N° Order ……..Département/Faculté/UMBB/2019

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE M'HAMED BOUGARA-BOUMERDES

Faculté de la **T**echnologie

Département **I**ngénierie des **S**ystèmes **E**lectriques

Mémoire de Master

Réalisé par

LAMRI Mohamed Amine

Filière : Génie biomédical

Spécialité **:** Instrumentation biomédicale

Détection et classification des hémorragies intracrâniennes dans les images TDM par le Deep Learning

---

Soutenu le 08 Décembre 2020 devant le jury :

MESSAOUDI	Noureddine	MCA	UMBB	Président
HOCINE	Fayza	MCB	UMBB	Examinateur
AMMAR	Mohammed	MCA	UMBB	Promoteur

Année Universitaire : 2019/2020

REMERCIEMENTS

Je tiens à exprimer mes sincères remerciements et ma profonde gratitude à mon promoteur Mr. AMMAR Mohammed pour sa disponibilité, ses conseils, ses encouragements et pour son aide précieuse qui m'a permis d'aboutir à l'achèvement de ce travail.

Nous remercions Mr. MESSAOUDI Noureddine président du jury et Mme. HOCINE Fayza membre examinateur du jury de nous avoir honorés en acceptant de juger et d'évaluer notre travail.

Enfin, que tous ceux qui ont participé de près ou de loin à la réalisation de ce travail, en particulier l'ensemble des enseignants qui sont à l'origine de tout notre savoir, soient remerciés.

Dédicaces

Je dédie ce modeste travail :

À mes chers parents
À mon frère Sofiane, sa femme et leurs deux adorables petites

filles Serine et Nélia

À mon cher petit frère Salim

À toute ma famille

Et À tous mes amis

…Amine L.

Résumé

Le présent travail se veut comme un outil d'aide à la détection de l'hémorragie intracrânienne (HIC) et de ses cinq (05) sous-types. Le modèle retenu après les différentes expérimentations est à base de l'architecture VGG-16. A partir d'une image TDM cérébrale en entrée notre modèle sera capable de détecter la présence d'une HIC chez un patient, et pourra la classée dans un de ses cinq sous-types avec un taux de classification de 96%. Par ailleurs, des notions importantes sur les HIC et la technique du Deep Learning sont rapportées dans les premières parties de ce mémoire.

Mots-clés : Hémorragie Intracrânienne, TDM, Detection, Classification, Apprentissage Profond, CNNs, VGG-16.

Abstract

The present work is intended as a tool to aid in the detection of intracranial hemorrhage (ICH) and its five (05) subtypes. The model retained after the various experiments is based on the VGG-16 architecture. From an input brain CT image our model will be able to detect the presence of an ICH in a patient, and will be able to classify it in one of its five subtypes with an accuracy of 96%. Otherwise, important concepts of ICH and Deep Learning are reported in the first parts of this thesis.

Keywords : Intracranial Hemorrhage, CT, Detection, Classification, Deep Learning, CNNs, VGG-16.

ملخص

يهدف هذا العمل إلى أن يكون أداة للمساعدة في الكشف عن النزيف الدماغي وأنواعه الفرعية الخمسة. يعتمد النموذج الذي تم الاحتفاظ به بعد التجارب المختلفة على بنية VGG-16 . من خلال صورة التصوير المقطعي المحوسب للدماغ، سيكون نموذجنا قادرًا على اكتشاف نزيف دماغي لدى المريض، وسيكون قادرًا على تصنيفها في أحد أنواعه الفرعية الخمسة بدقة تصل إلى 96٪. بالإضافة إلى ذلك، قمنا بذكر مفاهيم مهمة حول النزيف الدماغي وتقنية التعلم العميق في الأجزاء الأولى من هذه الأطروحة.

كلمات مفتاحية : النزيف الدماغي، التصوير المقطعي، الكشف، التصنيف، التعلم العميق، CNNs، VGG-16 .

TABLE DES MATIERES

Table des matières

LISTE DES FIGURES

Liste des figures

LISTE DES TABLEAUX

Liste des tableaux

LISTE DES ABRÉVIATIONS

Liste des abréviations

AVC	Accident Vasculaire Cérébral
HIC	Hémorragie Intracrânienne
TDM	Tomodensitométrie
HU	Hounsfield Unit
FOV	Field Of View
IRM	Imagerie par Résonnance Magnétique
RF	Radio-Fréquence
CNN	Convolutional Neural Networks
DNN	Deep Neural Networks
FC	Fully Connected
GPU	Graphics Processing Unit
IA	Intelligence Artificielle
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
MNIST	Mixed National Institute of Standards and Technology
ReLU	Rectified Linear Units
RVB	Rouge Vert Bleu
SVM	Support Vector Machine
Tanh	Tangente hyperbolique
CSV	Comma Separated Values
TSV	Tab-Separated Values
SQL	Structured Query Language
Adam	Adaptive moment estimation
SGD	Stochastic Gradient Descent
RMSprop	Root Mean Square prop
Adadelta	Adaptive learning rate algorithm

INTRODUCTION GÉNÉRALE

Introduction générale

"L'hémorragie peut être minuscule mais grave. C'est ce qui rend le travail d'un radiologue si difficile, et c'est pourquoi il arrive que l'on passe à côté de quelque chose. Si un patient souffre d'un anévrisme et qu'il est renvoyé chez lui, il peut mourir", détaille Pratik Mukherjee, professeur de radiologie à l'université de San Francisco [1].

Chaque jour, un grand nombre de personnes souffrant de traumatismes au cerveau est emmené aux urgences. Afin d'établir un diagnostic, les médecins ont recours à un scanner qui réalise environ 30 images par patient. Les analyser est une tâche qui nécessite une très grande concentration, et le nombre de patients à traiter peut parfois être très conséquent.

Afin d'aider les médecins et radiologistes et de leur faire économiser un temps précieux, des outils d'aide au diagnostic capable de traiter les images et de détecter les hémorragies y compris l'HIC (l'hémorragie intracrânienne) commencent à être développé.

Pour parvenir à un tel résultat, un slogan est scandé « L'IA (Intelligence Artificielle) pour sauver des vies ». Les avancés de l'IA sont vouées à bouleverser le monde de la santé, et plus particulièrement sa branche du Deep Learning qui permet entre autres à partir de données radiologiques massives de développer des algorithmes de traitement, détection, classification et segmentation d'images médicales [2].

Dans la lignée et perspective de ce qui a été dit plus haut que s'inscrit notre présent travail. Notre principal objectif est de développer un algorithme par la technique du Deep Learning capable de détecter l'HIC et ses cinq (05) sous-types.

Pour parvenir à cette fin, il sera d'abord nécessaire de chercher et de s'inculquer de quelques notions du domaine médical ainsi que du domaine de l'IA et de ses branches du Machine Learning et du Deep Learning. C'est pourquoi, ce manuscrit sera organisé en trois (03) grands chapitres, à savoir :

Un premier chapitre où nous introduirons d'abord quelques notions de base concernant l'HIC : sa définition, ses cinq (05) sous types, son diagnostic et son traitement. Ensuite, nous aborderons les modalités d'imageries pour le diagnostic des HIC, en l'occurrence la Tomodensitométrie (TDM) et l'Imagerie par Résonance Magnétique (IRM).

Un deuxième chapitre où nous présenterons le Machine Learning, ses domaines d'utilisation, ses approches et ses différents algorithmes. Nous nous intéresserons ensuite, au Deep Learning (qui est la sous branche la plus en vogue du Machine Learning) et plus particulièrement aux réseaux de neurones convolutifs CNNs pour la conception des systèmes de

prédiction et de diagnostic médical. Nous verrons à la fin de ce chapitre les architectures les plus courantes des CNNs et quelques métriques de leur évaluation.

Un dernier chapitre, où nous détaillerons la problématique sur laquelle nous avons travaillé, les différentes ressources matérielles et logiciels que nous avons utilisé, les différentes expérimentations réalisées, et enfin nous terminerons par une discussion des résultats d'évaluation obtenus.

Ce manuscrit sera clôturé par une conclusion générale et quelques perspectives.

CHAPITRE I

Contexte Médical

CHAPITRE I: Contexte Médical

I.1 Introduction

L'hémorragie intracrânienne, saignement qui se produit à l'intérieur du crâne, est un grave problème de santé nécessitant un traitement médical rapide et souvent intensif. Les hémorragies intracrâniennes représentent environ 10% des accidents vasculaires cérébraux (AVC) dans les pays industrialisés, où l'AVC est la cinquième cause de décès [3]. L'identification de l'emplacement et du type de toute hémorragie présente est une étape critique dans le traitement du patient.

Le diagnostic nécessite une intervention urgente. Lorsqu'un patient présente des symptômes neurologiques aigus tels que des maux de tête sévères ou une perte de conscience, des spécialistes hautement qualifiés examinent les images médicales du crâne du patient pour rechercher la présence, l'emplacement et le type d'hémorragie [4]. Le processus est compliqué et prend souvent du temps.

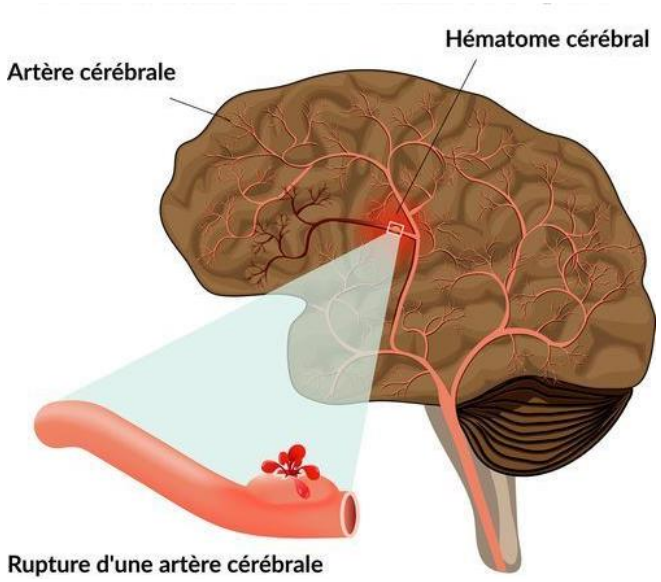


Figure I.1 Accident Vasculaire Cérébral Hémorragique

I.2 Types des hémorragies intracrâniennes

I.2.1 Hémorragie intra parenchymateuse

Elle est causée par la rupture d'un vaisseau sanguin du cerveau. Ce type d'hémorragie se produit le plus souvent chez les personnes souffrant d'hypertension, surtout en cas d'artériosclérose des vaisseaux sanguins. L'hémorragie peut aussi provenir d'un traumatisme craniocérébral ou, rarement, de vaisseaux sanguins présentant une malformation congénitale [5].

I.2.2 Hémorragie intraventriculaire

C'est des saignements qui pénètrent les ventricules du cerveau, qui sont habituellement remplis de liquide céphalorachidien.

Les symptômes d'une hémorragie intraventriculaire sont un œdème cérébral, une pression artérielle anormale, des convulsions, une détérioration clinique majeure accompagnée d'anémie, une hypotension et une acidose métabolique [5].

I.2.3 Hémorragie sous-arachnoïdienne

L'hématome se situe dans le cerveau, entre l'arachnoïde et le tissu cérébral. La cause la plus fréquente est la rupture d'un anévrisme (dilatation de la paroi d'une artère cérébrale) ou de vaisseaux sanguins présentant une malformation congénitale [5].

I.2.4 Hémorragie sous-durale

L'hématome se situe entre deux méninges, la dure-mère et l'arachnoïde. La cause la plus fréquente est un traumatisme craniocérébral [5].

I.2.5 Hémorragie péridurale

L'épanchement de sang (hématome) se situe entre l'os du crâne et la dure-mère (méninge la plus superficielle). La cause la plus fréquente est un traumatisme craniocérébral [5].

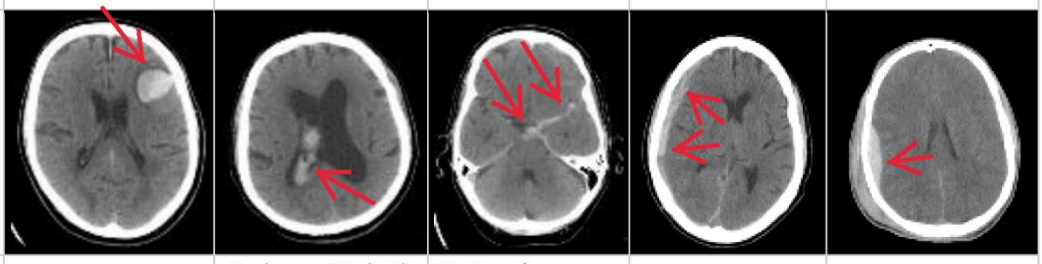
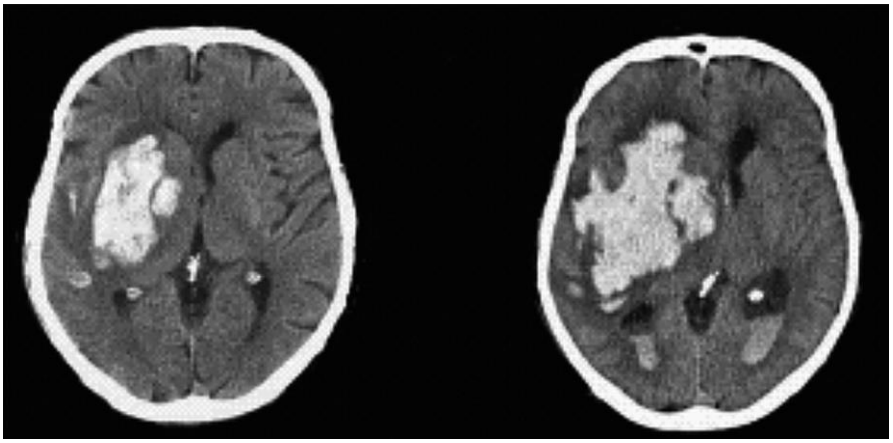
	Intraparenchymal	Intraventricular	Subarachnoid	Subdural	Epidural
Location	Inside of the brain	Inside of the ventricle	Between the arachnoid and the pia mater	Between the Dura and the arachnoid	Between the dura and the skull
Imaging					
Mechanism	High blood pressure, trauma, arteriovenous malformation, tumor, etc	Can be associated with both intraparenchymal and subarachnoid hemorrhages	Rupture of aneurysms or arteriovenous malformations or trauma	Trauma	Trauma or after surgery
Source	Arterial or venous	Arterial or venous	Predominantly arterial	Venous (bridging veins)	Arterial
Shape	Typically rounded	Conforms to ventricular shape	Tracks along the sulci and fissures	Crescent	Lentiform
Presentation	Acute (sudden onset of headache, nausea, vomiting)	Acute (sudden onset of headache, nausea, vomiting)	Acute (worst headache of life)	May be insidious (worsening headache)	Acute (skull fracture and altered mental status)

Figure I.2 Sous types de l'hémorragie intracrânienne

I.3 Diagnostic de l'hémorragie intracrânienne

Le scanner cérébral est l'examen nécessaire et suffisant pour le diagnostic positif d'une hémorragie intracrânienne (HIC). Une hyperdensité montre à l'évidence le site et le volume de l'hémorragie.



Tomodensitométrie cérébrale initial Tomodensitométrie de contrôle (24h)

Figure I.3 Prise de volume progressive d'une HIC en phase aiguë du saignement

L'Imagerie par Résonnance Magnétique (IRM) et en complément, l'angiographie cérébrale sont indiquées pour rechercher une cause de l'hémorragie autre que l'hypertension artérielle.

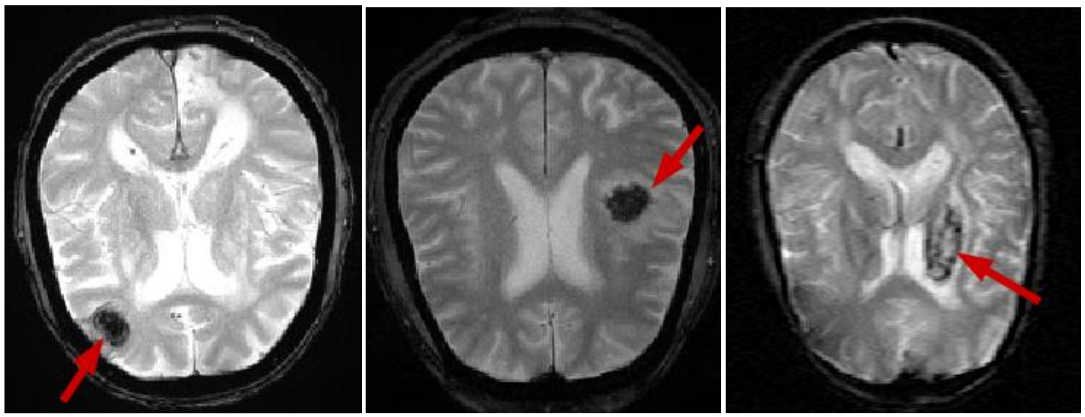


Figure I.4 IRM pondérée en T2* de trois patients atteints d'une HIC. Sur chaque coupe, les tissus hémorragiques sont visibles en hypo-signal (mis en évidence par une flèche rouge).

I.4 Traitement de l'hémorragie intracrânienne

Le traitement de l'HIC spontanée est d'abord médical. A la phase initiale, il concerne la correction prudente et progressive de l'hypertension artérielle, le contrôle de la glycémie, et la prévention des complications de décubitus [6].

Le traitement d'une hémorragie cérébrale peut être neurochirurgical lorsque l'hématome est volumineux et compressif. Son ablation se fait par craniotomie avec aspiration des caillots et hémostase. La récupération neurologique dépend de la localisation et du volume de l'hémorragie initiale. Plus l'hémorragie est petite et plus grandes sont les chances de récupération après rééducation [6].

I.5 Modalités d'imagerie pour le diagnostic

Face à la survenue d'une HIC, les deux examens d'imagerie cérébrale utilisés sont le scanner cérébral sans injection de produit de contraste et à moindre degré l'IRM.

Le scanner cérébral sans injection est actuellement l'examen de premier recours dans l'évaluation d'une HIC en urgence.

L'accès à l'IRM en urgence n'est pas encore possible dans la plupart des hôpitaux, l'IRM nécessite par ailleurs que les patients soient coopérants, ce qui n'est pas toujours le cas dans le contexte de l'HIC aigue.

Dans ce qui suit, une petite introduction aux modalités d'imagerie utilisées dans l'imagerie cérébrale.

I.5.1 La Tomodensitométrie (TDM)

Le scanner est une chaîne radiologique avec un tube à rayons X et un ensemble de détecteurs disposés en couronne. Le principe repose sur la mesure de l'atténuation d'un faisceau de rayons X qui traverse un segment du corps. Le tube et les détecteurs tournent autour de l'objet à examiner. De multiples profils d'atténuation sont obtenus à des angles de rotation différents. Ils sont échantillonnés et numérisés. Les données sont rétro projetées sur une matrice de reconstruction puis transformées en image analogique [7].

Un faisceau de rayons X traversant un objet homogène d'épaisseur x subit une atténuation définie par la relation :

$$Log\ Io/I = \mu x \qquad (1.1)$$

Avec Io : intensité incidente du faisceau

I : intensité émergente

μ : coefficient d'atténuation de l'objet traversé

x : épaisseur de l'objet

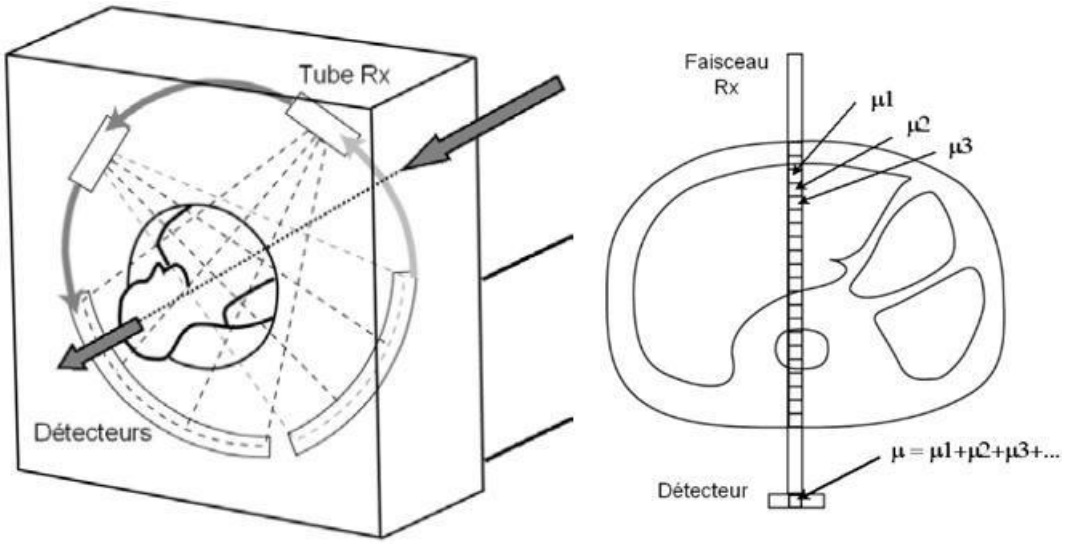


Figure I.5 Le tube et les détecteurs tournent autour du patient. De multiples mesures d'atténuations sont effectuées selon différents angles de rotation du tube. L'atténuation mesurée par un détecteur dépend de toutes les structures traversées et la valeur de μ est une valeur moyenne.

La matrice est un tableau composé de n lignes et n colonnes définissant un nombre de carrés élémentaires ou pixels. Les matrices actuelles sont le plus souvent en 512. A chaque pixel de la matrice de reconstruction correspond une valeur d'atténuation ou de densité. En fonction de sa densité, chaque pixel est représenté sur l'image par une certaine valeur dans l'échelle des gris. Les coefficients de densité des différents tissus sont exprimés en unités Hounsfield HU. L'éventail varie de –1000 à +4000, avec le choix d'une valeur de zéro pour l'eau, -1000 pour l'air et +1000 pour le calcium. L'œil humain ne distinguant que 16 niveaux de gris, les 2000 paliers de densité ne peuvent être vus simultanément sur l'écran. La fenêtre correspond aux densités qui seront effectivement traduites en niveaux de gris à l'écran. Deux paramètres modulables définissent la fenêtre utile de densités [7].

- le niveau (level) : valeur centrale des densités visualisées
- la largeur de la fenêtre (window) détermine le nombre de niveaux de densité.

En augmentant la fenêtre l'image s'enrichit de niveaux de gris mais le contraste diminue entre les structures de l'image. En diminuant la fenêtre, le contraste augmente [8].

I.5.1.1 Chaine radiologique

I.5.1.1.1 Générateur de rayons X

Le générateur alimente le tube à rayons X. Il délivre une haute tension continue (80 à 140 kV) ainsi qu'un milli-ampérage constant (de 10 à 500 mA) [7]. Il a une puissance totale disponible de 50 à 60 kW. Il est le plus souvent placé (« embarqué ») dans le statif.

I.5.1.1.2 Tube

Les tubes doivent être extrêmement performants. En effet ils doivent être capables :

- d‘absorber de fortes contraintes thermiques d‘où la nécessité d'une capacité calorique élevée (exprimée en unités chaleur).
- d'évacuer la chaleur grâce à une dissipation thermique importante (permettant de réaliser une deuxième hélice si la première a porté le tube à sa charge thermique maximale).

Ils sont à anode tournante, à foyer fin de l‘ordre du mm, avec émission continue. Ils doivent en outre supporter les contraintes mécaniques de la force centrifuge des statifs de dernière génération dont la vitesse de rotation est de 0,5 seconde pour 360°.

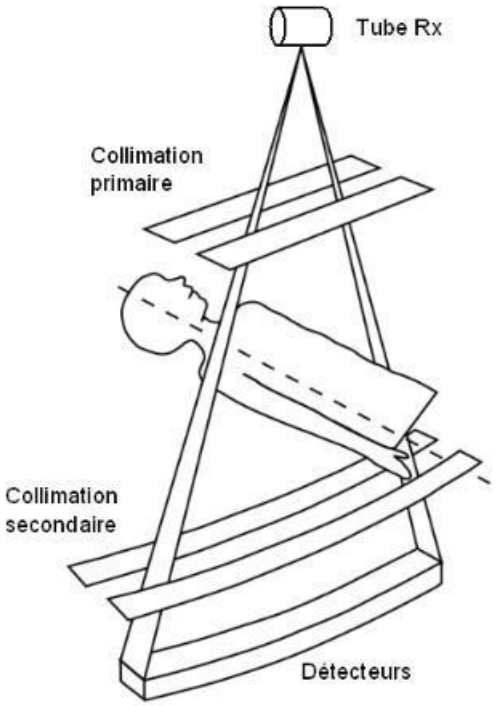


Figure I.6 Collimation primaire et secondaire.

I.5.1.1.3 Filtrage et collimation

Le filtrage et la collimation permettent la mise en forme du faisceau de rayons X.

Le filtrage est effectué par une lame métallique de faible épaisseur. Il permet d‘obtenir un spectre de rayonnement étroit, d‘approcher le monochromatisme.

La collimation primaire est située en aval du filtrage. Elle calibre le faisceau de rayons X en fonction de l‘épaisseur de coupe désirée. Elle limite l‘irradiation inutile. La collimation secondaire est placée avant le détecteur et doit être parfaitement alignée avec le foyer et la collimation primaire. Elle limite le rayonnement diffusé par le patient [7].

I.5.1.1.4 Système de détection

Les détecteurs transforment les photons X en signal électrique. On distingue deux types de détecteurs :

- Chambres d'ionisation au xénon : Les photons X sont directement transformés en signal électrique. Leur efficacité (rendement) est faible (60 à 70% de l'énergie est absorbée).
- Détecteurs solides : ils sont utilisés par la plupart des scanners actuels et sont parfois nommés incorrectement semi-conducteurs. Les photons X sont absorbés par un scintillateur (céramique) et convertis en photons lumineux, eux-mêmes convertis en signal électrique par une photodiode comme illustre la fig I.7. Leur efficacité est excellente et offrent des temps de réponse rapides avec une faible rémanence.

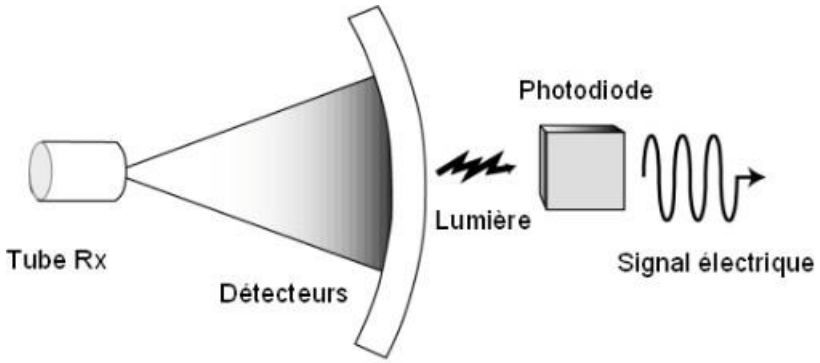


Figure I.7 Principe du détecteur solide

I.5.1.2 Scanner mono-coupe et multi-coupes

Le scanner mono-coupe comporte dans l'axe Z une seule couronne de détecteurs. De 500 à 900 éléments sont disposés dans l'axe x sur environ 50 ° en éventail. Une seule coupe est acquise par rotation.

- Le scanner multi-coupes comporte de multiples couronnes de détecteurs (de 8 à 34 actuellement). Le principe est la subdivision de la couronne de détecteurs dans l'axe Z.

Ainsi si un scanner mono-coupe possède par exemple une couronne avec 900 éléments répartis dans l'axe X, le scanner multi-coupe équivalent, dans le cas d'une subdivision en 16 dans l'axe Z possédera une matrice de 900x16 soit 14400 éléments (fig I.8). Une coupe peut être obtenue par une couronne ou par la combinaison des signaux de plusieurs couronnes de détecteurs adjacente. Les scanners actuels utilisent simultanément 4 couronnes réelles ou combinées pour acquérir 4 coupes simultanées par rotation [7].

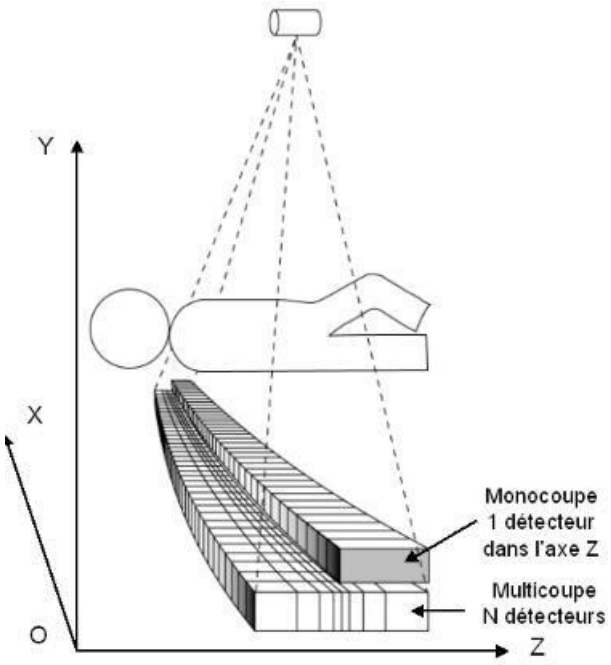


Figure I.8 Comparaison du système de détection en scanner mono-coupe et multi-coupes. L'axe Oz est l'axe du patient.

- Effet de cône : Le principal facteur limitant le nombre de coupes simultanées par rotation

est l'artefact de cône. Sur les scanners multi-coupes, la projection du faisceau de rayons X représente dans l'axe Z un cône [7]. Les rangées centrales de détecteurs sont atteintes perpendiculairement à l'axe de rotation, tandis que les rangées les plus externes sont atteintes obliquement par les rayons X (fig I.10). Cette obliquité dégrade la qualité de l'image en périphérie. Lorsqu'un détecteur périphérique est activé isolément, la largeur du volume traversé par le faisceau de rayons X devient plus importante que la largeur du détecteur. Par ailleurs, cette obliquité entraîne une réduction de l'efficacité des détecteurs périphériques, surtout s'ils sont de petite taille et séparés par de nombreux septa [7]. Si plusieurs détecteurs sont associés ou que le détecteur périphérique est plus large, la largeur du volume traversé est proche de l'épaisseur de coupe.

