the therapy planning process, especially in the treatment and follow-up stages.

MRI is the investigation tool of choice for the segmentation of brain tumors such as Glioblastoma tumors and the evaluation of treatment response [Bangiyev et al., 2014]. Usually, an expert radiologist uses an MRI scanner to generate a sequence of images (Flair, T1, T1 contrast, T2, ...etc) to identify different regions of the tumors. This variety of images helps radiologists to extract different types of information about the tumor (e.g., shape, volume,...etc).

A large number of studies have been carried out for automating the task of radiologists, i.e., algorithms for the localization and segmentation of the whole tumor and its sub-regions. These algorithms are classified into two categories [Menze et al., 2014]: first *Generative probabilistic models* [Van Leemput et al., 2001; Gering et al., 2002; Prastawa et al., 2004; Seghier et al., 2008] that incorporate *domain-specific prior knowledge* about tumor, appearance, and spatial distribution of the different tissue types. However, the limit of these models is that they require recording several parameters on the features of the tumors in order to obtain an automated image segmentation. In the case of Glioblastoma tumors, they have unpredictable properties to incorporate in *Generative probabilistic models*. *Discriminative models* [Ho et al., 2002; Zhang et al., 2004; Schmidt et al., 2005; Görlitz et al., 2007; Zikic et al., 2012] directly learn a decision boundary and the characteristics of tumors and how to segment it from MRI images using manually annotated data by radiologists without any domain knowledge. Many discriminative models need large training data such as Deep Learning algorithms, in particular, Convolutional Neural networks (CNNs) to extract a large hierarchy of latent feature maps through learned kernels. CNNs network architecture is the most suitable machine learning solution to handle 4 multi-modalities (Flair, T1, T1 contrast, T2) of MRI images for the issue of Glioblastoma tumors due to the ability of CNNs to take into account the correlation between the adjacent voxels, in addition to that images have a spatially correlated nature. But the biggest challenges that we face are: first Glioblastoma tumors have four unpredictable properties: different sizes, shapes, contrasts, moreover, these tumors appear anywhere in the brain. Second, Glioblastoma tumors have four sub-regions [Menze et al., 2014] which are respectively 4 types of intra-tumoral structures: edema, non-enhancing core, and necrotic core, enhancing core, and healthy tissue. Third, Deep Learning and CNNs architectures are computationally expensive[1] and they need a very efficient GPU implementation in addition to a large memory.

The aim of this section is to propose a methodology to develop a fully automatic method for the segmentation of glioblastoma brain tumors using MRI images. To achieve this goal, we propose the following contributions:

1. From the first era of Neural Networks until Deep Learning, there is not a specific method to develop a Neural Network architecture. As previously mentioned in chapter 3, Cybenko's theorem proof [Cybenko, 1989] proved that any neural network with one hidden layer and *Sigmoidal* activation function can approximate any *continuous function* but it does not define the hyperparameters, for example, how many neurons can be included in the hidden layer. Most of nowadays architectures follow *trial-and-error*[2] process which does not use any guided approach to obtain the architecture and hyperparameters. So the first challenging issue in this section is how to develop a strategy that can be used, after that, to develop a CNNs architecture for the segmentation of Glioblastomas brain tumors. To overcome this issue, we propose two algorithms to build, develop, and train in one time efficient CNNs architectures, that take into account the architecture's structure in addition to the choices of hyperparameters (epochs, batch size, layers, optimizer,...etc). Moreover, the proposed training strategy takes into account the most influencing hyper-parameters of Neural Networks, then, by bounding and setting a roof to these hyperparameters we can accelerate the training phase and the process of hyperparameters optimization much better than *trial-and-error*.
2. Solving the issue of vanishing gradient[3] (See figure 4.1), where it is known that the adjustment of the parameters of CNN architecture starts at the output layer

1. AlexNet network architecture [Krizhevsky et al., 2012] with 60 million parameters takes between five and six days to train on two GPUs.
2. The trial-and-error process involves running a model in a loop of all possible configurations of hyperparameters until we get the best model with the suitable hyperparameters.
3. Vanishing gradient issue discovered for the first time in Hochreiter's diploma thesis in 1991.

and ends at the first layers for many iterations until the convergence of the loss function. So, the gradient of the error when it comes to the first layers, it comes with value zero ($\frac{\partial E_{total}}{\partial W} = 0$) because of using a deep architecture with many layers and because of using chain rule; for example *sigmoid* activation fuction squashes the numbers to be between [0,...,1] or *Tanh* numbers between [-1,...,1] (See figure 4.1). For very large values of *sigmoid* (or *Tanh*) the derivative starts to close from zero, therefore, multyplying N small derivatives with M layers leads the gradient to decrease with each layer (for instance taking 0.9 as a weight for a network of 3 layers: 0.9*0.9*0.9 equal 0.7). As it is mentioned, the adjustment of parameters (i.e., weights and biases) using *backprogation* algorithm starts from the last layer and it gets back to the first layer, thus, using many layers makes the gradient of the error signal ”vanish” until it arrives with value equals zero, consequently, and according to the rule of updating the parameters (See equation 4.1), the value of previous parameters (i.e., W and b) equals the value of actual parameters. As a result, the issue of vanishing gradient causes the stopping of training phase, therefore, extracting a new representation of the input observations.

$$
\begin{aligned}
W^{+} &= W - \eta(\frac{\partial E_{total}}{\partial W}) \\
b^{+} &= b - \eta(\frac{\partial E_{total}}{\partial b}) \\
If \quad \frac{\partial E_{total}}{\partial W} &= 0 \quad and \quad \frac{\partial E_{total}}{\partial b} = 0 \quad then: \\
W^{+} &= W - \eta * (0) \\
W^{+} &= W \\
b^{+} &= b - \eta * (0) \\
b^{+} &= b
\end{aligned}
\tag{4.1}
$$

Where $\frac{\partial E_{total}}{\partial p}$ is the gradient of the total error with respect to parameter p (p is weight or bias), W (W^{+}) and b (b^{+}) are trainable parameters (weights and biases respectively), η is the learning rate.

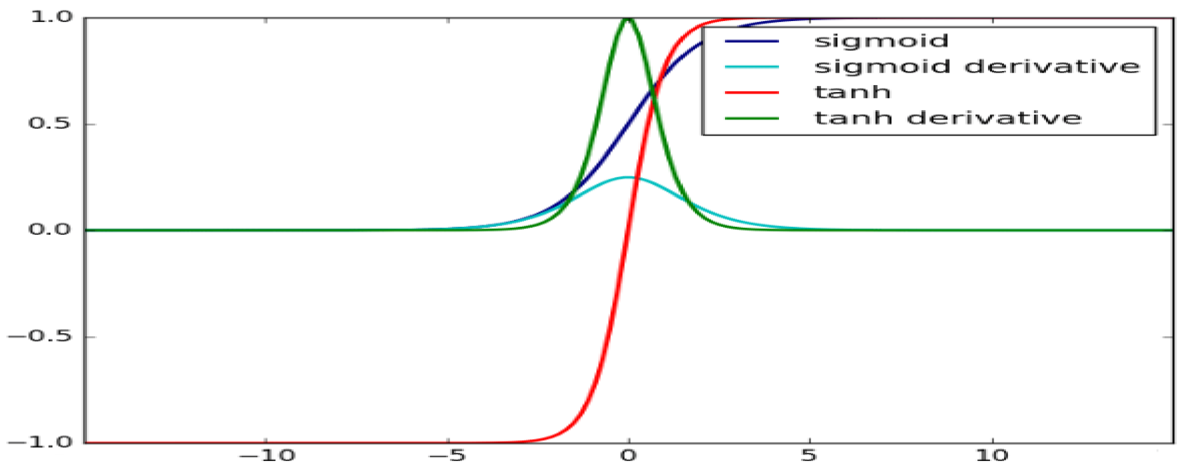


Figure 4.1 – An illustration of non-Linear activation functions: Sigmoid and Tanh with their derivatives. The principal axes are the neuron's input and the output after applying an activation function.

4.2.2 Incremental XCNet Algorithm

As previously mentioned, there is no specific method to develop a CNNs network architecture for the issue of brain tumor segmentation. Indeed, it is so difficult to answer questions related to hyperparameters and architecture's structure of a deep learning model in advance and directly because these questions depend on the type of data and application. Therefore, questions such as the number of neurons in the hidden layers, the number of layers or of batch size...etc are the central question in the development of a CNNs network capable of detecting all glioblastoma brain tumors. consequently, our idea for answering these questions is to develop two capable algorithms that can answer these questions by themselves. First, Incremental XCNet algorithm (See algorithm 1) is based on the development of a generic algorithm; where from a different configuration, it generates a different CNNs architecture. XCNet Algorithm is based on an incremental technique; we create a base model (each base model called block) then we train this block using ELOBA_λ (See algorithm 2) until the block stops learning new relevant features. At this point, we add another block depending on the results and so on until we obtain an efficient architecture. Incremental XCNet algorithm combines the development and training of a deep learning model at the same time. These two algorithms are different from *trial-and-error* process; in the field of machine learning, *trial-and-error* process builds an architecture, then it tries all possible configurations of hyperparameters without any guided approach. Moreover, we have shown through experiments that putting a roof to each of the hyperparameters could avoid the combinatorial explosion problem and more branching in the search space. Furthermore, the use of a *trial-and-error* process leads to high complexity in terms of testing all possible configurations, and perhaps we will not reach to the architecture that has a high performance, especially with low computing resources.

Algorithm 1: Incremental XCNet

```
  Data: X: the number of convolution layers, $S \in \{0, 1\}$: the number of pooling
        layers, F: the number of filters, FC: the number of fully connected layers
  Result: M: a CNNs trained model
1 Creating a block (X , S) and FC: $M_0 \leftarrow (X, S) + FC$ # Creating the first block
2 $M_0 = ELOBA\_\lambda$ ( $M_0$ )  # Running the training using ELOBA_$\lambda$ to adjust the
    Network's parameters
3 while (The new added block is not null) do
4     Removing FC and freeze all layers of $M_{i-1(i \in \mathbb{N})}$ : $M_{i-1(i \in \mathbb{N})} \leftarrow (X, S)$
5     Doubling the number of filters
6     Adding a new block (X , S) and FC: $M_{i(i \in \mathbb{N})} \leftarrow M_{i-1(i \in \mathbb{N})} + (X, S) + FC$
7     $M_{i(i \in \mathbb{N})} = ELOBA\_\lambda$ ( $M_{i(i \in \mathbb{N})}$ )
8 end while
```

The algorithm of Incremental XCNet is based on adding a new block (block has a set of Convolution and Pooling layers) after each training phase. The user sets the first base model (first block) that will be accumulated using the Incremental XCNet algorithm. Then, as long as we go with the training phase, the model starts to evolve over time and adds other blocks to the first block. At the end of the training phase, we obtain automatically another larger model based on the first base model. For example, the first base model is two convolution layers (2C) and one Max-Pooling layer (1P), so the first base model is 2C+1P, after the first iteration of training using ELOBA_λ algorithm (see algorithm 2), the Incremental XCNet algorithm adds another two convolution layers and one Max-Pooling layer to the first base model, so the model becomes 2C+1P+2C+1P, and so on for the next iterations.

The algorithm of Incremental XCNet is related with the algorithm of ElOBA_λ (algorithm 2), in which unlike other CNNs-based models that define the architecture at the beginning then they train it. Here, we focus on the development of a CNNs model to give a high predictive performance and at the same time, designing an optimized architecture in terms of layers and parameters. For achieving these goals, we follow a simple technique:

in algorithm 1: line 1 we create a base model (first block) with a number of Convolutions, i.e., X is the number of Convolution layers and it is predefined at the beginning. Then, in algorithm 1: line 2, we train this base model with the algorithm of ELOBA_λ, the first execution of (algorithm 1: line 1 and line 2), gives $M_0 = (X, S) + FC$, i.e., we have a known number of Convolution and Pooling layers and a number of fully connected layers. Then in algorithm 1: line 4, we remove the fully connected layers (classifier) then we freeze all layers of the last base model M_i (i.e., the parameters inside the layers will not update any more). Then, in algorithm 1: line 5 we double the number of filters, i.e., when the number of features is small (and this number depends on the model's size) then S=0 (see figure 1 (a) block (4)), otherwise S=1. In algorithm 1: line 6 we add a new block and fully connected layers to the last base model. Then, in algorithm 1: line 7, we continue the training using ELOBA_λ, after the first execution of (algorithm 1: line 6 and line 7) we will get $M_1 = M_0 + (X, S) + FC$, i.e., besides M_0 we have a known number of Convolution and Pooling layers and a number of fully connected layers.

All our architectures based on 2D image patches, where these architectures predict the pixel's class which is the center of the 2D patch. In addition, after several experiments on the size of the patch that gives the best result, we found that the size of 32*32 pixels provides good results. The algorithm of Incremental XCNet takes as an input a (4*32*32) patch matrix in which the width*height is (32*32) and the number of channels is four: flair, T1, T1c, T2. The outputs are four different classes: healthy tissue, edema, non-enhancing (solid) core, and necrotic (or fluid-filled) core, enhancing core. Our architectures are designed and trained from scratch based on the general rule that is taken from ILSVRC[4] challenge and from [Szegedy et al., 2015]: every time we add more layers, we obtain high performance models. Moreover, instead of using kernels with a large receptive field, we choose to use small filters (3 * 3). Small filters have proved in [Simonyan and Zisserman, 2014], that they keep a lot of information, and at the same time, they reduce the number of operations. Also, Incremental XCNet Algorithm doubles the number of filters after each pooling layer. The strategy of the Incremental XCNet Algorithm allows CNNs architecture to be deeper in terms of layers, which is needed to detect high level, complicated and hierarchical features. In the next section, we combine

4. ILSVRC: Large Scale Visual Recognition Challenge

the developed architectures in one architecture.

4.2.2.1 Extended Models

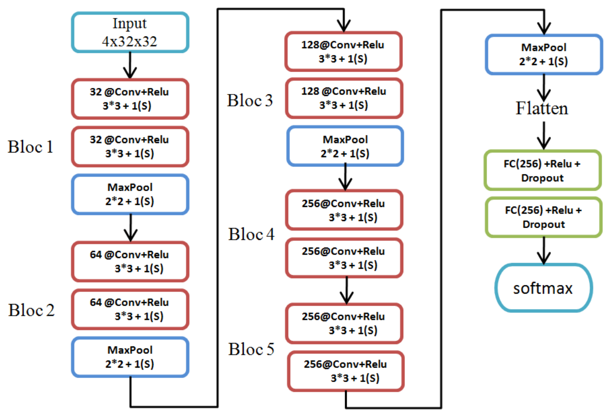
To improve the performance of each architecture, we combine the instance of the algorithm Incremental XCNet: 2CNet (See figure 4.2(a)) and 3CNet (See figure 4.2(b)). Both architecturess (2CNet and 3CNet respectively) produce complete predicted images h and g, each image with size $n{*}n$. Then, next step is image fusion, in which we combine these two images (h and g) to get more accurate result (i.e., the final image). The non-parametric fusion function between g and h is performed using the following equation:

$$f(i,j) = \begin{cases} I_{h(i,j)} & \text{if} \quad I_{h(i,j)} = I_{g(i,j)} \\ min(I_{h(i,j)}, I_{g(i,j)}) & \text{if} \quad I_{h(i,j)} \neq I_{g(i,j)} \end{cases} \tag{4.2}$$

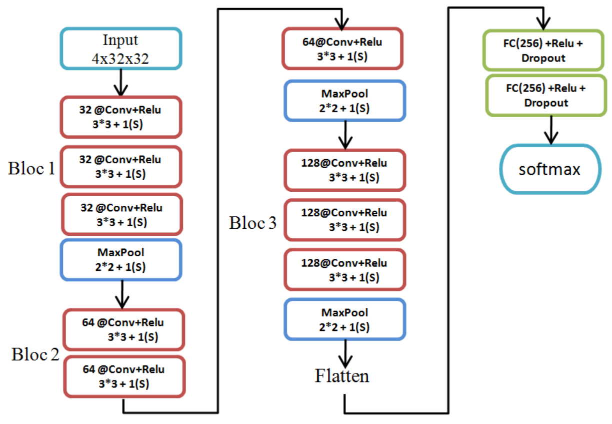
where $I_{h(i,j)}$, (respectively $I_{g(i,j)}$) is the intensity of pixel h(i,j) (respectively g(i,j)), in which $\forall (i,j) \in [1,n]^2$. The combination of Incremental XCNet's instances (i.e, 2CNet and 3CNet) and the fusion function, gives us at the end the EnsembleNet (figure 4.2(c)) model. EnsembleNet aggregates the segmentation results that are obtained from 2CNet and 3CNet models using a non-parametric fusion function. Moreover, the fusion function is used after getting the segmentation results of 2CNet architecture and 3CNet for the following reasons:

— The fusion of 2CNet and 3CNet at the fully connected layers level will lead to obtaining a large architecture with a large number of parameters. Obviously, the resulted architecture, in this case, will require huge computation resources to train it, in addition, it might lead to overfitting due to using a large number of parameters, otherwise, it will take a long time to converge (or to give the prediction results). Thus, the solution is to use the fusion function after getting the segmentation results to minimize the training and the prediction time.

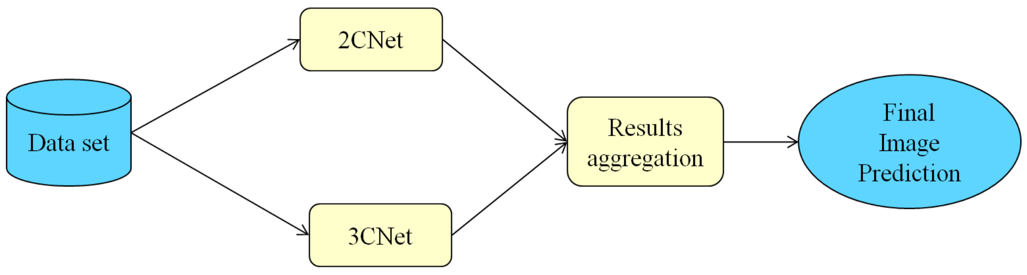
— Glioblastoma tumors have many connected regions (blobs) with different sizes and shapes. Some studies [Urban et al., 2014] propose a global threshold operation as a post-processing to remove these false blobs (i.e., non-tumor region classified as tumor region) which are generated by the architecture but this threshold also removes some small tumor regions. Overall, this threshold could improve the segmentation results, but it is not a good method for patient-specific. For the



a) 2CNet



b) 3CNet



c) EnsembleNet

Figure 4.2 – Schematic representation of our proposed CNNs architectures. (a) is the architecture of 2CNet, (b) is the architecture of 3CNet, (c) is the architecture of EnsembleNet. The symbols used inside the boxes indicate: Conv: convolution, #@: is the number of filters, Relu: is the non-linear function Relu, #*#: is the size of filter (or the size of sub-window in the case of pooling), S: stride, FC: fully connected layers.

example in figure 4.3, patient has 11 tumor regions inside 3D MR volume: 1 region with 20442 voxels, 7 regions with 1 voxels, 1 region with 2 voxels, 1 region with 4 voxels, 1 region with 32 voxels. If we apply a fixed threshold = 3000 voxels as used in [Urban et al., 2014], we will get only the first region. Thus, to address this issue of false blobs automatically we propose a non-parametric fusion function that aggregates the predicted pixels that are true (tumor regions) with a high rate from two deep learning architecture (2CNet and 3CNet).

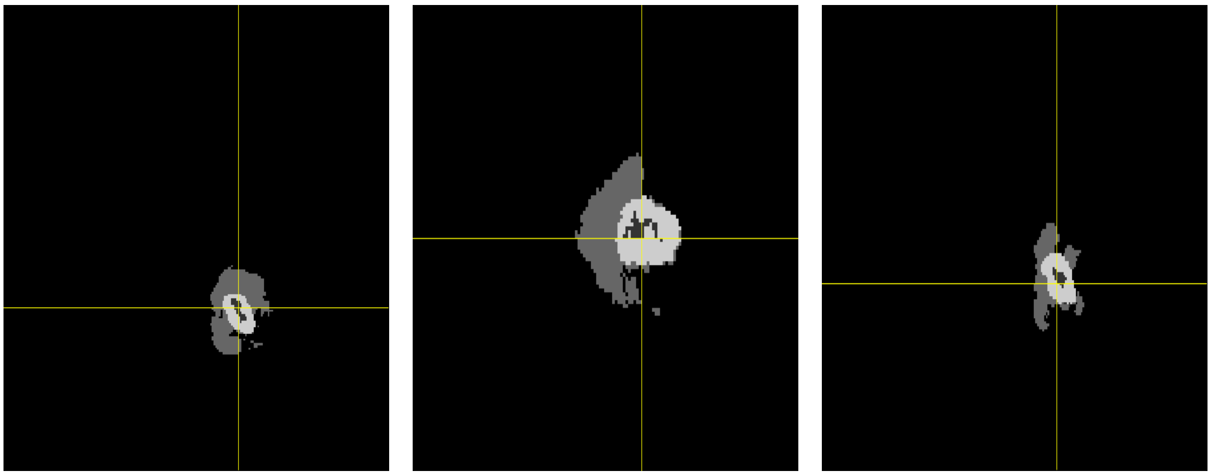


Figure 4.3 – An illustration of an example of patient ID= *Brats17_2013_13_1* MRI image from BraTS 2017 dataset with 3 views from left to right: Axial, Sagittal and Coronal respectively.

4.2.3 ELOBA_λ Algorithm

When investigating the most used methods for training deep learning architectures, it is interesting to note that there is no specific method for training CNN architectures. Users have many hyperparameters that should be selected before starting the training phase of CNNs architectures without any guarantee that this architecture will succeed or fail to converge to the global minimum. For these reasons, we propose a new strategy to unify the training method of CNNs architectures that will be called: ELOBA_λ (E: Epochs, L: Learning rate, O: Optimizer, B: Batch size, A: Architecture, λ is the number of epochs). ELOBA_λ (algorithm 2) is used for training our 2CNet and 3CNet architectures. The algorithm of ELOBA_λ takes into account the most influencing hyperparameters (i.e., Epochs, Batch size, Learning rate, Optimizer). First, we create a base model architecture, then we train (algorithm 2: line 5) this base model. In the next step

Algorithm 2: ELOBA_λ

```
Data: $M$: an untrained CNNs base model, $\lambda$: Epochs, Bs: Batch size
      $\in [bs\_min, bs\_max]$, K: a positive integer , Lr: Learning rate
      $\in [r\_min, r\_max]$, R: a positive number, Op: $\in$ Optimizer = {Adam,
      RMSprop, SGD}
Result: $M$: a trained CNNs base model
1  for ($j = 1 : j \leftarrow j{+}1 : j = 3$) do
2      Op = Optimiser (j)
3      for ($Lr = r\_max : Lr \leftarrow \frac{Lr}{R} : Lr = r\_min$) do
4          for ($Bs = bs\_min : Bs \leftarrow Bs + K : Bs = bs\_max$) do
5              $Training(M, \lambda, Bs, Lr, Op)$ # Running the back-propagation with the
                hyper-parameters ($\lambda$, Bs, Lr, Op)
6              $Dice_t(M)$                # Computing the Dice coefficient
7              while ($Dice_t(M) > Dice_{t-1}(M)$ )    # Where $Dice_{t-1}(M)$ is the Dice
                score of the last iteration, in the first execution ($Dice_0(M)$) is equal
                to 0
8               do
9                  Saving the model $M$ and its parameters
10                 $Training(M, \lambda, Bs, Lr, Op)$
11                 $Dice_t(M)$
12             end while
13         end for
14     end for
15 end for
```

(algorithm 2: line 6), we compute Dice score (See section 2.6), where the Dice score in the first iteration is initialized to 0. If we get a better Dice score than the last iteration (algorithm 2: line 7), then we save the model (algorithm 2: line 9) and continue the training for another λ epochs. If we get a worse Dice score, then we back to the hyperparameters: we increase the Batch size (algorithm 2: line 4) then we continue the training for λ epochs and so on until the Batch size becomes greater than *bs_max*. In the same way, we decrease the Learning rate (algorithm 2: line 3) and we continue the training phase. When the Learning rate becomes very small less than *r_min* (i.e., because small steps need a lot of time to converge), then we should change the optimizer[5] to another one (e.g., RMSprop, SGD) (algorithm 2: line 2) and initialize the Learning rate (algorithm 2: line 3) and the Batch size (algorithm 2: line 4) to the first state. The stopping criteria are met when there is no improvement after testing all the optimizers (i.e., Adam, RMSprop, SGD). In this case, we need to change the CNNs architecture which is done using the algorithm 1.

5. See section 4.2.4.3 for further discussion on this point

The algorithm of ELOBA_λ is based on exploring different search spaces: Optimizers, Batch size, Learning rate for many training epochs. This way helps to converge more toward the global minimum by trying different search spaces. Indeed, ELOBA_λ is used to unify the methods of training and to get the optimal hyperparameters (i.e., epochs, Batch size, Learning rate, Optimizer) for each deep learning architecture. Training a deep learning architecture for brain tumor segmentation of Glioblastoma tumors is solved as an optimization of a non-convex function. Non-convex function has a "cosine" function shape with many local minimum points and one global minimum, in practice, we don't expect to achieve the global minimum, but we try to achieve at least one of the local minimum points that are localized before the global minimum point. Moreover, optimizing a non-convex function is an NP-hard problem and it's not easy to avoid all local minimum points of this function. One of the techniques that have an impact on the optimization of non-convex function is using the batch size, where adding or reducing the batch size will lead to introducing some noise; with this way, we can escape at least from the first minimum points, but using learning rate will not help a lot to get rid of these points. For these reasons, we start by using batch size until we explore all its search space then we adjust a little bit learning rate to improve more the segmentation results.

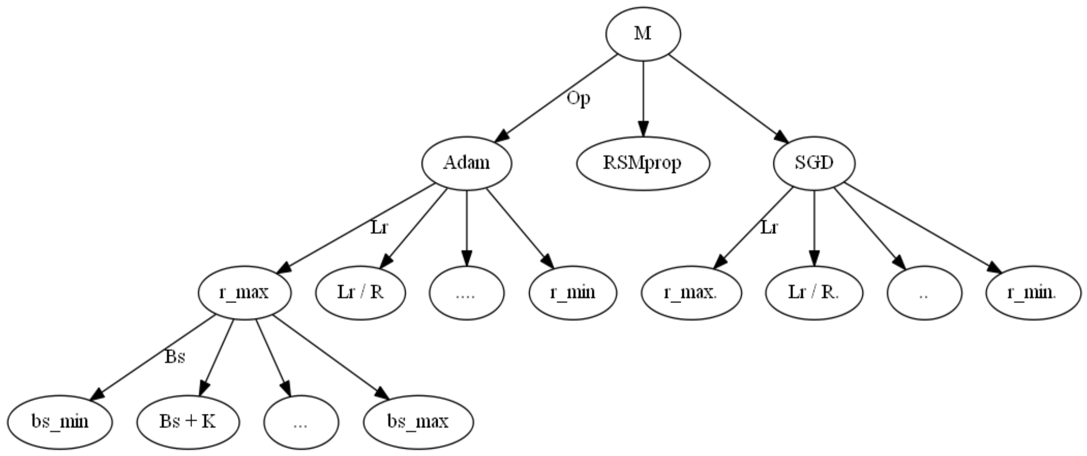


Figure 4.4 – illustration of the training strategy ELOBA_λ. Where "M" is the Model architecture, "Op" is the Optimizer, "Lr" is the Learning rate, "Bs" is the Batch size.

Figure 4.4 shows the training method of ELOBA_λ in order to explore different search spaces. As we can see, from a base model architecture at the higher-level node "M", each node (e.g., Optimizer = Adam) after that discovers the range of lower-level nodes (e.g., Learning rate = *r_max*) to the leaf nodes (e.g., Batch size = *bs_min*). The algorithm of

ELOBA_λ attempts to find the global minimum by exploring three search spaces: Optimizers, Batch size, Learning rate. As each path represents a potential hyperparameter, so ELOBA_λ tests the response of each path in this tree using depth-first search.

The goal of the Incremental XCNet and ELOBA_λ algorithms is to develop, train at the same time of an efficient deep learning architecture, in addition to accelerating the training phase and optimizing the hyperparameters. Incremental XCNet automates network architecture engineering. It aims to learn a network topology that can achieve the best performance on a brain tumor segmentation task. Moreover, in this thesis, we are interested more in improving the segmentation performance in terms of the evaluation metrics that are mentioned in the next section.

4.2.4 Experiments and Results

This section presents the results of our experiments and some discussions. Also, our brain tumor segmentation method is evaluated using the BRATS dataset which has real patient data with Glioblastoma brain tumors (both high- and low-grade). Finally, we explain the libraries and frameworks that are used in our work. Then, we evaluate our proposed architectures with a set of metrics that are used in state-of-the-art such as Dice score, Sensitivity (Recall), Specificity, and Hausdorff Distance.

4.2.4.1 Dataset Preprocessing

To remove noise and to enhance the quality of images, especially MRI images of the BRATS dataset that are generated using different MRI devices and acquisition protocols, in addition to patient motion after administration of Gd-DTPA (See section 2.5.).

The applied preprocessing operations in addition to normalization are performed as follows:

1. Removing the 1% highest and lowest intensities: this technique proved experimentally that it helps to remove some noise at the tail of the histogram, where this step has provided good results in many research [Havaei et al., 2017; Tustison, 2013].
2. Subtracting the mean value and dividing by the standard deviation in all channels:

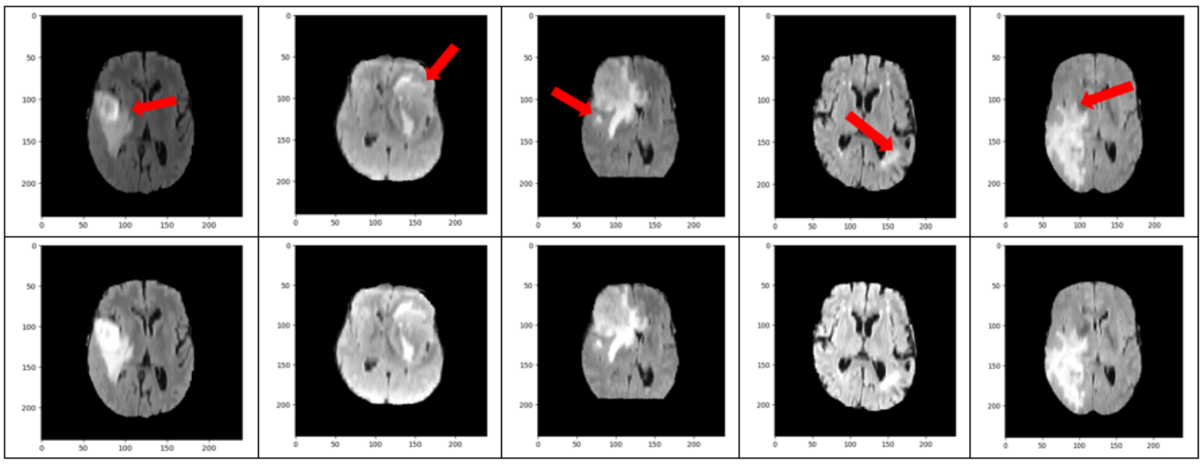


Figure 4.5 – An illustration of an example of 5 Flair scans before (top row) and after (bottom row) the proposed intensity normalization. Arrow red indicates where the images have improved after applying the normalization steps (see bottom row). We show 5 Flair scans only for simplicity.

