REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEURE ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITEIBNKHALDOUN-TIARET

MEMOIRE

Présenté à:

FACULTÉMATHEMATIQUE
SETINFORMATIQUEDÉPAR
TEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de:

MASTER

Spécialité : Génie Informatique

Par :

BOUZIRI SALIMA

FARHA MESSAOUDA

Sur le thème

Segmentation des images médicales cérébrales Par l'apprentissage approfondie dans un Environnement Cloud

---

Soutenu publiquement le 09 /07/2023 à Tiaret devant le jury composé de:

Mr DAHMANI YOUSEF		Université Ibn-khaldoun	Président
Mr MESSAOUD HAMEURLAINE	M.C.A	Université Tissemsilte	Encadreur
Mr BAGHDADI MOHAMED		Université Ibn-khaldoun	Examinateur

2022-2023

Remerciement

En premier lieu, nous tenons à exprimer notre profonde gratitude à Dieu.

Merci Allah,

pour la force,. Je suis reconnaissante pour toutes les bénédictions que vous avez versées sur moi.

À mon encadreur, je vous remercie du fond du cœur pour votre expertise, votre patience et votre dévouement tout au long de cette recherche. Votre mentorat précieux m'a permis de me développer académiquement et professionnellement. Vos conseils éclairés et votre soutien indéfectible ont été essentiels à ma croissance personnelle et à la réalisation de ce projet.

*Enfin, je tiens à exprimer ma gratitude envers les membres du jury (*MR Dahmani Yousef & Baghdadi Mohamed*) qui ont consacré leur temps et leur expertise pour évaluer mon travail. Votre engagement envers l'excellence académique et votre précieuse contribution à cette journée de discussion sont hautement appréciés.*

Dédicace

Je profite de cette occasion spéciale pour adresser une dédicace toute particulière à mes parents, sœurs, fiancé et amis qui m'ont apporté un soutien inconditionnel tout au long de mon parcours.

Pour mes parents, je suis tellement reconnaissante pour vos amours indéfectibles, vos soutiens inconditionnels et vos encouragements constants. Votre confiance en moi a été une inspiration et une motivation. Je ne serais pas là aujourd'hui sans vos amours et vos dévouements.

Pour mes sœurs (Souad, Fatima, Ikram) vous êtes mes partenaires, mes amis les plus proches et mes meilleurs amis. Ta présence dans ma vie a été un cadeau précieux. Vos encouragements et votre soutien indéfectibles ont toujours été un atout pour moi. Merci d'avoir été là à chaque étape.

A mon fiancé, tu es mon roc, mon pilier de force et mon plus grand soutien. Vous avez été à mes côtés en

m'encourageant et en me motivant dans tous mes projets. Votre patience, votre amour et votre compréhension sont inestimables. Je suis tellement reconnaissante de t'avoir dans ma vie.

Et enfin, à mes amies, vous êtes ma famille choisie. Votre amitié sincère, vos encouragements et votre présence constante ont illuminé mon chemin. Merci d'avoir partagé les hauts et les bas avec moi, et d'avoir toujours cru en moi.

Cette dédicace est un témoignage de ma gratitude et de mon amour envers vous tous. Votre soutien inébranlable a été la clé de ma réussite. Je suis honorée de vous avoir dans ma vie, et je sais que notre lien restera fort, peu importe les défis qui se présentent.

Merci d'être mes piliers, mes inspirations et mes sources de bonheur. Cette réussite est aussi la vôtre.

Je vous aimes tous de touts mon cœur.

BOUZIRI SALIMA

ملخص:

يعتبر تجزئة ورم الدماغ من الصور ثلاثية الأبعاد من المهام الأكثر أهمية وصعوبة في مجال معالجة الصور الطبية ، حيث يمكن أن يؤدي التصنيف اليدوي بمساعدة الإنسان إلى التنبؤ والتشخيص غير الصحيحين. علاوة على ذلك ، إنها عملية صعبة عندما يكون هناك كمية هائلة من البيانات للمساعدة. يصبح استخراج مناطق ورم الدماغ من صور التصوير بالرنين المغناطيسي أمرًا صعبًا بسبب التنوع الكبير في ظهور أورام المخ ومدى تشابهها مع الأنسجة الطبيعية. في هذه الورقة ، قمنا بتصميم بنية U-Net معدلة في إطار التعلم العميق لاكتشاف وتجزئة أورام الدماغ من صور التصوير بالرنين المغناطيسي. تم تقييم النموذج المطبق على صور أصلية مقدمة من مجموعات بيانات حوسبة الصور الطبية والتدخلات بمساعدة الكمبيوتر BRATS 2020. تم تحقيق دقة الاختبار بنسبة 99.4٪ باستخدام مجموعة البيانات المذكورة أعلاه. تظهر مراجعة مقارنة مع أوراق أخرى أن نموذجنا الذي يستخدم U-Net يؤدي بشكل أفضل من النماذج الأخرى القائمة على التعلم العميق.

الكلمات الدالة : ورم في المخ ;تجزئة ; التصوير بالرنين المغناطيسي ; كشف الورم ;U_Net.

ABSTRACT:

Segmentation of brain tumor from 3D images is one of the most important and difficult tasks in the field of medical image processing as a manual human-assisted categorization can result in incorrect prediction and diagnosis. Furthermore, it is a difficult process when there is a huge amount of data to assist. Extracting brain tumor regions from MRI images becomes challenging due to the great variety of appearances of brain tumors and how similar they are to normal tissues. In this paper, we have designed modified U-Net architecture under a deep-learning framework for the detection and segmentation of brain tumors from MRI images. The applied model has been evaluated on genuine images provided by Medical Image Computing and Computer-Assisted Interventions BRATS 2020 datasets. Test accuracy of 99.4% has been achieved using the above-mentioned dataset. A comparative review with other papers shows our model using U-Net performs better than other deep learning-based model.

KEYWORDS: BRATS; brain tumor; segmentation; MRI; U_net; tumor detection.

Table des matières

LISTE DES FIGURES

Chapitre 1 : Imagerie médicale cérébrale

LISTE DES TABLEAUX

Chapitre 1 : Imagerie médicale cérébrale

Chapitre 3 : Contribution et Résultat

Introduction Général

Introduction

L'intelligence artificielle (IA) est un domaine de l'informatique qui vise à développer des systèmes capables de réaliser des tâches qui nécessitent normalement l'intelligence humaine. Ces systèmes utilisent des algorithmes et des modèles pour analyser, comprendre, raisonner et prendre des décisions à partir de données.
Le deep learning, quant à lui, est une sous-branche de l'intelligence artificielle qui se concentre sur l'apprentissage automatique de représentations de données à travers des réseaux de neurones artificiels profonds.

L'intelligence artificielle, en particulier l'apprentissage en profondeur, permet aux ordinateurs d'apprendre à partir de données et de reconnaître des modèles complexes.
L'intelligence artificielle et le deep learning ont également révolutionné de nombreux domaines, dont la segmentation des images médicales. Avec l'avènement de ces technologies avancées, la segmentation des images médicales devient plus précise, efficace et automatisée, ce qui ouvre de nouvelles opportunités dans le domaine de la santé.
la segmentation des images médicales cérébrales est cruciale pour un diagnostic précis, un suivi des patients efficace, une planification chirurgicale précise et une évaluation des traitements. En permettant une analyse détaillée des structures cérébrales, la segmentation améliore les capacités diagnostiques, la prise de décision médicale et les résultats cliniques.

Problématique

La segmentation des images médicales cérébrales présente plusieurs défis et problèmes. comme Complexité des structures cérébrales et Variabilité des données, Temps nécessaire à la segmentation manuelle, Besoin de méthodes automatiques.

Face à ces défis, l'utilisation de techniques d'apprentissage en profondeur, telles que les réseaux de neurones convolutionnels (CNN), offre de nouvelles perspectives pour la segmentation précise des images médicales cérébrales. Ces approches permettent d'apprendre à partir de grands ensembles de données annotées et de capturer les caractéristiques complexes des structures cérébrales, améliorant ainsi la précision et l'efficacité de la segmentation.

Objectif

Les objectifs spécifiques de notre travail de recherche ou de notre projet sont les suivants :

Explorer les méthodes de segmentation automatique des images médicales cérébrales : Nous cherchons à étudier et à mettre en œuvre des techniques avancées d'apprentissage en profondeur pour la segmentation automatique des images médicales cérébrales. Nous souhaitons examiner les approches basées sur les réseaux de neurones convolutionnels (CNN) et évaluer leur performance par rapport aux méthodes traditionnelles de segmentation.

Utiliser des techniques d'apprentissage en profondeur : Nous avons l'intention de mettre en œuvre des modèles de deep learning, tels que les réseaux de neurones convolutionnels (CNN), pour apprendre à segmenter les différentes structures cérébrales à partir des images médicales. Nous explorerons également des architectures populaires, telles que U-Net, qui se sont révélées efficaces dans la segmentation d'images médicales.

Organisation du mémoire

Après cette introduction général ,la structure de reste de notre travaille est le suivant :

Le premier chapitre :

Dans ce chapitre, nous allons aborderons le domaine de l'imagerie médicale cérébrale en mettant l'accent sur l'anatomie cérébrale, les tumeurs cérébrales et l'imagerie par résonance magnétique (IRM).

Le deuxième chapitre :

Dans ce chapitre, nous aborderons la segmentation des images médicales cérébrales, une étape cruciale dans l'analyse des données d'imagerie pour le diagnostic et le traitement des affections cérébrales. Nous examinerons différentes approches de segmentation utilisées dans ce domaine,et les différentes approches de segmentation.

Ainsi que l'importance de l'apprentissage en profondeur et de l'évaluation pour améliorer la précision et la fiabilité des résultats. Nous explorerons également le potentiel du diagnostic assisté par ordinateur dans l'amélioration des soins de santé liés aux affections cérébrales.

Le troisième chapitre :

Dans ce chapitre, nous avons exploré l'environnement de travail en nuage (Cloud) et l'utilisation du langage de programmation Python dans le contexte de la segmentation d'images médicales cérébrales. Nous avons constaté que l'environnement de travail en nuage, tel que Google Colab, offre de nombreux avantages pour la mise en œuvre de nos projets.

CHAPITRE 01

IMAGERIE MÉDICALE CÉRÉBRALE

Introduction

Dans ce chapitre nous avons parler sur L'imagerie médicale ce qui est une discipline essentielle dans le domaine de la santé qui permet d'obtenir des images détaillées de l'intérieur du corps humain. Elle joue un rôle crucial dans le diagnostic, le suivi et le traitement des différentes pathologies.

L'anatomie cérébrale est l'étude de la structure du cerveau, qui comprend différentes régions, structures et connexions. Comprendre l'anatomie cérébrale est fondamental pour appréhender les processus neurologiques et les maladies qui y sont associées.

Les tumeurs cérébrales sont des masses anormales de cellules qui se développent dans le cerveau. Elles peuvent être bénignes (non cancéreuses) ou malignes (cancéreuses), et peuvent causer des symptômes tels que des maux de tête, des troubles de la vision, des troubles de la parole, etc. Leur détection et leur caractérisation précoces sont cruciales pour un traitement efficace.

L'IRM (Imagerie par Résonance Magnétique) est l'une des modalités d'imagerie médicale les plus utilisées pour l'étude du cerveau. Elle utilise des champs magnétiques et des ondes radio pour créer des images détaillées et en coupe du cerveau. L'IRM permet d'observer les structures anatomiques du cerveau, de détecter les tumeurs et d'évaluer leur étendue et leur progression.

En résumé, l'imagerie médicale, en particulier l'IRM, joue un rôle essentiel dans l'étude de l'anatomie cérébrale et la détection des tumeurs cérébrales. Ces avancées technologiques permettent aux professionnels de la santé de diagnostiquer précisément les pathologies cérébrales, de suivre leur évolution et de mettre en place des traitements adaptés.

1.1 Notion générale de l'image

1.1.1 Définition d'une image

Une image numérique est un fichier informatique pouvant être vu comme un tableau de nombres. Sa définition est le nombre total de points (pixels) qui la composent, et est donnée sous la forme nombre de pixels sur une ligne X nombre sur une colonne. [1]

1.1.2 Type d'images

1.1.2.1 Image numérique

Une image numérique est composée de cellules ou pixels de tailles fixes, qui contiennent des niveaux de gris ou des informations de couleur provenant de l'image réelle ou calculés à partir d'une description interne de la scène. Lorsqu'une image est numérisée, elle est acquise et traitée pour être stockée sous forme binaire. Une image numérique est définie par le nombre de pixels en largeur et en hauteur, ainsi que par les valeurs que chaque pixel peut prendre, que ce soit un scalaire pour les images en niveau de gris ou un vecteur RGB pour les images en couleur. Ces valeurs sont incluses dans N. [2]

1.1.2.2 Image aux niveaux de gris

Une image peut être considérée comme une fonction bidimensionnelle f(x, y) où x et y représentent les coordonnées spatiales dans un plan. La valeur f à chaque paire de coordonnées (x, y) est appelée l'intensité ou le niveau de gris de l'image à ce point. Lorsque les coordonnées (x, y) et les valeurs d'intensité f sont discrètes et finies, nous qualifions cette image de numérique. Le traitement numérique de l'image se réfère à la manipulation de ces images numériques à l'aide d'un ordinateur. [2]

1.1.2.3 Image en couleur

En général, les images en couleur sont basées sur trois couleurs primaires : Rouge, Vert et Bleu (RVB), et utilisent souvent 8 bits pour chaque composante de couleur, ce qui signifie que chaque pixel nécessite 24 bits pour représenter les trois composantes. Chaque composante de couleur peut prendre des valeurs dans l'intervalle de 0 à 255. Une méthode simple pour convertir une image RVB en une image en niveaux de gris consiste à utiliser la formule suivante :

gris = (bleu + vert + rouge) / 3. Cela revient à attribuer la même valeur de gris à chacune des trois composantes RVB. [2]

1.1.2 Imagerie médicale

L'imagerie médicale joue un rôle crucial dans la recherche clinique, l'étude des maladies et le développement de nouveaux traitements. Elle englobe de nombreuses techniques d'imagerie complémentaires qui exploitent les avancées majeures de la physique du 20e siècle. Ces techniques incluent :

- Les ondes radio et les rayons X.
- L'utilisation de substances radioactives.
- L'utilisation des champs magnétiques. [3]

1.2. Anatomie cérébral

1.2.1. Le cerveau

Le cerveau, qui constitue la majeure partie de l'encéphale, est divisé en deux hémisphères : le droit et le gauche. Sa surface présente de nombreux replis et des circonvolutions limitées par des sillons. Certains de ces sillons plus profonds, appelés scissures, permettent de diviser chaque hémisphère en quatre lobes distincts. Le lobe frontal est impliqué dans la pensée, la conceptualisation, la planification et la perception consciente des émotions. Le lobe pariétal est responsable des gestes, de l'orientation spatiale et de la reconnaissance visuelle. Le lobe occipital est principalement dédié à la vision, tandis que le lobe temporal joue un rôle essentiel dans l'interprétation des sons, du langage et de la mémoire. [4]

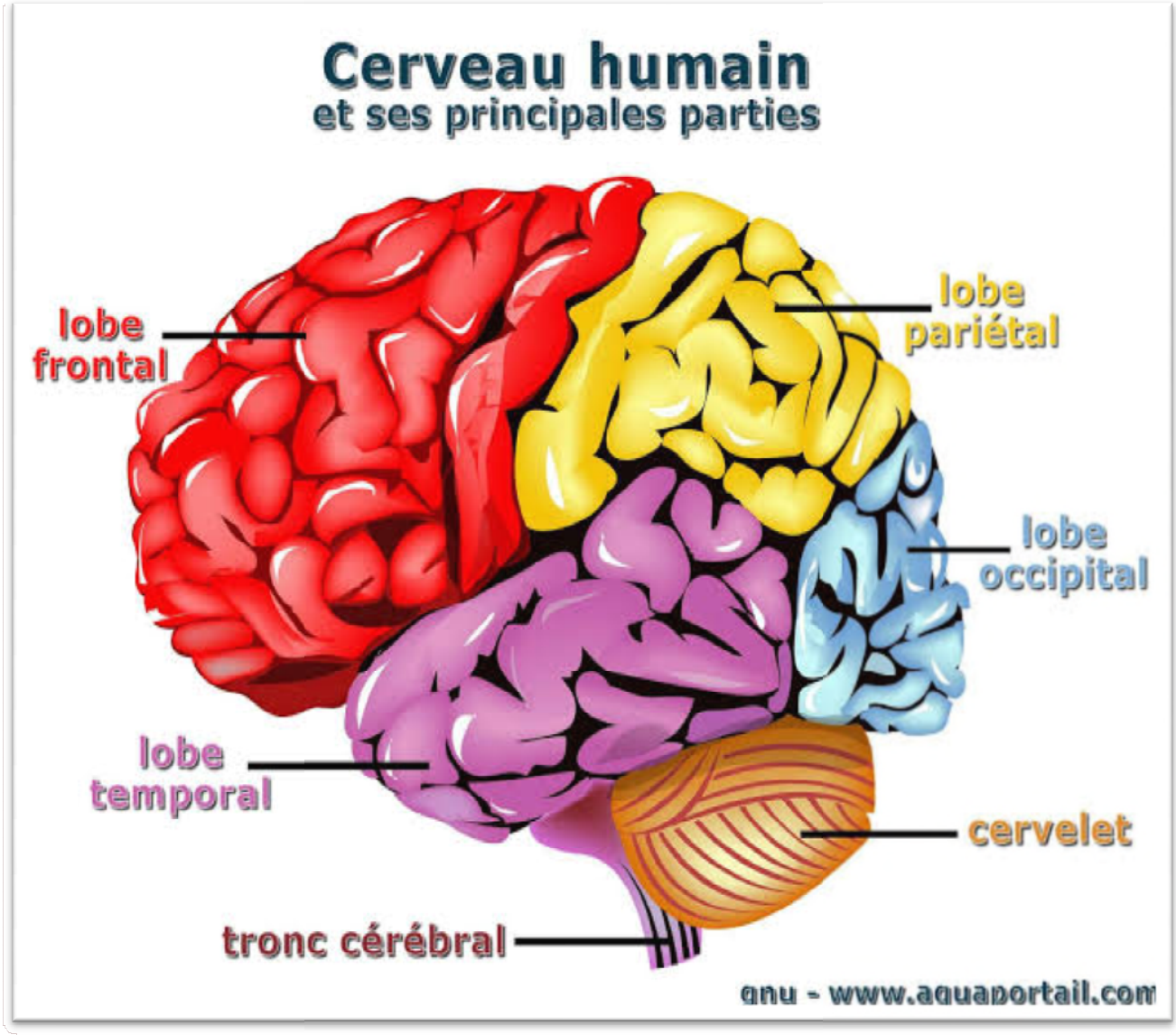


Figure 1.1: Le cerveau

Le tableau ci-dessous présente les principales fonctions de chaque lobe.

Lobes	Principales fonctions
Lobes frontaux	arole et langage, raisonnement, mémoire, prise de décision, personnalité, jugement, mouvements. Le lobe frontal droit gère les mouvements du côté gauche du corps, et inversement, le lobe frontal gauche gère les mouvements du côté droit
Lobes pariétaux	Lobes pariétaux Lecture, repérage dans l'espace, sensibilité. Là aussi, le lobe pariétal droit gère la sensibilité du côté gauche du corps et réciproquement
Lobes occipitaux	Vision
Lobes temporaux	Langage, mémoire, émotions

Tableau 1.1: Les principales fonctions des 4 lobes de l'hémisphère gauche

1.2.2. Le cervelet

Le cervelet se trouve entièrement dans la fosse postérieure, à l'arrière du tronc cérébral, et présente une forme pyramidale. Il est composé d'une partie médiane appelée vermis, et de deux lobes ou hémisphères cérébelleux de chaque côté. Sa surface externe est caractérisée par la présence de nombreux sillons courbes et concentriques, appelés lamelles du cervelet. Le tronc cérébral est relié au cervelet par trois paires de pédoncules cérébelleux : supérieur, moyen et inférieur.. Ces pédoncules permettent le passage des voies afférentes et efférentes provenant de la moelle épinière, du tronc cérébral, des voies vestibulaires et des noyaux des nerfs crâniens. Le rôle du cervelet est de réguler le tonus (dans le vermis cérébelleux) et de coordonner les mouvements (dans les hémisphères cérébelleux). À l'intérieur du cervelet, la substance grise constitue l'écorce, tandis que la substance blanche se trouve au centre, entourant les noyaux dentelés du cervelet (noyaux gris).

L'écorce du cervelet est composée de trois couches de cellules, la couche moyenne étant formée de cellules très spéciales appelées cellules de Purkinje, qui présentent de nombreuses ramifications.

1.2.3. Le tronc cérébral

Le tronc cérébral, situé dans la fosse postérieure entre le cerveau et la moelle épinière, est une structure centrale de l'encéphale. Il est composé de trois niveaux: le mésencéphale, la protubérance annulaire et le bulbe rachidien, qui sont décrits de haut en bas. Le tronc cérébral joue un rôle complexe en tant que passage des voies ascendantes (sensitives et cérébelleuses) et descendantes (motrices), qui transmettent les informations du cortex. Il abrite également les noyaux des nerfs crâniens moteurs et sensitifs (III à XII) répartis de chaque côté de la ligne médiane le long du tronc cérébral. De plus, le tronc cérébral abrite la substance réticulée, qui joue un rôle physiologique important dans la régulation de l'éveil, du sommeil, ainsi que d'autres formations essentielles telles que le locus Niger, qui contient des neurones dopaminergiques du système extrapyramidal [4].

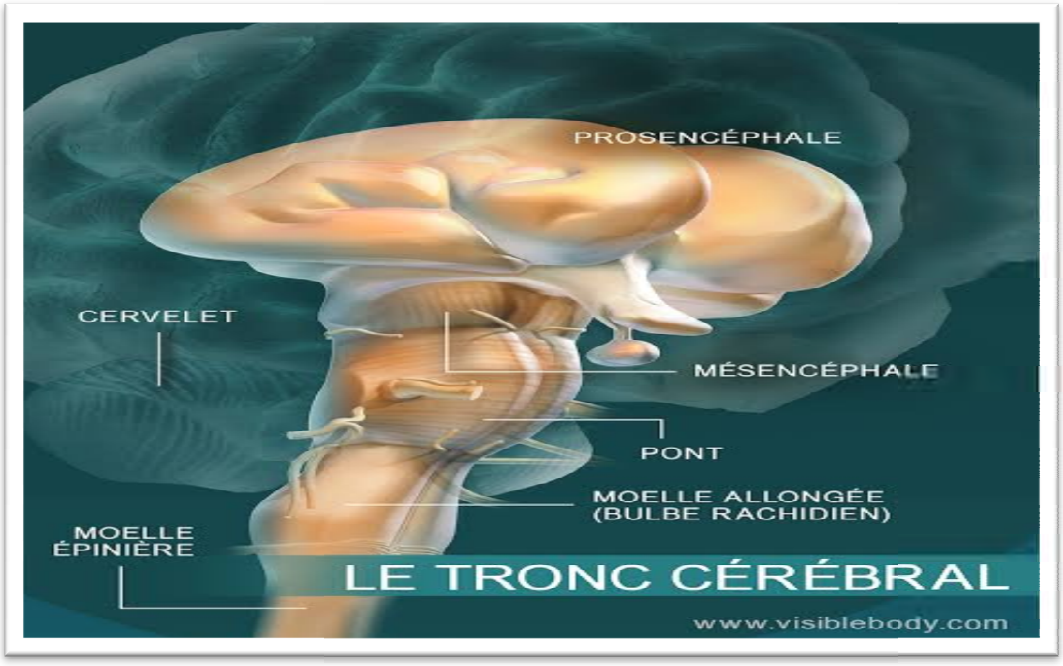


Figure 1. 1: Le tronc cérébral

1.2.4. Les composants de cerveau

Dans le cerveau, on distingue deux catégories de tissus : la matière grise et la matière blanche. La matière grise est constituée des corps cellulaires des neurones, ainsi que de leurs dendrites et d'autres types de cellules.. Elle joue un rôle crucial dans notre activité sensori-motrice et nos fonctions cognitives telles que la lecture, le calcul, l'attention et la mémoire. Les neurones sont des cellules spécialisées qui transmettent les informations. Ils sont interconnectés et communiquent entre eux par des signaux électriques et chimiques via les dendrites et les axones. Les axones sont les prolongements des neurones qui transmettent l'information sur de longues distances. Le message électrique se propage le long de l'axone et se ramifie pour atteindre d'autres neurones. Comme les neurones ne sont pas en contact direct, le message électrique est converti en message chimique pour être capté par les dendrites d'un autre neurone. Quant à la matière blanche, elle est constituée des axones enveloppés dans une gaine de myéline, qui permet la transmission efficace des informations entre les différentes régions de matière grise [5].

1.3. Les tumeurs cérébrales

1.3.1. Une tumeur

Les tumeurs peuvent être de taille variable en raison d'une multiplication excessive de cellules normales (tumeurs bénignes) ou anormales (comme les grains de beauté, les verrues, etc.). Les tumeurs bénignes se développent de manière localisée sans affecter les tissus environnants. En revanche, les tumeurs malignes (cancers) ont tendance à envahir les tissus voisins et à se propager à d'autres parties du corps, formant des métastases. [6]

1.3.2. La différence entre une tumeur bénigne et maligne

Une tumeur bénigne diffère d'une tumeur maligne, qui est considérée comme un cancer. Les tumeurs bénignes ont une croissance lente et se limitent à une zone spécifique, sans se propager à d'autres parties du corps sous forme de métastases. De plus, si une tumeur bénigne est entièrement enlevée, elle ne réapparaîtra pas. [7]

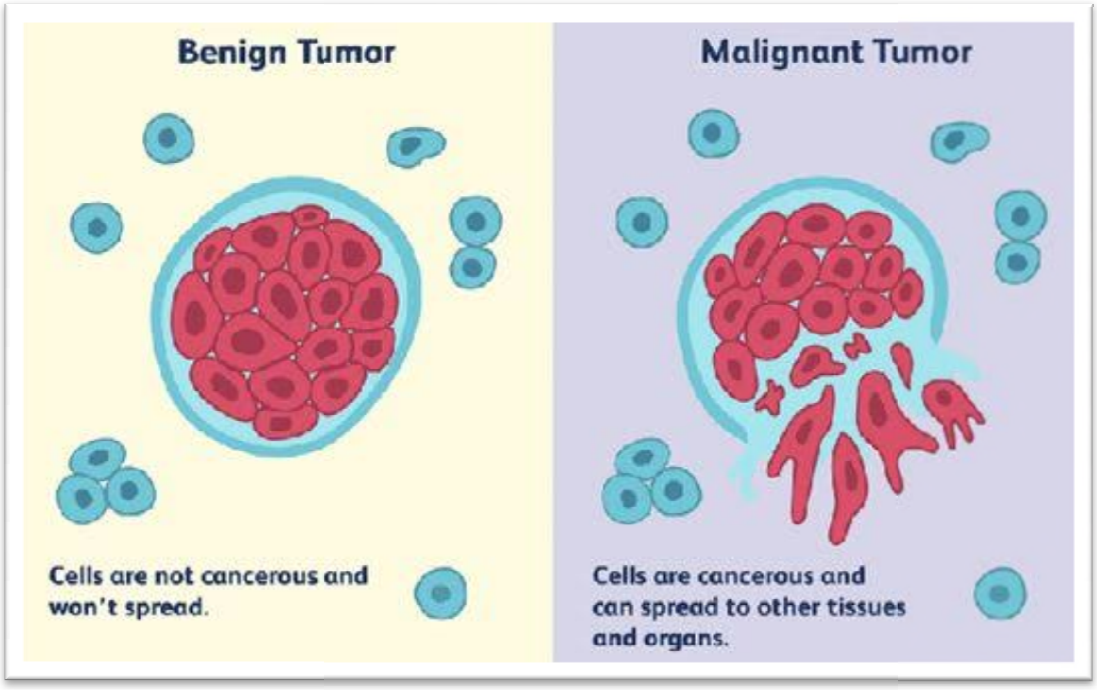


Figure 1. 1: La différence entre une tumeur bénigne et maligne

1.3.3. Les cause des tumeurs du cerveau

Le corps humain est constitué de cellules "normales" qui ont la capacité de se multiplier de manière incontrôlée et de se transformer en cellules cancéreuses. Ces cellules cancéreuses peuvent envahir les tissus sains et les endommager. Certaines de ces cellules ont la capacité de se détacher de leur site d'origine et de former des foyers secondaires dans d'autres parties du corps, appelés métastases. Dans la plupart des cas, les métastases atteignent le cerveau par le biais des vaisseaux sanguins. Il est encore inconnu pourquoi le cerveau est l'un des organes les plus fréquemment touchés par les métastases. Le risque qu'un cancer évolue en un cancer métastatique dépend de plusieurs facteurs, tels que le type de cancer, la taille et l'emplacement de la tumeur initiale, la vitesse de croissance de la tumeur initiale, la probabilité de propagation, ainsi que la durée pendant laquelle la tumeur initiale est présente dans le corps. [8]

1.3.4. La gradation des tumeurs

Le grade du cancer, basé sur l'apparence des cellules cancéreuses, permet de prédire la vitesse d'évolution et le risque de propagation du cancer. Habituellement noté de 1 à 4 ou 5, un grade plus élevé indique

une différence plus importante entre les cellules cancéreuses et les cellules saines, ainsi qu'une croissance plus rapide. Le stade du cancer informe les médecins sur la quantité de cancer dans le corps, sa localisation et son degré de propagation. Ces informations aident à déterminer les traitements appropriés. Le cancer peut se propager dans l'organe d'origine, les ganglions lymphatiques voisins ou à des sites distants. Différents examens permettent d'établir le stade, noté de 1 à 5. Un cancer de stade 1 est généralement de petite taille et n'a pas atteint d'autres sites que son lieu d'origine. Plus le stade est élevé, plus la taille de la tumeur ou l'étendue de la propagation est importante. Un cancer de stade 5 s'est généralement propagé à des sites distants. L'impact du stade et du grade sur le plan de traitement varie d'un type de cancer à l'autre. Votre médecin sera en mesure de vous fournir des informations spécifiques sur le stade et le grade de votre cancer et ce qu'ils signifient dans votre cas. [9]