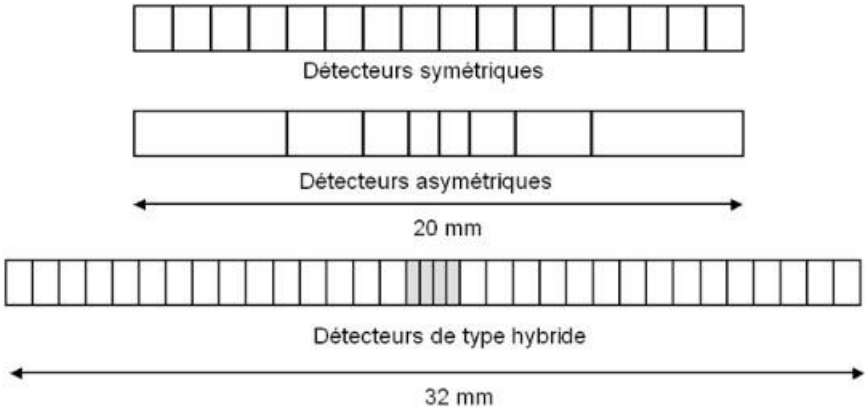


Figure I.9 Différents types de détecteurs des scanners multi-coupes.

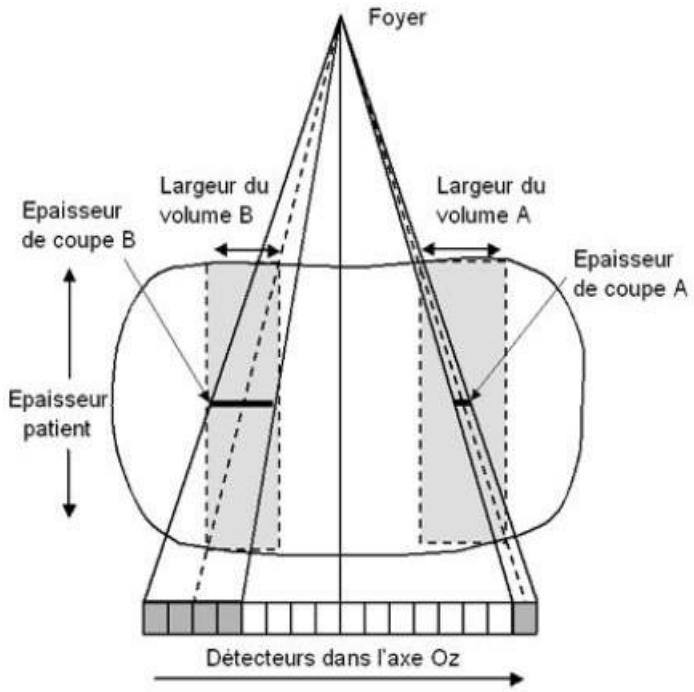


Figure I.10 Effet de cône

- Détecteurs symétriques ou asymétriques : l'arrangement des détecteurs dans l'axe Z

varie selon les constructeurs. Comme il est présenté sur la figure (fig I.9), on distingue ainsi des systèmes à détecteurs :

- symétriques : tous les détecteurs ont la même largeur.
- asymétriques : la largeur des détecteurs croit au fur et à mesure qu'ils s'écartent de la

perpendiculaire à l'axe de rotation.

L'utilisation de détecteurs périphériques plus larges permet de compenser les phénomènes liés à l'effet de cône. Des algorithmes de reconstruction sont nécessaires en cas de système à détecteurs symétriques. En fonction des options technologiques proposées par les constructeurs, le nombre et la largeur des détecteurs gouvernent :

- l'épaisseur de coupes minimale disponible (jusqu'à 0,5 mm).
- le nombre de coupes réalisées avec l'épaisseur minimale (2 à 4).
- la gamme des épaisseurs de coupe disponibles (de 0,5 à 10 mm).
- l'épaisseur maximale du volume couvert par rotation (de 20 à 32 mm actuellement) [7].

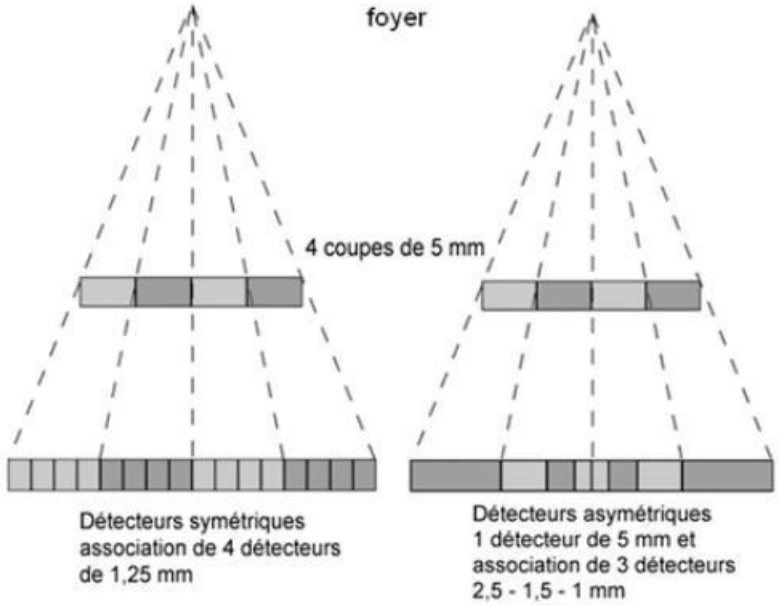


Figure I.11 Coupe de 5 mm par combinaison de détecteurs symétriques et asymétriques.

I.5.1.3 Paramètres d'acquisition

I.5.1.3.1 Collimation primaire

Elle est définie par la largeur de collimation du faisceau de rayons X à la sortie du tube. Elle détermine l'épaisseur nominale de coupe en acquisition mono-coupe. Elle peut varier de 1 à 10 mm [7]. En scanner multi-coupes, la collimation varie en fonction du nombre et des épaisseurs de coupe disponibles. Les valeurs actuelles de collimation primaire vont de 1 mm pour réaliser 2 coupes de 0,5 mm à 32 mm pour obtenir 4 coupes de 8 mm [7].

I.5.1.3.2 kV, mA et temps de rotation

Depuis plusieurs années les scanners hélicoïdaux mono-coupe permettent d'atteindre des temps d'acquisition sur 360° de 0,75 à 0,8 secondes. Le temps de rotation est de 0,5 secondes pour 360° sur les appareils les plus récentes multi-coupes et tous les examens peuvent bénéficier de cette vitesse de rotation. Ce temps de rotation conditionne la résolution temporelle, c'est à dire le temps d'acquisition d'une séquence. Il permet d'obtenir un temps d'acquisition par coupe plus court, de 250 msec par reconstruction partielle et proche de 100 msec par méthode multi-sectorielle. La résolution temporelle dans la coupe s'approche de celle de la tomodensitométrie par faisceau d'électron (TFE) qui est de 50 à 100 ms. Il devient possible avec une synchronisation cardiaque d'accéder à l'imagerie cardiaque [7].

I.5.1.3.3 Pitch

Le pitch se définit comme le rapport entre le pas de l'hélice (distance parcourue par la table pendant une rotation de 360° du tube) et la collimation du faisceau de RX. En acquisition mono-coupe, la collimation correspond à l'épaisseur nominale de coupe. Ce n'est plus le cas en acquisition multi-coupes, où la collimation correspond à 4 fois l'épaisseur nominale de coupe ou plus exactement 4 fois la largeur d'un détecteur. La valeur du pitch n'est donc plus la même d'un constructeur à l'autre selon que l'on considère pour calculer le pitch la collimation (pitch de collimation) ou bien l'épaisseur nominale d'acquisition et donc la largeur d'un détecteur (pitch de détection) [7].

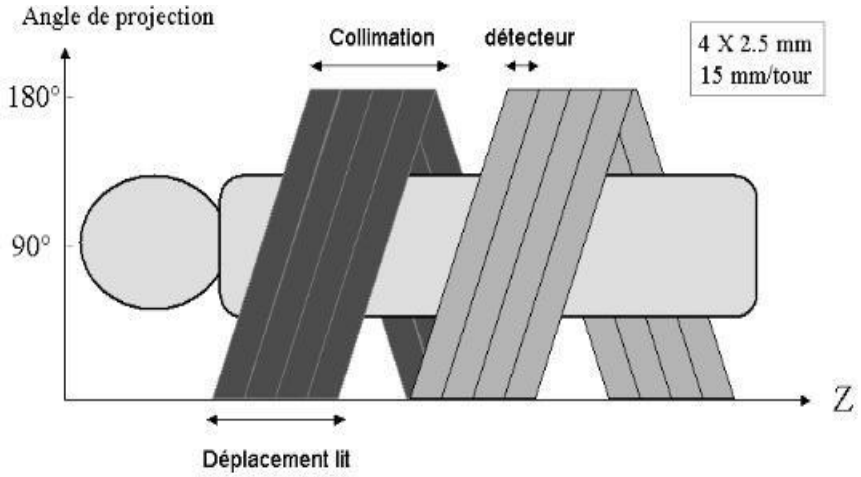


Figure I.12 Scanner multi-coupe (4 coupes simultanées) pitch de collimation de 1,5 et pitch de détection de 6.

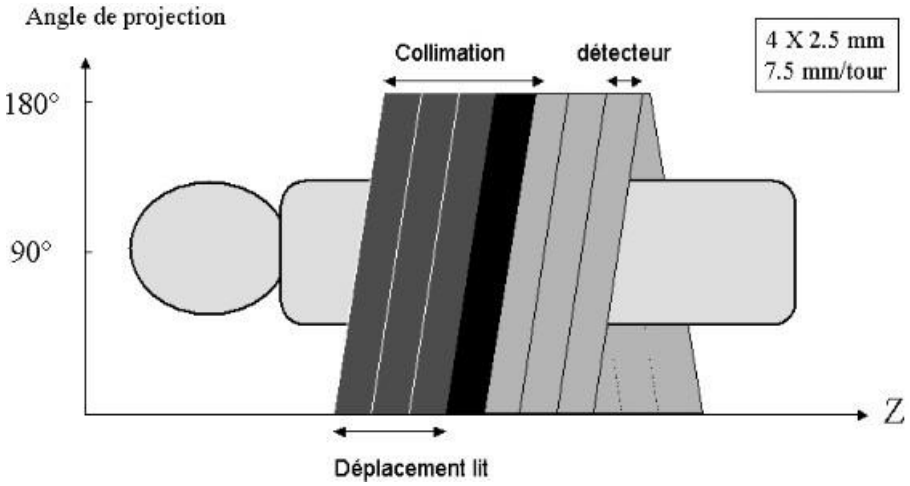


Figure I.13 Cas de chevauchement : Scanner multi-coupes (4 coupes simultanées pitch de 0,75).

I.5.1.4 Paramètres de reconstruction

I.5.1.4.1 Matrice de reconstruction

La matrice de reconstruction est habituellement une matrice de 512x512. Elle détermine en fonction du champ de reconstruction (FOV, Field Of View) la taille du pixel.

Taille du pixel (en mm) = champ de reconstruction (en mm) / nombre de lignes ou de colonnes de la matrice.

I.5.1.4.2 Filtre de reconstruction

Les profils d'atténuation recueillis par les détecteurs sont convertis par une transformée de Fourier en une gamme de fréquence avant l'étape de rétroprojection. Les spectres fréquentiels subissent également une fonction de filtrage. La sélection des fréquences élevées par des filtres « durs » ou spatiaux privilégie la représentation des limites anatomiques des structures tout en rendant plus visible le bruit de l'image. A l'inverse, l'élimination des fréquences élevées par des filtres « mous » ou de densité atténue le bruit et la visibilité des contours permettant une meilleure discrimination des structures à faible écart de densité [7].

Ces filtres optimisent l'image reconstruite selon la structure étudiée. Les filtres « mous » sont adaptés aux structures à faible contraste et les filtres durs aux structures à contraste naturel élevé, telles que l'os, le poumon.

I.5.1.4.3 Algorithmes d'interpolation

En scanner hélicoïdal, les données brutes (projections numérisées) ne peuvent être utilisées directement (contrairement au mode séquentiel) en raison du déplacement continu du patient durant l'acquisition. Si l'on reconstruit les images directement à partir des données ainsi recueillies, la qualité des images sera altérée par des artefacts de mouvement. Il est donc indispensable de calculer des données brutes planes à partir des données volumiques. Ce calcul est réalisé grâce à des algorithmes d'interpolation [7].

I.5.1.5 Qualité de l'image et irradiation

Les principaux facteurs de qualité de l'image en scanner sont la résolution spatiale, la résolution en contraste et la résolution temporelle. Certains artefacts peuvent dégrader la qualité de l'image. La qualité de l'image est indissociable de la dose délivrée donc de l'irradiation [7].

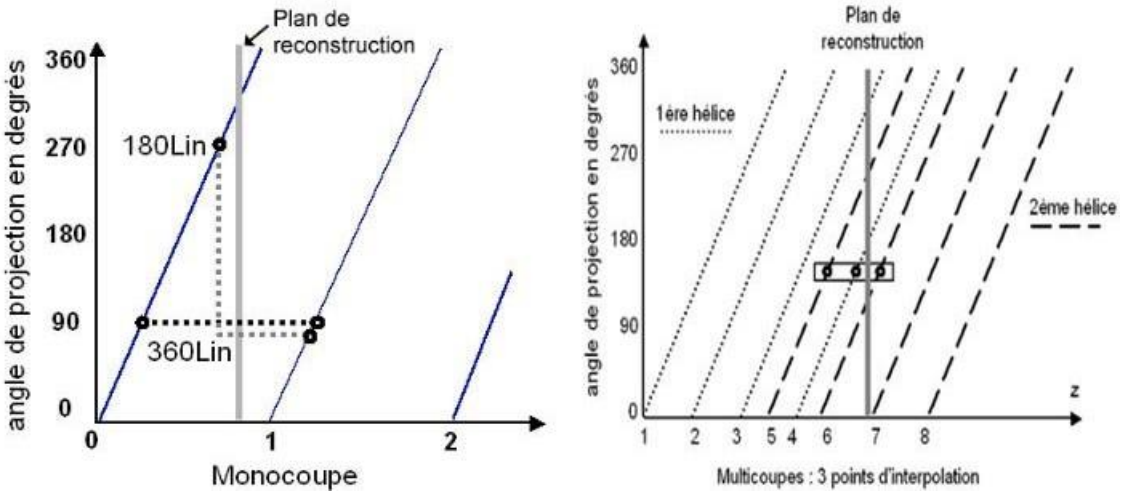


Figure I.14 Algorithme de reconstruction linéaire en scanner mono-coupes et multi-coupes respectivement.

I.5.1.5.1 Résolution en contraste

La résolution en contraste ou en densité est la possibilité de différencier des structures à faible contraste comme par exemple en scanner cérébral, la substance blanche et la substance grise. Elle dépend comme la montre la figure ci-dessous non pas tant du rapport signal sur bruit que du rapport contraste sur bruit [7].

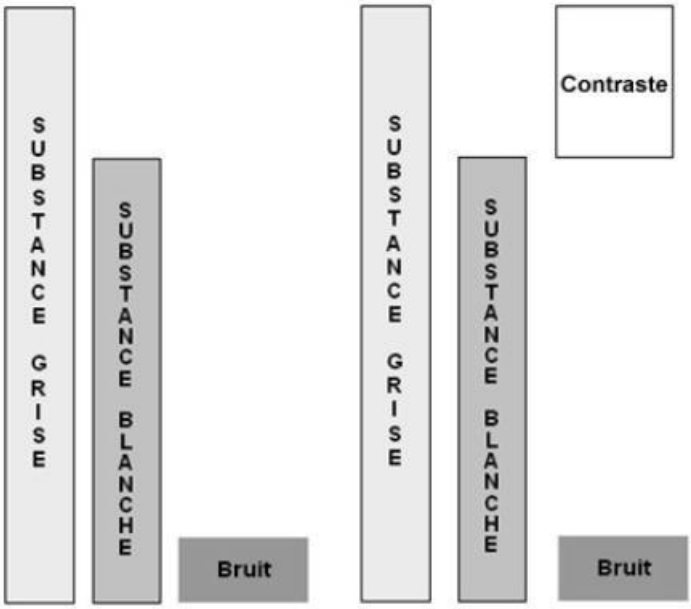


Figure I.16 Rapport contraste bruit.

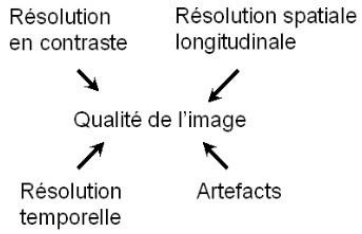


Figure I.15 Facteurs de qualité de l'image en scanner.

I.5.1.5.2 Résolution spatiale

Dans le plan de coupe, elle dépend de la taille du pixel de la matrice de reconstruction, qui est fonction du champ de vue et du nombre de lignes et de colonnes de la matrice (le plus souvent 512). Elle est identique en scanner hélicoïdal à celle obtenue en scanner séquentiel.

La résolution spatiale longitudinale dépend de la taille du voxel dans l'axe longitudinal Oz et correspond à l'épaisseur effective ou réelle de coupe [7].

I.5.1.5.3 Résolution temporelle

Le scanner multi-coupes permet des temps d'acquisitions 4 à 8 fois plus courts que le scanner mono-coupe. L'apport essentiel du scanner multi-coupes est l'amélioration de la résolution temporelle. Il devient possible de réaliser des coupes fines avec un pitch faible, sur un volume important, en un temps très court. Ces gains de temps substantiels sont surtout utiles pour limiter les artefacts d'exploration des organes mobiles, augmenter les possibilités d'exploration en apnée [7].

I.5.1.6 Artefacts

Ils résultent d'une discordance entre les valeurs de densité de l'image reconstruite et les valeurs réelle d'atténuation :

- les artefacts de volume partiel sont limités par le chevauchement des coupes.
- les artefacts de mouvement sont atténués avec les scanners qui offrent des temps d'acquisition courts.
- les artefacts de sous-échantillonnage sont dus à une insuffisance de mesures. Ils se traduisent par des lignes fines au sein de l'image. Pour les corriger, il faut augmenter le nombre de mesures en diminuant la vitesse de rotation ou en scanner multi-coupes en diminuant le pitch [7].

I.5.2 L'Imagerie par résonance magnétique (IRM)

I.5.2.1 Principe physique

L'imagerie par résonance magnétique (IRM) est une technique d'imagerie médicale produisant des images anatomiques ou fonctionnelles à partir d'un champ magnétique et d'ondes radiofréquences (RF). Le principe de l'IRM est de mesurer des paramètres relatifs au noyau d'un atome. L'hydrogène, naturellement présent dans le corps humain, est étudié pour plusieurs raisons. Il est tout d'abord le constituant majoritaire du corps humain en nombre d'atomes [9], ce qui le rend facilement détectable. Il compose aussi les molécules d'eau et de graisse, permettant d'obtenir des images précises des tissus mous. Enfin, l'hydrogène est sujet au phénomène de résonance magnétique nucléaire [10], expliqué ci-dessous.

Chaque atome est composé d'un noyau, contenant au moins un proton, et contenant ou non des neutrons. Le proton est une particule chargée positivement, animée d'un mouvement de rotation autour de son axe qui est caractérisé par une grandeur appelée spin. Les particules possédant un spin, ce qui est aussi le cas des neutrons, portent un moment magnétique les rendant comparables à de petits aimants. Les noyaux pour lesquels les spins des protons et des neutrons ne s'apparient pas tous deux à deux possèdent un spin. Or le noyau d'hydrogène sous sa forme la plus courante est composé uniquement d'un proton : le noyau d'hydrogène possède ainsi un spin nucléaire.

Grâce à cette propriété, chaque proton d'hydrogène présent dans le corps humain est affecté par la présence d'un champ magnétique B_0 : son axe de rotation s'aligne alors dans le même sens que B_0. L'influence du champ magnétique sur l'ensemble des protons est représentée par un vecteur d'aimantation orienté dans la même direction que B_0. En tournant autour de son axe à une fréquence de rotation spécifique, appelée fréquence de résonance ν_f, toujours en présence de B_0, chaque proton peut être sujet au phénomène de résonance magnétique : le proton peut alors passer d'un état énergétique à un autre par absorption d'un photon de fréquence ν_f. Il en résulte un changement d'orientation du vecteur d'aimantation.

Au cours d'un examen IRM, le sujet est soumis à un champ magnétique intense B_0. Les protons d'hydrogène présents dans son corps engendrent un vecteur d'aimantation orienté dans la même direction que B_0. La résonance des protons d'hydrogène est obtenue en pratique par l'application d'une impulsion de radiofréquence (RF) de fréquence égale à la fréquence de résonance de l'hydrogène. Le vecteur d'aimantation change alors d'orientation. Après arrêt de l'impulsion RF, le vecteur d'aimantation retourne progressivement à son orientation de départ, dans la direction de B_0 [11]. Le signal IRM est obtenu en mesurant deux paramètres relatifs à la relaxation de la composante longitudinale (paramètre T1) et transverse (paramètre T2) du vecteur d'aimantation après arrêt de l'impulsion RF. Chaque tissu a des valeurs de T1 et T2 différentes. En modulant spatialement l'intensité du champ magnétique, on peut obtenir une image dite pondérée en T1, en T2, ou en densité de protons selon les paramètres des séquences d'acquisition.

I.5.2.2 Spécificités des images et contraintes techniques

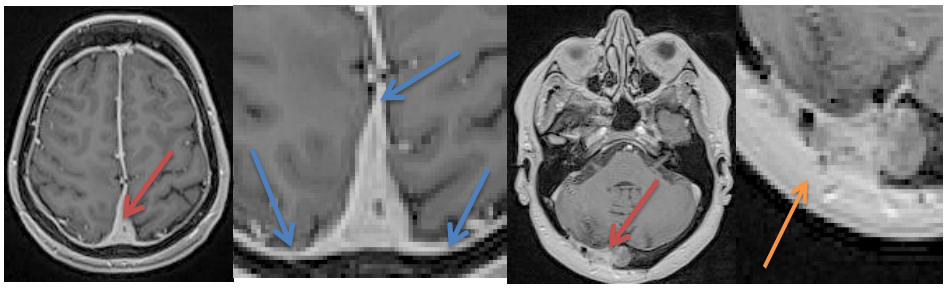
Les images IRM en T1 ou T2 sont dites anatomiques : elles permettent de cartographier la morphologie. Il peut en particulier s'agir de tumeurs ou de structures et tissus sains. L'apparence des tissus dépend de leurs temps de relaxation caractéristiques ainsi que de la quantité de protons d'hydrogène qui les composent [12]. La valeur de T1 d'un tissu peut avoir une influence majeure

sur la valeur de son intensité moyenne sur une image IRM, selon la pondération de cette image. Par exemple sur une image IRM pondérée en T1, les ventricules possèdent un signal de faible intensité (fig I.18a), alors que l'intensité associée à la matière blanche est élevée (fig I.18b). Le contraste est inversé pour une image en T2. Le signal tumoral IRM T1 et T2 peut être homogène ou hétérogène, ce qui renseigne sur la composition moléculaire de la tumeur (fig I.18e et fig I.18f).

L'intérêt de l'IRM réside aussi dans la disponibilité de plusieurs séquences acquises au cours du même examen (fig I.18), outre les images en T1 et T2. En injectant un produit de contraste paramagnétique au patient avant l'examen, il est par exemple possible de rehausser le signal tumoral sur une image pondérée en T1. L'image en T1 rehaussée au Gadolinium (Gd) (fig I.18c et fig I.18g) est l'image de référence dans le diagnostic, la planification et l'évaluation du traitement dans le cas de tumeurs cérébrales [13]. La séquence FLAIR (Fluid Attenuated Inversion Recovery) permet d'obtenir une image en T2 où le signal du liquide céphalo-rachidien est supprimé en modifiant des paramètres d'acquisition, et rend la présence d'œdème identifiable (fig I.18d et fig I.18h).

Si la résolution spatiale isotrope élevée (un voxel correspondant généralement à un volume cubique de $1mm^3$) et le contraste offerts par l'image IRM sont satisfaisants, différents facteurs peuvent compliquer l'interprétation de ces images. D'une part, la différenciation entre tumeur et tissu sain adjacent peut être complexe s'ils sont de composition chimique similaire, et par conséquence de même intensité moyenne IRM. C'est le cas lors de l'étude de méningiomes, situés à proximité des méninges (fig I.17a et fig I.17b), et parfois du crâne (fig I.17c et fig I.17d) qui sont deux structures présentant un signal d'intensité T1-Gd élevée.

D'autre part, l'effet de volume partiel, observé lorsque les voxels d'une image IRM sont de grande dimension, nuit aussi à la qualité de l'image. Diminuer la taille des voxels peut contrer l'effet de volume partiel, au prix d'une diminution du rapport signal-sur-bruit.



(a) (b) (c) (d)

Figure I.17 Images IRM T1-Gd de méningiomes (indiqués par une flèche rouge). Coupe IRM axiale (a) et zoom (b) d'un méningiome (méninges indiquées par une flèche bleue). Coupe IRM axiale (c) et zoom (d) d'un méningiome proche du tissu adipeux (indiqué par une flèche orange).

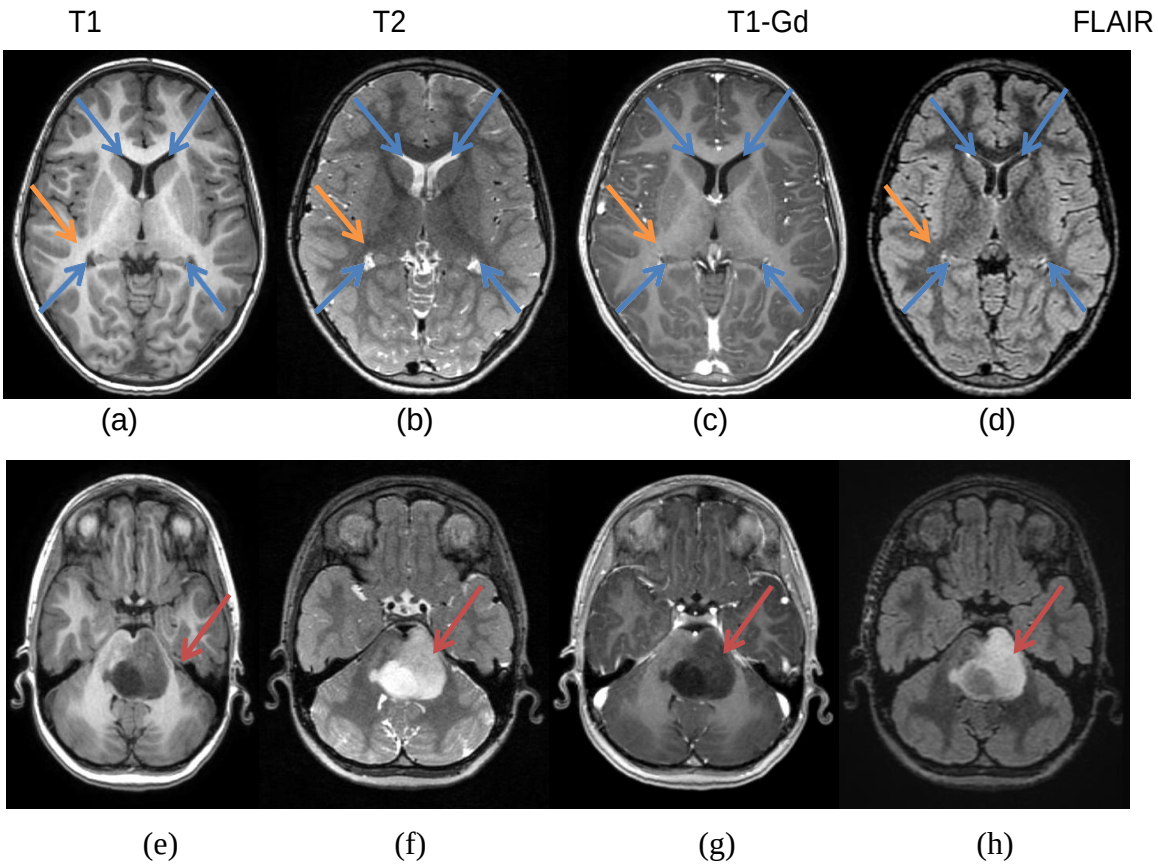


Figure I.18 Images de différentes séquences IRM de tumeurs cérébrales pour un même patient ayant une tumeur du tronc cérébral. Première ligne : visualisation des ventricules (en bleu) et de la matière blanche (en orange) sur une même coupe d'image. Deuxième ligne : visualisation des tumeurs (en rouge) sur une même coupe d'image

I.6 Conclusion

Dans ce chapitre, nous avons d'abord introduit quelques notions de base concernant l'hémorragie intracrânienne (HIC) : sa définition, ses cinq (05) sous types, son diagnostic et son traitement. La bonne compréhension de ces notions médicales nous aidera par la suite à atteindre notre objectif qui est la détection des HIC.

Ensuite, nous avons abordé les modalités d'imageries pour le diagnostic des HIC, en l'occurrence la Tomodensitométrie (TDM) et l'Imagerie par Résonance Magnétique (IRM). Par ailleurs, nous noterons que nous nous sommes plus attardés sur les différentes notions de la tomodensitométrie car notre base de données est constituée d'images TDM (scanner cérébral). Rappelons que le scanner est l'examen réalisé en première intention pour le diagnostic des HIC.

CHAPITRE II

Deep Learning pour la classification des images

CHAPITRE II: Deep Learning pour la classification des images

II.1 Introduction

L'émergence de l'intelligence artificielle (IA), dont les premiers concepts datent des années 50 dans le domaine médical est la conséquence de trois bouleversements radicaux : la numérisation des images médicales permettant leur paramétrage, le développement des algorithmes autorisant l'utilisation des données saisies en langage naturel, et l'apprentissage automatique (machine learning)/ l'apprentissage profond (deep learning) permettant à partir de données radiologiques massives de développer des algorithmes de traitement automatique d'images médicales [2]. L'apprentissage automatique et en particulier l'apprentissage profond suscitent un réel engouement dans l'imagerie médicale : les études abondent, tendant à montrer que l'IA pourrait, demain, diagnostiquer plus sûrement une pathologie qu'un radiologue. Les travaux actuels tendent à utiliser les réseaux de neurones convolutifs (en anglais CNN ou ConvNet pour Convolutional Neural Networks) pour concevoir un système de prédiction et de diagnostic médical [14].

Dans ce chapitre nous présentons les différentes notions de base concernant l'apprentissage automatique, l'apprentissage profond, les CNNs et les différents types de leurs architectures.