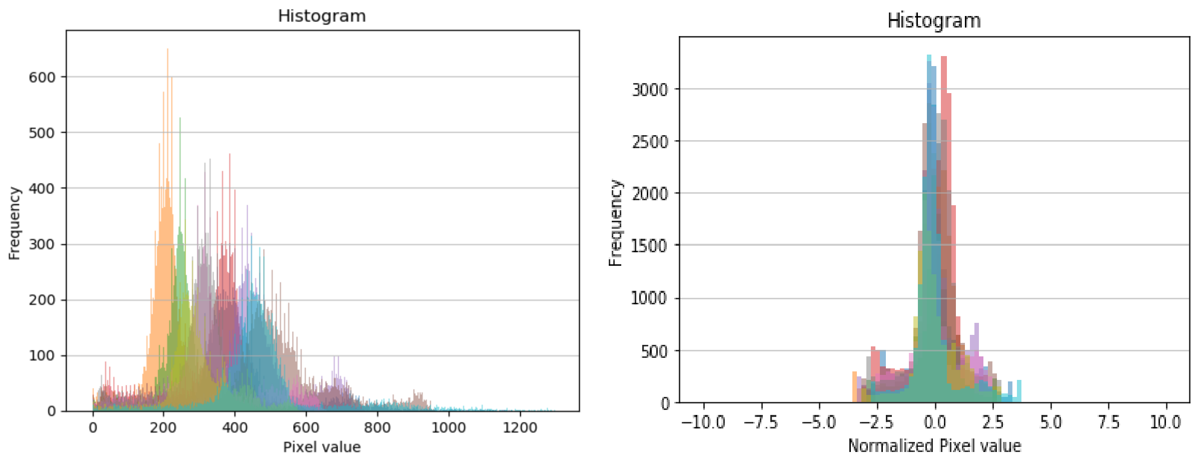


Figure 4.6 – Image intensity histograms of Flair scans of 10 subjects (Each subject is distinguished using different color) from the BRATS 2017 training dataset before (left) and after (right) the intensity normalization steps. We show 10 Flair scans only for simplicity.

this technique is used to center and to put the data on the same scale, i.e., bringing the mean intensity value and the variance between one and minus one [1,...,-1].

3. Selecting the slices that have more than 100 pixels representatives of each class among the four classes. After several experiments to select the best number of pixels representatives, we found that 100 pixels provide the best results, where in these slices, Glioblastoma multiforme tumors appear in their largest size.

The image intensity normalization effect is illustrated with Flair scans in Figure 4.5. The results shown in Figure 4.5 clearly demonstrated the effectiveness of the proposed intensity normalization steps. Moreover, the improvement is further confirmed by image intensity histograms of 5 subjects from the BRATS[6] 2017 training dataset, as shown in Figure 4.6.

4.2.4.2 Implementation

For the implementation of our CNNs architectures (2CNet, 3CNet, EnsembleNet), we use Keras[7] and Theano [Al-Rfou et al., 2016] as a backend. Keras is a high-level open source deep learning library that runs on top of Theano which can benefit from a massive parallel architecture such as GPU[8] to optimize deep learning architectures. In this work, all our results are obtained using Python environment on windows 64 bit, Intel Xeon processor CPU @2.10 GHz with 24 GB RAM, and for the training phase, it is done on Nvidia Quadro GPU K1200 with 4 GB memory.

4.2.4.3 ELOBA_λ hyper-parameters

Based on many experiments, we obtained the following hyperparameters:

— Optimizer (Op): ”Adam”, at the beginning of the ELOBA_λ algorithm, we used ”Adam” [Kingma and Ba, 2014] as an optimizer because it is faster than the other optimizers to converge toward the global minimum. The other optimizers that we used too: RMSprop and the last one is stochastic gradient descent (SGD) with Nesterov’s Momentum.

6. Each year, the Center for Biomedical Image Computing & Analytics at the University of Pennsylvania creates a new version of BRATS dataset, for instance, BRATS 2017 created in 2017.
7. Keras: https://keras.io/
8. GPU: Graphics Processing Unit

- Learning rate (Lr): R= 10^{-1}, R is a decreasing rate used to update the learning rate. r_max= 10^{-3}, r_min= 10^{-8}, the range [r_min, r_max] represents a search space (i.e., the space of all possible solutions) for the optimization algorithms. Thus, based on our experiments, the three chosen optimizers do not give guaranteed solutions out the range [r_min, r_max].
- Batch size (Bs): bs_min=32, bs_max=256, k=32, batch size is the number of training examples for one forward/backward pass. With the problem of non-convex functions, it is hard to get rid of the local minima points. Moreover, one of the techniques that influences the optimization of non-convex functions is the batch size, in which the use of small batch size leads to introduce some noise; this will helps to escapes from saddle points. And the use of big batch size (more than 256) leads to consume a lot of memory space. Thus, the best solution is to use a batch size in the range [32-256].
- The number of epochs λ= 20, where according to our experiments:
  - The algorithm of ELOBA_λ gives 20 epochs to each configuration of hyper-parameters. Thus, the first configuration of hyper-parameters for example, has 20 epochs, after getting a better result, the successful model will continue for another training with 20 epochs and so on, but if this model gets a worse Dice score then we update the hyper-parameters for the same model then we continue the training with the new hyper-parameters and so on. Thus, ELOBA_λ orients the training phase by testing the results (Dice score of the model) after every 20 epochs.
  - One of the biggest problems that we face with Deep learning models is the weights initialization. To address this issue, ELOBA_λ starts the training of a new deep learning model where the last model has stopped, in other words, the new model benefits from the last model in the initialization of weights. Thus, the algorithm 2 uses the first model to initialize the larger models.
  - ELOBA_λ rewards successful models that show an improvement of the Dice score during their first 20 epochs, however, it reduces the training time for poorly performing models.
  - Using a high number of epochs with a decreasing rate (Lr/10) will lead the learning rate (Lr) to become infinitesimally small, obviously the model will take

a long time to converge, and this point of weakness is well known in the Adagrad [Duchi et al., 2011] algorithm. Thus, after many experiments we found that 20 epochs is enough for the new model to obtain a small improvement compared to the last model because the new model will start the training where the last model has stopped, not from scratch, if the new model obtains a better result (Dice score) it will get another 20 epochs with the same hyper-parameters to converge more and to give a better result and so on (see the algorithm of ELOBA_λ, line 7 to line 12).

4.2.4.4 Results

In this section, we evaluate our models on a public dataset such as BraTS. We show the results of our proposed architectures with state-of-the-art architectures measuring both the segmentation performance (See section 2.6): Dice score, Sensitivity, Specificity and Hausdorff distance and the inference time. Our architectures 2CNet, 3CNet have deep architectures which help to extract relevant and very higher-level features. Moreover, EnsembleNet takes the advantage of these deep architectures to build a parallel architecture that brings together the most important features of the same subject. Thanks to ELOBA_λ training strategy that uses a simple and practical idea based on the use of many optimizers over a bounded range of training space.

The EnsembleNet's outputs are the "valid" pixels or pixels that are true with a high rate from both architectures 2CNet and 3CNet. Non-parametric fusion function (see equation 4.2) increases the Dice score over the complete region but at the same time it gives results less accurate over the core and the enhancing regions due to the fact that after getting the segmentation results by 2CNet and 3CNet, EnsembleNet model aggregates the results of 2CNet and 3CNets models using a non-parametric fusion function, this fusion function does not take any prior knowledge or any priority for any region in the predicted image, empirically this fusion function has improved the complete region compared to other regions because:

— The complete region holds all the other regions (core, enhancing). Thus, any improvement of any region will remains under the complete region and it will be also counted for the complete region.

— BraTS dataset is an unbalanced dataset, in which 98% of data are healthy tissue with label 0 and 1.1% of data are Edema with label 2 . In addition, it is well known that the healthy tissue class (label 0) and the edema class (label 2) do not belong to core or enhancing regions. Thus, any improvement of class two will be counted for the complete region.

— Equation 4.2 updates any confusion between labels of the same pixel to the label that has the minimum value, which obviously the zero value is the minimum value among all labels.

— The EnsembleNet model was influenced by the classes that have a minority of existence in the dataset because the 4 classes are not equivalent in any subject (input MRI image). In addition, we noticed that class zero (healthy tissue) and class two (edema) always win when the same pixel is confused.

Table 4.1 illustrates the results of the segmentation performance of state-of-the-art methods. As we can see, our models (2CNet, 3CNet, EnsembleNet) improved the Dice score compared to the state-of-the-art methods [Zikic et al., 2014; Axel et al., 2014; Pereira et al., 2015]. Moreover, EnsembleNet outperforms the method of [Havaei et al., 2017; Urban et al., 2014] in terms of Dice score and it obtains competitive results on other evaluation metrics (Specificity, Sensitivity). Our three models are also evaluated on the Hausdorff distance metric, it can be observed that our models achieve good results (i.e., 13 mm to 20 mm), in which the Hausdorff distance: [14.62 mm - 17.59 mm], [17.40 mm - 20.25 mm], [12.04 mm - 16.39 mm] for whole tumor, tumor core, enhancing tumor core respectively.

The figure 4.7 shows examples of the segmentation results. As we can see, 2CNet, 3CNet models detect almost most tumoral regions, these models also generate small regions (misclassified region) with different sizes which are classified as tumor regions (red circles). To address this issue, we use a simple but effective non-parametric fusion function, by using this fusion function we are able to detect which region of these regions are actually tumor regions. Non-parametric fusion function acts as a voting function that elects only the tumoral region with a high probability. EnsembleNet model (right column) aggregates the segmentation results of 2CNet and 3CNet models using this fusion function, as it can be observed that EnsembleNet detects only the tumor regions (blue circles) that are predicted by 2CNet and 3CNet in the first stage.

After the training phase, it comes the production phase where the trained model is

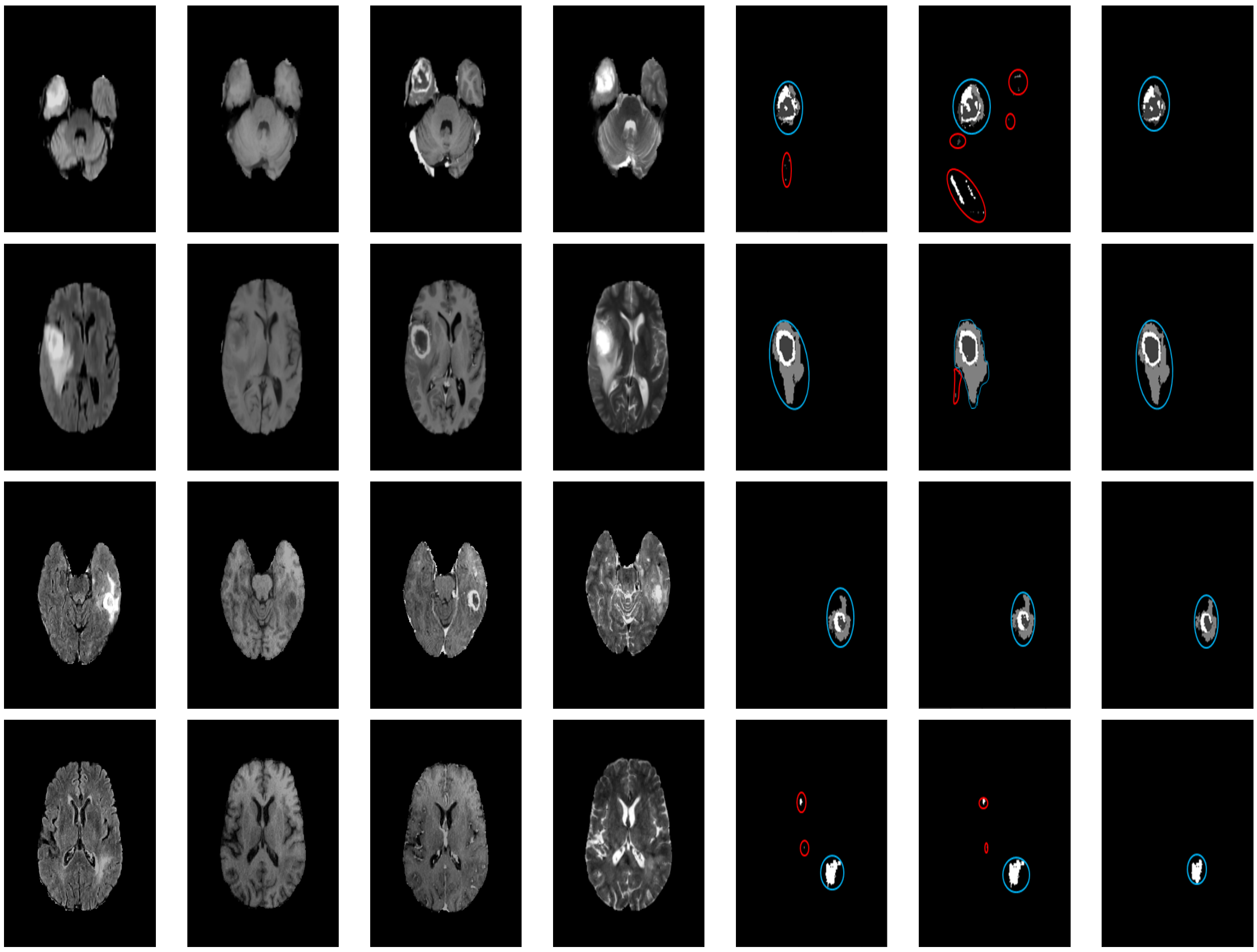


Figure 4.7 – Illustration of the segmentation results of 2 patients from the BRATS 2017 dataset. The first and second row for the 42^{th} and 77^{th} slice of patient ID = "$Brats17\_2013\_10\_1$". The third and fourth row for the 50th and 80th slice of patient ID= "$Brats17\_CBICA\_ABM\_1$". From **left** to **right**: Flair, T1, T1c, T2, segmentation results of 2CNet, segmentation results of 3CNet, segmentation results of EnsembleNet. Red circle: non-tumor classified as tumor, blue circle: tumor regions.

Table 4.1 – Segmentation results of our three proposed models which are noted by adding "*" to the architecture's name with state-of-the-art methods. WT, TC, ET denote Whole Tumor (complete), Tumor Core, Enhancing Tumor core respectively. Fields with (- , -/-) are not mentioned or given as a number in the published work. Best results in bold.

Methods	Dice			Specificity			Sensitivity			Hausdorff		
	WT	TC	ET	WT	TC	ET	WT	TC	ET	WT	TC	ET
2CNet*	0.88	0.80	**0.83**	0.74	0.76	0.77	0.88	**0.86**	0.78	17.59	20.25	16.39
3CNet*	0.87	0.81	**0.83**	0.74	0.78	**0.80**	**0.89**	0.84	0.74	16.86	18,99	13.67
EnsembleNet*	**0.89**	0.76	0.81	0.74	0.77	0.78	0.82	0.82	0.69	**14.62**	**17.40**	**12.94**
[Zikic et al., 2014]	0.84	0.74	0.69	-	-	-	-	-	-	-	-	-
[Urban et al., 2014]	0.88	**0.83**	0.72	-	-	-	-	-	-	-	-	-
[Axel et al., 2014]	0.79	0.68	0.57	-	-	-	0.79	0.67	0.63	-	-	-
[Pereira et al., 2015]	0.87	0.73	0.68	-	-	-	0.86	0.77	0.70	-	-	-
[Havaei et al., 2017]	0.88	0.79	0.73	**0.89**	**0.79**	0.68	0.87	0.79	**0.80**	-/-	-/-	-/-

used to infer/predict new data. And for benchmarking the models, inference time is used, i.e., the inference time represents the needed time to make a prediction on new data. Thus, to deploy deep learning on a large scale, it is necessary to improve the deep learning inference time. EnsembleNet is the fastest method among all other CNNs-based models based on the inference time (see Table 4.2).

Table 4.2 presents a comparison of our models (2CNet, 3CNet and EnsembleNet) with the state-of-the-art CNNs models. As we can see, our models outperform all cited CNNs models in the table in terms of inference time (21.39, 19.7, 21.49 respectively). We also notice that the inference time has decreased (21.49 seconds, almost eight times faster) compared to the state-of-the-art model of [Havaei et al., 2017] (three minutes), and about three times faster than [Urban et al., 2014].

The model of [Havaei et al., 2017] developed a deep learning architecture that exploits the local and the global features of the input patches using two pathways. The use of two pathways leads to losing time during the first and second stages of the fusion layer. However, our model EnsembleNet uses a parallel architecture technique but without losing time; EnsembleNet is the main process and it has 2 threads (2CNet, 3CNet) that run in parallel, and each model computes a different predicted image for the same input image. Then, we aggregate these two images (image for each model) by using a non-parametric fusion function. Thus, the timing required by EnsembleNet is the maximum between the

Table 4.2 – evaluation results of the state-of-the-art methods with our 3 proposed models which are noted by adding "*" to the architecture's name. Best results in bold.

Methods	Inference time (seconds) (architecture)
2CNet*	21.39 (GPU)
3CNet*	**19.72** (GPU)
EnsembleNet*	21.49 (GPU)
[Zikic et al., 2014]	- (-)
[Urban et al., 2014]	60 (GPU)
[Axel et al., 2014]	1200 (GPU)
[Pereira et al., 2015]	600 (GPU)
[Havaei et al., 2017]	180 (GPU)

timing required by 2CNet (21.39 seconds) and 3CNet (19.72 seconds) then we add the maximum to the timing required by the non-parametric fusion function (0.1 seconds). Therefore, the timing required by EnsembleNet is 21.49 (21.39 + 0.1) seconds. Hence, EnsembleNet needs only 21.49 seconds to segment the whole tumor and its subregions (see Table 4.2). The development of 3D CNNs model as used in [Urban et al., 2014] leads to getting a huge model that is hard to train it, i.e., during the training phase they need to compute the gradient for each 3D filter per pass. Moreover, it uses too much memory and massive computing due to using a large number of parameters. In addition, this model needs 40 hours for training while our model 2CNet reached the same complete Dice score with only 2D patches instead of 3D patches. Our EnsembleNet model uses a valid pixel-wise classification technique with a high rate from other models (2CNet, 3CNet) to predict the new image; this technique helps to replace the post-processing operation inside the algorithm. Thus the brain tumor segmentation pipeline does not need to use more operations after obtaining of predicted image.

4.2.5 Discussion and conclusion

In this section, we proposed three fully automatic models (2CNet, 3CNet, EnsembleNet) for the segmentation of Glioblastoma brain tumors using Deep Convolutional Neural Networks and MRI images. Through the empirical experiments, we proved that using two or three consecutive layers of convolution with small filters (3 x 3), gives a good performance in terms of segmentation performance and speed. The advantage of these models are: (i) they are incremental architectures; from a base model that is fed into our training strategy ELOBA_λ, we obtain a deeper model with high performance.

(ii) EnsembleNet takes the advantage of parallel architecture without losing time; 2CNet and 3CNet extract features at the same time from the same Glioblastomas MRI image then EnsembleNet aggregates the results of both models (2CNet and 3CNet) to get a more accurate result. (iii) optimized architectures; Incremental XCNet algorithm allows our models to get suitable architecture in terms of layers. (iv) these three models are end-to-end deep learning approaches, and fully automatic.

In addition, we developed a new training strategy ELOBA_λ. First, ELOBA_λ proved itself as an effective strategy and a simple method. Second, ELOBA_λ is created for giving us the exact time when we should change the model's architecture. Finally, ELOBA_λ is a dynamic and adaptable method, in which we can change all the hyper-parameters (i.e., epochs, Learning rate, Batch size, Optimizers) based on the workstation computing power and the architecture's size.

At the beginning of this section, we cited as issues of deep learning: the architecture's structure and the hyperparameters. To overcome these issues, we proposed two efficient, adaptable, and accurate algorithms: Incremental XCNet and ELOBA_λ that are capable to answer these questions at the end of the training phase, in particular, how many layers, neurons, batch size, learning rate, epochs.

Our intuition behind the proposed Incremental XCNet algorithm is as follows: we assume that we have unlimited computing and storage resources, where in practice, at a certain point, the memory gets overwhelmed by the blocks that are added after each iteration of training. That's the case here with our used memory, we could achieve only 3 and 5 blocks for each architecture, in this case, the number of blocks is limited by the computing and storage resources. In very large experiments, we need more computing and storage resources to exhaustively cover the entire search space. For these reasons, in the next section, we propose a new method for developing deep learning architectures.

4.3 Deep Learning-based selective attention

4.3.1 Problem statement

With the huge number of patients in the world and the massive medical data such as magnetic resonance imaging, positron emission tomography, ultrasound, computed tomography, X-rays...etc, emerged the need to discover new techniques for the treatment of patients. In addition, the diagnosis period plays a very important role, especially when we have patients with life-threatening diseases such as Glioblastoma brain tumors.

The addressed problem in this section is how to obtain the most accurate and reliable brain tumor segmentation in the light of many challenges:

1. Glioblastoma multiforme tumors have 3 sub-regions in addition to Healthy tissue: Necrotic and Non-Enhancing tumor, Peritumoral Edema, Enhancing tumor. Also, Gliomas, including Glioblastomas, invade the surrounding tissue rather than displacing it, causing unclear boundaries, and they have the same appearance as Gliosis, stroke, inflammation, and blood spots [Goetz et al., 2015].
2. The second challenge is that Convolutional Neural Networks do not perform well with highly unbalanced data such as the BRATS dataset where 98% of data [Havaei et al., 2017] are healthy tissue (i.e. non-tumorous). Training a CNNs architecture with unbalanced data will make predictions with low sensitivity, consequently, CNNs architecture will bias toward the healthy class. In medical applications, the most important metric for a clinical decision support system is sensitivity [Hashemi et al., 2018] toward tumoral regions.
3. The third challenge is common with the semantic image segmentation problem, where this challenge is about the reduction of features during the pooling layers and convolution striding [Chen et al., 2017].
4. The fourth challenge is about modeling the global context and spatial relationship among image patches. It is known that conventional deep learning architectures, in particular, CNNs architectures do not take into account the relationship among image patches [Zhao et al., 2018] and these patches together make up the entire MRI image.

5. The fifth challenge is about developing a light CNNs architecture that can run in GPU and CPU by focusing on maximizing features extraction instead of developing a very deep architecture.

4.3.2 Visual areas-based interconnected modules

The MRI technique is designed specifically to provide radiologists powerful visualization tools to analyze medical images. These tools produce images that show a contrast between the soft tissue of the brain (or other human organs such as liver) [Akram and Usman, 2011; El-Dahshan et al., 2014]. Some MRI images do not clearly show the border between brain regions, where the brain appears as a single mass. Thus, to extract only the tumoral regions from the whole brain, the proposed architecture has to extract more features about the healthy tissue and the tumoral regions. To solve this issue, we propose a CNNs architecture that is based on the maximization of the features' representation, in other words, these architectures have to extract many relevant features [Ghebrechristos and Alaghband, 2018] from the MRI images.

The idea of our inspired architectures is to develop a ”lightweight” and compressed deep learning architecture that can run in GPU and CPU with more focus on the maximization of feature extraction instead of developing a very deep architecture. In addition, our architecture is inspired by the structure of the visual cortex, in particular, the Occipito-Temporal pathway (OTP). Our idea behind this inspiration is as follows: the algorithm of CNNs [LeCun et al., 1989, 1998] originally is inspired by the visual system. In 1962, Hubel and Wiesel [Hubel and Wiesel, 1962] discovered that each type of neurons in the visual system, responds to one specific feature: vertical lines, horizontal lines, shapes ...etc. From this discovery, the algorithm of Neocognitron is developed [Fukushima and Miyake, 1982], but the problem of Neocognitron is: it does not have a supervised learning algorithm [LeCun et al., 2015], where in CNNs they solved this problem by using Backpropagation algorithm [Rumelhart et al., 1986]. Moreover, what was the ”missing part” of CNNs is the architecture's structure. Thus, in this work, we attempt to inspire from the visual system its internal structure. And by inspiring the structure of the visual system, we could add to the function of CNNs a specific architecture.

The proposed CNNs model is inspired by Occipito-Temporal pathway which has a special function called *selective attention*[9] that uses different *receptive field* sizes in successive layers to figure out the crucial regions (or object) among many regions (or objects) in an image. Therefore, using *selective attention* technique to develop a CNNs model, helps to maximize the extraction of relevant features from MRI images. The OTP structure made up of many interconnected areas (i.e. modules): V1, V2, V4 and IT (see figure 4.8) [Desimone et al., 1989; Manassi et al., 2013]. OTP structure is important and critical for object recognition [Desimone et al., 1989] and for the brain's memory [10] [Ono and Nishijo, 1992]. The design of our CNNs model based on these interconnected modules allows to maximize the features' representation of Glioblastomas tissue, also leads to detect and predict even small regions with fuzzy borders. Moreover, developing a model based on the rule of using many interconnected modules, is known in the state-of-the-art methods and it provides a good performance in many applications such as GoogleLetNet model [Szegedy et al., 2015] that is built based on the module of Inception, ResNet model [He et al., 2016] which is based on many modules of residual layers.

To determine a shape in the visual system, for example, a person's face, it will be processed from lines and edges at V1 area to shapes at V2 area to objects at V4 area to faces at the end at IT area [Herzog and Clarke, 2014; Manassi et al., 2013]. The OTP structure has four properties (i.e. N:1 to N:4):

1. Retina is the input of the visual system.
2. The visual cortex treats the forms hierarchically; object has different shapes, and shape has different lines and edges and so on.
3. OTP structure is composed of several interconnected visual areas (i.e. V1, V2, V4, IT), where each area responsible for a specific task.
4. Each visual area receives a *Receipted Field* larger than the one at the previous area.

9. Selective attention is a behavioral and cognitive process to filter stimuli received by the brain. Attention is a limited resource, so selective attention is a mechanism that allows us to focus on something that is relevant to the task at hand while tuning out irrelevant details. An example of selective attention is called the "cocktail party effect". There is quite a lot of research that has found a relation between selective attention mechanism and patients with an autism spectrum disorder, also between selective attention and people with sensory processing challenges.

10. Klüver-Bucy syndrome (KBS), experiments of Klüver and Bucy 1937

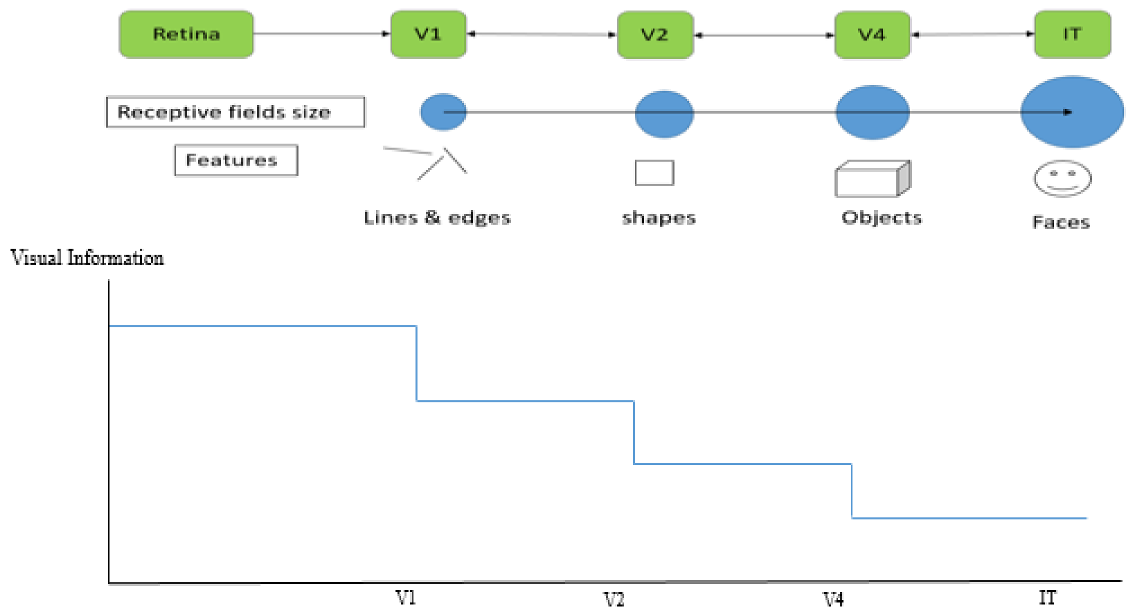


Figure 4.8 – Illustration of the brain visual cortex' regions and how the information gets processed and reduced from one region to another. **Top image** shows the structure of the OTP structure that is made up of many interconnected regions: V1, V2, V4, and IT, in addition to the task of each region, for instance, V1 is responsible for extracting lines and edges. **Bottom image** shows the quantity of information that passes from region to another.

In our work, we attempt to build a CNNs model inspired by the OTP structure, thus, we inspired:

1. The Retina represents the input images.
2. The hierarchical function using convolution operators.
3. We have developed two modules inspired by the area V1 and V2, interconnected to each other using direct and skip connections.
4. To take into account the last property (N: 4) of visual cortex, there are two ways:
   (a) Reducing the feature maps at each level using Pooling operation.
   (b) Or using larger kernels at each convolution layer incrementally.

The use of larger kernels (i.e. the property 4.b) increases the computational time, i.e., it increases the number of operations. On the other hand, the use of small kernels, has proved previously that they keep a lot of information and it allows a CNNs model to be deeper in terms of layers [Simonyan and Zisserman, 2014]. Thus, based on the issue of reducing the number of operations, the best option is the first one (i.e. 4.a).