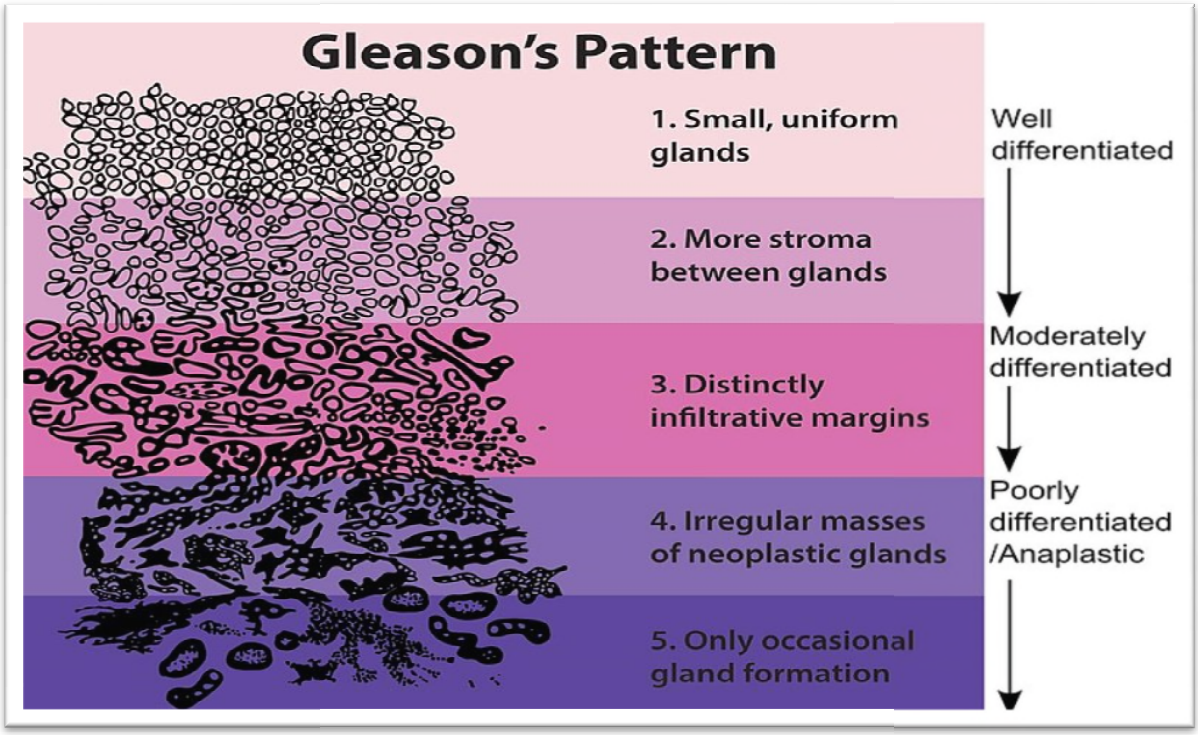


Figure 1. 1: La gradation des tumeurs

1.3.5. Le diagnostic de la tumeur

Pour diagnostiquer les métastases cérébrales, des techniques d'imagerie médicale sont utilisées afin d'obtenir une visualisation précise des structures cérébrales. Dans la plupart des cas, l'examen le plus couramment réalisé est l'imagerie par résonance magnétique (IRM) du cerveau. Cette technique, qui utilise des agents de contraste, permet de détecter des anomalies qui pourraient être moins visibles avec d'autres méthodes d'imagerie médicale, et de caractériser plus précisément la tumeur ou les métastases.[8]

1.4. Imagerie par résonance magnétique

Les avancées constantes dans les techniques d'imagerie médicale permettent d'améliorer les méthodes d'investigation, offrant ainsi des diagnostics plus précis et efficaces. Bien que la tomodensitométrie ait largement contribué à poser des diagnostics pour différentes pathologies, l'utilisation des rayons X présente toujours des inconvénients, tels que les risques tissulaires liés à l'irradiation. De nos jours, une autre technique d'imagerie médicale, appelée imagerie par résonance magnétique (IRM), est utilisée. Contrairement aux rayons X, cette technique exploite les propriétés magnétiques des noyaux atomiques pour obtenir des images détaillées.[4]

1.4.1. Définition d'IRM

L'IRM cérébrale est un examen radiologique spécifique du crâne qui se concentre principalement sur le cerveau. L'IRM, abréviation de l'Imagerie par Résonance Magnétique, est une technique d'imagerie médicale utilisant des ondes de radiofréquences similaires à celles présentes dans les téléphones portables. Ces ondes traversent le corps et sont ensuite analysées informatiquement pour produire des images très claires et détaillées des tissus internes. Actuellement, l'IRM est considérée comme l'examen le plus performant pour explorer et étudier le système nerveux central ainsi que les différentes régions du cerveau. [10].

Figure 1. 1: IRM cérébrale

1.4.2. Principe de l'IRM

Les atomes présents dans le corps diffèrent les uns des autres par le nombre de protons et de neutrons présents dans leur noyau. Certains atomes ont un nombre pair de particules, tandis que d'autres ont un nombre impair. C'est précisément sur les atomes ayant un nombre impair que l'aimantation agit. Actuellement, on parle d'imagerie protonique, car l'atome d'hydrogène, qui ne possède qu'un seul proton, est abondant dans le corps. La détection des autres atomes est plus difficile en raison de leur présence moindre. Ces noyaux, une fois placés dans un champ magnétique (de 0,2 à 1,5 Tesla), et excités par une onde radiofréquence, émettent un signal. Ce signal est ensuite capté par un dispositif informatique qui le transforme en une image. [4]

Tesla[3] : est l'unité de mesure du champ magnétique

1.4.3. Type d'IRM

L'IRM peut être utilisée pour explorer tous les organes, mais l'examen est généralement ciblé sur une partie spécifique du corps pour une analyse précise. Voici quelques exemples d'examens couramment réalisés :

- IRM ostéo-articulaire : pour examiner la colonne vertébrale, les os, les articulations (épaule, genou, cheville, poignet, pied, main), les tissus mous et les muscles.
- IRM neurologique : pour étudier le cerveau, la moelle épinière et les orbites.
- IRM du cou et du thorax.
- IRM abdominale et digestive : pour visualiser le foie, rechercher une surcharge en fer (hémochromatose), examiner les reins et réaliser une entéroIRM.
- IRM des seins : appelée aussi IRM mammaire.
- IRM du pelvis : pour observer l'utérus, les ovaires, les trompes, la vessie et la prostate.
- IRM du périnée : utilisée pour diagnostiquer les prolapsus (descentes d'organes).
- Angio-IRM ou IRM des vaisseaux.

Ces différents examens permettent d'obtenir des images détaillées des structures internes du corps dans les zones ciblées .[11]

1.4.4. La qualité d'image acquise par l'IRM

La qualité des images radiologiques est essentielle pour établir un diagnostic précis. Cependant, plusieurs facteurs peuvent influencer cette qualité. En imagerie par résonance magnétique (IRM), on distingue principalement quatre effets qui peuvent altérer les images : le bruit, le mouvement, les variations de champ et les effets de volume partiel.

Le bruit : Le bruit peut avoir différentes origines, notamment le bruit de l'appareillage lui-même. En imagerie par résonance magnétique (IRM), l'objectif est de maximiser le contraste entre les différents tissus tout en préservant une résolution élevée et un rapport signal/bruit

optimal. Cependant, il existe un compromis entre résolution et bruit, et le choix des paramètres d'acquisition est crucial.

Le mouvement : Le mouvement peut provenir de diverses sources, telles que la circulation sanguine ou la respiration, ainsi que les mouvements du patient pendant l'examen. Le mouvement peut diminuer la qualité de l'image et rendre son interprétation difficile. Par exemple, les mouvements de la tête peuvent causer des artefacts dans les IRM cérébrales.

Les variations du champ magnétique : Les variations du champ magnétique peuvent entraîner des variations d'intensité d'un même tissu dans différentes parties de l'image. Cela est dû au fait que le champ magnétique n'est pas parfaitement homogène dans l'espace et dans le temps lors de l'acquisition. Des méthodes de correction ont été développées pour compenser ces variations et les distorsions induites par le champ magnétique.

Les effets de volume partiel : Les effets de volume partiel sont liés au processus de numérisation du signal. Lorsqu'un pixel chevauche plusieurs objets, son niveau de gris sera une combinaison des niveaux de gris de chaque objet traversé. La prise en compte de ces effets est nécessaire dans les approches de segmentation où l'objectif est de mesurer les différents tissus. Dans certains cas extrêmes, ces effets peuvent entraîner une perte de contraste entre les différents tissus. Il est important de tenir compte de ces facteurs et de mettre en place des stratégies pour minimiser leurs effets indésirables lors de l'acquisition et de l'interprétation des images en IRM (Figure 1.6) . [1]

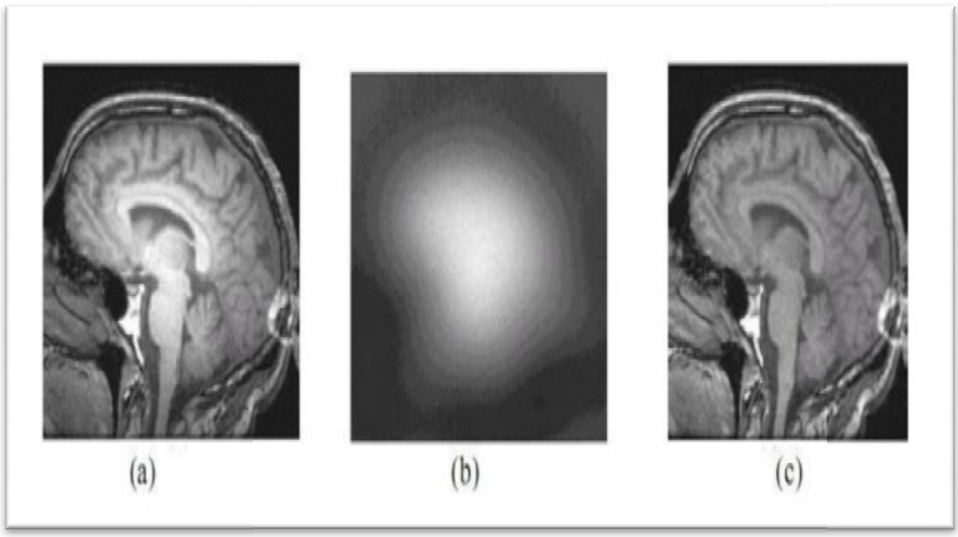


Figure 1. 1: L'inhomogénéité RF

(a) Image affectée par une inhomogénéité RF, (b) L'artefact RF isolé, (c) Image sans artefact.

1.4.5. Y a-t-il un risque ?

L'IRM est un examen sûr, à condition de respecter les contre-indications. Certaines contre-indications sont absolues, ce qui signifie que l'examen ne peut pas être réalisé dans ces cas :

Présence d'un pacemaker (stimulateur cardiaque) ou d'un défibrillateur implantable.

- Présence de clips neuro-chirurgicaux.
- Utilisation d'une pompe à insuline.
- Présence d'un corps étranger métallique intra-orbitaire .
- Grossesse au premier trimestre D'autres contre-indications sont relatives et doivent être discutées avec le personnel de radiologie:
- Présence d'un stent ou d'une endoprothèse vasculaire.
- Présence de valves cardiaques.
- Sutures avec agrafes .
- Utilisation d'une prothèse auditive
- Présence d'éclats métalliques.
- Présence de matériel d'ostéosynthèse
- Présence de prothèses articulaires
- Présence de matériel dentaire

Une insuffisance rénale est une contre-indication à l'injection de produit de contraste (Gadolinium), mais pas à l'IRM elle-même. [12]

1.4.6. Appareillage d'IRM

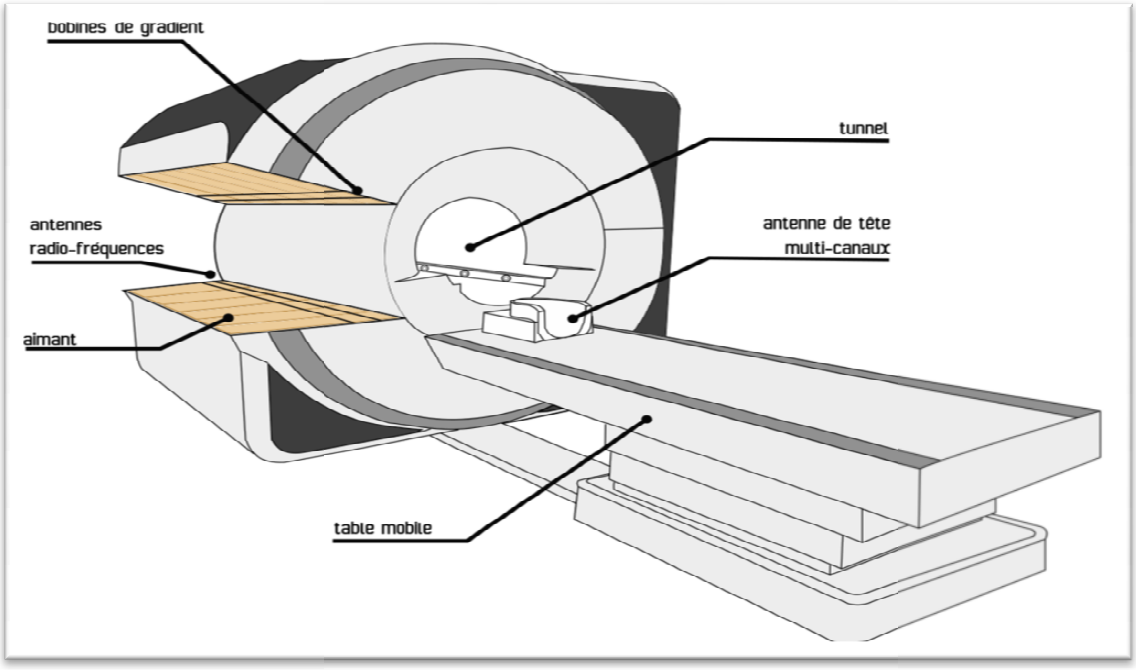


Figure 1. 1: Illustration schématique des principales composantes d'un appareil IRM

L'appareillage de l'IRM est constitués de :

1.4.6.1. L'aimant

Il s'agit d'une bobine qui produit un champ magnétique extrêmement puissant. Cette bobine est immergée dans de l'hélium liquide à une température proche du zéro absolu, ce qui la rend supraconductrice. Cela signifie que le courant électrique qui la traverse ne subit aucune perte d'énergie et peut circuler pendant de très longues périodes. En conséquence, l'aimant de l'IRM reste en fonctionnement en permanence, même lorsque la machine n'est pas utilisée.

1.4.6.2. Les bobine de gradient:

Les bobines de gradient permettent de modifier l'intensité du champ magnétique dans l'espace. Pendant l'acquisition des images, les gradients sont activés et désactivés à plusieurs reprises. Les gradients peuvent être appliqués dans toutes les directions.

1.4.6.3. L'antenne radio-fréquence:

L'équipement utilisé dans l'IRM permet à la fois (1) d'exciter la matière en utilisant des émetteurs et (2) de mesurer la réponse de ces tissus biologiques à l'excitation à l'aide de récepteurs. Les impulsions radiofréquences générées par l'antenne produisent un champ magnétique faible qui est perpendiculaire au champ magnétique principal créé par l'aimant. Les antennes de réception peuvent également être positionnées dans un équipement spécialement conçu pour la tête.[13]

Conclusion

Dans ce chapitre, nous avons couvert divers aspects des tumeurs cérébrales, en abordant les concepts clés liés à l'anatomie du cerveau et aux principales régions cérébrales ayant une importance dans le diagnostic clinique. Nous avons également examiné les principes fondamentaux de l'acquisition d'images médicales. Nous avons constaté que l'imagerie par résonance magnétique (IRM) est une technique d'imagerie médicale extrêmement précieuse pour l'observation du cerveau, fournissant généralement les informations les plus détaillées (avec une résolution spatiale typique de l'ordre du millimètre). Après avoir expliqué les principes physiques qui la sous-tendent, nous avons abordé la formation des images, les différentes séquences disponibles dans cette technique d'imagerie et leur pertinence dans le diagnostic des tumeurs cérébrales, ainsi que les limitations pouvant altérer la qualité de ces images. Dans notre travail, nous nous concentrons spécifiquement sur la détection automatique des tumeurs cérébrales à l'aide de l'apprentissage en profondeur (deep Learning), qui sera discuté

Dans le prochain chapitre.

[12]

CHAPITRE 02

SEGMENTATION DES IMAGES

Introduction

La segmentation est une étape essentielle dans l'analyse d'images, permettant d'identifier les différents objets présents dans une image (organes, os, etc., dans le cas des images médicales) en vue d'une interprétation ou d'un diagnostic. Cependant, la segmentation est considérée comme un problème complexe en raison de la diversité des images existantes de nos jours, qu'il s'agisse de photos numériques, d'images radar, d'images satellites, d'images médicales, etc. Il n'existe actuellement aucune méthode de segmentation efficace pour tous les types d'images. Les chercheurs ont développé de nombreuses méthodes de segmentation adaptées à différents domaines d'application. Dans la suite, nous aborderons le concept de segmentation et examinerons les différentes méthodes existantes. [4]

2.1. Définition de segmentation

Selon le dictionnaire Larousse, la segmentation est définie comme la division d'une unité anatomique en plusieurs éléments. En traitement d'image, la segmentation correspond au processus de partitionnement d'une image numérique en plusieurs régions ou ensembles de pixels distincts. [4]

2.2. Différentes approches de segmentation

Différentes méthodes de segmentation d'image ont été développées en fonction du domaine d'application, et elles peuvent être regroupées en deux catégories principales : les méthodes de segmentation basées sur les régions et les méthodes de segmentation basées sur les contours. Dans la suite, nous présenterons ces deux approches.

2.2.1. Approche contour

Un contour peut être défini comme la limite ou la frontière entre des régions adjacentes dans une image. Les méthodes basées sur les contours visent à délimiter les objets de l'image en se concentrant sur leurs contours. Elles exploitent les variations d'intensité entre les pixels de l'image, où une transition brusque ou un changement significatif indique une frontière entre régions voisines.

L'approche basée sur les contours consiste à identifier ces transitions qui existent entre les régions.

Cependant, les discontinuités dans l'image peuvent être causées non seulement par les structures de l'image elles-mêmes, mais aussi par des facteurs tels que les variations d'éclairage, comme les effets d'ombre. C'est pourquoi les contours ne sont pas toujours continus. Pour remédier à ce problème, des techniques ont été proposées pour obtenir des contours fermés et cohérents.

Il existe plusieurs méthodes utilisant l'approche basée sur les contours. Dans la suite, nous mentionnerons les plus importantes : [4]

- Les méthodes déformables
- Les méthodes dérivatives
- Les méthodes morphologiques

2.2.2. Approche régions

L'approche par segmentation basée sur les régions consiste à découper l'image en différentes régions. Les méthodes appartenant à cette catégorie se basent sur la similarité entre les pixels pour regrouper les régions. Les méthodes de cette approche manipulent directement les régions de l'image, les modifiant itérativement jusqu'à ce que certaines conditions souhaitées soient satisfaites. Certaines méthodes commencent par une partition initiale de l'image, qui est ensuite modifiée en divisant ou en fusionnant des régions, ce qui est appelé des méthodes de décomposition/fusion (ou "split and merge" en anglais). D'autres méthodes commencent avec quelques régions et les font croître par incorporation jusqu'à ce que toute l'image soit couverte, ce qui est appelé des méthodes de croissance de régions. Des méthodes basées sur la modélisation statistique conjointe de la régularité des régions et des niveaux de gris de chaque région existent également.[14]

2.3. Méthodes basées sur la Classification

2.3.1. Méthodes basées sur la Classification Supervisée

Il existe des méthodes où les classes sont préalablement connues avant d'effectuer l'identification des éléments de l'image. Elles nécessitent une phase d'apprentissage sur un échantillon représentatif

afin d'apprendre les caractéristiques de chaque classe, suivie d'une phase de décision pour déterminer l'appartenance d'un individu à une classe spécifique. Dans le cas qui nous intéresse, les données segmentées de l'ensemble d'apprentissage sont obtenues à partir d'un étiquetage manuel des images ou des régions d'intérêt en C classes de tissus (C1...Cc) par un ou plusieurs experts. Chaque classe Ci est associée à un ensemble d'apprentissage Ei, et les données de l'ensemble de test sont segmentées en fonction de ces ensembles Ei. Étant donné que la structure anatomique d'un cerveau peut varier d'un patient à l'autre, l'étiquetage doit être réalisé à nouveau pour chaque patient ou groupe de patients analysés, ce qui représente une tâche longue et fastidieuse pour les spécialistes. [15]

2.3.1.1. Réseaux de Neurones

Un réseau de neurones est une structure composée d'unités élémentaires interconnectées appelées nœuds, ayant des fonctions d'activation linéaires ou non linéaires. Dans les réseaux multicouches, ces nœuds sont regroupés en au moins deux ensembles : un ensemble d'entrée, un ensemble de sortie et éventuellement un ensemble de neurones cachés. Il existe de nombreux modèles de réseaux (tels que les réseaux de Hopfield, les perceptrons multicouches, etc.), où les nœuds sont complètement ou partiellement interconnectés avec les autres. Les connexions entrantes vers un nœud constituent l'ensemble des liens convergeant vers ce nœud, tandis que les connexions sortantes se dirigent vers d'autres nœuds. Chaque connexion entre les nœuds i et j est associée à un poids w, qui représente la force de l'influence du nœud i sur le nœud j. Les poids synaptiques w sont regroupés en un vecteur qui représente l'ensemble des poids.Un exemple est un vecteur de scalaires présenté à tous les nœuds d'entrée. Cet exemple est également associé à des valeurs y (le vecteur de sortie) que l'on souhaite apprendre. Les poids des connexions peuvent être modifiés au cours d'un cycle d'apprentissage.[15]

2.3.1.2. L'algorithme K-plus proches voisines [K-Nearest Neighbor] (KNN)

La méthode des k plus proches voisins (k-nearest neighbor, KNN) est une approche supervisée largement utilisée dans l'estimation statistique et la reconnaissance de modèles. Elle est considérée comme une technique non paramétrique, ce qui signifie qu'elle ne fait aucune hypothèse sur la distribution des données. L'algorithme KNN est l'un des algorithmes les

plus simples en apprentissage automatique. Il appartient à la catégorie de l'apprentissage paresseux (lazy learning), ce qui implique qu'il ne nécessite pas de phase d'entraînement explicite ou très intensive, ce qui le rend rapide à mettre en œuvre.

La méthode KNN repose sur l'hypothèse que les données sont présentées dans un espace de caractéristiques où les points de données sont représentés de manière métrique.. Les données peuvent être des scalaires ou même des vecteurs multidimensionnels. La méthode KNN se base principalement sur l'utilisation de deux paramètres essentiels : une fonction de similarité pour évaluer la comparaison entre les individus dans l'espace des caractéristiques, et la valeur de k qui détermine le nombre de voisins qui ont une influence sur la classification.. Pour évaluer la similarité entre deux vecteurs, la distance est utilisée. Elle mesure le degré de différence entre les vecteurs. Il existe plusieurs types de distances, parmi lesquels :

(1)

Distance euclidienne : x, y sont des vecteurs [16]

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

(2)

La distance de Minkowsky : x, y sont des vecteurs

p : paramètre

$$d(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i| \right)^{1/\mathrm{p}}$$

(3)

La distance de Manhattan :

x, y sont des vecteurs

$$d(x,y) = \sum_{i=1}^{n} |x_i - y_i|$$

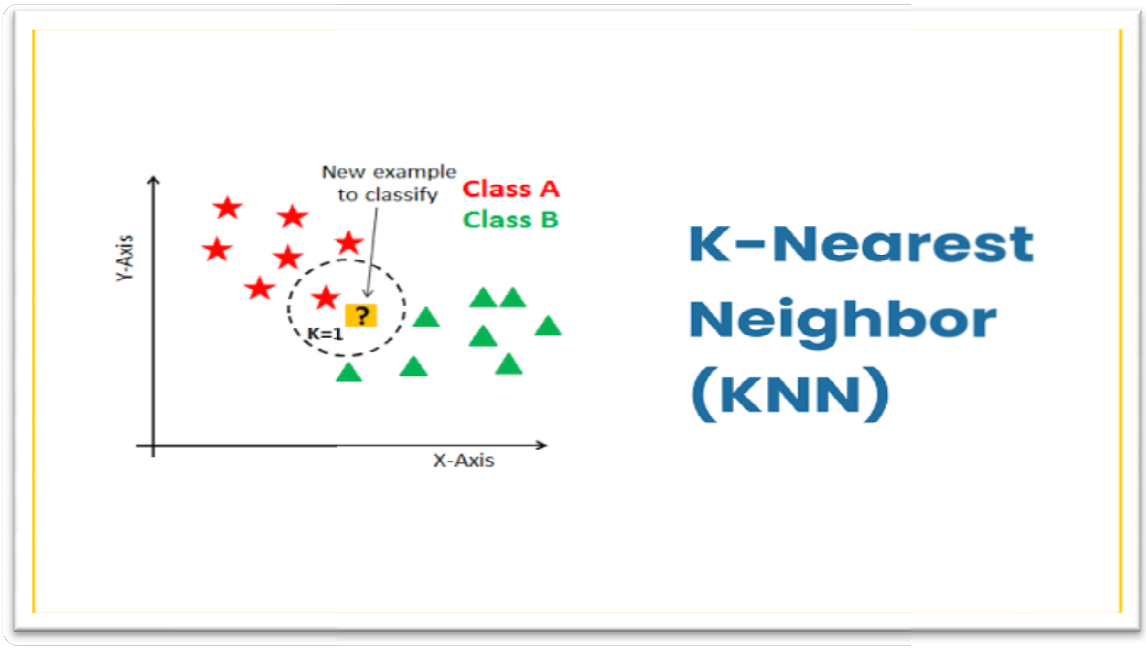


Figure 2.1: L'algorithme K-plus proches voisines [K-Nearest Neighbor]

2.3.2. Méthodes basées sur la Classification non Supervisée

Les méthodes non supervisées présentent l'avantage de ne pas nécessiter de base d'apprentissage ni de tâche préalable d'étiquetage manuel. Leur objectif est de diviser l'espace des individus (pixels) en zones homogènes en se basant sur un critère de ressemblance, c'est-à-dire la proximité des vecteurs d'attributs dans l'espace de représentation entre les individus. Ces méthodes peuvent être une automatisation des algorithmes décrits dans la section précédente (cf. 2.2), où l'ensemble d'apprentissage est remplacé par l'analyse d'une image fournissant des informations complémentaires. Les auteurs utilisent ainsi l'analyse d'une image IRM en tenseurs de diffusion pour extraire des paramètres permettant d'identifier les tissus cérébraux dans l'IRM classique. Notre démarche se concentre plutôt sur la description des algorithmes que nous jugeons pertinents pour notre étude. En particulier, nous développons des méthodes de génération de fonctions d'appartenance des voxels aux classes de tissus. En présentant en détail l'une de ces méthodes, les algorithmes de clustering, nous pourrons ensuite introduire la solution que nous avons développée.[15]

2.3.2.1. Classification par K-Means (K-Moyens)

L'algorithme k-means est largement reconnu et utilisé comme l'algorithme de regroupement en raison de sa facilité de mise en œuvre. Il divise les données d'une image en K clusters. Contrairement à d'autres méthodes hiérarchiques qui créent une structure en "arbre de clusters" pour représenter les regroupements, k-means crée un seul niveau de clusters. L'algorithme génère une partition des données dans laquelle les objets à l'intérieur de chaque cluster sont aussi proches que possible les uns des autres et aussi éloignés que possible des objets des autres clusters. Chaque cluster est défini par ses objets et son centroïde. K-means est un algorithme itératif qui minimise la somme des distances entre chaque objet et le centroïde de son cluster. La position initiale des centroïdes influence le résultat final, il est donc essentiel de les placer initialement le plus éloignés possible les uns des autres pour optimiser l'algorithme. K-means réattribue les objets aux clusters jusqu'à ce que la somme des distances ne puisse plus diminuer. Le résultat est un ensemble de clusters compacts et clairement séparés, à condition que la valeur K appropriée (nombre de clusters) ait été choisie. Les principales

étapes de l'algorithme k-means sont les suivantes : Choix aléatoire de la position initiale des K clusters. Réaffectation des objets à un cluster en fonction d'un critère de minimisation des distances (généralement une mesure de distance euclidienne). Une fois tous les objets placés, recalcule des K centroïdes. Répétition des étapes 2 et 3 jusqu'à ce qu'aucune réaffectation supplémentaire ne soit effectuée. [17]

