


République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université AMO de Bouira

Faculté des Sciences et des Sciences Appliquées

Département d'Informatique

Mémoire De Fin d'Etude De Master

Spécialité : **GSI**

DEEP-LEARNING APPLIQUE A LA SEGMENTATION DES IMAGES MEDICALES

Réalisé par :

— Saadi Hocine

— Saidi Yasser

Encadré par :

— Prof Z.Guessoum

— Prof Z.Chouiref

2020/2021

Remerciements

Louange à ***ALLAH*** par la grâce duquel les bonnes choses se réalisent, nous tenons à remercier notre ***DIEU*** de tout puissant, miséricordieux, de nous avoir donné la force, la santé, la volonté et le courage d'accomplir ce modeste travail.

Nous remercions également toutes les personnes qui nous soutenu de prés ou de loin, en particulier :

Notre encadrante **Prof Z.Guessoum** pour sa disponibilité, et son précieux suivi tout au long de la réalisation de ce travail.

Nous souhaitons également remercier **Prof Z.Chouiref** pour ses conseils et l'aide qu'elle nous a apporté.

Les membres du jury d'avoir consacré une partie de leur temps à la lecture de ce mémoire, et pour l'intérêt qu'ils ont porté à ce travail.

Tous nos enseignants du département d'Informatique de l'Université de Bouira Akli Mohand Oulhadj

À tous, un grand merci...

Dédicaces

J'ai l'honneur de dédier ce modeste travail à ceux qui m'ont encouragé et soutenu.

Mon très cher père qui était toujours à mes cotés à tous moment.

Ma très chère mère, qui me donne toujours l'espoir de vivre.

Mes frères et mes sœurs pour leur soutien.

Toute ma famille et ainsi mes amis et spécialement : **Houssam** et **Mohamed**.

Que dieu les garde pour moi.

Saidi Yasser..

J'ai l'honneur de dédier ce modeste travail à ceux qui m'ont encouragé et soutenu.

Ma très chère mère, qui me donne toujours l'espoir de vivre.

Mon très cher père qui était toujours à mes cotés à tous moment.

Mes frères pour leur soutien et spécialement :

Mohamed saadi pour leur encouragement

Toute ma famille et ainsi mes amis.

Que dieu les garde pour moi.

Saadi Hocine..

Table des matières

Résumé

La Segmentation d'images basée sur l'apprentissage en profondeur est désormais établie comme un outil robuste de segmentation d'images. Il a été largement utilisé pour séparer les zones homogènes en tant que premier élément critique du pipeline de diagnostic et de traitement.

L'objectif de ce travail est de proposer une approche de Deep Learning dans le domaine de l'épidémiologie pour détecter les tumeurs cérébrales.

Pour ce faire, nous avons choisi d'utiliser les réseaux de neurones convolutifs (CNN), où différents modèles ont été implémentés nous permettant d'obtenir les meilleurs résultats.

Dans cette étude, nous avons proposé deux approches de réseau neuronal convolutif et créé deux modèles fondés sur ces approches, les modèles créés ont été initiés et ont prouvé leur efficacité en atteignant une précision élevée et un score F1 dans leurs tests à l'aide de données BraTS 2019.

Mots clés : Segmentation d'images, réseau de neurones convolutifs, traitement d'images, classification d'images, imagerie médicale, intelligence artificielle, apprentissage automatique, apprentissage en profondeur, réseaux de neurones.

Abstract

Deep learning-based image segmentation is by now firmly established as a robust tool in image segmentation. It has been widely used to separate homogeneous areas as the first and critical component of diagnosis and treatment pipeline.

The objective of this work is to propose a Deep Learning approach in the field of epidemiology to detect brain tumer.

To do this, we have chosen to use the Convolutional Neural networks (CNN), where different models have been implemented allowing us to obtain the best results.

In this study, we proposed two Convolutional Neural Network approaches and crated two models based on these approaches, the created models were trained and prove their efficiency by achieving high accuracy and F1-Score in their testing using BraTS 2019 Data.

Key words : Image Segmentation,Convolutional neural network, image processing, image classification, Medical imaging ,Artificial intelligence, Machine Learning, Deep Learning, Neural networks.

الملخص

أصبحت عملية تجزئة الصور القائمة على التعلم العميق الآن كأداة قوية في تجزئة الصور. يتم استخدامها في نطاق واسع لفصل المناطق المتجانسة، و تعتبر مكون أول وحاسم في عمليات التشخيص والعلاج.

الهدف من هذا العمل هو اقتراح نهج من التعلم العميق في مجال علم الأوبئة للكشف عن ورم الدماغ.

للقيام بذلك، اخترنا استخدام الشبكات العصبية التلافيفية CNN ، حيث تم تنفيذ نماذج مختلفة مما يتيح لنا الحصول على أفضل النتائج.

في هذه الدراسة، اقترحنا نهجين للشبكة العصبية التلافيفية وصممنا نموذجين بناءً على هذه الأساليب، وتم تدريب النماذج التي تم إنشاؤها وإثبات كفاءتها من خلال تحقيق دقة عالية و F1-Score في اختبارها باستخدام بيانات BraTS 2019 .

الكلمات الدالة : التصوير الطبي، ورم الدماغ، الذكاء الاصطناعي، التعلم الآلي، التعلم العميق، الشبكات العصبية، الشبكة العصبية التلافيفية، معالجة الصور، تصنيف الصور،تجزئة الصور
.

Table des figures

Liste des tableaux

Liste des abréviations

AI	Artificial Intelligence
DL	Deep Learning
ML	Machine Learning
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
BRATS	Brain Tumor Segmentation
TIFF	Tagged Image File Format
IRM	Imagerie par Résonance Magnétique
HGG	High Grade Glioma
LGG	Low Grade Glioma
ResUNet	Deep Residual UNET
ReLU	Rectified Linear Unit
Conv	Convolution Layer

Liste des équations

Introduction générale

La Segmentation d'image consiste à séparer l'image en des zones simples avec une certaine homogénéité. C'est une partition de l'image en ensembles de pixels homogènes selon un critère prédéfini (ou voxel dans les cas des images 3D). La segmentation n'est pas unique et ses résultats sont différents selon : les algorithmes utilisés, les critères d'homogénéité, l'initialisation.

L'adaptabilité des systèmes de segmentation est le principal défi auquel sont confrontés les chercheurs dans le domaine de l'imagerie médicale. La difficulté réside dans la complexité des images médicales. Ce sont souvent des images non photographiques comme c'est le cas pour les images IRM, qui sont générées à partir d'ondes électromagnétiques. Ces ondes sont émises par les noyaux d'hydrogènes lors de leurs relaxations. Ce type d'image est caractérisé par la présence de formes complexes dont l'apparence varie d'un sujet à un autre. On remarque, également, la présence d'un bruit assez considérable dans la plupart des cas, et une résolution pas toujours satisfaisante.

Par ailleurs, le Deep Learning a récemment connu des succès opérationnels. Dans le traitement d'images médicales, notamment à travers les performances spectaculaires obtenues par les réseaux de neurones convolutifs (ConvNets) au challenge ImageNet 2012.

L'objectif de ce projet est de découvrir quelques approches de Deep Learning utilisées pour la segmentation d'images médicales, choisir une architecture de Deep Learning, réaliser son implémention et l'appliquer au dataset BraTS19.

Notre mémoire est organisée comme suit :

— **Chapitre 1 : Traitement d'image**
Dans ce chapitre, nous avons traité les notions de base nécessaires et la compréhension des techniques de traitement d'images et quelques méthodes de segmentation.

— **Chapitre 2 : Deep Learning**
Dans ce chapitre,nous parlerons de certains approches De Deep Learning.

— **Chapitre 3 : Les Architectures proposées pour la Segmentation d'images Biomédicales**
Dans ce chapitre, nous parlerons de quelques algorithmes courants utilisés pour La segmentation d'images médicales.

— **Chapitre 4 : Matériels et Méthodes**
Ce chapitre sera divisé en deux sections, Dans la première section, nous parlerons de base de données que nous utiliserons, en mentionnant tous ses détails. Après cela, nous passons aux méthodes section, où nous parlerons de nos modèles proposés.

— **Chapitre 5 : Réalisation**
Dans ce chapitre, nous allons implémenter nos modèles proposés et discuter des résultats obtenu.

Enfin, nous terminons ce travail par une conclusion générale.

Chapitre 1

Traitement d'images

1.1 Introduction

Les images sont l'une des solutions les plus importantes que les humains utilisent pour résoudre les problèmes Depuis la naissance des êtres humains, les connaissances et les informations se sont répandues et transmises à travers le monde. parce qu'une grande quantité d'informations peut être collectée avec une seule image. Un système de traitement d'image comprend principalement les fonctions suivantes : Acquisition d'image (pré-traitement pour la réduction du bruit), analyse de l'image pour obtenir une description synthétique des informations originales contenues dans l'image.

On désigne par traitement d'images numériques l'ensemble des techniques permettant de modifier une image numérique afin d'améliorer ou d'en extraire des informations.

Dans ce chapitre nous abordons les notions de base nécessaires à la compréhension des techniques de traitement d'images ensuite nous présentons les différentes techniques connues dans ce domaine.

1.2 Images numériques

La définition du terme « image » lui-même, tel qu'elle est donnée par exemple par le Petit Robert, englobe une multitude de significations distinctes. Cela va de la « reproduction exacte ou représentation analogique d'un être, d'une chose », à la « représentation mentale d'origine sensible » ou à des concepts plus physiques comme un « ensemble des points » où vont converger des rayons lumineux (cas des images optiques) [1].

Les images numériques sont constituées d'un ensemble de pixels (picture éléments), juxtaposés en lignes et en colonnes. Le pixel, (qui correspond à un point ou petit carré), est le plus petit élément que l'on peut trouver dans une image. Chaque pixel possède des caractéristiques propres, couleurs, luminosité, brillance, qui permettent de les différencier et de composer les images [2].

Pour un informaticien, une image est un type, une structure de données, qu'il appelle aussi structure de données maillées. Les images usuelles (celles qui peuvent être lues par un logiciel du commerce) s'appuient sur un support $\mathcal{D}$ dont la topologie est un maillage carré 2D régulier [3] :

$$\mathcal{D} = [0, C] \times [0, L]^{1} \tag{1.1}$$

1.3 Acquisition d'une image

L'acquisition d'images constitue un des maillons essentiels de toute chaîne de conception et de production d'images. Pour pouvoir manipuler une image sur un système informatique, il est avant tout nécessaire de lui faire subir une transformation qui la rendra lisible et manipulable par ce système. Le passage de cet objet externe (l'image d'origine) à sa représentation interne se fait grâce à une procédure de numérisation [4].

Ces systèmes de saisie, dénommés optiques, peuvent être classés en deux catégories principales :

— Les caméras numériques.

— Les scanners.

1. Nous notons que, C = nombre de colonnes et L = nombre de lignes

1.4 Caractéristiques d'une image numérique

1.4.1 Dimension

C'est la taille de l'image. Cette dernière se présente sous forme de matrice dont les éléments sont des valeurs numériques représentatives des intensités lumineuses (pixels). Le nombre de lignes de cette matrice multiplié par le nombre de colonnes nous donne le nombre total de pixels dans une image [4].

1.4.2 Résolution

La résolution d'une image est le nombre de pixels par pouce qu'elle contient (1 pouce = 2.54 centimètres). Elle est exprimée en "PPP" (points par pouce) ou DPI (dots per inch). Plus il y a de pixels (ou points) par pouce et plus il y aura d'information dans l'image (plus précise). Par exemple, une résolution de 300dpi signifie que l'image comporte 300 pixels dans sa largeur et 300 pixels dans sa hauteur, elle est donc composée de 90 000 pixels (300x300 ppp). Grâce à cette formule, il est facile de connaître la dimension maximale d'un tirage [5].

1.4.3 Bruit

Un bruit (parasite) dans une image est considéré comme un phénomène de brusque variation de l'intensité d'un pixel par rapport à ses voisins, il provient de l'éclairage des dispositifs optiques et électroniques du capteur [4].

1.4.4 Histogramme

L'histogramme des niveaux de gris ou des couleurs d'une image est une fonction qui permet de représenter visuellement les zones lumineuses dans une image.

Pour diminuer l'erreur de quantification, pour comparer deux images obtenues sous des éclairages différents, ou encore pour mesurer certaines propriétés sur une image, on modifie souvent l'histogramme correspondant [4].

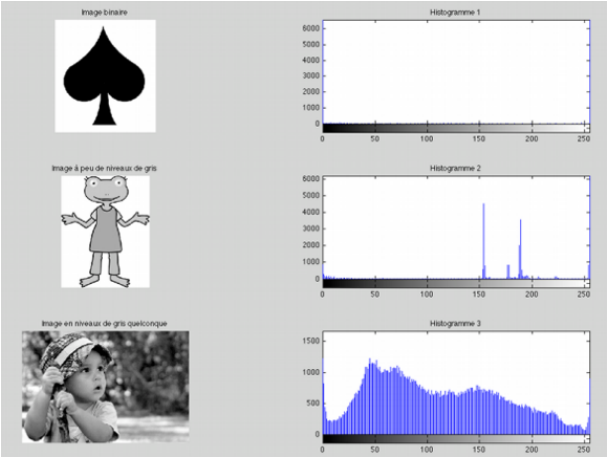


FIGURE 1.1 – Histogramme associé à des images.

1.5 Prétraitement d'images

Cette phase a lieu juste après l'acquisition des images elle permet d'améliorer la qualité de l'image.

Parmi les techniques de prétraitement d'images les plus importantes, il y a :

— La modification d'histogramme,

— La réduction du bruit par filtrage

1.5.1 Modification d'histogramme

La modification d'histogramme et les méthodes de traitement d'images font partie de la classe des traitements dits ponctuels : la valeur de chaque pixel est corrigée, et ce indépendamment des autres pixels [6].

Dans ce qui suit, on va présenter deux types de modification d'histogramme :

- Étirement d'histogrammes
- Égalisation d'histogramme

- **Étirement d'histogrammes :**
  Cette transformation consiste à la luminosité de l'image à l'aide de la règle de trois, la valeur de chaque pixel est remplacée par le résultat de la formule ci-dessous :

$$I'(x,y) = 255 \times \frac{I(x,y) - Imin}{Imax - Imin} \tag{1.2}$$

  — $I(x,y)$ $I'(x,y)$ et désignent les intensités du pixel de coordonnées (x,y) respectivement dans l'image mal exposée et la nouvelle image.
  — $Imin$ et $Imax$ sont souvent déterminées comme les niveaux de gris minimum et maximum de l'image.

- **Égalisation d'histogramme :**
  Cette transformation permet d'améliorer le constat, que consiste à appliquer une transformation sur chaque pixel de l'image à partir des opération sur chaque pixel :
  — Étape 1 : calcule l'histogramme $h(i) i \in [0, 255]$.
  — Étape 2 : normalisation de l'histogramme $h(i)$.
  — Étape 3 : densité de probabilité cumulative.
  — Étape 4 : transformation des niveaux de gris de l'image :

$$I'(x,y) = C(I(x,y)) * 255 \tag{1.3}$$

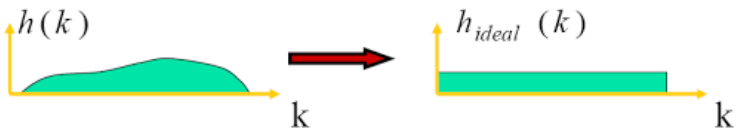


FIGURE 1.2 – Égalisation d'histogramme.

1.5.2 Réduction du bruit par filtrage

En général, le bruit d'images est considéré comme un champ aléatoire caractérisé par sa densité de probabilité f et sa fonction de répartition F. La nature plus ou moins impulsionnelle du bruit peut être décrite à l'aide de densité de probabilités de la forme [6] :

$$f(a) = C.exp(-K|a|^{\alpha}) \tag{1.4}$$

Différentes méthodes de filtrage ont été développées suivant le type et l'intensité du bruit, ou les applications auxquelles on destine l'image. Les premières et les plus simples de ces méthodes sont basées sur le filtrage linéaire stationnaire (invariant par translations), mais les limitations de ces techniques (en particulier leur mauvaise conservation des transitions) a conduit au développement des filtres "non-linéaire" [7].