4.3.2.1 First module V1

In this work, we consider the structure of V1 is the same as V2, V4, IT, but the output of these modules is different from module (e.g. V1) to another (e.g. V2). Thus, the module of V1 is composed as shown in figure 4.9 of: an input (phase 1), then we have convolution layers (phase 2), and a max-pooling layer (phase 3), then another convolution layers (phase 4), upsampling (phase 5) and concatenation layers (phase 6):

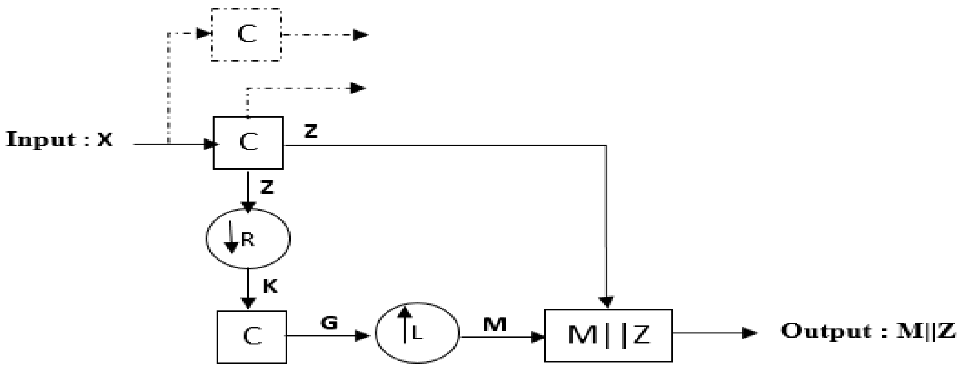


Figure 4.9 – An illustration of visual area V1. C is a convolution operator, Z and G are feature maps, K is the output of Pooling layer, M is a high-resolution feature map, $\downarrow R$ is a pooling layer, $\uparrow L$ is an upsampling operator, $M||Z$ is a concatenation layer between M and Z. Also, the continuous lines are direct and skip connections, and dashed lines are used to build the second module of visual area V1 (i.e. DenseConnectionOCM).

1. First phase: let's consider Retina as the input (i.e. X) to our model.
2. Second phase: obtaining hierarchical features is guaranteed by the method of convolution operation; convolution uses a filter bank (i.e. trainable parameters) to extract many levels of features across the entire input images. Moreover, from section 4.2, we found that using two consecutive convolution layers with kernel size equals to 3 × 3 provides a high performance. Thus, in this work we will use two consecutive convolution layers Z = Conv(Conv(X)). The output of convolution layer called feature maps Z which are calculated as follows (see equation 4.3):

$$U = b_i + \sum_{j=1}^{m} W_{ij} * V \tag{4.3}$$

Where b_i and W_{ij} $(i, j \in \mathbb{N})$ are trainable parameters, called respectively bias and weights. m is the size of the input $(m \in \mathbb{N})$. V and U are respectively input and output, and '*' is the convolution operation. After that we apply a non-linear activation function $Relu(U) = max(0, U)$. In CNNs architecture, W_{ij} is the kernel parameters.

3. Third phase: to reduce the feature maps dimensionality (as described in property N: 4), we use Max-Pooling 2×2 operation to provide the minimum reduction of feature maps in addition to the shifting invariant property. Max-Pooling takes the maximum value of each non-overlapping square (i.e. sub-region) of feature map Z. MaxPooling K_{ij} is computed as follows (see equation 4.4):

$$K_{i,j} = \max_{h} \; Z_{i+h,j+h} \tag{4.4}$$

   Where $h \in \mathbb{N}$ is the size of sub-region, $i, j \in \mathbb{N}$ are the stride values for the vertical and horizontal axes respectively.

4. Fourth phase: using trainable parameters G = Conv(Conv(K)) again to extract more feature maps from the Max-pooling output. Also, to compute the feature maps G, we use equation 4.3:

5. Fifth phase: to concatenate two types of features maps (i.e. feature maps of phase 2 and phase 5), where these two types of feature maps are not at the same scale. Thus, to obtain higher-resolution feature maps from lower-resolution feature maps in CNNs architecture, there are two techniques: (1) deconvolution, (2) upsampling. The first one uses trainable parameters, but the second technique does not use any trainable parameters, this technique works by inserting zeros padding internally between pixels, where the number of zeros is determined by the stride parameter.

   The final step is to convolve the sparse feature maps with trainable parameters (i.e. convolution operation) to obtain dense features maps, this technique helps to avoid the problem of Overfitting [Badrinarayanan et al., 2015]. The method of upsampling operator works as follows (see equation 4.5 and 4.6):

$$\begin{aligned} M_L[n] &= G(n)_{\uparrow L} \\ M_i(n) &= upsampling_{L,n}(G_i) \end{aligned} \tag{4.5}$$

$$M_i = \begin{cases} G_i(n/L) & , if(n/L) \in \; \mathbb{N} \\ 0 & , otherwise \end{cases} \tag{4.6}$$

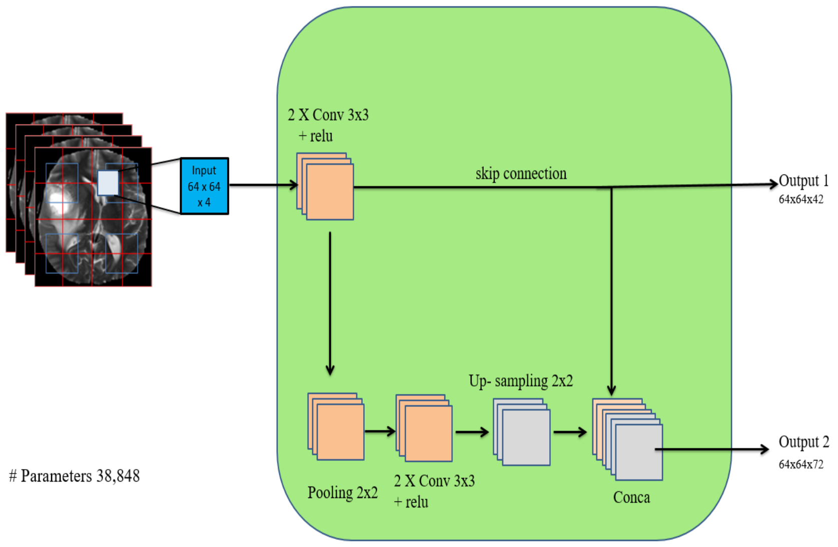
Upsampling operator works by inserting $L-1$ zeros between $G(n)$ and $G(n+1)$ for all n elements.

6. Sixth phase: is the concatenation layer, where in this layer, we concatenate the feature maps of phase 2 and phase 5.

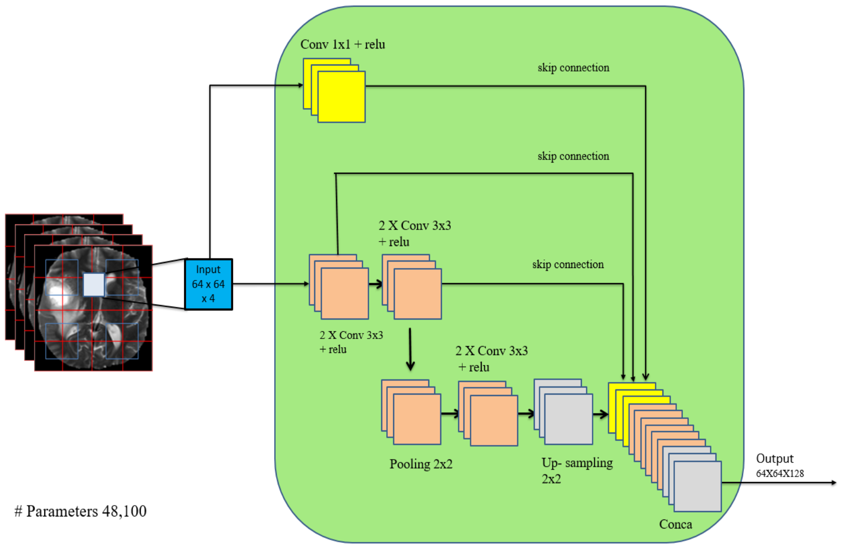
We used the phases from 1 to 6 to build a Sparse Connection OCM (OCcipito Module) module (see figure 4.10.a), and a Dense Connection OCM module (see figure 4.10.b). In the second module (i.e. Dense Connection OCM module), we added 1×1 convolution layer [Lin et al., 2013] which operates over N-dimensional volumes and also we considered it as a preliminary classification of the input images. Dense Connection OCM and Sparse Connection OCM modules are two representations of the visual area V1, in which from these two modules, we have deduced three CNNs architectures (see section 4.3.2.2.2), each with two modules V1 and V2. Moreover, we did not build the entire visual path, but for each architecture, we build two modules V1 and V2. These two modules attempt to model the structure of OTP, in addition to, they extract different representations from the input MRI images where this technique of extracting different representations is used by GoogleLetNet in its Inception module [Szegedy et al., 2015]. We have used two different types of convolution operations (i.e., 3×3 and 1×1) for the module Dense Connection OCM and Sparse Connection OCM to diverse the feature representations of the MRI images:

— Sparse Connection OCM module: applies a sparse connectivity, where we connect a subset of convolution layers to each other. This module has one input and two outputs (i.e. output 1 and output 2). We refer to this module as SparseConnectionOCM. The details are illustrated in figure 4.10.a.

— Dense Connection OCM module: applies two strategies (1) Dense connectivity and (2) 1×1 convolution operation, where we concatenate all high-level feature maps and low-level feature maps in the concatenation layer before the output. This module has one input and one output compared to the previous module (i.e., Sparse Connection OCM). We refer to this module as DenseConnectionOCM. The details are illustrated in figure 4.10.b.



(a) SparseConnectionOCM, where we use a skip connection to connect between low-level feature maps and a pre-output concatenation layer. This module has 38,848 parameters.



(b) DenseConnectionOCM, where we use multiple skip connections to connect data and low-level feature maps and pre-output concatenation layer together. This module has 48,100 parameters.

Figure 4.10 – Schematic representation of the SparseConnectionOCM and DenseConnectionOCM. Each orange/purple/yellow box corresponds to multi-channel feature maps. The input represents the multi-modal MRI images. Each output (i.e. output 1, output 2, and output) is an input to the next module. Conv is the convolution operator, 1 $\times$ 1 and 3 $\times$ 3 are the filter's size, pooling 2 $\times$ 2 is the max-pooling operator with 2 $\times$ 2 window. Relu is the non-linear activation function, upsampling is the upsampling operator, Conca is the concatenation layer, skip connection is a connection between two non-successive layers.

4.3.2.2 CNNs architectures

To connect between different modules of a CNNs architecture, we use two types of connections: first a direct connection from layer H^{L-1} to layer H^L through a mapping non-linear activation function f (see equation 4.7), secondly a special type of connection called skip connection that connect layer H^{L-i} to layer H^L where $i > 1$ (see section 4.3.2.2.1).

$$H^L = f(W^L \times H^{L-1} + b^L) \tag{4.7}$$

Where b^L, W^L are trainable parameters, called respectively bias and weights.

4.3.2.2.1 Skip connections

Skip connections help to backpropagate the gradient signal across several layers and leads to a fast convergence [Drozdzal et al., 2016] and since its first introduction [He et al., 2016], have been shown an improvement of accuracy for many computer vision tasks. For example in the field of biomedical image segmentation [Drozdzal et al., 2016] such as U-net [Ronneberger et al., 2015], and Fully Convolutional Networks (FCN) were applied for semantic segmentation [Long et al., 2015], also for image recognition such as ResNet [He et al., 2016], ResNext [Xie et al., 2017], DenseNet [Huang et al., 2017]. Skip connections (see figures 4.11) encourage a CNNs model to reuse low-level features such as lines, edges; where the model uses these low-level features many times with high-level features such as shapes, objects at the same level. Thus, the combination of low-level features and high-level features in MRI images using skip connections, helps to localize the tumor region and then to detect the shape and the boundaries of Glioblastoma intra-tumoral structures. Skip connections connect between two non-successive layers: layer H^{L-i} and layer H^L (see equation 4.8):

$$H^L = f(W^L \times H^{L-i} + b^L) \tag{4.8}$$

Where b^L, W^L are trainable parameters, called respectively bias and weights. H^{L-i}, H^L ($i \in \mathbb{N}$ & $2 \leq i$) are two layers.

4.3.2.2.2 Sparse and Dense architectures

CNNs are known for its ability to extract many hierarchical features from images. To develop a deep CNNs architecture, we have either pixel-wise approach or patch-wise approach. In the first approach, CNNs deal with Pixels, while in the second approach, CNNs deal with Patches. Our method is based on Patch-wise technique, and it takes

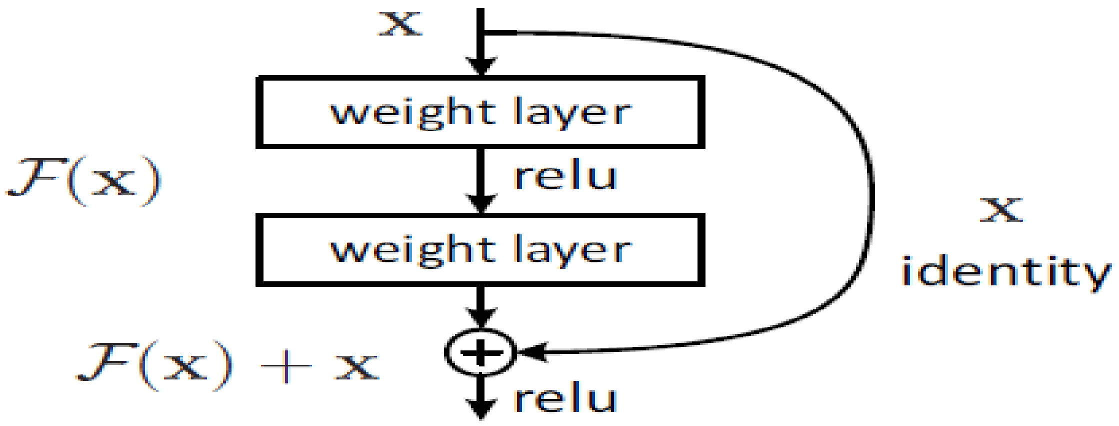


Figure 4.11 – An example of Skip connections [He et al., 2016].

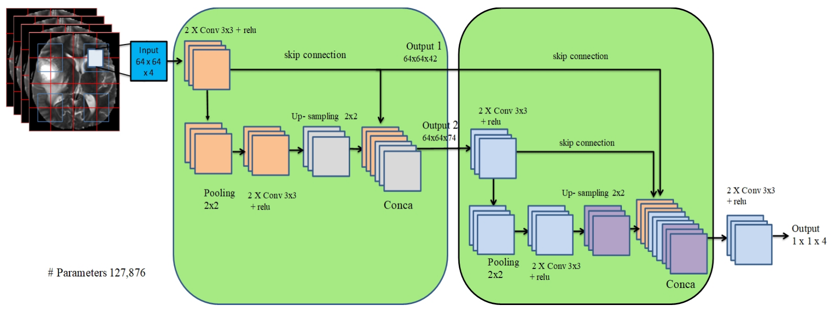
as input patches with limited size (in our case 64 x 64) to produce an output patch with predicted tumoral regions. The most used technique to generate patches is adjacent patches[11]. Moreover, pixel-wise classification is computationally expensive such as the architecture of AlexNet [Krizhevsky et al., 2012], VGGNet [Simonyan and Zisserman, 2014]. From another side, Fully Convolutional Networks (FCN) [Ronneberger et al., 2015; Çiçek et al., 2016; Milletari et al., 2016] is based on patch-wise classification, and it has shown a great performance and more accurate results, in addition, it is efficient in terms of computation compared to pixel-wise technique. In this work we investigate three variants of CNNs patch-wise architectures:

— Sparse MultiOCM architecture: this architecture made up of two modules of SparseConncentionOCM which has two outputs (i.e. output 1 and output 2). Then, we concatenate the first output (i.e. output 1) with a pre-output concatenation level (i.e. conca layer) of the second module, and the second output (i.e. output 2) is used as an additionnel input to the second module. We refer to this architecture as SparseMultiOCM. The details of this architecture are illustrated in figure 4.12.a.

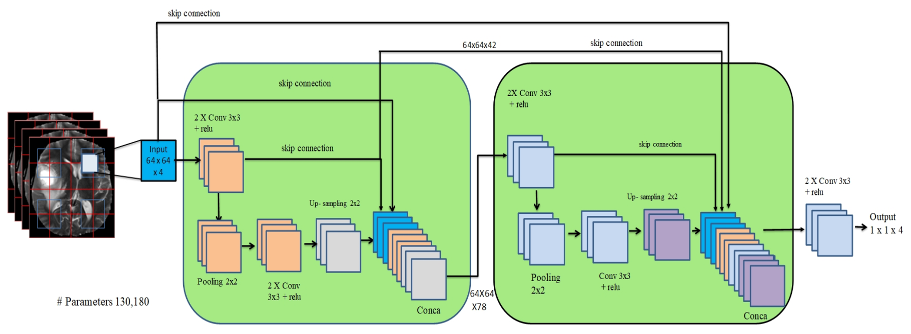
— Input Sparse MultiOCM architecture: in this architecture we investigate the effect of adding directly the input as another concatenation feature maps. We refer to this architecture as InputSparseMultiOCM. The details of this architecture are illustrated in figure 4.12.b.

— Dense MultiOCM architecture: in this architecture we attempt to extract different

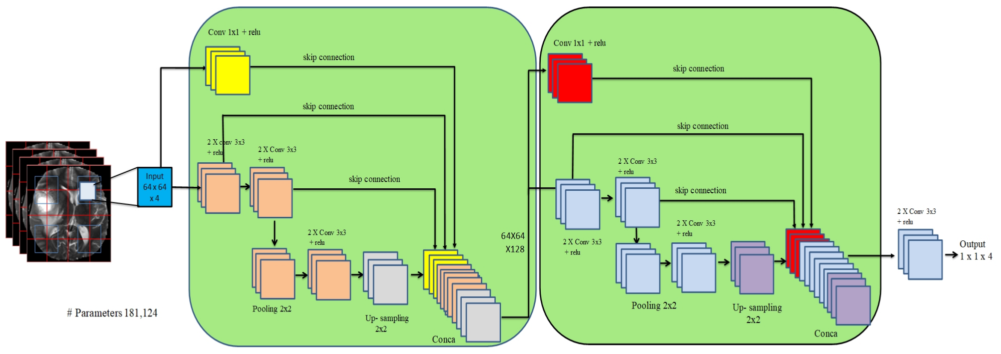
11. CNNs predict labels separately from each others; label of second patch will be predicted without any relation with the first patch and so on for third, fourth ...etc. see section 4.3.3 for further details



(a) SparseMultiOCM, we use two modules of SparseConnectionOCM to build this architecture. This architecture has 127,876 parameters.



(b) InputSparseMultiOCM, we concatenate the input with the pre-output concatenation level in two modules. This architecture has 130,180 parameters.



(c) DenseMultiOCM, we use two modules of DenseConnectionOCM to build the complete architecture. This architecture has 181,124 parameters.

Figure 4.12 – An illustration of the proposed CNNs architectures to solve the problem of fully automatic brain tumor segmentation. Each orange/red/purple/yellow box corresponds to multi-channel feature maps. The output 1 × 1 × 4 is a 1 × 1 convolution layer with 4 classes of BRATS dataset where we have 4 sub-regions: Healthy tissue, Necrotic and Non-Enhancing tumor, Peritumoral Edema, Enhancing core.

feature representations from the input MRI images. We refer to this architecture as DenseMultiOCM. The details of this architecture are illustrated in figure 4.12.c.

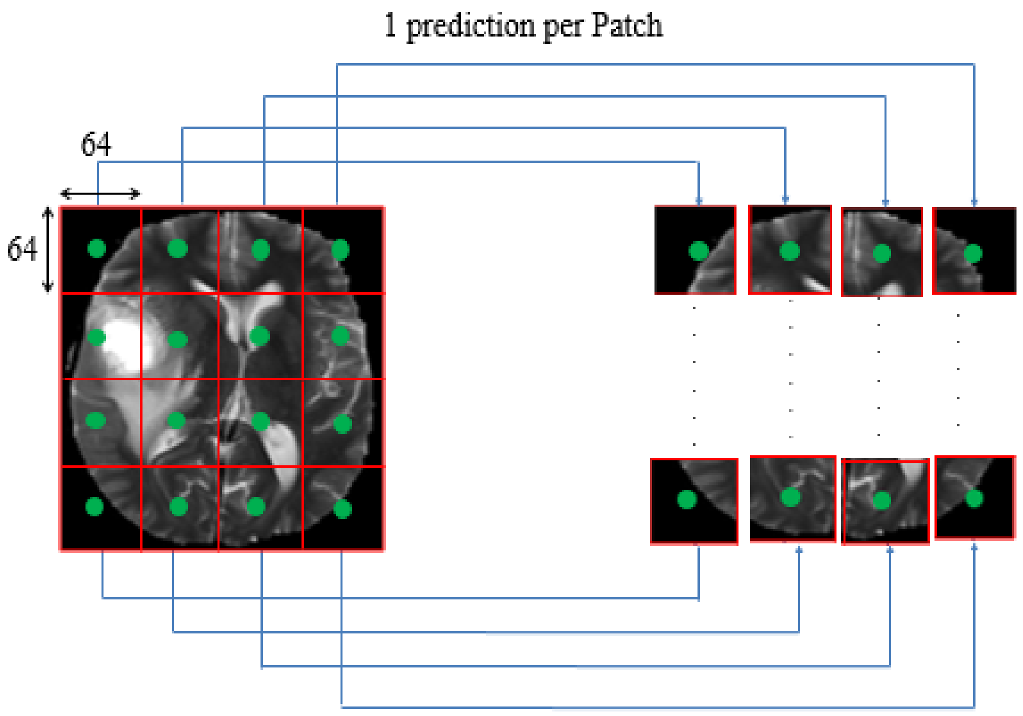
Finally, after building these architectures that are made up of two modules (i.e. SparseConncentionOCM, DenseConncentionOCM), we have added two consecutive convolution layers before the final output (i.e. output 1 × 1 × 4) to extract unified features. The motivation of adding these consecutive convolution layers is: we have concatenated several feature maps including input Patches (i.e. raw pixels) from many levels at the concatenation layer of the second module, thus we have heterogeneity of features and pixels at one level that need to be unified before classifying them into 4 classes.

4.3.3 Overlapping Patches

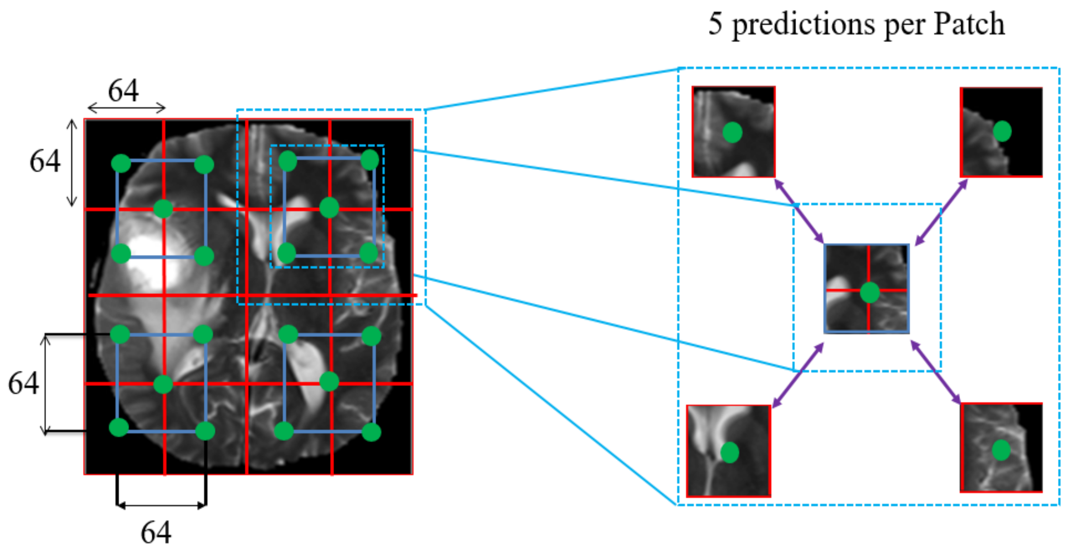
Our CNNs architectures are built upon 2D image Patches (Patch is a set of pixels), in which these architectures predict the pixel's class which is the center of the 2D Patch (i.e. see figure 4.13, green points). A lot of research [Urban et al., 2014; Pereira et al., 2015; Havaei et al., 2017; Ben naceur et al., 2018] use the technique of Adjacent Patches (see figure 4.13.a); Patches that are next to each other, but the limit of CNNs with Adjacent Patches is that CNNs do not take into account the fact that these Adjacent Patches together make up the entire image [Havaei et al., 2017; Pereira et al., 2017; Zhao et al., 2018], which means that if we mix the location of patches, for CNNs is the same thing with or without mixing the location of patches, and this is the weak point of CNN algorithm with the patch segmentation approach. To overcome this issue and to create a sort of memory in CNNs architecture for the location of patches as a human being does when we try to sort an image divided into multiple patches. The first thing we do is looking for some patches that have something in common and this is a natural algorithm implemented in our brains[12]. For these reasons, we extract Overlapping Patches (see figure 4.13.b) which help the architectures to see the local (Red Patches, see figure 4.13.b) and the larger context (Blue Patches, see figure 4.13.b), in addition, it provides a data augmentation [Hashemi et al., 2018] through a better balance of data examples; where through extracting a new patch by taking 25% from the other patches, the new patch is considered as a shifting of each patch horizontally and vertically on the width axis, then on the height axis respectively (see figure 4.14).

It is well-known that CNNs work by convolving each kernel with the corresponding Patch in the image, then this kernel moves to the next Patch until completing the entire image. Then, through the learning process, CNNs will classify these adjacent Patches into different classes (in our case 4 sub-regions), so the final segmented image will be the

12. This is the source of inspiration for many algorithms in literature, such as KNN, K-means [Zhang et al., 2019] or CRF [Wu et al., 2014], that are defined over all pixels or all labels of the full image, in this way, the prediction of a label is shared among all pixels, or labels through a proximity distance or a mean-field message, respectively.



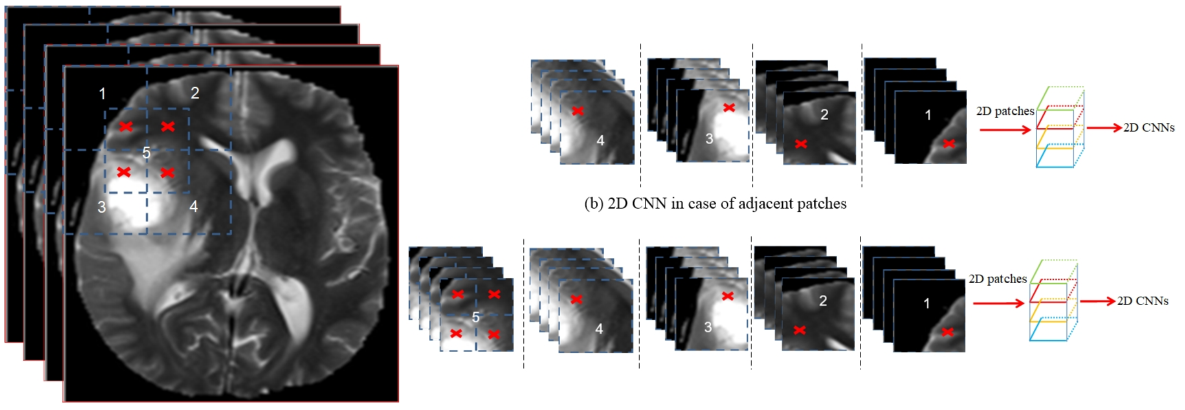
(a) Adjacent Patches.



(b) Overlapping Patches.

Figure 4.13 – Examples of A) Adjacent Patches (one prediction per Patch) and B) Overlapping Patches (5 predictions per Patch) from an input MRI image. The training dataset is created by extracting Overlapping Patches (i.e. Red and blue boxes). Best viewed in color.

result of only one pixel prediction (i.e., green point) per Patch as shown in figure 4.13.a. However, with the Overlapping Patches, the final segmented image will be the result of 5 predictions per Patch as shown in figure 4.13.b. Thus, with the technique of Overlapping Patches, CNNs is able to classify and see the larger context even using small Patches, in addition, it could learn to build a relationship among these Patches (See figure 4.14). In



(b) 2D CNN in case of adjacent patches

(a) Axial MRI image and the patches of the shaded region.

(c) 2D CNN in case of overlapping patches

Figure 4.14 – Examples of (a) an Axial MRI image and the patches of the shaded region, where the red cross signe represents the shared region with the patch number 5. (b) 2D CNNs in case of Adjacent Patches: at each epoch of training we use only 4 patches at the shaded region. (c) 2D CNNs in case of Overlapping Patches: at each epoch we use 4 patches in addition to a new patch (number 5) constructed from the four patches.

this case, CNNs can figure out through the concept of memory, and through the frequency of patches that come in the same position, that this patch for example is next to this patch. And because for each patch we have many patches with the same label around it, that can influence positively on the prediction of that patch at the center, so, we can say that overlapping patches is like voting but through time (See figure 4.14).

4.3.4 Class-Weighting technique

One of the main problems of medical imaging applications [Hashemi et al., 2018] is unbalanced data, we find, for example, in BRATS dataset, the class of interest (i.e. tumoral region) is the minority class which has almost 2% of data, while the healthy class has 98% of data [Havaei et al., 2017]. Training a CNNs model on an unbalanced data will bias toward the majority class (i.e. healthy class), and will produce a low Sensitivity. In literature, there are many proposed methods such as two training steps [Havaei et al., 2017], median frequency balancing [Badrinarayanan et al., 2015], asymmetric similarity loss function [Hashemi et al., 2018]. To overcome the issue of unbalanced data, we have conducted a preliminary experimental analysis, where we applied two techniques during the training stage: first step, we extract image Patches from the axial view randomly, and each class has the same number of training Patches, this technique is useful to mitigate the problem of unbalanced data [Zhao et al., 2018], but it turns out that this step is not enough to solve completely the issue as you can see in the experiments in Table 4.3 (first two experiments without using any weighting factors). That's why in the second step, we used class-weighting technique to weight the vote of each class, in other words, this technique gives to each output a specific weight W_j according to its contribution in the segmentation results. Also, as you can see from the same table that when we start adding weighting factors, the results of the segmentation started to be better in terms of the evaluation metrics compared to the first two experiments without the weighting factors. Here, we can prove 2 things; first, using the same number of patches will mitigate the issue, but it will not solve it. Secondly, the experiments with the weighting factors prove the efficiency of using weighted cross-entropy loss function compared to using the technique of equal sampling of patches.