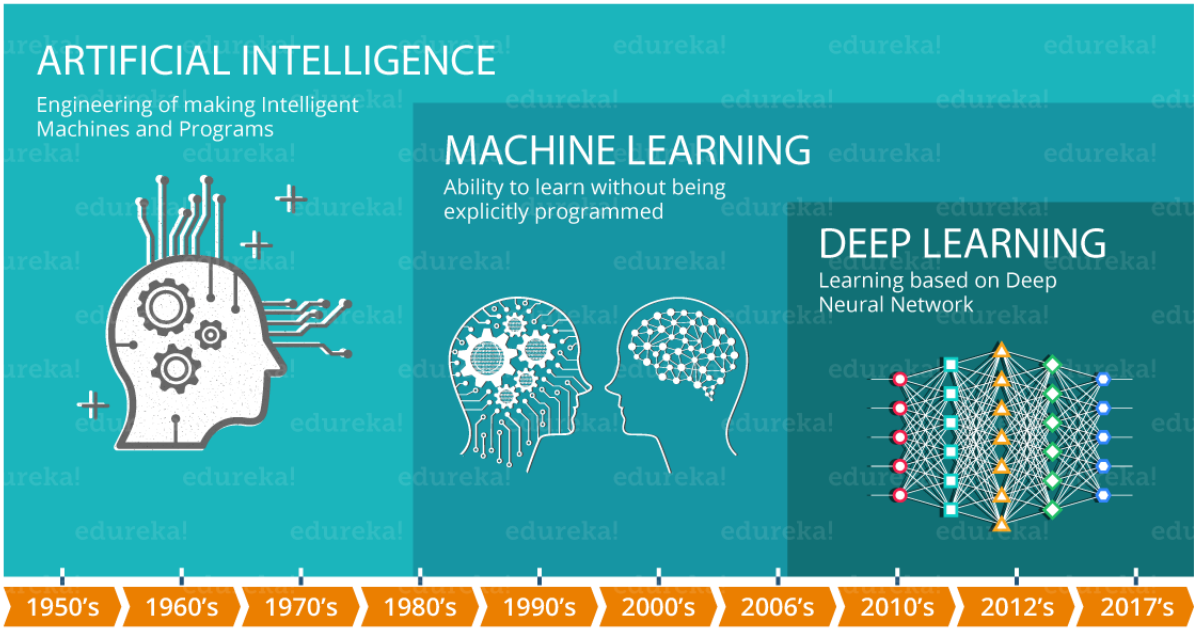


Figure II.1 Chronologie : Intelligence Artificielle, Apprentissage Automatique, Apprentissage Profond

II.2 Apprentissage automatique

L'apprentissage automatique, également appelé apprentissage machine ou apprentissage artificiel et en anglais Machine Learning, est une forme d'intelligence artificielle (IA) qui permet à un système d'apprendre à partir des données et non à l'aide d'une programmation explicite [15].

C'est une discipline consacrée à l'analyse des données. Le but de cette discipline est de créer de la connaissance de manière automatique à partir de données brutes (échantillons). Cette connaissance (ou bien modèle) peut alors être exploitée pour prendre des décisions [16].

II.2.1 Apprentissage itératif

Le machine learning permet aux modèles de se former sur des ensembles de données avant d'être déployés. Lorsqu'un modèle a été formé, il peut être utilisé en temps réel pour apprendre à partir des données. Les améliorations sur le plan de l'exactitude résultent du processus de formation et d'automatisation qui font partie de l'apprentissage automatique [15].

II.2.2 Approches de l'apprentissage automatique

Des techniques d'apprentissage automatique sont nécessaires pour améliorer l'exactitude des modèles prédictifs. Selon la nature du problème métier traité, il existe différentes approches qui varient selon le type et le volume des données. Dans ce qui suit, les différentes approches de l'apprentissage automatique.

II.2.2.1 Apprentissage supervisé

Cette approche a pour objectif la conception d'un modèle reliant des données d'apprentissage à un ensemble de valeurs de sortie (Les données d'entrée sont étiquetées).

L'apprentissage supervisé commence généralement par un ensemble de données bien défini et une certaine compréhension de la façon dont ces données sont classifiées. Vous pouvez, par exemple, créer une application d'apprentissage automatique capable de faire la distinction entre plusieurs millions d'animaux, en se basant sur des images et des descriptions écrites [15].

II.2.2.2 Apprentissage non supervisé

L'apprentissage non supervisé est utilisé lorsque le problème nécessite une quantité massive de données non étiquetées. Par exemple, les applications de réseaux sociaux, telles que Twitter, Instagram et Snapchat, exploitent toutes de très grandes quantités de données non étiquetées. Pour comprendre le sens de ces données, il est nécessaire d'utiliser des algorithmes qui classifient les données en fonction des tendances ou des clusters qu'ils décèlent.

L'apprentissage non supervisé mène un processus itératif, analysant les données sans intervention humaine. Il est utilisé avec la technologie de détection de spam envoyé par e-mail. Les e-mails normaux et les spams comportent un nombre de variables beaucoup trop élevé pour qu'un analyste puisse étiqueter les e-mails indésirables envoyés en masse. En revanche, les discriminants d'apprentissage automatique, basés sur la mise en cluster et l'association, sont appliqués pour identifier les courriers électroniques non désirés [15].

II.2.2.3 Apprentissage semi-supervisé

Les données d'entrée sont constituées d'exemples étiquetés et non étiquetés. Ce qui peut être très utile quand on a deux types de données, car cela permet de ne pas en laisser de côté et d'utiliser toute l'information [17].

II.2.2.4 Apprentissage par renforcement

L'apprentissage par renforcement est un modèle d'apprentissage comportemental. L'algorithme reçoit un feedback de l'analyse des données et guide l'utilisateur vers le meilleur résultat. L'apprentissage par renforcement diffère des autres types d'apprentissage supervisé, car le système n'est pas formé avec un ensemble de données exemple. Au lieu de cela, le système apprend plutôt par le biais d'une méthode d'essais et d'erreurs [15]. Par conséquent, une séquence de décisions fructueuses aboutit au renforcement du processus, car c'est lui qui résout le plus efficacement le problème posé.

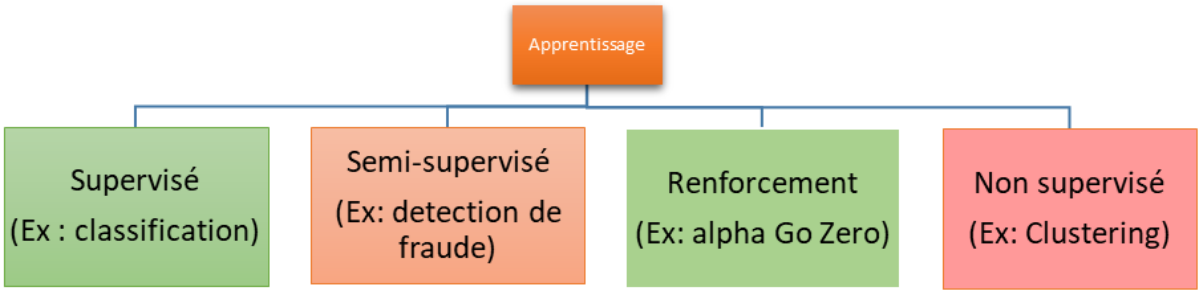


Figure II.2 Les différentes approches de l'Apprentissage automatique

II.3 Apprentissage supervisé (Supervised Learning)

L'apprentissage supervisé est le concept derrière plusieurs applications de nos jours : reconnaissance faciale de nos photos par les smartphones, filtres anti-spam des emails, etc. [18].

Plus formellement, étant donné un ensemble de données D, décrit par un ensemble de caractéristiques X, un algorithme d'apprentissage supervisé va trouver une fonction de mapping

entre les variables prédictives en entrée X et la variable à prédire Y. la fonction de mapping décrivant la relation entre X et Y s'appelle un modèle de prédiction [18].

$$f(X) \rightarrow Y \qquad (2.1)$$

Les caractéristiques (features en anglais) X peuvent être des valeurs numériques, alphanumériques ou des images. Quant à la variable prédite Y, elle peut être de deux catégories:

- **Variable discrète** : La variable à prédire peut prendre une valeur d'un ensemble fini de valeurs (qu'on appelle des classes). Par exemple, pour prédire si un patient a une HIC ou non, la variable Y peut prendre deux valeurs possibles : $Y \in \{POSITIF, NEGATIF\}$
- **Variable continue** : La variable Y peut prendre n'importe quelle valeur. Pour illustrer cette notion, on peut penser à un algorithme qui prend en entrée des caractéristiques d'un véhicule, et tentera de prédire le prix du véhicule (la variable Y).

La catégorie de la variable prédite Y fait décliner l'apprentissage supervisé en deux sous catégories :

- La classification
- La régression

II.3.1 Les algorithmes de classification

Quand la variable à prédire prend une valeur discrète, on parle d'un problème de classification. Parmi les algorithmes de classification, on retrouve : Machines à Vecteurs de Support (Support Vector Machine, SVM), Réseaux de Neurones (Neural Network), Naïve Bayésienne (Naïve Bayes), Régression Logistique (Logistic Regression).

Chacun de ses algorithmes a ses propres propriétés mathématiques et statistiques. En fonction des données d'entrainement (Training set) et nos features, on optera pour l'un ou l'autre de ces algorithmes. Toutefois, la finalité est la même : pouvoir prédire à quelle classe appartient une donnée (exemple : un nouveau patient a-t-il une HIC ou non ?).

Quand l'ensemble des valeurs possibles d'une classification dépasse deux éléments, on parle de classification multi-classes (Multi-class Classification) [18]. La figure ci-dessous illustre les deux types de classifications.

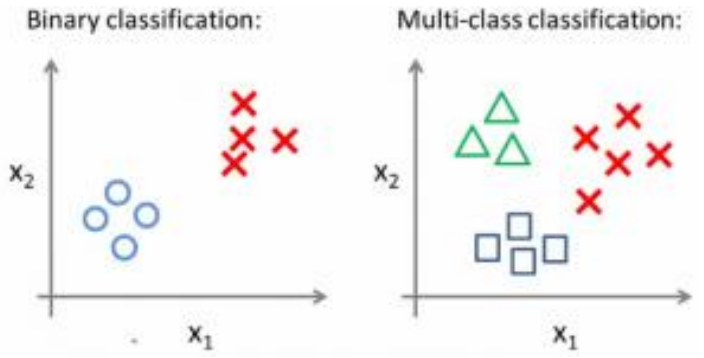


Figure II.3 Classification binaire et classification multi classes

II.3.2 Les algorithmes de régression

Un algorithme de régression permet de trouver un modèle (une fonction mathématique) en fonction des données d'entrainement. Le modèle calculé permettra de donner une estimation sur une nouvelle donnée non encore vue par l'algorithme (qui ne faisait pas partie des données d'entrainement).

Les algorithmes de régression peuvent prendre plusieurs formes en fonction du modèle qu'on souhaite construire. La régression linéaire est le modèle le plus simple : Il consiste à trouver la meilleure droite qui s'approche le plus des données d'apprentissage [18]. La fonction de prédiction sera donc une droite.

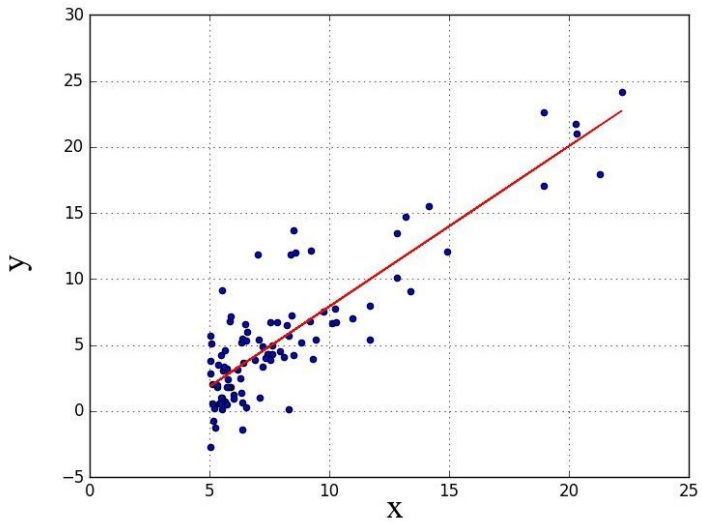


Figure II.4 Régression linéaire

Le modèle prédit par l'algorithme de régression linaire sera de la forme :

$Y = f(X) = \alpha * X1 + \beta$ (2.2) (α et β sont les coefficients de la droite).

Dans la vraie vie, supposer une corrélation linéaire entre les données n'est pas suffisant pour faire des modèles prédictifs complexes. Les données n'ont pas forcément une relation

linéaire entre elles, et plusieurs variables prédictives peuvent être nécessaires pour effectuer une prédiction réaliste. La régression polynomiale et la régression multivariée (à plusieurs variables) permettent de calculer des fonctions de mapping complexes qui s'adaptent bien aux données d'apprentissage.

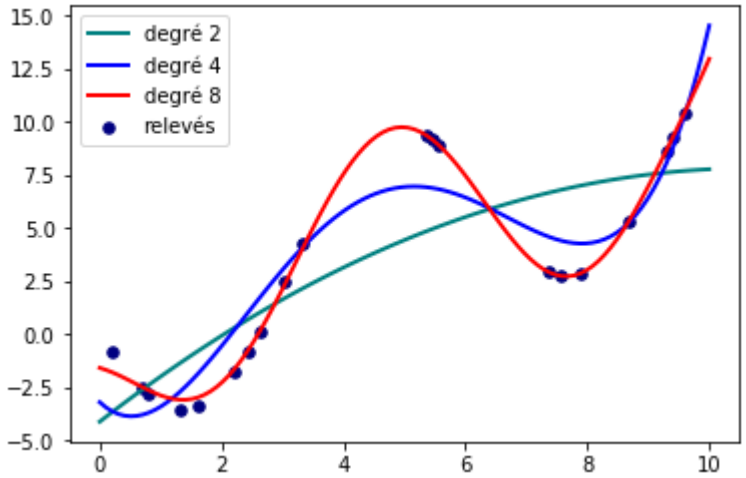


Figure II.5 Régression polynomiale avec des degrés de polynôme différents

Dans la figure ci-dessus, nous avons les tracés de trois fonctions polynomiales avec des degrés de polynôme différents. Généralement, plus on monte dans le degré du polynôme, plus nous avons des fonctions qui s'adaptent bien aux données.

II.4 Deep learning

Deep-Learning ou l'apprentissage profond est un sous-domaine de l'apprentissage automatique qui s'intéresse aux algorithmes inspirés par la structure et la fonction du cerveau, appelés réseaux de neurones artificiels [19]. Elle implique un type particulier de modèle mathématique qui peut être considéré comme une composition de blocs simples d'un certain type, dans une structure multicouche et où certains de ces blocs peuvent être ajustés pour mieux prédire le résultat final.

Dans l'apprentissage en profondeur, un modèle informatique apprend à effectuer des tâches de classification directement à partir d'images, de texte ou de son. Les modèles d'apprentissage en profondeur peuvent atteindre une précision de pointe, dépassant parfois les performances d'un niveau humain [20]. Les modèles sont formés en utilisant une grande masse de données étiquetées et des architectures de réseau neuronal qui contiennent de nombreuses couches.

II.4.1 Importance du Deep Learning

Les niveaux de précision de reconnaissance du Deep Learning n'ont jamais été aussi élevés. Des progrès récents ont amélioré le Deep Learning à tel point qu'il surpasse désormais les capacités humaines dans la réalisation de certaines tâches, telles que la classification d'objets dans des images [20].

Alors que les premières théories concernant le Deep Learning remontent aux années 1980, ce n'est que récemment qu'il est devenu exploitable. En voici les raisons :

- Le Deep Learning nécessite un vaste jeu de données labellisées.
- Le Deep Learning exige une puissance de calcul considérable. Les Processeurs

Graphiques GPU (Graphics Processing Unit) haute performance sont dotés d'une architecture parallèle, qui est efficace pour le Deep Learning. Lorsque l'on y associe des clusters (grappe de serveurs sur un réseau) ou du cloud computing (en français l'informatique en nuage : accès à des services informatiques (serveurs, stockage, mise en réseau, logiciels) via Internet), il devient possible pour les équipes de développement de réduire la durée d'entraînement de réseaux de Deep Learning, de plusieurs semaines à quelques heures ou moins.

II.4.2 Fonctionnement du Deep Learning

La plupart des méthodes de Deep Learning utilisent des architectures de réseaux de neurones, ce qui explique pourquoi il est souvent question de réseaux de neurones profonds pour désigner des modèles de Deep Learning.

Le terme « profond » se rapporte généralement au nombre de couches cachées du réseau de neurones. Les réseaux de neurones classiques ne comportent que 2 à 3 couches cachées, tandis que les réseaux profonds peuvent en compter jusqu'à 150 [21].

L'entraînement des modèles s'effectue à l'aide de vastes ensembles de données labellisées et d'architectures de réseaux de neurones qui apprennent des caractéristiques directement depuis les données, sans avoir à effectuer une extraction manuelle.

II.4.3 Difference entre le Deep Learning et le Machine Learning

Dans un processus de Deep Learning, l'extraction de caractéristiques pertinentes à partir d'images est réalisée de façon automatique. On parle alors d'un apprentissage « de bout en bout » : c'est-à-dire, qu'à partir de données brutes, un réseau se voit assigner des tâches à accomplir et apprend comment les automatiser. Par contre, dans un processus de Machine Learning l'extraction des caractéristiques pertinentes à partir d'images est réalisée

manuellement, et en s'appuyant sur ces caractéristiques, un modèle qui catégorise les objets de l'image est ensuite créé.

Un des avantages majeurs des réseaux de Deep Learning réside dans leur capacité à continuer à s'améliorer en même temps que le volume de données d'entrainement augmente [20].

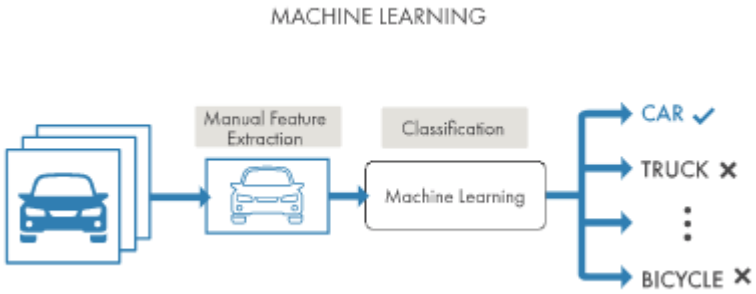


Figure II.6 Processus du Machine Learning

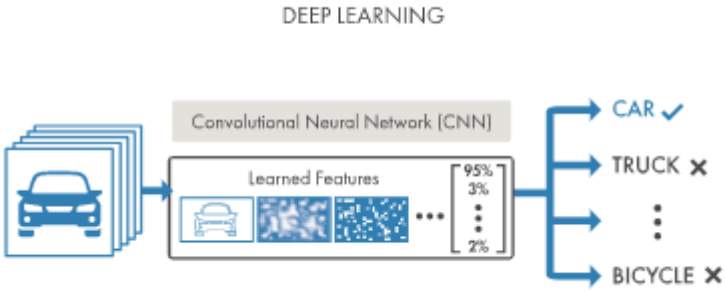


Figure II.7 Processus du Deep Learning

II.4.4 Critères de choix entre le Deep Learning et le Machine Learning

Pour réussir une application de Deep Learning, on a besoin d'un volume de données très important (des milliers d'images) pour entraîner le modèle, en plus d'un ou de plusieurs GPU (processeur graphique) pour traiter les données rapidement.

Si aucun de ces éléments n'est disponible, il est plus judicieux d'utiliser le Machine Learning plutôt que le Deep Learning. Ce dernier est plus complexe, et nécessite un minimum de quelques milliers d'images pour obtenir des résultats fiables. L'utilisation d'un GPU haute performance, permettra une analyse plus rapide de toutes ces images par notre modèle.

II.4.5 Création et entraînement des modèles de Deep Learning

Les trois méthodes les plus répandues dans la classification d'objets avec le processus du Deep Learning sont :

II.4.5.1 L'entraînement à partir de zéro

L'entrainement d'un réseau profond à partir de zéro, nécessite un volume de données labellisées très important et la conception d'une architecture de réseau qui apprendra les caractéristiques et le modèle [20]. C'est une méthode adaptée aux nouvelles applications.

Cette approche n'est pas très répandue car, en raison du grand volume de données et du rythme d'apprentissage, l'entraînement des réseaux peut facilement s'étaler sur plusieurs jours, voire sur plusieurs semaines.

II.4.5.2 Apprentissage par transfert (Transfer Learning)

La méthode d'apprentissage par transfert est la méthode la plus utilisée dans les applications de Deep Learning. Elle consiste à mettre à jour un modèle pré entraîné. Le processus utilise un réseau existant, tel que ResNet, Inception ou LeNet, qu'il faudra enrichir avec de nouvelles données contenant des classes auparavant inconnues du réseau. L'avantage de cette technique est qu'elle nécessite un volume de données beaucoup plus faible, réduisant ainsi le temps de calcul à seulement quelques heures ou minutes [20].

II.4.5.3 Extraction de caractéristiques

Cette approche consiste à utiliser le réseau en tant qu'extracteur de caractéristiques. Sachant que les différentes couches sont chargées d'apprendre certaines caractéristiques à partir des images présentées à l'entrée du réseau, nous pouvons alors récupérer ces mêmes caractéristiques depuis le réseau à n'importe quel moment du processus d'entraînement [20]. Ces dernières sont ensuite utilisées en tant que données d'entrée pour un modèle de Machine Learning tel que les arbres de décision, les forêts aléatoires ou encore les machines à vecteurs de support (SVM). Notons que cette approche de Deep Learning est moins fréquente et plus spécialisée.

II.5 Réseaux de neurones

En informatique, on appelle réseau de neurones un ensemble d'entités (les neurones) interconnectées. Dans la grande majorité des cas, les neurones sont en fait des fonctions calculées par un programme informatique, mais ils sont parfois réalisés sur des circuits électroniques [22].

Le concept des réseaux de neurones artificiels fut inventé en 1943 par deux chercheurs de l'Université de Chicago : le neurophysicien Warren McCullough, et le mathématicien Walter Pitts. Dans un article publié dans le journal Brain Theory, les deux chercheurs présentent leur théorie selon laquelle l'activation de neurones est l'unité de base de l'activité cérébrale [23].

Les méthodes d'apprentissage profond utilisent des architectures de réseaux de neurones, ce qui explique pourquoi les modèles d'apprentissage profond sont souvent appelés réseaux de neurones profonds (Deep Neural Networks – DNN).

Le fonctionnement d'un neurone formel est simple : c'est la somme pondérée des entrées à laquelle on applique une fonction d'activation. Les coefficients de pondération sont appelés poids synaptiques et la fonction d'activation utilise un seuil : la sortie vaut 1 si la somme pondérée est supérieure au seuil et 0 sinon :

$$Y = \begin{cases} 1 & si \sum_k W_k \, i_k > b \\ 0 & sinon \end{cases} \qquad (2.3)$$

Avec : - Y : Sortie du neurone formel

- i_k : Les entrées, avec k : indice de l'entrée

- W_k : Les poids synaptiques

- b : Le seuil d'activation

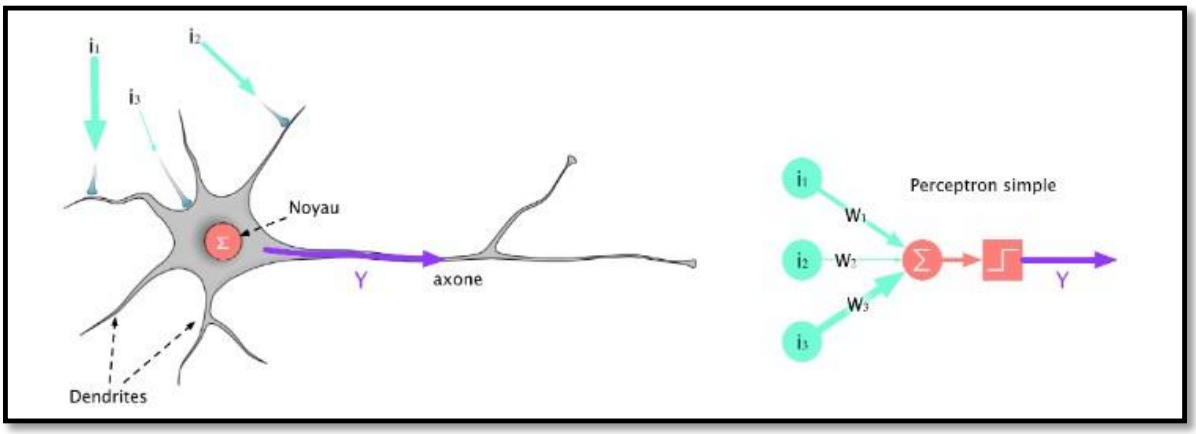


Figure II.8 Schéma d'un neurone biologique à gauche et d'un neurone formel à droit

II.6 Réseaux de neurones convolutifs CNNs

Le réseau neuronal convolutif (CNN, ou ConvNet) est l'un des algorithmes les plus populaires dans le domaine du Deep Learning, il s'est avéré performant dans les tâches de vision par ordinateur telles que la classification d'images, la détection d'objets, la localisation d'objets et le transfert de style neuronal [24]. Dans ce qui suit, on va expliquer les différentes couches qui composent un réseau de neurones convolutionnels : couche de convolution, couche de pooling (mise en commun) et couche entièrement connectée (Fully Connected FC).

II.6.1 Les différentes couches d'un CNN

II.6.1.1 Couche de convolution

Une couche de convolution transforme l'image d'entrée afin d'en extraire des entités. Dans cette transformation, l'image est convolue avec un filtre (kernel ou noyau).

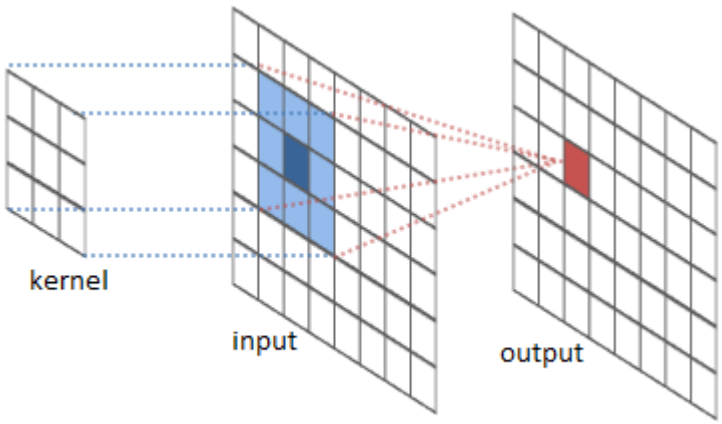


Figure II.9 Convolution d'une image

Un filtre est une petite matrice dont les dimensions sont inférieures à l'image à convolutionner. Il est également connu sous le nom de matrice de convolution ou masque de convolution.

Lors de la convolution d'une image colorée (image RVB : Rouge Vert Bleu) avec trois canaux, le canal des filtres doit également être de trois. En d'autres termes, en convolution, le nombre de canaux dans le filtre doit être le même que le nombre de canaux dans l'image d'entrée [25].

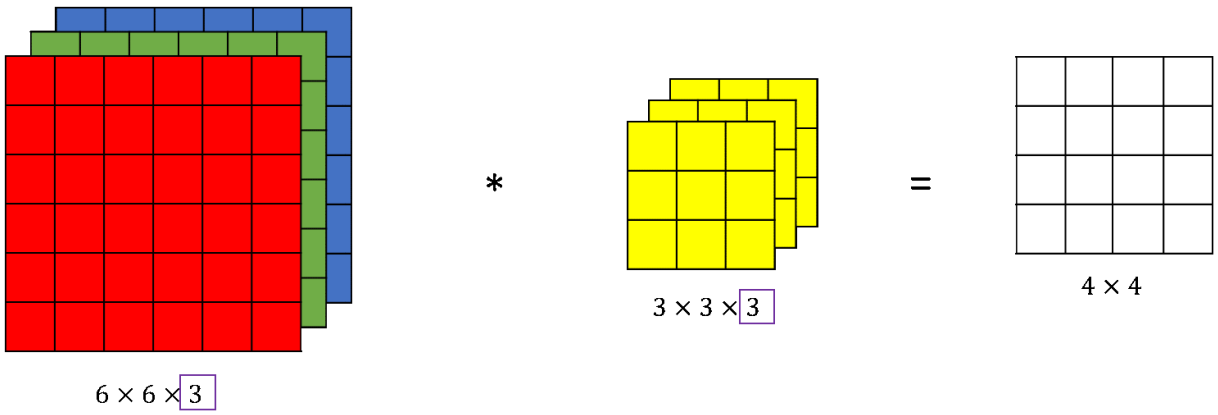


Figure II.10 Convolution sur une image RVB

Lorsque nous voulons extraire plusieurs fonctionnalités d'une image en utilisant la convolution, nous pouvons utiliser plusieurs filtres au lieu d'en utiliser un seul. Dans un tel cas, la taille de tous les filtres doit être la même. Les caractéristiques convolutées de l'image d'entrée sont empilées les unes après les autres pour créer une sortie de sorte que le nombre de canaux soit égal au nombre de filtres utilisés.

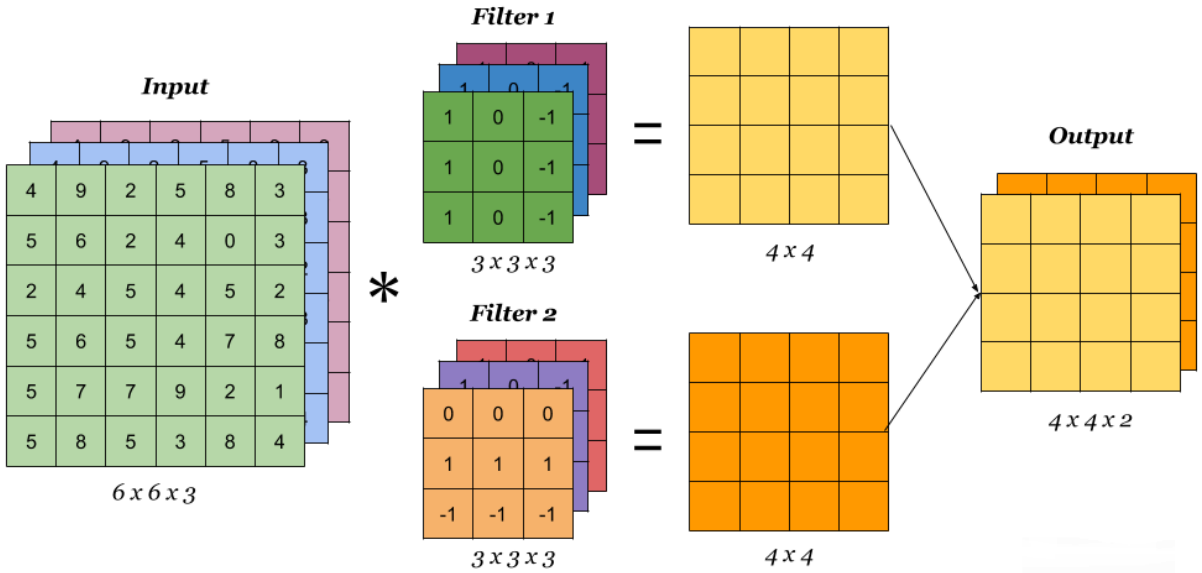


Figure II.11 Convolution d'une image RVB à l'aide de plusieurs filtres (noyaux)

La fonction d'activation est le dernier composant de la couche convolutionnelle pour augmenter la non-linéarité en sortie. Généralement, la fonction ReLU (Unité de Rectification Linéaire) ou la fonction Tanh (Tangente hyperbolique) est utilisée comme fonction d'activation dans une couche de convolution. La figure ci-dessous illustre une couche de convolution simple, où une image d'entrée 6X6X3 est convolue avec deux filtres de taille 4X4X3 pour obtenir une caractéristique convoluée (convolved feature) de taille 3X3X2, à laquelle la fonction d'activation est appliquée pour obtenir la sortie, également appelée carte de caractéristique (feature map).