1.6 Segmentation

La segmentation d'image est une phase d'importance capitale dans tout processus de vision par ordinateur [8].

Le processus de segmentation d'une image est le processus qui la découpe en région dans un but de simplifier la représentation de l'image pour la rendre plus facile à analyser et à interpréter.

La segmentation d'images est typiquement utilisée pour localiser des objets ou reconnaître des contours d'objets. Techniquement la segmentation d'images est un processus d'étiquetage des pixels qui permet de donner un ensemble de pixels couvrant la totalité de l'image ou un ensemble de contours délimitant des régions représentant une certain homogénéité de couleur, intensité, texture.

On peut diviser les méthodes de segmentation en deux approches, (i) la première est fondée sur la recherche de discontinuité locale (détection des contours). (ii) la deuxième cherche à détecter des zones de l'image présentent des caractéristiques d'homogénéité (extraction de région). Les deux approches sont duales en sens qu'une région définit une ligne par son contour et un contour fermé définit une région [9].

Les approches de segmentation :
— Segmentation par classification.
— Segmentation basée sur les contours.
— Segmentation basée sur les régions.

1.6.1 Segmentation par Classification :

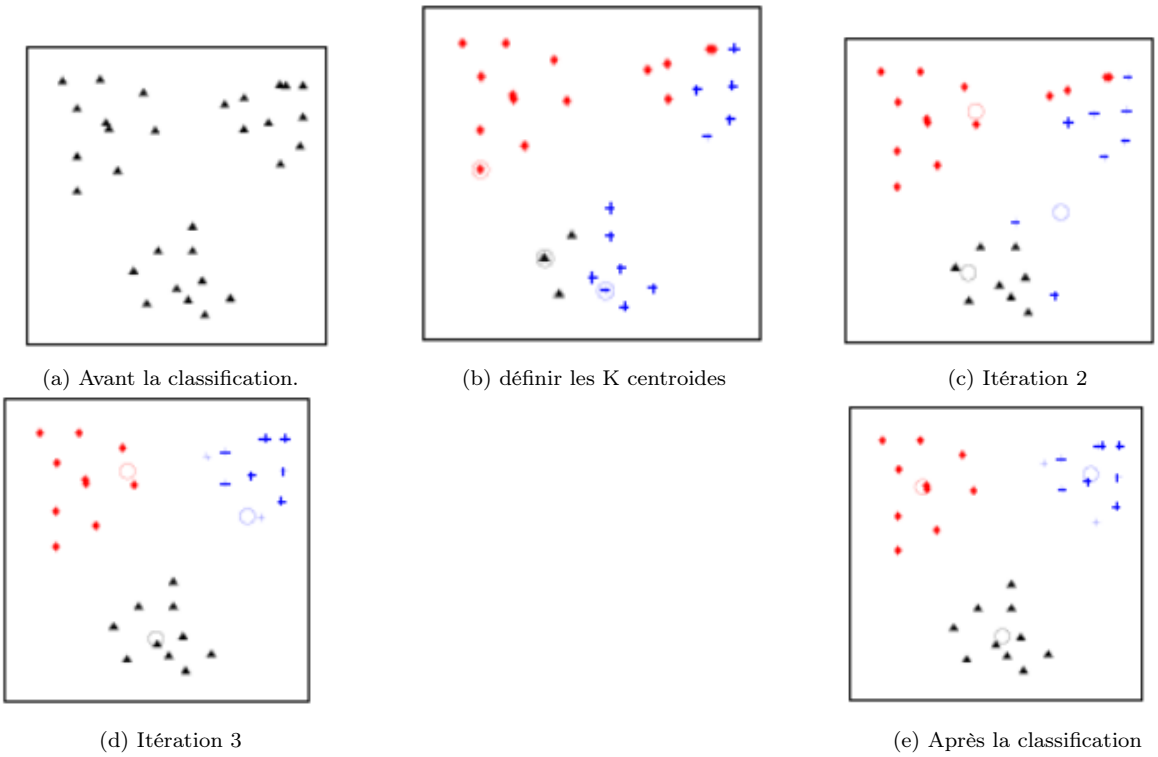
La segmentation par classification consiste à partitionner une image en un ensemble de classes disjointes, pour chaque pixel de l'image on attribue une étiquette parmi l'ensemble des étiquettes qui correspondent chacune à une classe. On dit qu'une classification est supervisée si le nombre de classes est connu a priori sinon il s'agit d'une classification non supervisée.

Parmi les méthodes les plus utilisées en segmentation d'images par classification l'algorithme de K-means (algorithme des centres mobiles) qui utilise la distance

quadratique moyenne comme critère d'évaluation d'une partition [10].

Algorithme de K-means :

1. Choisir un nombre de classe K.
2. Définir une classification C aléatoire (définir les K centroides Yi de façon aléatoire dans l'espace des niveaux de gris).
3. **Tant que** l'inertie intra classe n'est pas stable **faire** :
   Affecter chaque niveau de gris dont le centre est le plus proche ;
   Calculer les centres de gravité des classes de la nouvelle classification C' des classes ;
   C $\leftarrow$ C' ;
   Fin tant que
4. Afficher la classification obtenue.



(a) Avant la classification. (b) définir les K centroides (c) Itération 2

(d) Itération 3 (e) Après la classification

FIGURE 1.3 – Algorithme de K-Means [10].

Sur l'exemple ci-dessous, on peut observer une image avant et après la segmentation par classification :

(a) Image original

(b) Image après segmentation.

FIGURE 1.4 – Le résultat de l'algorithme de segmentation par classification.[11]

1.6.2 Détection de conteur

La détection de conteur et une technique de réduction d'information dans les images Les contours constituent en effet des indices riches, au même titre que les points d'intérêts, pour toute interprétation ultérieure de l'image [6]. Les contours dans une image proviennent des :

— discontinuités de la fonction de réflectance (texture, ombre).

— discontinuités de profondeur (bords de l'objet).

et sont caractérisés par des discontinuités de la fonction d'intensité dans les images

FIGURE 1.5 – Différents types de contours.

Le principe classique de la détection de contours repose sur l'étude des dérivées de la fonction d'intensité dans l'image : les extréma locaux du gradient de la fonction d'intensité et les passages par zéro du LaPlacien [6].

Filtrage linéaire

Le Filtrage linéaire est la méthode générale pour appliquer un filtre (par exemple dérivateur) sur une image. Cela signifie convoluer une image I(x, y) avec une fonction f(x, y).Un filtre linéaire transforme un ensemble de données d'entrée en un ensemble de données de sortie selon une opération mathématique appelée convolution.

Le produit de convolution de deux fonctions réelles ou complexes f et g, est une autre

fonction, qui se note généralement « $f * g$ » et qui est définie par :

$$g(x,y) = \omega \ * f(x,y) = \sum_{dx=-a}^{a} \sum_{dy=-b}^{b} \omega(dx,dy)\, f(x+dx, y+dy) \tag{1.5}$$

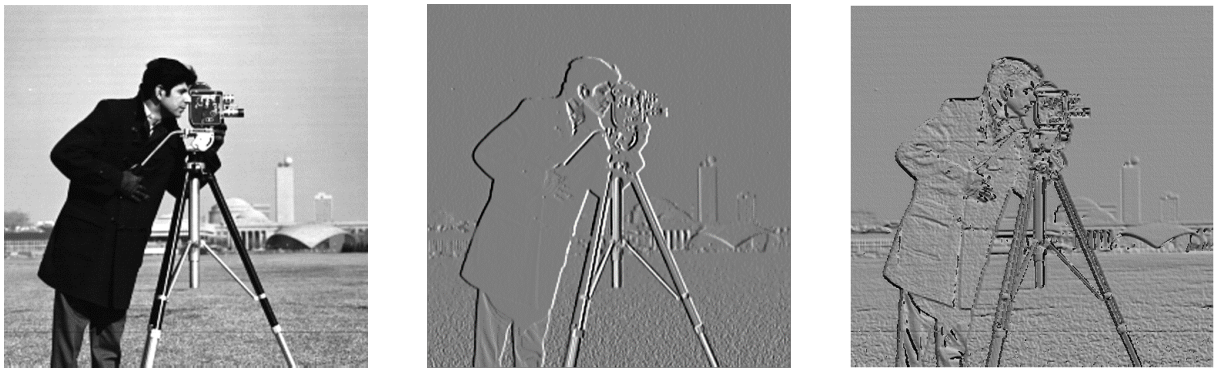
Gradien et laplcian

— **Le gradient d'une image :** Comme une image numérique n'est pas une fonction continue, la notion de dérivée n'est pas formellement définie et on utilisera un analogue appelé *gradient*.

Comme une image a 2 dimensions, le gradient de l'image f notee ∇f est une image vectorielle donnée par les deux dérivées partielles :

$$\nabla f = \left(\frac{\partial f}{\partial x} \ , \ \frac{\partial f}{\partial y} \right) \tag{1.6}$$

On peut observer une image et ses 2 dérivées partielles :



(a) Image original (b) Contours verticaux (c) Contours horizontaux

FIGURE 1.6 – Gradient de l'image camera [12].

On peut aussi regarder le vecteur du gradient sous sa forme polaire ($\| \nabla f \|$,$dir(\nabla f)$) avec :

— $\| \nabla f \|$ la norme du gradient :

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \tag{1.7}$$

— $dir(\nabla f$ la direction du gradient :

$$dir(\nabla f) = \text{atan2}\left(\frac{\partial f}{\partial y}, \frac{\partial f}{\partial x}\right) \tag{1.8}$$

Cela donne l'images suivante :

(a) Image original (b) $\| \nabla f \|$ (c) $\mathrm{dir}(\nabla f)$

FIGURE 1.7 – Gradient de l'image camera (forme polaire)[12]

La norme du gradient correspond à la valeur de la dérivée, et la direction du gradient permet de connecter la direction verticale correspondant au contour.

— **Le Laplacien d'une image :**

Le Laplacien est un opérateur du second ordre, pour une image d'intensité I(x, y), il est défini par :

$$\Delta^2 I\,(x,y) = \frac{\partial^2 I(x,y)}{\partial x^2} + \frac{\partial^2 I(x,y)}{\partial y^2} \tag{1.9}$$

Le Laplacien est fréquemment utilisé en amélioration d'images afin d'accentuer l'effet de contour :

- Invariant aux rotations de l'image.
- Sensibilité au bruit accrue par rapport au gradient.

Seuillage

Le seuillage est une technique simple, basée sur une mesure quantitative d'une grandeur, cette technique repose sur l'hypothèse que les objets peuvent être distingués par le niveau de gris. Un seuil optimal est celui qui permet de distinguer les objets entre eux ou différents objets du fond [8].

L'opération consiste à mettre à zéro tous les pixels ayant un niveau de gris inférieur à une certaine valeur (appelée seuil, en anglais treshold) et à la valeur maximale les pixels ayant une valeur supérieure.

L'image binaire est obtenue par la formule suivante :

$$I'_{(i,\ j)} = \begin{cases} 0 \ si \ I_{(i,\ j)} \ \leq Seuil \ , \ Seuil \ \in [0, 255] \\ 255 \ sinon \end{cases} \tag{1.10}$$

(a) Image original (b) Image binaire après seuillage

FIGURE 1.8 – Exemple d'un seuillage [13].

1.6.3 Segmentation en region

Segmentation en régions est une tâche importante dans le traitement d'images et la reconnaissance des formes. Elle est souvent considérée comme la première étape pour l'interprétation de l'image dans de nombreuses applications. Elle consiste à répartir les pixels d'une image de niveaux de gris en différentes classes en fonction de certaines propriétés.

Correspond aux algorithmes d'accroissement ou de découpage de région, l'accroissement de région est une méthode bottom-up : on part d'un ensemble de petites régions uniformes dans l'image (de la taille d'un ou de quelques pixels) et on regroupe les régions adjacentes de même couleur jusqu'à ce qu'aucun regroupement ne soit plus possible.

(a) Image original (b) Image après segmentation

FIGURE 1.9 – Le résultat de l'algorithme de segmentation en régions [13].

Plusieurs techniques de cette approche sont à distinguées :

Segmentation par croissance de régions (region-growing) :

C'est une méthode locale récursive qui a pour principe de faire croître une région avant de passer à la suivante, sans parcours particulier déterminé a priori (méthode par agrégation libre de pixels) [14].

On part d'un germe, il y a croissance suivant un critère de similarité jusqu'à atteindre le critère d'arrêt : par exemple convexité maximum, etc.

L'algorithme est globalement :
— choix des germes de régions
— intégration progressive des pixels voisins à chaque germe.

Les inconvénients de cet algorithme :
- Algorithme très sensible au bruit.
- Une mauvaise sélection des germes ou un choix du critère de similarité mal adapté peuvent entraîner des phénomènes de sous-segmentation ou de sur-segmentation.
- Il peut y avoir des pixels qui ne peuvent pas être classés.

Segmentation par division et fusion(Split and Merge) :

Ces méthodes combinent les deux méthodes décrites précédemment, la division de l'image en de petites régions homogènes, puis la fusion des régions connexes et similaires au sens d'un prédicat de regroupement. On part du principe que chaque pixel représente à lui seul une région. Deux régions seront fusionnées si elles répondent aux critères de

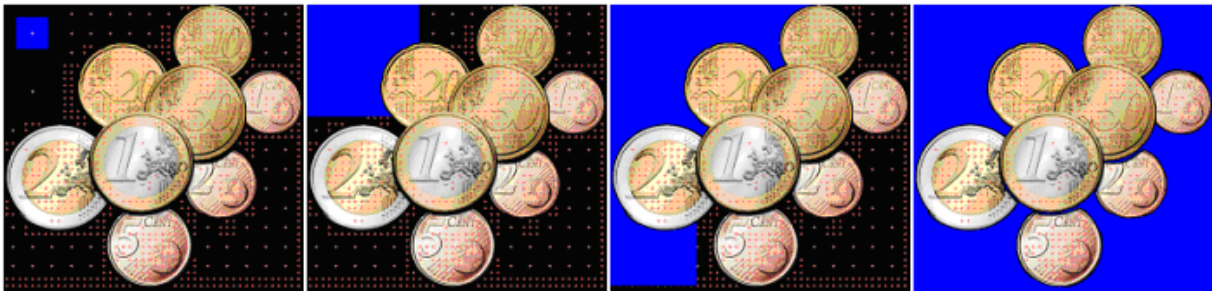


FIGURE 1.10 – Exemple de Segmentation par croissance de régions (region-growing)

similarité des niveaux de gris et d'adjacence de régions .On s'arrête quand le critère de fusion n'est plus vérifié [15].

La Division : On divise jusqu'à ce que la propriété soit vraie dans la sous-image.

La Fusion : On regroupe les régions adjacentes dont l'union vérifie la propriété de division et fusion (Split and merge).

Les inconvénients de cette méthode se situent à trois niveaux :

- Les régions obtenues ne correspondent pas, dans tous les cas, aux objets représentés dans l'image.
- Les limites des régions obtenues sont habituellement imprécises et ne coïncident pas exactement aux limites des objets de l'image.
- La difficulté d'identifier les critères pour agréger les pixels ou pour fusionner et diviser les régions

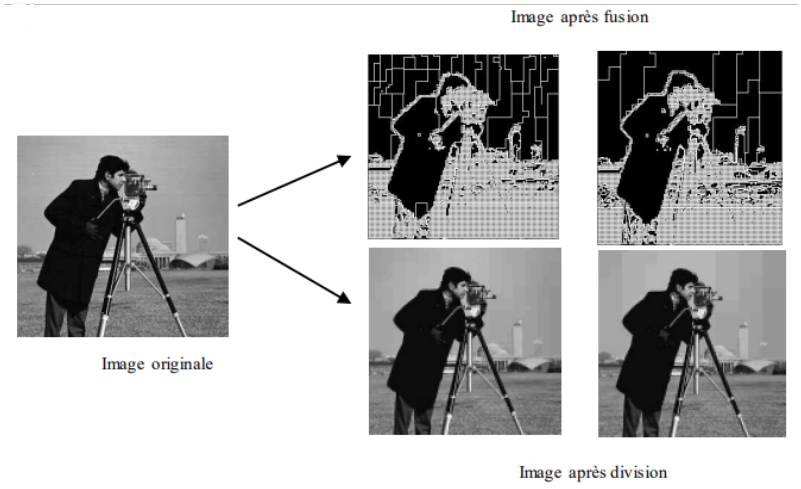


FIGURE 1.11 – Segmentation par division et fusion.