As mentioned previously in chapter 3, the forward propagation computes the inference of CNNs using Softmax function (see equation 4.9)

$$p_j = \frac{\exp^{z_j}}{\sum\limits_{k=1}^{K} \exp^{z_k}} \tag{4.9}$$

So, the technique of class-weighting is based on weighted-cross entropy loss function (See equation 4.10) to compute the error between the ground truth and the predicted image during the training phase:

$$LW(p,q) = -\sum_{j=1}^{K} W_j \times q_j \times \log(p_j) \tag{4.10}$$

Where K is the number of classes ($K \in \mathbb{N}$), q_j is the j^{th} element of the normalized ground truth vector ($q_j \in \mathbb{R}$) and p_j is the j^{th} element of the estimated vector ($p_j \in \mathbb{R}$) for the class j (j is an integer $\in [1..K]$). W_j ($0 < W_j < 1 \quad and \quad \sum\limits_{j=1}^{K} W_j = 1$) is a specific weight assigned to the class j. L(p,q) and LW(p , q) are two loss functions that represent the error between the estimated vector and the normalized ground truth vector. Softmax function gives the estimated vector p_j at the end after each forward propagation by squashing the outputs to be between 0 and 1, i.e. the outputs become as probabilities. Then these probabilities are fed into the weighted Cross-entropy loss function, this function after that computes the 'real' error based on the Cross-entropy function and the class-weight W_j for each class. Moreover, to make the prediction of the estimated vector accurate and faster, we use one-hot encoding, where this encoding gives all probabilities of the ground truth to one class (i.e. the correct class), and the other classes become zero, e.g. $q = [1,0,0,0]$, this vector q indicates that the first class is the correct class. In this case, to calculate the error we need only one operation instead of many operations for all classes. Softmax function attempts to predict this vector (i.e. q vector) after each forward/backward propagation until the end of training. The advantage of Weighted Cross-entropy Loss function is: it could compute the real error for each class, so as a result, it rewards or penalizes only the correct classes. The drawback of this function is how to calculate the weight for each class. To answer this question, we have conducted an experimental analysis of the effect of Class-Weighting technique on the segmentation results. As it is shown in Table 4.3, there are different and totally independent experiments. We have put our proposed CNNs architectures under different regularization techniques such as Instance normalization, Dropout (with different rates), L2 regularization (with different rates), but these regularization techniques did not improve a lot the segmentation results. To evaluate the segmentation performance of each experiment, we computed Average Dice score as follows:

$$\begin{aligned} Average\ Dice &= \frac{1}{N}\sum_i Dice_i \\ &= \frac{Complete + Core + Enhancing}{3} \end{aligned} \tag{4.11}$$

Where N is the number of Dice score sub-metrics (i.e., N=3).

Table 4.3 – the parameters and the results of different and independent experiments of the class-weighting technique. Healthy, Necrotic and Non-Enhancing tumor (NCR/NET), Peritumoral Edema (Edema), Enhancing tumor represent the different Glioblastomas tumor sub-regions, in which we assigned a percentage of contribution for each class. Moreover, for each experiment we show the Dice score of 10 MRI images from BRATS 2018. The last experiment (No = 14) has the best mis-classification costs for our proposed CNNs architectures. Fields with (/) mean we did not apply class-weighting technique.

No.	Patch type	Class-Weights				Dice score			Average Dice
		Healthy	NCR/NET	Edema	Enhancing tumor	Complete	Core	Enhancing	
1	Adjacent	/	/	/	/	0.760	0.595	0.659	0.671
2	Overlap	/	/	/	/	0,902	0,8	0,631	0,778
3	Adjacent	0.3	0.2	0.2	0.3	0.772	0.593	0.655	0.673
4	Overlap	0,3	0,2	0,2	0,3	0,899	0,793	0,736	0,809
5	Overlap	0,3	0,15	0,3	0,25	0,909	0,816	0,752	0,826
6	Overlap	0,3	0,12	0,33	0,25	0,901	0,826	0,745	0,824
7	Overlap	0,3	0,1	0,38	0,22	0,906	0,828	0,779	0,838
8	Overlap	0,3	0,06	0,425	0,215	0,912	0,83	0,775	0,839
9	Overlap	0,29	0,05	0,448	0,212	0,903	0,837	0,7505	0,830
10	Overlap	0,29	0,1	0,4	0,21	0,908	0,828	0,747	0,8277
11	Overlap	0,28	0,08	0,43	0,21	0,91	0,838	0,777	0,8417
12	Overlap	0,27	0,08	0,44	0,21	0,905	0,826	0,758	0,8297
13	Overlap	0,27	0,1	0,43	0,2	0,906	0,842	0,775	0,841
14	Overlap	0,28	0,08	0,43	0,21	0,916	0,866	0,812	0,865

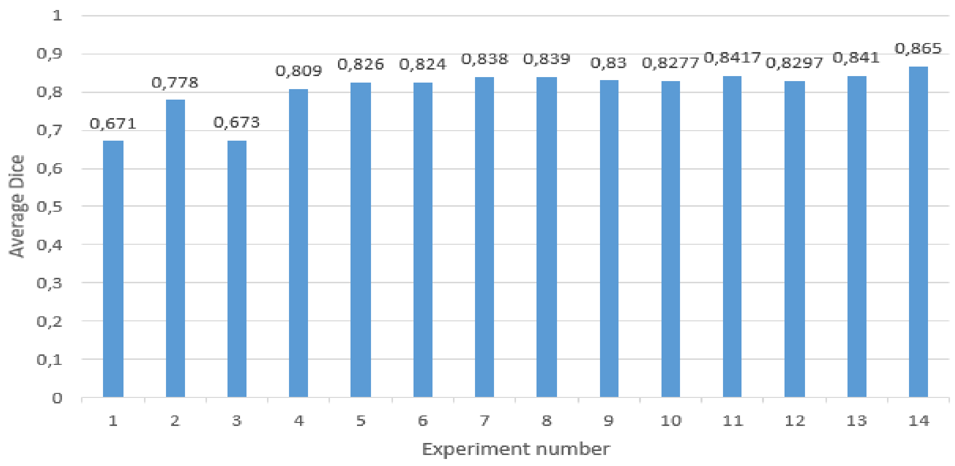


Figure 4.15 – The effect of class-weighting on the CNNs performance. Horizontal axis represents the experiment number in the table 4.3 (i.e., No.). The vertical axis represents the average Dice score of each experiment (Table 4.3 column 10).

1. First: we used Adjacent Patches (i.e., table 4.3, first row), and Overlapping Patches (i.e. table 4.3, second row) without any class-weighting or regularization techniques and as it is shown in the table 4.3 and figure 4.15, the first experiment with Adjacent Patches achieved on Average Dice 0.671. However, in the second experiment with Overlapping Patches achieved on Average Dice 0.778, which is remarkable that the difference between both experiments is 10.7%.

2. Second: we used Adjacent Patches (i.e., table 4.3, third row) and Overlapping Patches (i.e., table 4.3, fourth row), we used also the same mis-classification costs [0.3, 0.2, 0.2, 0.3] for both types of Patches. We have achieved with Adjacent Patches on Average Dice 0.673. However, with Overlapping Patches achieved on Average Dice 0.809, the difference between both experiments is 13.6%.

In summary, training CNNs architectures with Overlapping Patches provides a good segmentation result compared to using adjacent Patches, where we have reported representative results in table 4.3 and in figure 4.15. The experiments demonstrate the effectiveness and the efficiency of Overlapping Patches for training CNNs architectures. Moreover, the last experiment (No=14) provides the best mis-classification costs (The class-weight of Healthy tissue=0.28, the class-weight of Necrotic and Non-Enhancing tumor= 0.08, the class-weight of Peritumoral Edema= 0.43, the class-weight of Enhanced tumor= 0.21).

4.3.5 Experiments and Results

This section presents the results of our experiments and some discussions. Also, our brain tumor segmentation method is evaluated using the BRATS dataset which has real patient data with Glioblastoma brain tumors (both high- and low-grade). Finally, we explain the libraries and frameworks that are used in our work. Then, we evaluate our proposed architectures with a set of metrics that are used in state-of-the-art such as Dice score, Sensitivity, Specificity, and Hausdorff Distance.

4.3.5.1 Dataset Preprocessing

To remove some noises and to enhance the quality of MRI images, especially the MRI images of the BRATS dataset are generated using different MRI devices and acquisition protocols.

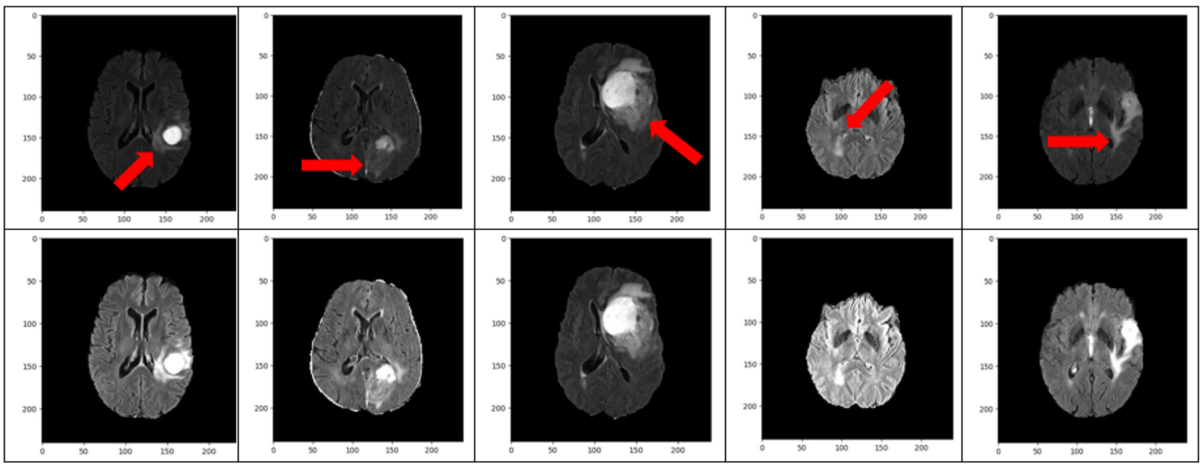


Figure 4.16 – An illustration of an example of 5 Flair scans before (top row) and after (bottom row) the proposed intensity normalization. Arrow red indicates where the images have improved after applying the normalization steps (see bottom row). We show 5 Flair scans only for simplicity.

Our preprocessing and normalization method of each 2D axial slice are as follows:

1. Removing 1% highest and lowest intensities: this technique helps to remove some noises at the tail of the histogram, where this step has provided good results in many research [Havaei et al., 2017; Tustison, 2013].
2. Subtracting the mean and dividing by the standard deviation of non-zero values in all channels: this technique is used to centre and to put the data in the same scale, i.e., bringing the mean intensity value at zero and the variance at one.

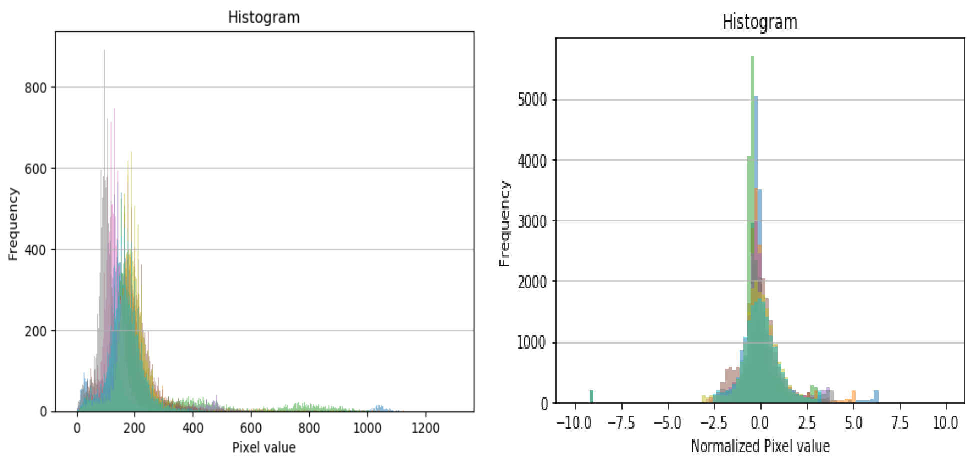


Figure 4.17 – Image intensity histograms of Flair scans of 10 subjects (Each subject is distinguished using different color) from the BRATS 2018 training dataset before (left) and after (right) the intensity normalization steps. We show 10 Flair scans only for simplicity.

3. In this step, we try to isolate the background from the tumoral regions by assigning the minimum values to -9, where it has been observed that using integer numbers between -5 to -15, fits our CNNs architectures. The application of the second pre-processing step led to bring the mean intensity of tumoral region, the healthy tissue + background to zero value as shown in the histogram (See figure 4.17). As we know, the intensity of background pixels of MRI images in BRATS data equals to 0, thus to isolate the zero pixels (i.e. background) from the other regions, we normalized the histogram of the MRI images by shifting the background pixels to another bin. We found that the bin -9 in many experiments, gives good results in the training and testing phases.

The image intensity normalization effect is illustrated with Flair scans in figure 4.16. The results shown in figure 4.16 clearly demonstrated the effectiveness of the proposed intensity normalization steps. Moreover, the improvement is further confirmed by image intensity histograms of 5 subjects from the BRATS 2018 training dataset, as shown in figure 4.17.

4.3.5.2 Post-processing

To remove some mis-classified and isolated regions (i.e., non-tumor regions) from the segmentation results of our proposed CNNs architectures, we have applied two post-processing techniques:

1. Using a global threshold for each slice (i.e., 2D MRI image) to remove small non-tumoral regions based on connected-components, after many experiments (e.g. 80, 100, 150, 200 ..etc), we found that using 110 pixels provide the best result for this global threshold. Thus, any connected-components smaller than 110 pixels will be removed. We refer to this Post-processing as Post-processing 1.
2. The second post-processing is applying morphological opening operators $g(x) = (f \ominus s) \oplus s$, where g(x) is the output and f is the input image, with a structuring element s of size 3 x 3 (see table 4.4). Structuring element in mathematical morphology is a set of pixels, it represents the level of observation, where it is used to differentiate regions in an image based on the size and shape of the structuring element. We used a small structuring element to capture small tumoral regions and to remove them later using an opening operator, in addition, to adding a sharpening effect such as emphasizing differences in adjacent pixels. Opening operator uses erosion $g(x) = f \ominus s$, then dilation $g(x) = f \oplus s$ in succession, where we noticed that this operator improves the results of the first post-processing (i.e., Post-processing 1). We refer to this Post-processing as Post-processing 2.

0	1	0
1	1	1
0	1	0

Table 4.4 – Illustration of the 3×3 structuring element

We have observed that the application of post-processing 1 + post-processing 2 to the output of the CNNs architectures provides a better segmentation result than the application of a single type of these post-processing techniques.

4.3.5.3 Implementation

To implement our proposed CNNs architectures, we used Keras which is a high-level open source Deep Learning library, and Theano [Al-Rfou et al., 2016] as a Back-end, where Theano exploits GPUs to optimize Deep Learning architectures. To compute the inference time, we used Python environment on Windows 64 bits, Intel Xeon processor CPU @ 3.30 GHz with 8 GB DDR4 RAM, and Nvidia Quadro GPU K2000 with 2 GB GDDR5 memory.

One of the problems with artificial neural networks, including deep learning, is the initialization of weights. To overcome this problem, we tested two most used techniques: ”Glorot_Uniform” [Glorot and Bengio, 2010], ”he_normal” [He et al., 2015]. he_normal initialization provides a fast convergence toward the global minimum compared to Glorot_Uniform, also in this work [He et al., 2016] they found that he_normal initialization gives the best results.
For the optimizer, we chose SGD with mini-batch size from 1 to 120, where mini-batch size equals to 8 provides the best results. The initial learning rate (LR) is $LR_0 = 0.001$, then it is deceased by the equation 4.12:

$$LR_i = 10^{-3} \times 0.99^{LR_{i-1}} \tag{4.12}$$

Where LR_i ($i \in \mathbb{N}^+$) is the new learning rate, LR_{i-1} is the learning rate of the last epoch, 0.99 is a decreasing factor.
To avoid the Overfitting problem, we have used two methods: (1) BRATS dataset has only 285 MRI images with both high- and low-grade, this number of training data is small to train a Deep Learning model on a multi-classification problem. Thus, we used a large number of Overlapping Patches (30.660 Patches) that provide a better balance of training data among the four classes (i.e. 4 sub-regions) of Glioblastomas tumors. In addition, Overlapping Patches technique provides a data augmentation [Hashemi et al., 2018] to the original BRATS dataset. (2) Theoretically, the error on the training and validation sets decreases after many epochs, but in practice, the error on the validation set at a certain point will start to increase again (See figure 4.18). Thus, at this point which is the minimum, the training is then stopped using early stopping technique (see equation 4.13).

$$E_o(t) = \min_{t' \leq t} E_v(t') \tag{4.13}$$

Where E_o, E_v are the optimal error (i.e. lowest validation set error) and the validation error, respectively on the validation set.

All our variants of CNNs architectures are trained from scratch using a large number of MRI image Patches (i.e. $64 \times 64 \times 4$) equals to 30.660 Patches. After many experiments, we found that splitting dataset into 70% for training 30% testing is the best distribution,

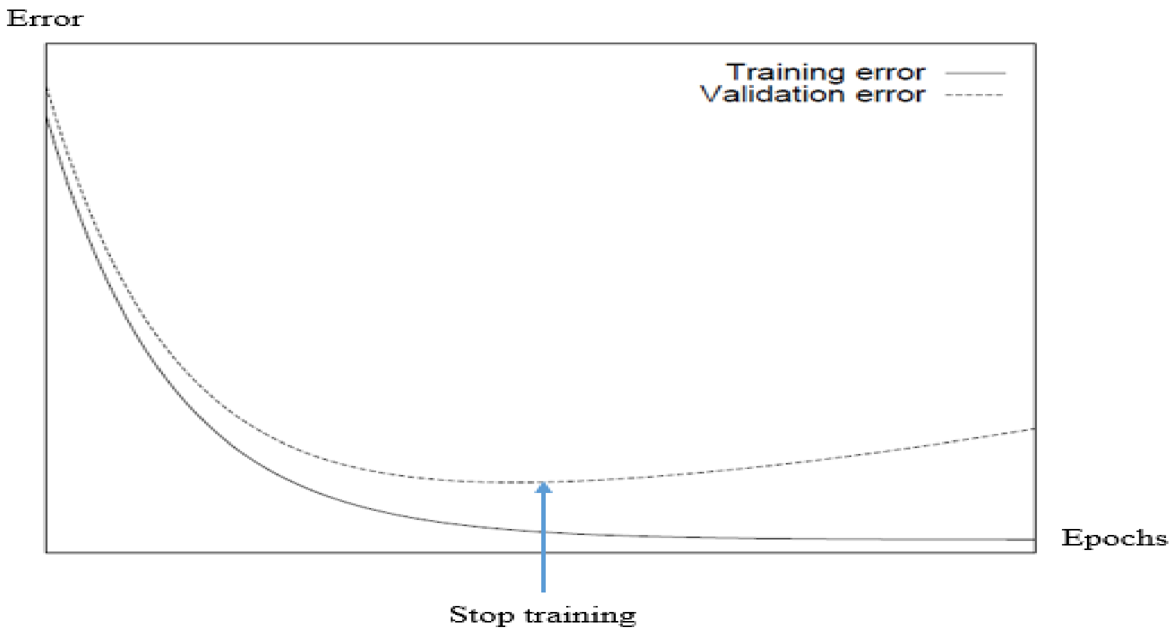


Figure 4.18 – An illustration of the early stopping technique. The figure shows the training and validation error curves. The principal axes are the epochs and the error value. Based on figure 1 [Prechelt, 1998]

then in the validation phase, we used BRATS 2018 validation set which contains 66 MRI images of patients with unknown grade. For the evaluation of our CNNs architectures on the validation dataset, we have used an online evaluation system [13]. We upload the segmentation results and the system provides the quantitative evaluations contain Dice score, Sensitivity, Specificity, and Hausdorff distance. For each one of these metrics we have three sub-metrics: Complete (i.e. Necrotic and Non-Enhancing tumor, Edema, Enhancing tumor), Core (i.e. Necrotic and Non-Enhancing tumor, Enhancing tumor), Enhancing (i.e. Enhancing tumor).

4.3.5.4 Results

4.3.5.4.1 Segmentation performance

In this section, we compare the segmentation performance of our 3 variants of CNNs architectures with the state-of-the-art methods.

Table 4.5 shows a comparison of our proposed Deep CNNs architectures with state-of-the-art methods, these comparisons are based on four metrics: Dice score, Specificity, Sensitivity and Hausdorff distance. It can be observed that there is no single method

13. Center for Biomedical Image Computing and Analytics University of Pennsylvania, Url:https://ipp.cbica.upenn.edu/

that surpasses all methods on all metrics simultaneously even to human rater (i.e., Radiologists, first row) and that due to the difficulties and the challenges (see problem statement section) and also data heterogeneity because of using MRI images acquired from several scanners and using different acquisition protocols. Second observation is: when we applied our architectures with post-processing techniques, i.e., DenseMulti-OCM*+ Post-processing [14] 1 & Post-processing 2, we notice a marked improvement over the segmentation results in terms of Dice score and Hausdorff distance. This improvement demonstrates the effectiveness of our proposed post-processing techniques. Third observation is: methods-based CNNs provide a better segmentation performance compared to machine learning methods such as [Ellwaa et al., 2016] who developed an random forest Iterative method. However, the method of [Ellwaa et al., 2016] provides 100% results in terms of specificity metric on the enhancing tumor class (see table 4.5). Also our method InputSparceMultiOCM and the method of [Chang et al., 2016] have competitive results and that due to the way of building the architectures, where both architectures re-introduce the input (i.e., raw pixels) in the concatenation layer. Moreover, the method

Table 4.5 – Comparison of the segmentation results of our three proposed models which are noted by adding "*" to the architecture's name with the state-of-the-art methods. WT, TC, ET denote Whole Tumor (complete), Tumor Core, Enhancing Tumor core respectively. Post denotes post-processing. Fields with (-) are not mentioned or given with a specific number in the published work. Numbers in bold are the best values.

Methods	Dice			Specificity			Sensitivity			Hausdorff		
	WT	TC	ET	WT	TC	ET	WT	TC	ET	WT	TC	ET
Human Rater [Menze et al., 2015]	0.88	0.93	0.74	-	-	-	-	-	-	-	-	-
[Havaei et al., 2017]	0.88	0.79	0.73	0.89	0.79	0.68	0.87	0.79	**0.80**	-	-	-
[Chang et al., 2016]	0.87	**0.81**	0.72	-	-	-	-	-	-	9.1	**10.1**	6.0
Iterative average (HGG + LGG)[Ellwaa et al., 2016]	0.82	0.72	0.56	0.985	0.99	**1.00**	0.835	0.805	0.63	-	-	-
DeepMedic + CRF [Kamnitsas et al., 2017]	0.847	0.67	0.629	-	-	-	0.876	0.607	0.662	-	-	-
[Zhao et al., 2018]	0.84	0.73	0.62	-	-	-	0.82	0.76	0.67	-	-	-
3CNet [Ben naceur et al., 2018]	0.87	**0.81**	**0.83**	0.74	0.78	0.80	**0.89**	**0.84**	0.74	16.86	18,99	13.67
2D-3D model [Mlynarski et al., 2019]	**0.90**	0.808	0.772	-	-	-	-	-	-	-	-	-
SparseMultiOCM*	0.858	0.746	0.713	0.993	0.996	0.998	0.855	0.756	0.771	21.519	15.160	11.140
SparseMultiOCM* + post 1	0.860	0.748	0.730	0.994	0.996	0.998	0.846	0.756	0.774	8.518	13.466	9.406
SparseMultiOCM* + post 1 & 2	0.860	0.744	0.725	0.994	0.996	0.998	0.842	0.751	0.753	**7.814**	13.522	6.713
InputSparseMultiOCM*	0.864	0.746	0.716	0.993	0.996	0.998	0.863	0.7548	0.784	16.899	14.669	10.104
InputSparseMultiOCM* + post 1	0.866	0.747	0.748	0.994	0.996	0.998	0.856	0.753	0.789	10.1868	11.834	6.142
InputSparseMultiOCM* + post 1 & 2	0.865	0.744	0.745	0.994	0.996	0.998	0.853	0.748	0.770	10.044	11.504	4.608
DenseMultiOCM*	0.862	0.737	0.710	**0.995**	**0.998**	0.998	0.848	0.710	0.76	14.488	15.673	6.582
DenseMultiOCM*+ post 1	0.861	0.736	0.725	**0.995**	**0.998**	0.998	0.841	0.709	0.759	8.514	13.444	5.142
DenseMultiOCM*+ post 1 & 2	0.860	0.733	0.732	**0.995**	**0.998**	0.998	0.838	0.702	0.740	8.521	13.324	**4.406**

14. The drawback of the first post-processing is not a patient-specific, but it is very effective when the brain tumor has only one connected region.

of DeepMedic + CRF [Kamnitsas et al., 2017] provides a good segmentation result for the metric of sensitivity on the whole tumor, but for the rest of metrics all our architectures outperformed the method of DeepMedic + CRF. Furthermore, DeepMedic + CRF, and 2D-3D model [Mlynarski et al., 2019] are computationally expensive and use a lot of memory due to using 3D MRI images and the need to compute the 3D gradients for each forward/backward pass. On the other hand, DenceMultiOCM uses only 2D MRI Patches and that makes it efficient in terms of computational time and in terms of memory. The method of [Mlynarski et al., 2019] obtained good results for the dice metric on the whole tumor class due to using 3D CNNs and using 2D CNNs feature maps as an additional input. [Zhao et al., 2018] developed a CNNs method with an integrated CRF technique in contrast to [Kamnitsas et al., 2017]. The method of [Zhao et al., 2018] predicts the final images by averaging the results from 3 views (axial, sagittal, and coronal). If we compare this method with DenseMultiOCM*+ Post-processing 1 & Post-processing 2, we find that our CNNs architecture outperformed and provides a better segmentation performance than [Zhao et al., 2018] in terms of all metrics except the sensitivity metric on the tumor core as shown in the table 4.5.

Figure 4.19 shows examples of the segmentation results of Glioblastoma with Low-Grade (LGG) and High-Grade (HGG) using 3 variants of CNNs architectures (i.e., SparseMultiOCM, InputSparseMultiOCM, DenceMultiOCM). It can be observed that the proposed architectures detect effectively the tumor region (from left to right, column 3 and 8) and its sub-regions: Red: Necrotic and Non-Enhancing tumor, Green: Peritumoral Edema, Yellow: enhancing tumor. Also, these architectures are able to detect and segment even small tumoral regions without any shift of the segmentation results to another spatial position (e.g. patients in row 2, 6, 10) as shown in figure 4.19, Moreover, we can see that our post-processing techniques improved the spatial segmentation results (from left to right, column 4, 5, 9 and 10). The performance of the segmentation results in figure 4.19 due to (1) Overlapping Patches that introduce more context to the architectures; using different neighborhood around the same pixels helps to take into account several posterior probabilities (Several decisions) which can be maximized over all labels of the training dataset. Thus, the final prediction of the pixel's label of the architectures with respect to their parameters will be based on the vote of majority of decisions. Moreover,

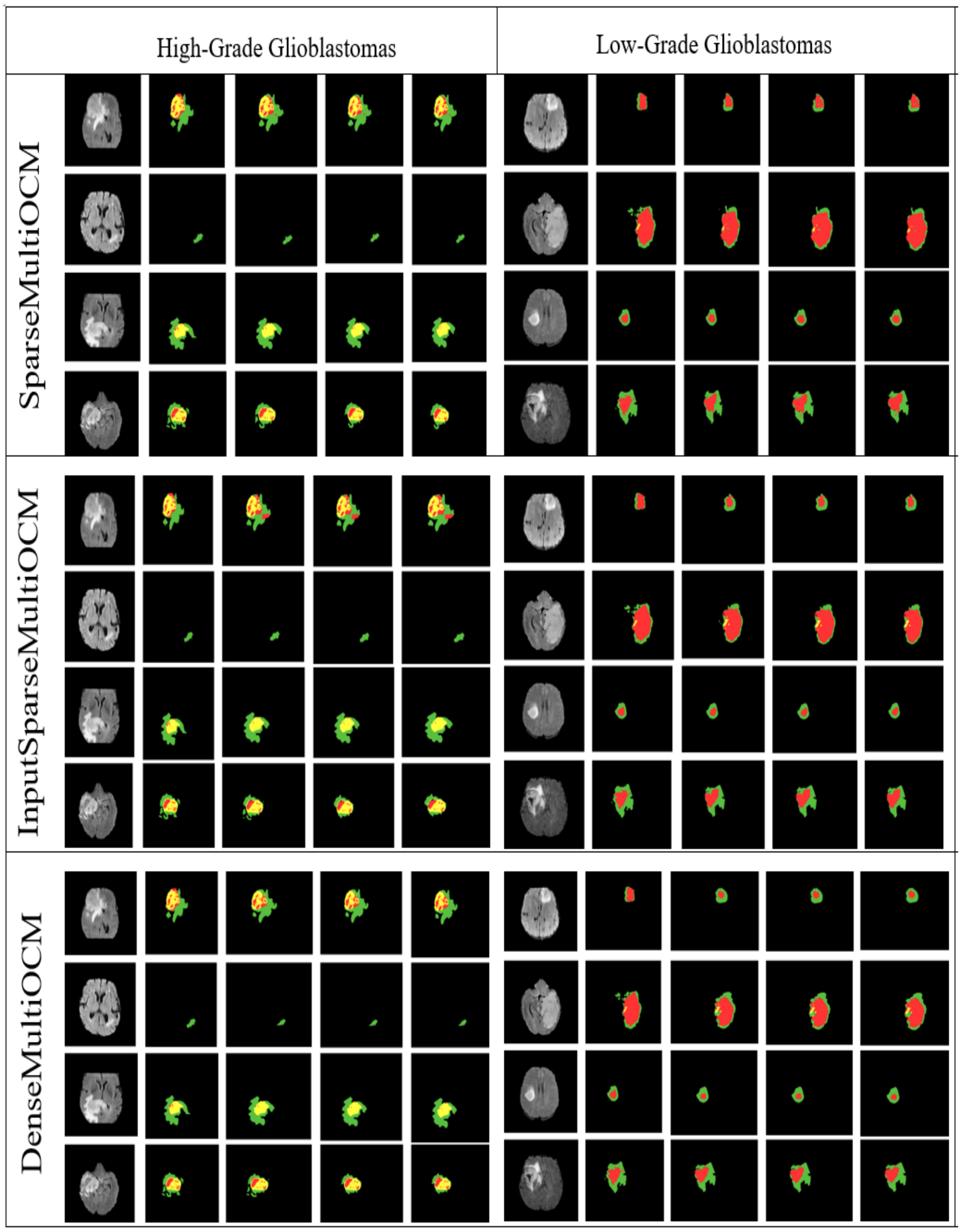


Figure 4.19 – Illustration of three CNNs architectures segmentation results for Glioblastoma brain tumors with High-Grade and Low-Grade. Each row represents segmentation results of a different patient's brain tumor from the Axial view. From left to right: the first five × twelve matrix are High-Grade, and the second five × twelve matrix are Low-Grade. Each architecture has 40 different images. The first and six columns are MR scans of different patients (we show Flair MRI modality only for simplicity), the second and the seventh column are the Ground truth segmentation that are created by the radiologists, the third and the eighth columns are the segmentation results of our 3 variants of CNNs architectures, the fourth and ninth column are the segmentation results of each architecture + post-processing 1, the fifth and the tenth column are the segmentation results of each architecture + post-processing 1 +post-processing 2. Colors indicate the tumor regions: Black: Healthy and background, Red: Necrotic and Non-Enhancing tumor, Green: Peritumoral Edema, Yellow: Enhancing tumor.