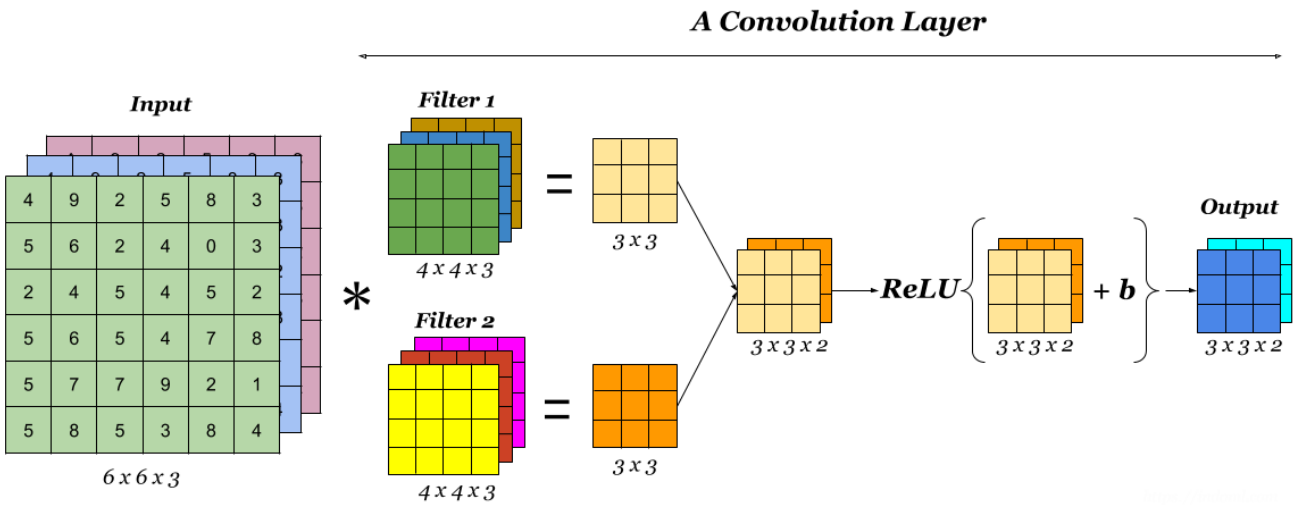


Figure II.12 Une couche de convolution

II.6.1.2 Couche de pooling

La couche de pooling (ou de regroupement) est utilisée pour réduire la taille de l'image d'entrée. Dans un réseau de neurones convolutionnels, une couche convolutionnelle est généralement suivie d'une couche de pooling. Cette dernière est généralement ajoutée pour accélérer le calcul et rendre plus robustes certaines des fonctionnalités détectées [25].

L'opération de pooling utilise un filtre et une foulée (stride). Dans l'exemple de la figure ci-dessous, le noyau 2X2 est utilisé pour regrouper l'image d'entrée de taille 4X4, avec une foulée de 2.

Il existe différents types de pooling. Le pooling par valeur maximale et le pooling par valeur moyenne sont les méthodes de pooling les plus couramment utilisées dans un réseau neuronal convolutionnel.

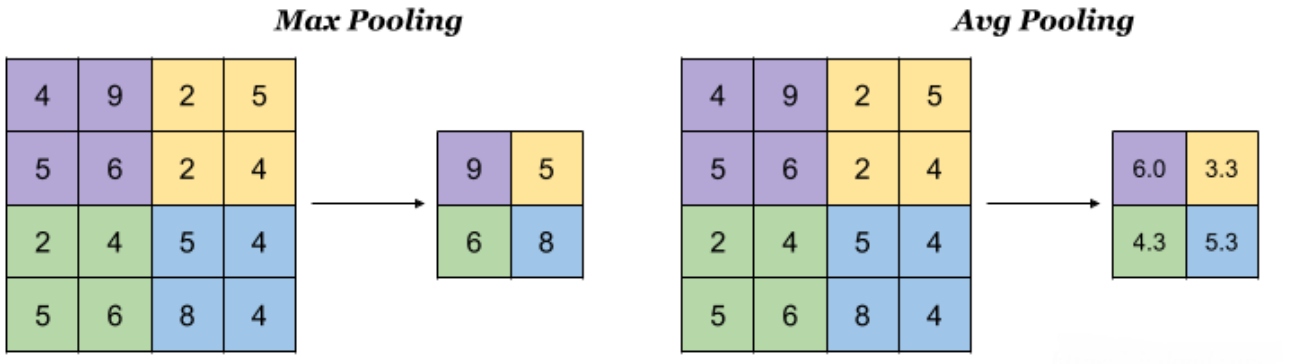


Figure II.13 Pooling max à gauche, pooling moyen à droite

- **Pooling Max :** Dans le pool max, à partir de chaque patch d'une carte de caractéristiques, la valeur maximale est sélectionnée pour créer une carte réduite.
- **Pooling moyen :** dans le pool moyen, à partir de chaque patch d'une carte de caractéristiques, la valeur moyenne est sélectionnée pour créer une carte réduite.

II.6.1.3 Couche entièrement connectée

Une couche entièrement connectée (FC) se trouve à l'extrémité d'un réseau neuronal convolutif. La carte de caractéristique produite par la couche précédente est aplatie en un vecteur. Ensuite, ce vecteur est appliqué à une couche entièrement connectée afin de capturer des relations complexes entre des caractéristiques de haut niveau. La sortie de cette couche est un vecteur caractéristique unidimensionnel [25].

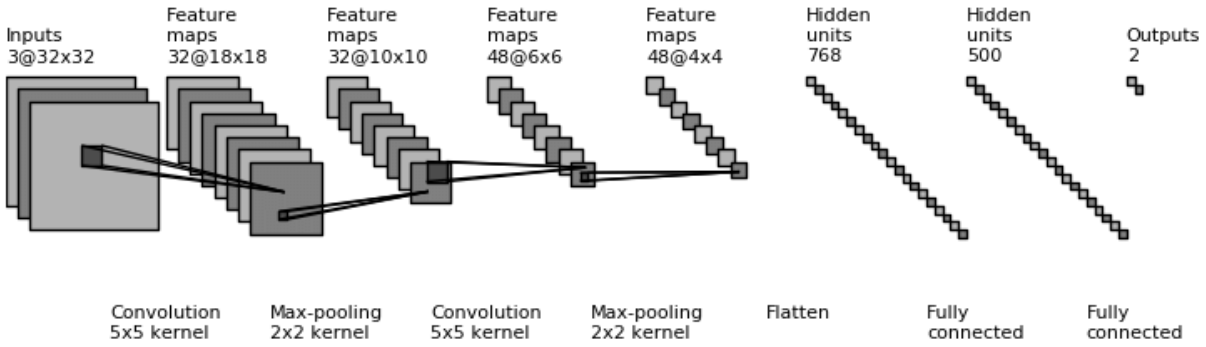


Figure II.14 Une architecture simple d'un CNN pour la classification d'images binaires

La figure ci-dessus représente une architecture simple d'un réseau neuronal convolutionnel utilisé pour la classification d'images binaires. Ce réseau classe une entrée entre deux classes différentes. Ce réseau prend une image RVB de taille 32x32x3 et la sortie est de taille deux (ce qui est égal au nombre de classes pour la classification). La première couche de ce réseau est une couche de convolution avec un filtre 5x5x3, la deuxième couche est une couche de pooling max avec une taille de filtre de 2x2, la troisième couche est une couche de convolution avec un filtre 5x5x3, la quatrième couche est une couche de regroupement max avec une taille de filtre de 2x2, la sortie est aplatie en un vecteur et envoyée aux deux dernières couches qui sont toutes deux des couches entièrement connectées.

II.6.2 Paramètres du filtre

Les couches de convolution et de pooling contiennent des filtres (ou noyaux) pour lesquels il est important de savoir comment ajuster leurs paramètres.

II.6.2.1 Dimensions du filtre

Un filtre de taille F×F appliqué à une entrée contenant C canaux est un volume de taille F×F×C qui effectue des convolutions sur une entrée de taille I×I×C et qui produit un feature map de sortie (carte de caractéristiques) de taille O×O×1.

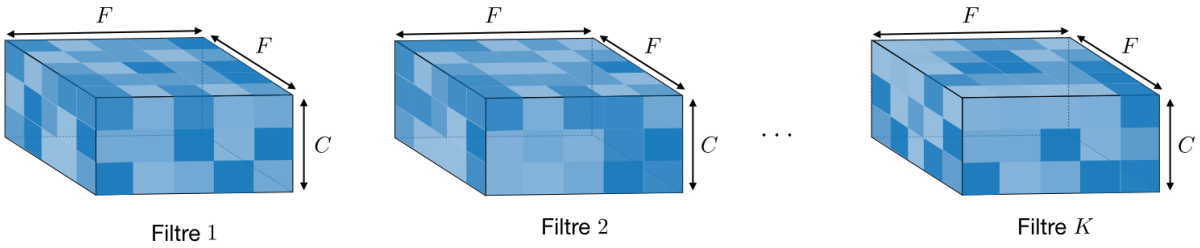


Figure II.15 Dimensions des filtres

Appliquer K filtres de taille F×F engendre un feature map de sortie de taille O×O×K.

II.6.2.2 Stride

Dans le contexte d'une opération de convolution ou de pooling, la stride S (la foulée) est un paramètre qui dénote le nombre de pixels par lesquels la fenêtre se déplace après chaque opération.

II.6.2.3 Padding

Parfois, il peut être intéressant de conserver une certaine dimension dans les tailles des images en sortie des convolutions. Le padding P consiste simplement à ajouter des zéros 0 tout autour d'une matrice (image) pour en augmenter la taille.

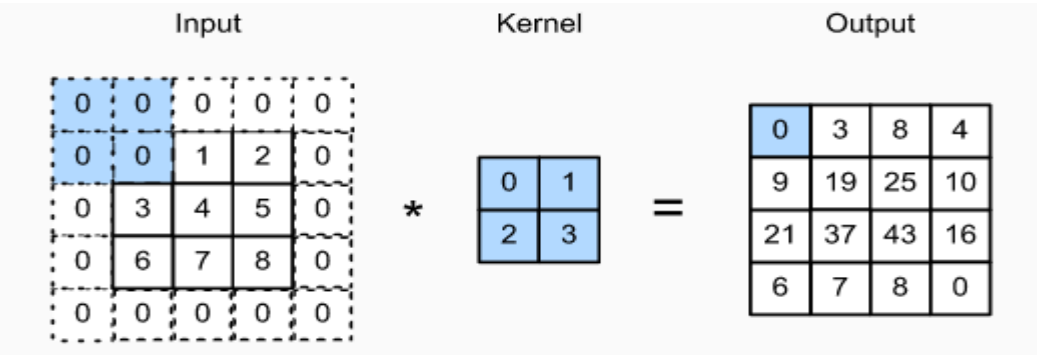


Figure II.16 Convolution entre une Image (3x3) et un filtre (2x2), avec Stride = 1 et Padding = 1

II.6.3 Réseaux de neurones convolutifs CNNs pour la classification d'images

Les CNNs sont à ce jour les modèles les plus performants pour classer des images. Certaines recherches sur la classification des images par CNNs ont atteint des performances rivalisant avec les experts humains. Voici quelques dates importantes dans l'évolution des CNNs :

- En 1980, Kunihiko Fukushima a introduit le néocognitron [26] qui est un réseau neuronal multicouche capable de reconnaître hiérarchiquement un modèle visuel à travers l'apprentissage. Ce réseau est considéré comme l'inspiration théorique des CNNs.
- En 1990 LeCun et al. a présenté le modèle pratique des CNNs [27] [28] et a développé LeNet-5 [29] pour la reconnaissance de caractères manuscrits. Ce travail a exploité la base de données MNIST contenant un grand ensemble de chiffres manuscrits.
- En 1997 S. Lawrence et al. [30] explorent l'idée d'utiliser un ConvNet pour la reconnaissance faciale afin d'extraire des fonctionnalités successivement plus grandes dans un ensemble hiérarchique de couches.
- En 2011 Chuan Zhou et al. [31] ont appliqué les CNNs à l'imagerie médicale en classant les régions d'intérêt sur les mammographies en masse ou en tissu normal.
- En 2012, un grand réseau de neurones convolutifs profonds, appelé AlexNet [32], conçu par Krizhevsky et al. a montré d'excellentes performances sur le défi de reconnaissance visuelle à grande échelle d'ImageNet (ILSVRC) [33]. Le succès d'AlexNet est devenu l'inspiration de différents modèles CNNs tels que ZFNet [34], VGGNet [35], GoogleNet [36], ResNet [37], DenseNet [38], CapsNet [39], SENet [40] etc. dans les années suivantes.

II.6.4 Architectures des CNNs

Il existe différents types d'architectures de CNNs, dans ce qui suit on présentera les plus courantes d'entre elles.

II.6.4.1 LeNet-5

LeNet-5 est l'une des architectures les plus simples. Il a 2 couches convolutionnelles et 3 couches entièrement connectées [29] (d'où « 5 », il est très courant que les noms des réseaux de neurones convolutifs dérivent du nombre de couches convolutionnelles et entièrement connectées dont ils disposent). La couche de pooling average (mise en commun moyenne) telle que nous la connaissons maintenant s'appelait une couche de sous-échantillonnage et avait des

poids entraînables (ce qui n'est pas le cas actuellement dans la conception des CNNs). LeNet-5 est devenu le « modèle » standard : empilement de couches de convolutions et de pooling, et fin du réseau avec une ou plusieurs couches entièrement connectées. Son architecture a environ 60 000 paramètres.

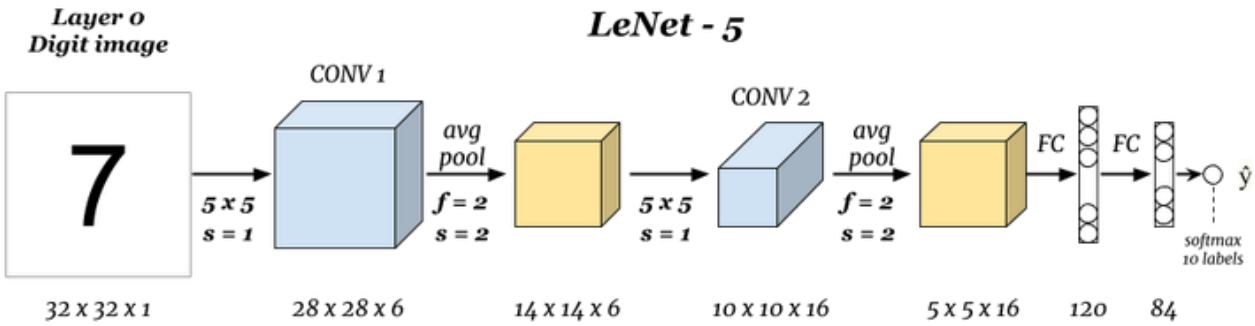


Figure II.17 Architecture LeNet-5

II.6.4.2 AlexNet

AlexNet, le vainqueur du défi ImageNet ILSVRC 2012, a été conçu par Alex Krizhevsky, Ilya Sutskever et Geoffery E. Hinton. Il a pu réduire le taux d'erreur du top 5 à 15,3% par rapport au taux d'erreur des finalistes de cette compétition qui a atteint un taux d'erreur de 26,2%.

L'architecture du réseau est similaire à l'architecture de LeNet, mais avec un plus grand nombre de filtres. Avec 60 Millions de paramètres, AlexNet a 8 couches : 5 convolutionnelles et 3 entièrement connectées, entre les deux, nous trouvons également des couches de Pooling et des unités linéaires rectifiées (ReLU) implémentées pour la première fois en tant que fonctions d'activation [32].

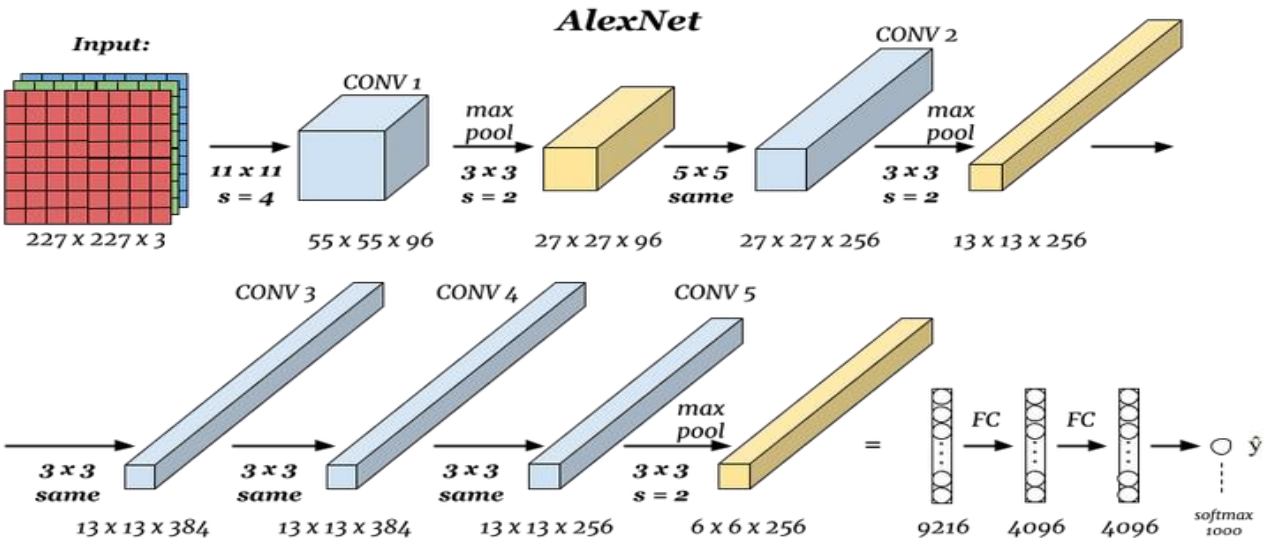


Figure II.18 Architecture AlexNet

II.6.4.3 VGGNet-16

Cette architecture conçue par Simonyan et Zisserman chercheurs du Visual Graphics Group à Oxford (d'où le nom VGG) a été le deuxième du défi ILSVRC 2014, Il a pu atteindre un taux d'erreur de 7,3% dans le top 5.

Le VGG-16 a 13 couches convolutionnelles, 3 couches entièrement connectées et utilise la fonction d'activation ReLU [35]. Ce réseau empile plus de couches sur AlexNet et utilise des filtres de plus petite taille (2 × 2 et 3 × 3). Il se compose de 138 millions de paramètres et occupe environ 500 Mo d'espace de stockage. Une variante plus profonde, VGG-19 a été également conçue.

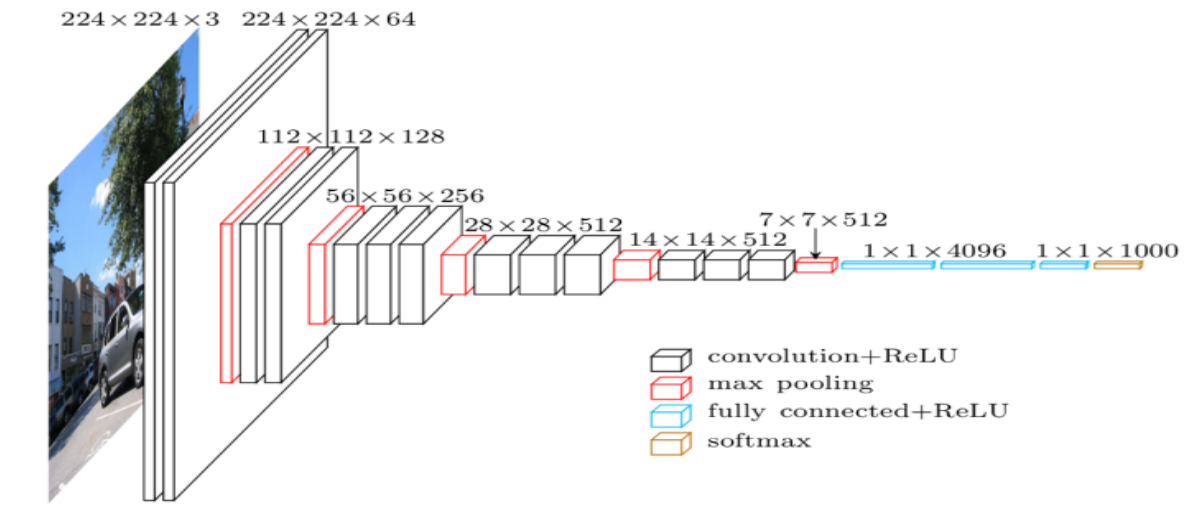


Figure II.19 Architecture VGGNet-16

II.6.4.4 GoogleNet/Inception

Le GoogleNet (ou Inception Network) a été le gagnant du défi ILSVRC 2014, Il a pu réaliser un taux d'erreur de 6,7% dans le top 5, ce qui était à peu près égale à la performance du niveau humain. Le modèle a été développé par Google, il a 22 couches avec 5 Millions de paramètres et comprend une implémentation plus intelligente de l'architecture LeNet d'origine [36]. Ceci est basé sur l'idée du module Inception.

L'idée de base derrière les modules Inception est qu'au lieu d'implémenter des couches convolutionnelles de divers hyperparamètres dans différentes couches, nous faisons toute la convolution ensemble pour produire un résultat contenant des matrices de toutes les opérations de filtrage ensemble. Dans la figure ci-dessous, un module Inception simple avec différentes couches convolutionnelles implémentées ensemble :

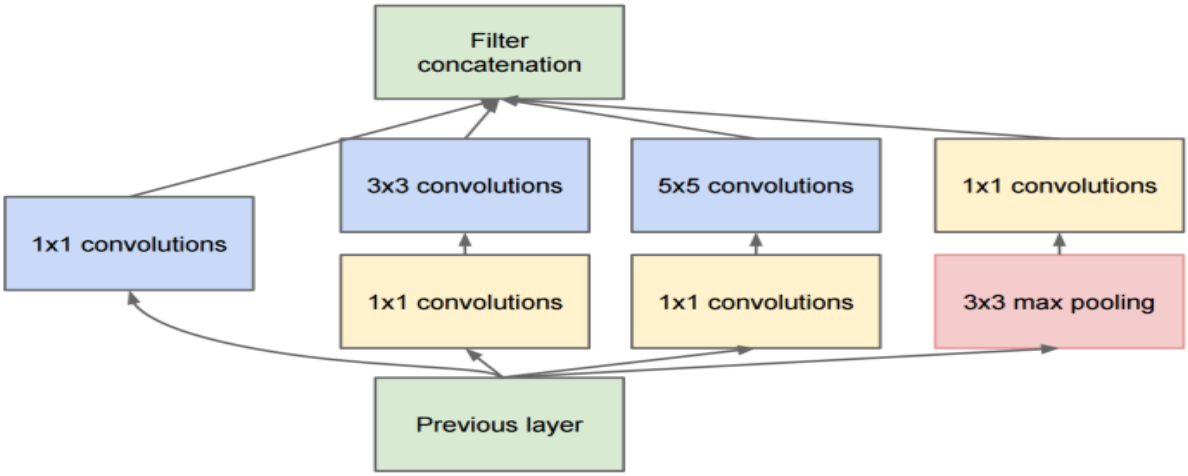


Figure II.20 Module Inception

L'architecture GoogleNet utilise plusieurs de ces modules Inception, comme le montre la figure ci-dessous :

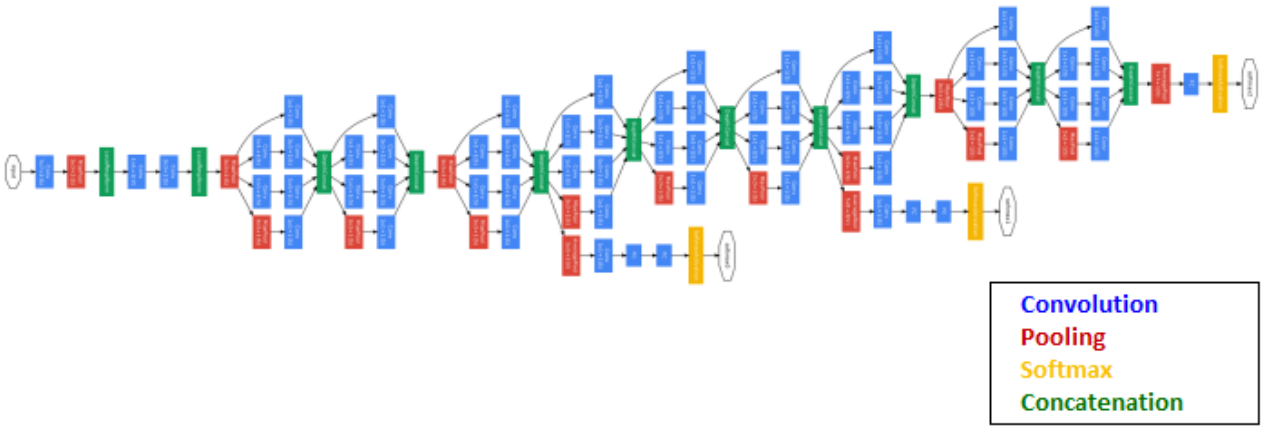


Figure II.21 Architecture GoogleNet

II.6.4.5 ResNet

Probablement après AlexNet, le développement le plus révolutionnaire dans le domaine du développement de l'architecture CNN s'est produit avec ResNet ou Residual Networks. L'idée qui a été insufflée dans cette architecture était la « connexion de raccourci d'identité » qui implique de transférer les résultats de quelques couches vers certaines couches plus profondes sautant ainsi certaines couches entre les deux ce qui aide le réseau à s'entraîner efficacement sur des milliers de couches, sans dégrader les performances à long terme [37].

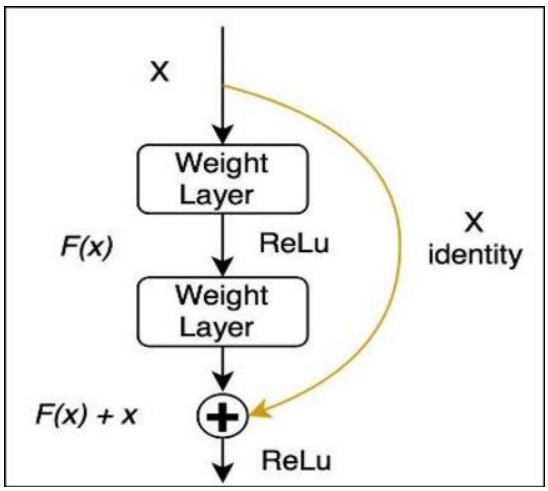


Figure II.22 La connexion par saut dans les modèles ResNet

Un ResNet-1001 profond de 1001 couches a atteint un taux d'erreur de 3,57% dans le top 5 du défi ILSVRC 2015 (ce qui bat en fait les performances au niveau humain sur l'ensemble de données), lui permettant ainsi de remporter tous les prix de ce challenge dans les domaines de la classification, de la détection et de la localisation.

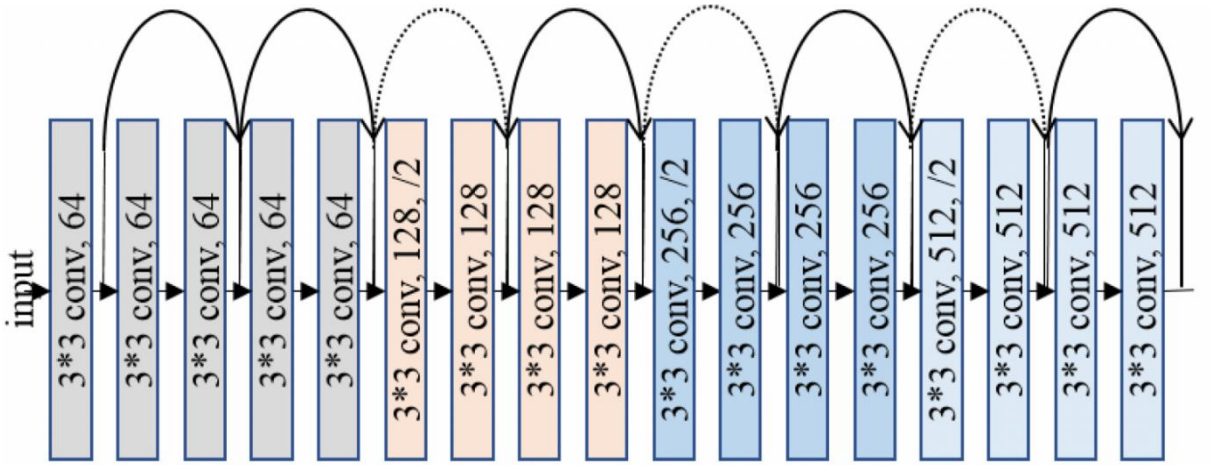


Figure II.23 Une architecture ResNet profonde à 18 couches

II.6.5 Métriques d'évaluation

Commençons par comprendre la terminologie de base utilisée dans les problèmes de classification avant de définir les différentes métriques d'évaluation.

Soit un problème de classification binaire. On représente les deux classes par : {X} et {Non X}. (Exemple : {X} : Hémorragique, {Non X} : Non Hémorragique). On appellera alors :

- True Positif (TP) : un échantillon de {X} prédit comme appartenant à {X}.
- True Negatif (TN) : un échantillon de {Non X} prédit de {Non X}.
- False Positif (FP) : un échantillon de {Non X} prédit de {X}.
- False Negatif (FN) : un échantillon de {X} prédit de {Non X}.