1.7 Conclusion

Nous avons introduit dans ce chapitre les notions de base de différentes techniques de traitement d'images. plusieurs méthodes de traitement ont été présenté nous avons présenté quelques-unes qui nous semblent les plus courantes dans le processus du traitement d'images.

Les pré-traitements d'images permettant d'améliorer la qualité de l'image, nous avons présenté un état de l'art des méthodes directes de la segmentation d'images les plus connues ainsi que les mesures d'évaluation des résultats notamment celles de la segmentation en contours. L'ensemble des techniques décrit dans les sections précédentes montre que le problème de la segmentation est bien un problème difficile. Ce qui a poussé les chercheurs à explorer de nouvelles perspectives en introduisant les méta-heuristiques d'optimisation, appliquées seules ou hybridées, dans l'espoir d'arriver à une segmentation la plus optimale possible.

Dans le prochain chapitre, nous allons présenter le Deep Learning plus en détail, et ses différentes méthodes d'apprentissage.

Chapitre 2

Deep Learning

2.1 Introduction

Dans le cadre de développer un algorithme capable d'automatiser la détection des tumeurs dans les images médicales en utilisant l'imagerie médicale, nous devons introduire un point important, qui est le Deep Learning.

Dans ce chapitre nous parlerons d'abord de Deep Learning avec quelques définitions, ensuite nous présenterons les réseaux de neurones, la correspondance entre neurone biologique et neurone artificiel, et nous détaillerons à la fin les différents types de Deep Learning.

2.2 Définition

Le Deep Learning ou « apprentissage profond » en français, est une technique de Machine Learning. Les systèmes d'apprentissage profond utilisent de nombreuses couches de neurones, capables d'interagir entre elles pour analyser des données non structurées et prédire des résultats. Ces systèmes sont capables d'apprendre et de s'améliorer de manière autonome. En effet, dès lors qu'une couche constate une erreur, les données sont alors systématiquement éliminées et renvoyées vers les précédents niveaux de couches afin de rectifier le modèle mathématique. Par la suite, lorsque ce modèle sera confronté à ces nouvelles données, il sera capable de les assimiler et de les dissocier sans que personne ne lui indique [16].

2.3 Réseaux de neurones

Les réseaux de neurones artificiels permettent de simuler d'une façon formelle le travail du cerveau humain, Les scientifiques ont découvert presque la façon dont le travail du cerveau humain en matière d'évolutivité et la probabilité de la mémoire d'apprentissage et la capacité de distinguer les objets et la capacité de prendre des décisions et comme nous le savons, le cerveau est constitué de milliards de neurones inter-connectés entre eux d'une manière très complexe par les cellules neuronales, ce qui forme un énorme réseau de neurones associés les uns aux autres.

Cette corrélation entre les cellules nerveuses donne à ces dernières la capacité de stocker et fournir des informations, des images, audio et succession de signaux qui reçoivent à travers les différents neurones, les réseaux de neurones permettent également d'apprendre par la répétition et l'erreur.

Les travaux de Hopfield en 1982 ont montrés que des réseaux de neurones artificiels étaient capables de résoudre des problèmes d'optimisation et ceux de Kohonen (1982) ont montré qu'ils étaient capables de résoudre des tâches de classification et de reconnaissance [17].

2.3.1 Neurone biologique

Le neurone est une cellule composée d'un corps cellulaire et d'un noyau. Le corps cellulaire se ramifie pour former ce que l'on nomme les dendrites, celles-ci sont parfois assez nombreuses que l'on parle alors de chevelure dendritique ou d'arborisation dendritique.C'est par les dendrites que l'information est acheminée de l'extérieur vers le soma (corps du neurone). L'information traitée par le neurone est propagée ensuite le long de l'axone (unique) pour être transmise aux autres neurones. La transmission entre deux neurones n'est pas directe. En fait, il existe un espace intercellulaire de quelques dizaines d'angströms (10-9m) entre l'axone du neurone afférent et les dendrites (on dit une dendrite) du neurone différent [18].

2.3.2 Neurone artificiel

Un neurone artificiel (formel) est un processeur élémentaire stimulé par des neurones qui le précèdent, qu'on appellera ses entrées, à chacune de ces entrées est associé un poids représentatif de la force de la connexion W et, en fonction de cette stimulation, on lui attribue une valeur qu'on appellera sa sortie.Qui se ramifie ensuite pour alimenter un nombre variable de neurones avals [18].

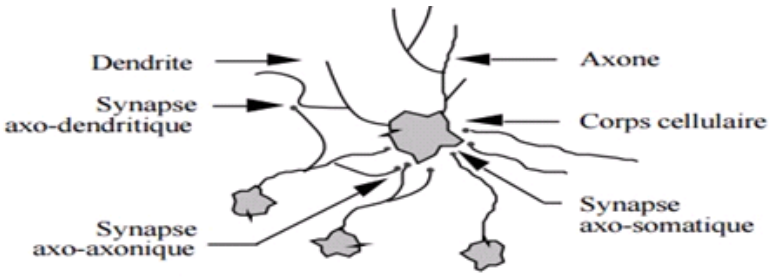


FIGURE 2.1 – Structure d'un neurone biologique [18].

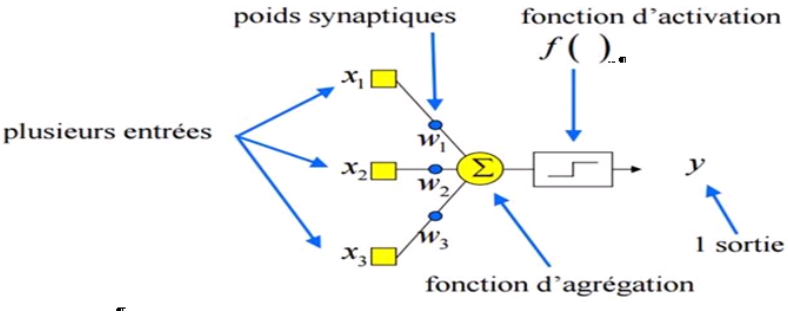


FIGURE 2.2 – Structure d'un neurone formel [18].

2.3.3 Correspondance entre neurone biologique et neurone artificiel

La structure d'un neurone artificiel est en fait inspirée de la structure des neurones biologiques. Les principales structures biologiques des neurones ont toutes leurs équivalents artificiels, ceci ayant pour but de reproduire leur fonctionnement de la meilleure façon possible (dune manière logique, simple et facilement représentable sur informatique).

NEURONE BIOLOGIQUE	**NEURONE ARTIFICIAL**
SYNAPSES	POIDS DE CONNEXIONS
AXONES	SIGNAL DE SORTIE
DENTRIDE	SIGNAL D'ENTRÉE
SOMMA	FONCTION D'ACTIVATION

TABLE 2.1 – Transition entre le neurone biologique et le neurone artificial

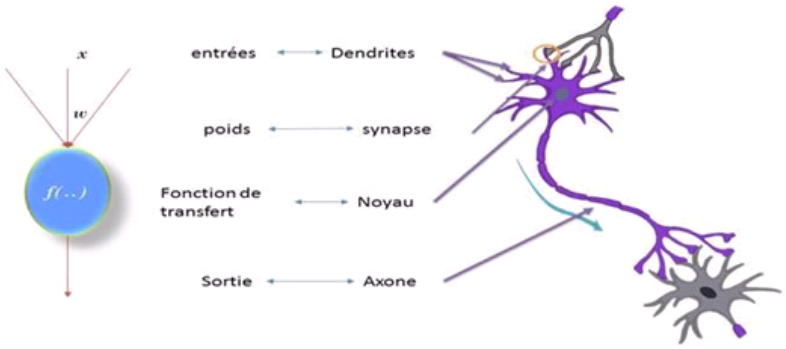


FIGURE 2.3 – Correspondance entre neurone artificiel et neurone biologique.

2.3.4 Comportement de neurone artificiel

— **Entrées du neurone « X » :**

Elles proviennent soit d'autres éléments processeurs (neurones), soit de l'environnement.

— **Le poids « W » (coefficient synaptique) :**

Est une valeur numérique associée à une connexion entre deux unités (neurones) qui reflète la force de relation (connexion) entre ces deux unités i et j, et il est noté par Wi.

— **La fonction d'agrégation (combinaison) « P » :**

Elle combine les entrées et les poids en calculant l'influence de chaque entrée en tenant en compte de son poids. Cette influence est calculée via la formule Suivante :

$$\sum W_i X_i \tag{2.1}$$

Où W est le poids de la connexion à l'entrée i. X : est le signal de l'entrée i.

— **La fonction de transfert (d'activation) :**

La fonction d'activation (la fonction de transfert) joue un rôle très important dans le comportement du neurone. Elle retourne une valeur représentative de l'activation du neurone, cette fonction a comme paramètre la somme pondérée des entrées ainsi que le seuil d'activation. Elle calcule la valeur de sortie à partir du résultat de la fonction de combinaison : $S = F(P)$.

Où :

S : est la valeur de sortie, F : est la fonction de transfert. La nature de cette fonction diffère selon le réseau,avec leurs équations mathématiques. Sachant que la différence avec les neurones biologiques est que l'état de ces derniers est

binaire, par contre la plupart des fonctions de transfert sont continuées et offrant une infinité de valeurs comprises dans l'intervalle [0, +1] ou [-1, +1].[19]

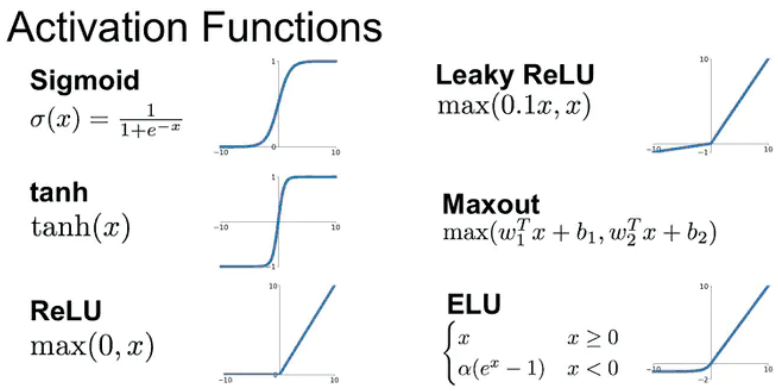


FIGURE 2.4 – Différents function d'activation.

2.4 Algorithmes de Deep Learning

Le deep learning possède de nombreux algorithmes, chacun spécialisé dans un domaine d'application spécifique , en va présenter quelques-uns en se concentre sur notre objectif principal la détection de brain-tumor dans l'image.

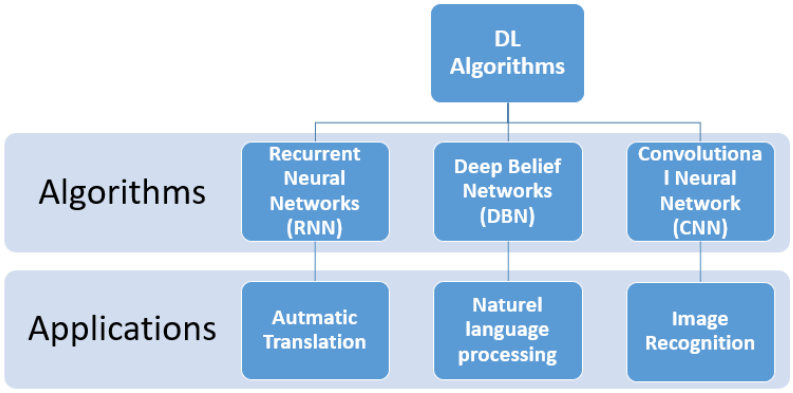


FIGURE 2.5 – Algorithmes de Deep Learning et leur applications.

2.4.1 Recurrent Neural Networks (RNN)

Un réseau neuronal récurrent (RNN) est une classe de réseaux neuronaux artificiels où les connexions entre les nœuds forment un graphe dirigé le long d'une séquence temporelle. Cela lui permet de présenter un comportement dynamique temporel. Dérivés de réseaux

de neurones à action directe, les RNN peuvent utiliser leur état interne (mémoire) pour traiter des séquences d'entrées de longueur variable [20].

Les réseaux récurrents permettent de traiter des données séquentielles une suite d'entrées x. En effet au temps t ils calculent leur sortie en fonction de l'entrée x(t) mais aussi de l'état de la couche cachée au temps précédent.

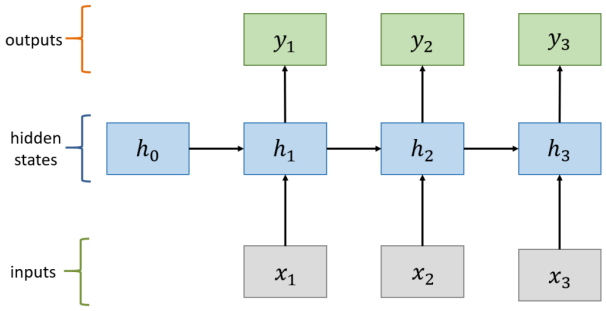


FIGURE 2.6 – Architecteur de RNN standard.

2.4.2 Convolutional Neural Network (ConvNet/CNN)

Les réseaux neuronaux convolutifs ont de larges applications dans la reconnaissance d'image et vidéo, les systèmes de recommandation et le traitement du langage naturel,sont quelques-uns des domaines dans lesquels les CNN sont largement utilisés [21].

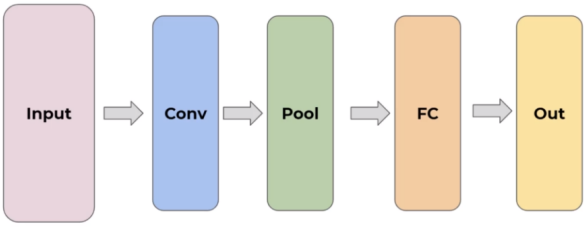


FIGURE 2.7 – Architecteur de CNN standard.

puisque notre objectif principal de cette étude est de détecter une tumeur cérébrale dans des images médicales, ce désigne le domaine de la vision par ordinateur qui est le domaine de spécialité des algorithmes CNN donc on a besoin de mieux les comprendre.

Comprendre le fonctionnement de Convolutional Neural Network :

Les CNN sont des architecture spécifique de réseaux de neurones extrêmement efficaces pour traiter les données d'image.

Les classifications d'images CNN prennent une image d'entrée, la traitent et la classent sous certains catégories(Par exemple. Chien, Chat, Tigre, Lion), Les ordinateurs voient une image d'entrée comme un tableau de pixels et cela dépend de la résolution de l'image. En fonction de la résolution de l'image, il verra $h * w * d$ (h = hauteur,w = largeur, d = dimension). Par exemple. Une image de matrice 6 x 6 x 3 de RVB (3 se réfère aux valeurs RVB) et une image de tableau 4 x 4 x 1 de matrice de niveaux de gris image [22].

On ne peut pas parler de convolutional layers sans parler des couches de pooling, le pooling est processus important dans un réseau convolutif, il extrait les valeurs importantes du pixels et permet de réduire la taille d'une image (le nombre de données traitées diminue et donc le temps de calcul sera également réduit) tout en préservant les caractéristiques [23].

Pooling : également connu sous le nom de subsamling ou downsamling ,est un processus simple où nous réduisons la taille ou la dimensionnalité de feature map , la proposition de cette réduction est de réduire le nombre de paramètres nécessaires au train , tout en conservant les caractéristiques les plus importantes.

il y a 2 types de Pooling les plus important que nous pouvons appliquer :

— **Max pooling** :renvoie la valeur maximale de la partie de l'image couverte par le filtre.

— **Averge pooling** :renvoie la moyenne de toutes les valeurs de la partie de l'image couverte par le filtre.