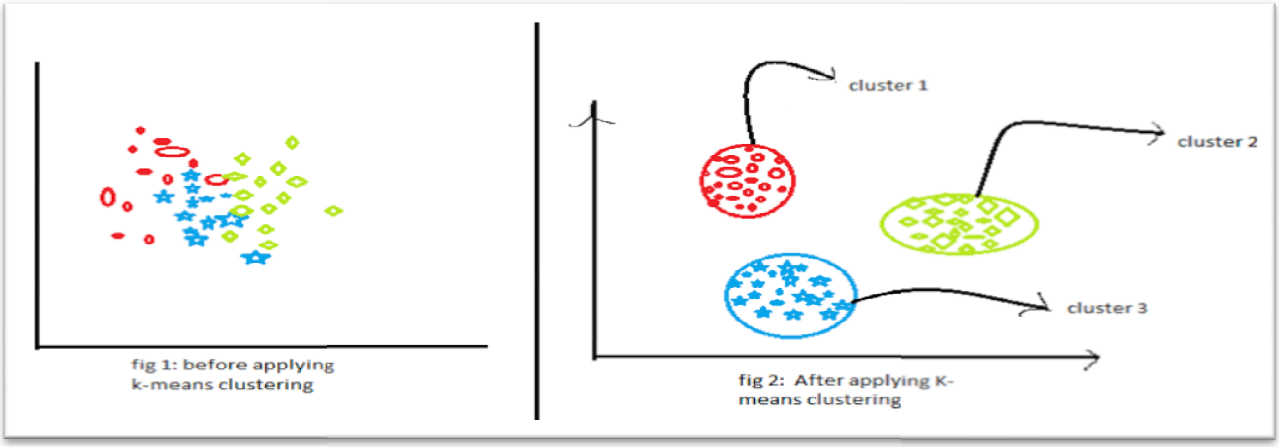


Figure 2.2: Classification par K-Means

2.3.2.2. Classification par C_ moyenne floue (FCM)

L'algorithme FCM est une méthode de classification non supervisée largement reconnue pour sa popularité et ses performances supérieures par rapport à d'autres méthodes telles que l'algorithme K-means ou les réseaux de neurones. Il a été développé par Bezdek en 1981 suite aux travaux de Dunn. L'algorithme FCM est basé sur une approche de réaffectation floue où les classes sont caractérisées par des prototypes ou des centres de gravité. Son application attribue un degré d'appartenance (compris entre 0 et 1) à chaque classe pour chaque observation à classifier, générant ainsi une partition floue. Comme la plupart des autres algorithmes de classification par partition, le FCM repose sur la minimisation d'un critère à travers un processus itératif.[18]

2.3.2.2. Classification par K_moyenne possibiliste(PCM)

L'introduction de Krishnapuram et Keller consiste en une approche possibiliste des C-moyennes connue sous le nom de Possibilistic C-Means (PCM).. Leur approche vise à améliorer les performances en présence de bruit, mais leur motivation principale est de résoudre le problème relatif des degrés d'appartenance générés par les FCM. En

effet, ces degrés d'appartenance sont interprétés comme des degrés de vérité relatifs décrivant l'appartenance d'un vecteur à chaque classe possible, ce qui implique une certaine forme de partage entre ces classes. Au lieu de cela, Krishnapuram et Keller préconisent de remplacer cette idée de partage par la notion de typicalité. Ils soutiennent que le résultat d'un regroupement devrait plutôt représenter la parenté absolue entre un objet et chacune des c classes possibles, indépendamment des autres classes restantes (c-1). [15]

2.4. Méthodes intelligentes

L'intelligence artificielle (IA), également connue sous le nom d'Intelligence Artificielle en anglais (AI pour Artificial Intelligence), englobe diverses techniques utilisées pour permettre aux machines de simuler une forme d'intelligence humaine. Son application s'étend à de nombreux domaines en constante expansion. L'idée de l'IA émerge dans les années 1950 grâce aux travaux du mathématicien Alan Turing. Dans son ouvrage "Computing Machinery and Intelligence", Turing pose la question de conférer aux machines une forme d'intelligence. Il présente alors un test, connu aujourd'hui sous le nom de "Test de Turing", dans lequel un individu interagit de manière anonyme avec un autre humain, puis avec une machine programmée pour fournir des réponses cohérentes. Si l'individu ne peut pas faire la distinction, la machine est considérée comme ayant passé le test et, selon Turing, peut être réellement considérée comme "intelligente". [19]

2.4.1. La Médecine et l'intelligence Artificielle

L'intelligence artificielle (IA) en médecine se réfère à l'utilisation de modèles d'apprentissage automatique pour explorer les données médicales et découvrir des informations précieuses afin d'améliorer les résultats en matière de santé et l'expérience des patients. Grâce aux avancées récentes en informatique et en sciences de l'information, l'IA joue un rôle de plus en plus essentiel dans les soins de santé modernes. Les algorithmes d'IA et autres applications optimisées par l'IA sont utilisés pour assister les professionnels de la santé dans des environnements cliniques et dans le cadre de recherches en cours. Actuellement, les principales utilisations courantes de l'IA dans le domaine médical sont l'assistance à la prise de décision clinique et l'analyse d'images.. Les outils d'aide à la décision clinique aident les fournisseurs de soins de santé à prendre des décisions éclairées concernant les traitements, les médicaments, la santé mentale et d'autres besoins des patients, en leur fournissant un accès rapide aux

informations pertinentes ou aux recherches appropriées pour leur patient. Les outils d'intelligence artificielle dans le domaine de l'imagerie médicale sont employés pour examiner les tomodensitogrammes, les radiographies, les IRM et d'autres types d'images, afin de repérer des lésions ou d'autres éléments pouvant échapper à la détection d'un radiologue humain. Les recherches et les études sur ces applications sont encore en cours, et les normes d'utilisation de l'intelligence artificielle en médecine sont en phase de développement. . Cependant, les possibilités offertes par l'IA pour aider les cliniciens, les chercheurs et les patients qu'ils servent ne cessent de croître. Il est évident à ce stade que l'intelligence artificielle deviendra un élément essentiel des systèmes de santé numériques qui influencent et soutiennent la médecine moderne .[20]

2.4.2. L'Imagerie Médicale et L'Intelligence Artificielle

L'intelligence artificielle (IA) occupe déjà une position centrale dans le domaine de l'imagerie médicale. Des études ont démontré que l'IA, grâce à des réseaux neuronaux artificiels, peut être aussi performante que les radiologues humains pour détecter les signes de tumeurs et d'autres maladies. En plus d'aider les médecins à repérer précocement les signes de pathologies, l'IA peut également faciliter la gestion du volume considérable d'images médicales en identifiant les éléments essentiels de l'historique d'un patient et en leur présentant les images pertinentes. [20]

À partir des années 90, l'analyse d'images par intelligence artificielle (IA) a connu une expansion significative grâce à l'introduction d'algorithmes d'apprentissage profond basés sur les réseaux de neurones et à l'amélioration de la puissance de calcul des ordinateurs. Ces méthodes, notamment efficaces pour la détection d'objets, ont ouvert la voie à des avancées majeures. Dans le domaine médical, les applications cliniques de ces approches d'IA sur les différentes modalités d'imagerie sont nombreuses et présentent un fort intérêt pour la détection, la reconstruction d'images et le développement de biomarqueurs d'imagerie afin d'aider au diagnostic, à l'évaluation pronostique et à la prédiction de la réponse aux traitements. Le terme de "radiomics", introduit par Gillies et al. En 2010, le terme "radiomics" a été introduit pour décrire le domaine de l'imagerie médicale qui applique des techniques d'intelligence artificielle au processus de traduction informatique de l'imagerie en données quantitatives de haute dimension, similairement aux autres domaines "omics". Ce domaine est

axé sur le développement de biomarqueurs d'imagerie et représente l'un des domaines de l'imagerie où les techniques d'intelligence artificielle sont largement utilisées. Bien que de nombreuses recherches en soient encore au stade expérimental, de nombreuses avancées sont extrêmement prometteuses. [21]

2.4.3. Solution d'Intelligence Artificielle pour le diagnostic des tumeurs

Les avancées technologiques liées à l'intelligence artificielle touchent de nombreux domaines d'application tels que la médecine, les transports, la cybersécurité, le commerce et l'industrie, et leur intégration dans notre quotidien s'accélère rapidement. Ces technologies révolutionnent des aspects entiers de notre vie quotidienne, ouvrant des perspectives jusqu'alors inimaginables. Face à ces changements profonds, les États reconnaissent l'IA comme un outil stratégique majeur, mais ils sont confrontés à l'absence d'un cadre juridique approprié pour faire face aux défis complexes de gouvernance soulevés par ces technologies d'intelligence artificielle. Cette situation incite de nombreux États et institutions à proposer une régulation plus solide dans ce domaine et à prendre en compte les questions éthiques qui accompagnent cette transformation. [22]

Ce projet ambitieux consiste à développer 15 outils d'intelligence artificielle pour la pathologie numérique à des fins de diagnostic et de traitement. L'utilisation de modèles d'apprentissage automatique (Deep learning) permettra d'analyser les données des lames de pathologie numériques (images numérisées d'échantillons de tissus de patients) afin de créer des outils de diagnostic plus efficaces et accessibles, facilitant ainsi la personnalisation des traitements par les oncologues dès les premiers stades de la maladie. Le projet PortrAIt vise également à accélérer le travail des pathologistes en détectant la présence de biomarqueurs spécifiques et en prédisant l'évolution des patients. Tous ces outils numériques seront regroupés au sein d'une plateforme évolutive et interopérable, offrant un environnement de recherche et de collaboration aux partenaires de PortrAIt, en vue de leur test et déploiement dans les centres de cancérologie et laboratoires français. En 2019, Gustave Roussy et Owkin ont signé un accord-cadre pour un partenariat de recherche et une collaboration à long terme, donnant ainsi naissance à RelaspRisk BC, la première solution pronostique d'intelligence artificielle basée sur la pathologie numérique permettant de prédire le risque de rechute chez les patientes atteintes de cancer

cérébral localisé, afin d'éviter des chimiothérapies inutiles après la chirurgie .[23]

2.4.4. Avantages de L'IA en médecine

2.4.4.1. Soins aux patients en toute connaissance de cause

L'incorporation de l'IA médicale dans les processus de travail des médecins permet de fournir aux praticiens un contexte précieux lorsqu'ils prennent des décisions concernant les soins à prodiguer. Cela leur permet d'accéder à des informations factuelles sur les traitements et les procédures, même lorsque le patient est encore présent dans leur cabinet.

2.4.4.2. Réduction des erreurs

Il est évident que l'IA joue un rôle essentiel dans l'amélioration de la sécurité des patients. Une analyse approfondie de 53 études évaluées par des pairs, portant sur l'impact de l'IA sur la sécurité des patients, a démontré que les outils d'aide à la décision optimisés par l'IA peuvent considérablement améliorer la détection des erreurs et la gestion des médicaments.

2.4.4.3. Réduction du coût des soins

L'intelligence artificielle offre de nombreuses possibilités pour réduire les coûts dans le domaine de la santé. Parmi celles-ci, on trouve la diminution des erreurs de médication, l'assistance médicale virtuelle personnalisée, la prévention des fraudes et l'amélioration des flux de travail administratifs et cliniques. Toutes ces opportunités sont particulièrement prometteuses.

2.4.4.4. Accroître l'engagement du médecin-patient

De nombreux patients se posent des questions en dehors des horaires d'ouverture habituels. L'intelligence artificielle peut offrir une assistance 24 heures sur 24 grâce à des agents conversationnels capables de répondre aux questions de base et de fournir aux patients des ressources lorsque les établissements de santé ne sont pas ouverts. De plus, l'IA peut être utilisée pour trier les questions et signaler les informations nécessitant un examen approfondi, permettant ainsi d'alerter les prestataires de soins sur les changements de santé nécessitant une attention particulière.

2.4.4.5. Pertinence contextuelle

L'un des atouts majeurs de l'apprentissage en profondeur réside dans la capacité des algorithmes d'IA à utiliser le contexte pour différencier différentes informations. Par exemple, dans une note clinique qui inclut à la fois une liste des médicaments actuels d'un patient et une recommandation d'un nouveau médicament par le fournisseur de soins, un algorithme d'IA bien entraîné peut utiliser le traitement du langage naturel pour identifier les médicaments qui doivent être ajoutés aux antécédents médicaux du patient. [20].

2.5. L'apprentissage approfondi

2.5.1. Définition

Le deep Learning, également connu sous le nom d'apprentissage profond, est un sous-domaine du machine Learning et de l'intelligence artificielle (IA). Il se caractérise par l'utilisation d'algorithmes qui imitent le fonctionnement du cerveau humain à l'aide d'un vaste réseau de neurones artificiels. Ces systèmes sont ainsi capables d'apprendre, de prédire et de prendre des décisions de manière autonome. [24]

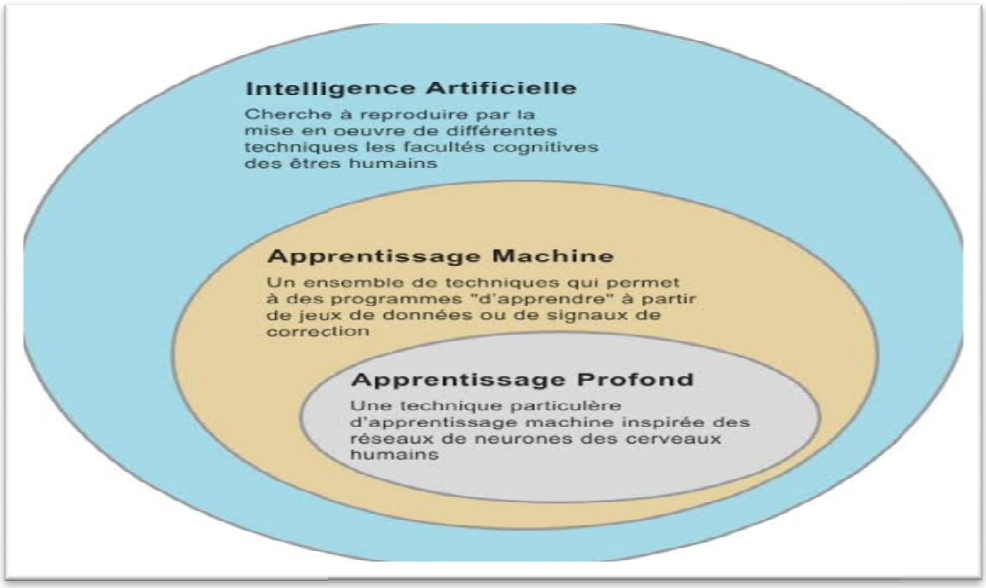


Figure 2.3: Définition de deep Learning

2.5.2. L'importance de l'apprentissage profond (DL) :

L'apprentissage automatique fonctionne exclusivement avec des ensembles de données structurées et semi-structurées, tandis que

l'apprentissage en profondeur peut traiter à la fois des données structurées et non structurées.

- Les algorithmes d'apprentissage en profondeur sont capables d'effectuer des opérations complexes de manière efficace, contrairement aux algorithmes d'apprentissage automatique.
- Les algorithmes d'apprentissage automatique utilisent des échantillons de données étiquetées pour extraire des modèles, tandis que l'apprentissage en profondeur accepte de grandes quantités de données en entrée et analyse ces données pour en extraire les caractéristiques d'un objet.
- Les performances des algorithmes d'apprentissage automatique diminuent à mesure que le volume de données augmente, ce qui nécessite l'utilisation de l'apprentissage en profondeur pour maintenir les performances du modèle. [12]

2.5.3. L'apprentissage supervisé

L'apprentissage supervisé, également connu sous le nom d'apprentissage avec supervision, est le paradigme d'apprentissage le plus largement utilisé dans le domaine du Machine Learning et du Deep Learning. Comme son nom l'indique, il implique de superviser l'apprentissage de la machine en lui fournissant des exemples de la tâche qu'elle doit accomplir, c'est-à-dire des données étiquetées. Ce type d'apprentissage trouve de nombreuses applications dans des domaines tels que la reconnaissance vocale, la vision par ordinateur, les régressions et les classifications. La plupart des problèmes liés à l'apprentissage automatique et à l'apprentissage profond font appel à l'apprentissage supervisé, il est donc crucial de bien saisir son fonctionnement [25].

2.5.3.1. Comment fonctionne l'apprentissage supervisé ?

L'apprentissage supervisé permet à la machine d'acquérir des compétences dans une tâche spécifique en étudiant des exemples de cette tâche. Par exemple, en lui présentant des millions de photos de chiens, elle peut apprendre à reconnaître une photo de chien. De même, en exposant la machine à des millions d'exemples de traductions du français vers le chinois, elle peut apprendre à traduire efficacement du français au chinois . [25]

En général, la machine est capable d'apprendre une relation

f : x ⟶ y qui relie x à y en ayant analysé des millions d'exemples d'associations x ⟶ y.

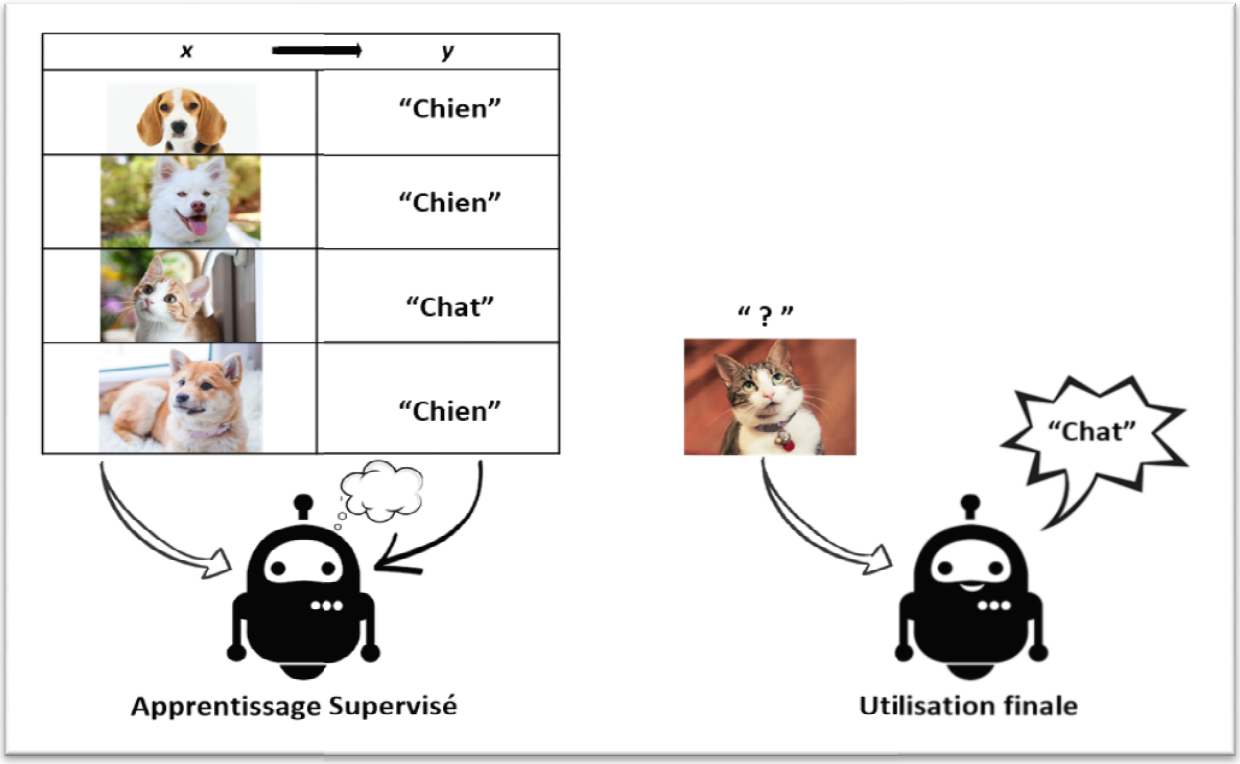


Figure 2.4: l'apprentissage supervisé

L'apprentissage supervisé se déroule en quatre étapes :

1. Importer un ensemble de données (x, y) qui contient nos exemples.
2. Développer un modèle avec des paramètres aléatoires.
3. Élaborer une fonction de coût qui évalue les erreurs entre le modèle et l'ensemble de données.
4. Mettre en place un algorithme d'apprentissage pour trouver les paramètres du modèle qui minimisent la fonction de coût. [25]

2.5.4. L'apprentissage non supervisé

L'apprentissage non supervisé, également appelé apprentissage non étiqueté, est une technique qui implique l'entraînement de modèles sans avoir à étiqueter manuellement ou automatiquement les données au préalable. Les algorithmes organisent les données en groupes en se basant sur leurs similarités, sans nécessiter l'intervention de l'humain. [26]

2.5.4.1. Comment fonctionne l'apprentissage non supervisé ?

L'apprentissage non supervisé, également appelé apprentissage non étiqueté, est une technique qui implique l'entraînement de modèles sans avoir à étiqueter manuellement ou automatiquement les données au préalable. Les algorithmes organisent les données en groupes en se basant sur leurs similarités, sans nécessiter l'intervention de l'humain. [26]

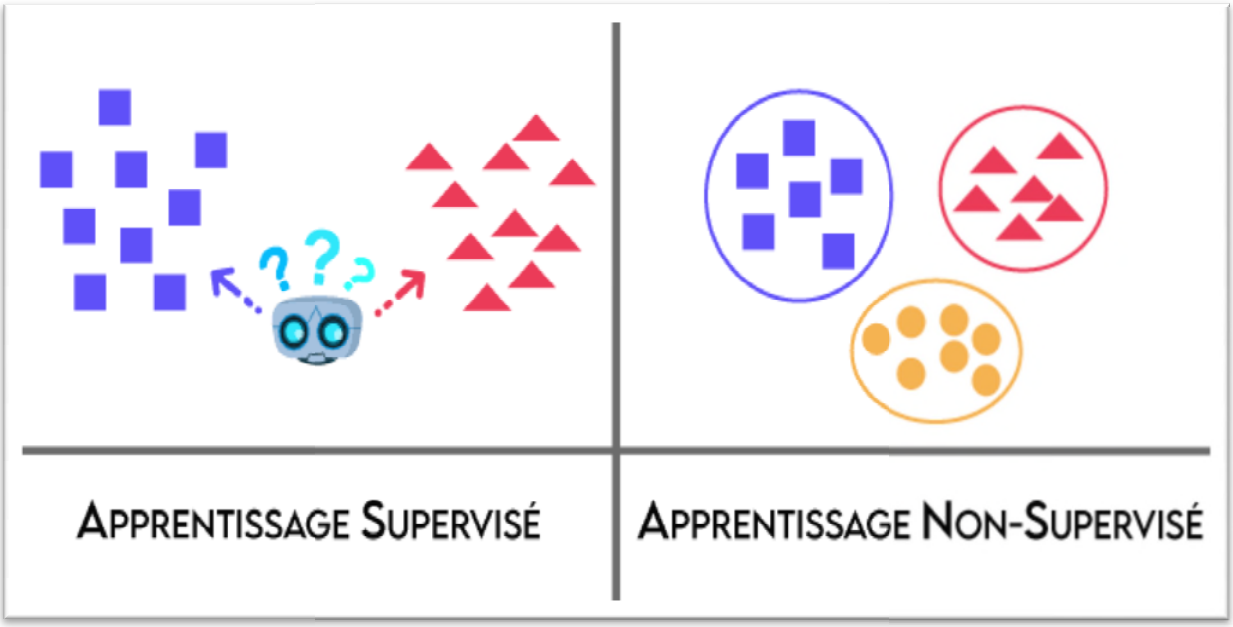


Figure 2.5: L'apprentissage non supervisé

2.5.5. L'apprentissage semi- supervisé

L'apprentissage semi-supervisé peut être considéré comme une approche qui combine les principes de l'apprentissage non supervisé et de l'apprentissage supervisé. Dans cette méthode, nous disposons généralement d'un petit ensemble de données étiquetées et d'un ensemble beaucoup plus grand de données non étiquetées. L'apprentissage semi-supervisé consiste à entraîner un modèle d'apprentissage en utilisant à la fois les données étiquetées et les données non étiquetées. L'idée principale est d'utiliser la similarité entre les données étiquetées et les données non étiquetées pour attribuer des étiquettes aux données non annotées. Ainsi, l'apprentissage semi-supervisé se situe quelque part entre l'apprentissage supervisé, qui ne se base que sur des données étiquetées, et l'apprentissage non supervisé, qui n'utilise que des données non étiquetées. Cette approche permet

d'améliorer considérablement les résultats sans avoir besoin de recourir à l'annotation manuelle fastidieuse, coûteuse et chronophage. [27]

2.5.6. L'apprentissage renforcé

L'apprentissage par renforcement est une approche où un agent autonome (par exemple, un robot, un agent conversationnel, un personnage de jeu vidéo, etc.) apprend à prendre des actions en fonction de l'expérience acquise afin d'optimiser une récompense quantitative sur une période de temps. L'agent interagit avec son environnement et prend des décisions en fonction de son état actuel. En retour, l'environnement offre à l'agent une récompense, pouvant être positive ou négative. L'objectif de l'agent est d'apprendre, par le biais d'expériences répétées, une stratégie décisionnelle optimale (appelée politique), qui associe à chaque état une action à exécuter, de manière à maximiser la somme des récompenses au fil du temps.. [28]

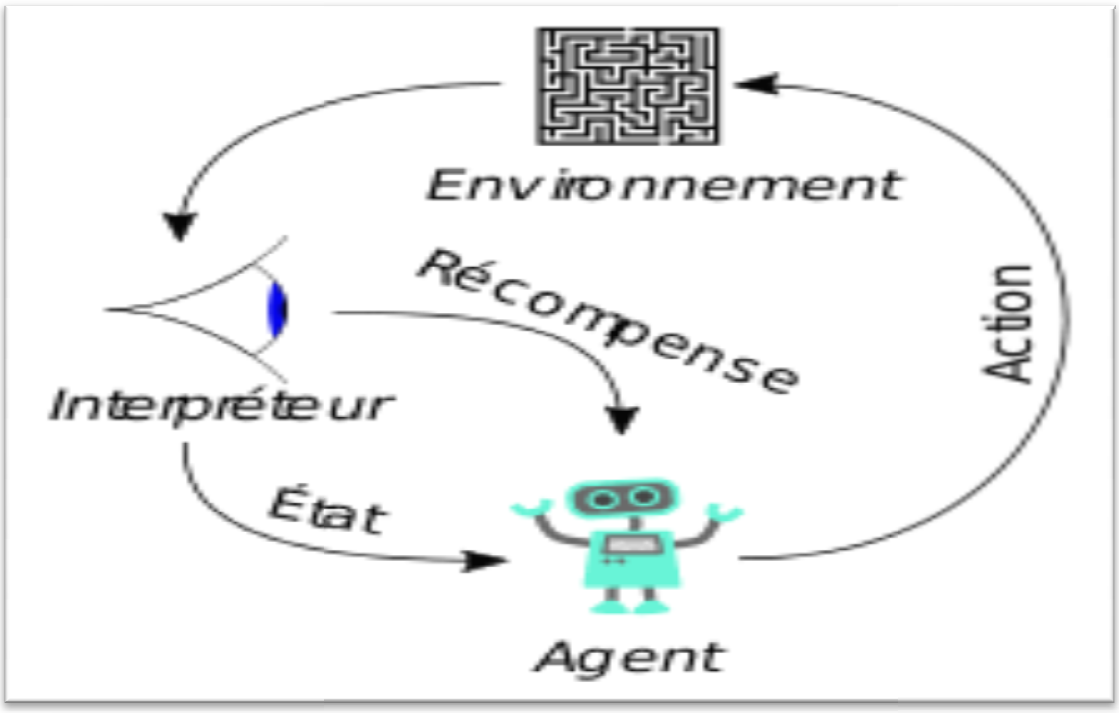


Figure 2.6: L'apprentissage renforcé

2.5.7. Réseau de neurones covolutifs (CNN)

Les réseaux de neurones convolutifs (CNN) sont des modèles spécialisés dans le traitement de données, présentant une structure en

grille. Ils se sont révélés extrêmement efficaces dans de nombreux domaines, tels que la reconnaissance et la classification d'images et de vidéos. Ils sont utilisés pour l'identification de visages, d'objets, de panneaux de signalisation et pour la conduite autonome des véhicules. Dans le domaine de l'apprentissage automatique (ML), les réseaux convolutifs sont une variante des réseaux de neurones feed-forward, et leur conception s'inspire des processus biologiques. Les CNN sont caractérisés par quatre principales opérations qui les composent, illustrées dans leur structure:

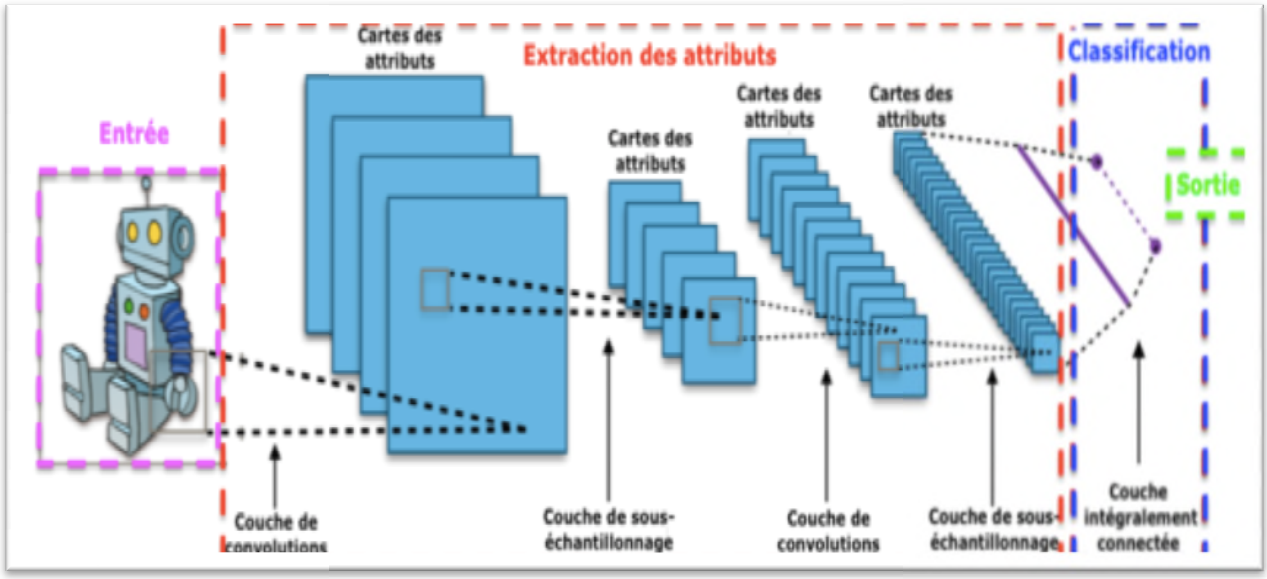


Figure 2.7: Réseau de neurones covolutifs (CNN)

2.5.7.1. La couche convolution

La couche de convolution est un élément essentiel des réseaux de neurones convolutifs, et elle est généralement présente en tant que première couche. Les couches de convolution sont constituées de filtres, qui sont des tableaux de valeurs appelés featuremaps. Chaque couche de convolution prend une image en entrée et produit une featuremap. Chaque featuremap est obtenue en appliquant le filtre à l'image. Par exemple, si l'image a une taille de 5x5 et que le filtre est de taille 3x3, la featuremap aura une taille de 3x3. La couche de convolution reçoit donc plusieurs images en entrée et calcule la convolution de chacune d'entre elles avec chaque filtre. Les filtres sont spécifiquement conçus pour

détecter les caractéristiques que l'on souhaite retrouver dans les images. [29]

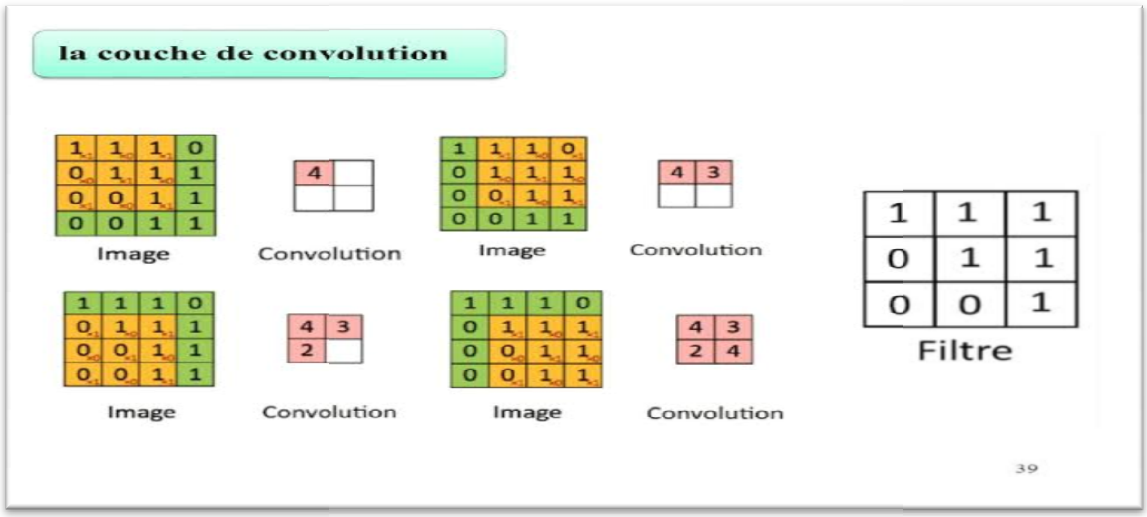


Figure 2.8: La couche convolution

2.5.7.2. La couche Rectified Linear Unit

ReLU (Rectified Linear Units) désigne la fonction réelle non-linéaire définie par ReLU(x)=max (0, x).