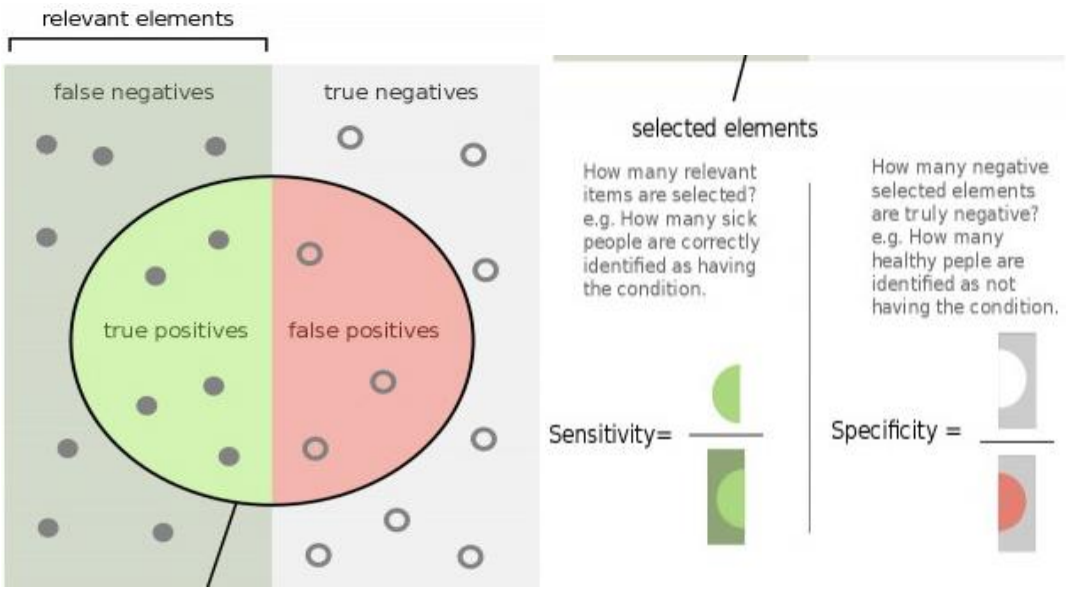


Figure II.24 Terminologie de base dans les problèmes de classification

II.6.5.1 Rappel ou sensibilité ou TPR (True Positive Rate)

Nombre d'éléments correctement identifiés comme positifs sur le total des vrais positifs.

$$TPR = TP / (TP + FN) \qquad (2.4)$$

II.6.5.2 Spécificité ou TNR (True Negative Rate)

Nombre d'éléments correctement identifiés comme négatifs sur le total des négatifs.

$$TNR = TN / (TN + FP) \qquad (2.5)$$

II.6.5.3 Précision

Nombre d'éléments correctement identifiés comme positifs sur le nombre total d'éléments identifiés comme positifs.

$$Précision = TP / (TP + FP) \qquad (2.6)$$

II.6.5.4 Matrice de confusion

Les paramètres définis plus haut permettent de poser la matrice de confusion, comme montré dans la figure (Fig 25) ci-dessous :

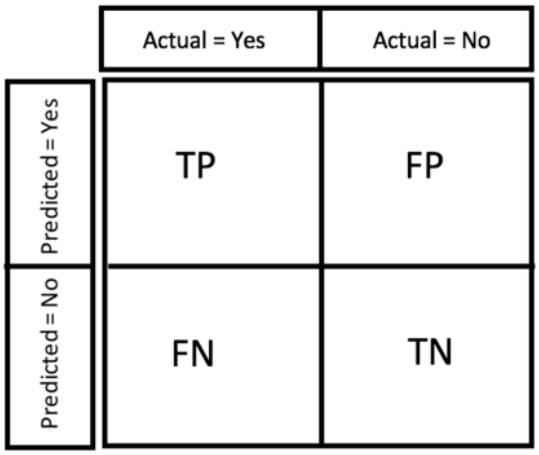


Figure II.25 Matrice de confusion

II.6.5.5 Accuracy

L'accuracy (ACC) est la métrique la plus utilisée pour la mesure de la performance des modèles, mais n'est pas suffisante pour réellement évaluer un modèle.

Il s'agit du rapport entre le nombre de prédictions correctes et le nombre total d'échantillons d'entrée.

$$Accuracy = \frac{Nombre\ de\ prédictions\ correctes}{Nombre\ total\ d'échantillons\ d'entrée} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{N} + \mathrm{P}} \qquad (2.7)$$

II.6.5.6 Perte logarithmique

La Perte logarithmique ou Log Loss, fonctionne en pénalisant les fausses classifications. Cela fonctionne bien pour la classification multi-classes. Lorsqu'on travaille avec Log Loss, le classificateur doit attribuer une probabilité à chaque classe pour tous les échantillons. Supposons qu'il existe N échantillons appartenant à M classes, alors la perte logarithmique est calculée comme suit :

$$LogLoss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} * \log\left(p_{ij}\right) \qquad (2.8)$$

- y_{ij} indique si l'instance i appartient à la classe j ou pas.
- p_{ij} indique la probabilité que l'instance i appartient à la classe j.

Une valeur proche de 0 indique une bonne performance.

II.7 Conclusion

Dans ce chapitre, nous avons présenté le Machine Learning, ses domaines d'utilisation, ses approches et ses différents algorithmes. Nous nous sommes ensuite, plus intéressés au Deep Learning qui est la sous branche la plus en vogue du Machine Learning. Nous avons vu ce qu'est le Deep Learning, et comment il se différencie des algorithmes de Machine Learning traditionnelles.

Les travaux actuels tendent à utiliser les réseaux de neurones convolutifs CNNs (un des algorithmes les plus populaires dans le domaine du Deep Learning) pour concevoir un système de prédiction et de diagnostic médical. C'est pourquoi nous nous sommes basés sur ces derniers. Nous avons défini les CNNs avec leurs différentes couches, vu quelques étapes majeures de leur évolution et cité les différents exploits qui ont été accomplis avec. Enfin nous avons présenté les architectures les plus courantes des CNNs et quelques métriques de leur évaluation.

CHAPITRE III

Implémentation, résultats et discussions

CHAPITRE III: Implémentation, résultats et discussions

III.1 Introduction

Dans ce chapitre, nous allons présenter la problématique sur laquelle nous avons travaillé, les différentes ressources matérielles et logiciels que nous avons utilisé, les différentes expérimentations réalisées, et enfin on terminera par une discussion des résultats d'évaluation obtenus.

III.2 Problématique étudiée

Le présent travaille s'intéresse à la détection et la classification des hémorragies intracrâniennes. On a pour but de détecter l'HIC et de la classer dans l'un de ses cinq (05) sous types vus dans le chapitre un (01), en utilisant les techniques du Deep Learning et plus précisément les CNNs vus dans le chapitre deux (02).

Cette problématique a fait l'objet d'un défi international sur Kaggle. Ce dernier est une plateforme web organisant des compétitions en science des données (extraction de connaissance d'ensembles de données) dans laquelle les entreprises et les chercheurs publient des données et les statisticiens et les data miners (les explorateurs de données) s'affrontent pour produire les meilleurs modèles de prédiction et de description des données.

Le défi dans cette compétition était de développer un algorithme pour détecter l'hémorragie intracrânienne aiguë et ses sous-types à l'aide d'un ensemble de données d'images riches fourni par la Radiological Society of North America (RSNA) en collaboration avec des membres de l'American Society of Neuroradiology et MD.ai.

III.3 Ressources

III.3.1 Ordinateur

Pour mener ce travail, on utilise un ordinateur portable avec les caractéristiques suivantes :

- Processeur : Intel(R) Core (TM) i7-3630QM CPU @ 2.40GHZ
- RAM : 8.00 Go

Mais comme cité dans le chapitre précédent, le Deep Learning nécessite :

- Un vaste jeu de données labellisées.

- Une puissance de calcul considérable qu'on retrouve dans les machines dotées de GPU puissant. Quand ces machines sont associées au cloud computing, il devient alors possible de réduire la durée d'entraînement du réseau de plusieurs semaines à quelques heures.

De ce qui a précédé, on déduit qu'il n'est pas recommandé de faire nos expérimentations localement avec notre laptot personnel peu puissant. Pour remédier à cette situation, nous utiliserons les deux plates-formes cloud gratuites Google Colaboratory et Kaggle Notebooks qui fournissent un GPU totalement gratuit.

III.3.2 Google Colaboratory

Google Colab ou Colaboratory est un service cloud, offert par Google (gratuit), basé sur Jupyter Notebook et destiné à la formation et à la recherche dans le Machine Learning [41]. Cette plateforme permet de développer des applications de Deep Learning à l'aide de bibliothèques populaires telles que Keras, TensorFlow, PyTorch et Open CV directement dans le cloud. En outre, elle permet d'écrire et d'exécuter du code Python dans le navigateur et cela sans aucune configuration requise, un accès gratuit aux GPU et un partage facile.

Jupyter Notebook est une application Web Open Source permettant de créer et de partager des documents contenant du code (exécutable directement dans le document), des équations, des images et du texte. Avec cette application il est possible de faire du traitement de données, de la modélisation statistique, de la visualisation de données, du Machine Learning, du Deep Learning, etc. [41]

Les GPU disponibles dans Colab incluent souvent les Nvidia Tesla K80, T4, P4 et P100 (leur prix tourne autour de 2000-3000 dollars $ sur eBay**)** avec environ 12 GB de mémoire. Il n'y a aucun moyen de choisir le type de GPU auquel on peut nous connecter dans Colab à un moment donné. Colab offre aussi un CPU Intel Xeon Dual Core – 2.33 GHz avec environ 13.3GB de RAM et la session est limitée à 12 heures.

III.3.3 Kaggle Notebooks

Kaggle est assez similaire à Colab :

- C'est aussi un service cloud gratuit de Google.
- Il est aussi basé sur les Notebooks Jupyter.
- Exécute aussi du code Python et ouvre accès à des dizaines de bibliothèques pour

développer des algorithmes de Deep Learning.

- Ne requiert aucune configuration, et offre un GPU gratuit.

D'autre part, Kaggle offre d'autres spécifications :

- GPU Tesla P100 avec une RAM de 17.1 GB.
- Un CPU Intel Xeon Dual Core – 2.33 GHz avec 16.4 GB de RAM.
- Limitation de la session à 9 heures.
- Accès au GPU limité à 30 heures/semaine contrairement à Colab qui donne un

accès illimité au GPU.

- Propose de nombreux jeux de données qu'on peut importer facilement.

III.3.4 Langage de programmation

III.3.4.1 Python

Python est un langage de programmation open source, libre et multi-plateformes, à la fois puissant et facile à maîtriser [42], il dispose d'un système de type dynamique et d'une gestion automatique de la mémoire. Actuellement, il est le langage de programmation le plus utilisé au monde et il est utile dans presque tous les domaines liés de près ou de loin à l'informatique [43].

Python offre certaines des meilleures flexibilités et fonctionnalités aux développeurs qui augmentent non seulement leur productivité mais aussi la qualité du code. Il dispose aussi d'un grand nombre de bibliothèques et d'outils open source qui facilitent la charge de travail.

Ces points forts, permettent à Python de gagner la bataille en tant que langage préféré de Machine Learning et de Deep Learning et font de lui le choix idéal pour développer des modèles ML et DL.

III.3.5 Bibliothèques

Python fourni un large éventail de bibliothèques adaptées au Deep Learning, dans ce qui suit, une description des bibliothèques les plus importantes que nous avons utilisé :

III.3.5.1 TensorFlow

TensorFlow est incontestablement le poids lourd des bibliothèques de DL. Il est le plus actif sur GitHub [44]. Proposé par Google, TensorFlow, permet aux débutants comme aux professionnels de créer, de former et de gérer des réseaux de neurones profonds.

TensorFlow est assez efficace en termes de classification, perception, compréhension, découverte, prédiction et création de données.

III.3.5.2 Keras

Keras n'est pas une bibliothèque en tant que tel mais une API (Application Programming Interface) de haut niveau qui se place au-dessus de TensorFlow. Depuis 2017, Keras est directement intégré dans TensorFlow, il propose plusieurs des éléments constitutifs et des outils nécessaires à la création d'un réseau de neurones, tels que : les couches de convolutions, les fonctions d'activation et de coût, le Pooling, etc.

Keras a été développé par François Chollet, Chercheur chez Google et est disponible sous licence MIT [45].

III.3.5.3 Pandas

Pandas est une bibliothèque d'analyse de données Python et est principalement utilisé pour la manipulation et l'analyse de données. Il intervient avant que l'ensemble de données ne soit préparé pour la formation. Pandas est un outil parfait pour la fusion de données [45]. Il est conçu pour une manipulation, une lecture, une agrégation et une visualisation des données rapides et faciles.

Pandas explore généralement des données contenues dans un fichier CSV ou TSV ou une base de données SQL.

III.3.5.4 NumPy

L'un des packages les plus fondamentaux de Python, *NumPy* est un package de traitement de tableaux à usage général. Il se concentre sur la gestion de données multidimensionnelles étendues et les fonctions mathématiques complexes opérant sur les données [46].

NumPy facilite les opérations mathématiques sur les tableaux et leur vectorisation. Cela améliore considérablement les performances et accélère le temps d'exécution en conséquence.

III.3.5.5 Matplotlib

Matplotlib est un package de visualisation en Python pour les tracés 2D de tableaux. Il permet la visualisation de données multiplateforme basée sur les tableaux NumPy [46]. L'un de ses plus grands avantages est qu'il permet d'accéder visuellement à d'énormes quantités de données dans des visuels facilement interprétable.

Matplotlib facilite la création des étiquettes, des grilles, des légendes et de certaines autres entités de mise en forme.

III.3.6 Base d'images

La base de données d'images que nous avons utilisée est disponible sur la plateforme Kaggle, accessible directement sur le cloud à partir d'un Notebook Kaggle, ou en téléchargement sur https://www.kaggle.com/c/rsna-intracranial-hemorrhage-detection/ data?select=rsna-intracranial-hemorrhage-detection (Fichier volumineux de 181,30 Go).

Cette base a été fourni par la Radiological Society of North America (RSNA) en collaboration avec des membres de l'American Society of Neuroradiology et MD.ai. Elle contient plus de 25 000 examens CT cérébral étiquetés par plus de 60 bénévoles, fournissant ainsi exactement :

- 752803 images CT d'apprentissage.
- 121232 images CT de test.

Toutes les images fournies sont au format DICOM et de taille 512x512.

Concernant les données d'apprentissage, elles sont fournies dans un fichier CSV, sous la forme d'un ensemble d'images *Id* et de plusieurs étiquettes, une pour chacun des cinq sous-types d'hémorragie, plus une étiquette supplémentaire pour *any*, qui devrait toujours être vraie *si l'une* des étiquettes de sous-type est vraie.

Il existe également une colonne cible *Label*, indiquant si un type d'hémorragie existe ou non dans l'image indiquée.

Il y aura donc, **6** lignes par image *Id*. Ci-dessous un exemple des données d'apprentissage pour une image donnée.

```
Id, Label
1_epidural_hemorrhage, 0
1_intraparenchymal_hemorrhage, 0
1_intraventricular_hemorrhage, 0
1_subarachnoid_hemorrhage, 1
1_subdural_hemorrhage, 0
1_any, 1
```

III.4 Analyse exploratoire de données (Exploratory Data Analysis EDA)

L'analyse, l'exploration et la mise en forme des données est une étape fondamentale dans tout projet d'intelligence artificielle. En effet la pertinence des résultats obtenus dépend généralement plus de la qualité de mise en œuvre de ces étapes que de la méthode de modélisation utilisant ces données en entrée [47].

L'objectif de l'analyse exploratoire est d'obtenir une vision globale d'un jeu de données en recherchant des régularités, des relations entre variables, ou même des groupes homogènes. C'est grâce à cela que l'on peut définir les outils et méthodes de modélisation les plus adéquats pour répondre au problème posé [48].

III.4.1 Visualisation des données

III.4.1.1 Visualisation des premières images d'apprentissage

Les images médicales sont stockées dans un format spécial appelé fichiers DICOM (*.dcm). Ils contiennent une combinaison de métadonnées d'en-tête ainsi que des tableaux d'images brutes pour les données de pixels. En Python, une bibliothèque populaire pour accéder et manipuler les fichiers DICOM est le module pydicom. Pour charger les données on utilise la ligne de commande suivante : *pydicom.dcmread* (''chemin complet de l'image'').

Ci-dessous quelques exemples des images Dicom formant notre ensemble d'apprentissage.

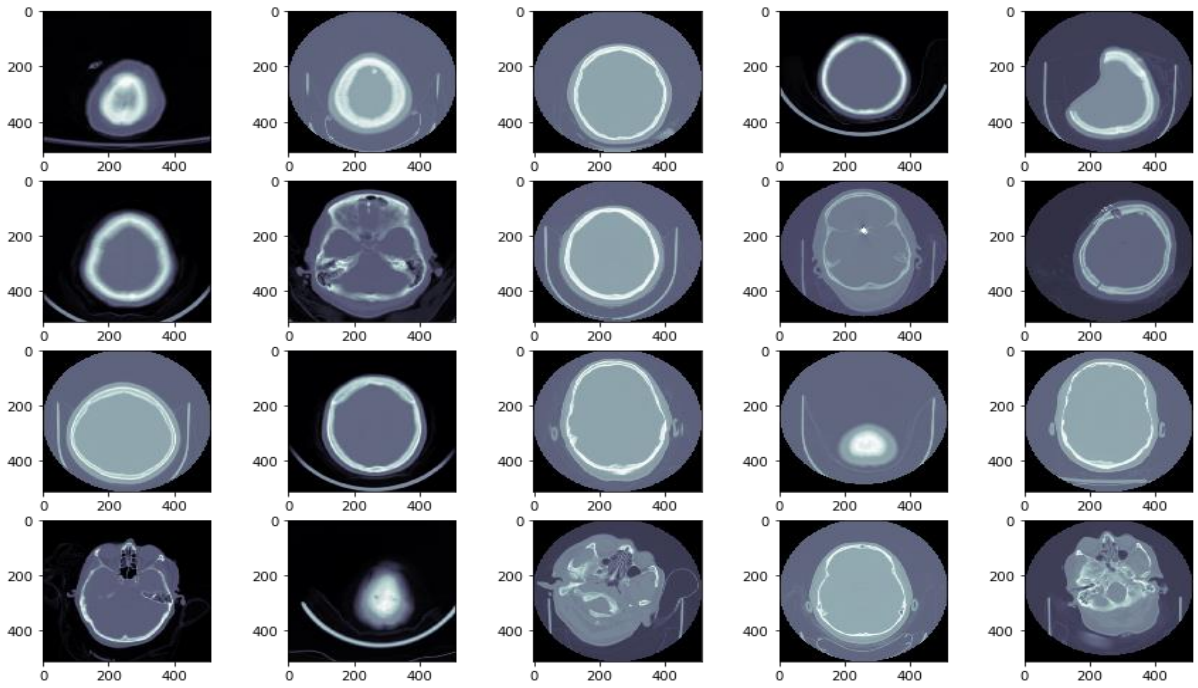


Figure III.1 Les 20 premières images d'apprentissage visualisées avec le filtre « bone »

III.4.1.2 Visualisation des différents types d'HIC

III.4.1.2.1 Hémorragie épidurale

Pour trouver les images contenant une hémorragie épidurale, deux conditions doivent être vérifiées.

1. Sub_type = epidural.
2. Label = 1

La syntaxe suivante est utilisée :

view_images(train[(train['Sub_type'] == 'epidural') & (train['Label'] == 1)

Ci-dessous les dix premiers cas d'hémorragies épidurales rencontrées en parcourant la base de données de formation.

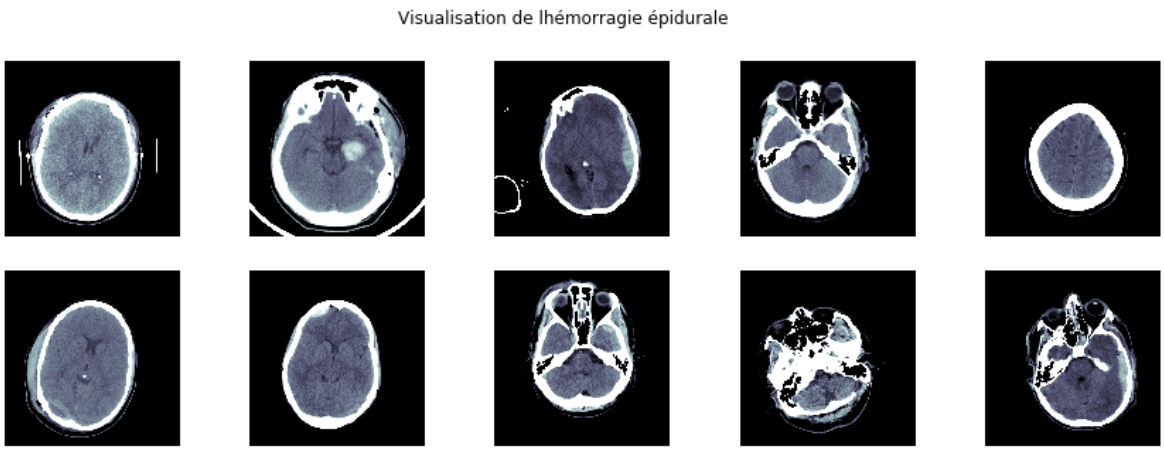


Figure III.2 Visualisation de 10 exemples de l'hémorragie épidurale

Le même raisonnement est utilisé pour visualiser les autres types de l'hémorragie intracrânienne présents dans notre base d'images de formation (Training files).

Nous pouvons ainsi visualiser :

III.4.1.2.2 Hémorragie intraparenchymateuse

Ci-dessous les dix premiers cas d'hémorragies intraparenchymateuses rencontrées en parcourant la base de données de formation.

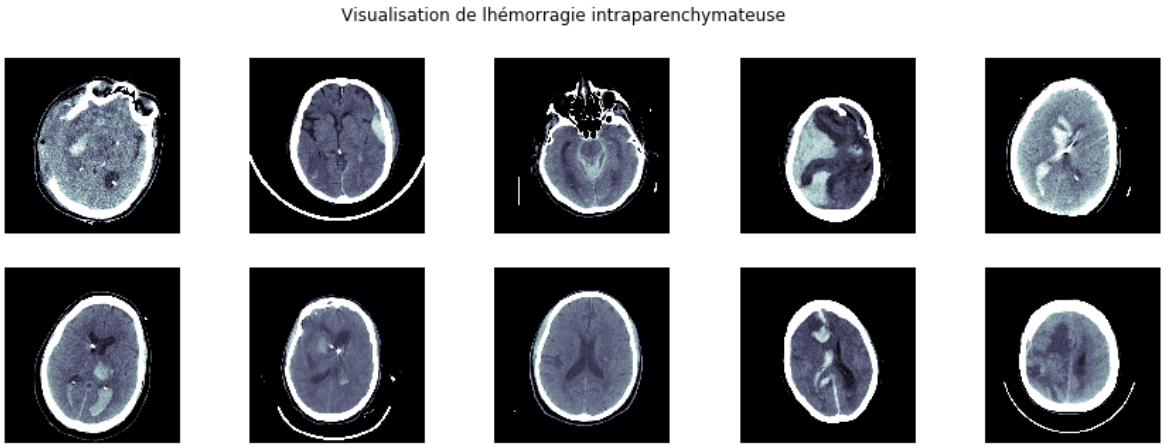


Figure III.3 Visualisation de 10 exemples de l'hémorragie intraparenchymateuse

III.4.1.2.3 Hémorragie intraventriculaire

Ci-dessous les dix premiers cas d'hémorragies intraventriculaires rencontrées en parcourant la base de données de formation.

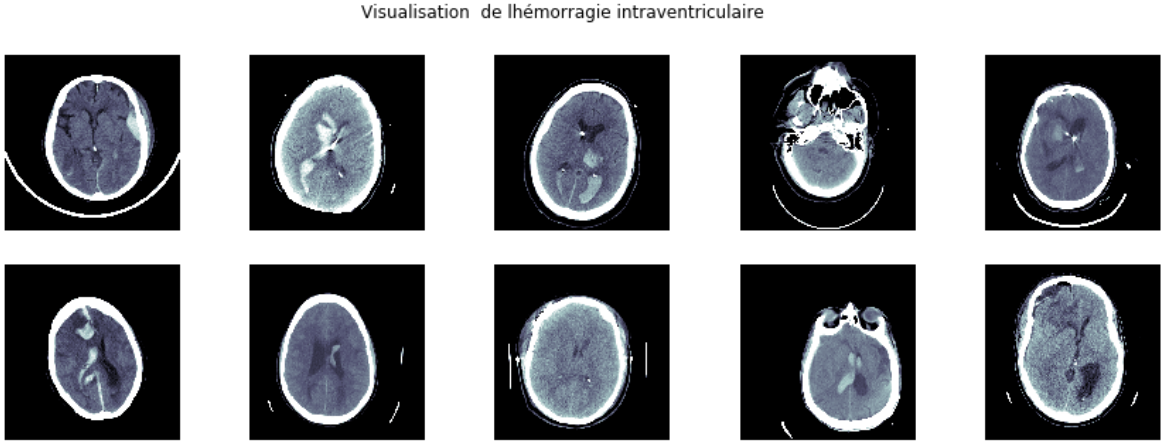


Figure III.4 Visualisation de 10 exemples de l'hémorragie intraventriculaire

III.4.1.2.4 Hémorragie subarachnoïdienne

Ci-dessous les dix premiers cas d'hémorragies subarachnoïdiennes rencontrées en parcourant la base de données de formation.

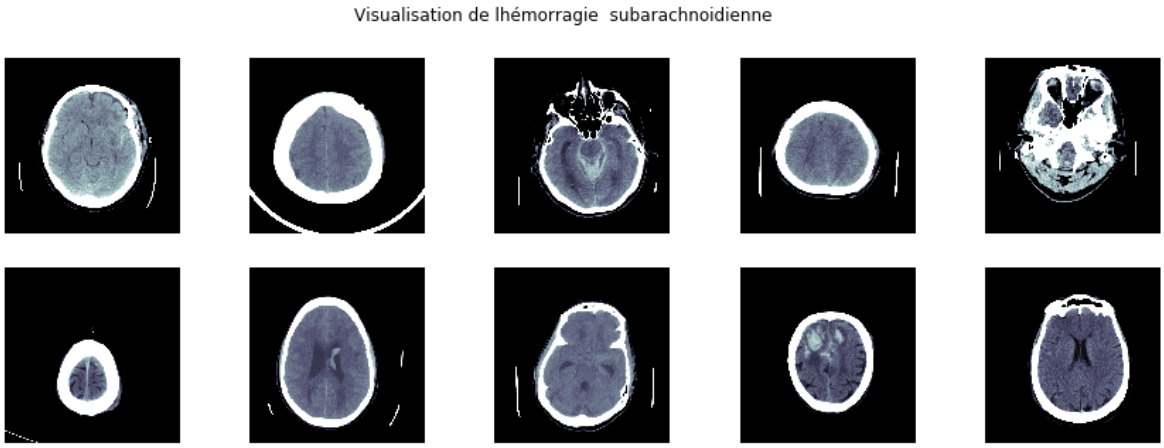


Figure III.5 Visualisation de 10 exemples de l'hémorragie subarachnoïdienne

III.4.1.2.5 Hémorragie sous-durale

Ci-dessous les dix premiers cas d'hémorragies sous-durales rencontrées en parcourant la base de données de formation.

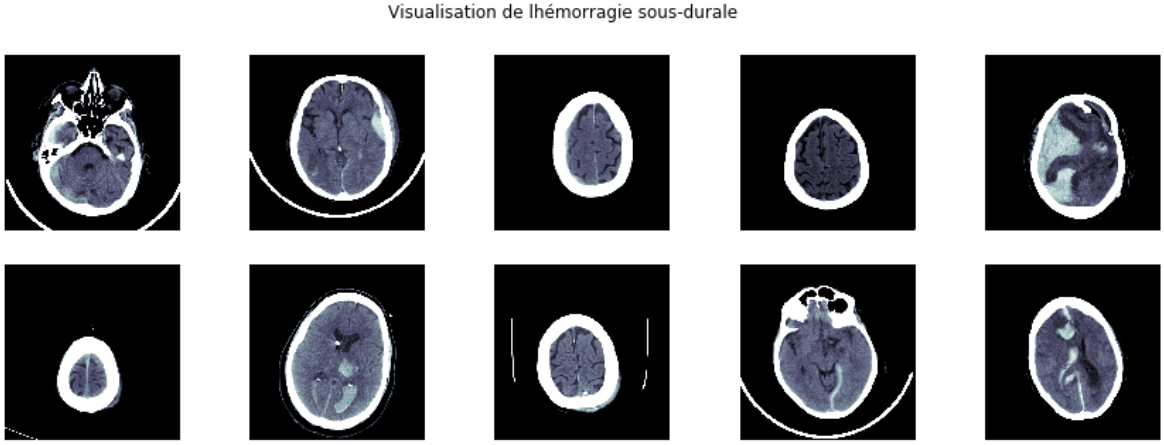


Figure III.6 Visualisation de 10 exemples de l'hémorragie sous-durale

III.4.2 Analyse des données

III.4.2.1 Distribution des données cibles

III.4.2.1.1 Distribution des données cibles par label

Les images cibles sont labelisées par :

- « 0 » (Négatif) : absence d'une HIC
- « 1 » (Positif) : présence d'une HIC

On utilise la commande *traindf.groupby('id').label.sum()* pour compter le nombre de cas positifs et négatifs. ''*traindf*'' est un tableau de format CSV contenant les données d'apprentissage.

Pour visualiser les informations extraites du ''traindf'' on utilise la libraire *Seaborn*. *Seaborn* est une librairie qui vient s'ajouter à Matplotlib comme outil de visualisation en Python et qui offre les avantages suivants :

- Génère des graphiques d'une grande qualité esthétique.
- Créer facilement des analyses statistiques sophistiquées.
- Interagi avec les Dataframes de Panda.

Ci-dessous un diagramme illustrant le nombre de cas négatifs et positifs présents dans notre ensemble de données.

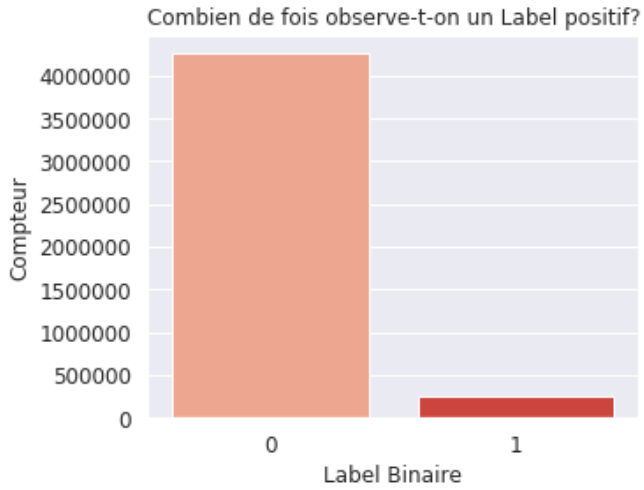


Figure III.7 Nombre de cas négatifs '0' et positifs '1'

En utilisant la commande *train.Label.value_counts()* on obtient avec précision le nombre exact de chacun des deux cas.

Label	Nombre
0 (Négatif, Absence d'hémorragie)	4260600
1 (Positif, Présence d'hémorragie)	256242

Tableau III.1 Nombre de cas 'Négatifs' et 'Positifs'

De ce qui a précédé nous remarquons qu'il y a beaucoup plus d'occurrences nulles que de valeurs cibles positives.

III.4.2.1.2 Distribution des données cibles par sous-types

Maintenant nous nous intéressons aux proportions d'occurrences positives dans chaque sous type de l'HIC.

En utilisant la commande ci-dessous de la librairie Seaborn, nous obtenons le diagramme à barres qui montre la proportion des cas positifs à l'intérieur de chaque sous type.