Ce vecteur en sortie de la partie convolutive est connecté à l'entrée du deuxième partie qui est composée de fully connected layers (Dense layers). Le rôle de ce La partie consiste à combiner les caractéristiques de la sortie de la première partie pour classer l'image [23].

La sortie de la deuxième partie est une dernière couche comprenant un neurone pour chaque catégorie.

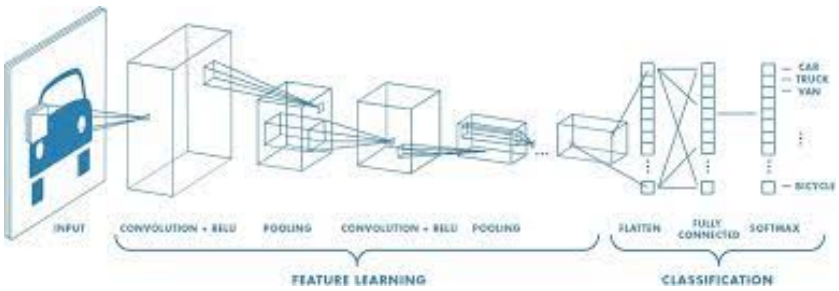


FIGURE 2.8 – Architecteur de CNN standard.

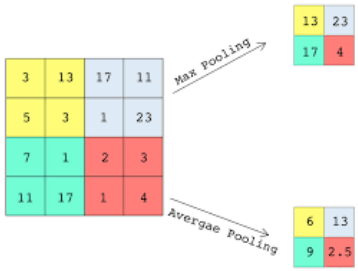


FIGURE 2.9 – résultats de max et average pooling avec un filtre $2 * 2$.

2.5 Conclusion

Dans ce chapitre nous avons présenté le Deep Learning ainsi que le concept de fonctionnement des réseaux de neurones. Nous avons également présenté les déférents algorithmes de le Deep Learning et les types des réseaux de neurones comme le CNNs. D'aprés l'étude de Deep Learning on peux conclure que cette technique est un bon outil qui nous permet de résoudre notre problème sur la segmentation des tumeurs cérébrales dans les images médicaux.

Dans le prochain chapitre, nous allons présenter quelques architecteures de reseaux de norones convulutionel pour la segmentation des image medicaux.

Chapitre 3

Les Architectures proposées pour la Segmentation d'images Biomédicales

3.1 Introduction

Après avoir vue le Deep Learning, les réseaux de norones, le CNNs dans le chapitre précédent. nous allons voir comment faire la segmentation des image médicales afin d'automatiser le processus de détection des tumeurs cérébrales. Nous présenterons donc les algorithmes ou méthodes utilisés pour réaliser ce travaille ? Tout d'abord nous allons

décrire la problématique ensuite nous allons passer à la présentation de l'architecture globale de la solution utilisée et qui donne des bon résultat sur la segmentation des images médicales.

3.2 Description du problème

Dans le domaine médical, globalement l'interprétation et l'analyse manuelle des images 3D est une tâche longue et ardue, c'est pourquoi la segmentation des images 3D est nécessaire pour que les spécialistes puissent facilement prendre la décision finale.

Dans cette direction, le défi de segmentation multimodale des tumeurs cérébrales est organisé chaque année, afin de mettre en évidence des approches efficaces et d'indiquer la voie vers ce problème difficile.

3.3 Description de la solution proposée

Dans cette partie, nous apportons des réponses au problème sur lequel nous travaillons en analysons la solution proposée qui est segmentation des images médicale avec l'architecture U-Net et ResUNet.

3.3.1 Architecture U-Net :

U-Net développée par Zhengxin Zhang et al. [1] en 2015, était un CNN unique développé pour la segmentation d'images biomédicales,est maintenant devenu un réseau d'encodeur-décodeur très populaire pour la segmentation sémantique il a une architecture UP-Down unique qui a un chemin Contractuel et un chemin Expansif.

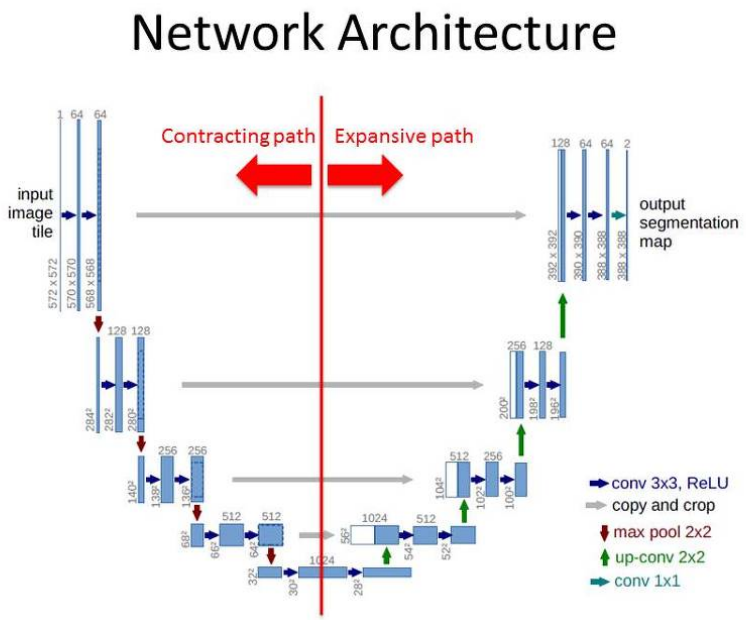


FIGURE 3.1 – Architecture U-Net

1. U-Net : Convolutional Networks for Biomedical Image Segmentation https ://arxiv.org/pdf/1505.04597.pdf

L'architecture du réseau [2] est illustrée dans la figure 3.1 Elle se compose d'un chemin de contraction (côté gauche) et d'un chemin expansif (côté droit).

Chemin Contractuel :

Le chemin contractuel suit l'architecture typique d'un réseau convolutif. Il consiste en l'application répétée de deux convolutions 3x3 (unpadded convolutions), chacune suivie d'une unité linéaire rectifiée (ReLU) et d'une opération de pooling 2x2 max avec foulée 2 pour le downsampling. À chaque étape de downsampling, nous doublons le nombre de canaux de fonctionnalités [24].

Chemin Expansif :

Chaque étape du chemin expansif consiste en un sur-échantillonnage de la carte des caractéristiques suivi d'une convolution 2x2 (convolution vers le haut) qui divise par deux le nombre des feature channels,une concaténation avec feature map rognée de manière correspondante à partir du chemin de contraction et deux 3x3 convolutions , chacun suivi d'un ReLU. Le recadrage est nécessaire en raison de la perte de pixels de bordure dans chaque convolution. À la dernière couche un 1x1 convolution est utilisé pour mapper chaque vecteur de caractéristiques à 64 composants au nombre de cours souhaité Au total, le réseau compte 23 couches convolutives permettre une carrelage sans couture de la sortie map segmentation il est important de sélectionner la taille de la tuile d'entrée de telle sorte que tous 2x2 max-pooling les opérations sont appliquées à une couche avec un même taille x et y [24].

Après avoir vue l'architercture globale de U-Net, nous allons expliquer la structure de cette architecture.

2. De la figure 3.1 Chaque case bleue correspond à un multi-channel feature map. Le nombre de channels est noté au dessus de la boite. La taille x-y est fournie sur le bord inférieur gauche de la boîte. Les cases blanches représentent cartes de caractéristiques. Les flèches indiquent les différentes opérations[24]

FIGURE 3.2 – Structure U-Net

La structure de U-Net :

1. Le chemin de downsampling dans U-Net se compose de 4 blocs avec la couche suivante
   — 3*3 CONV(RELU*Batch Normalization et Dropout used)
   — 3*3 CONV(RELU*Batch Normalization et Dropout used)
   — 2*2 Feature maps doubler au fur et à mesure que nous descendons les blocs à partir de 64, puis 128, 256 et 521
2. Bottleneck se compose de 2 CONV layers avec BAth Normalization Dropout
3. Le chemin up sampling se compose de 4 blocs avec les couches suivantes
   — des couches de déconvolution
   — concaténation avec feature map appertier du correspondant concating path
   — 3*3 CONV(RELU*Batch Normalization et Dropout used)
   — 3*3 CONV(RELU*Batch Normalization et Dropout used)

Les avantages de U-NET :

— En Deep Learning, un grand ensemble de données est nécessaire pour entraîner le modèle. En termes de temps, de budget et de ressources matérielles, il peut être difficile d'assembler une telle quantité de données pour résoudre le problème de classification des images.
— U-Net n'a pas besoin de plusieurs exécutions pour effectuer la segmentation d'images et peut apprendre avec très peu d'images étiquetées qui sont bien adaptées à la segmentation d'images en biologie ou en médecine. Les auteurs [3] ont testé leur architecture sur quelques défis de segmentation d'images et ont obtenu moins d'erreurs que les réseaux de neurones convolutifs classiques.

3. Les auteurs de cet article : https ://arxiv.org/pdf/1505.04597.pdf

3.3.2 Architecture ResUNet :

ResUNet fait référence à Deep Residual Unet. Il s'agit d'une architecture encodeur-décodeur développée par Zhengxin Zhang et al. [4] pour la segmentation sémantique. Il a été initialement utilisé pour l'extraction routière des images aériennes à haute résolution dans le domaine de l'analyse d'images de télédétection. Plus tard, il a été adopté par les chercheurs pour de nombreuses autres applications telles que la segmentation des polypes, la segmentation des tumeurs cérébrales, la segmentation des images humaines et bien d'autres.

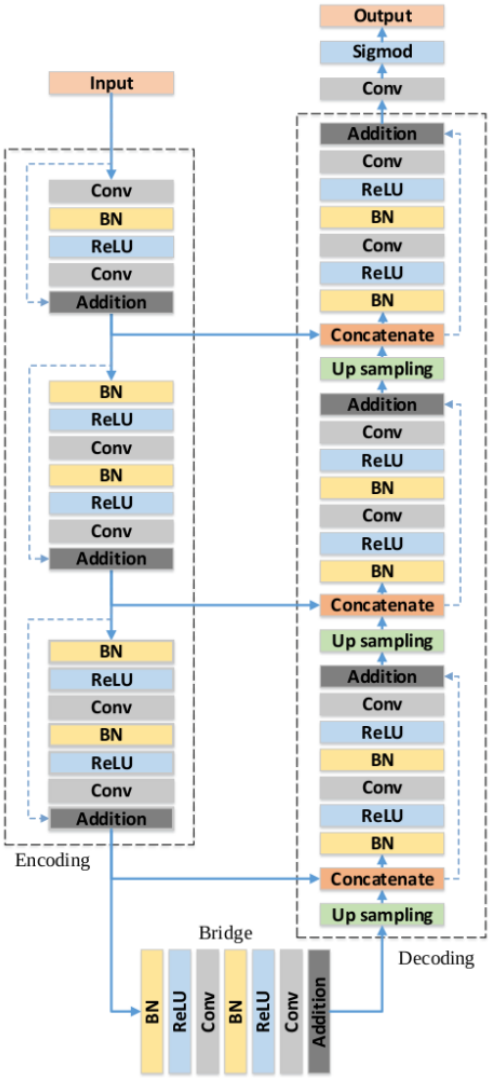


FIGURE 3.3 – L'architecture de ResUnet [25].

4. Road Extraction by Deep Residual U-Net https ://arxiv.org/pdf/1711.10684.pdf

ResUNet est un réseau de neurones entièrement convolutif conçu pour obtenir des performances élevées avec moins de paramètres. ResUNet tire parti à la fois de l'architecture UNet et du Deep Residual Learning.

La structure de ResUNet :

Le ResUNet se compose d'un réseau d'encodage, d'un réseau de décodage et d'un pont reliant ces deux réseaux, à la manière d'un U-Net. Le U-Net utilise deux convolutions 3 x 3, chacune étant suivie d'une fonction d'activation ReLU. Dans le cas de ResUNet, ces couches sont remplacées par un bloc résiduel pré-activé.

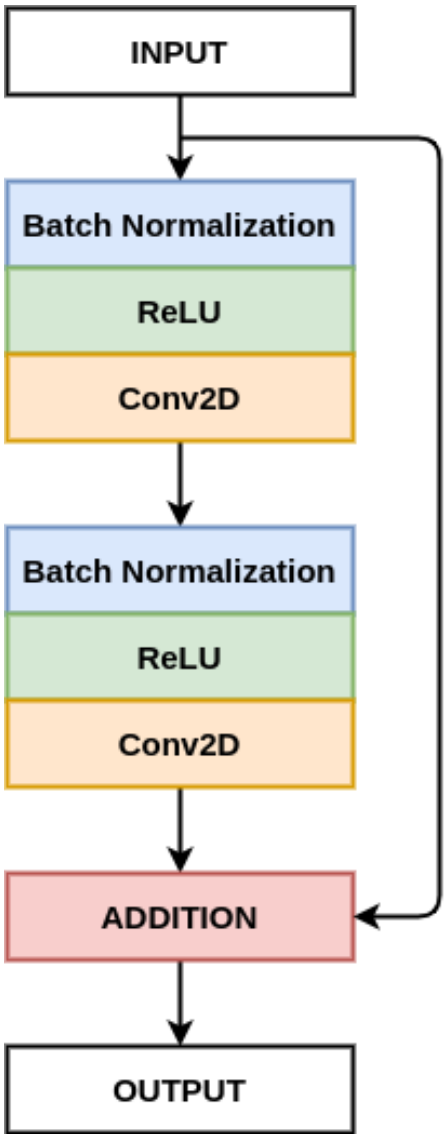


FIGURE 3.4 – Bloc Résiduel pré-activé [25]

■ **Encodeur(Encoder) :** L'encodeur prend l'image d'entrée et la fait passer à travers différents blocs d'encodeur, ce qui aide le réseau à apprendre une représentation abstraite. L'encodeur se compose de trois blocs d'encodeur, qui sont construits à l'aide du bloc résiduel pré-activé. La sortie de chaque bloc codeur agit comme une connexion de saut pour le bloc décodeur correspondant.

Pour réduire les dimensions spatiales (hauteur et largeur) des cartes de caractéristiques, la première couche de convolution 3×3 utilise une foulée de 2 dans le deuxième et le troisième bloc d'encodeur. Une valeur de foulée de 2 réduit les dimensions spatiales de moitié, soit 256 à 128 [26].

■ **Pont(Bridge) :** Le pont se compose également d'un bloc résiduel pré-activé avec une valeur de foulée de 2 [26].

■ **Décodeur(Decoder) :** Le décodeur prend la carte des caractéristiques(en anglais features map) du pont et les connexions de saut de différents blocs d'encodeur et apprend une meilleure représentation sémantique, qui est utilisée pour générer un masque de segmentation.

Le décodeur se compose de trois blocs de décodeur, et après chaque bloc, les dimensions spatiales de la carte de caractéristiques sont doubles et le nombre de canaux de caractéristiques est réduit.

Chaque bloc décodeur commence par un upsampling 2×2, qui double les dimensions spatiales des cartes de caractéristiques. Ensuite, ces cartes de caractéristiques sont ensuite concaténées avec la connexion de saut appropriée du bloc d'encodeur. Ces connexions de saut aident les blocs de décodeur à obtenir la fonctionnalité apprise par le réseau de codeurs. Après cela, les cartes de caractéristiques de l'opération de concaténation sont passées à travers un bloc résiduel pré-activé.

La sortie du dernier décodeur passe par une convolution 1×1 avec activation sigmoïde. La fonction d'activation sigmoïde donne le masque de segmentation représentant la classification au niveau des pixels [26].

Les avantages de ResUNet :

— L'utilisation de blocs résiduels permet de créer des grilles plus profondes sans se soucier de l'atténuation ou de l'explosion du gradient. Il contribue également à promouvoir la formation de réseaux.
— Les connexions de saut riches dans ResUNet facilitent un meilleur flux d'informations entre les différentes couches, ce qui contribue à améliorer les

gradients de flux pendant l'entraînement (back-propagation).