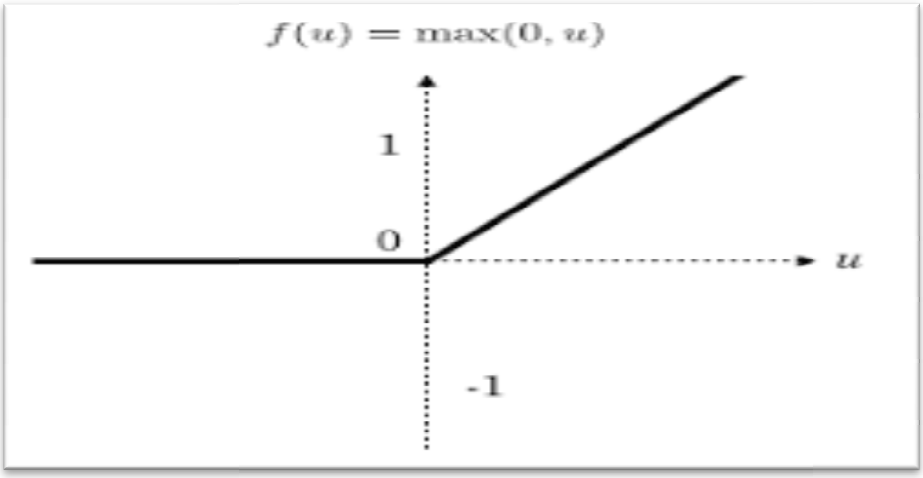


Figure 2.9: Fonction d'activation ReLU Fonction d'activation ReLU

La fonction de correction ReLU est utilisée pour remplacer toutes les valeurs négatives en entrée par des zéros. Elle agit en tant que fonction d'activation dans le réseau. [30]

2.5.7.3. La couche pooling

La couche de pooling est généralement positionnée entre deux couches de convolution. Elle prend en entrée plusieurs featuremaps et applique l'opération de pooling à chacune d'entre elles. Le pooling est une opération de réduction qui divise l'image en blocs et conserve uniquement la valeur maximale de chaque bloc. Cela permet de réduire la taille de l'image tout en préservant les caractéristiques les plus importantes. En sortie, on obtient le même nombre de featuremaps qu'en entrée, mais elles sont considérablement réduites en taille .[29]

Figure2.10: Couche pooling (Max pooling)

2.5.7.4. La couche entièrement connectée

Après plusieurs couches de convolution et de max-pooling, les processus de raisonnement de haut niveau dans le réseau neuronal sont effectués via des couches entièrement connectées. La couche entièrement connectée, également connue sous le nom de couche Multi Layer Perceptron (MLP), est utilisée pour identifier les motifs distincts et en déduire la classe correspondante. Cette couche est caractérisée par une matrice de connexion entièrement connectée, où chaque neurone de la couche précédente est connecté à chaque neurone de la couche suivante. La couche entièrement connectée utilise des fonctions d'activation, telles que la fonction softmax dans la couche de sortie. Son objectif est de classer l'image d'entrée dans différentes classes en fonction des données d'apprentissage fournies. [29]

2.5.8. Modèle VGG16

VGG16 est un modèle de réseau de neurones convolutifs développé par K. Simonyan et A. Zisserman. Les détails de son implémentation sont décrits dans le document intitulé "Very Deep Convolutional Networks for Large-Scale Image Recognition". Ce modèle a obtenu une précision de test de 92,7% sur ImageNet, une base de données comprenant plus de 14 millions d'images réparties en 1000 classes. Le choix de VGG16 s'explique par sa structure profonde composée de 16 couches : [31]

Figure 2. 11: VGG-16 architecture

Les deux premières couches convolutionnelles sont constituées de 64 filtres de caractéristiques avec une taille de noyau de 3x3. Lorsque l'image d'entrée, qui est une image RVB avec une profondeur de 3, est passée à travers ces deux couches, les dimensions deviennent 224x224x64. Ensuite, la sortie est transmise à une couche de pooling maximal avec un pas de 2. Les troisième et quatrième couches convolutionnelles sont composées de 128 filtres de caractéristiques avec une taille de noyau de 3x3. Après ces deux couches, une autre couche de pooling maximal est appliquée avec un pas de 2, réduisant ainsi les dimensions à 56x56x128. Les cinquième, sixième et septième couches sont des couches convolutionnelles avec un noyau de taille 3x3, utilisant chacune 256 cartes de caractéristiques. Ces couches sont suivies d'une

couche de pooling maximal avec un pas de 2. Les huitième à treizième couches forment deux ensembles de couches convolutionnelles avec un noyau de taille 3x3. Chacun de ces ensembles contient 512 filtres de caractéristiques. Après ces couches, une couche de pooling maximal est appliquée avec un pas de 1. Les quatorzième et quinzième couches sont des couches entièrement connectées cachées avec 4096 unités, suivies d'une couche de sortie softmax (seizième couche) avec 1000 unités. [32]

2.6. Evaluation de la segmentation des images

L'évaluation de la segmentation repose généralement sur des critères qualitatifs et quantitatifs. Cette évaluation est essentielle car elle permet aux chercheurs de :

- Comparer la méthode de segmentation avec d'autres méthodes.
- Sélectionner l'algorithme le plus approprié pour leur application et ajuster ses paramètres en fonction du problème à résoudre. Le paramétrage des algorithmes est un défi bien connu pour les chercheurs.
- Pour une comparaison objective des méthodes de segmentation, il est préférable d'utiliser des images de synthèse où les caractéristiques et les zones sont parfaitement localisées. Cependant, ces images de synthèse présentent l'inconvénient de ne pas reproduire toutes les situations possibles et de ne pas refléter suffisamment la réalité. [2]

2.6.1. Evaluation non-supervisée

L'évaluation des résultats de segmentation non supervisée consiste à mesurer l'homogénéité des régions ou les contrastes entre les régions. Ces critères d'évaluation peuvent être classés en deux grandes catégories :

- Les critères de contraste : ils se concentrent sur la variabilité entre les régions.
- Les critères d'adéquation au modèle : ils évaluent l'uniformité en termes d'intensité ou de couleur à l'intérieur des régions. [2]

2.6.2. Évaluation supervisée

L'évaluation supervisée implique la comparaison entre le résultat produit par un algorithme et une segmentation de référence (appelée vérité-terrain). La segmentation de référence peut être une image de synthèse ou une image créée manuellement par un expert dans le domaine d'application, qui marque les contours (par exemple, par des médecins, des géographes, etc.), en utilisant des outils de dessin informatiques. Dans le contexte de la segmentation de l'IRM cérébrale, de nombreuses images avec des fragments de référence sont fournies par divers laboratoires, et les critères basés sur le recouvrement avec la référence sont généralement privilégiés. Deux types d'images peuvent être utilisés. :[2]

a) Des images simulées (fantômes)

Les images utilisées sont générées en simulant le processus d'acquisition d'une image IRM à partir d'un modèle anatomique réaliste. Ainsi, il est possible de comparer les résultats d'un algorithme de segmentation avec le modèle anatomique sous-jacent. Les bases de données simulées, telles que BrainWeb 1 fournies par le McConnell Brain Imaging Center, sont largement utilisées pour évaluer les performances des algorithmes de segmentation d'IRM cérébrales.

b) Des images réelles

Les images sont segmentées manuellement par un expert, ce qui présente l'avantage de se baser sur des acquisitions réelles représentant la réalité. Cependant, la segmentation manuelle est une tâche laborieuse et sujette à une variabilité significative tant entre les experts qu'au sein d'un même expert.Après avoir obtenu le résultat de la segmentation, qu'il provienne d'images réelles ou simulées, il est ensuite évalué en le comparant à une segmentation de référence à l'aide d'une mesure de similarité.. Plusieurs mesures ont été proposées, dont le coefficient de Dice (Dice, 1945). Le coefficient de Dice, également appelé indice Kappa, permet de quantifier la similitude entre deux régions. Numériquement, en utilisant le nombre de vrais positifs (VP_K), de faux positifs (FP_K) et de faux négatifs (FN_K) pour la classe k, le coefficient

de Dice dk pour la classe k est donné par l'équation suivante : d_k=(2VP_K)/(2VP_K+FN_K+FP_K) Le coefficient de Dice varie entre 0 et 1, où une valeur de 1 indique une segmentation parfaite. Une autre mesure couramment utilisée est la précision (accuracy), qui permet d'évaluer les performances du modèle de la manière suivante : Précision=Nombre d'images correctement prédites/Nombre total d'images testées×100% .[2]

2.7. Diagnostic assiste par ordinateur

Un système d'aide au diagnostic médical, également connu sous le nom de diagnostic assisté par ordinateur, est utilisé pour soutenir le processus de diagnostic médical. Il est composé d'une base de données et d'un moteur de recherche qui permet d'accéder aux informations contenues dans la base de données. Ce système peut proposer des diagnostics différentiels en fonction des données préalablement renseignées après un examen clinique, fournir des estimations pronostiques ou signaler des informations manquantes nécessaires à l'établissement d'un diagnostic. [33]

Selon l'orientation de l'examen, une IRM de la tête peut comprendre des centaines à des milliers d'images individuelles dans différentes directions spatiales. Dans le domaine de la médecine courante, l'interprétation des données fournies par les ensembles de données médicales IRM repose généralement sur l'expertise et l'expérience du spécialiste, utilisant le principe de l'"opinion d'expert". Ce processus est généralement suffisant pour évaluer la plupart des problématiques de neuroradiologie, telles que la recherche, la description et l'évaluation des lésions focales dans la pratique clinique quotidienne. Cependant, il devient plus difficile lorsque des maladies ne sont détectables que par des modifications volumétriques subtiles ou des perturbations de signaux, dépassant souvent les capacités de perception humaine. C'est pourquoi un développement relativement récent se concentre sur l'exploration de ces ensembles de données à l'aide de méthodes assistées par ordinateur. Cela peut offrir des avantages significatifs en fournissant aux cliniciens des données

quantifiées sur des mesures spécifiques telles que le diamètre cortical régional, les modifications volumétriques associées aux tumeurs cérébrales ou à la sclérose en plaques, ou encore les concentrations de lactate dans les tissus cérébraux des patients atteints de maladies mitochondriales. [34]

- **Morphométrie cérébrale**

Le domaine relativement récent de la morphométrie cérébrale, qui vise à quantifier la forme et le volume du cerveau à l'aide de mesures, se divise en deux méthodes distinctes : l'analyse basée sur les voxels et l'analyse basée sur les surfaces. La première méthode est adaptée pour mesurer les concentrations et les volumes régionaux des différents tissus cérébraux, tandis que la morphométrie basée sur les surfaces offre des avantages pour capturer des propriétés telles que la configuration des surfaces, l'épaisseur et la formation des gyrus du cortex. En utilisant un groupe témoin, il est possible de mesurer l'épaisseur corticale régionale, et grâce à la morphométrie basée sur les voxels, des mesures inaccessibles auparavant, telles que la surface corticale régionale, la courbure du cortex et la profondeur des sillons, peuvent être déterminées. Le groupe témoin permet de comparer statistiquement les mesures morphométriques des patients à celles d'un grand nombre de sujets sains, obtenues à l'aide des mêmes scanners. Les influences physiologiques, telles que la taille de la tête, les différences entre les sexes ou les variations normales de la substance grise liées à l'âge (augmentation chez les enfants, diminution avec l'âge), sont prises en compte dans ces analyses. [34]

- **Les problématiques oncologiques**

La neuroradiologie quantitative joue un rôle important dans la compréhension des problèmes oncologiques. Des mesures unidimensionnelles ou bidimensionnelles de la taille de la lésion, ainsi que des données tridimensionnelles dans les plans principaux, sont souvent utilisées dans ce domaine. Cependant, les mesures bidimensionnelles peuvent présenter des erreurs importantes, dues à la variation entre les examinateurs et les différentes positions du patient,

pouvant entraîner des variations de plus de 50%. Par conséquent, la volumétrie manuelle réalisée par un expert est considérée comme la méthode de référence, bien qu'elle soit précise mais longue. La délimitation visuelle de la tumeur par rapport aux tissus non affectés peut également poser des difficultés considérables. Les méthodes d'analyse assistées par ordinateur offrent des avantages dans ce contexte. Une fois qu'une classification a été effectuée, les classes tissulaires sont immédiatement reconnues, ce qui permet d'obtenir des résultats stables au fil du temps, même en présence d'erreurs systématiques de classification. Les méthodes fondamentales incluent des méthodes d'analyse génératives, qui utilisent des modèles pré-définis basés sur la neuroanatomie et la présentation tissulaire, ainsi que des méthodes discriminatoires, où les caractéristiques typiques des classes tissulaires peuvent être reconnues et apprises par le programme informatique à l'aide d'un ensemble de données d'entraînement. Une quantité importante de données pour l'apprentissage permet d'améliorer la précision de la prédiction du type tissulaire. Par exemple, un algorithme discriminatoire appelé BraTumIA, développé par Bauer et al. à l'Université de Berne, a montré une précision de la volumétrie tumorale comparable à celle des experts cliniques lors de premiers essais cliniques. Ces classifications tissulaires peuvent également être utilisées pour estimer le pronostic et personnaliser le traitement (« radiomique »), ainsi que pour être analysées avec des profils génétiques (« radiogénomique »). La spectroscopie par résonance magnétique (SRM) est une méthode non invasive basée sur l'IRM qui permet de caractériser les profils métaboliques dans les tissus neuronaux. Différentes méthodes de mesure sont disponibles en pratique clinique. La spectroscopie par voxel unique est simple et robuste en termes de manipulation et de technique, et permet la quantification absolue des concentrations de métabolites en utilisant des analyses de référence basées sur l'eau présente dans les tissus, leur molarité et la concentration en protons. La SRM quantifiée est utilisée cliniquement pour différencier le degré de malignité des tumeurs cérébrales et les distinguer des métastases, des lymphomes ou des maladies inflammatoires du système nerveux central. Elle est également utilisée

comme marqueur de substitution pour les troubles de la maturation cérébrale, l'épilepsie et les maladies neurométaboliques. [34]

- **Application et développement futur**

L'imagerie quantitative ne peut pas remplacer l'expertise d'un neuroradiologue, car l'interprétation des résultats générés par ordinateur nécessite toujours les connaissances d'un expert. Cependant, les avancées dans ce domaine, associées aux progrès de l'informatique, sont considérables. En particulier, pour les problèmes spécifiques à la neuroradiologie où l'observation à l'œil nu ou les méthodes de mesure traditionnelles peuvent entraîner des erreurs, l'introduction du diagnostic assisté par ordinateur est prévisible. Cela concerne notamment la détection et la classification des réductions du volume cérébral dans le diagnostic de la démence, les mesures précises du volume des lésions dans le diagnostic et le suivi des maladies tumorales ou de la sclérose en plaques, ainsi que l'évaluation des modifications de l'architecture corticale dans les troubles du développement chez l'enfant ou le diagnostic de l'épilepsie. En utilisant des bases de données et des informations existantes, ces nouvelles approches méthodologiques offrent un potentiel prometteur pour une estimation personnalisée du risque individuel de chaque patient, bien que des questions éthiques et morales se posent concernant la gestion de la classification préalable des patients en "classes de risque". Dans un avenir proche, une interaction plus étroite entre les disciplines médicales, les sciences naturelles, l'ingénierie et surtout les spécialités éthiques sera cruciale pour un développement positif de ce domaine encore récent. [34]

Conclusion

En conclusion, la segmentation des images médicales, en particulier dans le domaine de l'imagerie cérébrale, est une étape cruciale pour de nombreux objectifs cliniques, tels que le diagnostic précoce, le suivi des patients et l'amélioration des traitements. Les approches traditionnelles de segmentation manuelle sont souvent chronophages et sujettes à des variations interprétatives.

Cependant, l'avènement de l'apprentissage en profondeur a ouvert de nouvelles perspectives dans le domaine de la segmentation. Les réseaux de neurones convolutionnels (CNN) ont montré leur efficacité en capturant les caractéristiques importantes des images médicales et en réalisant des segmentations précises. Des architectures spécifiques comme VGG16 et U-Net ont été développées pour répondre aux défis de la segmentation en fournissant des connexions entre les différentes couches du réseau.

Chapitre 03

Contribution et Résultat

3.1. Introduction

Dans ce chapitre, nous aborderons l'utilisation du langage de programmation Python et de l'environnement de travail Colab pour mettre en œuvre un modèle U-Net. Le modèle sera appliqué à l'ensemble de données BRATS 2020, qui contient des images médicales cérébrales. Nous explorerons les étapes de prétraitement des données, le chargement et la séparation des ensembles de données, ainsi que l'implémentation du modèle U-Net dans Colab. L'objectif est d'utiliser ce modèle pour effectuer la segmentation des images médicales cérébrales et évaluer sa performance sur l'ensemble de données BRATS 2020.

3.2. Environnement de travail

3.2.1. Python

Python est un langage de programmation orienté objet et multi-plateformes, disponible en open source. Il est largement utilisé dans divers domaines tels que le développement logiciel, l'analyse de données et la gestion d'infrastructures grâce à ses bibliothèques spécialisées. Contrairement au langage HTML qui est principalement utilisé pour la programmation web, Python offre une polyvalence qui lui permet d'être utilisé dans de nombreux contextes.

3.2.1.1 Les principaux usages de Python

Python est principalement employé pour l'écriture de scripts et l'automatisation de tâches fastidieuses, notamment pour interagir avec les navigateurs web. Cependant, Python trouve également des applications dans les domaines suivants :

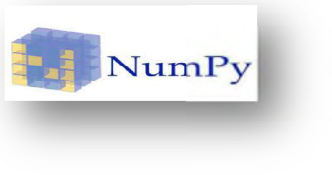
- ✓ Programmation d'applications ;

- ✓ Génération de code ;
- ✓ Création de services web ;
- ✓ Pratique de la méta-programmation. [35]

Nous citons Les bibliothèques Python suivants :

Figure 3.1: Les bibliothèques Python.

- **NumPy**



À l'origine, Python n'était pas conçu comme un outil destiné au calcul numérique. Toutefois, l'introduction de NumPy a joué un rôle déterminant dans l'élargissement des capacités de Python en matière de fonctions mathématiques, formant ainsi une base solide pour le développement de solutions en apprentissage automatique (machine learning). [36]

- **OpenCV**

OpenCV, abréviation d'Open Computer Vision, est une bibliothèque open-source qui a été initialement développée par Intel. Elle est spécialement conçue pour le traitement d'images en temps réel. Suite à la transition de support, la société de robotique Willow Garage a repris le développement de la bibliothèque, puis elle a été ensuite prise en charge par la société ItSeez. Depuis l'acquisition d'ItSeez par Intel en 2016, le support d'OpenCV est à nouveau assuré par Intel. [37]

- **Matplotlib**

NumPy, SciPy et Matplotlib sont des bibliothèques qui, ensemble, offrent une alternative à l'utilisation du langage statistique propriétaire MATLAB. Ces bibliothèques sont conçues de manière à fournir des fonctionnalités similaires à celles de MATLAB, ce qui explique leur similarité. [35]

- **Pandas**

Pandas est une bibliothèque Python construite sur NumPy, offrant des fonctionnalités avancées de manipulation et d'analyse de données. Elle est principalement utilisée pour effectuer des opérations sur des dataframes, facilitant ainsi l'analyse de données et la mise en œuvre d'algorithmes de machine Learning. [36]

- **SciPy**

Combinée avec NumPy, cette bibliothèque est un outil incontournable pour effectuer des calculs mathématiques, scientifiques et techniques. SciPy se concentre sur des opérations mathématiques telles que l'interpolation numérique, l'intégration, l'algèbre linéaire et les statistiques.[36]

- **Scikit-learn**

À ses débuts, Scikit-learn était conçu comme une extension tierce de la bibliothèque SciPy. Aujourd'hui, il s'est développé en une bibliothèque autonome et est devenu l'une des plus populaires sur GitHub. [36]

- **Tensorflow**

TensorFlow est une bibliothèque open-source de calcul numérique utilisée dans le domaine de l'apprentissage automatique basé sur les réseaux neuronaux. Elle a été développée en 2015 par l'équipe de recherche Google Brain dans le but de l'utiliser en interne pour les produits Google. [36]

- **Keras**

Initialement, Keras était une plateforme utilisée pour expérimenter rapidement les réseaux neuronaux profonds. Cependant, elle a rapidement évolué pour devenir une bibliothèque autonome dédiée à l'apprentissage automatique. Keras propose un ensemble complet d'outils d'apprentissage automatique, ce qui permet aux entreprises de traiter efficacement les données textuelles et d'image. [36]

3.2.2. Google Colab

Colaboratory, souvent abrégé en "Colab", est un produit développé par Google Research. Il offre la possibilité à quiconque d'écrire et d'exécuter du code Python de son choix directement depuis un navigateur. Cet environnement est spécialement conçu pour le machine learning, l'analyse de données et l'éducation. Plus techniquement, Colab est un service hébergé de notebooks Jupyter qui ne nécessite aucune configuration préalable et permet un accès gratuit à des ressources informatiques, y compris des GPU. [38]

3.2.2. 1.Avantages de Colab

Colab offre plusieurs avantages :

- Il permet d'améliorer vos compétences en programmation Python.
- Il facilite le développement d'applications de deep learning en utilisant des bibliothèques populaires telles que Keras, TensorFlow, PyTorch et OpenCV. [39]

3.2.2. 2. Google colab

Activer Google Colab et développer votre premier modèle en Deep Learning

Etape 1 : Créer nouveau dossier sur Google Drive

Pour commencer, accédez à votre compte Gmail (ou G Suite) et ouvrez l'application Google Drive. Créez ensuite un nouveau dossier. Dans cet exemple, j'ai nommé le dossier "app" dans mon Google Drive. Bien entendu, vous pouvez choisir un nom différent selon vos préférences. [39]

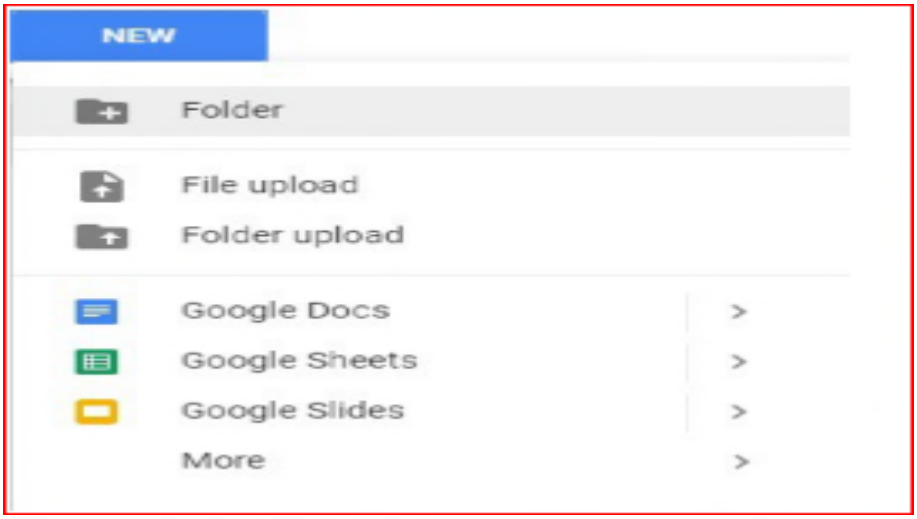


Figure 3.2: création nouveau dossier

Une fois le dossier créé, vous devriez obtenir un écran similaire à celui-ci :

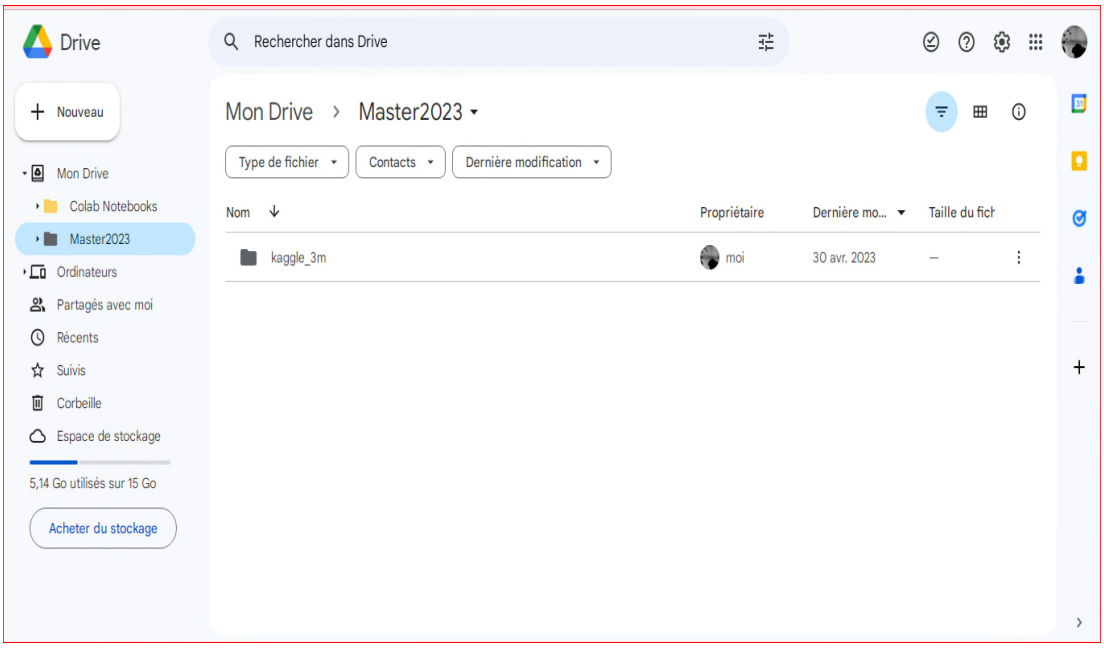


Figure 3.3: nouveau dossier créer

Étape 2 : Créer un nouveau fichier Colab

Dans votre nouveau dossier, faites un clic droit avec votre souris puis sélectionnez **More > Colaboratory.**

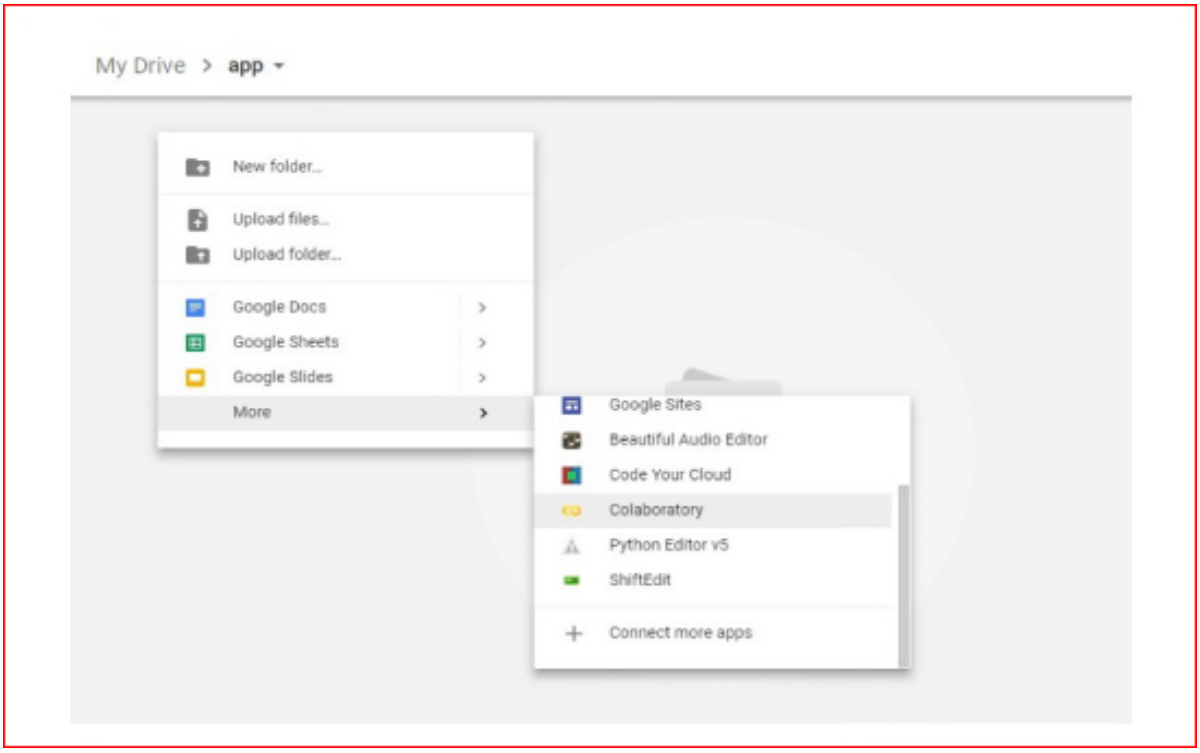


Figure 3.4: nouveau fichier colab

Une fois dans le nouveau fichier, vous pouvez le renommer en cliquant sur le nom en haut du document.