Sns.barplot (x=subtype_counts.index, y=subtype_counts.values, palette="set2")

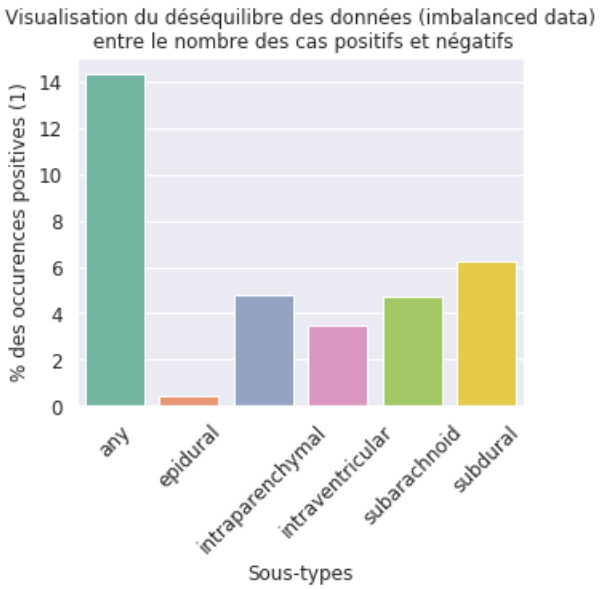


Figure III.8 Proportion des cas positifs par sous-types

En entrant dans les détails de chaque sous-type, nous pouvons voir que nous devons faire face à un déséquilibre de haut niveau.

L'hémorragie épidurale est le pire des cas. Pour ce type, nous n'avons que quelques (<1%) d'occurrences positives. Il sera difficile de former un modèle suffisamment robuste et qui n'a pas tendance à surajuster.

III.4.2.2 Nombre d'échantillons

Vérifions le nombre d'échantillons présent dans notre tableau de données d'apprentissage. Pour cela on utilise la commande *traindf.id.nunique()*. Ci-dessous le résultat obtenu après exécution :

```
Out[15] 752803
```

Cela signifie que nous disposons de 752803 échantillons à identifiant unique.

Vérifions maintenant si cela correspond au nombre d'images de formation que nous avons.

Pour ce faire, il suffit de calculer la longueur du fichier contenant les images de formation en utilisant la commande *len(train_files)*, nous obtenons :

```
[19]: train_size = len(train_files)
      train_size

Out[19] 752803
```

Nous remarquons bien que le nombre de données de formation correspond au nombre d'images de formation.

De la même façon nous vérifions la correspondance entre le nombre de données de test avec le nombre d'images de test que nous avons. Nous obtenons le même résultat, qui est de 121232 échantillons, comme nous pouvons le voir ci-dessous.

```
Out[20] 121232
```

Nous pouvons aussi comparer la quantité de données de formation avec celle du test, comme montrer ci-dessous :

```
train_size/test_size

Out[21] 6.209606374554573
```

Nous pouvons voir que nous disposons presque 6,21 fois plus d'images d'entraînement que d'images de test.

III.4.2.3 Dimensions des images

Vérifions la dimension des images d'entrainement et de test que nous avons. Pour ce faire nous procédons de la manière suivante :

Sachant que la dimension de chaque image est une métadonnée contenue dans le fichier de format DICOM, alors pour récupérer cette information il suffit de :

- Lire le fichier DICOM : dataset = *pydicom.dcmread(''chemin du fichier'')*

- Récupérer le nombre de ligne de l'image : *nbre_ligne = dataset.Rows*

- Récupérer le nombre de colonne de l'image : *nbre_colonne = dataset.Columns*

En utilisant une boucle nous pouvons récupérer les dimensions de toutes les images et dessiner les diagrammes suivants :

a. Images d'entrainement :

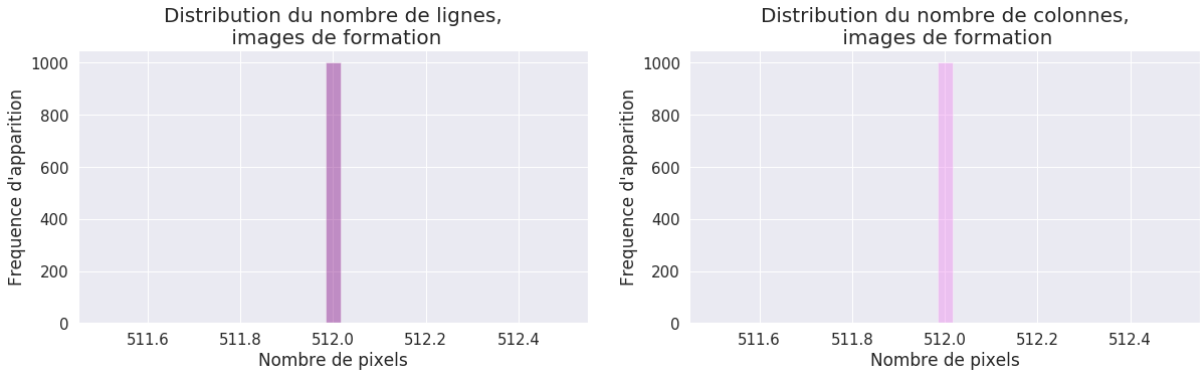


Figure III.9 Dimensions des images de formation

b. Images de test :

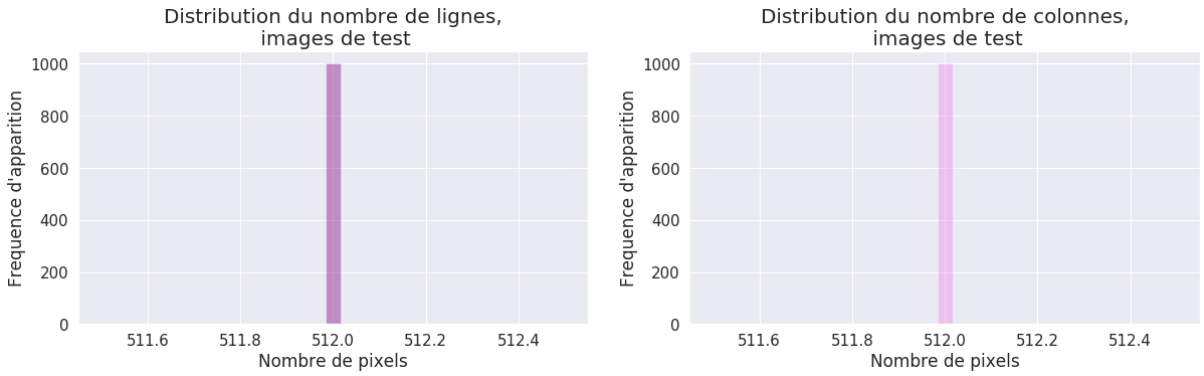


Figure III.10 Dimensions des images de test

Comme on peut le voir sur les différents diagrammes ci-dessus, les images sont toutes de dimensions 512x512.

III.5 Prétraitement

III.5.1 Prétraitement des fichiers DICOM

Voyons voir comment sont étiquetés les images dans la base de données d'entrainement : *train_files[0:10] :*

```
Out[22] ['ID_d15c5a2fb.dcm',
         'ID_2475533c5.dcm',
         'ID_164904d44.dcm',
         'ID_009d566e0.dcm',
         'ID_f304adf54.dcm',
         'ID_6f383ae4c.dcm',
         'ID_d5e80561b.dcm',
         'ID_7631b623b.dcm',
         'ID_971532946.dcm',
         'ID_5ed46866e.dcm']
```

Figure III.11 Etiquetage des images dans la base de données d'entrainement

Maintenant voyons le tableau des données d'entrainement qu'on a :

traindf = pd.read_csv("Chemin du tableau de données")

traindf.head() :

	ID	Label
0	ID_12cadc6af_epidural	0
1	ID_12cadc6af_intraparenchymal	0
2	ID_12cadc6af_intraventricular	0
3	ID_12cadc6af_subarachnoid	0
4	ID_12cadc6af_subdural	0

Figure III.12 Exemple des données d'entrainement

On remarque dans la figure ci-dessus que la colonne ID contient l'étiquette de l'image (ID_alphanum) suivi du sous-type de l'HIC. Grace à la fonction *str.rsplit* de la librairie Pandas nous pouvons fractionner la colonne ID en deux colonnes et ainsi créer une colonne *id* (Contenant les étiquettes des images) et une autre *sous-type* (contenant le sous-type de l'HIC). Cette démarche nous permettra de charger facilement les images à partir de la colonne *id* du tableau de données d'entrainement.

III.5.2 Que donne un fichier DICOM ?

Outre les images numériques issues des examens médicaux, les fichiers DICOM véhiculent aussi nombre d'informations textuelles concernant le patient (état civil, âge, poids, etc.), l'examen réalisé (région explorée, technique d'imagerie utilisée, etc.), la date d'acquisition [49], les détails techniques (orientation de l'image, dimensions de l'image, centre de la fenêtre, largeur de la fenêtre, etc.).

Voyons ça de plus près à l'aide de ces lignes de codes :

dataset= pydicom.dcmread(‘‘Chemin de l’image’’)

print(dataset)

```
(0008, 0018) SOP Instance UID                   UI: ID_c5c23af94
(0008, 0060) Modality                           CS: 'CT'
(0010, 0020) Patient ID                         LO: 'ID_9630cc49'
(0020, 000d) Study Instance UID                 UI: ID_c5409f3ace
(0020, 000e) Series Instance UID                UI: ID_5db30227b2
(0020, 0010) Study ID                           SH: ''
(0020, 0032) Image Position (Patient)           DS: ['-125.000', '-118.882', '33.678']
(0020, 0037) Image Orientation (Patient)        DS: ['1.000000', '0.000000', '0.000000
(0028, 0002) Samples per Pixel                  US: 1
(0028, 0004) Photometric Interpretation         CS: 'MONOCHROME2'
(0028, 0010) Rows                               US: 512
(0028, 0011) Columns                            US: 512
(0028, 0030) Pixel Spacing                      DS: ['0.488281', '0.488281']
(0028, 0100) Bits Allocated                     US: 16
(0028, 0101) Bits Stored                        US: 16
(0028, 0102) High Bit                           US: 15
(0028, 0103) Pixel Representation               US: 1
(0028, 1050) Window Center                      DS: "40"
(0028, 1051) Window Width                       DS: "150"
(0028, 1052) Rescale Intercept                  DS: "-1024"
(0028, 1053) Rescale Slope                      DS: "1"
(7fe0, 0010) Pixel Data                         OW: Array of 524288 elements
```

Figure III.13 Informations textuelles contenues dans un DICOM

III.5.3 Fenêtrage

Pour comprendre l’intérêt du fenêtrage nous devons savoir que :

- Les unités Hounsfield sont une mesure pour décrire la radiodensité.
- Différents tissus ont des HUs différents.
- Notre œil ne peut détecter qu'un changement d'environ 6% des niveaux de gris (16 nuances de gris).
- Étant donné 2000 UHs d'une image (-1000 à 1000), cela signifie qu'une échelle de gris couvre 8 UHs.
- Par conséquent, il peut se produire un changement de 120 HUs, sans que notre œil soit capable de détecter un changement d'intensité dans l'image.
- L'exemple d'une hémorragie dans le cerveau montre des HUs pertinentes de l'ordre de 8 à 70. Nous ne pourrons pas voir de changements importants dans l'intensité pour détecter l'hémorragie.
- C'est la raison pour laquelle nous devons concentrer les 256 nuances de gris dans une petite plage / fenêtre d'unités HU. (LA FENÊTRE)
- Le niveau signifie où cette fenêtre est centrée.

Voyons maintenant comment les différents ensembles de données Dicom diffèrent dans la distribution des valeurs de pixels :

a. Distribution des pixels dans les images brutes :

On récupère les valeurs des pixels des images sous forme d'un tableau grâce à la commande *flatten()* qui va transformer la matrice image de dimension 512 * 512 en un vecteur de dimension 1* (512*512).

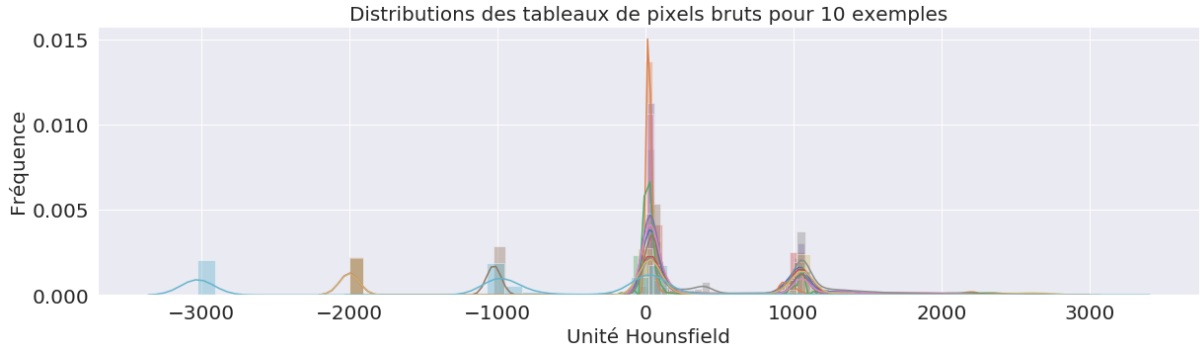


Figure III.14 Distribution des pixels dans des images brutes

On remarque que le mode (ou la valeur dominante) est situé à zéro (0) pour tous les exemples de tableaux de valeurs de pixels bruts. Cela correspond probablement à l'air. Par conséquent, nos valeurs de pixels brutes ne sont pas données en unités HU.

b. Distribution des pixels dans les images redimensionnées :

Nous pouvons transformer l'image en unités HU en mettant à l'échelle à l'aide de deux paramètres (qu'on peut trouver dans les informations textuelles des images DICOM) qui sont la pente et l'interception (RescaleSlope and RescaleIntercept) :

*image_redimensionnée = image * dataset.RescaleSlope + dataset.RescaleIntercept*

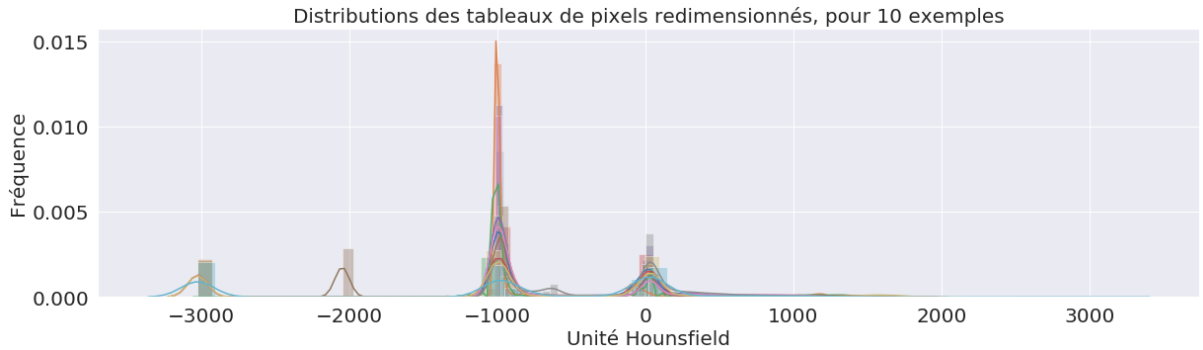


Figure III.15 Distribution des pixels dans des images redimensionnées

Dans la figure ci-dessus nous pouvons voir que le mode est maintenant situé à -1000 qui est l'HU de l'air.

A présent, essayons de comprendre les -2000 et -3000 HUs dans la figure précédente.

Pour cela on propose de fixer les valeurs inférieures à – 1000 à la valeur de l'HU de l'air (-1000) et voir l'impact en image :

image_redimensionnée [image_redimensionnée < -1000] = -1000

Affichons les images redimensionnées et les images redimensionnées et filtrées ensemble pour pouvoir comparer :

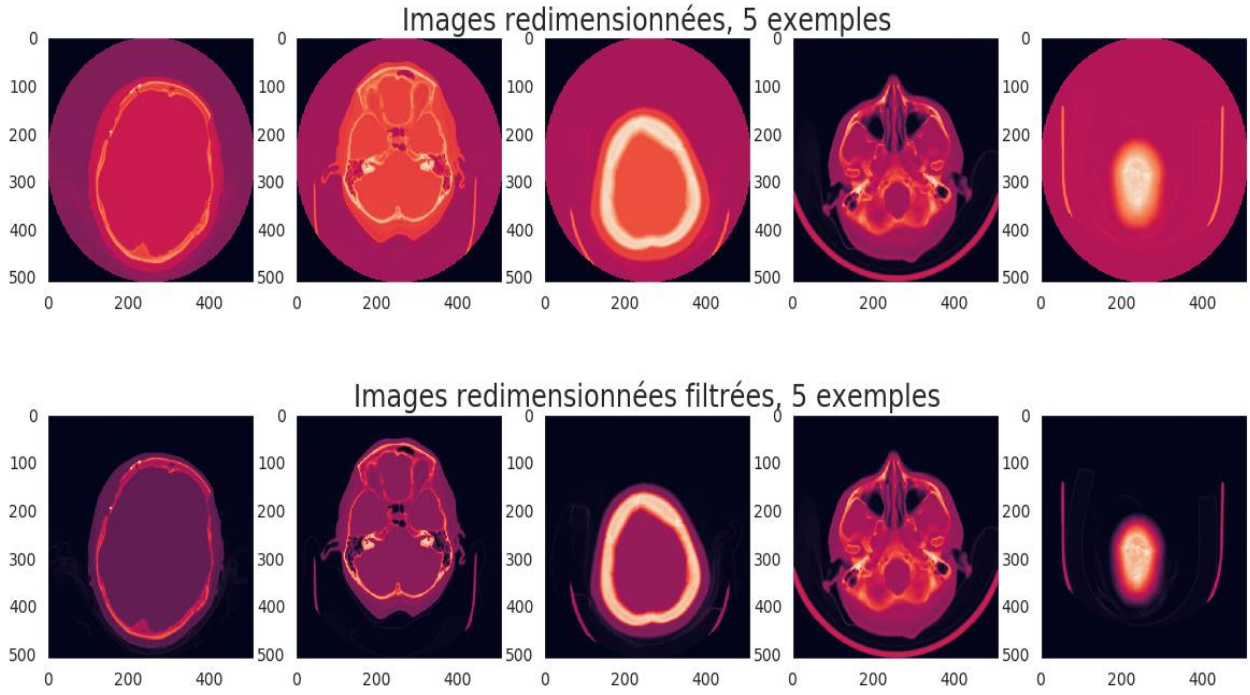


Figure III.16 Images redimensionnées et images redimensionnées et filtrées

Nous pouvons voir que les deux cas (-2000 et -3000) correspondent à la région extérieure des scanners CT cylindriques qu'on peut considérer comme artefacts. Par conséquent, nous pouvons fixer ces valeurs à -1000 sans souci et ainsi avoir un background noir correspondant à l'air.

III.5.4 Les fenêtres des médecins

En regardant à nouveau les informations textuelles contenues dans les fichiers Dicom, nous pouvons voir (Fig 17) qu'il existe déjà un centre et une largeur de fenêtre (surligné en jaune) donnés pour nous.

```
(0028, 0102) High Bit                    US: 15
(0028, 0103) Pixel Representation        US: 1
(0028, 1050) Window Center               DS: "40"
(0028, 1051) Window Width                DS: "150"
(0028, 1052) Rescale Intercept           DS: "-1024"
(0028, 1053) Rescale Slope               DS: "1"
(7fe0, 0010) Pixel Data                  OW: Array of 524288 elements
```

Figure III.17 Une partie des informations textuelles contenues dans un DICOM

Ces deux paramètres ont été défini par un médecin qui a déterminé selon lui la meilleure plage pour visualiser l'hémorragie. Donc, si nous mettons simplement 256 nuances de gris dans une fenêtre, cela différerait d'un patient à l'autre car les plages de fenêtres données sont différentes. Par conséquent, nous introduirions une source de variation qui n'est pas donnée par les unités HU originales des images.

Pour comprendre mieux ce problème on va collecter les centres et les largeurs de fenêtres d'environ 1000 images pour voir la variété des fenêtres préférées des médecins. Ensuite, nous allons essayer de configurer un niveau et une largeur de fenêtre fixes qui couvrira la majorité de toutes les propriétés de la fenêtre. De cette façon, nous pouvons comparer si une taille de fenêtre personnalisée fixe est mieux adaptée que les tailles de fenêtre de médecins individuelles.

Dans la figure ci-dessous, la représentation des distributions des centres et largeurs de fenêtres utilisées par les médecins :

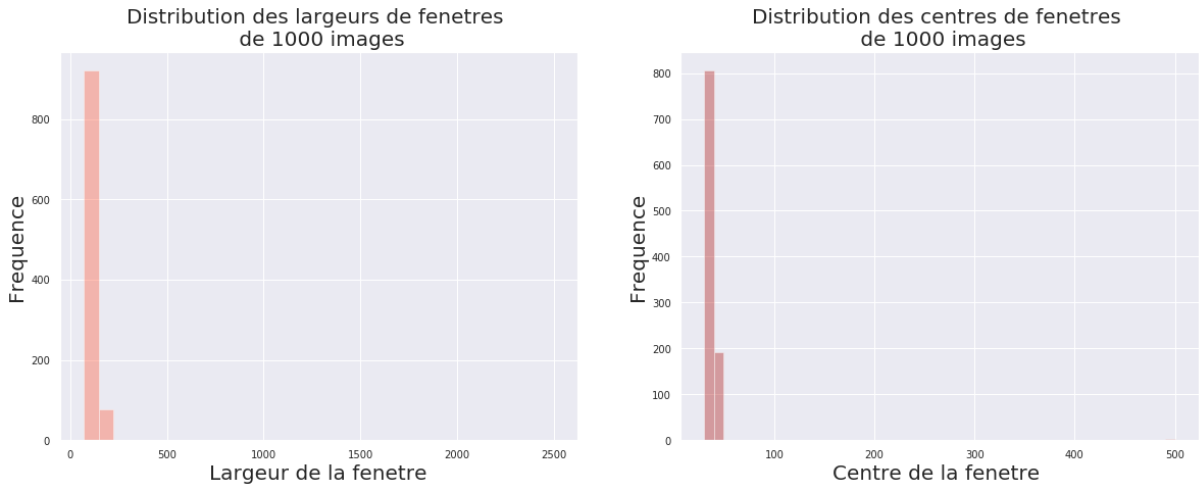


Figure III.18 Distribution des paramètres des fenêtres utilisées par les médecins

Pour une meilleure interprétation on va utiliser une échelle logarithmique :

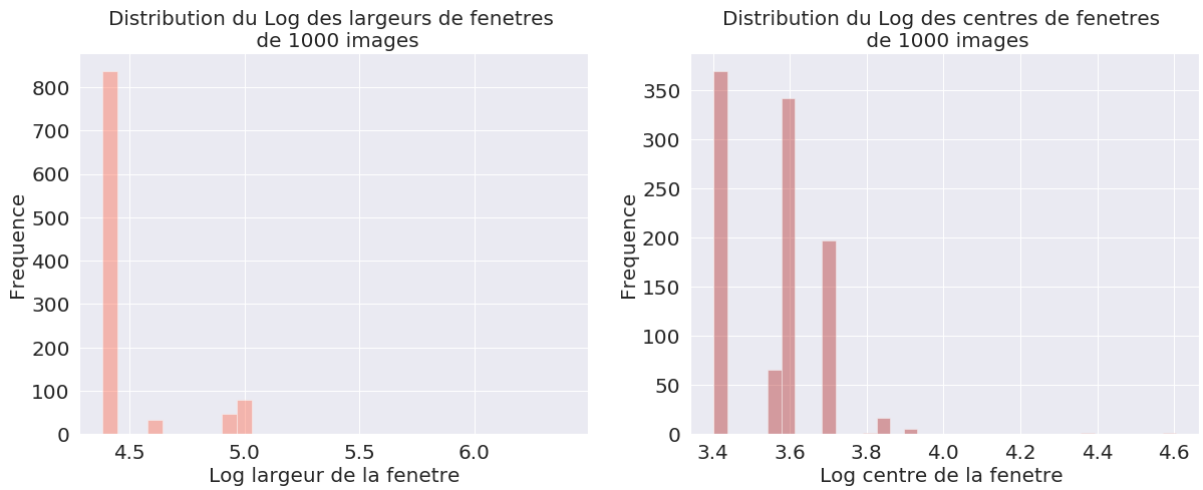


Figure III.19 Distribution logarithmique des paramètres des fenêtres utilisées

Comme on peut le voir dans la figure ci-dessus il existe des valeurs multiples, éparpillées et extrêmes dans les largeurs et les centres des fenêtres. On s'attendait à ce que les médecins se concentrent sur le tissu cérébral dont nous savons que les valeurs de HU le concernant se situent à peu près entre 8 et 70 HU.

Dans ce qui suit nous allons essayer de définir notre propre taille de fenêtre, et faire ensuite une comparaison des résultats obtenus par les différentes fenêtres utilisées.

III.5.5 Notre fenêtre personnalisée

Pour configurer notre propre largeur et centre de fenêtre, nous allons nous baser sur la figure ci-dessous qui représente un échantillon significatif des paramètres de fenêtrage utilisés par les médecins.

	win_width	win_level
ID_30687de66.dcm	80	36
ID_af423ce68.dcm	80	40
ID_2ce81347c.dcm	80	30
ID_3f7c431e5.dcm	80	30
ID_70505b69c.dcm	80	36
ID_7844811d3.dcm	100	40
ID_b1c0f8bbf.dcm	80	36
ID_6d5869a7d.dcm	80	36
ID_2d311a529.dcm	80	30
ID_ca96ed9ce.dcm	150	40
ID_b253caa03.dcm	150	40
ID_f1ff6943f.dcm	80	30
ID_3b1b58ed3.dcm	80	36
ID_b241537a3.dcm	80	30
ID_14d85b761.dcm	100	40
ID_eabe0c122.dcm	80	30
ID_b29d4caf1.dcm	80	47
ID_899363dee.dcm	80	36
ID_6f54405ac.dcm	150	40
ID_f793fe481.dcm	80	30

Figure III.20 Paramètres des fenêtres (largeur et centre) utilisées par les médecins

On remarque que :

- La majorité des centres est située entre 30 et 40 HUs.
- Les largeurs des fenêtres varient entre 70 et 150 HUs.

La fenêtre choisie est centrée à 30 et a une largeur de 80. Ce serait proche de la médiane dans les deux cas. De plus, nous avons remarqué que cette combinaison de largeur et de centre est très utilisée.

Ci-dessous une figure pour comparer quelques images avant (image brute) et après différents traitements (redimensionnement, fenêtrage médecin, fenêtrage personnalisé) :

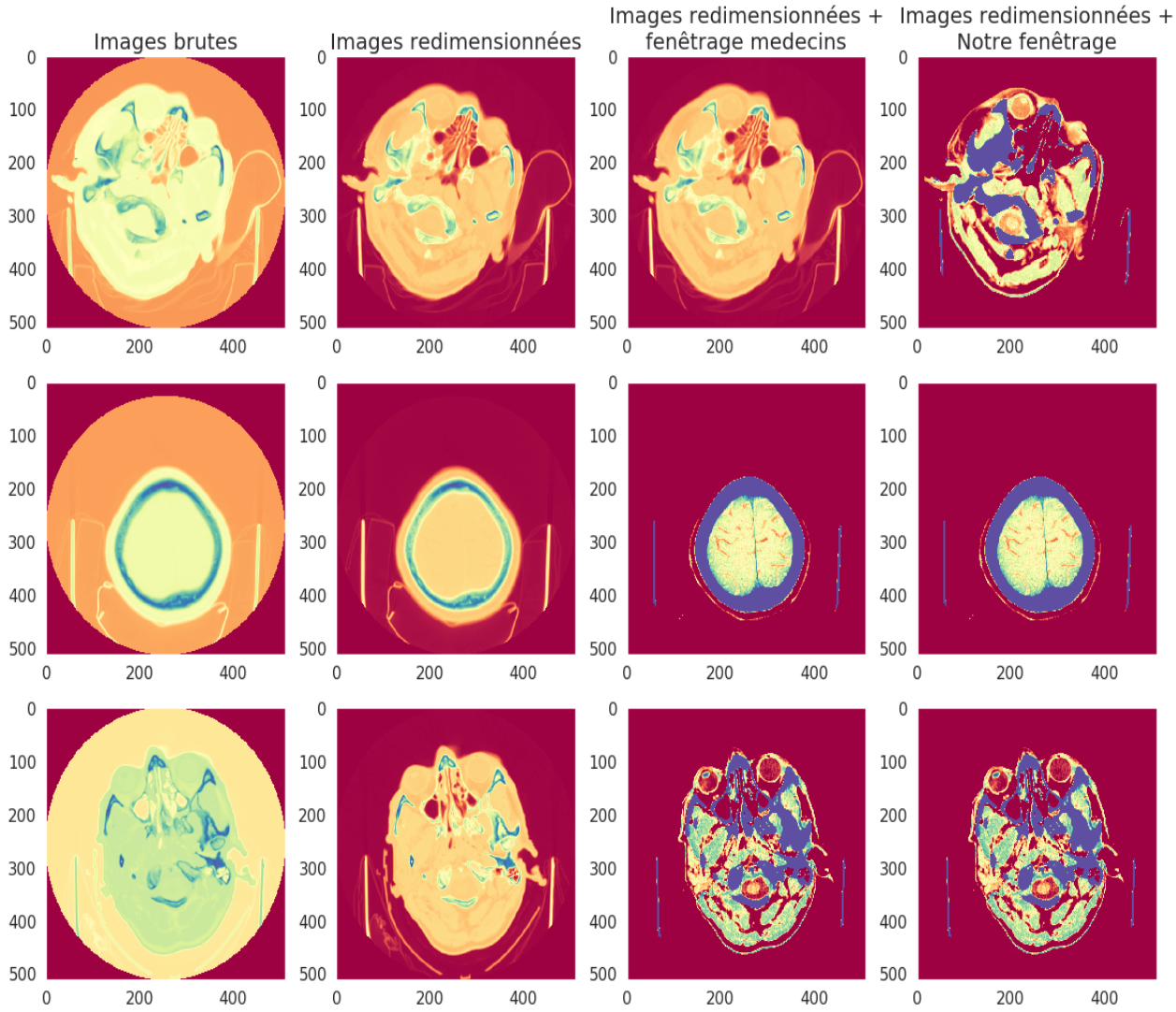


Figure III.21 Images avant et après différents traitements, trois exemples

On remarque que :

- Les images brutes sont peu utiles et ne peuvent être exploitées directement pour faire un diagnostic.

- Après redimensionnement, les régions extérieures au crane ont été assimilés à l'air, nous pouvons ainsi voir une amélioration de la qualité des images mais ces dernières ne sont pour autant exploitables à cause du problème de contraste.
- Après un fenêtrage de médecins le problème de contraste est résolu, nous pouvons enfin voir des jolis motifs qui rendent l'image utile et exploitable.
- Pour les deux derniers exemples, notre fenêtre personnalisée donne le même résultat que la fenêtre choisie par le médecin, par contre pour le premier exemple on peut apercevoir les même contours et motifs mais les valeurs de pixels diffèrent.