3.4 Conclusion

Dans ce chapitre, nous avons présenté la solution qui répond à la problématique de ce projet. Nous avons parlé des l'architectures qui nous permettent de résoudre notre problème de la segmentation des images médical, ainsi que le fonctionnement et la structure de ces architectures. Dans le chapitre suivant, on va présenter notre base de données, ensuite on décrivons nos modèles proposés.

Chapitre 4

Matériels et méthodes

4.1 Introduction

Le domaine de l'imagerie médicale a connu des changements révolutionnaires ces dernières années, en raison de l'avancement dans le domaine de Deep Learning, pour de nombreuses maladies, comme une tumeur au cerveau.

Segmentation des tumeurs cérébrales, application réussie de telles approches basées sur le Deep Learning lité, en particulier en ce qui concerne le test et la détection rapides de la maladie.

BraTS [1] s'est toujours concentré sur l'évaluation de méthodes de pointe pour la segmentation des tumeurs cérébrales en imagerie par résonance magnétique (IRM) multimodale. BraTS 2019 utilise des IRM préopératoires multi-institutionnelles, BraTS'19 se concentre également sur la prédiction de la survie globale des patients, via des analyses intégratives des caractéristiques radiomiques et des algorithmes d'apprentissage automatique. Enfin BraTS'19 entend évaluer expérimentalement l'incertitude dans les segmentations tumorales [27].

Ce chapitre sera divisé en deux sections, matériels et méthodes. Dans la première section nous parlerons de la base de données BraTS19, en mentionnant tous ses détails.

Après cela, nous passons à la section méthodes, où nous parlerons de nos modèles.

1. BraTS : https ://www.med.upenn.edu/cbica/brats2019/data.html

4.2 Matériels

4.2.1 Base des données

Dans ce travail, nous avons utilisé l'ensemble de données BraTS 2019 [2] pour les expériences, qui contient des images de 335 patients (259 HGG et 76 LGG),elle est disponible sous forme de fichiers NIfTI(.nii.gz).La dimension de la base de données et 240*240*155 (sagitall ,Coronal , Axial),chaque élément a 155 tranches sont en fait des images IRM.

Chaque patient a été scanné avec quatre séquences [3] :

— Flair (faild Attenuated Inversion Recovery)

— T1 séquence pondérée

— T1Ce séquence post-contraste

— T2 séquence pondérée

— segmentation mask

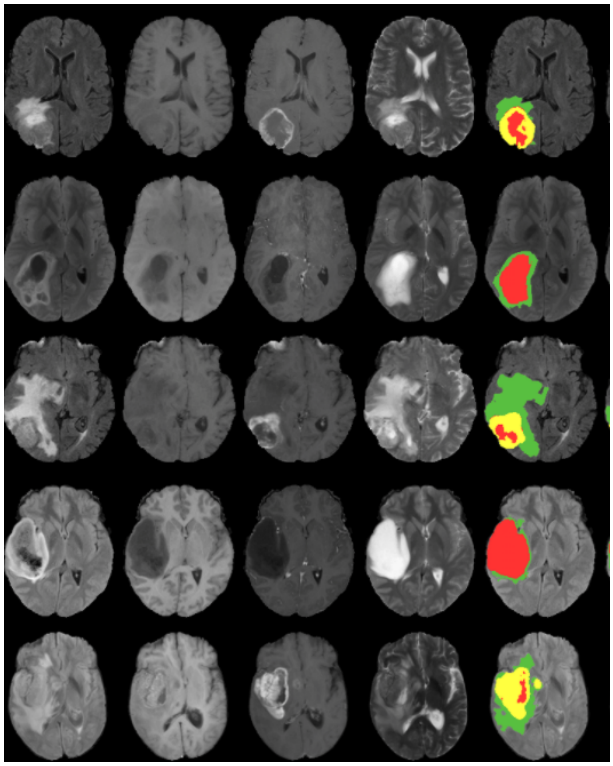


FIGURE 4.1 – Exemple sur l'ensemble de données BraTS 2019.

2. BraTS2019 : https ://drive.google.com/drive/folders/1tx5RIwtEyrEH$_r u2MDyHvEoR0F4VYn7v?usp$ = *sharing*

3. De la Figure 4.1, montrez la coupe axiale des images IRM en modalité Flair, T1, T1ce et T2, vérité terrain (GT). et enhancing tumor (jaune), edema (vert) et non-enhancing tumor (rouge).

Différence entre HGG et LGG ?

LGG se développe plus lentement et moins agressivement avec des degrés d'infiltration et de prolifération cellulaires inférieurs à ceux de HGG

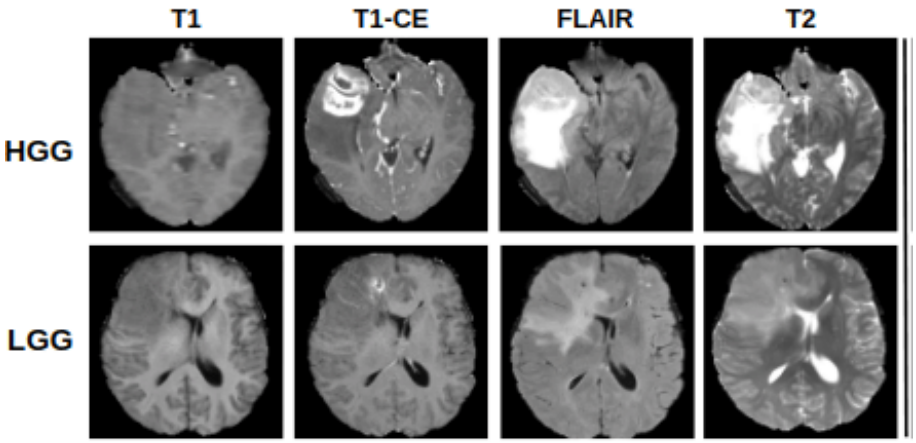


FIGURE 4.2 – Dataset BraTS 2019.

Les constituants des gliomes sont :

- Edema :collecte de liquide ou d'eau , est mieux vu dans FLAIR et Séquence pondérée T2
- Necrosis : Accumulation de cellules mortes
- Enhancing tumor :indiquer la rupture hémato-encéphalique,vu dans T1c
- Non Enhancing : régions de la région qui ne sont ni Edema, ni necrosis , enhancing tumor

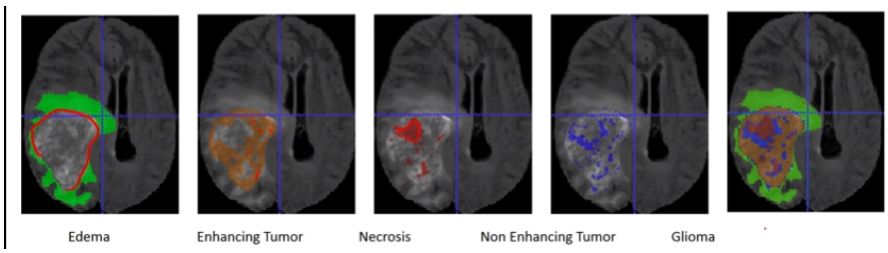


FIGURE 4.3 – Les constituants des gliomes.

4.2.2 Data préparation :

En raison de la consommation de RAM, les données ont été traitées de manière appropriée, Afin d'obtenir une analyse des résultats exacte et fiable, un algorithme de

sélection sera utilisé, nous essaierons de ne conserver que les analyses où les points d'infection peuvent apparaître.

1. Tout d'abord, les dossiers HGG et LGG ont été traités un par un pour extraire le volume 3D de dimension (N,155,240,240,4). Maintenant, HGG contient 259 scans de patients et LGG contient 76 scans de patients.
2. Maintenant, chaque donnée a été pré-traités un par un. Maintenant, comme toutes les 155 tranches ne montrent pas la région tumorale, ici seule la partie médiane, c'est-à-dire de la 30ème à la 120ème tranche, a été prise pour créer les données finales, et enfin toutes ont été remodelées en (N1, 240, 240,4) pour les données et (N1, 240, 240,4) pour la vérité terrain(ground truth) en utilisant un codage one-hot.
   Ici, N1 = 90X259 = 23 310 pour HGG, N1 = 90X76 = 6,840 pour LGG.
   Remarquer :23 310 est un grand nombre de données et nécessite beaucoup de RAM, donc dans ce travail, nous allons travailler avec les données LGG = 6840.
3. Ensuite, chaque donnée est recadrée au centre avec la dimension finale de (N1,192,192,4).
4. Enfin, les données ont été divisées au hasard en données d'entraînement, de validation et de test avec respectivement 60%, 20%,20%

Data	Nombre d'images
Training set	4104
Test set	1368
Validation set	1368
Total	6840

TABLE 4.1 – Représentation du Dataset.

4.3 Méthode

Dans le chapitre trois, nous avons parlé de l'architecture qui nous permet de résoudre notre problème de segmentation d'images médicales, et proposé l'architecture de UNet et ResUNet, qui sont basées sur l'algorithme CNN.

Avant de parler de nos modèles et ses descriptions, nous parlerons des techniques que nous avons utilisées pour obtenir les meilleurs résultats :

1. **Optimiseur Adam** [4] **:** signifie Adaptive Moment Estimation. Il calcule également différents taux d'apprentissage. Adam fonctionne bien dans la pratique, est plus rapide et surpasse les autres techniques [28].
2. **Dice Loss : Fonction de perte utilisé** La formule originale de Sørensen était destinée à être appliquée à des données discrètes . Il a été présenté à la communauté de vision par ordinateur par Fausto Milletari, Nassir Navab et Seyed Ahmad Ahmadi.en 2016 pour la segmentation d'images médicales 3D [29].

$$D = \frac{2 \sum_i^N p_i g_i}{\sum_i^N {p_i}^2 + \sum_i^N {g_i}^2} \tag{4.1}$$

   L'équation du coefficient Dice, dans laquelle pi et gi représentent des paires de valeurs de pixels correspondantes de prédiction et ground truth, respectivement. Dans le scénario de détection de limite, les valeurs de pi et gi sont soit 0, soit 1, représentant si le pixel est limite (valeur de 1) ou non (valeur de 0). Par conséquent, le dénominateur est la somme des pixels limites totaux de la prédiction et de la vérité terrain, et le numérateur est la somme des pixels limites correctement prédits car la somme n'augmente que lorsque pi et gi correspondent (tous deux de valeur 1).
3. **Fonctions d'activation :**
   — **ReLU : La fonction d'activation linéaire rectifiée** ReLU signifie unité d'activation linéaire rectifiée et est considéré comme l'un des rares jalons de la révolution de l'apprentissage en profondeur. C'est simple mais vraiment meilleur que les fonctions d'activation de ses prédécesseurs telles que sigmoïde ou tanh. Formule de la fonction d'activation ReLU :

$$f(x) = max(0, x) \tag{4.2}$$

   — **Softmax :** Softmax est une fonction d'activation très intéressante car elle mappe non seulement notre sortie sur une plage [0,1], mais mappe également chaque sortie de manière à ce que la somme totale soit 1. La sortie de Softmax est donc une distribution de probabilité. La fonction softmax est souvent utilisée dans la couche finale d'un classificateur basé sur un réseau de

4. Adam optimiseur original article : https ://arxiv.org/abs/1412.6980

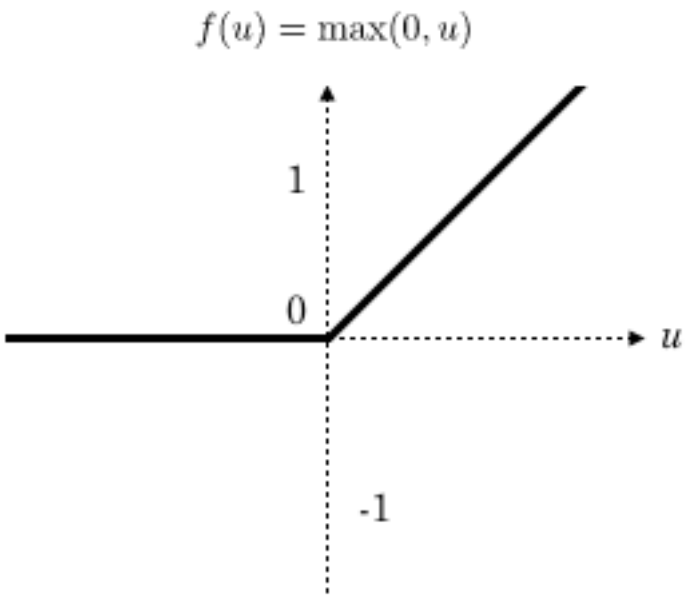


FIGURE 4.4 – ReLU : La fonction d'activation linéaire rectifiée.

neurones. De tels réseaux sont généralement formés sous un régime de perte logarithmique (ou d'entropie croisée), donnant une variante non linéaire de la régression logistique multinomiale [30]. Formule de la fonction d'activation Softmax :

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{4.3}$$

— **Sigmoid :** La courbe de fonction d'activation sigmoïde (ou logistique) ressemble à une forme de S. Le principal avantage du sigmoïde est que la sortie est toujours comprise entre 0 et 1.

Formule de la fonction d'activation Sigmoid :

$$y = \frac{1}{1 + e^{-x}} \tag{4.4}$$

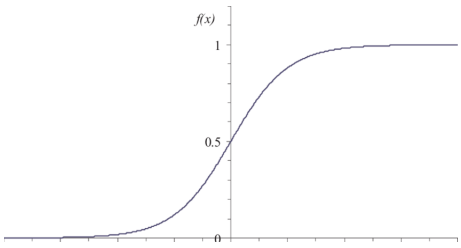


FIGURE 4.5 – Sigmoid : La fonction d'activation

4.3.1 Modèle UNet :

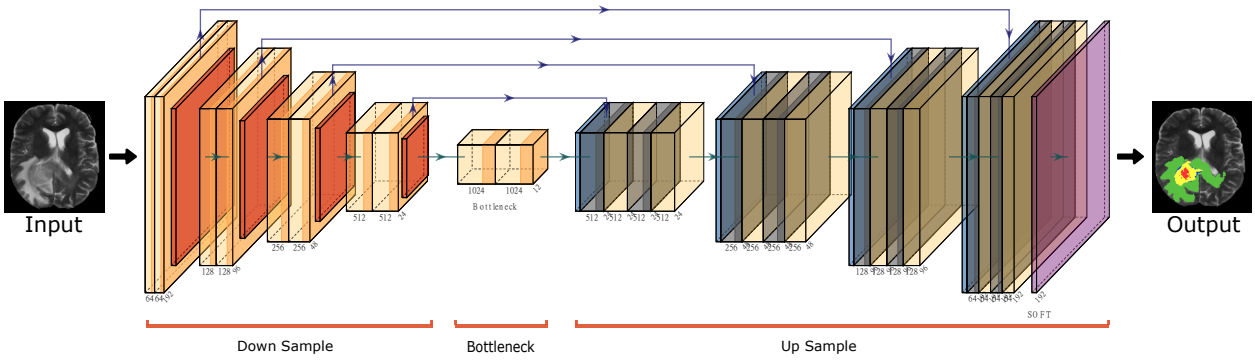


FIGURE 4.6 – L'architecure du Modèle U-Net.

La Description du modèle

Comme nous le savons, UNet composée de trois chemins importants : Downsampling,Bottleneck et Upsampling. Dans notre modèle, nous avons fait un seul changement, qui est de changer la forme d'entrée de 512 à 192.

1. **Forme d'entrée(Input shape) :** Forme d'entrée avec (192,192,4)
2. **Downsampling :** ce chemin est composé de 4 blocs avec input shape de (192,192,4) les couche suivante :
   — 3*3 Conv(ReLU*Batch Normalization et Dropout used)
   — 3*3 Conv(ReLU*Batch Normalization et Dropout used)
   — 2*2 Feature maps doubler au fur et à mesure que nous descendons les blocs àpartir de 64, puis 128, 256 et 521.
3. **Bottleneck** se compose de 2 Conv layers avec Batch Normalization Dropout.
4. **Upsampling :** ce chemin est composé de 4 blocs avec les couches suivantes :
   — des couches de déconvolution
   — concaténation avec feature map appertier du correspondant concating path
   — 3*3 Conv(ReLU*Batch Normalization et Dropout used)
   — 3*3 Conv(ReLU*Batch Normalization et Dropout used)
5. **Sortir(Output) :** couche de convolution 1*1 avec fonction d'activation Softmax.