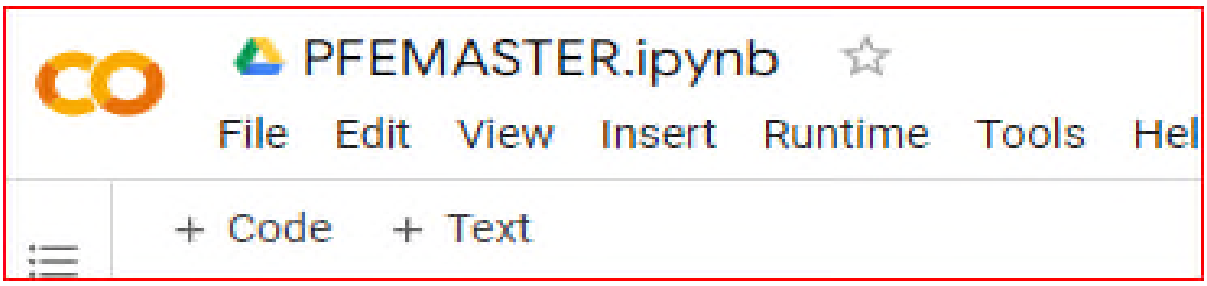


Figure 3.5: Rennomer le nom de fichier

Étape 3 : Exécuter du code Python de base

Nous pouvons dès maintenant commencer à utiliser Colab

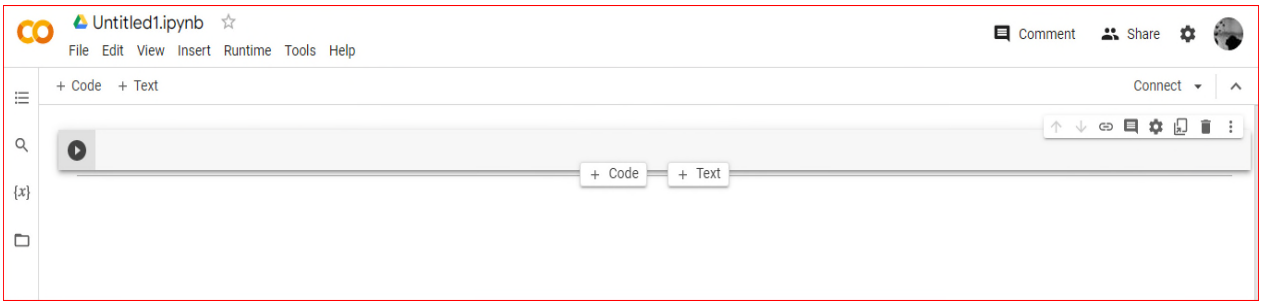


Figure 3.6: Nouveau notebook ouvert dans la plateforme colab.

3.2.3. Google drive

Accède aux données stockées dans Google drive on exécute la commande

```
from google.colab import drive

drive.mount('/content/drive')
```

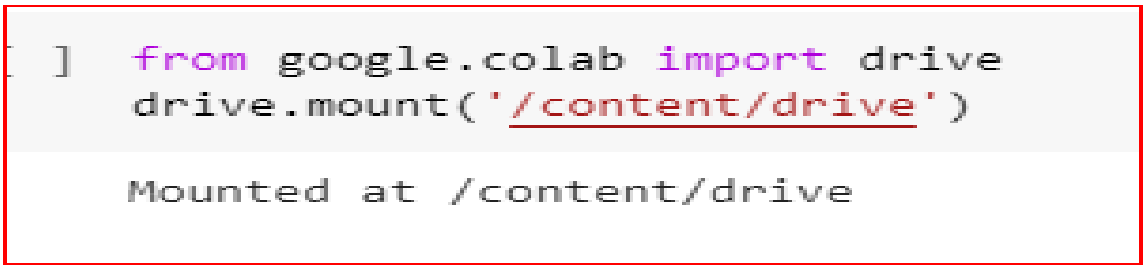


Figure 3.7: Monter le drive dans le dossier

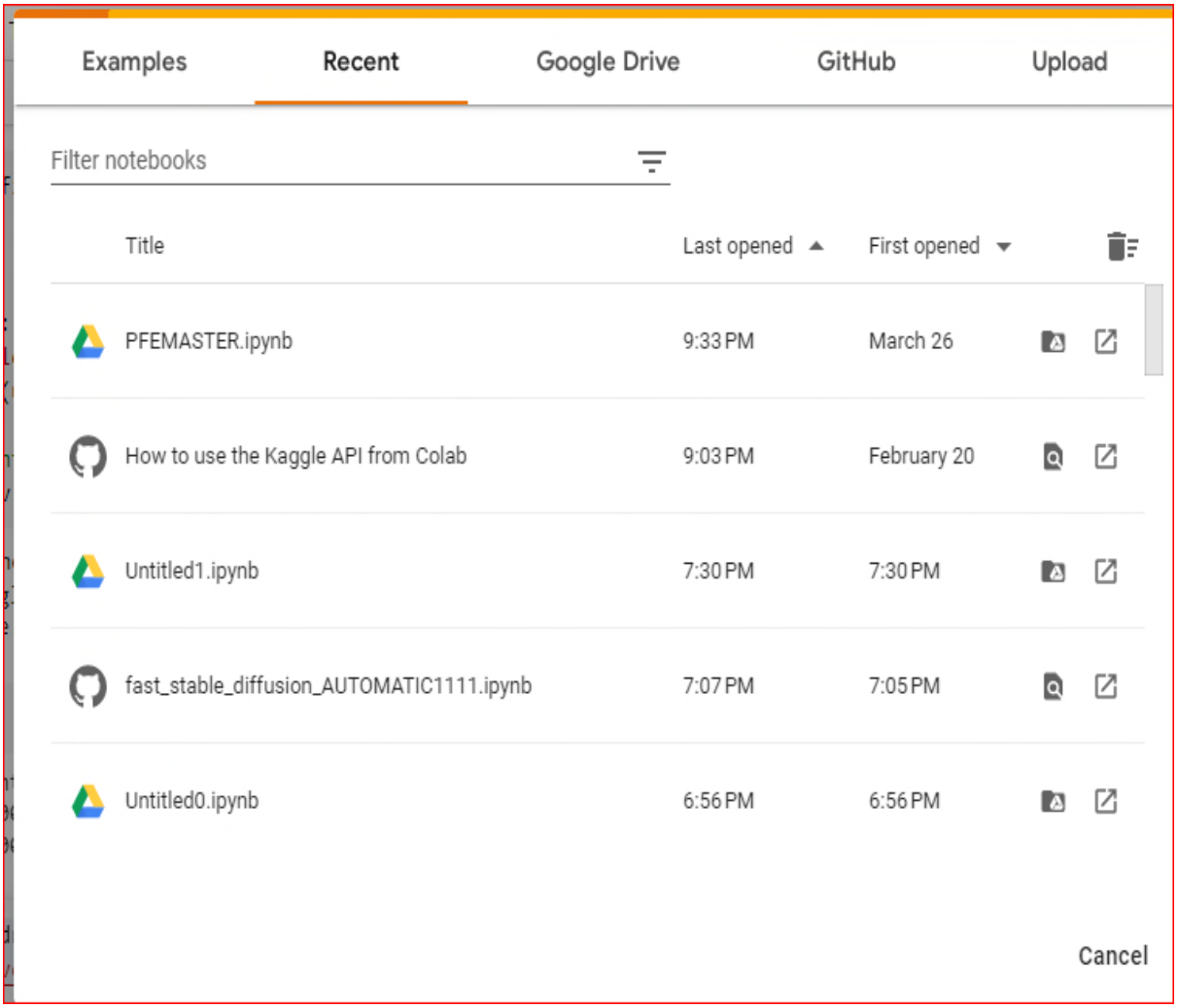


Figure 3.8: Comment ouvrir un notebook dans drive.

- Importer les bibliothèques Python (Python Libraires)

Figure 3.9: Comment importer une bibliothèque dans Colab.

- Choisir l'accélérateur matériel

Dans la barre des options choisir "exécution"(Runtime) puis "modifier le type d'exécution" (Change Runtime Type) et mettre l'option accélérateur matériel (Hardware Accelerator) en mode GPU ou TPU (Tensor Processing Unit). Bien expliquer dans ces figures suivantes :

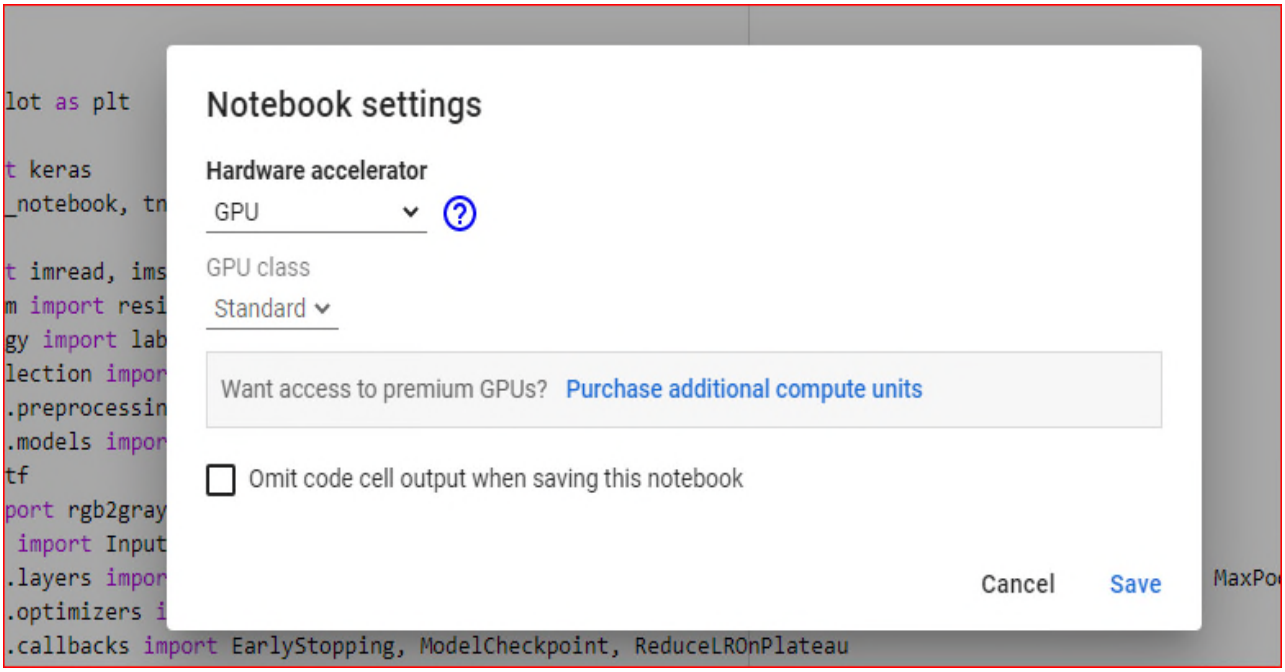


Figure 3.10: Accélérateur matériel GPU.

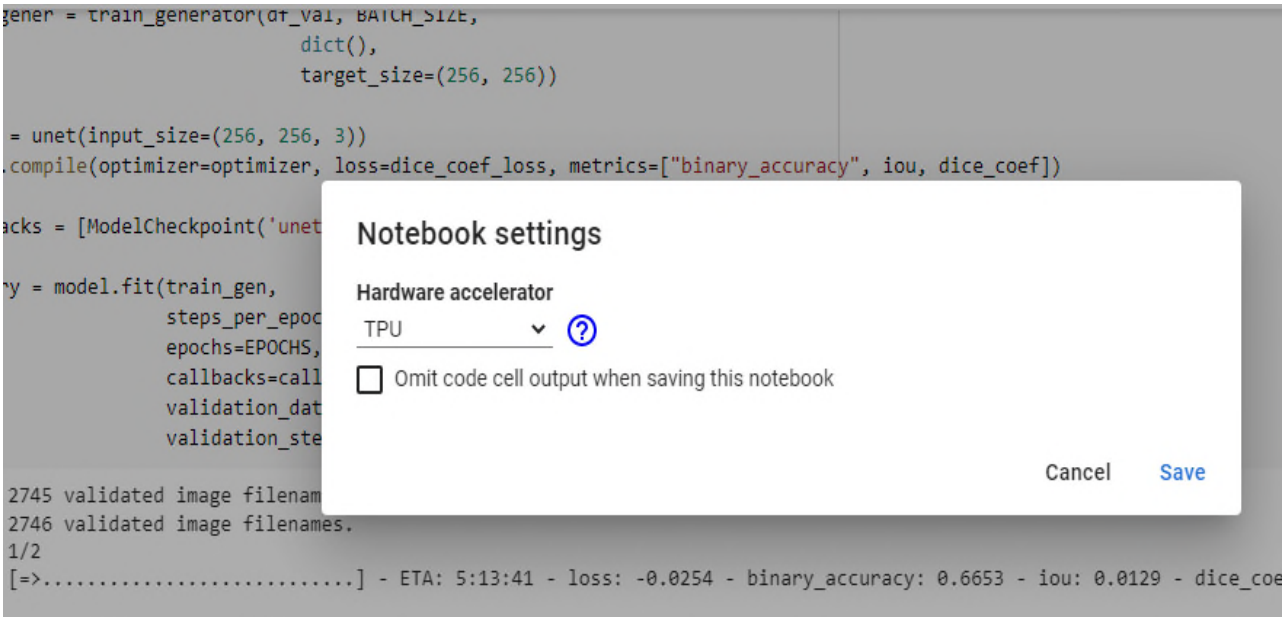


Figure 3.11: Accélérateur matériel TPU.

- Importer Dataset depuis Kaggle
  Pour télécharger Dataset depuis kaggle suivre les étapes suivant :

Etape 1 : Visitez WWW.Kaggle.com . Accédez à votre profil et cliquez sur compte

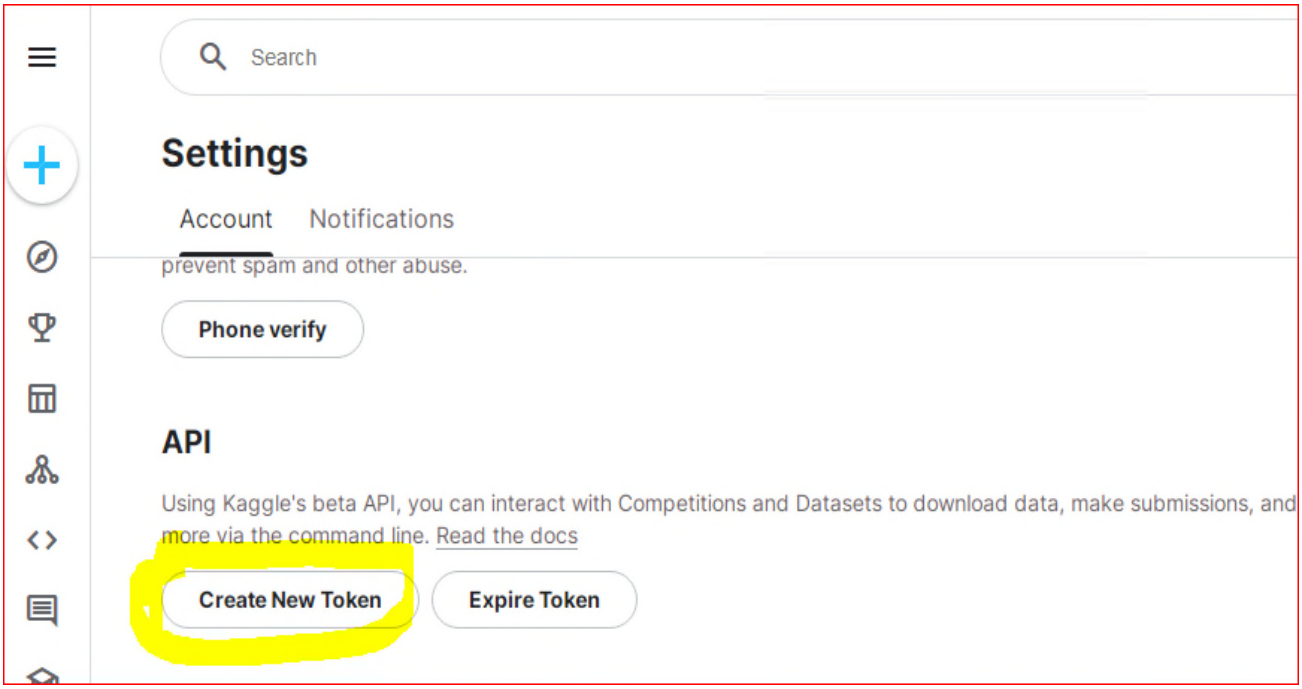


Figure 3.12: Créer un nouveau jeton API

- Télécharger le fichier Kaggle.json sur votre machine

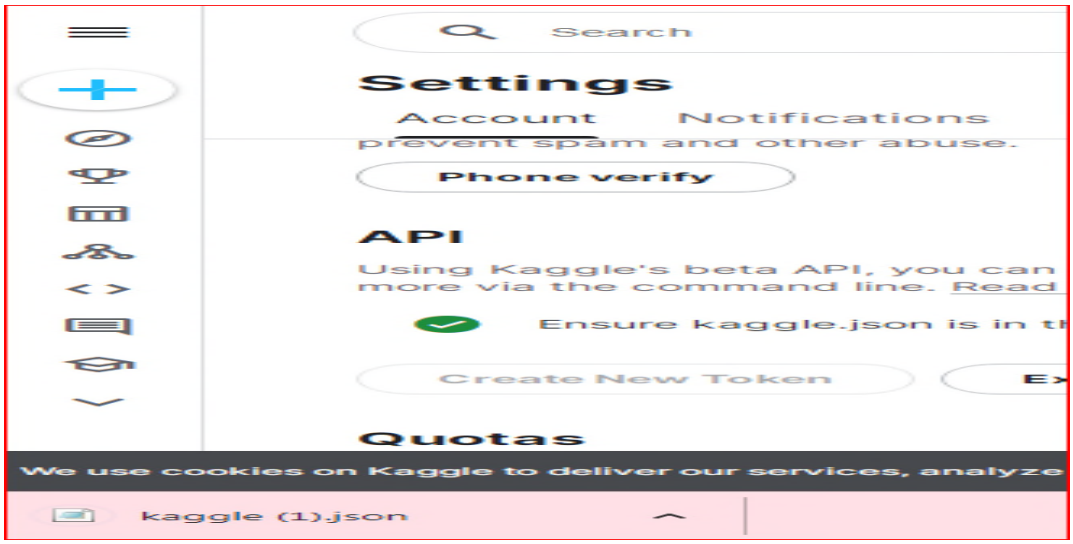


Figure 3.13: Télécharger Kaggle.json sur la machine

- Télécharger maintenant le fichier Kaggle.json

```
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.
format(
      name=fn, length=len(uploaded[fn])))

# Then move kaggle.json into the folder where the API expects to
find it.
!mkdir -
p ~/.kaggle/ && mv kaggle.json ~/.kaggle/ && chmod 600 ~/.kaggle/
kaggle.json
```

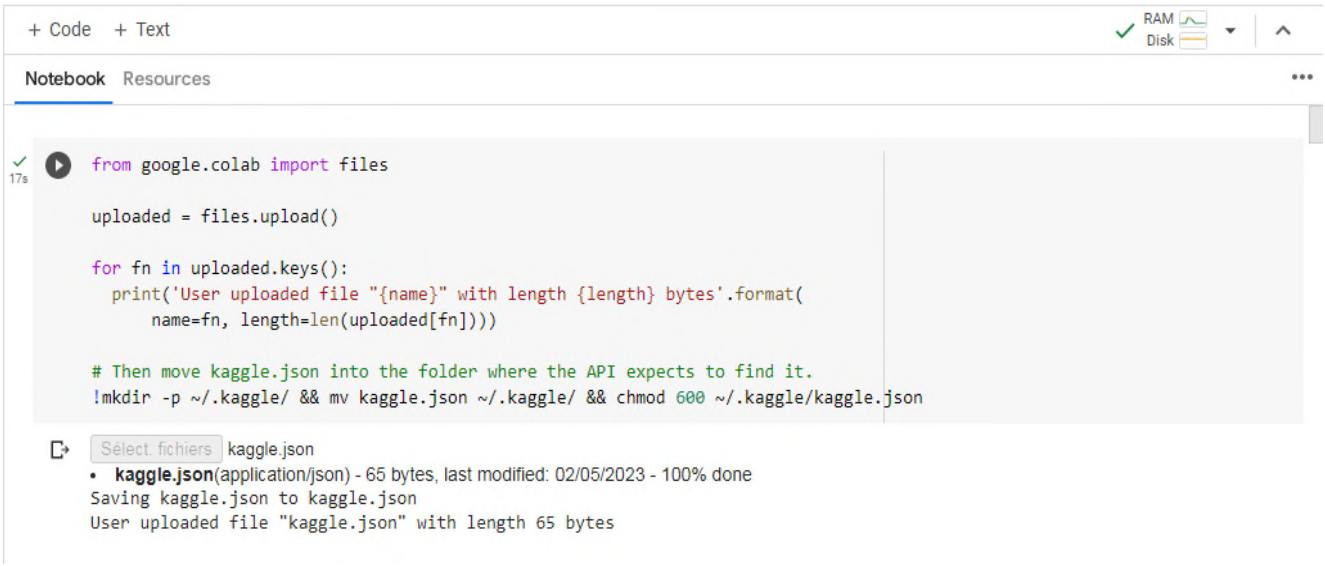


Figure 3.14: Code de téléchargement de fichier Kaggle.json

- Télécharger l'ensemble de données requis à partir de kaggle par la commande suivant :

```
[3] !kaggle datasets download -d awsaf49/brats20-dataset-training-validation
brats20-dataset-training-validation.zip: Skipping, found more recently modified local copy (use --force to force download)

[4] !kaggle datasets download -d rastislav/model-x80-dcs65
model-x80-dcs65.zip: Skipping, found more recently modified local copy (use --force to force download)

[5] !kaggle datasets download -d rastislav/modelperclasseval
modelperclasseval.zip: Skipping, found more recently modified local copy (use --force to force download)

[6] !kaggle datasets download -d awsaf49/brats20-dataset-training-validation
brats20-dataset-training-validation.zip: Skipping, found more recently modified local copy (use --force to force download)
```

Figure 3.15: Code Téléchargement DATASET

- Si votre fichier est un fichier ZIP, vous pouver le décompresser avec le code suivant :

```
from zipfile import ZipFile
zf = ZipFile('/content/brats20-dataset-training-validation.zip', 'r')
zf = ZipFile('/content/brats20-dataset-training-validation.zip', 'r')
zf.extractall('New__Folder')
zf = ZipFile('/content/model-x80-dcs65.zip', 'r')
zf = ZipFile('/content/modelperclasseval.zip', 'r')
zf.extractall('path_to_extract_folder')
zf.close()
```

```
[7] from zipfile import ZipFile
    zf = ZipFile('/content/brats20-dataset-training-validation.zip', 'r')
    zf = ZipFile('/content/brats20-dataset-training-validation.zip', 'r')
    zf.extractall('New__Folder')
    zf = ZipFile('/content/model-x80-dcs65.zip', 'r')
    zf = ZipFile('/content/modelperclasseval.zip', 'r')
    zf.extractall('path_to_extract_folder')
    zf.close()
```

Figure 3.16: Code Décompression fichier ZIP

3.3. Modèle U_Net

3.3.1. Implémentation

On va suivre les étapes d'implémentation suivantes pour avoir segmenter les images IRM en utiliser le model U_Net