Using (2) the technique of *selective attention* that is based on the rule of interconnected modules, this technique plays a very important role of features maximization inside each module.

4.3.5.4.2 Inference time and Computational benefits

Developing a Deep CNNs model and getting a fast Inference time is one of the challenging issues known in state-of-the-art. One of the factors that is very important to

Table 4.6 – Results of training time, inference time and the number of parameters of the state-of-the-art segmentation methods with our 3 proposed models which are noted by adding "*" to the architecture's name. "-" indicates that the information is not provided in the published paper.

Methods	Inference time (seconds)(GPU)	Inference time (seconds)(CPU)	training time (hours)(GPU)	parameters
Radiologist [Kaus et al., 2001]	≈14400	-	-	-
[Havaei et al., 2017]	180 (GPU)	-(-)	- (GPU)	802,368
[Chang et al., 2016]	93 (GPU)	-(-)	2 (GPU)	130,400
Iterative average (HGG + LGG)[Ellwaa et al., 2016]	- (-)	-(-)	- (-)	-
DeepMedic + CRF [Kamnitsas et al., 2017]	30 (GPU)	-(-)	72 (GPU)	-
[Zhao et al., 2018]	540 (GPU)	-(-)	288 (GPU)	-
3CNet [Ben naceur et al., 2018]	19.72 (GPU)	-(-)	5 (GPU)	1,072,420
2D-3D model [Mlynarski et al., 2019]	- (GPU)	-(-)	- (GPU)	-
SparseMultiOCM*	10.77 (GPU)	18.51 (CPU)	1 (GPU)	127,876
InputSparseMultiOCM*	11.68 (GPU)	19.62 (CPU)	1 (GPU)	130,180
DenseMultiOCM*	15.98 (GPU)	24.55 (CPU)	1 (GPU)	181,124

reduce the Inference time is the number of parameters (weights and biases) [Chen et al., 2014], where there is a direct correlation between these two (i.e., the number of parameters and the Inference time). Table 4.6 shows a comparison of our proposed Deep CNNs architectures with the state-of-the-art methods, this comparison is based on 3 metrics: inference time, training time, and the number of parameters. SparseMultiOCM architecture has 8 times less parameters compared to 3CNet with the same GPU, and we have noticed that the Inference time decreased to almost the half: 10.77 seconds (SparseMultiOCM) compared to 19.72 seconds (3CNet). In addition, the inference time on CPU decreased to 18.51 seconds, almost 7% faster. SparseMultiOCM could segment the whole brain tumor only in 10.77 seconds on GPU and 18.51 seconds on CPU, while radiologists spend on average 4 hours [Kaus et al., 2001] to segment only one brain tumor. However, SparseMultiOCM could segment 1415 brain tumor (i.e., 1415 patients) in the same period of time (4 hours ≈ 14400 seconds). Also, it can be observed that our architectures (i.e., SparseMultiOCM, InputSparseMultiOCM, DenceMultiOCM) outperformed all methods that are cited in the table 4.6 in terms of inference time (10.77, 11.68, 15.98 on GPU and 18.51, 19.62, 24.55 on CPU respectively) and the needed time for training (1 hour of training on GPU) with a lower number of parameters.

SparseMultiOCM model decreased a lot the inference time, almost 16 times faster on GPU and 9 times faster on CPU in comparison with the state-of-the-art methods such as [Havaei et al., 2017], and almost 50 times faster on GPU and 29 times faster on CPU than the method of [Zhao et al., 2018] where their CNNs model took 12 days (288 hours) of training.

The computational benefits of our architectures in terms of training time (one hour) and inference time (12 seconds on GPU and 20 seconds on CPU) are due to: (1) features maximization through *selective attention* technique. (2) We used two consecutive layers of convolution, where with this technique we proved that it gives a good performance in terms of segmentation performance and speed [Ben naceur et al., 2018]. (3) Limited number of filters in each layer, as the use of many filters at each layer leads to redundant and noisy features, in addition, the use of small filters (3 x 3) have proved in [Simonyan and Zisserman, 2014] that they keep a lot of information and reduce the number of multiplications between the input (or the previous layer) and the large number of filters. (4) Also, the use of fully convolutional networks [Long et al., 2015] instead of fully connected layers as the classifier, helped the architecture to give accurate results and to reduce the training time due to sparse connectivity. (5) The use of skip connections helps to reduce the time of training [Szegedy et al., 2017]. (6) Applying convolution to multi-resolution features after applying Max-pooling and Up-sampling allows us to detect relevant features with lower parameters and operations, in addition, this technique contributed to reduce the inference time. (7) Finally, The use of Overlapping Patches helps to reduce the Inference time due to using small Patches; it is known that CNNs architecture is a set of feature maps that represent the extracted features from the input data, and the use of small input Patches leads to obtaining small feature maps, thus getting small feature maps size helped to reduce the number of operations and, consequently, the required computational time.

To focus on the effectiveness of Deep learning models, we measured the metric of Dice whole density, i.e., Dice whole divided by the number of parameters. The best model is the one that has a high Dice whole density, in other words, the model that uses effectively its parameters. Measuring information density is one of the most used metrics to measure the effectiveness of deep learning models; information density shows us the ability of each model to use its parameters efficiently [Canziani et al., 2016], [Bianco et al., 2018]. In figure 4.20.a, we can clearly see that the most efficient models are SparseMultiOCM, [Chang et al., 2016] and InputSparseMultiOCM, where their Dice whole density is greater than 6.66. Also, from the figure 4.20.b of density information, it can be seen that although 3CNet [Ben naceur et al., 2018] and [Havaei et al., 2017] have a large number of parameters but they do not take the advantage of their capacity in the learning process; these models have a huge number of parameters that do not contribute to the segmentation results. Moreover, our proposed SparseMultiOCM model is efficiently designed for the given task (brain tumor segmentation problem), it achieved the highest Dice whole density score among the state-of-the-art methods with 9x less parameters.

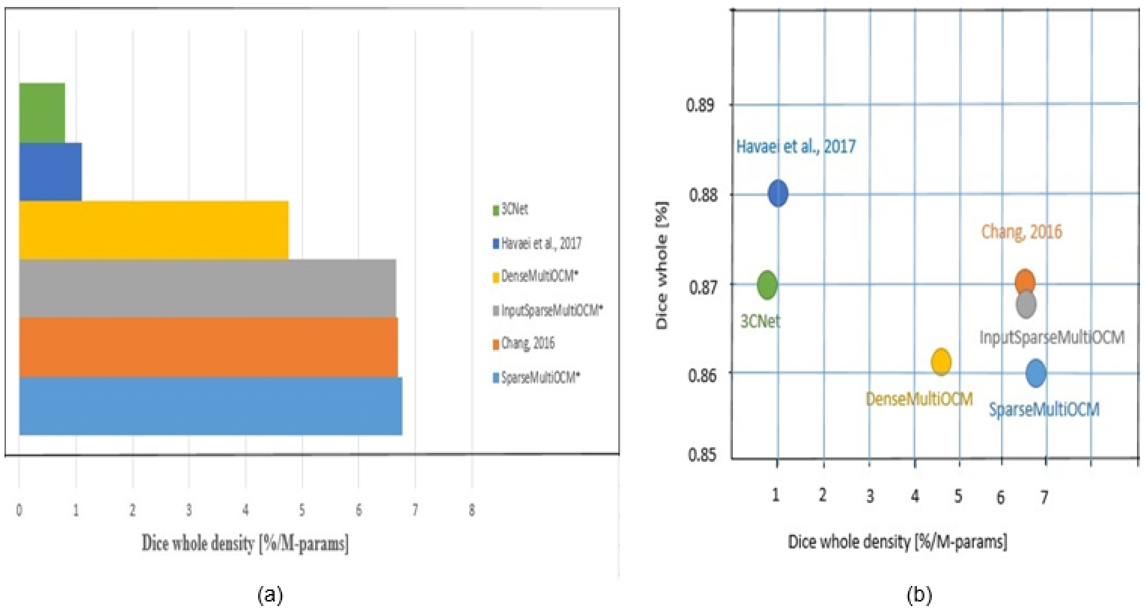


Figure 4.20 – Dice whole density (a) and Dice whole vs. Dice whole density (b). Dice whole Density metric gives us an insight about the efficiency of each model.

As a summary, the design of Deep CNNs architectures is very important, where our proposed architectures (i.e., SparseMultiOCM, InputSparseMultiOCM, DenseMultiOCM) achieve the tradeoff between the segmentation performance and the inference time. Moreover, we achieved inference time on average 12 seconds on GPU and 20 seconds on CPU, i.e., the number of operations and parameters of our proposed architectures are optimized to be executed on GPU and CPU in a small range of 14 seconds compared to the state-of-the-art methods.

4.3.6 Discussion and Conclusion

In this section, we presented a fully automatic brain tumor segmentation method using End-to-End Deep CNNs and MRI images of patients with Glioblastoma brain tumors. Our three variant architectures are built upon 2D Axial images Patches that are equally and randomly sampled from the training dataset. This technique of generating image Patches in addition to using the Weighted Cross-entropy loss function during the training phase helped to solve the issue of unbalanced data.

Moreover, we proposed a new Overlapping Patches technique to model the global context and relationship among image Patches that led the architectures to take into account the local and larger context without using larger Patches. Also, we demonstrated that using Overlapping Patches is better than using Adjacent Patches to train Deep CNNs architectures. The proposed architectures are based on the rule of interconnected modules that are connected by skip connections to build the technique of *selective attention*. The latter technique plays a very important role in maximizing the features representation within each module.

The proposed Deep CNNs architectures in this section (i.e., SparseMultiOCM, InputSparse-MultiOCM, and DenseMultiOCM) perform nearly identically without applying post-processing, due to the way of building those architectures and to the fact of having a similar number of parameters, while it is interesting to note that the architecture of DenseMultiOCM has the best results after applying the post-processing step. This improvement after applying post-processing is an indicator of the importance of this step for reducing the misclassified regions.

The advantages of our proposed architectures are (i) first, these architectures have a similar segmentation performance with radiologists as confirmed by the extensive experiments and the evaluation metrics (i.e. Dice, Sensitivity, Specificity, Hausdorff distance). (ii) The efficient design of our proposed architectures in addition to the slice-by-slice segmentation technique makes these architectures computationally efficient. (iii) These architectures are an End-to-End Deep Learning approach, and fully automatic without any user interventions.

Finally, the obtained results illustrate that our proposed architectures achieved state-of-the-art performance in terms of accuracy and speed, and give a very accurate result for the segmentation of Glioblastoma tumors with low- and high grade.

In the next chapter, we present the issue of training deep learning architectures with unbalanced data, in addition to the issue of misclassified regions (i.e., false positives and false negatives) that are generated by Deep CNNs architectures. Furthermore, those issues (i.e., unbalanced data and misclassified regions) are treated independently in two different sections.

Chapter 5

Deep learning with unbalanced data and misclassified regions

Contents

5.1 Introduction

In this chapter, we solve the limits of training deep learning architectures, in particular, Deep convolutional neural networks with two issues independently. The first issue is unbalanced data where we find a class of interest has a minority of data compared to other classes. The second issue is misclassified regions; Deep CNNs architectures predict tumoral regions where the tumor is located, in addition to false positive regions and false negative regions which are wrong predictions. The experiments are carried out on Glioblastoma MRI images of the BRATS dataset. This chapter aims to develop new techniques for the proposed Deep CNNs architectures in chapter 4 to solve independently these two issues of unbalanced data and misclassified regions. We present in section 5.2 in more detail the issue of Unbalanced data and its impact on the final segmentation results, in addition to our proposed Online Class-weighting function to solve this issue (Unbalanced data). Then, in section 5.3, we present another issue known in Deep learning with especially medical data as misclassified regions (i.e. false positives and false negatives) that are produced by the proposed Deep CNNs architectures in chapter 4.

5.2 Deep learning with Online class-weighting

5.2.1 Problem statement

Current state-of-the-art image segmentation methods are based on Deep CNNs architectures, where in general we find a feature extractor with a bank of trainable parameters. Then, pooling layers to make the images less sensitive and invariant to small translations (i.e. resisting to local translation). The last stage in Deep CNNs architectures is a classifier that classifies each pixel (or voxel) into one of many predefined classes. In the field of medical image classification and segmentation, we can classify methods-based Deep learning into two categories: 2D or 3D models. For the first category (i.e. 2D-based methods), there are many studies such as [Axel et al., 2014; Pereira et al., 2015; Chang et al., 2016; Zhao et al., 2018; Havaei et al., 2017], where these methods use 2-dimensional patches (2D patches) as an input to train Deep CNNs network which is used after that to classify Glioblastoma image data. For the second category (i.e. 3D-based methods), we can cite studies such as [Urban et al., 2014; Kamnitsas et al., 2016, 2017], where these methods use 3-dimensional image patches (3D patches) as an input to Deep CNNs network.

In Deep learning, the most common problem among image segmentation methods is unbalanced data, where we find a class of interest has the minority of data compared to other classes. For instance, as it is shown in (See figure 5.1), healthy class has more than 98% of data compared to other classes of interest: edema class, non-enhancing core and necrotic core class, enhancing core class. This kind of issues makes Artificial Neural Networks, including Convolutional Neural Networks, bias toward the more frequent label (i.e. healthy class). Thus, training a CNNs architecture with this type of data will make predictions with low sensitivity, where the most important part in medical applications is to make the architecture more sensitive toward the lesion-class (i.e. tumoral regions).

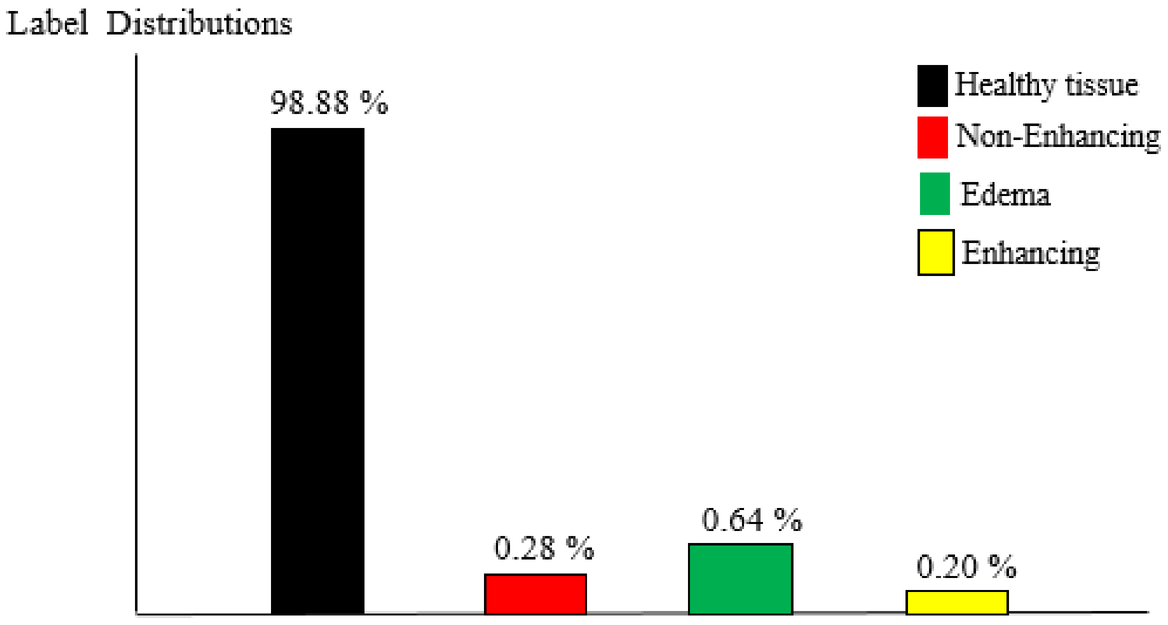


Figure 5.1 – Illustration of labels distributions among the 4 classes in BRATS dataset: edema class, non-enhancing core and necrotic core class, enhancing core class, and healthy tissue class. Healthy tissue has 98% of data, non-enhancing core and necrotic core has 0.28% of data, Edema has 0.64% of data, and enhancing core has 0.20% of data.

The issue of unbalanced data is common across multi-label image segmentation tasks, where there are many proposed methods in state-of-the-art. In overall, there are two categories of solutions to this issue: some methods try to mitigate this problem by proposing equal sampling of training images patches [Lai, 2015; Havaei et al., 2017], on the other hand, some methods propose a new loss functions: cross entropy-based median frequency balancing [Badrinarayanan et al., 2017], cross entropy-based weight map [Ronneberger et al., 2015], combination of sensitivity and specificity [Brosch et al., 2016], asymmetric similarity loss function [Hashemi et al., 2018]. Most of the cited methods define usually a specific function dedicated for well-defined applications, however, they may do not work well for other applications because of the dependence between the proposed function and

the input data.
Our purpose in this chapter is to overcome the issue of training Deep CNNs architectures with unbalanced data that is due to using classical loss functions. To this end, we propose a new method called Online Class-Weighting which is based on weighted cross-entropy loss function to rebalance between the contribution of each class on the final segmentation results. Then, to evaluate the impact and the effectiveness of our new proposed Online Weighted Cross-entropy loss function, we train a Deep CNNs architecture with Online Class-Weighting function to tackle and improve the architecture's accuracy proposed in chapter 4 for the issue of fully automatic brain tumor segmentation. In the next section, we present our proposed Online class-weighting for solving the issue of training Deep CNNs architectures with unbalanced data.

5.2.2 Online class-weighting approach

Cross-entropy loss function is one of most used loss function in multi-classification problems (See section 3.2.2), in particular, image-related applications [Badrinarayanan et al., 2017; Ronneberger et al., 2015]. However, the issue of cross-entropy loss function is that it biases toward to class of the majority of data, in our case, toward the healthy tissue. As mentioned previously in chapter 3, for a multi-classification problem the forward propogation computes the inference (i.e. estimated probability) of CNNs architecture by using Softmax function (See equation 5.1):

$$p_j = \frac{\exp^{z_j}}{\sum\limits_{k=1}^{n} \exp^{z_k}} \qquad for\ j \in \{1, .., n\} \tag{5.1}$$

Where $p_j \in [0, 1]$ is the estimated probability of the class *"j"*, with $j \in \{1, .., n\}$, and n is the number of classes, *"z"* is the neuron's input of the output layer.

Then, CNNs architectures in multi-classification problem, use cross-entropy as a loss function (See equation 5.2) to compute the error $Loss(p, q)_j$ between the *ground truth* and the predicted image during the training phase. The latter loss functun (i.e. cross-entropy) takes as input the estimated probability p_j and the *ground truth* q_j.

$$Loss(p,q)_j = -\frac{1}{n}(\times \sum_j q_j \times \log(p_j)) \tag{5.2}$$

As mentioned previously, cross-entropy loss function (See equation 5.2) biases toward the class of majority of data. And to make this loss function insensitive and unbiased, we propose an Online Class-Weighting function that works by introducing, first, weighting factors W to the cross-entropy function to obtain Weighted cross-entropy loss function (See equation 5.3):

$$Loss(p,q)_j = -\frac{1}{n}(\sum_j W_j \times q_j \times \log(p_j)) \quad W_j \in [0,1] \;\&\; \sum_{j=1}^{n} W_j = 1 \tag{5.3}$$

Where Loss $(q,p)_j$ is the loss function that represents the error between the estimated probability $p_j \in$ [0,1] and the *ground truth* class $q_j \in$ {0,1}. n is the number of classes, $w_j \in$ [0, 1] is the weighting factor assigned to the class j. The *ground-truth* q_j is a *one-hot* probability distribution over the n classes. *One-hot encoding* gives all probabilities of the *ground truth* to one class (i.e. the correct class), and the other classes become zero. For instance, vector q = [1,0,0,0] indicates that the correct class among the four classes is the first class with value "1". In this case, to calculate the overall error using equation 5.3, we need only one operation instead of many operations for all classes.

The estimated probability is computed using Softmax function (See equation 5.1) which is used in the last layer of Neural Network after a forward propagation, where this function squashes the output to become between 0 and 1 as probabilities. Then, these probabilities are fed into Weighted cross-entropy function to compute the error between the estimated probability and the *ground truth*. The latter function defines a weighting factor $w_j \in$ [0, 1] assigned to the class j. The main issue of the Weighted cross-entropy loss function is how to find the weighting factors w_j for each class, in our case, we have four classes: healthy tissue and background class, non-enhancing core class, edema class, contrast-enhancing core class. To overcome this issue, we propose Online Class-Weighting function that allows to find the weighting factors w_j for each class during the training phase using the training rate progress of each class with respect to different training iterations. Online Class-Weighting function starts by initializing the weighting factors based on computed measurements such as an evaluation metric or an error's change. In the beginning, the Online Class-Weighting function measures the performance by evaluating the training results with and without the initialization of the weighting factors of each class. This evaluation helps to measure the training accuracy of each class. After that, new training is launched while rewarding each class with an adapted weighting factor w_j until the Online Class-Weighting function reaches the stopping criteria (e.g., the number of iterations). Weighting factor w_j is computed based on the previous accuracy of the corresponding class. Furthermore, Online class-weighting, is generic and it does not has any *prior knowldege* about the data, in addition, it can be applied with any number of classes and suitable for different possible weighting factor initialization and evaluation metrics. Online Class-Weighting algorithm describes as follows:

Algorithm 3: Online Class-Weighting

```
Data: training dataset of $m$ instances $(X_{1:m}, Y_{1:n})$, where an input image
      $X_t \in \mathbb{R}^{d\times d}$, and a target image $Y_j \in \mathbb{N}^{d\times d}$, $j \in \{1, ..., n\}$ and $n$ is the
      number of classes. $Z$, $S$ are respectively the total number of epoches and
      a set of epoches. $Z$, $S$, $\in \mathbb{N}^+$, where $0 < S < Z$.
Result: a set of weighting factors $\{w_j, / \sum_{j=1}^{n} w_j = 1, w_j \in [0, 1]\}$, and the final
      evaluation results $M_j$ for each class $j$.
1  Initialization of the $n$ weighting factors $w_{1,j} = 1$
2  Launching training for $S$ epoches with the initialized weighting factors $w_{1,j}$
3  Evaluation of the first training results for each class $(M_{1,j})$
4  Re-initialization of Weighting factors $w_{2,j}$ where $w_{2,j}$ = f(X), $\sum_{j=1}^{n} w_{2,j} = 1$
5  Launching second training for $S$ epoches with the initialized $w_{2,j}$
6  Re-evaluation of the second training results for each class $(M_{2,j})$
7  Computing stopping criteria: $C$ = g(Error$_2$(training), Error$_2$(testing))
8  Initialization of the iteration index: $i = 3$
9  while (stopping criteria is not satisfied) do
10     Updating the weighting factors: $w_{i,j} = h(M_{i-1,j}, M_{i-2,j})$, where $\sum_{j=1}^{n} w_{i,j} = 1$
        and $i$ is the iteration index of the current $S$ epoches
11     Launching training with the updated weighting factors $w_{i,j}$
12     Re-evaluation of the training results for each class $(M_{i,j})$
13     Computing stopping criteria: $C$ = g(Error$_i$(training), Error$_i$(testing))
14     Iteration index $i$ equals to: $i = i + 1$
15 end while
```

The algorithm of Online Class-Weighting works as a policy for the assessment of a Deep Learning model performance. Lines from 1 to 6 evaluate the behavior of a Deep learning model without and with applying the weighting factors. This assessment consists of computing an evaluation metric (e.g. Dice coefficient, area under precision-recall curve [Hashemi et al., 2018]) of the training results for each class, i.e., for Healthy tissue class, Necrosis and Non-enhancing tumor class, Edema class, and Enhancing tumor class. After that, we use these measurements to calculate the change in these weighting factors over the training phase. Line 7 computes the stopping criteria based on the training and testing errors. Lines 10 and 11 calculate the new weighting factors based on the evaluation metrics between the current training iteration and the two previous ones. To prove the effectiveness of Online Class-Weighting approach, we carry out a series of experiment in the next section.

5.2.3 Experiments and Results

5.2.3.1 Online Class-Weighting hyperparameters

After the definition of the Online Class-Weighting algorithm, here we introduce different hyperparameters:

1. We initialize the weighting factors W_j using two methods
   - Mean distributions: each weighing factor W_j, where $j \in \{1, ..., n\}$ and n is the number of classes, is initialized according to mean distributions of all classes: Healthy regions class, Necrosis, and Non-enhancing tumor class, Edema class, and Enhancing tumor class (See equation 5.4).

$$W_j = f(X) = \frac{1}{n}\sum_{j=1}^{n} X_j \tag{5.4}$$

     Where W_j is the weighting factor of the class ”j”, with $j \in [1, ..., n]$ and ”n” is the number of classes, X_j is frequency distributions of the class ”j”.
   - Relative frequency distributions: each weighting factor W_j of each class is initialized according to its distribution (See figure 5.1, $C_1 = 98.88, C_2 = 0.28, C_3 = 0.64, C_4 = 0.20$, where C is the class number) in the BRATS dataset (see equation 5.5)

$$W_j = f(X) = \frac{CF_j}{\sum_{j=1}^{n} X_j} \tag{5.5}$$

Where CF_j is the class frequency of the class "j", with $j \in [1, ..., n]$ and "n" is the number of classes, X_j is frequency distributions of the class "j".

As it is shown in table 5.1, the Online Class-Weighting algorithm with Mean distribution initialization provides better segmentation results than with Relative frequency distributions. And this performance is due to the fact that the Mean distribution has an equal distance to all classes compared to the relative frequency distributions that weight the minority classes more than the dominant class (i.e. Healthy tissue).

Table 5.1 – Results of Online Class-weighting with two methods of initialization. For each experiment, we show the Dice score of 10 MRI images from the BRATS-2018 dataset.

Experiment	Initialization	Dice score		
		Complete	Core	Enhancing
1	Relative frequency distributions	0,88	0,79	0,72
2	Mean distributions	0.88	0.83	0.75

2. The second point is about the selection of the best candidates for each iteration (i) during the training phase,where the selection process is carried out as an equation of distance between the training results of iteration $(i-1)$ and iteration $(i-2)$. After each S epoches (i.e. $S = 5$) of training, the algorithm updates the weighting factor according to the equation h: $w_{i,j} = h(M_{i-1,j}, M_{i-2,j})$, where M$_{i-1,j}$ and M$_{i-2,j}$ are the Dice coefficient of the training iteration $(i-1)$ and $(i-2)$ respectively for class "j" where $j \in \{1, ..., n\}$ and "n" is the number of classes. The equation h elects two candidates in each iteration as follows:

$$Candidate(class)_u = \underset{j\in[1:n]}{Argmax}(M_{i-1,1} - M_{i-2,1}), ...(M_{i-1,4} - M_{i-2,4)} \tag{5.6}$$

$$Candidate(class)_v = \underset{j\in[1:n]}{Argmin}(M_{i-1,1} - M_{i-2,1}), ...(M_{i-1,4} - M_{i-2,4)} \tag{5.7}$$

Where in each iteration the Online Class-Weighting algorithm chooses two candidates (u and v); equation 5.6 returns the class u that showed an improvement for the last S epoches, in this case the algorithm rewards this class. And equation 5.7 returns the class v that did not get an improvement because it is stuck in the suboptimal point at the local minima of the loss function. Therefore, the stuck class has more frequent labels in the dataset, in this case the algorithm of Online class-weighting penalizes this class.

3. After selecting the best candidates, the Online Class-Weighting algorithm computes the new weighting factors as follows:

$$W_{i,u} = W_{i-1,u} + \beta \qquad W_{i,v} = W_{i-1,v} - \beta \tag{5.8}$$

   Where $W_{i,u}$, and $W_{i,v}$ are the best candidates for the iteration i, β is the rate of weight's change, where β in our experiments is initialized to 0.2 for obtaining a soft weight's change between two successive iterations, and it provided good segmentation performance.

4. The stopping criteria of Online Class-Weighting algorithm is computed as follows:

   (a) If the error's change (EC) between training and testing sets is greater than a threshold th equals, in our experiment to $th = 0.1$:
   EC = | Error(testing) - Error(training) | $> th$.

   (b) If testing set accuracy does not improve for a number of epoches K. We found $K = 20$ provides the best results in our experiments (See figure 5.2).

   (c) If the above two criteria (i.e. a and b) are always false, thus the algorithm stops after a defined number of epoches Z, where in our case study $Z = 240$.

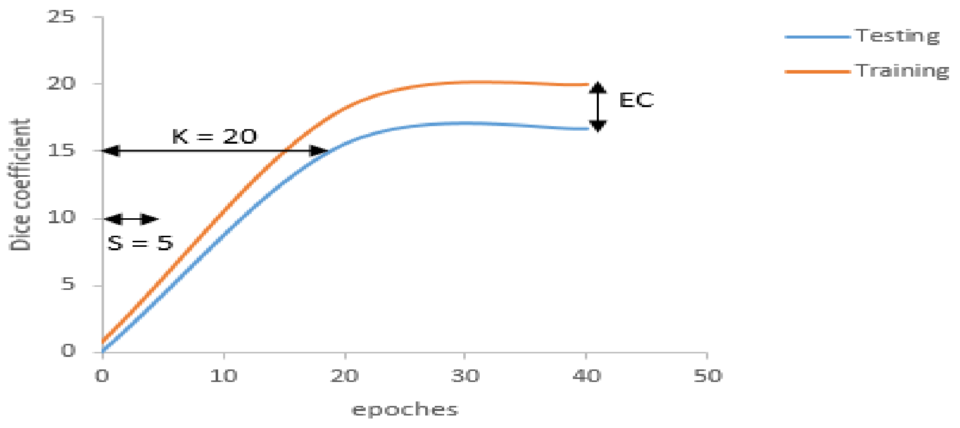


Figure 5.2 – Illustration of an example of the stopping criteria curve. EC is the Error's change, K is the total number of epochs, S is the number of iterations. The online class weighting algorithm stops after satisfying the following conditions: EC, K, S periods.