4.3.2 Modèle ResUNet :

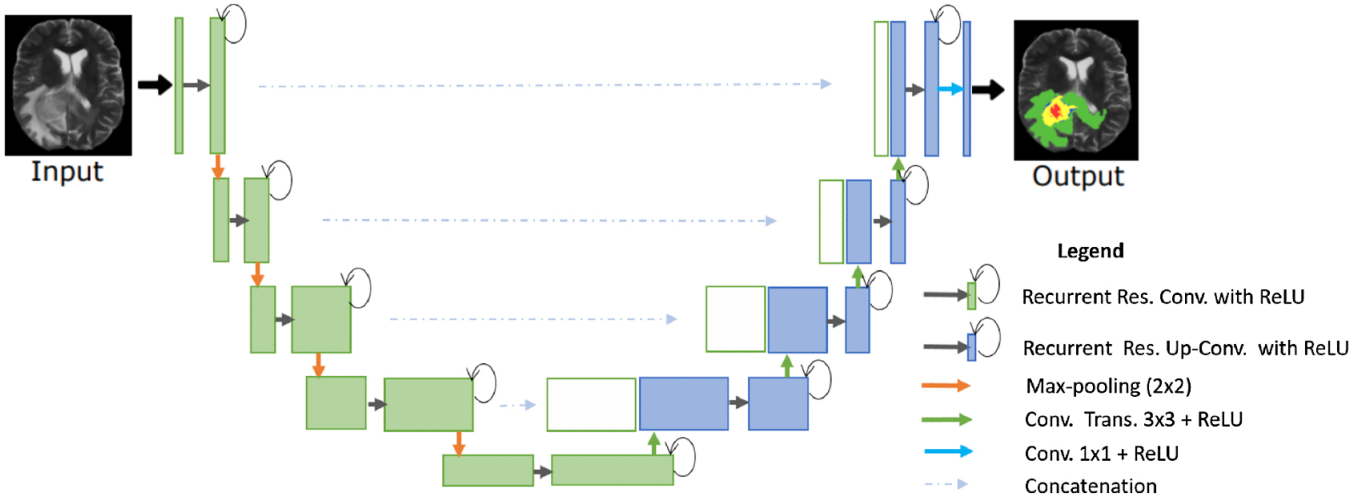


FIGURE 4.7 – L'architecure du Modèle ResUNet.

La Description du modèle

Dans ResUNet, nous collectons les composants de chaque bloc UNet dans un bloc résiduel,la forme d'entrée toujours la même (192,192,4).

1. **Forme d'entrée(Input shape) :** Forme d'entrée avec (192,192,4)
2. **Encodeur :** ce chemin est composé de 4 blocs résidue, et chaque sortie de bloc agit comme une connexion de saut pour le bloc décodeur correspondant :
   — 3*3 Conv(Trans + ReLU)
   — 3*3 Conv(Trans + ReLU) + Recurrent
   — 2*2 Feature maps doubler au fur et à mesure que nous descendons les blocs àpartir de 64, puis 128, 256 et 521.
3. **Bottleneck** Le pont se compose également d'un bloc résiduel pré-activé avec une valeur de foulée de 2.
4. **Decoder :** ce chemin est composé de 4 blocs résidue.
   — Upsampling(2*2)

— concaténation avec feature map appertier du correspondant concating path
— 3*3 Conv(Trans + ReLU)
— 3*3 Conv(Trans + ReLU) + Recurrent

5. **Sortir(Output) :** couche de convolution 1*1 avec fonction d'activation Sigmoid.

4.4 Conclusion

Dans ce chapitre, nous avons discuté la base de donnée que nous avons utilisé dans notre travail. Puis nous avons évoqué la phase de préparation de ce dernier. Au final, nous avons proposé des solutions pour étendre notre objectif. Les résultats et les outils de cette mise en œuvre seront présentés dans le chapitre suivant.

Chapitre 5

Réalisation

5.1 Introduction

Dans ce chapitre, nous parlerons des différentes bibliothèques utilisées, de l'environnement et du langage de programmation choisi pour implémenter notre modèle.Ensuite, nous décrivons l'implmentation de notre modèle et présentons son entraînement sur l'ensemble de données, après cela dans la phase de test et de prédiction, nous évaluerons nos modèles formés. À la fin, nous discuterons les résultats obtenus.

5.2 Les outils de développement

5.2.1 Langage de programmation

Le langage de programmation choisi pour notre méthode présentée précédemment s'est concentré sur le langage python

Python :

Python [1]est un langage de programmation puissant et facile à apprendre. Il a des données de haut niveau structures et permet une approche simple mais efficace de la programmation orientée objet.

Python est un langage de programmation interprété, un langage de script et est un peu lent par rapport à d'autres langages compilés en C ou C++ pour faire des calculs. Offres Python plusieurs bibliothèques (packages) pour le traitement des données, les calculs matriciels, l'analyse et les données visualisation. Pour le Deep Learning, python a des environnements de travail comme caffe, tensorow, keras, pytorch. Ces environnements de travail sont très utiles car ils permettent de gérer l'algorithme de rétropropagation

1. https ://www.python.org/

pour les grands réseaux de neurones, les CNN, les U-net, etc. Ils assurent également la parallélisation des calculs sur le GPU.

Le choix de ce langage présente les avantages suivants :

— C'est totalement gratuit.

— Il est facile à apprendre, lire, comprendre, utiliser et écrire.

— Il fonctionne sur tous les principaux systèmes d'exploitation et plates-formes informatiques.

5.2.2 Environnement de développement

Tout entraînement du réseau de neurones nécessite une forte puissance de calcul. Notre méthode est basée sur un réseau convolutif profond, qui implique un grand nombre de points à entraîner. La formation sur le PC portable aurait également pris beaucoup de temps. Nous avons utilisé Google Colab et Amazon Sagemaker studio.

■ **Google Colab :** Google Colaboratory ou Colab, un outil Google simple et gratuit pour vous initier au Deep Learning ou collaborer avec vos collègues sur des projets en science des données [31].

Colab permet :

— de développer des applications en Deep Learning en utilisant des bibliothèques Python populaires telles que Keras, TensorFlow, PyTorch et OpenCV.

— Il est facile à apprendre, lire, comprendre, utiliser et écrire.

— d'utiliser un environnement de développement (Jupyter Notebook) qui ne nécessite aucune configuration. Mais la fonctionnalité qui distingue Colab des autres services est l'accès à un processeur graphique GPU, totalement gratuitement.

■ **Amazon Sagemaker studio :**

Amazon SageMaker est un service entièrement géré permettant aux développeurs et aux spécialistes des données de créer, former et déployer rapidement et facilement des modèles de machine learning [32].

5.2.3 Bibliothèques utilisées :

Plusieurs bibliothèques ont été utilisées. Dans notre travail, nous avons besoin des bibliothèques suivantes :

1. **TensorFlow :**

   TensorFlow [2] est une plate-forme open source de bout en bout pour l'apprentissage automatique. Il dispose d'un écosystème complet et flexible

2. https ://www.tensorflow.org/

d'outils, des bibliothèques et des ressources communautaires qui permet aux chercheurs de poussez l'état de l'art en matière de ML et les développeurs créent et déploient facilement des applications basées sur ML.

De nombreuses entreprises utilisent TensorFlow telles que Google, Twitter, Intel et Coca-Cola, il est très apprécié pour ses nombreux bienfaits que nous listons ci-dessous :

— Multi-platform (Linux, Mac OS, Windows and even Android and iOS).
— APIs in Python, C ++, Java.
— Documentation extrêmement bien fournie avec de nombreux exemples et tutoriels.

2. **Keras**

   Keras [3] est une API d'apprentissage en profondeur écrite en Python, s'exécutant au-dessus de l'apprentissage automatique plate-forme TensorFlow.

3. **SimpleITK**

   SimpleITK [4] est une interface open source simplifiée de la boîte à outils de segmentation et d'enregistrement Insight. La bibliothèque d'analyse d'images SimpleITK est disponible dans plusieurs langages de programmation, notamment C ++, Python, R...

4. **Matplotlib** Matplotlib [5] est une bibliothèque de suivi Python 2D qui produit des images de qualité publication dans divers formats papier et environnements interactifs sur toutes les plateformes.

5. **NumPy** NumPy [6] est une bibliothèque pour le langage de programmation Python, a destinée à manipuler des matrices ou tableaux multidimensionnels, ainsi qu'une grande collection des fonctions mathématiques de haut niveau pour opérer sur ces tableaux.

6. **Pandas** Pandas [7]est une analyse de données open source rapide, puissante, flexible et facile à utiliser.

7. **Scikit-learn**

   Scikit-learn [8] est une bibliothèque en Python qui fournit de nombreux algorithmes d'apprentissage. Il comprend des fonctions pour l'estimation de la régression logistique, des algorithmes de classification et des vecteurs de support Machines.

3. https ://keras.io/
4. https ://simpleitk.org/
5. https ://matplotlib.org/
6. https ://numpy.org/
7. https ://pandas.pydata.org/
8. https ://scikit-learn.org/

5.2.4 Métriques d'évaluation de la classification :

Les 4 métriques d'évaluation de classification que tout Data Sciences doit connaître : **Accuracy, F1-Score, Precision, et Recall :**

1. **Accuracy**

   Accuracy est la métrique de classification par excellence. C'est assez facile à comprendre. Et facilement adapté à un problème de classification binaire et multiclasse.

$$Accuracy = (TP + TN)/(TP + FP + FN + TN) \tag{5.1}$$

   - **TP** = vrais positifs.
   - **TN** = vrais négatifs.
   - **FP** = faux positifs.
   - **FN** = faux négatifs.

   Accuracy est la proportion de vrais résultats parmi le nombre total de cas examinés.

   Quand utiliser ?

   Accuracy est un choix valide d'évaluation pour les problèmes de classification qui sont bien équilibrés et non asymétriques ou aucun déséquilibre de classe

2. **F1-Score** Le score F, également appelé score F1, est une mesure de la précision d'un modèle sur un ensemble de données. Il est utilisé pour évaluer les systèmes de classification binaire, qui classent les exemples en «positif» ou «négatif» [33].

$$F1 - Score = 2 * (Precision * Recall)/(Precision + Recal) \tag{5.2}$$

   Quand utiliser ?

   Le F-score est couramment utilisé pour évaluer les systèmes de recherche d'informations, tels que les moteurs de recherche, et de nombreux types de modèles d'apprentissage automatique, en particulier dans le traitement du langage naturel [33].

3. **Precision**

Commençons par la précision, qui répond à la question suivante : quelle proportion de Positifs prédits est vraiment Positif ?

$$Precision = (TP)/(TP + FP \tag{5.3}$$

Quand utiliser ?

La précision est un choix valable de métrique d'évaluation lorsque nous voulons être très sûrs de notre prédiction. Par exemple : si nous construisons un système pour prédire si nous devons réduire la limite de crédit sur un compte particulier, nous voulons être très sûrs de notre prédiction ou cela peut entraîner le mécontentement du client.

4. **Recall**

   Une autre mesure très utile est le Recall, qui répond à une question différente : quelle proportion de Positifs réels est correctement classée ?

$$Recall = (TP)/(TP + FN) \tag{5.4}$$

Quand utiliser ?

Recall est un choix valable de métrique d'évaluation lorsque nous voulons capturer autant de points positifs que possible. Par exemple : si nous construisons un système pour prédire si une personne a un cancer ou non, nous voulons capturer la maladie même si nous ne sommes pas très sûrs.

5.3 Implémentation

Dans cette section, nous parlerons de la mise en œuvre de notre projet. Dans la première, nous décrirons l'algorithme de filtrage que nous avons utilisé pour nettoyer notre base de données et choisissez que de bonnes images.Ensuit en va décrire la création de nos modèles.

5.3.1 Algorithme de filtrage

Notre base de données contient de nombreuses images différentes pour chaque patient, ces images diffèrent d'un gliome haut bas à un gliome moins bas

L'implémentation de l'algorithme de filtrage est illustrée ci-dessous dans la figure 5.1

```
[4]: def load_data(path):
         my_dir = sorted(os.listdir(path))
         data = []
         gt = []
         for p in tqdm(my_dir):
             data_list = sorted(os.listdir(path+p))
             img_itk = sitk.ReadImage(path + p + '/'+ data_list[0])
             flair = sitk.GetArrayFromImage(img_itk)
             img_itk = sitk.ReadImage(path + p + '/'+ data_list[1])
             seg =  sitk.GetArrayFromImage(img_itk)
             img_itk = sitk.ReadImage(path + p + '/'+ data_list[2])
             t1 =  sitk.GetArrayFromImage(img_itk)
             img_itk = sitk.ReadImage(path + p + '/'+ data_list[3])
             t1ce =  sitk.GetArrayFromImage(img_itk)
             img_itk = sitk.ReadImage(path + p + '/'+ data_list[4])
             t2 =  sitk.GetArrayFromImage(img_itk)
             data.append([flair,t1,t1ce,t2])
             gt.append(seg)
         data = np.asarray(data,dtype=np.float32)
         gt = np.asarray(gt,dtype=np.uint8)
         return data,gt
```

FIGURE 5.1 – Algorithme de filtrage.

5.3.2 Modèles créés

Au cours de notre expérimentation, nous avons créé plusieurs modèles avec différentes architectures basées sur les approches proposées

L'architectures des modèles est illustrée ci-dessous :

Modèle ResUnet : L'architecture de ce modèle est très grande, nous l'avons donc divisée en cinq parties, les première et deuxième parties représentent down sampling et les parties restantes sont représentées up sampling comme indiqué dans le tableau ci-dessous :

Model: "Modèle ResUnet"

Layer (type)	Output Shape	Param #	Connected to
input_16 (InputLayer)	[(None, 192, 192, 4)	0	
lambda_1 (Lambda)	(None, 192, 192, 4)	0	input_16[0][0]
conv2d_157 (Conv2D)	(None, 192, 192, 64)	2368	lambda_1[0][0]
batch_normalization_150 (BatchN	(None, 192, 192, 64)	256	conv2d_157[0][0]
conv2d_159 (Conv2D)	(None, 192, 192, 64)	320	lambda_1[0][0]
activation_93 (Activation)	(None, 192, 192, 64)	0	batch_normalization_150[0][0]
batch_normalization_151 (BatchN	(None, 192, 192, 64)	256	conv2d_159[0][0]
conv2d_158 (Conv2D)	(None, 192, 192, 64)	36928	activation_93[0][0]
add_47 (Add)	(None, 192, 192, 64)	0	batch_normalization_151[0][0] conv2d_158[0][0]
batch_normalization_152 (BatchN	(None, 192, 192, 64)	256	add_47[0][0]
activation_94 (Activation)	(None, 192, 192, 64)	0	batch_normalization_152[0][0]
conv2d_160 (Conv2D)	(None, 96, 96, 128)	73856	activation_94[0][0]
batch_normalization_153 (BatchN	(None, 96, 96, 128)	512	conv2d_160[0][0]
conv2d_162 (Conv2D)	(None, 96, 96, 128)	8320	add_47[0][0]
activation_95 (Activation)	(None, 96, 96, 128)	0	batch_normalization_153[0][0]
batch_normalization_154 (BatchN	(None, 96, 96, 128)	512	conv2d_162[0][0]
conv2d_161 (Conv2D)	(None, 96, 96, 128)	147584	activation_95[0][0]

TABLE 5.1 – Partie 01 du modèle ResUNet.