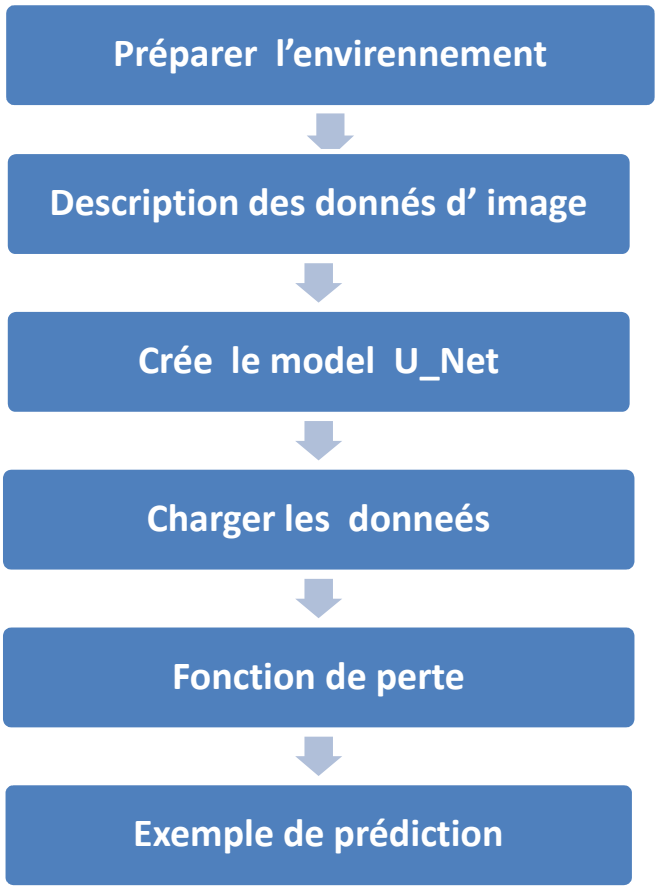


Figure 3.17: Les étapes d'implémentation

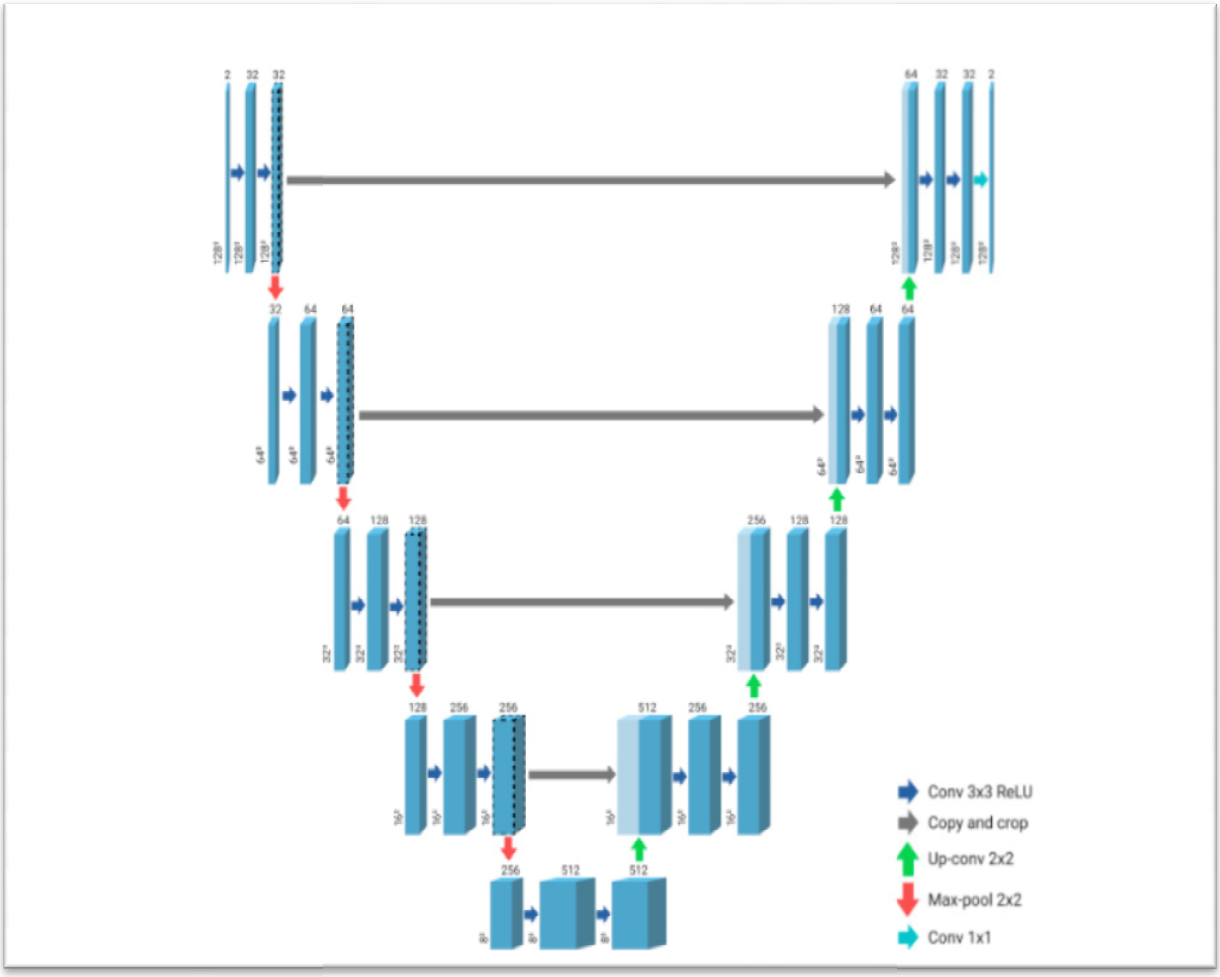


Figure 3.18: Architecture U_Net

Visuellement, l'architecture de U-Net adopte une forme en "U" avec une symétrie et se compose de trois sections : la contraction, le goulot d'étranglement et l'expansion. La première section, également appelée encodeur, est utilisée pour extraire le contexte d'une image. Elle est constituée de couches de convolution et de couches de max pooling qui captent les caractéristiques de l'image et réduisent sa taille pour réduire le nombre de paramètres du réseau. Cette section implique l'application répétée de deux couches de convolution 3x3, suivies d'une fonction d'activation ReLU et d'une normalisation par lots (batch normalization). Ensuite, une opération de max pooling 2x2 est appliquée pour réduire les dimensions spatiales. Le goulot d'étranglement, également connu sous le nom de pont, joue un rôle crucial en reliant l'encodeur au réseau de décodeurs pour assurer un flux d'informations complet. Il est constitué de deux couches de convolution 3x3, suivies chacune d'une fonction d'activation ReLU. Le deuxième bloc, qui est le décodeur,

permet une localisation précise en utilisant des convolutions transposées pour restaurer la taille initiale de l'image. Le processus de décodage débute par une étape de sur-échantillonnage (upsampling) de la carte des caractéristiques, suivi d'une couche de convolution 2x2 transposée. Ensuite, deux couches de convolution 3x3 sont appliquées, chaque convolution étant suivie d'une fonction d'activation ReLU. Après cela, la sortie du dernier décodeur est passée par une couche de convolution 1x1 avec une fonction d'activation sigmoïde. Le modèle 3D-Unet, introduit peu après U-Net, est utilisé pour traiter des volumes plutôt que des pixels. Il utilise des convolutions tridimensionnelles (3x3x3, 2x2x2 ou 1x1x1) au lieu des convolutions bidimensionnelles. Ce modèle peut être directement entraîné sur des images en 3D, sans avoir besoin de les entraîner sur chaque coupe individuellement. L'un des avantages importants de U-Net est sa capacité à effectuer la segmentation d'image en prédisant les pixels de l'image. [40]

Pour notre implémentation, nous nous sommes appuyés sur des implementations de réseaux de neurones en utilisant Python provenant de la plateforme "Kaggle". Kaggle est un site qui rassemble une communauté dédiée à l'intelligence artificielle et aux sciences des données. Il offre un accès à une vaste collection de codes et de données nécessaires pour mener des travaux de science des données. Kaggle propose plus de 50 000 ensembles de données publics et 400 000 notebooks publics, offrant ainsi la possibilité de réaliser rapidement toutes sortes d'analyses. [2]

- Préparer l’environnement
  On va utiliser le modèle U_Net pour segmenter les images IRM

```
import os
Run cell (Ctrl+Enter)
cell has not been executed in this session
import matplotlib.pyplot as plt
import cv2
from tensorflow import keras
from tqdm import tqdm_notebook, tnrange
from glob import glob
from skimage.io import imread, imshow, concatenate_images
from skimage.transform import resize
from skimage.morphology import label
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model, load_model, save_model
import tensorflow as tf
from skimage.color import rgb2gray
from tensorflow.keras import Input
from tensorflow.keras.layers import Input, Activation, BatchNormalization, Dropout, Lambda, Conv2D, Conv2DTranspose, MaxPooling2D, concatenate
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
from tensorflow.keras import backend as K
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
```

```
+ Code + Text
RAM
Disk
Notebook Resources
[9] # neural imaging
29s !pip install nilearn
    import nilearn as nl
    import nibabel as nib
    import nilearn.plotting as nlplt
    !pip install git+https://github.com/miykael/gif_your_nifti # nifti to gif
    import gif_your_nifti.core as gif2nif



    # ml libs
    import keras
    import keras.backend as K
    from keras.callbacks import CSVLogger
    import tensorflow as tf
    from tensorflow.keras.utils import plot_model
    from sklearn.preprocessing import MinMaxScaler
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import classification_report
    from tensorflow.keras.models import *
    from tensorflow.keras.layers import *
    from tensorflow.keras.optimizers import *
    from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping, TensorBoard
```

```
[10] # DEFINE seg-areas
     SEGMENT_CLASSES = {
         0 : 'NOT tumor',
         1 : 'NECROTIC/CORE', # or NON-ENHANCING tumor CORE
         2 : 'EDEMA',
         3 : 'ENHANCING' # original 4 -> converted into 3 later
     }

     # there are 155 slices per volume
     # to start at 5 and use 145 slices means we will skip the first 5 and last 5
     VOLUME_SLICES = 100
     VOLUME_START_AT = 22 # first slice of volume that we will include
```

Figure 3.19: Code de préparation de l'environnement

- Importer les bibliothèques

```
# Let's see whether Nilearn is installed
try:
    import nilearn
except ImportError:
    # if not, install it using pip
    !pip install nilearn
```

```
[ ] import os
    import cv2
    import glob
    import PIL
    import shutil
    import numpy as np
    import pandas as pd
    import seaborn as sns
    import matplotlib.pyplot as plt
```

Figure 3.20: Code d'importation des bibliothèques

- Descriptions des données d'image

```
+ Code + Text
Notebook Resources
```

```
[11]
    TRAIN_DATASET_PATH = '../content/New__Folder/BraTS2020_TrainingData/MICCAI_BraTS2020_TrainingData/'
    VALIDATION_DATASET_PATH = '../content/New__Folder/BraTS2020_ValidationData/MICCAI_BraTS2020_ValidationData'

    test_image_flair=nib.load(TRAIN_DATASET_PATH + 'BraTS20_Training_001/BraTS20_Training_001_flair.nii').get_fdata()
    test_image_t1=nib.load(TRAIN_DATASET_PATH + 'BraTS20_Training_001/BraTS20_Training_001_t1.nii').get_fdata()
    test_image_t1ce=nib.load(TRAIN_DATASET_PATH + 'BraTS20_Training_001/BraTS20_Training_001_t1ce.nii').get_fdata()
    test_image_t2=nib.load(TRAIN_DATASET_PATH + 'BraTS20_Training_001/BraTS20_Training_001_t2.nii').get_fdata()
    test_mask=nib.load(TRAIN_DATASET_PATH + 'BraTS20_Training_001/BraTS20_Training_001_seg.nii').get_fdata()


    fig, (ax1, ax2, ax3, ax4, ax5) = plt.subplots(1,5, figsize = (20, 10))
    slice_w = 25
    ax1.imshow(test_image_flair[:,:,test_image_flair.shape[0]//2-slice_w], cmap = 'gray')
    ax1.set_title('Image flair')
    ax2.imshow(test_image_t1[:,:,test_image_t1.shape[0]//2-slice_w], cmap = 'gray')
    ax2.set_title('Image t1')
    ax3.imshow(test_image_t1ce[:,:,test_image_t1ce.shape[0]//2-slice_w], cmap = 'gray')
    ax3.set_title('Image t1ce')
    ax4.imshow(test_image_t2[:,:,test_image_t2.shape[0]//2-slice_w], cmap = 'gray')
    ax4.set_title('Image t2')
    ax5.imshow(test_mask[:,:,test_mask.shape[0]//2-slice_w])
```

```
+ Code + Text
[ ] # Skip 50:-50 slices since there is not much to see
    fig, ax1 = plt.subplots(1, 1, figsize = (15,15))
    ax1.imshow(rotate(montage(test_image_t1[50:-50,:,:]), 90, resize=True), cmap ='gray')
```

```
[ ] # Skip 50:-50 slices since there is not much to see
    fig, ax1 = plt.subplots(1, 1, figsize = (15,15))
    ax1.imshow(rotate(montage(test_mask[60:-60,:,:]), 90, resize=True), cmap ='gray')
```

```
niimg = nl.image.load_img(TRAIN_DATASET_PATH + 'BraTS20_Training_001/BraTS20_Training_001_fla
nimask = nl.image.load_img(TRAIN_DATASET_PATH + 'BraTS20_Training_001/BraTS20_Training_001_seg.nii')

fig, axes = plt.subplots(nrows=4, figsize=(30, 40))


nlplt.plot_anat(niimg,
                title='BraTS20_Training_001_flair.nii plot_anat',
                axes=axes[0])

nlplt.plot_epi(niimg,
               title='BraTS20_Training_001_flair.nii plot_epi',
               axes=axes[1])

nlplt.plot_img(niimg,
               title='BraTS20_Training_001_flair.nii plot_img',
               axes=axes[2])

nlplt.plot_roi(nimask,
               title='BraTS20_Training_001_flair.nii with mask plot_roi',
               bg_img=niimg,
               axes=axes[3], cmap='Paired')
plt.show()
```

Figure 3.21: Description des données d'image

- Créer un model ||U_Net

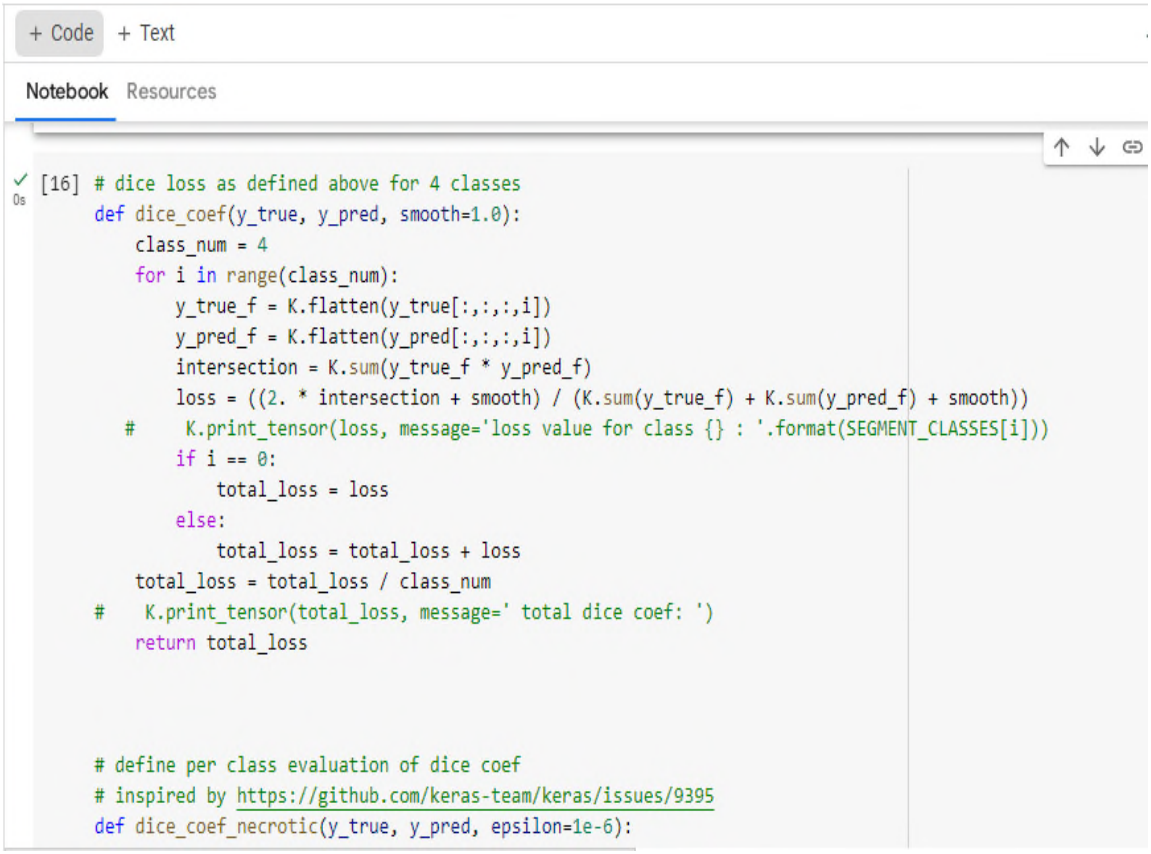
  Fonction de perte

```
[16] # dice loss as defined above for 4 classes
     def dice_coef(y_true, y_pred, smooth=1.0):
         class_num = 4
         for i in range(class_num):
             y_true_f = K.flatten(y_true[:,:,:,i])
             y_pred_f = K.flatten(y_pred[:,:,:,i])
             intersection = K.sum(y_true_f * y_pred_f)
             loss = ((2. * intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) + smooth))
         #     K.print_tensor(loss, message='loss value for class {} : '.format(SEGMENT_CLASSES[i]))
             if i == 0:
                 total_loss = loss
             else:
                 total_loss = total_loss + loss
         total_loss = total_loss / class_num
     #    K.print_tensor(total_loss, message=' total dice coef: ')
         return total_loss



     # define per class evaluation of dice coef
     # inspired by https://github.com/keras-team/keras/issues/9395
     def dice_coef_necrotic(y_true, y_pred, epsilon=1e-6):
```

+ Code + Text

```
def dice_coef_edema(y_true, y_pred, epsilon=1e-6):
    intersection = K.sum(K.abs(y_true[:,:,:,2] * y_pred[:,:,:,2]))
    return (2. * intersection) / (K.sum(K.square(y_true[:,:,:,2])) + K.sum(K.square(y_pred[:,:,:,2])) + epsilon)

def dice_coef_enhancing(y_true, y_pred, epsilon=1e-6):
    intersection = K.sum(K.abs(y_true[:,:,:,3] * y_pred[:,:,:,3]))
    return (2. * intersection) / (K.sum(K.square(y_true[:,:,:,3])) + K.sum(K.square(y_pred[:,:,:,3])) + epsilon)


# Computing Precision
def precision(y_true, y_pred):
        true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
        predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
        precision = true_positives / (predicted_positives + K.epsilon())
        return precision


# Computing Sensitivity
def sensitivity(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    return true_positives / (possible_positives + K.epsilon())
```

```
# Computing Specificity
def specificity(y_true, y_pred):
    true_negatives = K.sum(K.round(K.clip((1-y_true) * (1-y_pred), 0, 1)))
    possible_negatives = K.sum(K.round(K.clip(1-y_true, 0, 1)))
    return true_negatives / (possible_negatives + K.epsilon())
```



+ Code + Text

Notebook Resources

```
[16] # dice loss as defined above for 4 classes
     def dice_coef(y_true, y_pred, smooth=1.0):
         class_num = 4
         for i in range(class_num):
             y_true_f = K.flatten(y_true[:,:,:,i])
             y_pred_f = K.flatten(y_pred[:,:,:,i])
             intersection = K.sum(y_true_f * y_pred_f)
             loss = ((2. * intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) + smooth))
         #     K.print_tensor(loss, message='loss value for class {} : '.format(SEGMENT_CLASSES[i]))
             if i == 0:
                 total_loss = loss
             else:
                 total_loss = total_loss + loss
         total_loss = total_loss / class_num
     #    K.print_tensor(total_loss, message=' total dice coef: ')
         return total_loss



     # define per class evaluation of dice coef
     # inspired by https://github.com/keras-team/keras/issues/9395
     def dice_coef_necrotic(y_true, y_pred, epsilon=1e-6):
```

```
# Computing Specificity
def specificity(y_true, y_pred):
    true_negatives = K.sum(K.round(K.clip((1-y_true) * (1-y_pred), 0, 1)))
    possible_negatives = K.sum(K.round(K.clip(1-y_true, 0, 1)))
    return true_negatives / (possible_negatives + K.epsilon())
```

```
IMG_SIZE=128
```

+ Code + Text

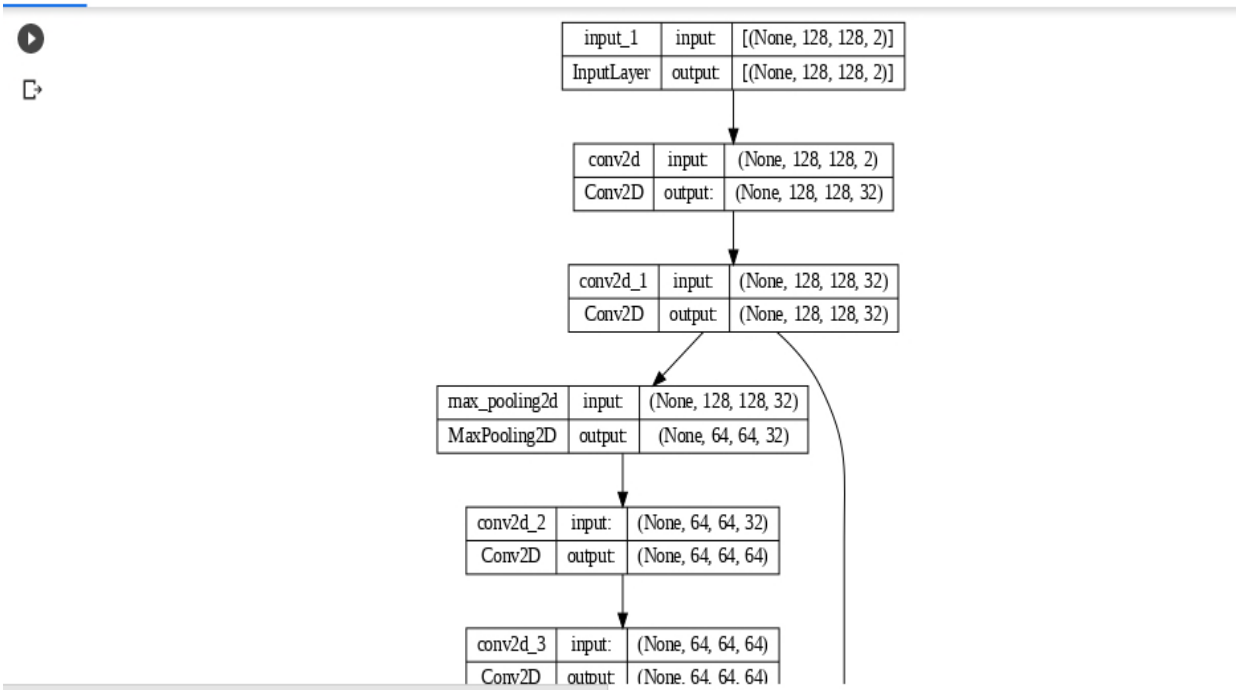
Notebook Resources

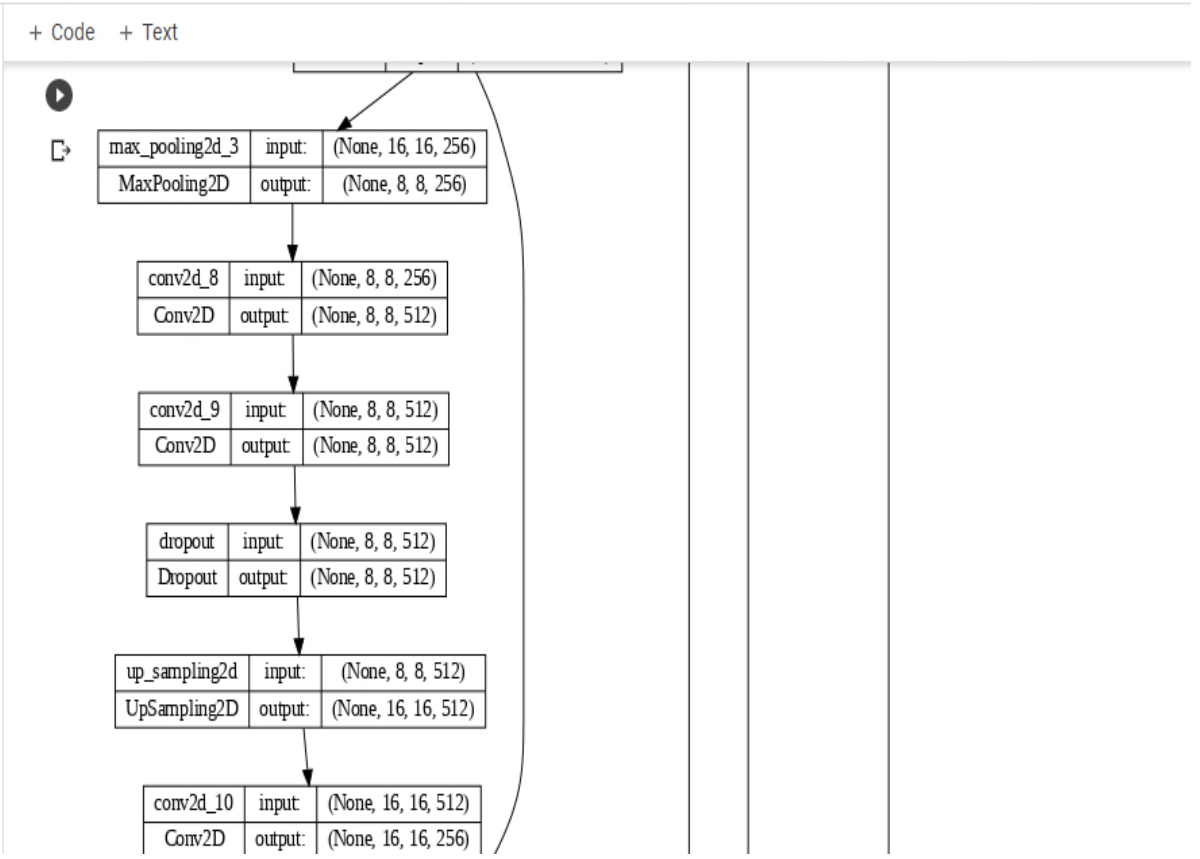
```
# source https://naomi-fridman.medium.com/multi-class-image-segmentation-a5cc671e647a
[18]
def build_unet(inputs, ker_init, dropout):
    conv1 = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(inputs)
    conv1 = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv1)

    pool = MaxPooling2D(pool_size=(2, 2))(conv1)
    conv = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(pool)
    conv = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv)

    pool1 = MaxPooling2D(pool_size=(2, 2))(conv)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(pool1)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv2)

    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(pool2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv3)


    pool4 = MaxPooling2D(pool_size=(2, 2))(conv3)
    conv5 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(pool4)
    conv5 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv5)
    drop5 = Dropout(dropout)(conv5)
```

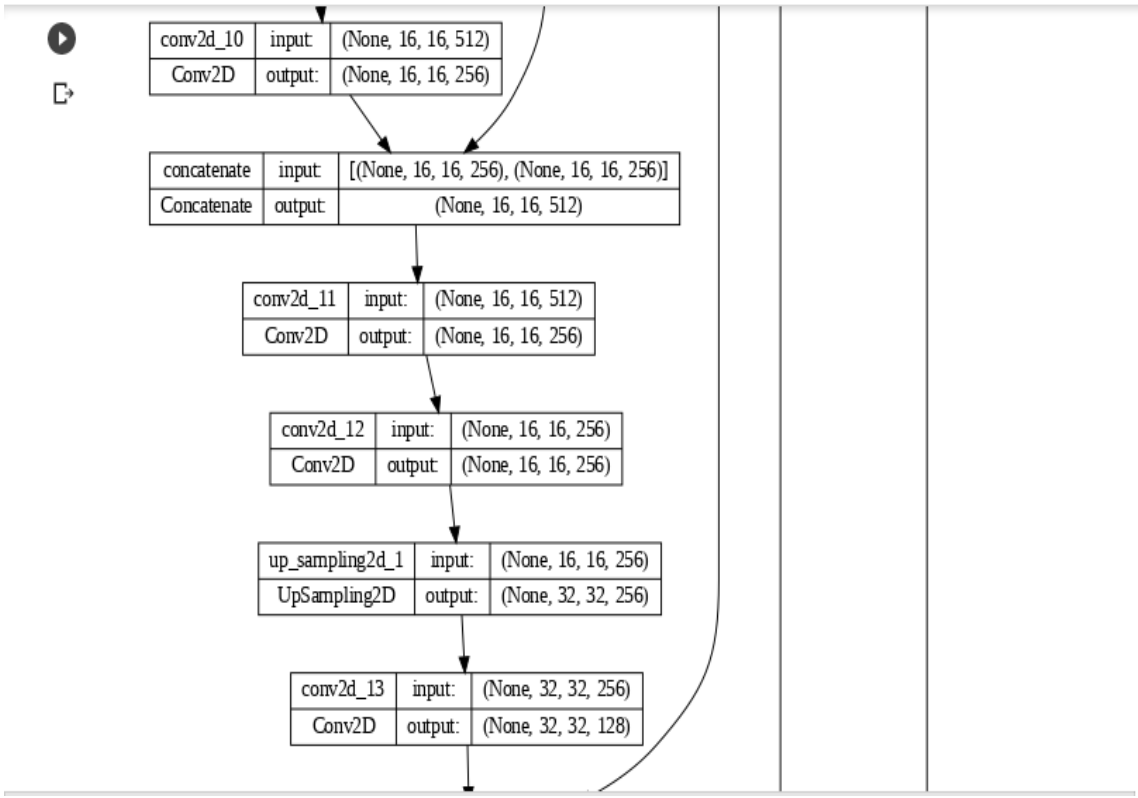
```
[19] plot_model(model,
                show_shapes = True,
                show_dtype=False,
                show_layer_names = True,
                rankdir = 'TB',
                expand_nested = False,
                dpi = 70)
```

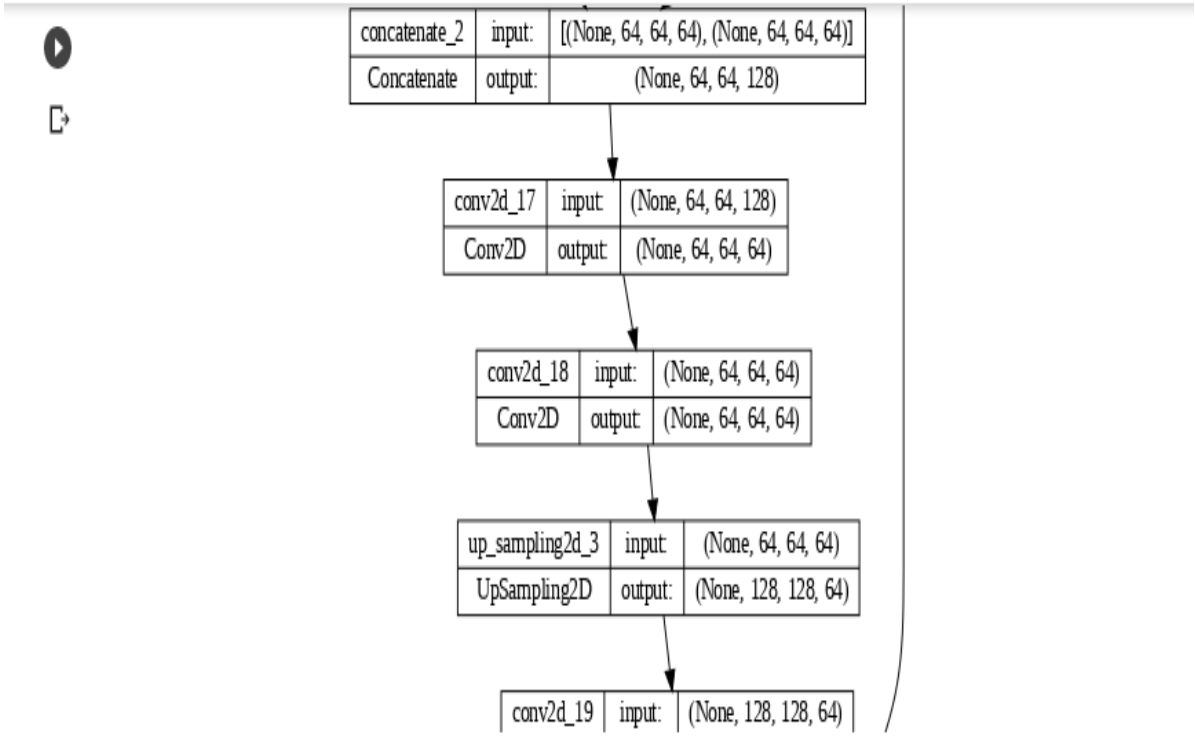
Figure 3.22: Model U_Net

Architecture model U_Net


input_1 | input: | [(None, 128, 128, 2)]
InputLayer | output: | [(None, 128, 128, 2)]
conv2d | input: | (None, 128, 128, 2)
Conv2D | output: | (None, 128, 128, 32)
conv2d_1 | input: | (None, 128, 128, 32)
Conv2D | output: | (None, 128, 128, 32)
max_pooling2d | input: | (None, 128, 128, 32)
MaxPooling2D | output: | (None, 64, 64, 32)
conv2d_2 | input: | (None, 64, 64, 32)
Conv2D | output: | (None, 64, 64, 64)
conv2d_3 | input: | (None, 64, 64, 64)