5.2.3.2 Results

To demonstrate the performance of the Online Class-Weighting algorithm, we compare our loss function with state-of-the art Cross-entropy and Focal loss [Lin et al., 2017] functions. Table 5.2 shows the results of our proposed DenseMultiOCM architecture (See section 4.3.2.2.2) with 3 loss functions: Focal loss, Cross-Entropy, and Online Class-Weighting. As it is shown in table 5.2, DenseMultiOCM + Online Class-weighting outperform Focal loss in all metrics: Dice score of Healthy, NCR/NET, Edema, Enhancing, Complete, and Core. Moreover, DenseMultiOCM + Online Class-weighting have competitive results with Cross-Entropy loss function on 3 metrics: Dice score of Healthy, NCR/NET, and Edema. Consequently, DenseMultiOCM + Online Class-weighting algorithm outperforms the results of Cross-Entropy and Focal loss function, in addition to providing the best results on 5 over 6 metrics.

Table 5.2 – Results of Online Class-weighting against Cross-entropy and Focal loss functions. Healthy, Necrosis and Non-enhancing tumor (NCR/NET), Edema, Enhancing tumor represent the different Glioblastoma tumor subregions segmentation results of DenseMultiOCM architecture with 3 different loss functions: Focal loss, Cross-Entropy, and Online Class-Weighting. Moreover, for each experiment we show the Dice score of 10 MRI images from BRATS-2018 dataset. Values in bold are the best results.

Methods	Dice score					
	Healthy	NCR/NET	Edema	Enhancing	Complete	Core
DenseMultiOCM + Focal loss	0.99	0.38	0.70	0.60	0.79	0.62
DenseMultiOCM + Cross-Entropy loss	0.99	**0.63**	0.80	0.72	0.87	0.81
DenseMultiOCM + Online Class-weighting	**0,99**	0,61	**0,80**	**0,75**	**0,88**	**0,83**

Table 5.3 illustrates the results of the segmentation performance of state-of-the-art methods. As it is shown, our proposed DenseMultiOCM architecture with Online Class-Weighting improved the segmentation results in terms of Dice, sensitivity, specificity, Hausdorff distance. Moreover, DenseMultiOCM architecture + Online Class-Weighting with/without post-processing step provide a better segmentation performance than [Axel et al., 2014; Zhao et al., 2018; Kamnitsas et al., 2017] in terms of Dice score (more than 3%) and Specificity (more than 12%). Also, DenseMultiOCM with Online Class-Weighting obtained competitive results with [Pereira et al., 2015; Chang et al., 2016; Havaei et al., 2017; Urban et al., 2014] in terms of Dice coefficient and Hausdorff distance. According to the standard deviation $std \in [\pm 0.01, \pm 0.004]$ in table 5.3, DenseMultiOCM architecture with Online Class-Weighting could achieve 100% prediction on some patients MRI images in terms of specificity, i.e. the prediction of DenseMultiOCM architecture with Online Class-Weighting corresponds to the manual segmentation of radiologists.

Table 5.3 – Segmentation results of our proposed DenseMultiOCM and Online Class-Weighting (DOCW) method with state-of-the-art methods on Brats dataset. WT, TC, ET denote Whole Tumor (complete), Tumor Core, Enhancing Tumor core respectively. (±) is the standard deviation. Post 1 and Post 2 denote post-processing 1 and post-processing 2 respectively. Fields with (-) are not mentioned in the published work.

Methods	Dice score			Sensitivity			Specificity			Hausdorff		
	WT	TC	ET	WT	TC	ET	WT	TC	ET	WT	TC	ET
[Urban et al., 2014]	0.88	0.83	0.72	-	-	-	-	-	-	-	-	-
[Axel et al., 2014]	0.79	0.68	0.57	-	-	-	0.79	0.67	0.63	-	-	-
[Pereira et al., 2015]	0.87	0.73	0.68	-	-	-	0.86	0.77	0.70	-	-	-
[Chang et al., 2016]	0.87	0.81	0.72	-	-	-	-	-	-	9.1	10.1	6.0
[Havaei et al., 2017]	0.88	0.79	0.73	0.89	0.79	0.68	0.87	0.79	0.80	-	-	-
[Kamnitsas et al., 2017]	0.847	0.67	0.629	-	-	-	0.876	0.607	0.662	-	-	-
[Zhao et al., 2018]	0.84	0.73	0.62	-	-	-	0.82	0.76	0.67	-	-	-
DOCW	0.863 ±0.1	0.752 ±0.2	0.71 ±0.3	0.863 ±0.2	0.763 ±0.3	0.774 ±0.3	0.992 ±0.01	0.996 ±0.01	0.998 ±0.003	23.37 ±28.2	18.55 ±23.7	11.97 ±24.6
DOCW + Post 1	0.87 ±0.1	0.753 ±0.2	0.73 ±0.3	0.86 ±0.2	0.76 ±0.3	0.776 ±0.3	0.994 ±0.01	0.996 ±0.01	0.998 ±0.004	14.71 ±22.5	16.33 ±21.8	9.83 ±20.7
DOCW + Post 1 + Post 2	0.864 ±0.1	0.75 ±0.2	0.72 ±0.3	0.853 ±0.2	0.76 ±0.3	0.76 ±0.3	0.994 ±0.01	0.996 ±0.01	0.998 ±0.003	14.63 ±22.4	16.37 ±21.9	8.14 ±20.04

[Havaei et al., 2017; Zhao et al., 2018] proposed a method based on an equal sampling of training images patches to solve the problem of unbalanced data. However, this method mitigates the impact of unbalanced data problem on the segmentation performance but it does not solve it completely. In our experiments, we also trained the proposed DenseMultiOCM architecture + Online Class-Weighting with the technique of equal sampling of training images but we failed to obtain better results. Thus, in this case, methods-based loss function are the best choice and more robust. Cross entropy-based median frequency balancing [Badrinarayanan et al., 2017], cross entropy-based weight map [Ronneberger et al., 2015] propose a modified cross-entropy loss function, but each of which adapts this function to a specific application, which is not the case with our proposed Online Class-Weighting loss function. Online Class-Weighting does not assume any distribution or *prior knowledge* about the application fields before the training phase; all the updating of the weighting factors w_j from iteration to another is done based on the training results of the 4 classes during the training phase. In addition, one of the important metrics for a clinical decision support system [Hashemi et al., 2018] is sensitivity, where DenseMulti-OCM with Online Class-Weighting achieved high accuracy on the sensitivity metric (See table 5.3). As a summary, our proposed DenseMultiOCM with Online Class-Weighting achieved high segmentation results, and so did with the Post-processing techniques, where the Post-processing 2 improved a lot the Hausdorff distance metric. The first conclusion from these segmentation results is that Gioblastoma brain tumors contain in most cases one connected region, i.e. we have demonstrated this hypothesis after applying the Post-processing 1, where we have seen an improvement in the segmentation performance. The second conclusion is that the class of Enhancing tumor does not have much border with the healthy tissue, i.e. this is demonstrated after applying the Post-processing 2, where we have seen a decrease in the surface of mis-classified Enhancing tumor region and an improvement in the Hausdorff distance metric.

5.2.4 Discussion and Conclusion

In this section, we presented our proposed Online Class-Weighting algorithm for solving the issue of unbalanced data. The performance and efficiency of this algorithm are demonstrated through a series of experiments and a comparison with state-of-the-art methods. Moreover, the architecture of DenseMultiOCM + Online Class-Weighting + post-processing step 1 (DOCW + Post 1) showed high performance in detecting Glioblastoma tumors compared to Cross-Entropy, Focal loss functions, and state-of-the-art methods.

The advantage of Online Class-Weighting is due to the fact that it optimizes the weighting factors w_j for each class during the training phase by searching for the best hyperparameters using the training progress results with respect to each class. Online Class-Weighting is a generic loss function and it can be applied to different input data and different applications.

In the next section (Section 5.3), we present the issue of misclassified regions (i.e. false positive and false negatives regions) and its impact on the segmentation performance, in addition to our proposed solution to this issue (i.e. misclassified regions). Where the proposed solution in the following section is independent of the issue of unbalanced data presented in section 5.2.

5.3 Boosting performance using deep transfer learning approach

5.3.1 Problem statement

Brain tumor segmentation through MRI images analysis is one of the most challenging issues in the medical field. Among these issues, Glioblastoma (GBM) brain tumors invade the surrounding tissue rather than displacing it, causing unclear boundaries, furthermore, Glioblastoma tumors in patients MRI scans have the same appearance as Gliosis, Stroke, Inflammation, and blood spots. Moreover, the second challenging issue is that automatic brain tumor segmentation methods can predict tumoral regions, in addition to, it produces misclassified regions.

In this section, we focus on two major issues in misclassified regions: (1) false positive regions – where the model predicts non-tumor regions as tumor regions but in fact they are not-, and (2) false negative regions – where the model classifies some regions as non-tumor regions but in fact they are. In chapter 4 we addressed the problem of false positive regions by two steps: we used a global threshold for each slice to remove small non-tumoral regions based on connected-components, then in the second step, to enhance the post-processing step more we used a morphological opening operator. Despite the success of these two post-processing steps, further steps are required to improve the segmentation results. In addition, these steps are not patient-specific, i.e., they do not apply to MRI images of all patients. The main reason of these two issues (i.e. false positive regions and false negative regions) is the classifier of Deep CNNs, where in our case the classifier is the Softmax function. Softmax function gives an estimated vector at the end after each forward propagation of Deep CNNs, by normalizing the outputs to stay between 0 and 1, i.e., the outputs become as probabilities. Then, we pick the result of the forward pass based on the maximum probability among all probabilities. And this maximum probability represents the most potential class and the closest to the ground truth, in our case, one class out of the 4 predefined classes, i.e., Necrotic and Non-Enhancing tumor, Peritumoral Edema, Enhancing tumor and healthy tissue. Softmax function is a simple and suitable classifier for the training phase, but it is not adequate for the prediction phase and the problem of instance segmentation. To overcome this issue, we investigate two Glioblastoma brain tumors segmentation pipelines, where we extended the proposed Deep CNNs architectures in chapter 4 to the issue of misclassified regions. The first step in this pipeline is feature maps extraction from the last layer of the Deep CNNs architecture. Then, in the second step, the extracted features maps become the dataset of training and testing to other machine learning algorithms. The third step depends on the type of pipeline, where for the first pipeline we combine the results of the Softmax classifier with two other algorithms Random forest (RF) and Logistic regression (LR) using a voting function. For the second pipeline, the extracted features maps are successively fed to two algorithms: principal component analysis and support vector machine (PCA-SVM).

5.3.2 Methods

5.3.2.1 Fusion solution-based predictions aggregation

In this section, to overcome the issue of misclassified regions in particular false positives and false negatives. We used our proposed SparseMultiCOM architecture in chapter 4 (See section 4.3.2.2.2) to extract for each patient's MRI images hierarchical features from final convolutional layers before Softmax because the last hidden layer has the largest receptive field and more representative higher-level features than the lower-level layers in early layers. Then, we save these features in files for using them later in training and testing different machine learning methods.

In 1962, Hubel and Wiesel [Hubel and Wiesel, 1962] discovered that each type of neurons in the visual system, responds to one specific feature: vertical lines, horizontal lines, shapes, ...etc. From this discovery, the algorithm of CNNs was developed [Fukushima, 1980; LeCun et al., 1989]. The algorithm of CNNs detects lower-level features such as lines, edges,...etc in lower layers of the architecture. And higher-level features such as shapes and objects in higher layers of the architecture. These hierarchical features help CNNs algorithm to better locate the boundaries of the tumor regions with the boundaries of healthy tissue. As it is shown in figure 5.3, SparseMultiCOM architecture is able to detect a lot of features about Glioblastoma brain tumors, in particular, the boundaries

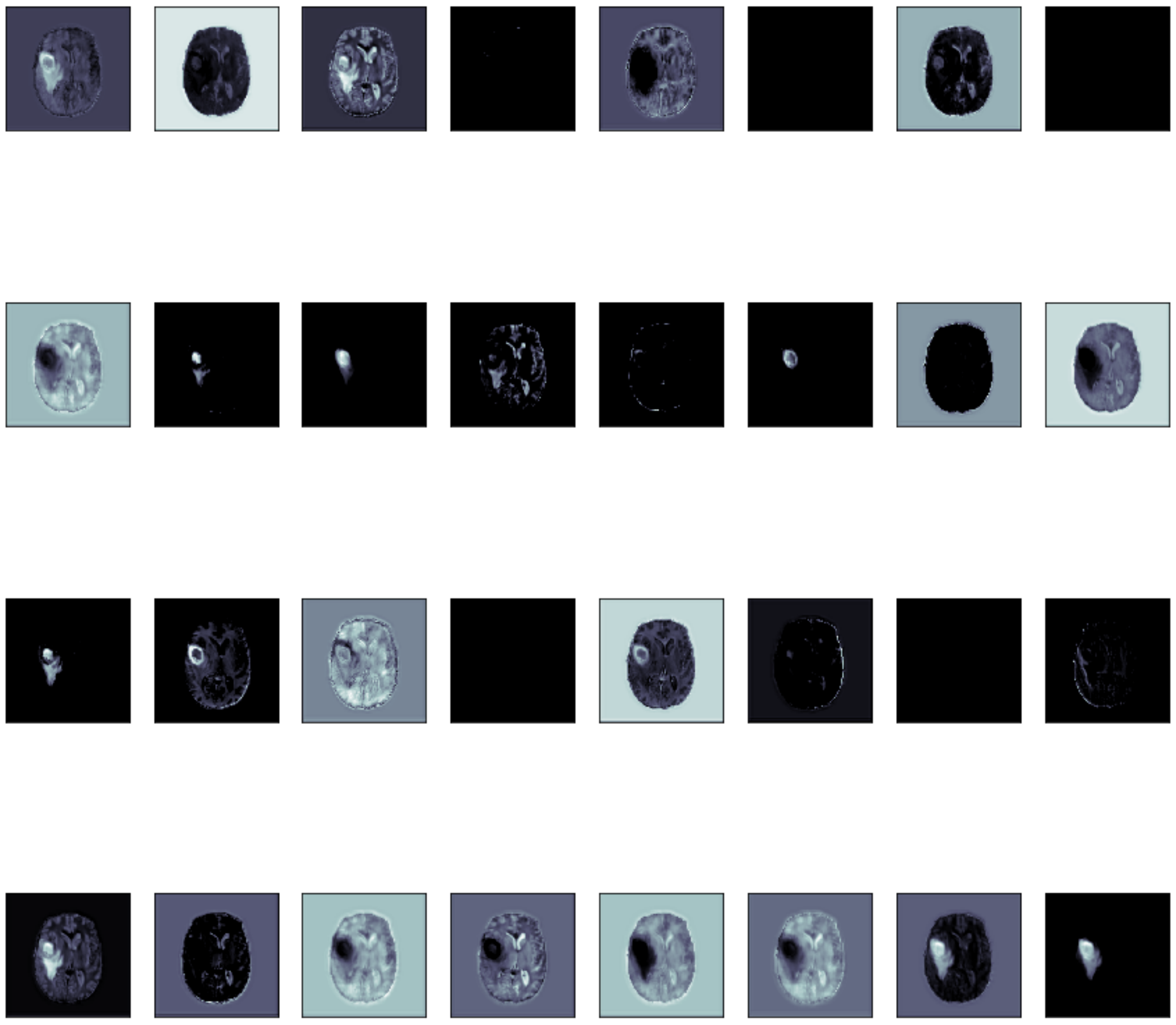


Figure 5.3 – Feature maps visualization of layer 16 of SparseMultiCOM architecture. Layer 16 has 32 features maps, where each image in this figure represents a feature map. Feature maps in CNNs architectures have filters to detect specific features the most representative in the patients' MRI images.

of the tumor regions with healthy tissue and background. Moreover, we can see that some neurons in figure 5.3 such as in the first line, the third, sixth, and eighth images do not respond to any neurons, which can be used in futures to tune these neurons, and consequently, improving the segmentation performance.

Our proposed approach is a hybrid approach between the deep learned features extracted from the last convolutional layer and machine learning methods. Moreover, our proposed approach is based on 3 steps: first, we trained SparseMultiCOM architecture from scratch, the second step is feature maps extraction, then these features are stored in two files (for training and testing). The final step is training and testing two machine learning methods and combining their segmentation results with the result of Softmax in one MRI image for each patient. We use two machine learning methods to learn specific features to each patient's MRI image, in addition, these methods reduce significantly the computational cost compared to using deep learning end-to-end architectures.

The flowchart of our proposed pipeline to detect Glioblastoma brain tumors (See figure 5.3) is composed of two parts: the first part is SparseMultiCOM architecture and the second part is composed of machine learning methods such as Random forest (RF) and Logistic regression (LR). To integrate the LR and RF methods in SparseMultiCOM architecture, we first extracted the feature maps of the last hidden layer before the output (without softmax normalization). Then, we store these extracted feature maps in two files for training and testing. After that, we trained the two aforementioned algorithms (i.e., LR and RF). Moreover, in the training phase, we used the technique of hybrid Random-Grid search (See figure 5.8) to optimize the hyperparameters, for random forest, the best hyperparameters are: number of trees=500, depth=70, purity criterion=Gini index. After the training phase, we combined the segmentation results of these three classifiers (LR, RF, and Softmax) into one result to diagnose the presence of tumor in each extracted feature maps and to diagnose the class of this tumor in each pixel for each patient's MRI images.

To develop a Deep CNNs architecture, there are either a pixel-wise approach or a patch-wise approach. The first approach deals with pixels, while the second approach deals with patches. In practice, the patch-wise classification approach provides good segmentation results compared to pixel-wise classification [Long et al., 2015]. In addition, it is less prone to overfitting, and these advantages are due to the parameters sharing between neurons in the network. Patch-wise approach takes as input patches with limited size, in our case, patches with size 64 x 64 x 4 (4 is the number of MRI modalities).

After training phase of SparseMultiCOM architecture (See section 4.3.2.2.2), we extracted feature maps of the last layer before applying the Softmax classifier to train two other classifiers: LR, RF. The motivation behind the choice of these algorithms: RF uses the technique of ensemble learning by creating many decision trees on a subset of the data,

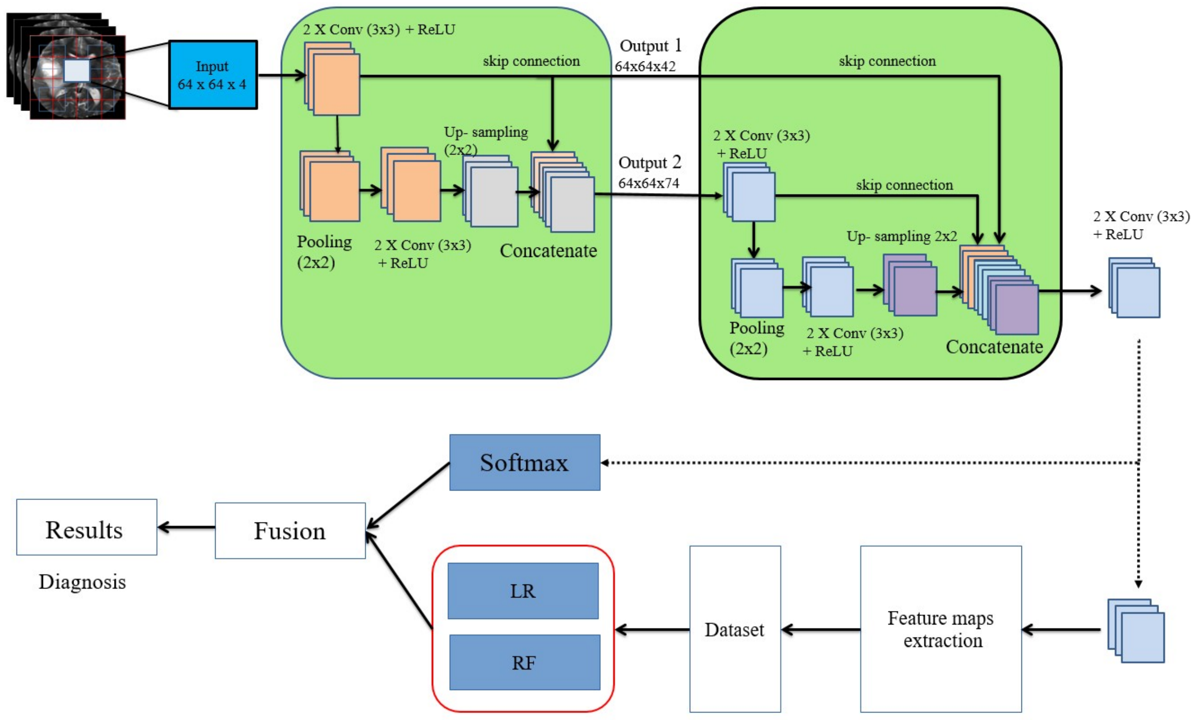


Figure 5.4 – Flowchart fusion solution for predicting Glioblastoma brain tumors. This flowchart has two parts: the first part (two green boxes) represent the DCNNs architecture, the second part (three blue boxes) represent the aggregation results of Softmax, LR, and RF into a single prediction result for each patient's MRI images.

then it combines the outputs of these trees to obtain the final prediction. Moreover, the ensemble learning technique is less prone to overfitting because it reduces the variance, in addition, it is less prone to outliers, and noise and one more important advantage of RF is that it doesn't require input features to be scaled. Consequently, aggregating the results of many classifiers improves the segmentation performance. For the LR algorithm, it is an efficient algorithm in terms of computational resources. Also, LR doesn't require input features to be scaled. Furthermore, it can deal with sparse and dense features since the input features are the extracted feature maps of the last hidden layer of DCNNs before the Softmax function, so due to using Relu as an activation function, we will get a lot of sparse features. In addition, if we zoom inside a tumor region of glioblastoma, we can see that it is possible to classify a small region as a binary classification problem. Thus, by using a set of input features, LR is able to classify the boundary between each pair of classes in MRI images of glioblastoma.

The motivation behind the aggregation of many classifiers' outputs is that using many classifiers in parallel helps to extract a lot of discriminative features from the training data [Hinton et al., 2015]. Moreover, using many classifiers as an ensemble learning method assists to avoid the issue of overfitting [Srivastava et al., 2014] better than individual classifiers. Where the objective of using the ensemble learning method is to make the classifiers as different as possible, therefore, the diversity of the classifiers leads to minimizing the correlations between their errors.

5.3.2.2 Semi automatic-based Support vector machine

Brain tumor segmentation is primarily used for diagnosis, treatment, and follow-up. The developed pipeline is applied to Glioblastoma which are brain tumors and life-threatening diseases. These tumors have four classes: Necrotic and Non-Enhancing tumor, Edema, Enhancing tumor and healthy tissue. To interpret MRI images, a radiologist employs a manual segmentation. Furthermore, it is known that manual segmentation in MRI images is a time-consuming and tedious procedure. In general, there are three methods to obtain a brain tumor segmentation image: manual, semi-automatic and fully automatic. In this section, we propose a second pipeline to solve the issue of misclassified regions (i.e. false positives and false negatives).

In the first approach (See section 5.3.2.1), we proposed an approach based on a combination of different classifiers. In this section, we investigate another approach (second pipeline) to overcome the issue of misclassified regions. The second pipeline is based on using other types of machine learning methods, here we used principal component analysis (PCA) followed by support vector machine (SVM). We first trained SparseMultiOCM architecture from scratch as in chapter 4 (See section 4.3.2.2.2). Second, we extract feature maps from the last convolutional layer before softmax (without normalization). Then, the quantity of extracted features maps is reduced using PCA before classifying them using an SVM method. The goal of reducing the number of features is to reduce computational costs.

In this approach, we followed the same first approach procedures (See section 5.3.2.1). In the first step, we used the proposed SparseMultiOCM architecture in chapter 4 (See section 4.3.2.2.2). For the second step, we extracted the feature maps from the last convolutional layer before Softmax. Then, we store the extracted feature maps in two separate files for the training and testing phases. The final step is training PCA-SVM to reduce the large dimensionality of feature maps for each patient MRI image and to classify each pixel into 4 classes: edema class, non-enhancing core and necrotic core class, enhancing core class, and healthy tissue class.

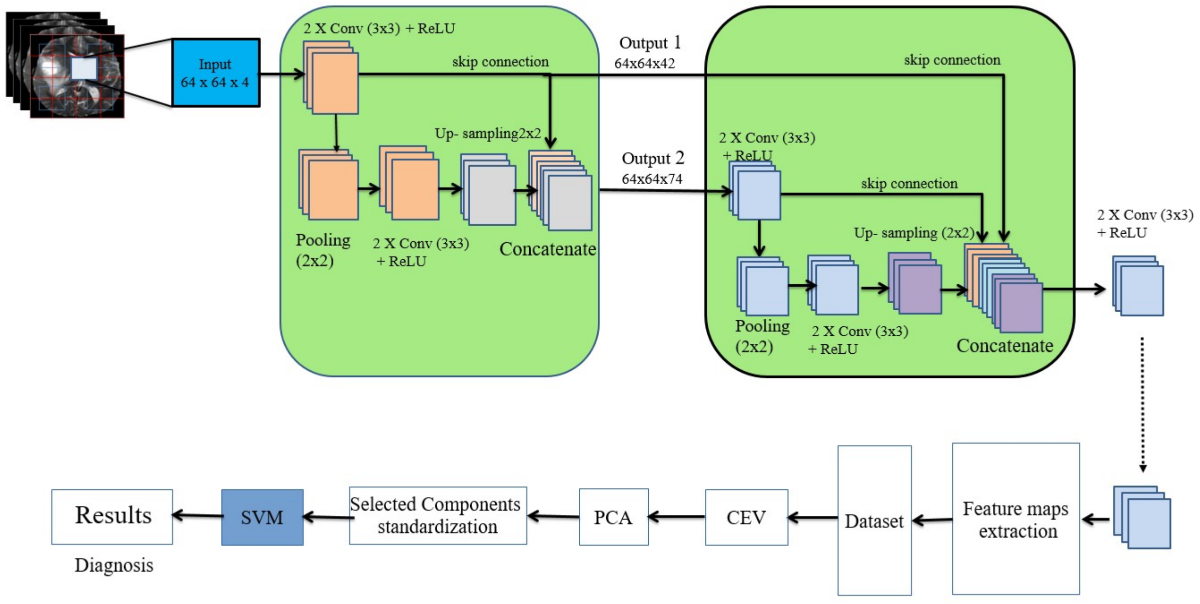
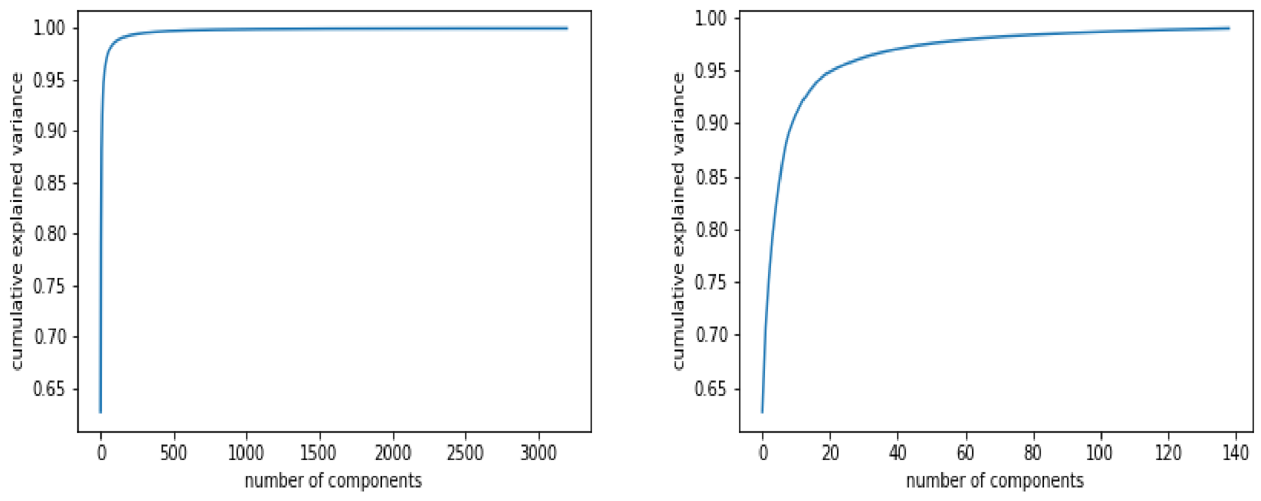


Figure 5.5 – Illustration of the flowchart of PCA-SVM solution for predicting Glioblastoma brain tumors. This flowchart starts by a SparseMultiOCM architecture (two green boxes), then the extracted feature maps from this architecture will be reduced by applying cumulative explained variance (CEV), then the reduced selected principal components are fed into an SVM method to predict the class of each pixel in patient's MRI images.

The flowchart of the second proposed pipeline to detect Glioblastoma brain tumors (see figure 5.5) is composed of 3 parts: first part is using the SparseMultiOCM architecture with its used parameters in chapter 4 (See section 4.3.2.2.2). The second part is extracting feature maps from the last convolutional layer before normalization using Softmax function. The third part is for reducing the number of features, first, we used cumulative explained variance[1] (CEV) technique to obtain the percentage of variance on each principal component. The second step: based on of technique of CEV, we select the first K components in order to maximize the variance of these K principal components. Because, the first principal components cover most variance (so-called explained variance) compared to the last components, i.e, most variance has the most useful information in

1. Cumulative explained variance calculates the variance on each principal components, then it adds the variance of the first principal component to the second one and so on until we reach all principal components. Thus, the whole variance is the sum of variances of all individual principal components.

the extracted features. The technique of CEV helps to reduce the number of redundant features in addition to reducing noise; in practice the last principal components correspond to noise, in addition, noise in the BRATS dataset does not have a high variance allow it to be extracted among the first principal components. After applying CEV technique on the extracted features, we observed that 99% of variances are concentrated on only 139 components out of 3200 components; which means that 3061 of components hold redundant features and noise (See figure 5.6).



(a) all components (all dimensions) (b) 139 components (139 dimensions)

Figure 5.6 – Cumulative explained variance curve of (a) all principal components and of (b) 139 principal components. It can be seen that we only need 139 principal components out of 3200 components to represent 99% of the features. The remaining components represent redundant features and noise.

As it is shown in figure 5.6, the figure 5.6.(a) represents the dimension of all features, in other words, the cumulative variance of all principal components, and figure 5.6.(b) shows only 139 dimensions (components) of all components that have 3200 dimensions. Thus, from these two figures we can conclude that 139 components represent 99% of variances. After computing CEV, we applied principal components analysis (PCA) to reduce the dimensionality of features from 3200 to 139 principal components; for each patch, we use only 139 principal components. Reducing the dimensionality of features using CEV and PCA helps to get new features more representative without getting redundant features and noise which need a lot of preprocessing to remove them.