add_48 (Add)	(None, 96, 96, 128)	0	batch_normalization_154[0][0] conv2d_161[0][0]
batch_normalization_155 (BatchN	(None, 96, 96, 128)	512	add_48[0][0]
activation_96 (Activation)	(None, 96, 96, 128)	0	batch_normalization_155[0][0]
conv2d_163 (Conv2D)	(None, 48, 48, 256)	295168	activation_96[0][0]
batch_normalization_156 (BatchN	(None, 48, 48, 256)	1024	conv2d_163[0][0]
conv2d_165 (Conv2D)	(None, 48, 48, 256)	33024	add_48[0][0]
activation_97 (Activation)	(None, 48, 48, 256)	0	batch_normalization_156[0][0]
batch_normalization_157 (BatchN	(None, 48, 48, 256)	1024	conv2d_165[0][0]
conv2d_164 (Conv2D)	(None, 48, 48, 256)	590080	activation_97[0][0]
add_49 (Add)	(None, 48, 48, 256)	0	batch_normalization_157[0][0] conv2d_164[0][0]
batch_normalization_158 (BatchN	(None, 48, 48, 256)	1024	add_49[0][0]
activation_98 (Activation)	(None, 48, 48, 256)	0	batch_normalization_158[0][0]
conv2d_166 (Conv2D)	(None, 24, 24, 512)	1180160	activation_98[0][0]
batch_normalization_159 (BatchN	(None, 24, 24, 512)	2048	conv2d_166[0][0]
conv2d_168 (Conv2D)	(None, 24, 24, 512)	131584	add_49[0][0]
activation_99 (Activation)	(None, 24, 24, 512)	0	batch_normalization_159[0][0]
batch_normalization_160 (BatchN	(None, 24, 24, 512)	2048	conv2d_168[0][0]
conv2d_167 (Conv2D)	(None, 24, 24, 512)	2359808	activation_99[0][0]

TABLE 5.2 – Partie 02 du modèle ResUNet.

add_50 (Add)	(None, 24, 24, 512)	0	batch_normalization_160[0][0] conv2d_167[0][0]
up_sampling2d_23 (UpSampling2D)	(None, 48, 48, 512)	0	add_50[0][0]
concatenate_23 (Concatenate)	(None, 48, 48, 768)	0	up_sampling2d_23[0][0] add_49[0][0]
batch_normalization_161 (BatchN	(None, 48, 48, 768)	3072	concatenate_23[0][0]
activation_100 (Activation)	(None, 48, 48, 768)	0	batch_normalization_161[0][0]
conv2d_169 (Conv2D)	(None, 48, 48, 256)	1769728	activation_100[0][0]
batch_normalization_162 (BatchN	(None, 48, 48, 256)	1024	conv2d_169[0][0]
conv2d_171 (Conv2D)	(None, 48, 48, 256)	196864	concatenate_23[0][0]
activation_101 (Activation)	(None, 48, 48, 256)	0	batch_normalization_162[0][0]
batch_normalization_163 (BatchN	(None, 48, 48, 256)	1024	conv2d_171[0][0]
conv2d_170 (Conv2D)	(None, 48, 48, 256)	590080	activation_101[0][0]
add_51 (Add)	(None, 48, 48, 256)	0	batch_normalization_163[0][0] conv2d_170[0][0]
up_sampling2d_24 (UpSampling2D)	(None, 96, 96, 256)	0	add_51[0][0]
concatenate_24 (Concatenate)	(None, 96, 96, 384)	0	up_sampling2d_24[0][0] add_48[0][0]
batch_normalization_164 (BatchN	(None, 96, 96, 384)	1536	concatenate_24[0][0]
activation_102 (Activation)	(None, 96, 96, 384)	0	batch_normalization_164[0][0]
conv2d_172 (Conv2D)	(None, 96, 96, 128)	442496	activation_102[0][0]
batch_normalization_165 (BatchN	(None, 96, 96, 128)	512	conv2d_172[0][0]

TABLE 5.3 – Partie 03 du modèle ResUNet.

conv2d_174 (Conv2D)	(None, 96, 96, 128)	49280	concatenate_24[0][0]
activation_103 (Activation)	(None, 96, 96, 128)	0	batch_normalization_165[0][0]
batch_normalization_166 (BatchN	(None, 96, 96, 128)	512	conv2d_174[0][0]
conv2d_173 (Conv2D)	(None, 96, 96, 128)	147584	activation_103[0][0]
add_52 (Add)	(None, 96, 96, 128)	0	batch_normalization_166[0][0] conv2d_173[0][0]
up_sampling2d_25 (UpSampling2D)	(None, 192, 192, 128	0	add_52[0][0]
concatenate_25 (Concatenate)	(None, 192, 192, 192	0	up_sampling2d_25[0][0] add_47[0][0]
batch_normalization_167 (BatchN	(None, 192, 192, 192	768	concatenate_25[0][0]
activation_104 (Activation)	(None, 192, 192, 192	0	batch_normalization_167[0][0]
conv2d_175 (Conv2D)	(None, 192, 192, 64)	110656	activation_104[0][0]
batch_normalization_168 (BatchN	(None, 192, 192, 64)	256	conv2d_175[0][0]
conv2d_177 (Conv2D)	(None, 192, 192, 64)	12352	concatenate_25[0][0]
activation_105 (Activation)	(None, 192, 192, 64)	0	batch_normalization_168[0][0]
batch_normalization_169 (BatchN	(None, 192, 192, 64)	256	conv2d_177[0][0]
conv2d_176 (Conv2D)	(None, 192, 192, 64)	36928	activation_105[0][0]
add_53 (Add)	(None, 192, 192, 64)	0	batch_normalization_169[0][0] conv2d_176[0][0]
conv2d_178 (Conv2D)	(None, 192, 192, 1)	65	add_53[0][0]

Total params: 8,233,665
Trainable params: 8,224,449
Non-trainable params: 9,216

TABLE 5.4 – Partie 04 du modèle ResUNet.

Modèle UNet : L'architecture de ce modèle est divisée en deux parties, la première partie représente le Down-Sampling et la deuxième partie est représentée le Up-Sampling comme indiqué dans le tableau ci-dessous :

Model: "Modèle UNet"

Layer (type)	Output Shape	Param #	Connected to
input (InputLayer)	(None, 192, 192, 4)	0	
block1_conv1 (Conv2D)	(None, 192, 192, 64)	2368	input[0][0]
block1_conv2 (Conv2D)	(None, 192, 192, 64)	36928	block1_conv1[0][0]
block1_batch_norm (BatchNormali	(None, 192, 192, 64)	256	block1_conv2[0][0]
block1_pool (MaxPooling2D)	(None, 96, 96, 64)	0	block1_batch_norm[0][0]
block2_conv1 (Conv2D)	(None, 96, 96, 128)	73856	block1_pool[0][0]
block2_conv2 (Conv2D)	(None, 96, 96, 128)	147584	block2_conv1[0][0]
block2_batch_norm (BatchNormali	(None, 96, 96, 128)	512	block2_conv2[0][0]
block2_pool (MaxPooling2D)	(None, 48, 48, 128)	0	block2_batch_norm[0][0]
encoder_dropout_1 (Dropout)	(None, 48, 48, 128)	0	block2_pool[0][0]
block3_conv1 (Conv2D)	(None, 48, 48, 256)	295168	encoder_dropout_1[0][0]
block3_conv2 (Conv2D)	(None, 48, 48, 256)	590080	block3_conv1[0][0]
block3_batch_norm (BatchNormali	(None, 48, 48, 256)	1024	block3_conv2[0][0]
block3_pool (MaxPooling2D)	(None, 24, 24, 256)	0	block3_batch_norm[0][0]
block4_conv1 (Conv2D)	(None, 24, 24, 512)	1180160	block3_pool[0][0]
block4_conv2 (Conv2D)	(None, 24, 24, 512)	2359808	block4_conv1[0][0]
block4_batch_norm (BatchNormali	(None, 24, 24, 512)	2048	block4_conv2[0][0]
block4_pool (MaxPooling2D)	(None, 12, 12, 512)	0	block4_batch_norm[0][0]

TABLE 5.5 – Partie 01 du modèle UNet.

block5_conv1 (Conv2D)	(None, 12, 12, 1024)	4719616	block4_pool[0][0]
up_pool1 (Conv2DTranspose)	(None, 24, 24, 1024)	9438208	block5_conv1[0][0]
merged_block1 (Concatenate)	(None, 24, 24, 1536)	0	block4_batch_norm[0][0] up_pool1[0][0]
decod_block1_conv1 (Conv2D)	(None, 24, 24, 512)	7078400	merged_block1[0][0]
up_pool2 (Conv2DTranspose)	(None, 48, 48, 512)	2359808	decod_block1_conv1[0][0]
merged_block2 (Concatenate)	(None, 48, 48, 768)	0	block3_batch_norm[0][0] up_pool2[0][0]
decod_block2_conv1 (Conv2D)	(None, 48, 48, 256)	1769728	merged_block2[0][0]
decoder_dropout_1 (Dropout)	(None, 48, 48, 256)	0	decod_block2_conv1[0][0]
up_pool3 (Conv2DTranspose)	(None, 96, 96, 256)	590080	decoder_dropout_1[0][0]
merged_block3 (Concatenate)	(None, 96, 96, 384)	0	block2_batch_norm[0][0] up_pool3[0][0]
decod_block3_conv1 (Conv2D)	(None, 96, 96, 128)	442496	merged_block3[0][0]
up_pool4 (Conv2DTranspose)	(None, 192, 192, 128	147584	decod_block3_conv1[0][0]
merged_block4 (Concatenate)	(None, 192, 192, 192	0	block1_batch_norm[0][0] up_pool4[0][0]
decod_block4_conv1 (Conv2D)	(None, 192, 192, 64)	110656	merged_block4[0][0]
pre_output (Conv2D)	(None, 192, 192, 64)	4160	decod_block4_conv1[0][0]
output (Conv2D)	(None, 192, 192, 4)	260	pre_output[0][0]

Total params: 31,350,788
Trainable params: 31,348,868
Non-trainable params: 1,920

TABLE 5.6 – Partie 02 du modèle UNet.

5.4 Training and tests

Dans cette section, nous parlerons des phases d'entraînement et de test de nos modèles. Il est divisé en deux sous-sections, la première sous-section sera dédiée à l'entraînement de nos modèles à l'aide d'un ensemble d'images de train, notez que nos modèles sont entraînés sur un nombre d'époques (5), et la seconde contiendra le test de nos modèles en utilisant un ensemble d'images de test.

5.4.1 Training

Tous les modèles sont entraînés sur une base de données du (2.57 GB) en raison de problèmes de ressources techniques, 76 patients avec 4104 images comme cela est résumé dans le tableau ci-dessous :

Data	Nombre d'images
Training set	4104
Test set	1368
Validation set	1368
Total	6840

TABLE 5.7 – Représentation du Dataset.

Tous les résultats (Accuracy, Loss, F1-Score, Precision et Recall et Temps) obtenus à partir de la phase d'entraînement sont présentés ci-dessous pour chaque modèle.

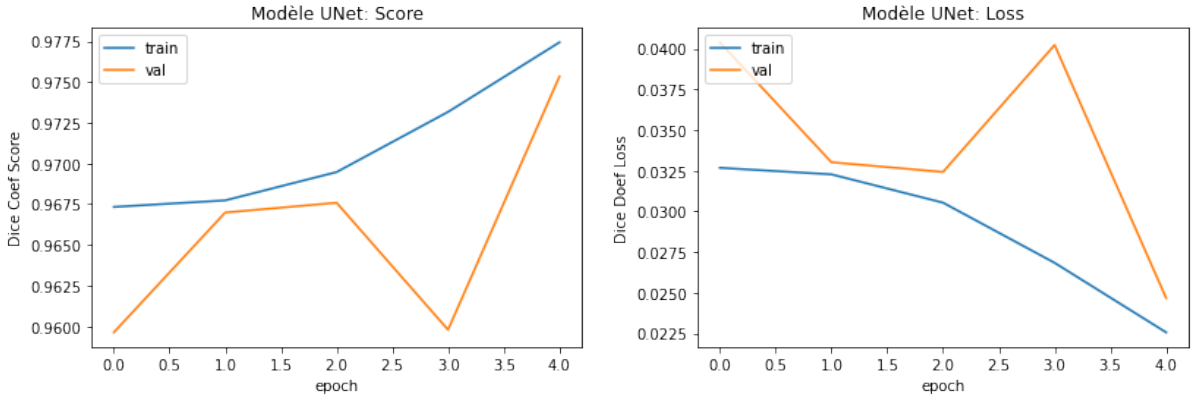
1. **Modèle UNet :**

Modèle UNet - Entraînement					
Nombre d'époques	**1**	**2**	**3**	**4**	**5**
Accuracy	0.9673	0.9677	0.9695	0.9732	**0.9774**
Loss	0.0327	0.0323	0.0305	0.0268	**0.0226**
F1-Score	0.9673	0.9675	0.9695	0.9733	**0.9771**
Precision	0.9674	0.9676	0.9718	**0.9789**	0.9785
Recall	0.9672	0.9674	0.9672	0.9678	**0.9757**
Temps	**20m 32s**	**41m 3s**	**1h 1m 47s**	**1h 22m 19s**	**1h 42m 51s**

TABLE 5.8 – Accuracy, Loss, F1-Score, Precision et Recall et Temps pour le Modèle UNet

Maintenant le graphique pour : Loss, Accuracy, F1-Score, Precision et Recall :

— **Accuracy(Score) et Loss :**

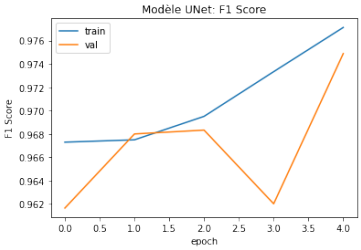


(a) Accuracy du Modèle UNet selon 5 époques.

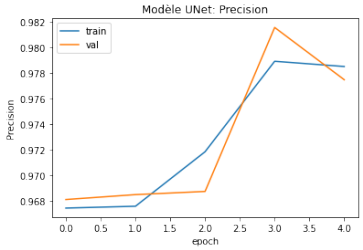
(b) Loss du Modèle UNet selon 5 époques.

FIGURE 5.2 – Accuracy(Score) et Loss du Modèle UNet selon 5 époques.

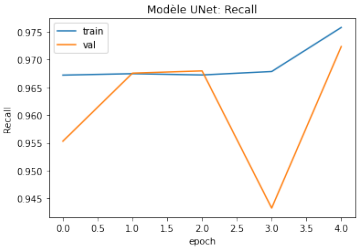
— **F1-Score et Precision et Recall :**



(a) F1-Score du Modèle UNet selon 5 époques.



(b) Precision du Modèle UNet selon 5 époques.



(c) Recall du Modèle UNet selon 5 époques.

FIGURE 5.3 – F1-Score et Precision et Recall du Modèle UNet selon 5 époques.