+ Code + Text
max_pooling2d_3 | input: | (None, 16, 16, 256)
MaxPooling2D | output: | (None, 8, 8, 256)
conv2d_8 | input: | (None, 8, 8, 256)
Conv2D | output: | (None, 8, 8, 512)
conv2d_9 | input: | (None, 8, 8, 512)
Conv2D | output: | (None, 8, 8, 512)
dropout | input: | (None, 8, 8, 512)
Dropout | output: | (None, 8, 8, 512)
up_sampling2d | input: | (None, 8, 8, 512)
UpSampling2D | output: | (None, 16, 16, 512)
conv2d_10 | input: | (None, 16, 16, 512)
Conv2D | output: | (None, 16, 16, 256)


conv2d_10 | input: | (None, 16, 16, 512)
Conv2D | output: | (None, 16, 16, 256)
concatenate | input: | [(None, 16, 16, 256), (None, 16, 16, 256)]
Concatenate | output: | (None, 16, 16, 512)
conv2d_11 | input: | (None, 16, 16, 512)
Conv2D | output: | (None, 16, 16, 256)
conv2d_12 | input: | (None, 16, 16, 256)
Conv2D | output: | (None, 16, 16, 256)
up_sampling2d_1 | input: | (None, 16, 16, 256)
UpSampling2D | output: | (None, 32, 32, 256)
conv2d_13 | input: | (None, 32, 32, 256)
Conv2D | output: | (None, 32, 32, 128)


concatenate_2 | input: | [(None, 64, 64, 64), (None, 64, 64, 64)]
Concatenate | output: | (None, 64, 64, 128)
conv2d_17 | input: | (None, 64, 64, 128)
Conv2D | output: | (None, 64, 64, 64)
conv2d_18 | input: | (None, 64, 64, 64)
Conv2D | output: | (None, 64, 64, 64)
up_sampling2d_3 | input: | (None, 64, 64, 64)
UpSampling2D | output: | (None, 128, 128, 64)
conv2d_19 | input: | (None, 128, 128, 64)

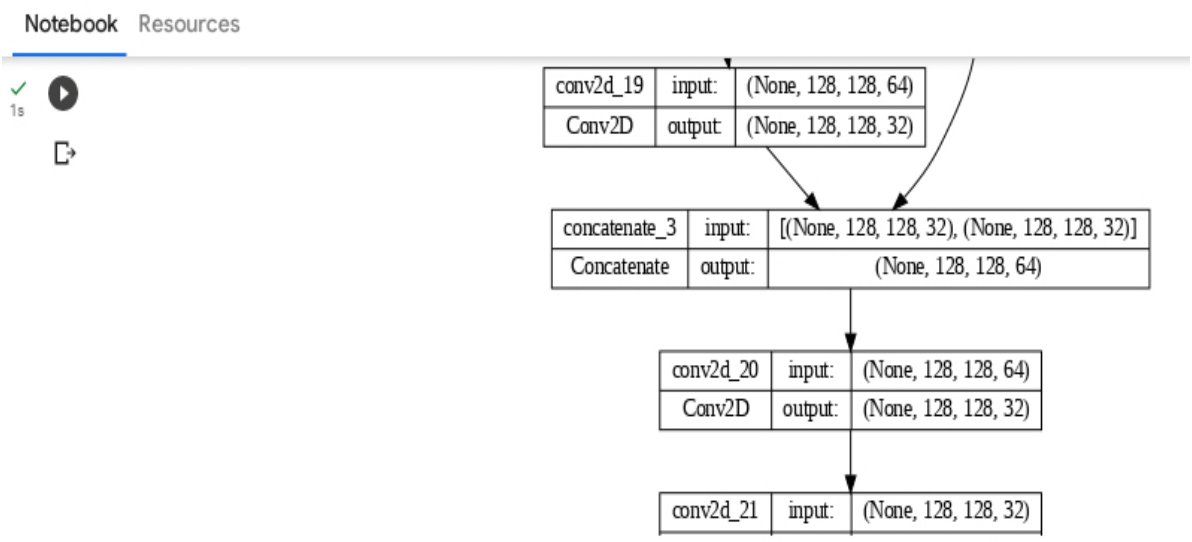


Figure 3.23: Architecture Model

- Charger les données

```
[20] # lists of directories with studies
     train_and_val_directories = [f.path for f in os.scandir(TRAIN_DATASET_PATH) if f.is_dir()]

     # file BraTS20_Training_355 has ill formatted name for for seg.nii file
     train_and_val_directories.remove(TRAIN_DATASET_PATH+'BraTS20_Training_355')


     def pathListIntoIds(dirList):
         x = []
         for i in range(0,len(dirList)):
             x.append(dirList[i][dirList[i].rfind('/')+1:])
         return x

     train_and_test_ids = pathListIntoIds(train_and_val_directories);


     train_test_ids, val_ids = train_test_split(train_and_test_ids,test_size=0.2)
     train_ids, test_ids = train_test_split(train_test_ids,test_size=0.15)
```

```
[21] class DataGenerator(keras.utils.Sequence):
         'Generates data for Keras'
         def __init__(self, list_IDs, dim=(IMG_SIZE,IMG_SIZE), batch_size = 1, n_channels = 2, shuffle=True):
             'Initialization'
             self.dim = dim
             self.batch_size = batch_size
             self.list_IDs = list_IDs
             self.n_channels = n_channels
             self.shuffle = shuffle
             self.on_epoch_end()

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

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

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

Notebook Resources

```
[21]
                 data_path = os.path.join(case_path, f'{i}_seg.nii');
                 seg = nib.load(data_path).get_fdata()

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

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

             # Generate masks
             y[y==4] = 3;
             mask = tf.one_hot(y, 4);
             Y = tf.image.resize(mask, (IMG_SIZE, IMG_SIZE));
             return X/np.max(X), Y

     training_generator = DataGenerator(train_ids)
     valid_generator = DataGenerator(val_ids)
     test_generator = DataGenerator(test_ids)
```

```
[22] # show number of data for each dir
     def showDataLayout():
         plt.bar(["Train","Valid","Test"],
         [len(train_ids), len(val_ids), len(test_ids)], align='center',color=[ 'green','red', 'blue'])
         plt.legend()

         plt.ylabel('Number of images')
         plt.title('Data distribution')

         plt.show()

     showDataLayout()
```

```
WARNING:matplotlib.legend:No artists with labels found to put in legend.  Note that artists whose label start with an under
```

```
[23] csv_logger = CSVLogger('training.log', separator=',', append=False)


     callbacks = [
     #     keras.callbacks.EarlyStopping(monitor='loss', min_delta=0,
     #                                   patience=2, verbose=1, mode='auto'),
           keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.2,
                                     patience=2, min_lr=0.000001, verbose=1),
     #  keras.callbacks.ModelCheckpoint(filepath = 'model_.{epoch:02d}-{val_loss:.6f}.m5',
     #                                  verbose=1, save_best_only=True, save_weights_only = True)
             csv_logger
         ]
```

Figure 3.24: Charger les données

- Model de train

```
[24] K.clear_session()

     # history =  model.fit(training_generator,
     #                      epochs=35,
     #                      steps_per_epoch=len(train_ids),
     #                      callbacks= callbacks,
     #                      validation_data = valid_generator
     #                      )
     # model.save("model_x1_1.h5")
```

```
[25] model = keras.models.load_model('../content/path_to_extract_folder/model_per_class.h5',
                                     custom_objects={ 'accuracy' : tf.keras.metrics.MeanIoU(num_classes=4),
                                                      "dice_coef": dice_coef,
                                                      "precision": precision,
                                                      "sensitivity":sensitivity,
                                                      "specificity":specificity,
                                                      "dice_coef_necrotic": dice_coef_necrotic,
                                                      "dice_coef_edema": dice_coef_edema,
                                                      "dice_coef_enhancing": dice_coef_enhancing
                                                     }, compile=False)

     history = pd.read_csv('../content/path_to_extract_folder/training_per_class.log', sep=',', engine='python')

     hist=history

     ############## ########## ####### #######

     # hist=history.history

     acc=hist['accuracy']
     val_acc=hist['val_accuracy']
```

Notebook Resources

```
[25]
1s
f,ax=plt.subplots(1,4,figsize=(16,8))

ax[0].plot(epoch,acc,'b',label='Training Accuracy')
ax[0].plot(epoch,val_acc,'r',label='Validation Accuracy')
ax[0].legend()

ax[1].plot(epoch,loss,'b',label='Training Loss')
ax[1].plot(epoch,val_loss,'r',label='Validation Loss')
ax[1].legend()

ax[2].plot(epoch,train_dice,'b',label='Training dice coef')
ax[2].plot(epoch,val_dice,'r',label='Validation dice coef')
ax[2].legend()

ax[3].plot(epoch,hist['mean_io_u'],'b',label='Training mean IOU')
ax[3].plot(epoch,hist['val_mean_io_u'],'r',label='Validation mean IOU')
ax[3].legend()

plt.show()
```

Figure 3.25: Model de train

- Exemple de prédiction

```
# mri type must one of 1) flair 2) t1 3) t1ce 4) t2 ------- or even 5) seg
# returns volume of specified study at `path`
def imageLoader(path):
    image = nib.load(path).get_fdata()
    X = np.zeros((self.batch_size*VOLUME_SLICES, *self.dim, self.n_channels))
    for j in range(VOLUME_SLICES):
        X[j +VOLUME_SLICES*c,:,:,0] = cv2.resize(image[:,:,j+VOLUME_START_AT], (IMG_SIZE, IMG_SIZE));
        X[j +VOLUME_SLICES*c,:,:,1] = cv2.resize(ce[:,:,j+VOLUME_START_AT], (IMG_SIZE, IMG_SIZE));

        y[j +VOLUME_SLICES*c] = seg[:,:,j+VOLUME_START_AT];
    return np.array(image)


# load nifti file at `path`
# and load each slice with mask from volume
# choose the mri type & resize to `IMG_SIZE`
def loadDataFromDir(path, list_of_files, mriType, n_images):
    scans = []
    masks = []
```

```
[30] def predictByPath(case_path,case):
         files = next(os.walk(case_path))[2]
         X = np.empty((VOLUME_SLICES, IMG_SIZE, IMG_SIZE, 2))
       #  y = np.empty((VOLUME_SLICES, IMG_SIZE, IMG_SIZE))

         vol_path = os.path.join(case_path, f'BraTS20_Training_{case}_flair.nii');
         flair=nib.load(vol_path).get_fdata()

         vol_path = os.path.join(case_path, f'BraTS20_Training_{case}_t1ce.nii');
         ce=nib.load(vol_path).get_fdata()

       #   vol_path = os.path.join(case_path, f'BraTS20_Training_{case}_seg.nii');
       #   seg=nib.load(vol_path).get_fdata()


         for j in range(VOLUME_SLICES):
             X[j,:,:,0] = cv2.resize(flair[:,:,j+VOLUME_START_AT], (IMG_SIZE,IMG_SIZE))
             X[j,:,:,1] = cv2.resize(ce[:,:,j+VOLUME_START_AT], (IMG_SIZE,IMG_SIZE))
       #       y[j,:,:] = cv2.resize(seg[:,:,j+VOLUME_START_AT], (IMG_SIZE,IMG_SIZE))
```



```
[26]     return np.array(image)


    # load nifti file at `path`
    # and load each slice with mask from volume
    # choose the mri type & resize to `IMG_SIZE`
    def loadDataFromDir(path, list_of_files, mriType, n_images):
        scans = []
        masks = []
        for i in list_of_files[:n_images]:
            fullPath = glob.glob( i + '/*'+ mriType +'*')[0]
            currentScanVolume = imageLoader(fullPath)
            currentMaskVolume = imageLoader( glob.glob( i + '/*seg*')[0] )
            # for each slice in 3D volume, find also it's mask
            for j in range(0, currentScanVolume.shape[2]):
                scan_img = cv2.resize(currentScanVolume[:,:,j], dsize=(IMG_SIZE,IMG_SIZE), interpolation=cv2.INTER_AREA).astyp
                mask_img = cv2.resize(currentMaskVolume[:,:,j], dsize=(IMG_SIZE,IMG_SIZE), interpolation=cv2.INTER_AREA).astyp
                scans.append(scan_img[..., np.newaxis])
                masks.append(mask_img[..., np.newaxis])
        return np.array(scans, dtype='float32'), np.array(masks, dtype='float32')

    #brains_list_test, masks_list_test = loadDataFromDir(VALIDATION_DATASET_PATH, test_directories, "flair", 5)
```

```
# mri type must one of 1) flair 2) t1 3) t1ce 4) t2 ------- or even 5) seg
# returns volume of specified study at `path`
def imageLoader(path):
    image = nib.load(path).get_fdata()
    X = np.zeros((self.batch_size*VOLUME_SLICES, *self.dim, self.n_channels))
    for j in range(VOLUME_SLICES):
        X[j +VOLUME_SLICES*c,:,:,0] = cv2.resize(image[:,:,j+VOLUME_START_AT], (IMG_SIZE, IMG_SIZE));
        X[j +VOLUME_SLICES*c,:,:,1] = cv2.resize(ce[:,:,j+VOLUME_START_AT], (IMG_SIZE, IMG_SIZE));

        y[j +VOLUME_SLICES*c] = seg[:,:,j+VOLUME_START_AT];
    return np.array(image)


# load nifti file at `path`
# and load each slice with mask from volume
# choose the mri type & resize to `IMG_SIZE`
def loadDataFromDir(path, list_of_files, mriType, n_images):
    scans = []
    masks = []
```

Notebook Resources

```
showPredictsById(case=test_ids[1][-3:])
showPredictsById(case=test_ids[2][-3:])
showPredictsById(case=test_ids[3][-3:])
showPredictsById(case=test_ids[4][-3:])
showPredictsById(case=test_ids[5][-3:])
showPredictsById(case=test_ids[6][-3:])


# mask = np.zeros((10,10))
# mask[3:-3, 3:-3] = 1 # white square in black background
# im = mask + np.random.randn(10,10) * 0.01 # random image
# masked = np.ma.masked_where(mask == 0, mask)

# plt.figure()
# plt.subplot(1,2,1)
# plt.imshow(im, 'gray', interpolation='none')
# plt.subplot(1,2,2)
# plt.imshow(im, 'gray', interpolation='none')
# plt.imshow(masked, 'jet', interpolation='none', alpha=0.7)
# plt.show()
```

Figure 3.26: Code de prédiction

- Evaluation

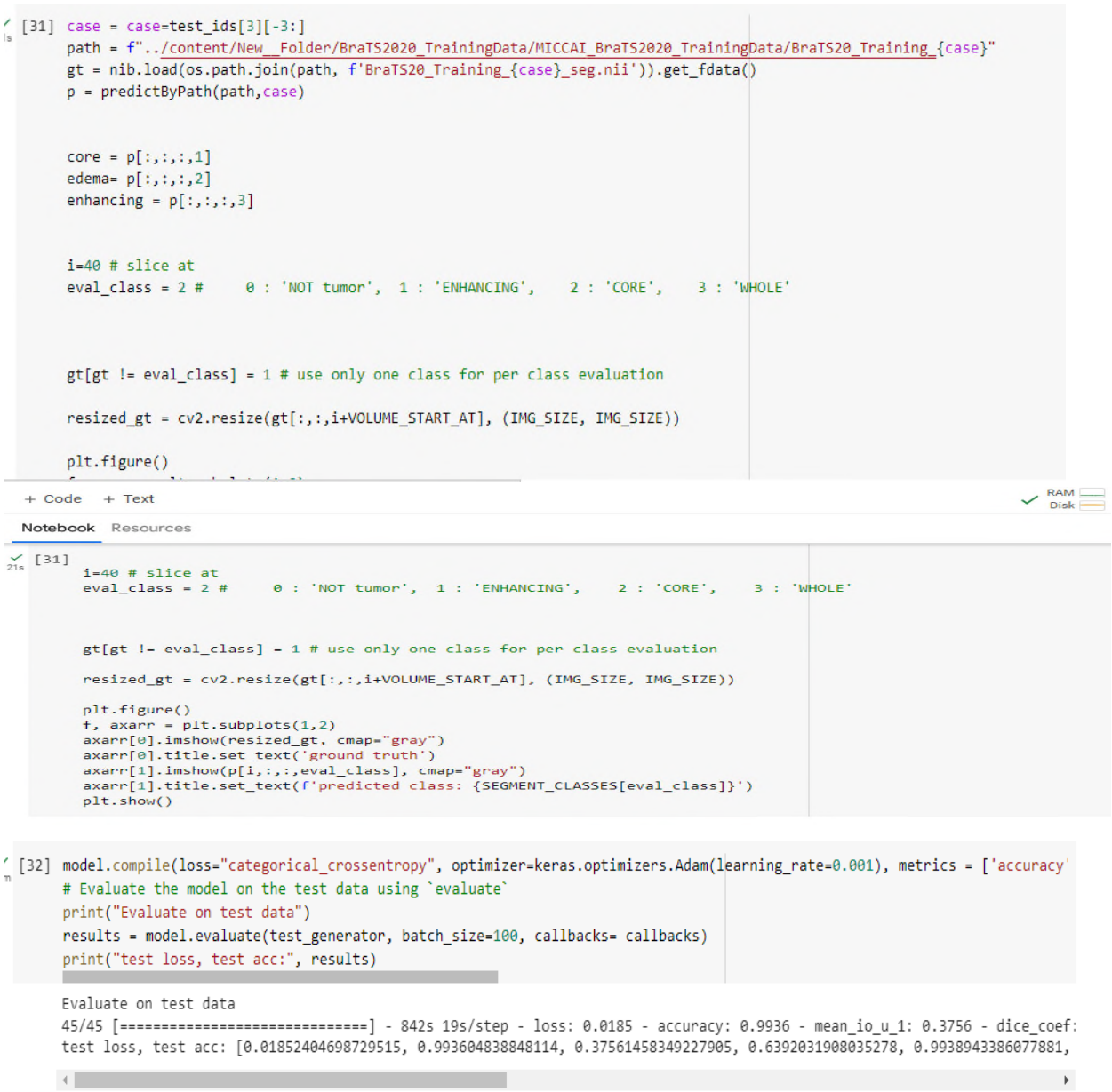
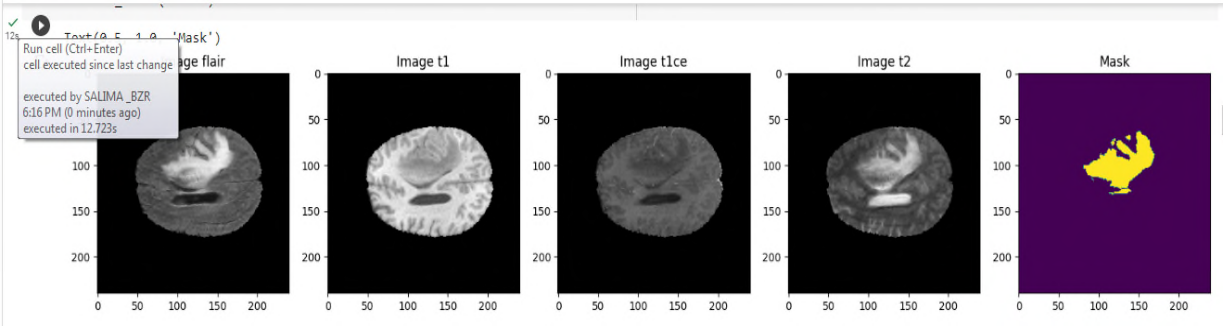


Figure 3.27: Code d'évaluation model

3.3.2. Résultats

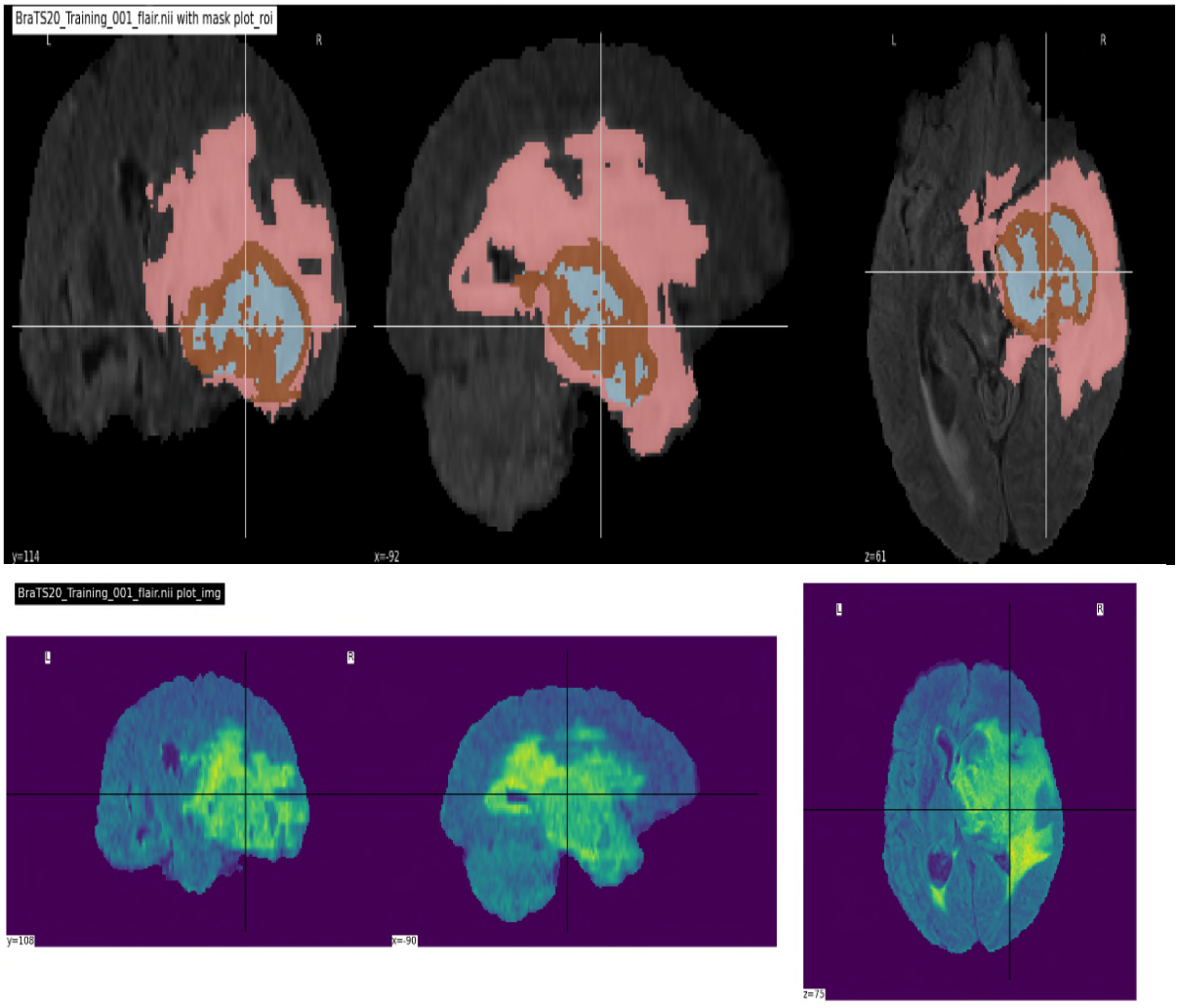
L'ensemble de données d'images utilisé dans notre projet est l'ensemble de données Brats2020, qui contient des images d'IRM 3D au format "nii". Pour visualiser correctement cet ensemble de données et comprendre les images afin de réaliser d'autres opérations, nous avons utilisé plusieurs bibliothèques d'imagerie neuronale telles que "nilearn" et "nibabel". Ces bibliothèques ont

été utilisées pour charger les images au format "nii" et les afficher sous différentes formes. Les images données ci-après: [41]



, Figure 3.28: Représentation visuelle d'une image IRM fournie dans le jeu de données

Afficher des segment de tumeur en utilisant différents effets



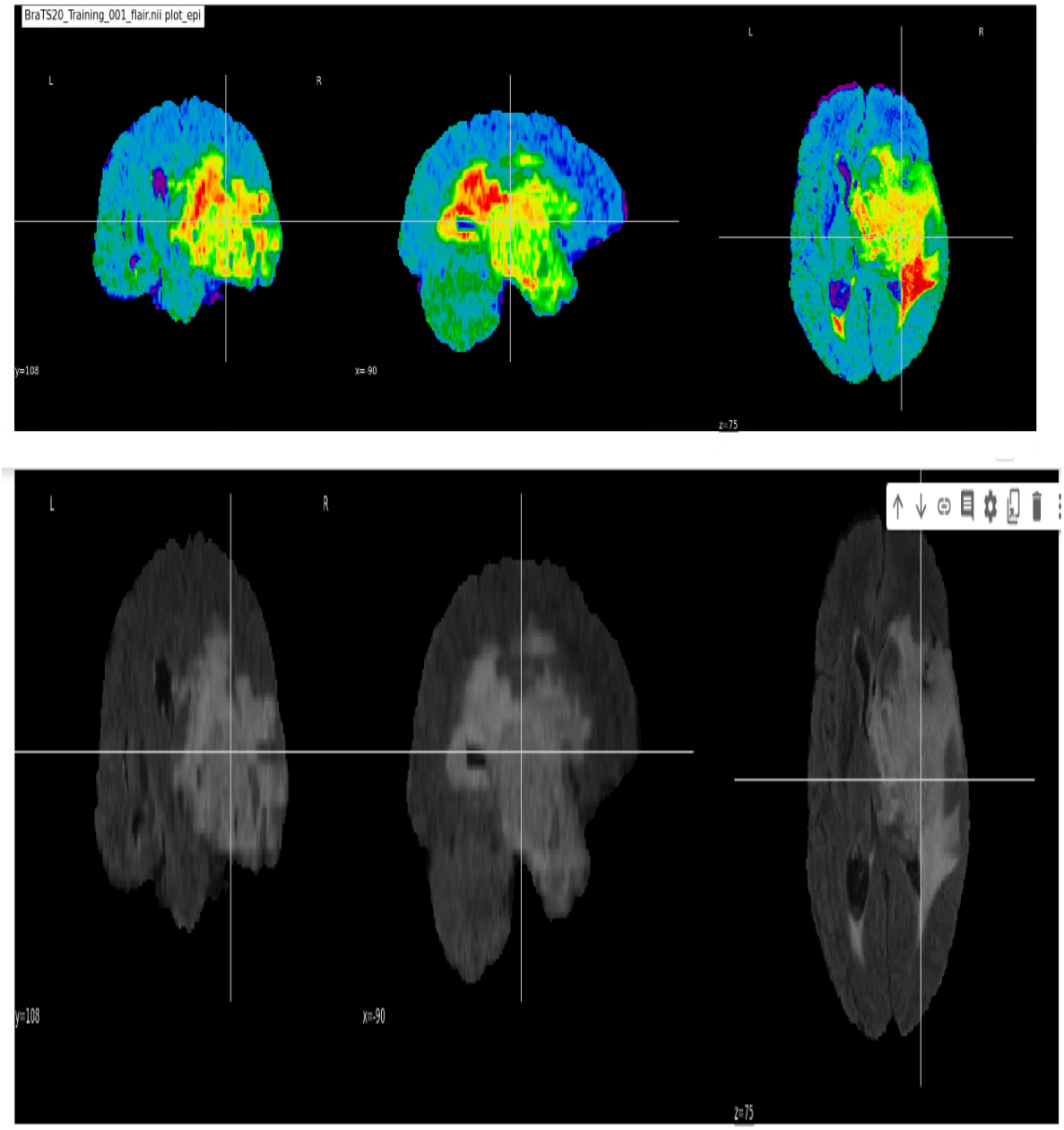


Figure 3.29: Afficher des segment de tumeur

- Nombre de données utilisées pour training/test/validation
  La division de l'ensemble de données BRATS2020 en ensembles d'entraînement, de validation et de test permet d'entraîner, d'optimiser et d'évaluer les performances des modèles de segmentation des tumeurs cérébrales.

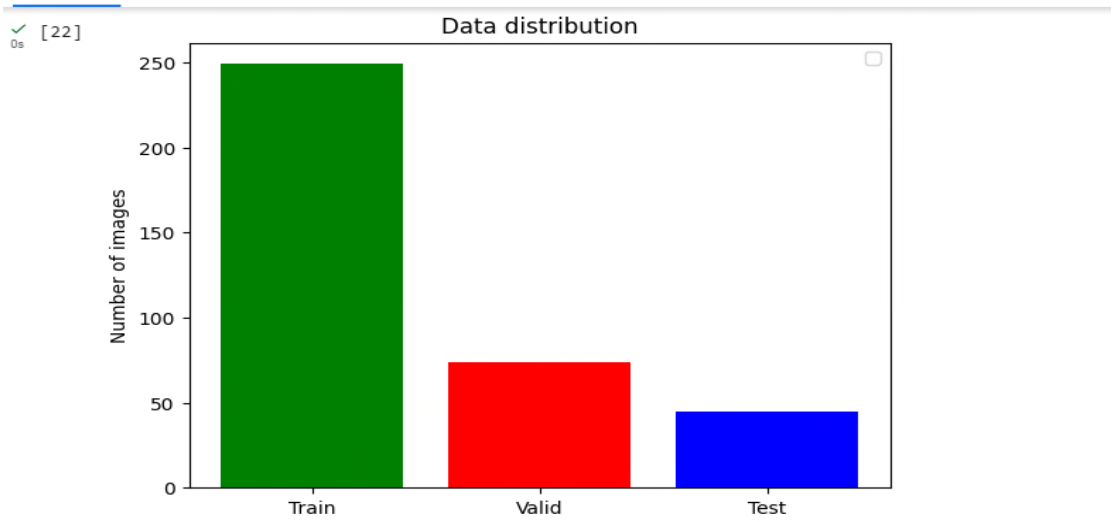


Figure 3.30: Nombre de données utilisées

- Entraînement

Le modèle a été entraîné pendant 45 époques et enregistré dans un fichier h5 nommé "model_per_class.h5". Pendant l'entraînement du modèle, nous avons enregistré toutes les métriques pour chaque époque. Les graphiques présentent les métriques d'entraînement, où la ligne bleue représente la métrique d'entraînement et la ligne rouge représente la métrique de validation. L'axe des ordonnées (y) indique le nombre d'époques, tandis que l'axe des abscisses (x) représente le score.

(A) montre la précision d'entraînement et de validation à différentes époques. On observe que la précision d'entraînement est légèrement supérieure à celle de la validation, et qu'elles atteignent un plateau aux alentours de 20 époques.

(B) présente la perte d'entraînement et de validation, avec une différence d'environ 0,01. Nous pouvons donc conclure qu'il s'agit d'un bon ajustement du modèle.

(C) illustre une augmentation significative des désaccords avec l'augmentation du nombre d'époques, tant pour l'entraînement que pour la validation.

Enfin, (D) montre la moyenne de l'indice de Jaccard (IoU) pour l'ensemble d'apprentissage et de validation. Le score moyen de l'IoU pour l'entraînement et la validation atteint une valeur supérieure à 0,5 après environ 15 époques, et supérieure à 0,8 après environ 25 époques, ce qui est considéré comme un bon score à obtenir.[41]

(A) (B) (C) (D)

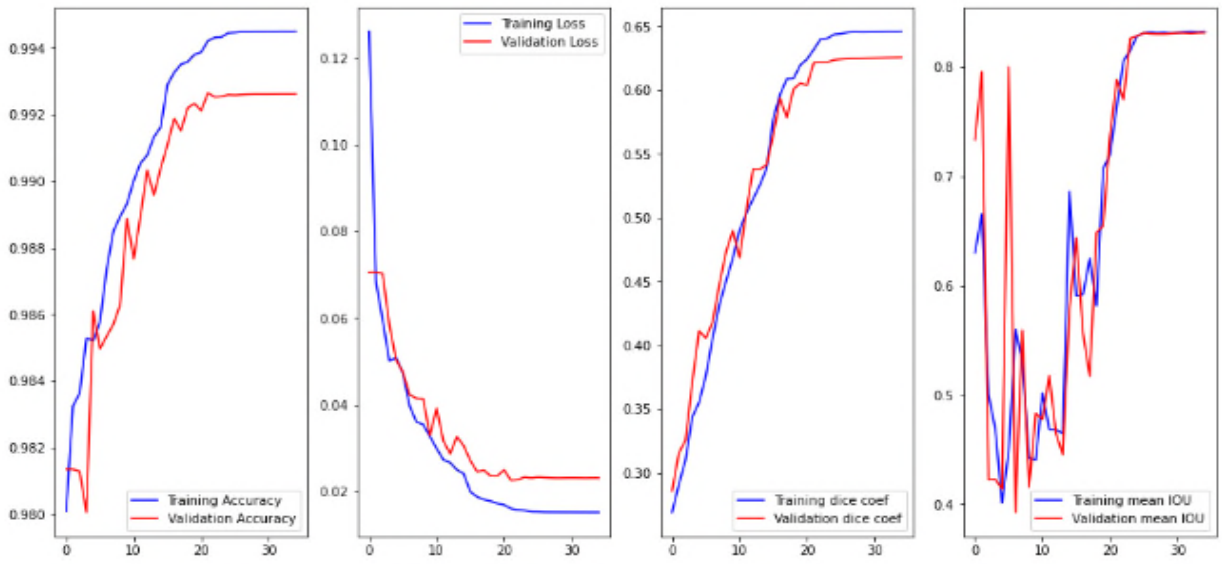
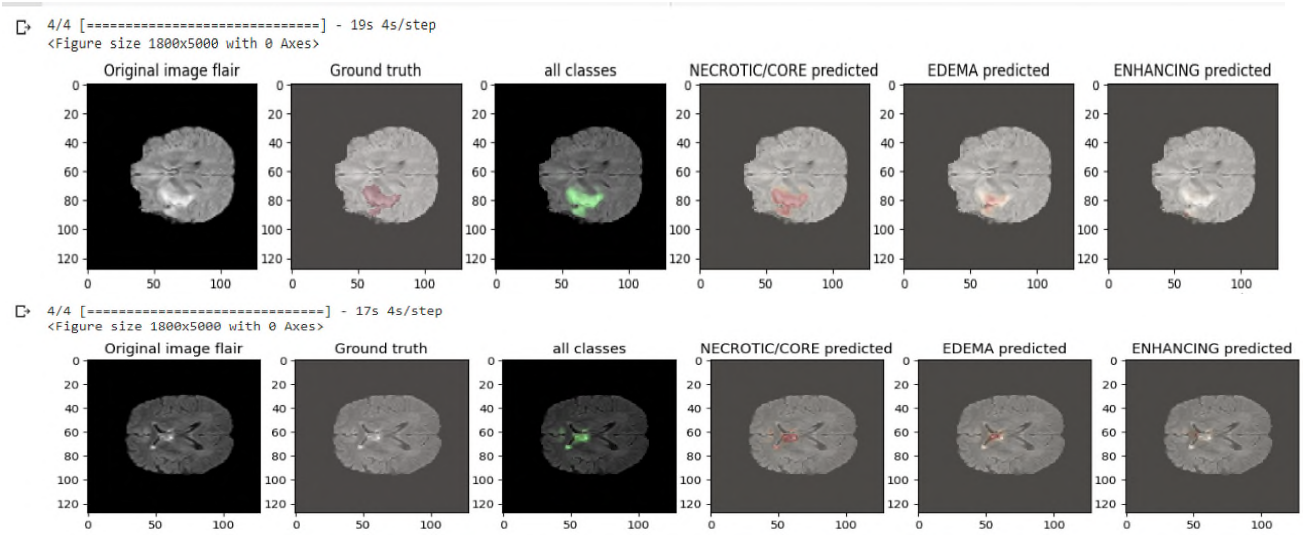


Figure 3.31: (A) Graphique décrivant la précision (accuracy), (B) (loss) le perte pour chaque époque, (C) Graphique décrivant le coefficient de dé (Dice Coefficient), and (D) et la moyenne IOU pour chaque époque.

Résultat de Prédiction

Afin de prédire la présence d'une tumeur dans le cerveau et de classifier ses classes, le modèle a été formé et utilisé. Pour visualiser correctement les prédictions, l'image IRM d'origine, les images de vérité terrain et les classes prédites ont été tracées. Des images aléatoires ont été sélectionnées pour illustrer la précision des prédictions de l'algorithme. Dans les prédictions ci-dessous, nous avons choisi des images d'un cerveau contenant une tumeur, et nous pouvons constater que nos prédictions correspondent exactement à la vérité terrain. Dans la Figure III.32, nous observons les images de gauche à droite, où (A) représente l'image originale, (B) est l'image de vérité terrain, (C) montre l'ensemble de la classification de la tumeur, (D) représente la prédiction pour la région nécrotique/centrale, (E) représente la prédiction pour la région de l'œdème, et enfin (F) représente la région d'amélioration. [41]

(A) (B) (C) (D) (E) (F)



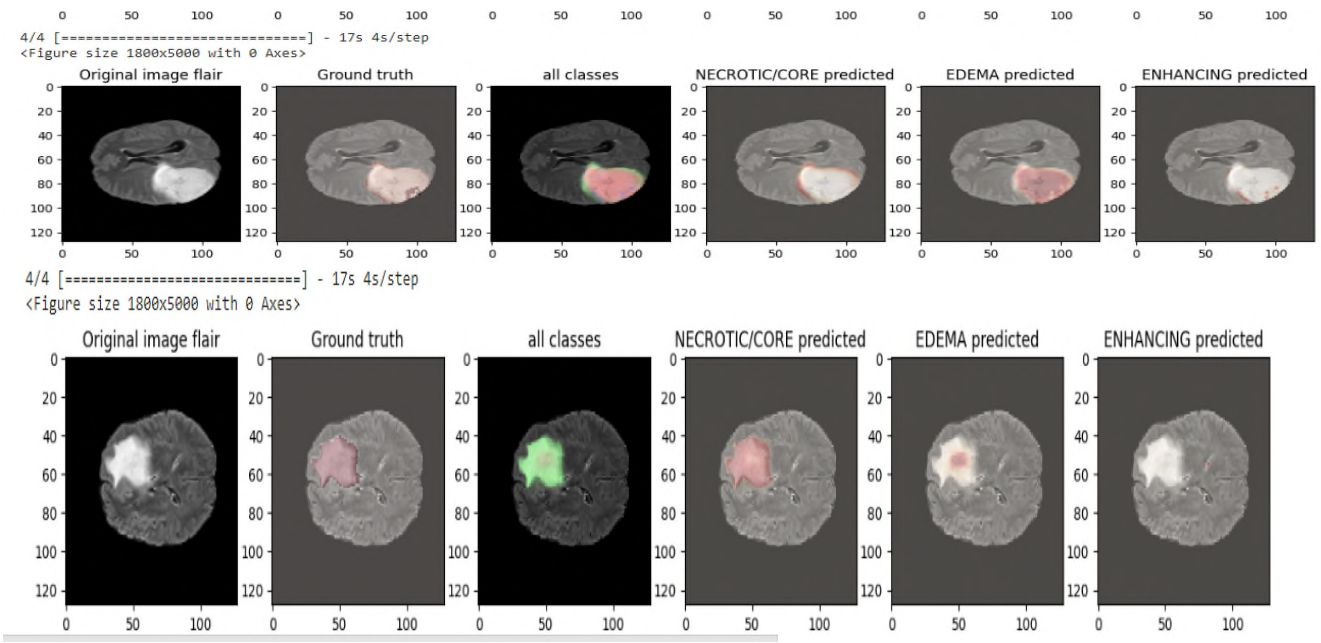
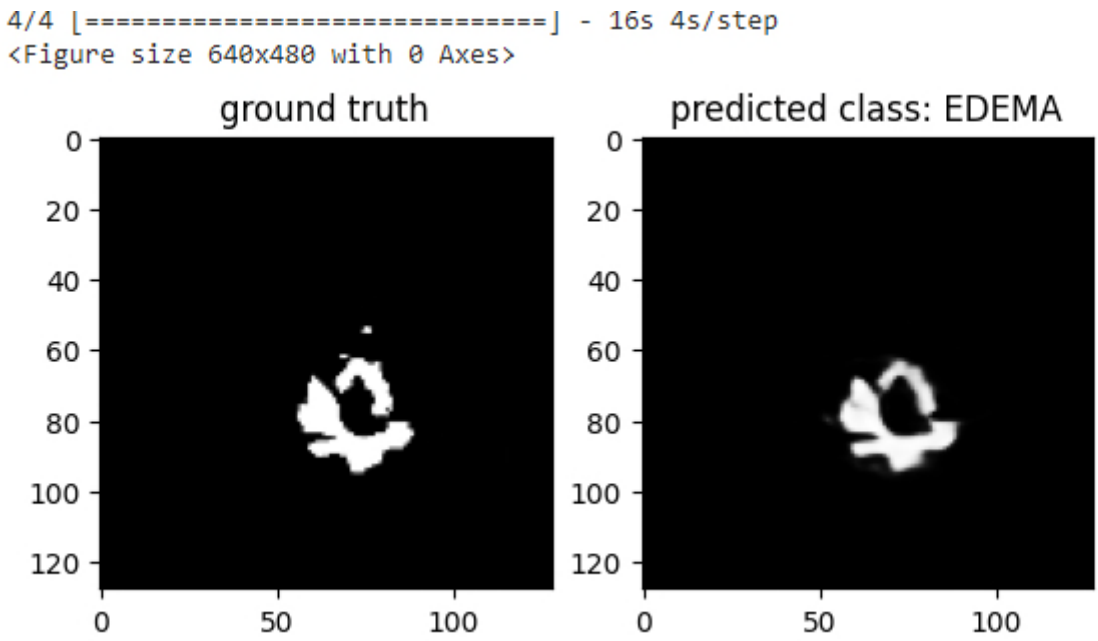


Figure 3.32: Prédiction des données d'échantillon

- Evaluation