The last step in this flowchart (See figure 5.5) consists of applying an SVM method that is one of the most powerful methods for classification problems as it can deal with many forms of data and classification problems (binary, multi, linear, non-linear). To develop an SVM method: first, we need a lot of data, here we use the computed principal components through applying the method of PCA. Second, we need to specify the type of problem, in our case, it is a multi-classification issue (4 classes). Usually, a multi-classification problem is a non linearly separable issue, so to verify the type of problem (linear, non-linear), we draw the first three components in 3D space (See figure 5.7).

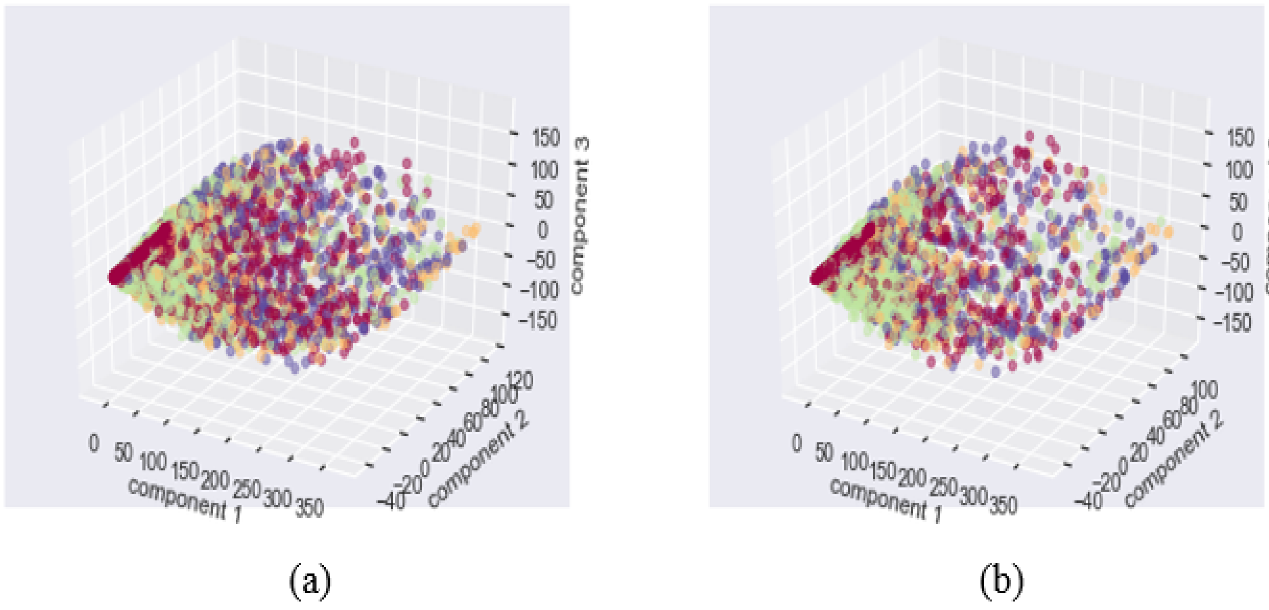


Figure 5.7 – Scatter plot for the first three principal components in 3D space in (**a**) training and (**b**) testing datasets. It can be seen that in the training and testing dataset, the issue of brain tumor segmentation is a nonlinear multi-classification issue, i.e., we can not draw a hyperplane in 3D space to separate the classes.

As it is shown in figure 5.7.(a) and figure 5.7.(b), we drew only 3 principal components from 139 principal components, and as expected the issue of multi-classification in our case is non linearly separable; we can not draw a line (or a hyperplane in 3D space) to separate two classes. From figure 5.7, we conclude that the issue that we are dealing with, is multi-classification and non-linearly separable issue. Thus, this conclusion helps us to determine the hyperparameters of SVM method especially the *kernel*. In general, SVM method has three types of kernels: *linear*, *polynomial* and *radial basis function*

(RBF). First we can eliminate linear kernel because it is used for linearly separable issues. Second, *polynomial kernel* is computionally expensive and needs a lot of memory even with 139 principal components. Thus, the best choice for the SVM method is using *RBF kernel*. Moreover, for the other hyperparameters that are used to optimize hinge loss function [Rosasco et al., 2004] such as *gamma* γ and *C slack-penalty*. Because SVM is sensitive to outliers and feature scaling and as we mentioned earlier the problem of multi-classification of brain tumor segmentation is a non-linearly separable issue which means, in this case, *C slack-penality* should be greater than zero to transform the issue into soft-margin classification; some instances could be on the street of the decision boundary (margin violations) between classes. Furthermore, the value of *gamma* γ controls the influence of each training instance in its search space, in this case, and because of outliers, *gamma* γ should be a little bit high. As there is no straightforward method to determine *gamma* γ and *C slack-penality*, we used the technique of cross-validation with 5 folds in addition to a hybrid of *Random* and *Grid* search (See figure 5.8) on a small dataset of 30 brain images from the training set. And the result is as the following: *gamma* $\gamma = 0.001$ and *C slack-penality* $= 5$.

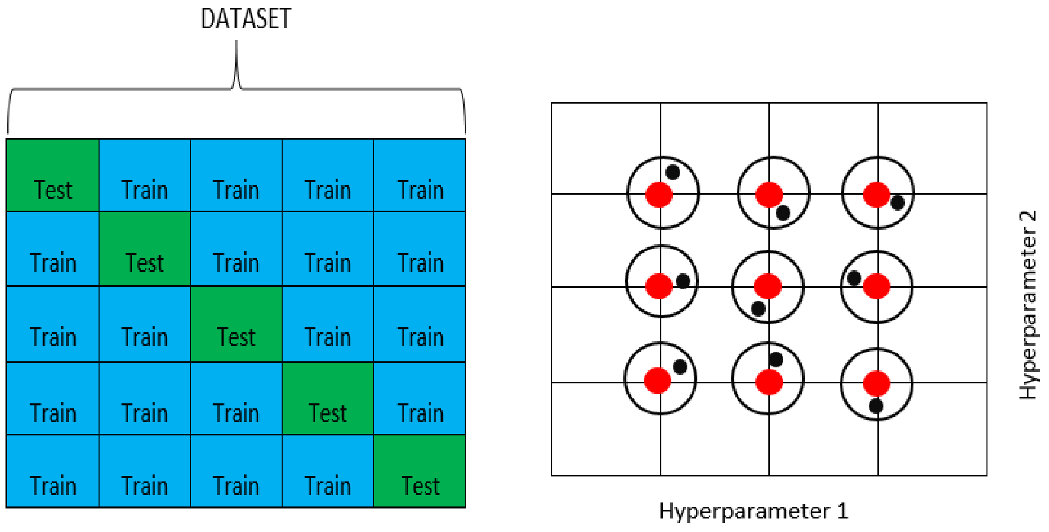

DATASET
Test Train Train Train Train
Train Test Train Train Train
Train Train Test Train Train
Train Train Train Test Train
Train Train Train Train Test
Hyperparameter 1
Hyperparameter 2

Figure 5.8 – **Left**: At each fold, a separate partition of the dataset is used for testing while the remaining partitions are used for training. Then, the results are averaged over all folds. **Right**: An illustration of the hybrid Random-Grid search technique, where we start by random search (red points) until we obtain potential combinations of hyperparameters, then we use a grid search (black points) to more precise and refine the combinations.

Moreover, SVM method is sensitive to feature scaling, and since the *kernel RBF* is used to compute the SVM's loss function. In this case, the *RBF kernel* assumes that all features are centered around zero and the magnitude of the variance of all features is the same. So, to make the features have the same scaling, we standardized the computed principal components of PCA to be centered with 0 mean and a standard deviation of 1 using the following equation (See equation 5.9):

$$Standardization(X) = \frac{X \ - \ U}{\sigma} \tag{5.9}$$

Where X is the sample instance, U is the Mean of the training set and σ is the Standard Deviation. The reduced features are scaled into a limited range using Standardization (X) technique; the variance of this range has a fixed value equals to 1 with Means equals to 0.

After presenting different approaches in sections 5.2 and 5.3. In the next section, we present the effectiveness of our proposed approaches with a series of experiences.

5.3.3 Experiments and Results

This section presents the results of our experiments and some discussions. Also, our brain tumor segmentation method is evaluated using the BRATS dataset which has real patient data with Glioblastoma brain tumors (both high- and low-grade). Finally, we explain the libraries and frameworks that are used in our work. Then, we evaluate our proposed architectures with a set of metrics that are used in state-of-the-art such as Dice score, Sensitivity, Specificity, and Hausdorff Distance.

5.3.3.1 Data preprocessing and implementation

The applied preprocessing technique in this section is the same as in chapter 4 (See section 4.3.5.1) with 3 steps: the first step is removing 1% highest and lowest intensities, the second step is subtracting the mean and dividing by the standard deviation of non-zero values in all channels, and the final step is isolating the background and healthy tissue from the tumor regions. The optimization of SparseMultiOCM architecture is achieved using stochastic gradient descent (SGD) with mini-batch size equals to 8 and learning rate computed as follows:

$$LER_i = 10^{-3} \times 0.99^{LER_{i-1}} \tag{5.10}$$

Where the initial learning rate (LER) was $LER_0 = 0.001$, LER_i ($i \in \mathbb{N}^+$) is the new learning rate, LER_{i-1} is the learning rate of the last epoch, 0.99 is a decreasing factor. The SparseMultiOCM architecture was implemented on Keras[2] which is a high-level open source Deep Learning library, and Theano [Al-Rfou et al., 2016] as a Back-end, where Theano exploits GPUs to optimize Deep Learning architectures. The proposed machine learning methods: LR, RF, PCA, and SVM were implemented using Scikit-learn [Pedregosa et al., 2011] package. In the training phase, we used the same amount of data for each class; we used 10k learning points for each class in the phase of training.

All our results are obtained using Python environment on windows 64-bit, Intel Xeon processor 2.10 GHz with 24 GB RAM, and the training is done on Nvidia Quadro GPU K1200 with 4 GB memory.

One-vs-One versus One-vs-Rest: one-vs-one approach splits the multi-class classification problem into multiple binary classification problems, in other words, for each pair of classes, we will have a binary classification model. In this case, the number of models, and in turn, datasets, is: N (N-1)/2, where N is the number of classes. Moreover, the problem of Glioblastoma brain tumors segmentation has 4 different classes, this gives 6 binary classification models. One-vs-rest approach splits the multi-class classification problem into one binary classification problem per class. In this case, the number of models, and in turn, datasets, is N, where N is the number of classes. Furthermore, for the issue of Glioblastoma brain tumors segmentation, we need 4 binary classification models. We have chosen the one-vs-one approach for two main reasons: first, in the SVM algorithm, the training time scales are proportional to the number of dataset samples, in this case, the performance of SVM's kernel will be influenced by a large number of samples. Consequently, using one-vs-one to train a SVM model would help to reduce the runtime and memory consumption by using only a subset of the training dataset. Second, using on-vs-rest to solve the problem of brain tumor segmentation as a binary classification problem per class will create another issue that is unbalanced data, in this

2. https://keras.io/

case, the model would give more importance to one class more than the other one, and that is due to training a model on a dataset of one class against 3 classes regrouped in one class. So, for the issue of segmentation of glioblastoma brain tumors and due to limited resources, the best solution is the One-vs-One approach.

5.3.3.2 Results

In this section, we evaluate our proposed brain tumor segmentation pipelines on a public BRATS 2019 dataset using 4 evaluation metrics: Dice score, Sensitivity, Specificity, Hausdorff distance. Our proposed pipelines SparseMultiOCM + LR-RF and SparseMultiOCM + PCA-SVM help to extract relevant features from each patient MRI image, Where these features are used after that in the second stage to predict the class of each pixel among the four classes: edema, non-enhancing core and necrotic core, enhancing core, and healthy tissue. Moreover, The pipelines outputs are the valid pixel with high rate from each machine learning method (e.g. LR and RF). This technique of combining the results of several methods for the same subject increases the prediction of tumor region in addition to decreasing the error and the misclassified regions.

In this section, we focus and give a more attention to two important evaluation metrics: Sensitivity and Specificity. These two metrics are mostly used to measure the percentage of misclassified regions, i.e. the percentage of false positives and false negatives regions. Sensitivity attempts to measure the false negative regions, and specificity does the same for false positive regions.

Table 5.4 – Evaluation results of fusion solution-based predictions aggregation pipeline on BRATS 2019 validation set. WT, TC, ET denote whole tumor, tumor core, enhancing tumor, respectively. Std denotes standard deviation.

	Dice score			Sensitivity			Specificity			Hausdorff		
	WT	TC	ET	WT	TC	ET	WT	TC	ET	WT	TC	ET
Mean	0.84	0.70	0.61	0.84	0.71	0.69	0.99	0.99	1.0	22.64	20.40	13.76
Std	0.13	0.23	0.33	0.16	0.26	0.28	0.01	0.01	0.01	26.10	24.63	24.90
Median	0.88	0.77	0.77	0.89	0.80	0.81	0.99	1.0	1.0	8.37	10.68	3.16
25 quantile	0.84	0.58	0.43	0.82	0.61	0.60	0.99	0.99	1.0	3.61	6.40	2.0
75 quantile	0.91	0.87	0.86	0.94	0.92	0.89	1.0	1.0	1.0	37.29	22.20	11.0

Table 5.4 and Table 5.5 show the segmentation results of our proposed SparseMul-

Table 5.5 – Evaluation results of fusion solution-based predictions aggregation pipeline on BRATS 2019 testing set. WT, TC, ET denote whole tumor, tumor core, enhancing tumor, respectively. Std denotes standard deviation.

	Dice score			Hausdorff		
	WT	TC	ET	WT	TC	ET
Mean	0.84709	0.75889	0.73703	12.99701	15.4957	6.03933
Std	0.15312	0.25993	0.23841	23.97851	25.62727	16.45033
Median	0.89588	0.85913	0.8148	4.30077	8.09315	2.23607
25 quantile	0.83621	0.74323	0.70943	3	4	1.41421
75 quantile	0.92368	0.9143	0.87902	7.95064	14.65374	3.74166

tiOCM with LR-RF pipeline. The prediction of tumor regions is performed using 2D patches with size equals to 64 x 64 x 4 (4 corresponds to using different modalities such as T1, post-contrast T1, T2, and FLAIR). Then, we extract the feature maps of the last hidden layer before Softmax function, then we feed these features into different machine learning algorithms such as RF and LR. The last step, we combine the results of SparseMultiOCM, RF, and LR into one 3D image using voting technique; where the most predicted label among the predictions (e.g., 1, 1 and 0) of these classifiers becomes the result (in this case the label becomes 1) of the pixel's label. As it is shown in Table 5.4 and Table 5.5 on the validation and testing datasets, we are able to achieve segmentation results comparable to the top-performing methods in state-of-the-art such as the work of [Zhao et al., 2018]. Moreover, the achieved median score of the proposed pipelined is high: 0.88, 0.89 for whole tumor in terms of dice score for the validation and the testing sets, respectively). The high values provided are due to obtaining segmentation performance for most patient MRI images. The architecture of SparseMultiOCM with LR-RF pipeline is also evaluated on Sensitivity metric: 0.84 for the whole tumor. On Specificity metric: 0.99 for the whole tumor. On the Hausdorff distance metric, as it can be observed that our pipeline achieve a good segmentation performance: 11 mm to 37 mm on validation set and 3 mm to 7 mm (for Hausdordd distance O is the best result) on the testing set.

Table 5.6 – Evaluation results of semi-automatic-based support vector machine pipeline on some subjects of BRATS 2019 training set. WT, TC, ET denote whole tumor, tumor core, enhancing tumor, respectively.

Patient ID	Dice score			Sensitivity			Specificity			Hausdorff		
	WT	TC	ET	WT	TC	ET	WT	TC	ET	WT	TC	ET
”2013_10_1”	1	0.58996	0.24228	1	0.74294	0.19889	1	0.96818	0.99043	0	29.03446	26.41969
”2013_11_1”	1	0.60334	0.14959	1	0.80945	0.25156	1	0.96026	0.97204	0	13	13
”2013_12_1”	1	0.46689	0.18173	1	0.79719	0.28464	1	0.89892	0.93791	0	27.09243	30.09983

Table 5.6 shows the segmentation results of our proposed semi-automatic method that is based on SparseMultiOCM with PCA-SVM. In this table, we show the segmentation performance with different metrics (Dice score, Sensitivity, Specificity, Hausdorff distance). As it is shown, our method provides good segmentation results especially on tumor core and enhancing tumor regions and that is due to the representative extracted features from the layer 16 of the SparseMultiOCM architecture and the selected principal components of PCA. The part of PCA-SVM is semi-automatic; after extracting the feature maps from the architecture, the user needs to interact and to select the whole tumor region. Then, PCA-SVM starts to predict the GBM brain subregions inside the whole tumor: Healthy tissue, Necrotic and Non-Enhancing tumor, Peritumoral Edema, and Enhancing core. As it is shown in Table 5.6, the proposed pipeline achieves good results with a performance comparable to state-of-the-art methods. Moreover, the achieved mean score is high in terms of dice score: 1, 0.59, 0.24 for the whole tumor, tumor core, and enhancing tumor. Our SparseMultiOCM with PCA-SVM pipeline is evaluated on Sensitivity metric: 1, 0.74, 0.19 for the whole tumor, tumor core, and enhancing tumor. On Specificity metric: 1, 0.97, 0.99 for the whole tumor, tumor core, and enhancing tumor. On the Hausdorff distance metric, as it can be observed that our pipeline achieves a good segmentation performance: 0 mm (0 distance value is the best), 29.03 mm, 26.42 for the whole tumor, tumor core, and enhancing tumor. These numbers translate the large improvement of state-of-the-art performance in terms of the evaluation metrics: Dice score, Sensitivity, Specificity, and Hausdorff distance.

5.3.4 Discussion and Conclusion

In this section, we developed two MRI image segmentation pipelines for Glioblastoma brain tumors and dedicated to solving the issue of misclassified regions (i.e. false positive and false negative). These pipelines are composed of a Deep CNNs architecture and learned features maps. To overcome the issues of false positive and false negative regions, we extracted the feature maps from the last convolutional layer of the Deep CNNs architecture before applying Softmax normalization to train another two machine learning algorithms: random forest, logistic regression, and PCA-SVM. These algorithms showed a high impact on the segmentation performance.

Our experimental results show that the proposed brain tumor segmentation pipelines im-

proved the evaluation metrics: Dice score, Sensitivity, Specificity, Hausdorff. Moreover, these pipelines are based on two approaches: the first pipeline SparseMultiOCM with LR-RF is a fully automatic brain tumor segmentation, while the second pipeline is a semi-automatic method based on SparseMultiOCM with PCA-SVM and learned feature maps.
The proposed SparseMultiOCM architectue with LR-RF provides a high segmentation performance especially in terms of Specificity (99%, 99%, 100%) and Sensitivity (84%, 71%, 69%) for the whole tumor, tumor core, and enhancing tumor, respectively.

Chapter 6

Conclusion and perspectives

Contents

6.1 Summary of contributions

In this thesis, we proposed methods for the segmentation of brain tumors using MRI images, where these methods are applied to segment Glioblastoma brain tumors with both high- and low-grade. The proposed methods are based on Deep Convolutional Neural Networks and machine learning. In addition, through this thesis, we solved several limitations that face current learning-based methods with medical images.

As a summary, Glioblastoma Multiform tumors are the grade IV of Glioma tumors according to WHO organization and these tumors represent 46,6% of primary malignant brain tumors. Glioblastoma tumors touch older adults more than children, and the incidence of Glioblastoma tumors rises with age. Furthermore, the grade and the delineation of the tumor region and its sub-regions, in addition to, obtaining clinical relevant information such as target volume and tumor volume are very important on the selection of therapies: radiotherapy or chemotherapy.

In Chapter 2, we presented a survey of classical and modern state-of-the-art approaches in the field of MRI image segmentation, in particular, Glioblastoma brain tumors segmentation. Moreover, we discussed some statistics about Glioblastoma tumors with low- and high grade and different challenges and limitations of MRI images such as noise, intensity inhomogeneity, patients motions, ...etc. From 2014 to the present, the most effective methods for the segmentation and classification of Glioblastoma brain tumors are based on Deep learning, in particular, different variants of Deep Convolutional Neural Networks. To measure the segmentation performance and the computational benefits of our proposed methods in an objective manner with radiologists' performance and state-of-the-art methods, we presented 6 evaluation metrics: Dice scores, Sensitivity, Specificity, Hausdorff distance, Inference time, and Dice density.

In Chapter 3, we presented Deep learning, in particular Deep Convolutional Neural Networks (Deep CNNs). Deep CNNs have evolved for many years since its first introduction in the works of [Fukushima, 1980; LeCun et al., 1989]. From 2012 until the present, many variants of Deep CNNs architectures appeared with newly added techniques such as ReLU activation function, Dropout regularization technique, Upsampling, weights initialization, and GPU implementation to take advantage of parallel processing. The research on lesion segmentation, in particular, Glioblastoma brain tumors segmentation and classification with Deep Learning started in 2014. Moreover, the success of Deep CNNs architectures is due to the availability of large public datasets, learning methods, and frameworks that benefit from massive parallel architectures such as GPUs to parallelize Deep CNNs operations.

In Chapter 4, we focused on the segmentation of patients' MRI images with Glioblastoma high- and low grade using two approaches of Deep CNNs. The main contributions of the first approach address 3 issues: (i) developing an efficient Deep CNNs architecture and optimizing hyperparameters, (ii) computational costs of training large and Deep CNNs architectures, (iii) vanishing grading and speeding up the training phase. To overcome these issues, we proposed two algorithms: Incremental XCNet algorithm and ELOBA_λ algorithm. Incremental XCNet is based on the idea of developing a generic algorithm that can be used to develop different Deep CNNs architecture. This algorithm has 3 steps:

firstly, adding a limited block of Convolution, Pooling, and fully connected layers (FC), secondly evaluating the added blocks, thirdly removing FC, and freezing all Convolution layers. Then, we repeat these steps until the stopping criteria are achieved. ELOBA_λ algorithm is developed specifically to unify the training methods and to optimize the hyperparameters of Deep CNN architectures such as epochs, batch sizes, learning rate, optimizers, number of layers. ELOBA_λ algorithm is a graph with 4 different search spaces: root node is the Deep CNNs architecture, second level is for the optimizers, third level is for the learning rate, and leaf node is for the batch size. ELOBA_λ algorithm starts the training of Deep CNNs architecture from the first block at the root node, then it explores the graph using depth-first search, and each time it succeeds to get a good combination of hyperparameters, it stores them for the next block. If the combination doesn't work for some blocks, ELOBA_λ algorithm continues the process of parsing the graph using depth-first search until it explores all the nodes of the graph. Our proposed approach is evaluated on BRATS dataset, where it provided good segmentation results compared to the state-of-the-art methods, with mean Dice scores: 89 % (Whole tumor), 76 % (Tumor core), and 81 % (Enhancing tumor), in addition, in terms on Haussdorff distance: 14.62 (Whole tumor), 17.40 (Tumor core), 12.94 (Enhancing tumor). Furthermore, the first approach obtained good results on mean Inference time: 19 on GPU.
For the second part of this chapter, the main contributions of the second approach relate to 2 issues: (i) modeling context and spatial relationship, (ii) and developing an efficient Deep CNNs architecture in terms of segmentation performance and computational costs (GPU and CPU). To overcome these issues, we proposed a Deep CNNs architecture inspired by the *visual cortex*, in particular, *Occipito-Temporal Pathway* (OTP). OTP uses different *receptive fields* in successive regions to extract more relevant features and to figure out the important regions or objects in the input images, this technique in the human visual system called *selective attention.* The second proposed approach in this chapter is based on two elements: firstly, *selective attention* to maximize the extraction of relevant features from MRI images. Secondly, it based on some techniques learned from the first approach (Incremental XCNet) such as the use of two or three convolutional layers with 3×3 filter in successive layers provides good results and less prone to the issue of overfitting. To address the issue of modeling context and spatial relationship, we proposed Overlapping Patches instead of conventional Adjacent Patches. The limit

of Deep CNNs architectures with Adjacent Patches is that Deep CNNs do not take into account the fact that these Adjacent Patches together make up the entire image. Thus, to overcome this issue, we extracted Overlapping Patches such as the local and the larger context by using small Patches and without adding more additional Networks for each input image resolution. The second approach obtained good segmentation performance results in terms of Specificity: 99.5 % (Whole tumor), 99.8 % (Tumor core), 99.8 % (Enhancing tumor), and in terms of Hausdorff distance: 8.521 (Whole tumor), 13.324 (Tumor core), 4.406 (Enhancing tumor). In addition, the second approach is the fastest approach among the cited state-of-the-art methods in terms of average inference time: 12 seconds on GPU and 20 seconds on CPU. Furthermore, we measured the effectivness of the second approach per parameter: 6.73 Dice density.

In Chapter 5, we focused on the challenges and limitations of Deep learning with two issues: unbalanced data and misclassified regions. The issue of unbalanced data makes Deep CNNs bias toward the more frequent label (i.e. class with the majority of data). Training a Deep CNNs architecture with highly unbalanced data such as BRATS dataset that has 98% healthy tissue (i.e. non-tumorous), will provide predictions with low sensitivity. Moreover, the most important part of tumor segmentation applications is how to make Deep CNNs more sensitive toward the tumoral regions. To overcome this issue, we proposed the Online Class-Weighing loss function, first we added weighting factors W $\in$ [0,1] to the conventional Cross-entropy loss function to balance between the data examples for each class, hence, we can mitigate the bias issue. For these reasons, we proposed Online Class-Weighting algorithm as an adaptive algorithm for each batch of images. After S epochs of training, the algorithm computes the Dice scores of each class. And based on the latter (Dice scores), the algorithm computes the dominant class and the dominated class among the predefined classes. For the first class, the algorithm reduces the weighting factors corresponding to this class, and for the second class, the algorithm increases the weighting factors corresponding to this class also, and so on until the stopping criteria or the end of epochs are achieved. Moreover, Online Class-Weighting loss function is evaluated against focal loss and cross-entropy loss and it obtained very promising results in terms of Dice scores: 88% (Whole tumor), 83% (Core tumor), 75% (Enhancing tumor).

For the second part of this chapter, the main contributions address the issues of misclassified regions. The reason behind this issue is due to the fact that Deep CNNs architectures predict tumor regions, in addition, these architectures produce misclassified regions: false positives regions and false negatives regions. To overcome this issue, we proposed two efficient pipelines trained on two-stages. The first pipeline is based on a combination of 3 machine learning algorithms: Random Forest, Logistic regression, and the results of SparseMultiOCM architecture. In the second pipeline, we reduced the dimensionality of the extracted features using cumulative explained variance and principal component analysis, after that, we fed the reduced 139 principal components to a support vector machine classifier. Experimental testing demonstrated that our methodology of two-stages of training yields good results than a single-stage training approach. Our best approach SparseMultiOCM architecture with LR-RF achieved a good performance in terms of Dice scores: 84.71 % (Whole tumor), 75.89 % (Tumor core), and 73.70 %(Enhancing tumor), and in terms of Hausdorff distance: 12.99 (Whole tumor), 15.50 %(Tumor core), and 6.03 (Enhancing tumor) and the best segmentation results in terms of Specificity (99%, 99%, 100%) and Sensitivity (84%, 71%, 69%) for the whole tumor, tumor core, and enhancing tumor, respectively.

6.2 Perspectives

In this thesis, we addressed several issues: Glioblastoma brain tumors segmentation, unbalanced data, modeling context and spatial relationship, misclassified regions (false positives and false negatives), in addition to, how to develop an efficient Deep CNNs architecture, vanishing gradient issue, and automatic hyper-parameters optimization. Moreover, we used in this thesis the data of the BRATS challenge which is a limited MRI images dataset. We intend in the future work to use and combine the BRATS dataset with other public and clinical source of MRI data for improving the segmentation results of the proposed methods. To combine two or more MRI images datasets, we need first to solve the issue of image registration that attempts to match several MRI images acquired using different scanners to one template. In last years, appeared many methods, where they solved this issue using Deep neural networks, such as the one proposed in [Dalca et al., 2019], are of particular interest in the future works.

Moreover, another important aspect is fusing the decisions of many classifiers, such as the case of the voting function that is developed in this thesis to combine the results of Deep Neural networks architectures and machine learning methods. Fusing many decisions of many classifiers needs also to estimate the confidence and the uncertainty of the segmentation. The proposed voting function is a simple function that combines the segmentation results pixel-per-pixel and based on the majority of votes of many classifiers. The technique of voting does not provide accurate segmentation results and without the estimation of segmentation uncertainty, it could elect the wrong majority of votes. For these reasons, several approaches have been proposed in the literature to solve this issue using Kalman Filter, such as the one suggested in [Glodek et al., 2013], are of particular interest.

Another important direction of the future work would be to propose an extension to the proposed method of Online class-weighting for the issue of unbalanced data. The limits of the Online class-weighting loss function is that it adjusts only two classes at one time until the end of the training phase, where in our case, we have 4 classes. In future work, we intend to extend the Online class-weighting function by proposing proportional updating of all classes according to their contributions at one time using Robin Hood index (See figure 6.1). In addition to incorporating the constraints of the weighting factors with the loss function using Lagrange function $L(\alpha)$ (See equation 6.1):

$$\begin{aligned} L(\alpha) &= -\frac{1}{n}(\sum_{j}^{n} W_j \times q_j \times \log(p_j)) - \sum_{j=1}^{n}\alpha_j(W_j - 1) \quad s.t.\ \ W_j \in [0,1], j = 1,2,...,n \\ &= -\frac{1}{n}(\sum_{j}^{n} W_j \times q_j \times \log(p_j)) - \sum_{j=1}^{n}\alpha_j W_j + \sum_{j=1}^{n}\alpha_j \quad s.t.\ \ W_j \in [0,1], j = 1,2,...,n \end{aligned} \tag{6.1}$$

Where $\alpha = (\alpha_1, ..., \alpha_n) \geq 0$ is the Lagrange multiplier corresponding to each class.

We intend to model the problem of unbalanced data as a distribution of 'wealth' inequalities. We calculate the 'total wealth' that should be transferred from the 'rich' class to the 'poor' class to achieve a state of perfect equality in the distribution of wealth.