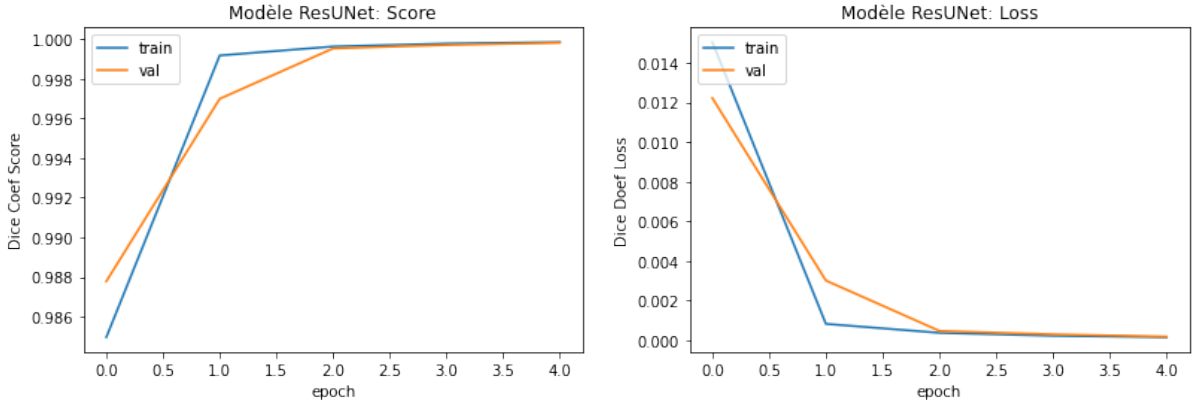
2. **Modèle ResUNet** :

Modèle ResUNet - Entraînement					
Nombre d'époques	**1**	**2**	**3**	**4**	**5**
Accuracy	0.9511	0.9887	0.9990	0.9995	**0.9997**
Loss	0.0489	0.0113	0.0010	0.0005	**0.0003**
F1-Score	0.9777	0.9962	**1.000**	**1.000**	**1.000**
Precision	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Recall	0.9574	0.9924	**1.000**	**1.000**	**1.000**
Temps	**30m 27s**	**1h 1m 1s**	**1h 31m 31s**	**2h 2m 7s**	**2h 32m 41s**

TABLE 5.9 – Accuracy, Loss, F1-Score, Precision et Recall et Temps pour le Modèle ResUNet

Maintenant le graphique pour : Loss, Accuracy, F1-Score, Precision et Recall :

— **Accuracy(Score) et Loss :**

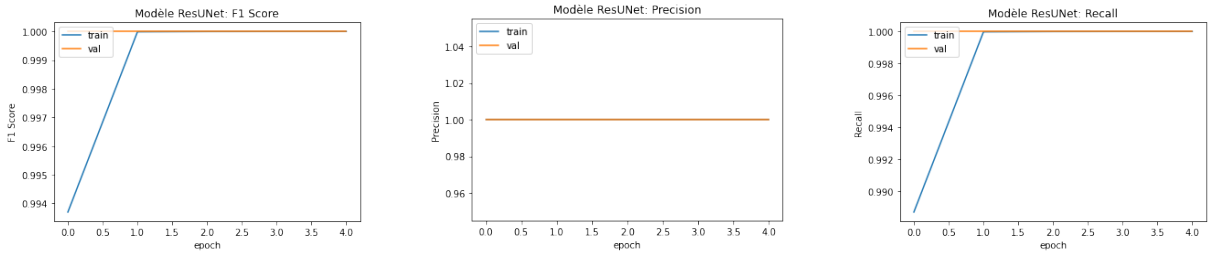


(a) Accuracy du Modèle ResUNet selon 5 époques.

(b) Loss du Modèle ResUNet selon 5 époques.

FIGURE 5.4 – Accuracy(Score) et Loss du Modèle ResUNet selon 5 époques.

— **F1-Score et Precision et Recall :**



(a) F1-Score du Modèle ResUNet selon 5 époques.

(b) Precision du Modèle ResUNet selon 5 époques.

(c) Recall du Modèle ResUNet selon 5 époques.

FIGURE 5.5 – F1-Score et Precision et Recall du Modèle ResUNet selon 5 époques.

5.4.2 Tests

Après avoir formé nos modèles, nous passons maintenant à la phase d'évaluation, où nous testerons nos modèles dans l'ensemble de test de 684 images Tous les résultats des tests sont présentés dans les tableaux ci-dessous :

1. **Modèle UNet** :

Modèle UNet - Tests	
Nombre d'époques	5
Accuracy	0.9733
loss	0.0266
F1-Score	0.9719
Precision	0.9753
Recall	0.9686

TABLE 5.10 – Résultats des tests du modèle UNet

2. **Modèle ResUNet** :

Modèle ResUNet - Tests	
Nombre d'époques	5
Accuracy	0.9997
loss	0.0002
F1-Score	1.0
Precision	1.0
Recall	1.0

TABLE 5.11 – Résultats des tests du modèle ResUNet

5.5 Analyse et discussion

Dans cette section, nous comparerons les modèles en fonction de leurs résultats obtenus, et déterminerons quel modèle parmi eux est le meilleur.

Comparaison des modèles

A partir de la figure ci-dessus, il est évident que le modèle ResUnet avec 5 époques d'entraînement a les meilleures valeurs pour chaque métrique.

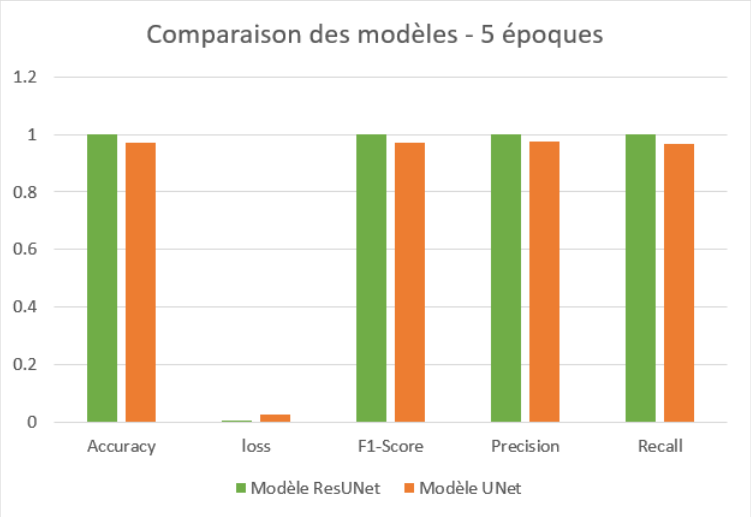


FIGURE 5.6 – Comparaison Entre Unet et ResUnet

Enfin, nous pouvons dire que dans tous les modèles d'approches proposés, ResUNet entraîné sur 5 époques, la plus élevé pour la détection de Brain Tumer en termes de métriques.

Comparaison avec d'autres travaux

Après avoir connu le meilleur modèle que nous ayons créé, il est maintenant temps de le comparer à un autre projet existant. ResNet Architecture [9] ,le résultat de la comparaison et illustré dans le tableau suivant :

ResUNet vs ResNet		
Modèle	**ResUNet**	**ResNet**
Accuracy	0.9997	0.9200
loss	0.0002	0.0800

TABLE 5.12 – Modèle ResUNet VS Modèle Resnet

9. github : https ://github.com/bclwan/MRI_Brain_Segmentation

5.6 Les Obstacles

Deep Learning ou L'apprentissage en profondeur est l'une des technologies les plus excitantes de cette décennie. Il a déjà fait son chemin dans de nombreux domaines, notamment la vision par ordinateur, la reconnaissance vocale, le traitement du langage naturel, la reconnaissance vocale et bien d'autres, obtenant des résultats similaires ou supérieurs aux humains.

Pour construire des modèles Deep Learning, avec une prévisibilité élevée et peu d'erreurs, nous devons entraînement avec une grande quantité de données , et pour s'entraîner, nous avons besoin de ressources matérielles informatiques puissantes telles que (CPU, GPU et espace de stockage).

Lors de la construction de nos modèles, nous avons trouver quelque problèmes et obstacles, qui sont présentés ci-dessous :

— Pendant que nous construisions nos modèles, nous n'avons trouvé aucun problème avec la disponibilité des données, les données sont partout, mais le seul problème que nous avons eu était est le manque de ressources informatiques ou services d'informatique en nuage(cloud) , nous avons utilisé **Google Collab** [31] où nous ne pouvons utiliser qu'un seul GPU avec un nombre limité de mémoires de 12 Go. Mais la quantité de ressources qu'il offre ne nous a pas aidés, alors nous nous sommes tournés vers Amazon SageMaker [32], dont nous avons parlé dans la section environnement de développement, mais ce n'est pas gratuit.

— L'autre problème que nous connaissons tous est Internet, nous serons honnêtes, nous avons pu former nos modèles avec 5 époques après environ une semaine, et la raison en est Internet fourni par Algérie Télécom. "très chanceux de pouvoir entraîner les modèles sur 5 époques et d'obtenir des scores de précision élevés."

5.7 Conclusion

Nous avons présenté dans le dernier chapitre le langage de programmation, les bibliothèques et les environnements utilisé pour l'implémentation de notre projet.Après cela, nous avons présenté quelque métriques dévaluation de la segmentation, puis nous sommes passés à l'implémentation décrivant le filtrage algorithme et nos modèles,ensuite, les phases training et testing ainsi que les différents résultats obtenus des comparaisons de chaque modèle ,enfin la comparaison finale avec un autre travail déjà existant.

Conclusion générale

Ce travail rentre dans le cadre du projet de fin d'étude pour l'obtention du diplôme « Master en Genie de Systèmes Informatiques ».

En guise de conclusion, nous rappelons que l'objectif de ce travail est de concevoir et d'implémenter un modéle pour la segmentation des tumeurs cérébrales à laide de Deep Learinig.

Dans ce manuscrit, nous avons présenté une méthode de segmentation des tumeurs cérébrales entièrement automatique basée sur les réseaux de neurones profonds

Le traitement automatique des images à laide des réseaux de neurones profonds s'impose dans le but d'aider le médecin lors du diagnostic ou le chirurgien lors de la réalisation d'un geste opératoire.

Les réseaux proposés sont adaptés aux glioblastomes (à la fois de bas et de haut grade) représentés sur les images IRM. Ici, nous donnons une description des différents choix de modèles que nous avons jugés nécessaires pour obtenir des performances compétitives. Nous explorons en particulier différentes architectures basées sur des réseaux de neurones convolutifs (CNN), c'est-à-dire des réseaux de neurones profonds spécifiquement adaptés aux données d'images.

CNN a été l'algorithme choisi dans notre étude car il offre une excellente fonctionnalité extracteur et la façon dont il traite les images en ont fait un ajustement parfait à notre projet. Les résultats que nous avons obtenu après avoir testé nos modèles prouve l'efficacité de nos modèles proposés.

Dans ce projet, nous avons présenté les notions de base nécessaires et la compréhension des techniques de traitement d'images et quelques méthodes de segmentation, par ailleurs, étudié les approches Deep Learning. puis nous sommes passés aux matériaux et méthodes que nous avons utilisés pour l'implémentation de nos modèles et une description détaillée de ce dernier. Enfin, nous avons discuté les résultats obtenus en implémentant et en testant

les deux modèles.

nous souhaitons que ce travail soit au niveau de la tâche qui nous a été confiée.

Travail futur :

Passons aux travaux et perspectives futurs. On sait qu'il n'y a pas de travail parfait. Dans un avenir proche, nous sommes impatients de rendre nos modèles encore plus précis et les rendre applicables dans le monde réel.

Bibliographie

[1] "Serge Faraut - EAT - ASM - MAP". *"L'image numérique"*. http://www.map.toulouse.archi.fr/works/panoformation/imagenum/imagenum.htm.

[2] "Alessandro ANZANI". *"la retouche d'image"*. http://tecfaetu.unige.ch/staf/staf-f/anzani/staf14/ex2/.

[3] L. Guigues. "Modèles multi-échelles pour la segmentation d'images". In : 2003.

[4] AISSA BRAHIM Salim KADDOUR Chakib. "Compression des images fixes par fractale basée sur la triangulation de Delaunay et la quantification vectorielle". In : (2010), p. 3-5.

[5] "Canson-infinity". *"Qu'est-ce que la 'résolution' d'une image ?"*. https://www.canson-infinity.com/fr/faq/qu-est-ce-que-la-resolution-d-une-image..

[6] J-P. COCQUEREZ et S. PHILIPP. "Analyse d'images : Filtrage et segmentation". In : (1995), p. 84-100.

[7] "H. Maître". *"ILTRES DE DEBRUITAGE : PRINCIPES ET PERFORMANCES"*. http://www.tsi.enst.fr/pages/enseignement/ressources/beti/filtres_lin_nlin/mti.html.

[8] AYADI Abdelghafar. "APPLICATION D'UNE VARIENTE D'ALGORITHMESGENETIQUES A LA SEGMENTATION D'IMAGES". In : (2012), p. 16-17.

[9] J.J ROUSSELLE. "Les Contours Actifs : Une méthode de segmentation , Application a l'imagerie médicale, thèse de doctorat, Université François Rabelais de Tours". In : (2003), p. 18.

[10] M. Bhattacharya3 Dharmveer Singh Rajoot1 P. K. Singh2. "A New Technique For Feature Selection And Cluster Center Initialization". In : (2010), p. 4-5.

[11] "Murugan N.". *"Image Segmentation K-means Clustering"*. https://perso.esiee.fr/~perretb/I5FM/TAI/convolution/index.html#detection-de-contours.

[12] "Benjamin PERRET". *"Détection de contours"*. https://perso.esiee.fr/~perretb/I5FM/TAI/convolution/index.html#detection-de-contours.

[13] "DISHAYLOO". *"Seuillage d'image"*. https://commons.wikimedia.org/wiki/File:Rauschbild.png.

[14] "Aymeric HISTACE". *"Traitement numérique de l'image :Bases de la segmentation d'images"*. http://aymeric.histace.free.fr/documents/Segmentation.pdf.

[15] Mr HOUASSINE CHARIF. "Segmentation d'images par une approche biomimétique hybride". In : (2012), p. 23.

[16] "DELUZARCHE C.". *"Deep Learning FuturaTech"*. https://www.futura-sciences.com/tech/definitions/intelligence-artificielle-deep-learning-17262.

[17] Patrice WIRA. "Réseaux de neurones artificiels : Architecture et applications". In : (avril 2009), p. 6-28.

[18] KOUAME KHODISSOU BARETY J-B. "ETUDE ET MISE EN PLACE D'UNE INTELLIGENCE ARTIFICIELLE AUTONOME CAPABLE DE SAUVEGARDER ET SÉCURISER DES DONNÉES INFORMATIQUES". In : (2019).

[19] Ridha GHAYOULA. "Contribution à l'optimisation de la Synthèse des Antennes Intelligentes par les Réseaux de Neurones". In : (2008).

[20] Samuel. DUPOND. "A thorough review on the current advance of neural network structures". Annual Reviews in Control." In : (2019), p. 200-230.

[21] "PRABHU.". *"Understanding of Convolutional Neural Network (CNN) Deep Learning"*. https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148.

[22] "ICHI.PRO". *"Compréhension du réseau neuronal convolutif (CNN) - Deep Learning"*. https://ichi.pro/fr/comprehension-du-reseau-neuronal-convolutif-cnn-deep-learning-168732222943560.

[23] H. Khiter. H. SELMANE. "Détection Histologique du Cancer en utilisant le Deep Learning". Masters thesis, Computer Science Department, University of Bouira, Bouira, Algeria." In : (2019).

[24] Olaf RONNEBERGER, Philipp FISCHER et Thomas BROX. "U-Net : Convolutional Networks for Biomedical Image Segmentation". In : *CoRR* abs/1505.04597 (2015). arXiv : 1505.04597. URL : http://arxiv.org/abs/1505.04597.

[25] Zhengxin ZHANG, Qingjie LIU et Yunhong WANG. "Road Extraction by Deep Residual U-Net". In : *CoRR* abs/1711.10684 (2017). arXiv : 1711.10684. URL : http://arxiv.org/abs/1711.10684.

[26] "NIKHIL KUMAR TOMAR". *"What is RESUNET"*. https://idiotdeveloper.com/what-is-resunet/.

[27] Spyridon (Spyros) BAKAS. *Brats MICCAI Brain tumor dataset*. 2020. DOI : 10.21227/hdtd-5j88. URL : https://dx.doi.org/10.21227/hdtd-5j88.

[28] Lantian LI, Weizhi XU et Hui YU. "Character-level neural network model based on Nadam optimization and its application in clinical concept extraction". In : *Neurocomputing* 414 (2020), p. 182-190. ISSN : 0925-2312. DOI : https://doi.org/10.1016/j.neucom.2020.07.027. URL : https://www.sciencedirect.com/science/article/pii/S0925231220311346.

[29] Fausto MILLETARI, Nassir NAVAB et Seyed-Ahmad AHMADI. "V-Net : Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation". In : *CoRR* abs/1606.04797 (2016). arXiv : 1606.04797. URL : http://arxiv.org/abs/1606.04797.

[30] Himanshu S. "Activation Functions : Sigmoid, tanh, ReLU, Leaky ReLU, PReLU, ELU, Threshold ReLU and Softmax basics for Neural Networks and Deep Learning". In : (2019).

[31] "Simon PRUD'HOMME". *"Initiation au Deep Learning avec Google Colab"*. https://moov.ai/fr/blog/deep-learning-avec-google-colab/.

[32] "Amazon web SERVICES". *"Questions fréquentes (FAQ) sur Amazon SageMaker"*. https://aws.amazon.com/fr/sagemaker/faqs/.

[33] "Thomas WOOD". *"F-Score"*. https://deepai.org/machine-learning-glossary-and-terms/f-score.