III.5.6 Visualisation de quelques cas positifs après prétraitement des images

La figure Ci-dessous (Fig 22) montre six cas atteint d'HIC qu'on visualise après avoir effectué les différentes opérations de prétraitement qui sont le redimensionnement, le filtrage et le fenêtrage (utilisation de notre fenêtre personnalisée).

Le type d'hémorragie présent chez le patient est mentionné au-dessus de chaque image. On rappellera que l'étiquette « any » est activée quand au moins un type d'hémorragie est présent.

On pourra remarquer dans la figure ci-dessous que :

- L'hémorragie apparait avec la couleur verte qui est très en contraste avec les différents tissus présents tout autour d'elle, et par conséquent, elle est facilement détectable après avoir effectué notre prétraitement.
- Le type d'hémorragie est défini selon son emplacement, sa forme et sa proximité d'autres structures.
- Un patient peut être atteint de plusieurs types d'hémorragie à la fois.

La différence de contraste entre les différents tissus et l'hémorragie en elle-même, l'utilisation de la même fenêtre, des mêmes filtres, et d'un même redimensionnement, aidera considérablement notre algorithme de Deep Learning dans l'extraction automatique des caractéristiques, et c'est ce qui nous permettra d'atteindre des performances très poussées dans notre détection et classification.

De ce qui a précédé nous pouvons déduire de la grande importance de l'étape de prétraitement des images, sans quoi notre algorithme ne pourrait atteindre des performances intéressantes dans la détection et la classification d'images.

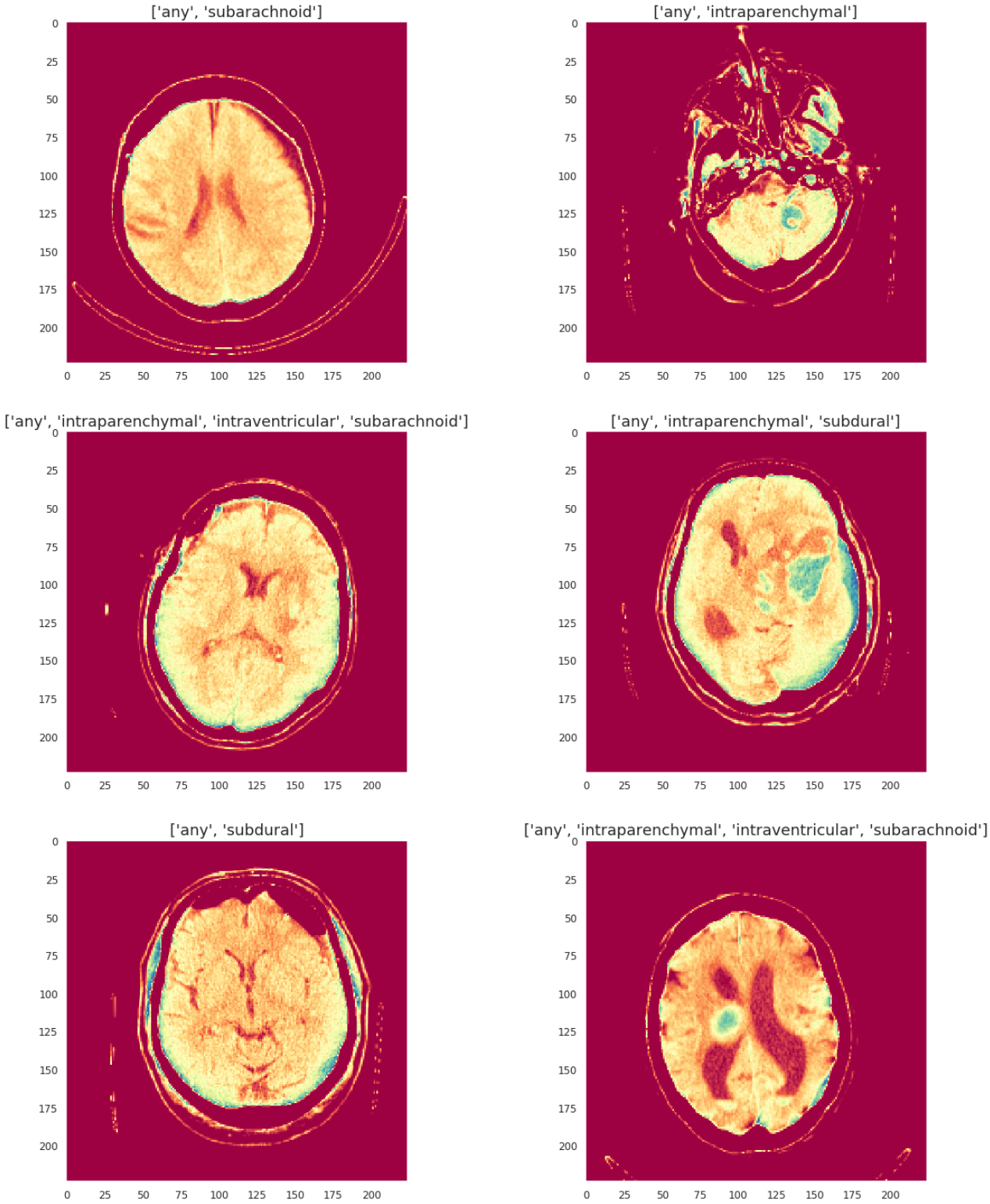


Figure III.22 Quelques cas positifs visualisés avec notre fenêtre personnalisée

III.6 Implémentation, résultats et discussions

III.6.1 Préparons-nous à commencer

Les premières lignes de notre code sont dédiées aux importations de certaines librairies et de leurs différents modules que nous allons utiliser dans la construction de notre programme :

- *NumPy* : Pour traiter et opérer des fonctions mathématiques complexes sur nos tableaux de données multidimensionnelles.
- *Pandas* : Pour **manipuler, fusionner et analyser nos données**.
- *Pydicom* : Pour manipuler, lire, modifier et écrire nos fichiers DICOM.
- *Keras* : Pour créer un modèle de Deep Learning séquentiel, charger et exploiter des modèles pré-entrainés, empiler des couches de neurones, configurer le processus d'apprentissage, lancer l'entrainement de notre modèle, évaluer les performances de notre modèle et générer des prédictions sur de nouvelles données.
- *Matplotlib et Seaborn* : Pour générer des graphiques, créer des analyses statistiques sophistiquées, et interagir avec les Dataframes de Pandas.

III.6.2 Préparation des images

Avant de pouvoir construire un modèle d'apprentissage en profondeur, nous devons configurer notre chargeur de données et notre préprocesseur d'images. Pour réaliser cela, on utilisera Keras.

Notre workflow est le suivant :

- Lire l'ensemble de données Dicom d'une image Dicom brute.
- Concentrons-nous sur notre fenêtre personnalisée. De cette façon, nous pouvons nous débarrasser des informations de l'image dont nous n'avons pas besoin.
- Nous devons nous assurer que toutes les images ont la même largeur et hauteur.
- Comme nous utiliserons des modèles pré-entrainés, nous devons nous assurer que la forme de l'image correspond à celle attendue par notre modèle. Dans le cas contraire nous devrons redimensionner l'image à la forme d'entrée de notre CNN.
- Augmenter nos données avec le module *ImageDataGenerator* de Keras, qui permet de créer un certain nombre de transformations aléatoires (rotation, zoom, décalage, etc.).

III.6.3 Préparation des données

Pour utiliser notre chargeur de données, nous devons transformer notre *traindf* (Training Dataframes : tableau des données de formation) en quelque chose de plus utile. À cette fin, nous devons avoir l'ID de l'image dans une colonne et les sous-types d'hémorragie dans d'autres colonnes individuelles.

Notre *traindf* sera transformé de la forme ci-dessous (Fig 23) :

	ID	Label
0	ID_12cadc6af_epidural	0
1	ID_12cadc6af_intraparenchymal	0
2	ID_12cadc6af_intraventricular	0
3	ID_12cadc6af_subarachnoid	0
4	ID_12cadc6af_subdural	0

Figure III.23 *traindf* avant transformation

A une forme plus utile grâce à la commande *pivot_table* de la librairie Pandas. Nous obtenons un *traindf* sous la forme souhaitée (Fig 24) :

subtype id	any	epidural	intraparenchymal	intraventricular	subarachnoid	subdural
ID_000012eaf	0	0	0	0	0	0
ID_000039fa0	0	0	0	0	0	0
ID_00005679d	0	0	0	0	0	0
ID_00008ce3c	0	0	0	0	0	0
ID_0000950d7	0	0	0	0	0	0

Figure III.24 *traindf* après transformation

III.6.4 Modèles pré-entrainés

A l'aide de la technique du *transfert learning*, nous allons exploiter et adapter cinq architectures connues, pré-entraînés sur la célèbre base de données d'images *ImageNet*, à notre problème de classification. Par la suite nous ferons une comparaison des différents résultats obtenus par chaque architecture et pour différents paramètres. A la suite de quoi nous pourrons choisir la meilleure architecture et les meilleurs paramètres pour résoudre notre problème.

Dans le tableau ci-dessous (Tab 2) l'énumération des différentes architectures exploitées et de certains détails les concernant :

Modèle	Taille	Top-1 Accuracy	Paramètres	Profondeur
ResNet50	98 MB	0.749	25,636,712	-
VGG16	528 MB	0.713	138,357,544	23
Xception	88 MB	0.790	22,910,480	126
InceptionV3	92 MB	0.779	23,851,784	159
InceptionResNetV2	215 MB	0.803	55,873,736	572

Tableau III.2 Les architectures CNNs exploitées

III.6.4.1 Construction de notre ligne de base

III.6.4.1.1 Construction du Modèle

Pour adapter les architectures pré-entrainées à notre problème, il suffit de :

- Geler ‘freeze’ (i.e. les paramètres internes ne peuvent plus être modifiés) les premières couches de convolutions du CNN.
- Remplacer les dernières couches *Fully-Connected* qui permettent de classifier l'image dans une des 1000 classes ImageNet par un classifieur avec six classes de sortie, représentants les cinq sous-types de l’HIC plus la classe ‘*any*’.

La suppression des dernières couches se fait en ajoutant l'argument : *include_top=False* lors de l'import du modèle pré-entraîné.

III.6.4.1.2 Compilation du Modèle

La compilation du modèle prend trois paramètres : optimiseur, perte et métriques

L'optimiseur contrôle le taux d'apprentissage. *Keras* offre plusieurs choix d’optimiseurs : SGD, RMSprop, Adadelta, Adam.

Nous utiliserons '**Adam**' (Adaptive Moment estimation) comme optimiseur car :

- Il ajuste le taux d'apprentissage tout au long de la formation. Le taux d'apprentissage détermine la vitesse à laquelle les poids optimaux pour le modèle sont calculés. Un taux d'apprentissage plus élevé implique une convergence rapide et donc un taux d'apprentissage insuffisant et faible implique un temps long pour l'ajustement du modèle.
- Il permet d’entrainer le CNN en moins de temps et plus efficacement.
- Ses besoins en mémoire sont relativement faibles.
- Fonctionne généralement bien même avec peu de réglage des hyperparamètres.

Le taux d'apprentissage (**LR** : Learning Rate) est un hyperparamètre qui contrôle dans quelle mesure le modèle doit être modifié en réponse à l'erreur estimée chaque fois que les poids du modèle sont mis à jour. Le choix du taux d'apprentissage est difficile, car une valeur trop petite peut entraîner un long processus de formation qui pourrait rester bloqué, tandis qu'une valeur trop grande peut entraîner l'apprentissage d'un ensemble de poids sous-optimal trop rapidement ou un processus de formation instable.

Le taux d'apprentissage peut être l'hyperparamètre le plus important lors de la configuration de notre réseau neuronal.

Nous utiliserons **'categorical_crossentropy'** pour notre fonction de perte. Il s'agit du choix de classification le plus courant. Un score inférieur indique que le modèle fonctionne mieux.

Les métriques « **Accuracy** (Précision) **»** et « **Loss** (Perte) » sont utilisées pour voir le score réalisé sur l'ensemble de validation lorsque nous entraînons le modèle.

III.6.4.1.3 Entrainement du modèle

La formation ou bien l'entrainement du modèle est possible grâce à la fonction **fit_generator**. Cette dernière requiert en entrée la définition d'un certain nombre d'arguments :

- **generator** : Un générateur dont la sortie doit être une liste de la forme : (entrées, cibles).

- **steps_per_epoch** : Spécifie le nombre total de pas effectué par le générateur dès qu'une époque est terminée. Nous pouvons définir la valeur de **steps_per_epoch** comme le nombre total d'échantillons dans notre ensemble de données divisé par la taille du lot (ou **batch size** : le nombre d'exemples vus par le réseau en une seule backpropagation).

- **Epochs** : C'est le nombre d'époques pour lesquelles nous voulons entraîner notre modèle. Une Epoque, correspond à un seul passage en entier de nos données de formation à travers le réseau neuronal.

- **Validation_data** : Un générateur que nous utilisons pour évaluer la perte et les métriques pour tout modèle après la fin de toute époque.

- **validation_steps** : Il spécifie le nombre total de pas effectués depuis le générateur avant qu'il ne soit arrêté à chaque époque et sa valeur est définie comme étant le nombre total de données de validation dans notre ensemble de données divisé par la taille du lot de validation.

Pour créer un modèle solide, nous devons suivre un protocole spécifique de fractionnement de nos données d'entrainement en deux ensembles : un pour la formation, et un

pour la validation. On utilise la fonction *validation_split=0.2*, pour réserver 20% de nos données de formation à la validation. L'idée est de nous entraîner sur nos données d'entraînement et d'ajuster notre modèle avec les résultats des métriques (précision, perte) que nous obtenons de notre ensemble de validation. Seule cette procédure nous garantit la capacité à généraliser du modèle, en évitant ainsi le phénomène du surapprentissage (overfitting).

III.6.5 Formation, Validation : Résultats et discussions

Ci-dessous un tableau (Tab 3) récapitulatif des différents résultats de formation des cinq modèles pré-entrainés. On varie dans le nombre maximal d'époques (10,20,30) et on garde fixe le Batch_size = 16, optimiseur = Adam et LR= 0.0001.

		ResNet50	VGG16	Xception	InceptionV3	InceptionResNetV2
MaxEpochs =10	Loss	0.53	0.53	0.51	0.66	0.63
	Accuracy	0.64	0.75	0.77	0.73	0.73
MaxEpochs =20	Loss	0.69	**0.46**	0.50	0.47	0.46
	Accuracy	0.64	**0.86**	0.77	0.74	0.76
MaxEpochs =30	Loss	0.57	0.48	0.46	0.53	0.46
	Accuracy	0.67	0.80	0.76	0.74	0.77

Tableau III.3 Tableau comparatif des résultats obtenus pour Batch_size=16, LR=0.0001

Pour le moment on retient la configuration : (modèle=VGG16, MaxEpochs=20), qui donne le meilleur résultat avec un Accuracy de 86% qui est acceptable, mais une valeur du Loss très élevée.

Les figures (Fig 25) et (Fig 26) ci-dessous représentent les variations du Loss et de sa validation, ainsi que les variations de l'Accuracy et de sa validation. On remarque bien qu'on n'a pas de problème de surapprentissage, car la courbe de validation est proche de celle de la métrique.

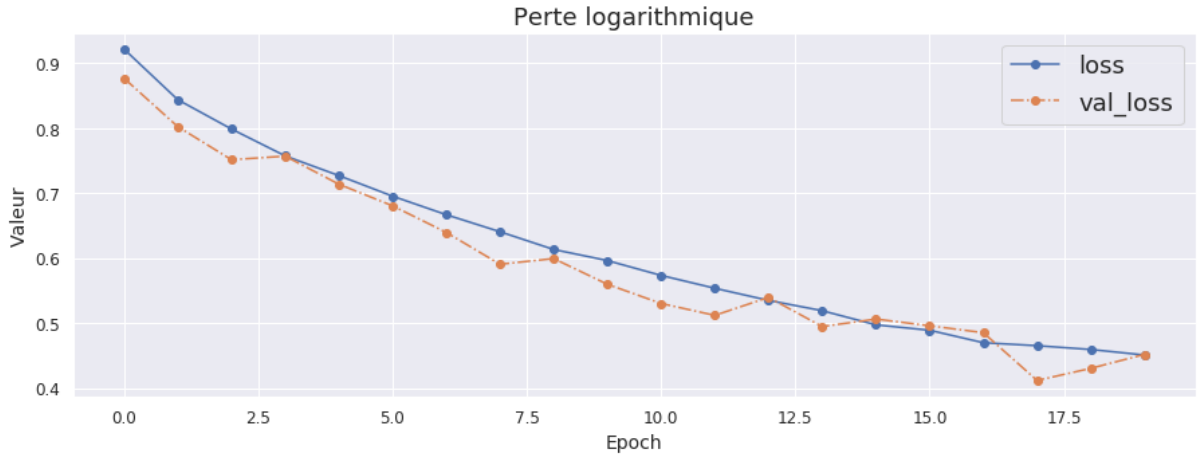


Figure III.25 Loss et Validation Loss de la configuration 1 retenue

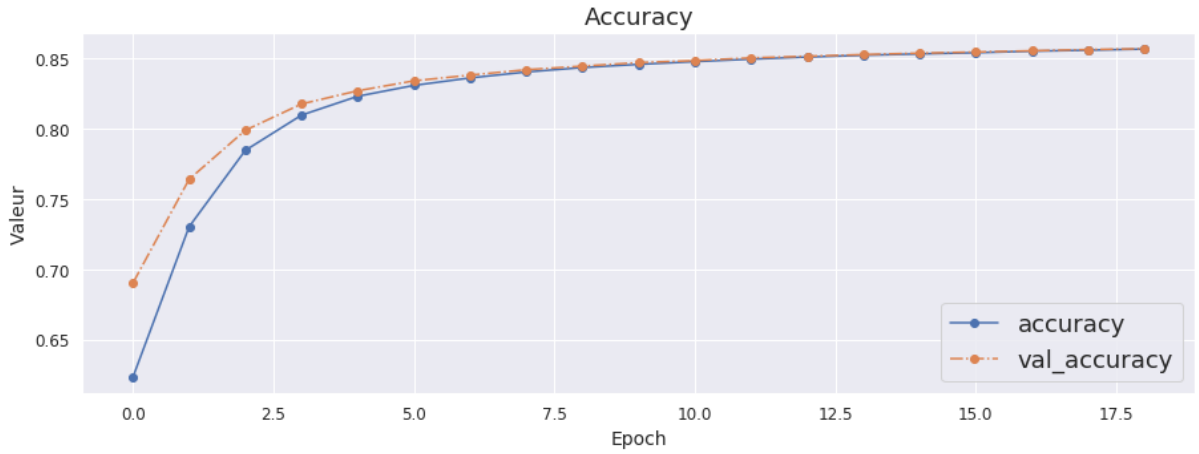


Figure III.26 Accuracy et Validation Accuracy de la configuration 1 retenue

Pour essayer d'améliorer les performances de ce modèle, on garde la même configuration et on augmente seulement le **Batch_size** à **32.** On visualise les résultats :

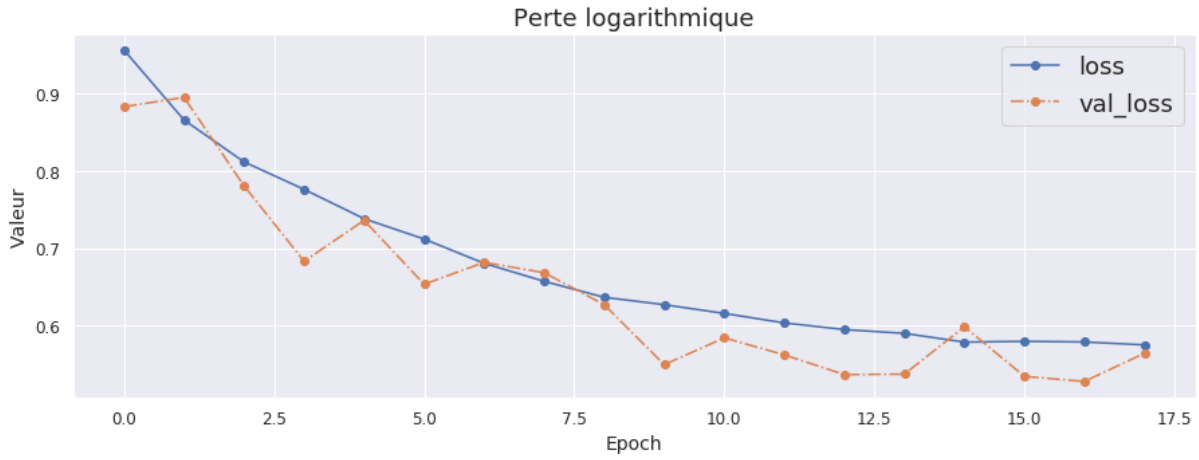


Figure III.27 Loss et Validation Loss, Batch_size = 32

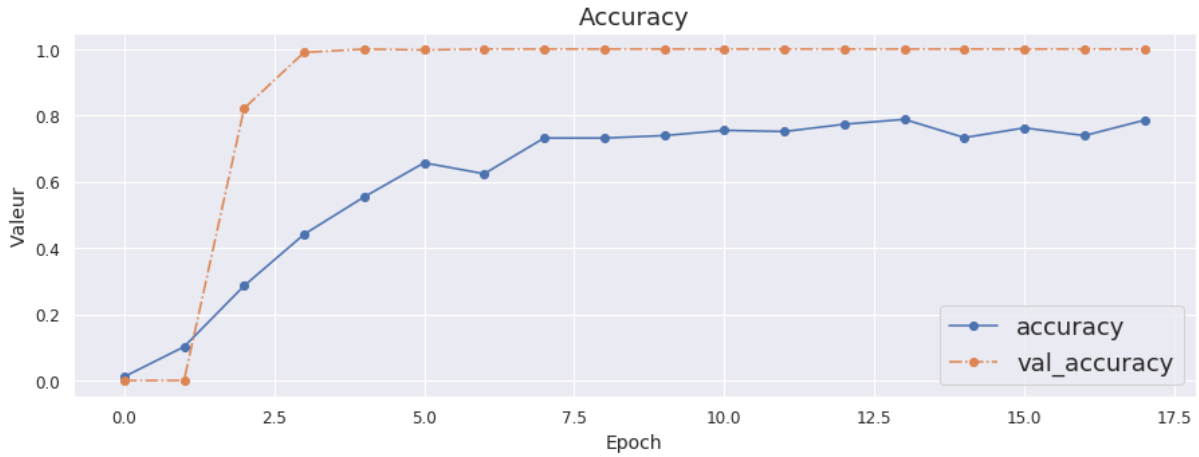


Figure III.28 Accuracy et Validation Accuracy, Batch_size = 32

On remarque qu'il y a une détérioration dans les performances du modèle, Accuracy dégradé à 79 % et Loss à 58%. On reconfigure le **Batch_size à 16**.

Pour confirmer que l'optimiseur **Adam** est celui qui donne les meilleures performances, on essaye à sa place l'optimiseur **SGD** (La descente de gradient stochastique) tout en gardant la configuration 1 retenue (architecture=VGG16, MaxEpochs=20, Batch_size =16 et LR= 0.0001**).** On obtient les résultats suivants :

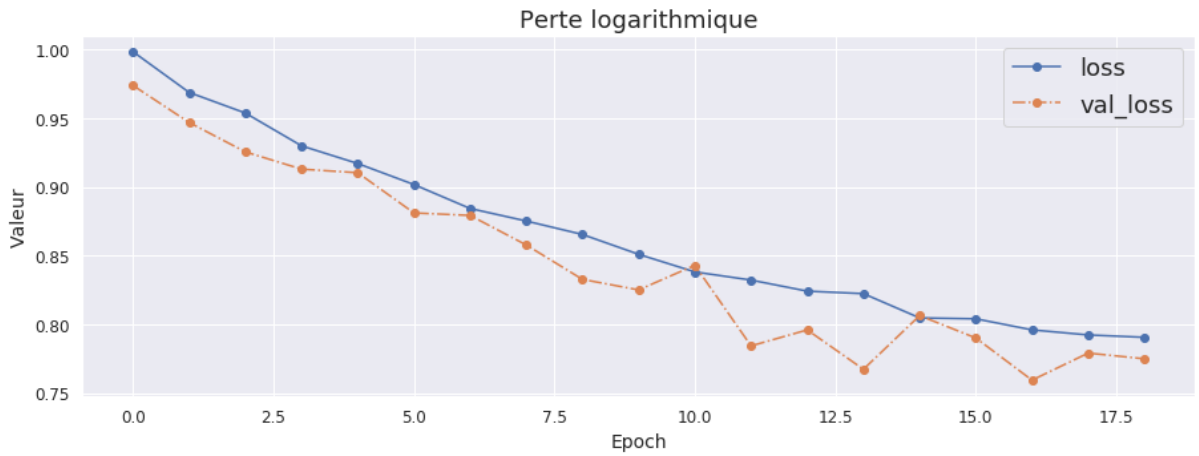


Figure III.29 Loss et Validation Loss, Optimiseur = SGD

On remarque bien dans les figures (Fig 29 et Fig 30) que les performances obtenues par l'optimiseur **SGD** sont très médiocres (Loss = 79%, Accuracy= 24%). On reconfigure donc, notre optimiseur sur **Adam.**

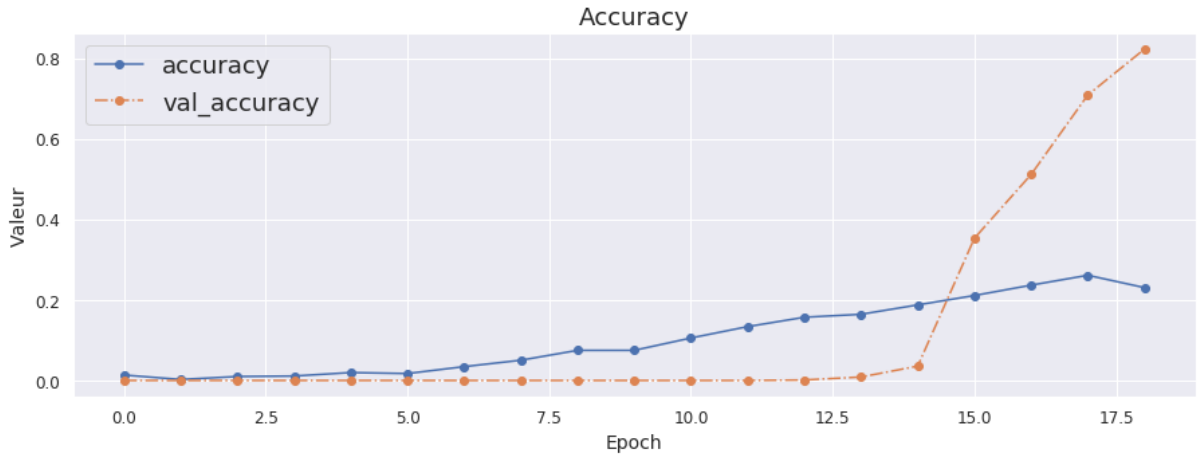


Figure III.30 Accuracy et Validation Accuracy, optimiseur = SGD

Comme cité plus haut dans ce chapitre, un des hyperparamètres les plus importants lors de la configuration de notre modèle est : le taux d'apprentissage (LR). On a travaillé sur ce dernier, et après plusieurs simulations avec différents LR, on décide de fixer le taux d'apprentissage à **0.001**. Ce taux nous a donné les meilleurs résultats.

Ci-dessous un tableau (Tab 4) récapitulatif des différents résultats d'entrainement obtenus pour la nouvelle configuration : Batch_size **= 16**, optimiseur = **Adam** et LR= **0.001.**

		ResNet50	VGG16	Xception	InceptionV3	InceptionResNetV2
MaxEpochs =10	Loss	0.24	0.23	0.25	0.24	0.22
	Accuracy	0.86	0.83	0.83	0.88	0.87
MaxEpochs =20	Loss	0.22	0.24	0.23	0.27	0.23
	Accuracy	0.88	0.94	0.87	0.91	0.89
MaxEpochs =30	Loss	0.23	**0.22**	0.23	0.24	0.25
	Accuracy	0.90	**0.96**	0.91	0.91	0.90

Tableau III.4 Tableau comparatif des résultats obtenus pour Batch_size=16, LR=0.001

A première vue, on remarque une nette amélioration des performances des cinq modèles entrainés. La configuration 2 retenue : Architecture =**VGG16**, MaxEpochs=**30.**

Cette configuration nous a permis d'atteindre un Accuracy de 96%, un excellent résultat à priori. Nous avons également pu diminuer le Loss jusqu'à 22%, un taux assez acceptable, mais qui peut être amélioré. Ci-dessous les figures (Fig 31 et Fig 32) représentants les résultats obtenus :

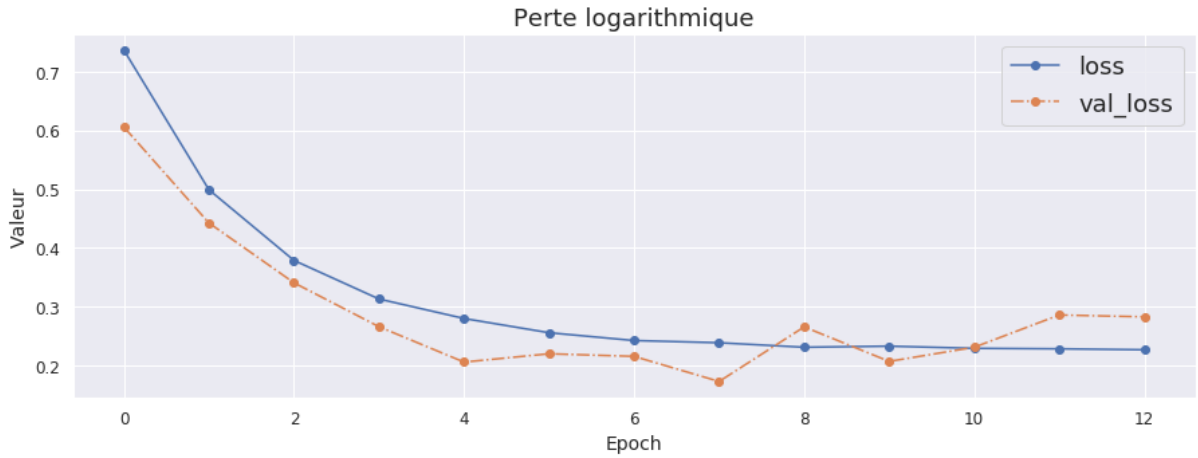


Figure III.31 Loss et Validation Loss de la configuration 2 retenue

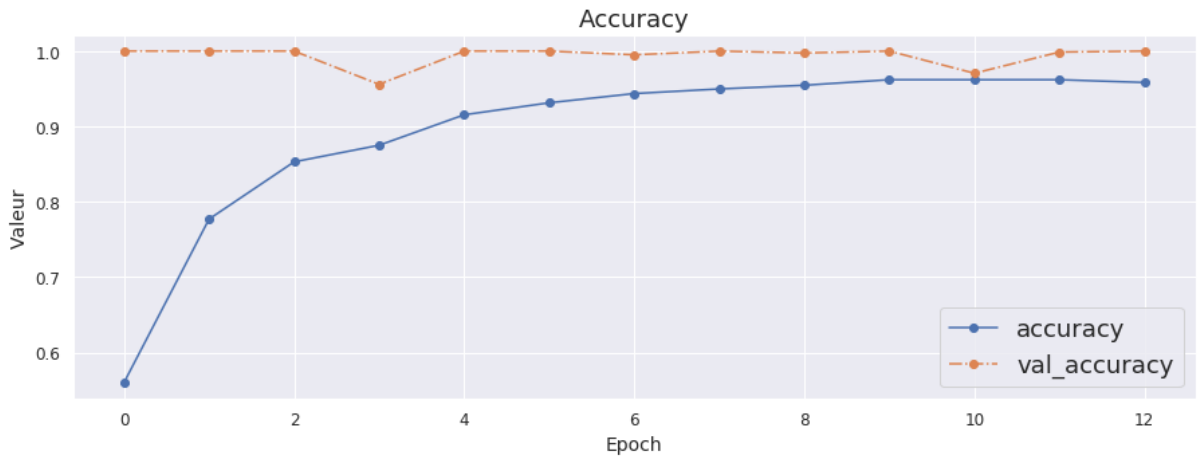


Figure III.32 Accuracy et Validation Accuracy de la configuration 2 retenue

Par ailleurs, on remarque aussi bien qu'on n'a pas de problème de surapprentissage (overfitting), la courbe de validation démontre bien que notre modèle parvient à généraliser sur des échantillons qui lui sont inconnus (les données de validation).