In our case, the rich class is the dominant class and the poor class is the dominated class. The 'total wealth transfer' and the redistribution of weights are computed as follows:

- Computing the average of the total 'individuals wealth', in our case, we compute the Dice score of each class, then we take the average.
- Selecting the individuals (i.e. classes) that are above and below the average.
- Computing Robin Hood (RH) index as follows:

$$RH = \frac{\sum_{i}^{n} |x_i - \bar{x}|}{2\sum_{i}^{n} x_i} \tag{6.2}$$

Where x_i represents the Dice score of the i^{th} class, $\bar{x}$ is the average of the n classes. RH represents a percentage (%) of wealth (in our case weighting factors) that should be transferred from classes above (i.e. dominant classes) the average to those below (i.e. dominated classes) the average. After many iterations, RH achieves 0, at this point we achieve the state of perfect equality in the distribution of weights among the predefined classes. Robin Hood index is also defined as the difference between the Lorenz curve and the line of perfect equality (See figure 6.1).

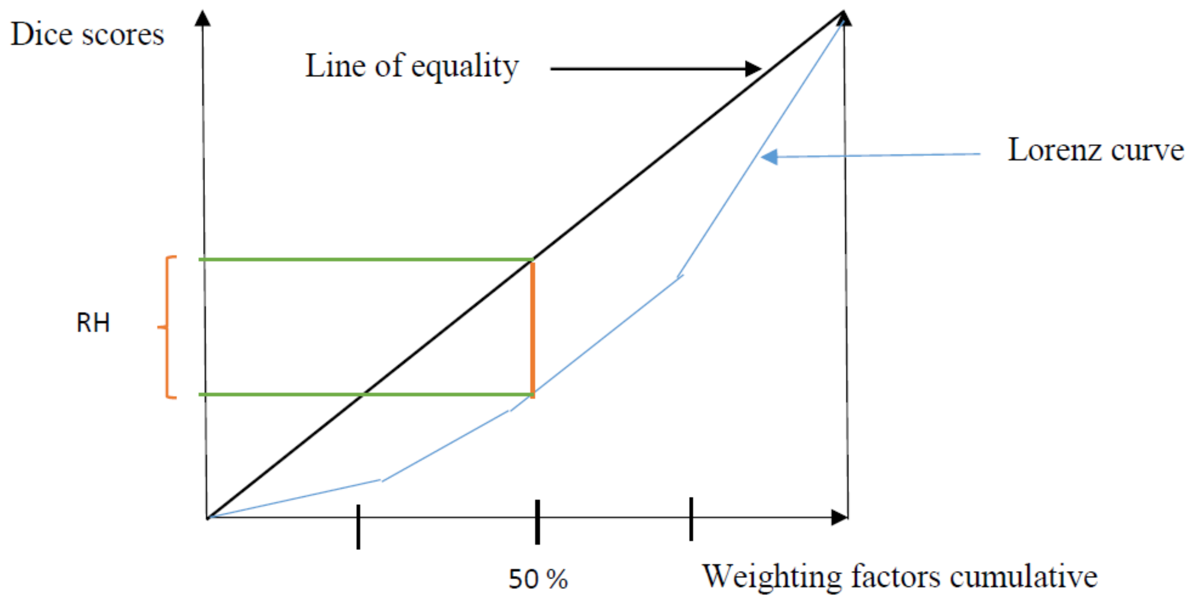


Figure 6.1 – Illustration of weighting factors cumulative distribution using Robin Hood index.

The last perspective of this thesis involves the issue of modeling context and spatial relationships. To develop a Deep CNNs architecture for the issue of MRI image segmentation, we have either a pixel-wise approach or a patch-wise approach. Moreover, these two approaches are based on dividing the MRI image into many patches. Then, the Deep CNNs architecture predicts the pixel's class which is the center of the patch (See left figure 6.2). As it is known in BRATS dataset, MRI image patches have much healthy regions than tumor regions, so even dividing a MRI image into patches, we could not model patches with equal pixels distribution over all classes. For these reasons, we intend in the future to model the 4 types of intra-tumoral structures: edema, non-enhancing core, and necrotic core, enhancing core, and healthy tissue, and the relation among them using Graph Neural Networks (See right figure 6.2). The image pixels become the vertices and the distances among the N-neighbor pixels become the edges.

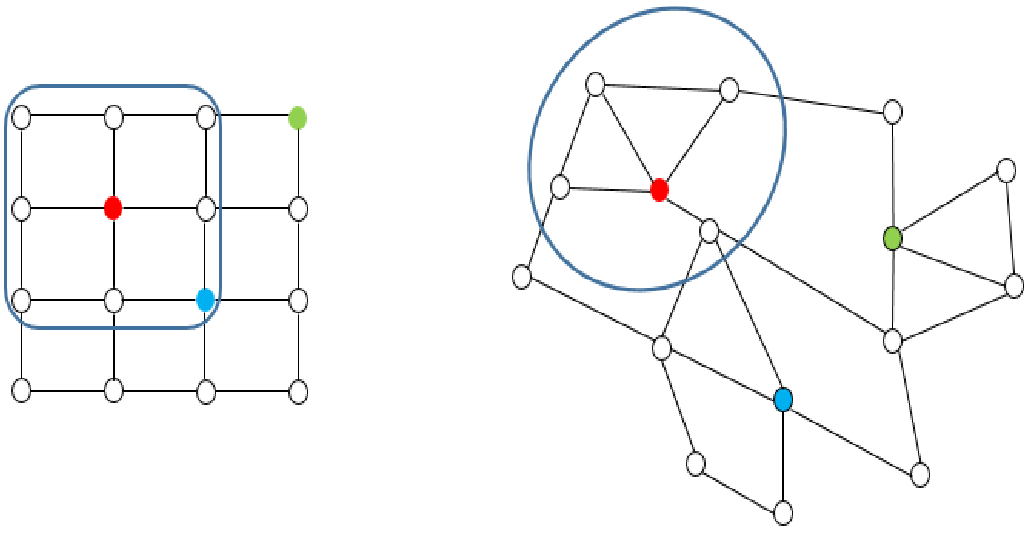


Figure 6.2 – Illustration of (**left**) 2D Convolutional Neural Networks and (**right**) Graph Neural Networks.

6.3 List of publications

Here, I list the publications that have been used to express my research contributions I have made in this thesis:

Journal/Conference Articles:

1. **Ben naceur, Mostefa**, Akil, M., Saouli, R., & Kachouri, R. (2020). Fully automatic brain tumor segmentation with deep learning-based selective attention using overlapping patches and multi-class weighted cross-entropy. *Medical Image Analysis*, 101692.
2. **Ben naceur, Mostefa**, Akil, M., Saouli, R., & Kachouri, R. (2020). Deep convolutional neural networks for brain tumor segmentation: boosting performance using deep transfer learning: preliminary results. *In MICCAI-BRATS 2019 Workshop: Multimodal Brain Tumor Segmentation Challenge.*
3. **Ben naceur, Mostefa**, Kachouri, R., Akil, M., & Saouli, R. (2019). A new online class-weighting approach with deep neural networks for image segmentation of highly unbalanced glioblastoma tumors. *In International Work-Conference on Artificial Neural Networks (pp. 555-567). Springer, Cham.*
4. **Ben naceur, Mostefa**, Saouli, R., Akil, M., & Kachouri, R. (2018). Fully automatic brain tumor segmentation using end-to-end incremental deep neural networks in mri images. *Computer methods and programs in biomedicine, 166, 39–49.*

Posters:

1. **Ben naceur, Mostefa**, Akil, M., Saouli, R., & Kachouri, R. (2019). Brain tumor segmentation with deep convolutional neural networks. *Ecole de Printemps, E-santé ESIEE-Paris, France.*
2. **Ben naceur, Mostefa**, Saouli, R., Akil, M., & Kachouri, R. (2018). Apprentissage en profondeur d'un réseau de neurones pour la segmentation et la classification en imagerie médicale. *Days of Theoretical and Applied Computer Studies, Computer Sciences department, University of Biskra, Algeria.*
3. **Ben naceur, Mostefa**, Saouli, R., Akil, M., & Kachouri, R. (2017). Deep neural networks for brain tumor segmentation. *Days of Theoretical and Applied Computer Studies, Computer Sciences department, University of Biskra, Algeria.*

Bibliography

Abd-Ellah, M. K., Awad, A. I., Khalaf, A. A., and Hamed, H. F. (2016). Design and implementation of a computer-aided diagnosis system for brain tumor classification. In *2016 28th International Conference on Microelectronics (ICM)*, pages 73–76. IEEE.

Akram, M. U. and Usman, A. (2011). Computer aided system for brain tumor detection and segmentation. In *Computer Networks and Information Technology (ICCNIT), 2011 International Conference on*, pages 299–302. IEEE.

Al-Rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., Ballas, N., Bastien, F., Bayer, J., Belikov, A., Belopolsky, A., et al. (2016). Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*.

Alegro, M. d. C., Amaro Junior, E., and Lopes, R. d. D. (2012). Computerized brain tumor segmentation in magnetic resonance imaging. *Einstein (São Paulo)*, 10(2):158–163.

Axel, D., Havaei, M., Warde-Farley, D., Biard, A., Tran, L., Jodoin, P.-M., Courville, A., Larochelle, H., Pal, C., , and Yoshua, B. (2014). Brain tumor segmentation with deep neural networks. *Proceedings MICCAI-BRATS*, page 01–05.

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2015). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*.

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495.

Bakas, S., Akbari, H., Sotiras, A., Bilello, M., Rozycki, M., Kirby, J. S., Freymann, J. B., Farahani, K., and Davatzikos, C. (2017). Advancing the cancer genome atlas glioma mri collections with expert segmentation labels and radiomic features. *Scientific data*, 4:170117.

Bangiyev, L., Rossi, M. E., Young, R., Shepherd, T., Knopp, E., Friedman, K., Boada, F., and Fatterpekar, G. M. (2014). Adult brain tumor imaging: state of the art. In *Seminars in roentgenology*, volume 49, pages 39–52.

Ben naceur, M., Saouli, R., Akil, M., and Kachouri, R. (2018). Fully automatic brain tumor segmentation using end-to-end incremental deep neural networks in mri images. *Computer Methods and Programs in Biomedicine*, 166:39–49.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160.

Bhandarkar, S. M. and Nammalwar, P. (2001a). Segmentation of multispectral mr images using a hierarchical self-organizing map. In *Computer-Based Medical Systems, 2001. CBMS 2001. Proceedings. 14th IEEE Symposium on*, pages 294–299. IEEE.

Bhandarkar, S. M. and Nammalwar, P. (2001b). Segmentation of multispectral mr images using a hierarchical self-organizing map. In *Proceedings 14th IEEE Symposium on Computer-Based Medical Systems. CBMS 2001*, pages 294–299. IEEE.

Bianco, S., Cadene, R., Celona, L., and Napoletano, P. (2018). Benchmark analysis of representative deep neural network architectures. *IEEE Access*, 6:64270–64277.

Brosch, T., Tang, L. Y., Yoo, Y., Li, D. K., Traboulsee, A., and Tam, R. (2016). Deep 3d convolutional encoder networks with shortcuts for multiscale feature integration applied to multiple sclerosis lesion segmentation. *IEEE transactions on medical imaging*, 35(5):1229–1239.

Canziani, A., Paszke, A., and Culurciello, E. (2016). An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678.*

Caselles, V., Catté, F., Coll, T., and Dibos, F. (1993). A geometric model for active contours in image processing. *Numerische mathematik*, 66(1):1–31.

Cates, J. E., Lefohn, A. E., and Whitaker, R. T. (2004). Gist: an interactive, gpu-based level set segmentation tool for 3d medical images. *Medical image analysis*, 8(3):217–231.

Cates, J. E., Whitaker, R. T., and Jones, G. M. (2005). Case study: an evaluation of user-assisted hierarchical watershed segmentation. *Medical Image Analysis*, 9(6):566–578.

Chang, P. D. et al. (2016). Fully convolutional neural networks with hyperlocal features for brain tumor segmentation. In *Proceedings MICCAI-BRATS Workshop*, pages 4–9.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*.

Chen, L.-C., Papandreou, G., Schroff, F., and Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*.

Cheng, K., Montgomery, D., Feng, Y., Steel, R., Liao, H., McLaren, D. B., Erridge, S. C., McLaughlin, S., and Nailon, W. H. (2015). Identifying radiotherapy target volumes in brain cancer by image analysis. *Healthcare technology letters*, 2(5):123–128.

Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., and Ronneberger, O. (2016). 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention*, pages 424–432. Springer.

Clark, M. C., Hall, L. O., Goldgof, D. B., Velthuizen, R., Murtagh, F. R., and Silbiger, M. S. (1998). Automatic tumor segmentation using knowledge-based techniques. *IEEE transactions on medical imaging*, 17(2):187–201.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

Dalca, A. V., Balakrishnan, G., Guttag, J., and Sabuncu, M. R. (2019). Unsupervised learning of probabilistic diffeomorphic registration for images and surfaces. *Medical image analysis*, 57:226–236.

DeAngelis, L. M. (2001). Brain tumors. *New England journal of medicine*, 344(2):114–123.

Desimone, R., Moran, J., and Spitzer, H. (1989). Neural mechanisms of attention in extrastriate cortex of monkeys. In *Dynamic interactions in neural networks: Models and data*, pages 169–182. Springer.

Drozdzal, M., Vorontsov, E., Chartrand, G., Kadoury, S., and Pal, C. (2016). The importance of skip connections in biomedical image segmentation. In *Deep Learning and Data Labeling for Medical Applications*, volume 10008, pages 179–187. Springer.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Dumoulin, V. and Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*.

El-Dahshan, E.-S. A., Mohsen, H. M., Revett, K., and Salem, A.-B. M. (2014). Computer-aided diagnosis of human brain tumor through mri: A survey and a new algorithm. *Expert systems with Applications*, 41(11):5526–5545.

Ellwaa, A., Hussein, A., AlNaggar, E., Zidan, M., Zaki, M., Ismail, M. A., and Ghanem, N. M. (2016). Brain tumor segmantation using random forest trained on iteratively selected patients. In *International Workshop on Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, pages 129–137. Springer.

Farahani, F., Reyes, M., Menze, B., Gerstner, E., Kirby, J., and Kalpathy-Cramer, J. (2013). Nci-miccai 2013 challenge on multimodal brain tumor segmentation (brats). *The challenge database contains fully anonymized images from the following institutions: ETH Zurich, University of Bern, University of Debrecen, and University of Utah and publicly available images from the Cancer Imaging Archive (TCIA)*, 8.

Fletcher-Heath, L. M., Hall, L. O., Goldgof, D. B., and Murtagh, F. R. (2001). Automatic segmentation of non-enhancing brain tumors in magnetic resonance images. *Artificial intelligence in medicine*, 21(1-3):43–63.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202.

Fukushima, K. and Miyake, S. (1982). Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern recognition*, 15(6):455–469.

Gao, X., Shi, M., Song, X., Zhang, C., and Zhang, H. (2019). Recurrent neural networks for real-time prediction of tbm operating parameters. *Automation in Construction*, 98:225–235.

Geremia, E., Clatz, O., Menze, B. H., Konukoglu, E., Criminisi, A., and Ayache, N. (2011). Spatial decision forests for ms lesion segmentation in multi-channel magnetic resonance images. *NeuroImage*, 57(2):378–390.

Gering, D. T., Grimson, W. E. L., and Kikinis, R. (2002). Recognizing deviations from normalcy for brain tumor segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 388–395. Springer.

Ghebrechristos, H. and Alaghband, G. (2018). Exploring deep learning using information theory tools and patch ordering. *International Conference on Learning Representations.*

Gibbs, P., Buckley, D. L., Blackband, S. J., and Horsman, A. (1996). Tumour volume determination from mr images by morphological segmentation. *Physics in Medicine & Biology*, 41(11):2437.

Glodek, M., Reuter, S., Schels, M., Dietmayer, K., and Schwenker, F. (2013). Kalman filter based classifier fusion for affective state recognition. In *International workshop on multiple classifier systems*, pages 85–94. Springer.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323.

Goetz, M., Weber, C., Binczyk, F., Polanska, J., Tarnawski, R., Bobek-Billewicz, B., Koethe, U., Kleesiek, J., Stieltjes, B., and Maier-Hein, K. H. (2015). Dalsa: domain adaptation for supervised learning from sparsely annotated mr images. *IEEE transactions on medical imaging*, 35(1):184–196.

Goodman, C. C. and Fuller, K. S. (2014). *Pathology-E-Book: Implications for the Physical Therapist*. Elsevier Health Sciences.

Gordon, J. and Shortliffe, E. (1984). The mycin experiments of the stanford heuristic programming project. *Rule-Based Expert Systems*.

Görlitz, L., Menze, B. H., Weber, M.-A., Kelm, B. M., and Hamprecht, F. A. (2007). Semi-supervised tumor detection in magnetic resonance spectroscopic images using discriminative random fields. In *Joint Pattern Recognition Symposium*, pages 224–233. Springer.

Grobner, T. (2006). Gadolinium–a specific trigger for the development of nephrogenic fibrosing dermopathy and nephrogenic systemic fibrosis? *Nephrology Dialysis Transplantation*, 21(4):1104–1108.

Gupta, A. and Dwivedi, T. (2017). A simplified overview of world health organization classification update of central nervous system tumors 2016. *Journal of neurosciences in rural practice*, 8(4):629.

Hashemi, S. R., Salehi, S. S. M., Erdogmus, D., Prabhu, S. P., Warfield, S. K., and Gholipour, A. (2018). Asymmetric loss functions and deep densely-connected networks for highly-imbalanced medical image segmentation: Application to multiple sclerosis lesion detection. *IEEE Access*, 7:1721–1735.

Havaei, M., Davy, A., Warde-Farley, D., Biard, A., Courville, A., Bengio, Y., Pal, C., Jodoin, P.-M., and Larochelle, H. (2017). Brain tumor segmentation with deep neural networks. *Medical image analysis*, 35:18–31.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

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

Herzog, M. H. and Clarke, A. M. (2014). Why vision is not both hierarchical and feedforward. *Frontiers in computational neuroscience*, 8:135.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

Ho, S., Bullitt, E., and Gerig, G. (2002). Level-set evolution with region competition: automatic 3-d segmentation of brain tumors. In *Object recognition supported by user interaction for service robots*, volume 1, pages 532–535. IEEE.

Holland, E. C. (2001). Progenitor cells and glioma formation. *Current opinion in neurology*, 14(6):683–688.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *CVPR*, volume 1, page 3.

Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154.

Hussain, S., Anwar, S. M., and Majid, M. (2018). Segmentation of glioma tumors in brain using deep convolutional neural network. *Neurocomputing*, 282:248–261.

Im, D. J., Kim, C. D., Jiang, H., and Memisevic, R. (2016). Generating images with recurrent adversarial networks. *arXiv preprint arXiv:1602.05110*.

Işın, A., Direkoğlu, C., and Şah, M. (2016). Review of mri-based brain tumor image segmentation using deep learning methods. *Procedia Computer Science*, 102:317–324.

Jiang, Y. and Uhrbom, L. (2012). On the origin of glioma. *Upsala journal of medical sciences*, 117(2):113–121.

Kalinić, H. (2009). Atlas-based image segmentation: A survey.

Kamnitsas, K., Ferrante, E., Parisot, S., Ledig, C., Nori, A. V., Criminisi, A., Rueckert, D., and Glocker, B. (2016). Deepmedic for brain tumor segmentation. In *International workshop on Brainlesion: Glioma, multiple sclerosis, stroke and traumatic brain injuries*, pages 138–149. Springer.

Kamnitsas, K., Ledig, C., Newcombe, V. F., Simpson, J. P., Kane, A. D., Menon, D. K., Rueckert, D., and Glocker, B. (2017). Efficient multi-scale 3d cnn with fully connected crf for accurate brain lesion segmentation. *Medical image analysis*, 36:61–78.

Kaus, M. R., Warfield, S. K., Nabavi, A., Black, P. M., Jolesz, F. A., and Kikinis, R. (2001). Automated segmentation of mr images of brain tumors. *Radiology*, 218(2):586–591.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Lai, M. (2015). Deep learning for medical image segmentation. *arXiv preprint arXiv:1505.02000*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, pages 609–616.

Lefohn, A. E., Cates, J. E., and Whitaker, R. T. (2003). Interactive, gpu-based level sets for 3d segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 564–572. Springer.

Letteboer, M. M., Olsen, O. F., Dam, E. B., Willems, P. W., Viergever, M. A., and Niessen, W. J. (2004). Segmentation of tumors in magnetic resonance brain images using an interactive multiscale watershed algorithm1. *Academic Radiology*, 11(10):1125–1138.

Lin, M., Chen, Q., and Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Lindberg, N., Kastemar, M., Olofsson, T., Smits, A., and Uhrbom, L. (2009). Oligodendrocyte progenitor cells can act as cell of origin for experimental glioma. *Oncogene*, 28(23):2266–2275.

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., Van Der Laak, J. A., Van Ginneken, B., and Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88.

Liu, L., Zhang, H., Rekik, I., Chen, X., Wang, Q., and Shen, D. (2016). Outcome prediction for patient with high-grade gliomas from brain functional and structural networks. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 26–34. Springer.

Logeswari, T. and Karnan, M. (2009). An improved implementation of brain tumor detection using segmentation based on soft computing. *Journal of Cancer Research and Experimental Oncology*, 2(1):006–014.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

Louis, D. N., Perry, A., Reifenberger, G., Von Deimling, A., Figarella-Branger, D., Cavenee, W. K., Ohgaki, H., Wiestler, O. D., Kleihues, P., and Ellison, D. W. (2016). The 2016 world health organization classification of tumors of the central nervous system: a summary. *Acta neuropathologica*, 131(6):803–820.

Manassi, M., Sayim, B., and Herzog, M. H. (2013). When crowding of crowding leads to uncrowding. *Journal of Vision*, 13(13):10–10.

Mazzara, G. P., Velthuizen, R. P., Pearlman, J. L., Greenberg, H. M., and Wagner, H. (2004). Brain tumor target volume determination for radiation treatment planning through automated mri segmentation. *International Journal of Radiation Oncology* Biology* Physics*, 59(1):300–312.

McDonald, R. J., McDonald, J. S., Kallmes, D. F., Jentoft, M. E., Murray, D. L., Thielen, K. R., Williamson, E. E., and Eckel, L. J. (2015). Intracranial gadolinium deposition after contrast-enhanced mr imaging. *Radiology*, 275(3):772–782.

Menze, B. H., Jakab, A., Bauer, S., Kalpathy-Cramer, J., Farahani, K., Kirby, J., and Burren, e. (2015). The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). *IEEE Transactions on Medical Imaging*, 34(10):1993–2024.

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

Menze, B. H., Van Leemput, K., Lashkari, D., Weber, M.-A., Ayache, N., and Golland, P. (2010). A generative model for brain tumor segmentation in multi-modal images. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 151–159. Springer.

Milletari, F., Navab, N., and Ahmadi, S.-A. (2016). V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571. IEEE.

Mlynarski, P., Delingette, H., Criminisi, A., and Ayache, N. (2019). 3d convolutional neural networks for tumor segmentation using long-range 2d context. *Computerized Medical Imaging and Graphics*, 73:60–72.

Moon, N., Bullitt, E., Van Leemput, K., and Gerig, G. (2002). Model-based brain and tumor segmentation. In *Object recognition supported by user interaction for service robots*, volume 1, pages 528–531. IEEE.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective.* MIT press.

Ono, T. and Nishijo, H. (1992). Neurophysiological basis of the klüver-bucy syndrome: Responses of monkey amygdaloid neurons to biologically significant objects.

Ostrom, Q. T., Bauchet, L., Davis, F. G., Deltour, I., Fisher, J. L., Langer, C. E., Pekmezci, M., Schwartzbaum, J. A., Turner, M. C., Walsh, K. M., et al. (2014). The epidemiology of glioma in adults: a "state of the science" review. *Neuro-oncology*, 16(7):896–913.

Ozkan, M., Dawant, B. M., and Maciunas, R. J. (1993). Neural-network-based segmentation of multi-modal medical images: a comparative and prospective study. *IEEE transactions on Medical Imaging*, 12(3):534–544.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Pereira, S., Oliveira, A., Alves, V., and Silva, C. A. (2017). On hierarchical brain tumor segmentation in mri using convolutional neural networks: a preliminary study. In *Bioengineering (ENBENG), 2017 IEEE 5th Portuguese Meeting on*, pages 1–4. IEEE.

Pereira, S., Pinto, A., Alves, V., and Silva, C. A. (2015). Deep convolutional neural networks for the segmentation of gliomas in multi-sequence mri. In *International Workshop on Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, pages 131–143. Springer.

Pham, D. L., Xu, C., and Prince, J. L. (2000). Current methods in medical image segmentation. *Annual review of biomedical engineering*, 2(1):315–337.

Pollak, L., Walach, N., Gur, R., and Schiffer, J. (1998). Meningiomas after radiotherapy for tinea capitis-still no history. *Tumori Journal*, 84(1):65–68.

Prastawa, M., Bullitt, E., Ho, S., and Gerig, G. (2004). A brain tumor segmentation framework based on outlier detection. *Medical image analysis*, 8(3):275–283.

Prastawa, M., Bullitt, E., Moon, N., Van Leemput, K., and Gerig, G. (2003). Automatic brain tumor segmentation by subject specific modification of atlas priors1. *Academic radiology*, 10(12):1341–1348.

Prechelt, L. (1998). Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761–767.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Rosasco, L., Vito, E. D., Caponnetto, A., Piana, M., and Verri, A. (2004). Are loss functions all the same? *Neural Computation*, 16(5):1063–1076.

Rosenblatt, F. (1961). Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Schmidt, M., Levner, I., Greiner, R., Murtha, A., and Bistritz, A. (2005). Segmenting brain tumors using alignment-based features. In *Fourth International Conference on Machine Learning and Applications (ICMLA'05)*, pages 6–pp. IEEE.

Schneider, T., Mawrin, C., Scherlach, C., Skalej, M., and Firsching, R. (2010). Gliomas in adults. *Deutsches Ärzteblatt International*, 107(45):799.

Seghier, M. L., Ramlackhansingh, A., Crinion, J., Leff, A. P., and Price, C. J. (2008). Lesion identification using unified segmentation-normalisation models and fuzzy clustering. *Neuroimage*, 41(4):1253–1266.

Shortliffe, E. H., Davis, R., Axline, S. G., Buchanan, B. G., Green, C. C., and Cohen, S. N. (1975). Computer-based consultations in clinical therapeutics: explanation and rule acquisition capabilities of the mycin system. *Computers and biomedical research*, 8(4):303–320.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Singh, A., Bajpai, S., Karanam, S., Choubey, A., and Raviteja, T. (2012). Malignant brain tumor detection. *International Journal of Computer Theory and Engineering*, 4(6):1002.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Stadlbauer, A., Moser, E., Gruber, S., Buslei, R., Nimsky, C., Fahlbusch, R., and Ganslandt, O. (2004). Improved delineation of brain tumors: an automated method for segmentation based on pathologic changes of 1h-mrsi metabolites in gliomas. *Neuroimage*, 23(2):454–461.

Sugita, R., Ito, K., Fujita, N., and Takahashi, S. (2010). Diffusion-weighted mri in abdominal oncology: clinical applications. *World journal of gastroenterology: WJG*, 16(7):832.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June-2015:1–9.

Tustison, N., W. M. A. B. (2013). Ants and árboles. In *Proceedings MICCAI-BRATS challenge*, pages 47–50.

Urban, G., Bendszus, M., Hamprecht, F., and Kleesiek, J. (2014). Multi-modal brain tumor segmentation using deep convolutional neural networks. *MICCAI BraTS (Brain Tumor Segmentation) Challenge. Proceedings, winning contribution*, pages 31–35.

Van Leemput, K., Maes, F., Vandermeulen, D., Colchester, A., and Suetens, P. (2001). Automated segmentation of multiple sclerosis lesions by model outlier detection. *IEEE transactions on medical imaging*, 20(8):677–688.

Visin, F., Kastner, K., Courville, A., Bengio, Y., Matteucci, M., and Cho, K. (2015). Reseg: A recurrent neural network for object segmentation. *arXiv preprint arXiv:1511.07053*.

Vovk, U., Pernus, F., and Likar, B. (2007). A review of methods for correction of intensity inhomogeneity in mri. *IEEE transactions on medical imaging*, 26(3):405–421.

Walter, A. W., Hancock, M. L., Pui, C.-H., Hudson, M. M., Ochs, J. S., Rivera, G. K., Pratt, C. B., Boyett, J. M., and Kun, L. E. (1998). Secondary brain tumors in children treated for acute lymphoblastic leukemia at st jude children's research hospital. *Journal of clinical oncology*, 16(12):3761–3767.

Wu, W., Chen, A. Y., Zhao, L., and Corso, J. J. (2014). Brain tumor detection and segmentation in a crf (conditional random fields) framework with pixel-pairwise affinity and superpixel-level features. *International journal of computer assisted radiology and surgery*, 9(2):241–253.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5987–5995. IEEE.

Yang, S. and Ramanan, D. (2015). Multi-scale recognition with dag-cnns. In *Proceedings of the IEEE international conference on computer vision*, pages 1215–1223.

Yazdani, S., Yusof, R., Karimian, A., Pashna, M., and Hematian, A. (2015). Image segmentation methods and applications in mri brain images. *IETE Technical Review*, 32(6):413–427.

Young, R. J. and Knopp, E. A. (2006). Brain mri: tumor evaluation. *Journal of Magnetic Resonance Imaging: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 24(4):709–724.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

Zeiler, M. D., Taylor, G. W., Fergus, R., et al. (2011). Adaptive deconvolutional networks for mid and high level feature learning. In *ICCV*, volume 1, page 6.

Zhang, C., Shen, X., Cheng, H., and Qian, Q. (2019). Brain tumor segmentation based on hybrid clustering and morphological operations. *International Journal of Biomedical Imaging*, 2019.

Zhang, J., Ma, K.-K., Er, M.-H., and Chong, V. (2004). Tumor segmentation from magnetic resonance imaging by learning via one-class support vector machine.

Zhang, Y., Brady, M., and Smith, S. (2001). Segmentation of brain mr images through a hidden markov random field model and the expectation-maximization algorithm. *IEEE transactions on medical imaging*, 20(1):45–57.

Zhao, X., Wu, Y., Song, G., Li, Z., Zhang, Y., and Fan, Y. (2018). A deep learning model integrating fcnns and crfs for brain tumor segmentation. *Medical image analysis*, 43:98–111.

Zikic, D., Glocker, B., Konukoglu, E., Criminisi, A., Demiralp, C., Shotton, J., Thomas, O. M., Das, T., Jena, R., and Price, S. J. (2012). Decision forests for tissue-specific segmentation of high-grade gliomas in multi-channel mr. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 369–376. Springer.

Zikic, D., Ioannou, Y., Brown, M., and Criminisi, A. (2014). Segmentation of brain tumor tissues with convolutional neural networks. *Proceedings MICCAI-BRATS*, pages 36–39.