```
[32] model.compile(loss="categorical_crossentropy", optimizer=keras.optimizers.Adam(learning_rate=0.001), metrics = ['accuracy'
     # Evaluate the model on the test data using `evaluate`
     print("Evaluate on test data")
     results = model.evaluate(test_generator, batch_size=100, callbacks= callbacks)
     print("test loss, test acc:", results)

Evaluate on test data
45/45 [==============================] - 842s 19s/step - loss: 0.0185 - accuracy: 0.9936 - mean_io_u_1: 0.3756 - dice_coef:
test loss, test acc: [0.01852404698729515, 0.993604838848114, 0.37561458349227905, 0.6392031908035278, 0.9938943386077881,
```

Figure 3.33: Scores métriques pour la formation et la validation

Nous remarquons que la précision du modèle se stabilise à une valeur 0.99, ce qui est un bon résultat pour nos données.

- Comparaison entre les accélérateur

On a fait cette comparaison en termes de temps d'exécution entre les trois

accélérateurs CPU, GPU et TPU.

- Exécution avec le CPU

Figure 3.34: Le temps d'exécution ave le CPU.

- Exécution avec le GPU

Figure 3.35: Le temps d'exécution ave le GPU.

- Exécution avec le TPU

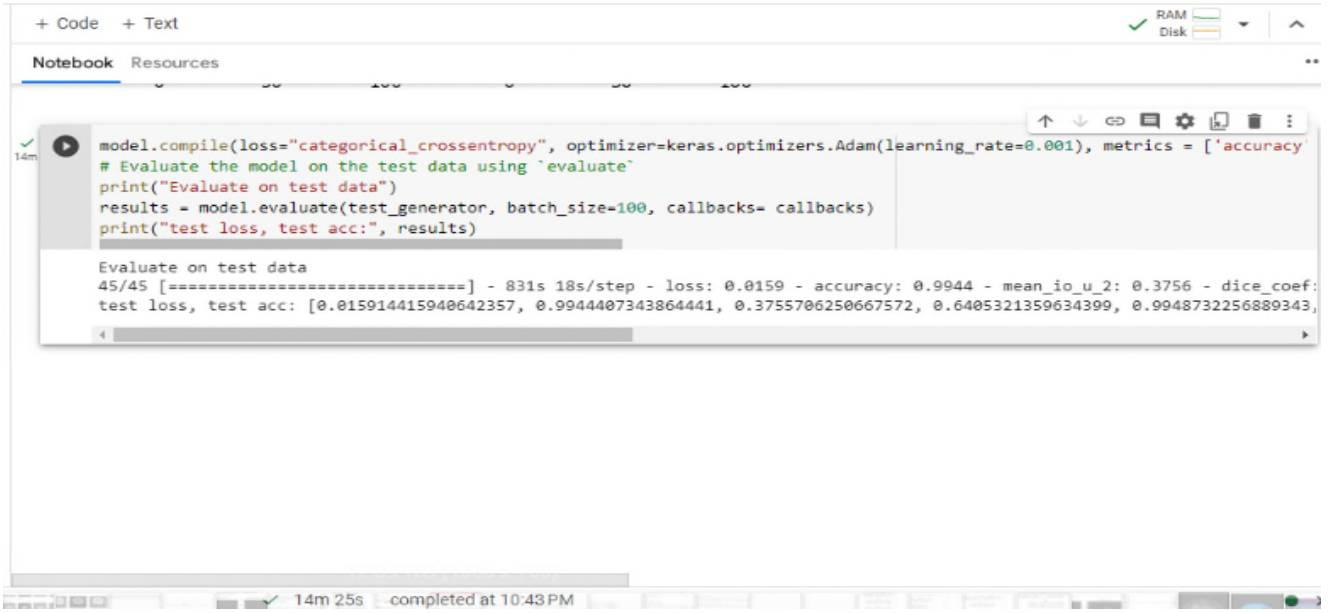


Figure 3.36: Le temps d'exécution ave le TPU.

Nous remarquons que le temps d'exécution de GPU est plus court plusieurs fois que CPU et que TPU.

3.2.4 Comparaison entre différent Modèle

Tableau3.2 :Comparaison entre les modèle

Metric	VGG16	ResNet	U_Net
Accuracy	98.69	97.18	99.4
Loss	04.33	09.85	01.59
Précision	99.22	98.14	99.49
Dice Coefficient	34.56	34.02	64.05

Après avoir effectué une comparaison, nous avons observé que U-Net surpassait les architectures courantes de CNN telles que ResNet et VGG-16 dans chaque échelle dans le domaine du traitement d'images. U-Net s'avère être un outil très utile et précis pour le traitement des images médicales. [41]

Conclusion

Dans ce chapitre, nous avons présenté l'utilisation de la plateforme de travail "Google Colab" et du langage de programmation "Python" en utilisant plusieurs bibliothèques telles que "OpenCV, Keras, NumPy, TensorFlow, etc.". Nous avons constaté que l'environnement complet de Google Colab était extrêmement bénéfique, offrant des ressources matérielles et logicielles de haute qualité pour nos implémentations. Cela nous a permis d'exécuter nos codes dans des conditions optimales. De plus, l'avantage de pouvoir utiliser différents accélérateurs matériels tels que le CPU, le GPU et le TPU a encouragé l'implémentation de modèles de deep learning nécessitant des ressources importantes en termes de temps d'exécution.

Ensuite, nous avons effectué l'implémentation du modèle U-Net dans un cadre d'apprentissage en profondeur pour la détection et la segmentation des tumeurs cérébrales à partir d'images IRM. Le modèle

a été évalué sur des images authentiques fournies par l'ensemble de données BRATS 2020 de Medical Image Computing and Computer-Assisted Interventions. Nous avons obtenu une précision de test de 99,3% en utilisant cet ensemble de données.

En conclusion, nous avons comparé les performances en utilisant trois accélérateurs disponibles dans le Cloud (CPU, GPU et TPU) et avons constaté que l'accélérateur GPU était le plus rapide. Cela démontre l'importance de l'implémentation des modèles de deep learning dans un environnement Cloud.

Conclusion général

Nous avons couvert pendant notre projet l’ Imagerie médicale du cerveau et segmentation des images médicales qui est considéré comme l'un des Grands domaines de la recherche en santé Les progrès de la technologie d'imagerie par résonance magnétique (IRM) ont également permis d'obtenir des images détaillées du cerveau, fournissant des informations précieuses pour le diagnostic et le traitement des maladies du cerveau, y compris les tumeurs.

La segmentation des images médicales est un domaine de recherche vaste et varié. Dans le contexte de l'imagerie cérébrale, la segmentation des images IRM cérébrales a plusieurs objectifs, tels que l'aide au diagnostic, le suivi de la progression de l'état d'un patient et les tests cliniques, entre autres.

Où l'objectif de ce projet était d'utiliser des techniques d'apprentissage en profondeur, en particulier les réseaux de neurones convolutifs (CNN), pour effectuer une segmentation automatique des images IRM cérébrales. Les CNN sont des modèles puissants capables d'apprendre des caractéristiques discriminatoires à partir de grandes quantités de données de formation, ce qui les rend adaptés à la tâche de segmentation complexe. Qui a montré de belles performances ces dernières années.

Nous nous concentrons également sur la segmentation des images médicales du cerveau à partir de l'ensemble de données BRATS2020. Cet ensemble de données a été spécialement conçu pour la segmentation des tumeurs cérébrales à partir d'images IRM 3D. La segmentation consiste à diviser les images en différentes régions anatomiques ou structures d'intérêt, ce qui permet de localiser et de quantifier avec précision les tumeurs.

En utilisant l'ensemble de données BRATS2020, nous pouvons entraîner et évaluer notre modèle (U__Net)de segmentation sur un large éventail de cas cliniques.

Cela nous permettra d'obtenir des résultats précis et fiables, contribuant ainsi à une meilleure compréhension et à un meilleur traitement des tumeurs cérébrales.

En résumé, ce projet se concentre sur la segmentation d'images médicales cérébrales à partir de l'ensemble de données BRATS2020. En utilisant des techniques d'apprentissage en profondeur et des réseaux de neurones convolutionnelles, nous visons à développer un modèle de segmentation automatique des images IRM 3D, ce qui contribuera à l'amélioration du diagnostic et du traitement des tumeurs cérébrales.

En conclusion, Les perspectives dans le domaine de la segmentation des images médicales sont vastes et offrent de nombreuses opportunités de recherche et de développement. Voici quelques-unes des principales perspectives à considérer

Comme Amélioration des performances, Segmentation en temps réel, Intégration de l'apprentissage actif et Validation clinique.

2 Bibliographie

1. **Magnard, B**. (2000). *Caractéristiques d'une image numérique*. Récupéré sur maxicours: https://www.maxicours.com/se/cours/caracteristiques-d-une-image-numerique/#:~:text=Une%20image%20num%C3%A9rique%20est%20un,X%20nombre%20sur%20une%20colonne

2. **CHERGUI, YAKOUT et BOUSSAHA, CHAIMAA.** Détection des anomalies par segmentation des images médecine cérébrales. [Memoire pour l'obtention du diplome de master]. TAIRET, Informatique Génei informatique : s.n., 21/06/2022. p. 4.

3. Le cerveau. *CEA.* [En ligne] France, 16 mars 2023. https://www.cea.fr/comprendre/Pages/sante-sciences-du-vivant/Essentiel-sur-images-medicales-aspx

4. **ATTAR, Tassadit et BENTAYEB, Nawel.** Segmentation des images. [Mémoire De fin de cycle En vue de l'obtention du diplôme De Master en Système Informatique]. 2014/2015. p. 32.

5. Le cerveau. *CEA.* [En ligne] France, 16 mars 2023. https://www.cea.fr/comprendre/Pages/sante-sciences-du-vivant/Essentiel-sur-le-cerveau-aspx.

6. tumeur. *INSTITUT NATIONAL DU CANCER.* [En ligne] FRANCE, 21 Juin 2004. https://www.e-cancer.fr/Dictionnaire/T/tumeur.

7. tumeur bénigne. *INSTITUT NATIONAL DU CANCER.* [En ligne] France. https://www.e-cancer.fr/Dictionnaire/T/tumeur-benigne#.

8. Métastases au cerveau. *LE JOURNAL DES FEMMES.* [En ligne] Mars 2023, France, 04 02 2021. https://sante.journaldesfemmes.fr/fiches-maladies/2692275-métastases-au-cerveau-cerebrales-sympotomes-irm-scanner-traitement-suivre-pronostic/.

9. Sosiété canadienne du cancer. *Stade et grade.* [En ligne] 2023. https://cancer.Ca/fr/cancer-information/what-is-cancer/stage-and-grade.

10. IRM Cérébrale. *ACRIM.* [En ligne] POLYCLINIQUE SAINT-COME/MAISON Médicale, 2019. https://www.acrim.fr/nos-examens/iem/cerebrale/.

11. IRM. *RADIOLOGIE92 .* [En ligne] du Sud Ouest Parisien. https://radiologie92.com/Vos-examens/irm.

12.**CHAHINEZ, OTMANI.** Détection de tumeur cérébrale avec l'apprentissage profond. [Mémoire Présenté pour obtenir le diplôme de master académique en informatique]. BISKRA, Université Mohamed Khider – BISKRA Département d'informatique : s.n., 27 06 2022. pp. 1-2.

13.**Loranger, Elisabeth, et al.** Imagerie par résonance magnétique. *Métodes en neurosciences cognitives.* [En ligne] 2022. https://psy3018.github.io/irm.html.

14. Segentation d'image. *WIKIPÉDIA.* [En ligne] 1 Aout 2017. [Citation : 19 Février 2021.] https://Wikipeadia.org/Wiki/Segmentation-d%27image.

15. **Hakima, Zouaoui.** Clustering par fusion floue de données appliqué à la segmentation d'images IRM. [Mémoire de Magister]. 2007-2008. pp. 31-37.

16.1. **ALI, LABIADI.** Sélection des mots clés basée sur la classification et l'extraction des régiles d'association. [Mémoire présenté à l'université du québec à trois-rivières]. JUIN 2017. p. 22.

2. **Magnard, Benjamin.** Caractéristiques d'une image numérique. *maxicours.* [En ligne] 2000. https://www.maxicours.com/se/cours/caracteristiques-d-une-image-numerique/#:~:text=Une%20image%20num%C3%A9rique%20est%20un,X%20nombre%20sur%20une%20colonne..

17. **Boughaba, Mohammed et Boukhris, Brahim.** L`apprentissage profond (Deep Learning) pour la classification et la recherche d'images par le contenu. OUARGLA, Faculté des Nouvelles Technologies de

l'Information et de la Communication Département d'Informatique : s.n., 2016-2017.

18. **FETHIA, MESBAH.** Détection d'objets par Deep Neural Network à l'aide du modèle YOLO en temps réel. 2021.

19. Intelligence artificielle. *FUTURA.* [En ligne] 13 Aout 2020. https://www-futura-sciences.com/tech/definitions/informatique-intelligence-artificielle-555/.

20. La médecine et l'intelligence artificiel. *IBM.* [En ligne] France. https://www.ibm.com/ca-fr/topics/artustry-intelligence-medicine.

21. Intelligence artificielle et imagerie médicale. *ScienceDirect.* [En ligne] January 2022. https://www.sciencedirect.com/science/article/pii/s0007455121004227 .

22. Diagnostic médical et intelligence artificielle. *COMITE CONSULTATION NATIONAL D'ETHIQUE.* [En ligne] 4, Paris, 11 Février 2005. [Citation : 21 Février 2022.] https://www.ccne-ethique.fr/node/531.

23. L'intelligence artificielle diagnostique et thérapeutique au service du cancer. *GUSTAVE ROUSSY.* [En ligne] 31 03 2023. https://www.gustaueroossy.fr/fr/portrait-lintelligence-artificielle-diagnostique-et-therapeutique-au-service-du-cancer.

24. Deep learning: qu'est-ce que c'est ? *FUTURA.* [En ligne] 2001. https://www.futura-sciences.com/tech/definitions/intelligence-artificielle-deep-learning-17262/

25. Apprentissage Supervisé. *MACHINE LEARNIA.* [En ligne] https://machinelearnia.com/apprentissage-supervise-4-etapes/.

26. Guide de l'intelligence artficielle. *Journal d'un net.* [En ligne] CCM Benchmark. https://www.journaldunnet.fr/web-tech/guide-de-l-intelligence-artificielle/1501309-a-apprentissage-non-supervise/.

27. Apprentissage semi supervisé. *datafranca.* [En ligne] France, 2017-2023. https://datafranca.org/wiki/Apprentissage_semi-supervise%C3%A9.

28. Apprentissage par renforcement. *Wikipedia.* [En ligne] https://fr.m.wikipedia.org/wiki/Apprentissage_par_renforcement.

29. Les couche de réseau convolitif. *Blent.* [En ligne] https://blent-ai/cnn-comment-ca-marche/.

30. Classez et segmentez des données visuelles. *OPENCLASSROOMS.* [En ligne] [Citation : 08 07 2022.] https://openclassrooms.com/fr/courses/4470531-classez-et-segmentation-des-donnees-visuelles/508336-decovrez-les-diffrentes-couches-dun-cnn.

31. VGG et Transfer Learning. *datacorner.fr.* [En ligne] 18, Benoit Cayla by Theme Freesia WordPress, 21 mars 2022. https://datacorner.fr/vgg-transfer.learning/.

32. *Tranfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images.* **Tammina, Strikanth, [éd.].** 10, Indian : s.n., Octobre 2019, Ineternational Journal of Scientific and Research Publication, Vol. 9, p. 9420.

33. Aide au diagnostic médical. *Wikipédia.* [En ligne] 2010. [Citation : 5 janvier 2021.] https://fr.m.wikipedia.org/wiki/Aide-au-diagnostic_m%C3%A9dical.

34. **Allen, Paul.** *Semantic Scholar.* [En ligne] novembre 2015. https://pdfs.semanticscholar.org/88a8/7edeacc80dfeead9a1b0594ef26b2872d9e8.pdf.

35. Python : qu'est-ce que c'est ? *FUTURA.* [En ligne] 2001. https://www.futura-sciences.com/tech/definitions/informatique-python-19349/.

36. Les bibliothèque Python à utiliser pour le machine learning. *Mobiskill.* [En ligne] france. https://mobiskill.fr/blog/conseils-emploi-tech/les-bibliotheque-python-a-utiliser-pour-le-machine-learning/.

37. OpenCV. *wikipedia.* [En ligne] septembre 2010. https://fr.wikipedia.org/wiki/OpenCV.

38. Colaboratory. *Google Colab.* [En ligne] https://research.google.com/colaboratory/faq.html.

39. **Simoon Prud'homme.** MOOV AI. *Initiation au Deep Learning avec Google Colab.* [En ligne] 2018. https://moov.ai/fr/blog/deep-learning-avec-google-colab.

40. **Belaidi, Nada.** U-Net : le réseau de neurones populaire en Computer Vision. *blent.* [En ligne] 08 Mars 2022. https://blent.ai/blog/a/unet-computer-vision.

41. *3D MRI Segmentation using U-Net Architecture for the detection of Brain Tumor.* **Sangui, Smarta, et al.** India : s.n., 2023, Vol. 218, pp. 542-553.