III.6.6 Tests : Résultats et discussions

Après avoir formé notre modèle nous allons le tester sur deux ensembles de données. Le Jeu de test 1 contient 3951 images et le Jeu de test 2 contient 35116 images.

III.6.6.1 Jeu de test 1

III.6.6.1.1 Prédiction de la sortie

Pour cela nous allons utiliser la fonction **Predict_generator,** qui à partir des données d'entrées de l'ensemble test 1 va prédire la sortie. La figure ci-dessous (Fig 33) représente les résultats de prédiction.

Subtypes Images	any	epidural	intraparenchymal	intraventricular	subarachnoid	subdural
ID_0000950d7	1.865054e-01	0.003874	3.815332e-02	2.123520e-02	0.084725	0.067478
ID_00032d440	1.198896e-01	0.001927	2.728727e-02	1.099879e-02	0.035104	0.037393
ID_0007256ab	1.198833e-01	0.002759	2.118385e-02	1.025757e-02	0.046748	0.053551
ID_0009db386	2.459015e-01	0.005002	3.564301e-02	8.426279e-03	0.100150	0.150847
ID_000d44b3c	6.421371e-01	0.007589	3.428724e-01	1.920389e-01	0.208245	0.150521
...	...	...	...	...	...	...
ID_327f58085	1.262724e-04	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000
ID_3281c6608	1.788139e-07	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000
ID_32956a237	2.748579e-03	0.000000	3.576279e-07	5.960464e-08	0.000005	0.000002
ID_32979ed41	0.000000e+00	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000
ID_329b258a8	5.960464e-08	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000

3951 rows × 6 columns

Figure III.33 Prédiction du Jeu de test 1

Comme nous pouvons le voir sur la figure ci-dessus (Fig 33) la fonction Predict permet de calculer les différentes probabilités d'existence de chaque sous-type de l'HIC chez les différents patients.

Pour avoir une prédiction sous forme binaire (0 ou 1) on doit fixer un seuil de prise de décision. Après comparaison des résultats obtenus avec plusieurs seuils, on fixe notre seuil à 0.25. On aura :

$$Prediction = \begin{cases} 0, & Si\ Prediction < 0.25 \\ 1, & Si\ Prediction \geq 0.25 \end{cases} \qquad (3.1)$$

En appliquant cette condition sur le résultat précédent (Fig 33), nous obtenons un tableau de prédiction binaire comme on peut le voir dans la figure ci-dessous (Fig 34). Ce tableau sera utilisé dans la suite de notre travail dans la phase d'évaluation de notre modèle.

Subtypes Images	any	epidural	intraparenchymal	intraventricular	subarachnoid	subdural
ID_0000950d7	1	0	0	0	0	0
ID_00032d440	0	0	0	0	0	0
ID_0007256ab	0	0	0	0	0	0
ID_0009db386	1	0	0	0	0	1
ID_000d44b3c	1	0	1	1	1	1
...	...	...	...	...	...	...
ID_327f58085	0	0	0	0	0	0
ID_3281c6608	0	0	0	0	0	0
ID_32956a237	0	0	0	0	0	0
ID_32979ed41	0	0	0	0	0	0
ID_329b258a8	0	0	0	0	0	0

3951 rows × 6 columns

Figure III.34 Prédiction binaire du Jeu de test 1

III.6.6.1.2 Evaluation du modèle

Pour évaluer notre modèle on utilise la fonction **Evaluate_generator**. Cette dernière utilise à la fois l'entrée et la sortie du Jeu de test. Elle prédit d'abord la sortie à l'aide de l'entrée d'entraînement, puis évalue les performances en les comparant à la sortie de notre Jeu de test. Cela donne donc une mesure de la performance, dans notre cas cela donne le Loss et l'Accuracy. Après exécution nous obtenons les scores suivants :

```
print("Loss = ", Test_score[0]),
print("Accuracy =", Test_score[1])
```

```
Loss =  0.1592008540092375
Accuracy = 0.9464692574336661
```

Figure III.35 Scores d'évaluation Jeu 1

Comme on peut le voir dans la figure (Fig 35) nous réalisons un score d'environ 16% en Loss, et 95% en Accuracy.

Ci-dessous (Fig 36) les matrices de confusion des sous-types de l'HIC. Ces matrices ont été calculé grâce à la fonction **multilabel_confusion_matrix** qui prend comme arguments en entrée les sorties réelles et les sorties prédites.

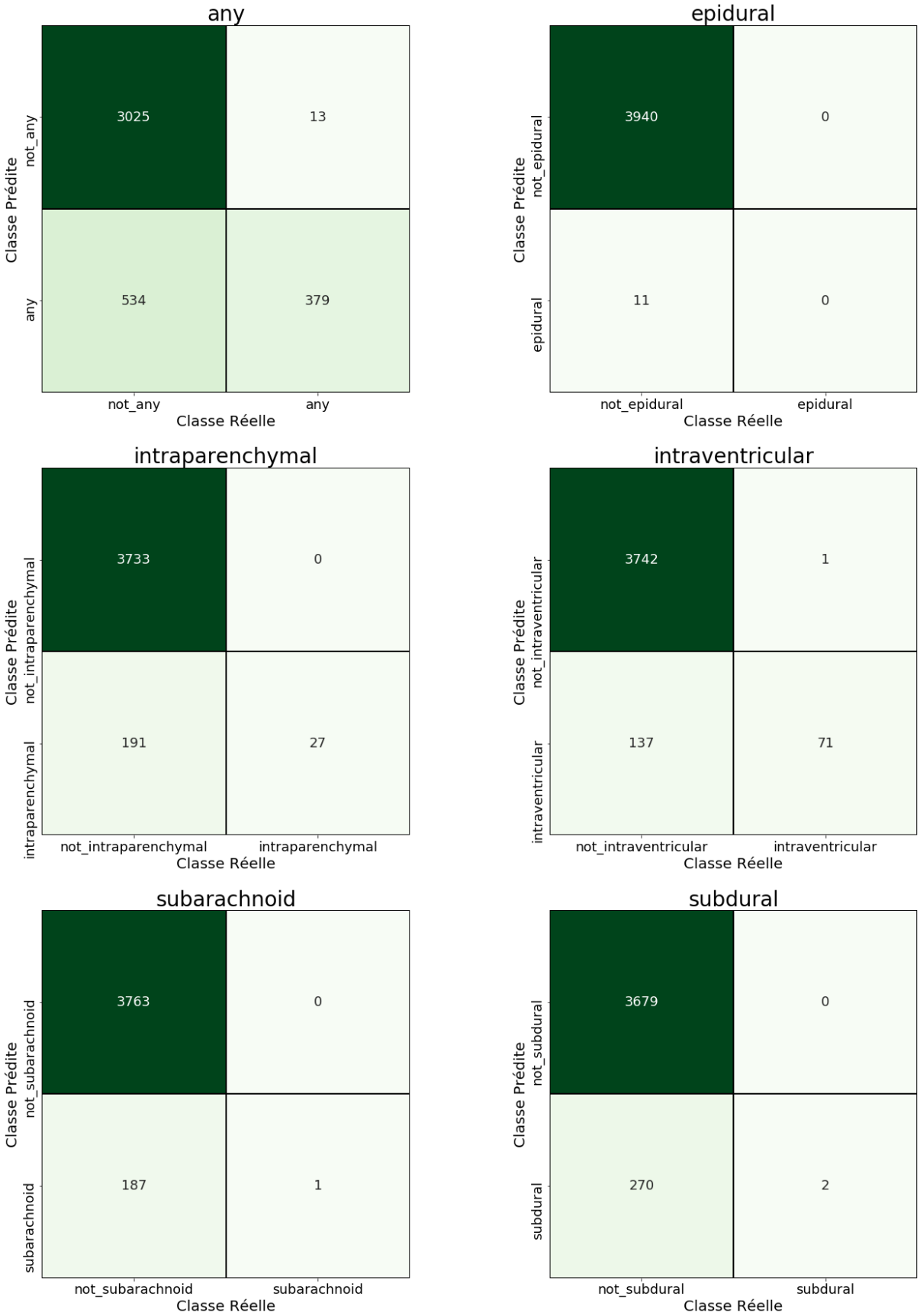


Figure III.36 Matrices de confusion des sous-types de l'HIC - Jeu 1

Sur la gauche des matrices les classes prédites et en bas d'elles les classes réelles. A partir de ces matrices on remarque :

- La grande majorité des cas est 'Négatif' et est bien classée (Le petit carré vert).
- La proportion des cas positifs est très minime, d'où la difficulté de pouvoir les détectés.
- Sur les 3951 images du Jeu de test 1, aucune HIC de type 'epidural' n'est présente. Comme vu dans la partie analyse des données ce sous-type est très rare, il ne représente que moins d'un pourcent (<1%) du total des sous-types.
- Notre modèle a pu détecter les rares cas 'subarachnoid' et 'subdural' positifs (1 cas et 2 cas resp.) présent dans le Jeu de test 1.
- Pour le sous-type 'intraventricular' notre Modèle a détecté correctement 71 cas positifs, et n'a raté qu'un (01) seul cas positif pour l'ensemble des images du Jeu de test 1.
- Pareil pour le sous-type 'intraparenchymal' ou notre modèle a pu détecter tous les cas positifs (27) de ce type, présents dans les 3951 images de tests.
- Toutefois, on remarque que le nombre des 'faux positifs' dans toutes ces matrices de confusion n'est pas à négliger, nous devons penser peut-être à changer de seuil.

III.6.6.2 Jeu de test 2

III.6.6.2.1 Prédiction de la sortie

Subtypes Images	any	epidural	intraparenchymal	intraventricular	subarachnoid	subdural
ID_0000950d7	0.126028	0.002068	0.039406	0.026017	0.041871	0.018900
ID_000178e76	0.815953	0.043473	0.493576	0.030754	0.385240	0.333018
ID_0001de0e8	0.088898	0.003135	0.013993	0.000620	0.033888	0.035466
ID_0002108bd	0.006541	0.000176	0.000798	0.000662	0.004153	0.001640
ID_000270f8b	0.588335	0.012865	0.348961	0.008836	0.072648	0.164167
...	...	...	...	...	...	...
ID_33708921f	0.008534	0.000259	0.000786	0.000388	0.003297	0.002592
ID_3370e6861	0.098005	0.001942	0.015278	0.005290	0.021959	0.044836
ID_33722c9e5	0.042245	0.001359	0.009214	0.004254	0.011142	0.018426
ID_33726693b	0.112418	0.002767	0.025948	0.026330	0.034790	0.039227
ID_3372741d2	0.058230	0.000910	0.004654	0.000434	0.021814	0.022793

35116 rows × 6 columns

Figure III.37 Prédiction du Jeu de test 2

Toujours à l'aide de la fonction **Predict_generator**, nous obtenons les résultats de prédiction sur le Jeu de test 2, comme nous pouvons voir sur la figure ci-dessus (Fig 37).

En appliquant un seuillage de décision à 0.25, nous obtenons un tableau de prédiction binaire comme on peut le voir dans la figure ci-dessous (Fig 38). Comme dans l'exemple précédent, ce tableau sera utilisé dans la suite de notre travail dans la phase d'évaluation de notre modèle sur ce deuxième Jeu de test.

Subtypes Images	any	epidural	intraparenchymal	intraventricular	subarachnoid	subdural
ID_0000950d7	0	0	0	0	0	0
ID_000178e76	1	0	1	0	1	1
ID_0001de0e8	0	0	0	0	0	0
ID_0002108bd	0	0	0	0	0	0
ID_000270f8b	1	0	1	0	0	0
...	...	...	...	...	...	...
ID_33708921f	0	0	0	0	0	0
ID_3370e6861	0	0	0	0	0	0
ID_33722c9e5	0	0	0	0	0	0
ID_33726693b	0	0	0	0	0	0
ID_3372741d2	0	0	0	0	0	0

35116 rows × 6 columns

Figure III.38 Prédiction binaire du Jeu de test 2

III.6.6.2.2 Evaluation du modèle

En utilisant toujours la fonction **Evaluate_generator** nous allons évaluer les performances de notre modèle sur le Jeu de test 2. Les performances en question sont le Loss et l'Accuracy. Après exécution nous obtenons les scores suivants :

```
print("Loss = ", Test_score[0]),
print("Accuracy =", Test_score[1])
```

```
Loss =  0.12035466225077113
Accuracy = 0.9577732885504437
```

Figure III.39 Scores d'évaluation Jeu 2

Comme on peut voir dans la figure ci-dessus (Fig 39), nous obtenons un meilleur score sur ce deuxième Jeu de test qui est à peu près dix fois plus grand que le premier. On a atteint environ 96% en Accuracy, et 12% en Loss.

Dans la Figure 40 dans la page qui suit, on peut voir les matrices de confusion des sous-types de l'HIC calculées pour ce deuxième Jeu de test en utilisant toujours la fonction **multilabel_confusion_matrix** qui prend comme arguments d'entrées : les sorties réelles (càd les données de test contenu dans un fichier csv.) et les sorties prédites qu'on peut voir plus haut sur la Figure 38.

Toujours, on retrouve sur la gauche des matrices les classes prédites et en bas d'elles les classes réelles.

En visualisant ces matrices de confusion on remarque :

- Pareil que dans le Jeu de test 1, la grande majorité des cas est 'Négatif' (TN) et est bien classée (Le petit carré grenat).

- Pareil que dans le Jeu de test 1, et comme remarquer lors de la partie analyse et visualisation des données, le nombre des cas positifs est très petit par rapport au nombre des cas négatifs. A titre d'exemple, pour le sous-type 'subarchnoid' on a 1194 cas positifs contre 33922 cas négatifs, (3.5%).

- Pareil aussi, sur les 35116 images du Jeu de test 2, aucune HIC de type 'epidural' n'est présente. Ce qui confirme la rareté de ce sous-type dans notre Jeu de test.

- Pour le sous-type 'intraparenchymal' : notre modèle a pu détecter 1367 cas positifs à ce sous-type et a raté 96 cas en les classant comme négatifs (des faux négatifs FN).

- Pour le sous-type 'intraventricular' : notre modèle a détecté correctement 1711 cas positifs, et a raté 70 cas positif pour l'ensemble des images du Jeu de test 2.

- Pour le sous-type 'subarachnoid' : notre modèle a pu détecter 1139 cas positifs à ce sous-type et a raté 55 cas en les classant comme négatifs (des faux négatifs FN).

- Pour le sous-type 'subdural' : notre modèle a détecté correctement 1854 cas positifs, et a raté 70 cas positif pour l'ensemble des images du Jeu de test 2.

- Même remarque concernant le nombre des 'faux positifs' dans toutes ces matrices de confusion qui reste non négligeable, même après avoir essayer de fixer un seuil de telle sorte à le minimiser. Pour régler ce problème nous devons penser à affiner plus notre modèle.

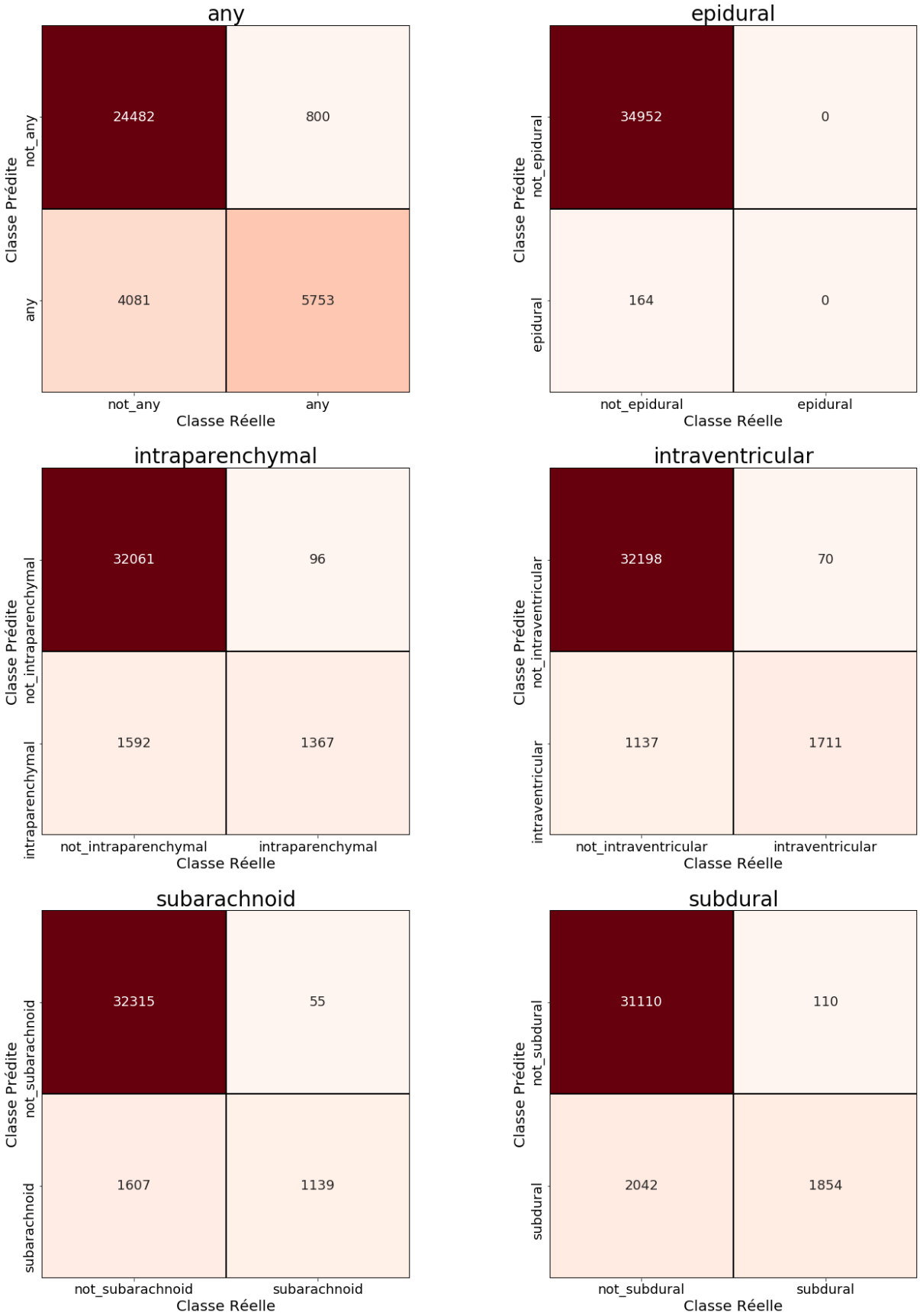


Figure III.40 Matrices de confusion des sous-types de l'HIC - Jeu 2

III.7 Conclusion

Dans ce troisième et dernier chapitre, nous avons présenté la problématique sur laquelle nous avons travaillé et les différentes ressources matérielles et logiciels que nous avons utilisé. Ensuite nous nous sommes concentrés sur l'analyse exploratoires et la mise en forme de nos données qui nous ont aidé dans le choix des outils et méthodes de modélisation pour répondre à notre problème. Après quoi nous avons entamé l'étape de prétraitement des images, sans quoi notre algorithme ne pourrait atteindre des performances intéressantes dans la détection et la classification d'images.

Dans la deuxième partie de ce chapitre, nous avons construit la ligne de base de notre modèle à l'aide de la bibliothèque Keras, ensuite nous l'avons compilé avec Adam comme optimiseur et categorical_crossentropy comme fonction de perte. Nous avons exploité et formé cinq architectures de CNN pré-entrainés afin de comparer leurs performances sur notre jeu de données. L'architecture retenue est VGG-16, cette dernière nous a permis d'atteindre un accuracy de 96% et un Loss de 12%.

CONCLUSION GENERALE

Conclusion générale

Dans le parcours de soin, la phase de diagnostic est essentielle pour l'orientation du patient et son suivi. Le Deep learning apporte de nouvelles solutions aux professionnels de santé pour gagner du temps et optimiser le bon diagnostic. Il ouvre de nouvelles perspectives dans le repérage des maladies et peut ainsi aider les médecins à détecter plus facilement les anomalies sur les radios des patients.

L'objectif n'est pas de remplacer le médecin par la machine, mais de l'accompagner dans l'analyse et l'interprétation des énormes volumes de données collectées. Le Deep learning permet également de favoriser les bons diagnostics et de lutter contre les erreurs médicales en générant des diagnostics différentiels et suggérant des examens complémentaires.

Le présent travail se veut comme un outil d'aide à la détection de l'HIC et de ses cinq (05) sous-types. Notre modèle, à partir d'une image TDM cérébrale en entrée pourra détecter la présence d'une HIC chez le patient, et pourra même la classée dans un de ses sous-types avec une précision de 96%.

Dans la première partie de ce mémoire nous nous familiariser avec les différentes notions de bases concernant l'HIC ainsi que les différentes modalités d'imageries utilisées pour son diagnostic, en l'occurrence la Tomodensitométrie (TDM) et l'Imagerie par Résonance Magnétique (IRM).

Ensuite nous nous sommes plongés dans le domaine de l'IA et de ses branches : le Machine Learning et le Deep Learning. Les travaux actuels tendent à utiliser les réseaux de neurones convolutifs CNNs (un des algorithmes les plus populaires dans le domaine du Deep Learning) pour concevoir des systèmes de prédiction et de diagnostic médical. C'est pourquoi nous nous sommes basés sur ces derniers. Nous avons défini les CNNs avec leurs différentes couches, vu quelques étapes majeures de leur évolution et cité les différents exploits qui ont été accomplis avec. Enfin nous avons présenté les architectures les plus courantes des CNNs et quelques métriques de leur évaluation.

Dans la partie implémentation de ce travail, nous avons en premier lieu cité les différentes ressources matérielles et logiciels que nous avons utilisé. Ensuite nous nous sommes concentrés sur l'analyse exploratoires et la mise en forme de nos données. Après quoi nous avons entamé l'étape de prétraitement des images, sans quoi notre algorithme ne pourrait atteindre des performances intéressantes dans la détection et la classification d'images.

Ensuite, nous nous sommes intéressés à la construction, la compilation et la formation de notre modèle à l'aide de la célèbre API Keras de la bibliothèque TensorFlow. Après comparaison de résultats d'évaluations de cinq architectures pré-entrainée l'architecture est VGG-16 a été retenue, cette dernière nous a permis d'atteindre un accuracy de 96% et un Loss de 12%.

Perspectives

L'objectif principal dans ce type de travail étant d'améliorer au maximum les performances du modèle construit, on propose pour ce, de :

- Construire un modèle à base de la nouvelle architecture EfficientNet. Selon la littérature cette dernière permet d'atteindre à la fois une plus grande précision et une meilleure efficacité par rapport aux CNN existants.
- Utiliser des fonctions de perte personnalisées et varier les hyperparamètres (LR, Nombre d'Epochs, Batch size, Steps, etc.) jusqu'à obtenir la meilleure combinaison.
- Effectuer plusieurs augmentations des données, car plus de données de formation on a, meilleur sera notre modèle.

.

.

BIBLIOGRAPHIE

Bibliographie

[1] https://siecledigital.fr/2019/10/24/cette-ia-aide-les-medecins-a-detecter-les-hemorragies-cerebrales-bien-plus-rapidement/

[2] J. Pierre, L. Brunelle, "Artificial intelligence and medical imaging: definition, state of the art and perspectives", DI, MSc, MRes - Department of DIRD, Arts & Métiers ParisTech - Product Management Technical, Amazon, Greater Seattle Area, USA

[3] RL. Sacco, SA. Mayer, Epidemiology of intracerebral hemorrhage. In: Feldmann E, eds. Intracerebral hemorrhage. New York: Futura Publishing Co. 1994.

[4] T. Steiner, J. Rosand, M. Diringer, Intracerebral hemorrhage associated with oral anticoagulant therapy: current practices and unresolved questions. Stroke. 2006.

[5] [Online]Available https://www.fragile.ch/fr/lesion-cerebrale/hemorragie cerebrale/

[6] JCP, Hémorragie cérébrale, Campus de Neurochirurgie, France, 2006

[7] S. Claeys, Bases physiques des rayons X - CERF 2001 - Solacroup, Boyer, Le Marec.

[8] J. Itzcovitz, D. Dormont, Scanographie à rayons X', Enseignement du DES Radiologie et Imagerie Médicale - Scanographie à rayons X, 2002.

[9] C. Lukaski. Methods for the assessment of human body composition: traditional and new. The American Journal of Clinical Nutrition, 46: 537–556, 1987.

[10] P. Hornak. The basics of MRI, 1996. URL https://www.cis.rit.edu/htbooks/mri.

[11] A. Pooley. Fundamental Physics of MR Imaging 1. Radiographics, 25(4): 1087–1099, 2005.

[12] A.Jacobs, T. S. Ibrahim, et R. Ouwerkerk. MR imaging: brief overview and emerging applications. Radiographics, 27(4): 1213–1229, 2007.

[13] R. Fink, M. Muzi, M. Peck, et K. A. Krohn. Multimodality brain tumor imaging: MR imaging, PET, and PET/MR imaging. Journal of Nuclear Medicine.

[14] H. Ye, F. Gao, Y. Yin, and al. "Precise diagnosis of intracranial hemorrhage and subtypes using a three-dimensional joint convolutional and recurrent neural network". Eur Radiol **29,** 6191–6201 (2019). https://doi.org/10.1007/s00330-019-06163-2

[15]] J. Hurwitz, D. Kirsch, "Machine Learning For Dummies", IBM Limited Edition 2018 John Wiley & Sons, Inc. 111 River St. Hoboken, NJ 07030-5774

[16] https://blogs.msdn.microsoft.com/big_data_france/2014/06/05/lapprentissage-automatique-

[17] O. Chapelle, B. Schölkopf and A. Zien. “Semi-supervised learnin. Adaptive computation and machine learning”. MIT Press, 2006A. Pooley. Fundamental Physics of MR Imaging 1. Radiographics, 25(4): 1087–1099, 2005.

[18] https://mrmint.fr/apprentissage-supervise-machine-learning

[19] I. Goodfellow, Y. Bengio, A. Courville, “Deep Learning”. MIT Press (2016)

[20] https://fr.mathworks.com/discovery/deep-learning.html

[21] A. Gibson, J. Patterson , “Deep Learning. Oreilly, Sebastopol (2017)”. ISBN 978-1491914250

[22] [Online]Availablehttps://raw.githubusercontent.com/mdasw/library/master/inform/these-philippe-poincot/chap3.pdf

[23] https://www.lebigdata.fr/reseau-de-neurones-artificiels-definition

[24] F.Sultana, A.Sufian, & P.Dutta, (2018, November). Advancements in image classification using convolutional neural network. In 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN) (pp. 122-129). IEEE.

[25] https://towardsdatascience.com/beginners-guide-to-understanding-convolutional-neural-networks-ae9ed58bb17d

[26] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” Biological Cybernetics, vol. 36, no. 4, pp. 193–202, Apr 1980. [Online]. Available: https://doi.org/10.1007/BF00344251

[27] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition,” Neural Computation, vol. 1, no. 4, pp. 541–551, Dec 1989.

[28] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, “Handwritten digit recognition with a back-propagation network,” in Advances in Neural Information Processing Systems 2, D. S. Touretzky, Ed. Morgan-Kaufmann, 1990, pp. 396–404. [Online]. Available: http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.Pdf

[29] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, pp. 2278–2324, Nov 1998.

[30] S. Lawrence, C. Lee, T, Ah. Chung, et al. Face recognition: A convolutional neural-network approach. IEEE transactions on neural networks, 1997, vol. 8, no 1, p. 98-113.

[31] W. Jun, C. Heang-Ping, Z. Chuan, et al. Computer-aided detection of breast masses: Four-view strategy for screening mammography. Medical physics, 2011, vol. 38, no 4, p. 1867-1876.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," Int. J. Comput. Vision, vol. 115, no. 3, pp. 211–252, Dec. 2015. [Online]. Available: http://dx.doi.org/10.1007/s11263-015-0816-y

[34] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in Computer Vision – ECCV 2014, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 818–833.

[35] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," CoRR, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[36] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[38] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," CoRR, vol. abs/1608.06993, 2016. [Online]. Available: http://arxiv.org/abs/1608.06993

[39] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," CoRR, vol. abs/1710.09829, 2017. [Online]. Available: http://arxiv.org/abs/1710.09829

[40] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," CoRR, vol. abs/1709.01507, 2017. [Online]. Available: http://arxiv.org/abs/ 1709.01507

[41] https://ledatascientist.com/google-colab-le-guide-ultime/

[42] https://desktop.arcgis.com/fr/arcmap/10.3/analyze/python/what-is-python-.htm

[43] M. Harrison, “Illustrated Guide to Python 3: A Complete Walkthrough of Beginning Python with Unique Illustrations Showing How Python Really Works.”, CreateSpace Independent Publishing Platform, (2017), ISBN 13 : 9781977921758

[44] S. Pattanayak, “Pro Deep Learning with TensorFlow”, Apress, (2017), 398 pp

[45] J. Moolayil, “Learn Keras for Deep Neural Networks : A Fast-Track Approach to Modern Deep Learning with Python Paperback”, Apress, (2018), ISBN-13: 978-1484242391

[46] F. Nelli, “Python Data Analytics: With Pandas, NumPy, and Matplotlib”, Apress, (2018), ISBN-13: 978-1484239124

[47] https://artificence.com/2017/09/13/introduction-a-lanalyse-exploratoire-et-la-preparation-des-donnees/

[48] F.Cailliez, J.P Pages, “Introduction à l’analyse des données” , Smash (2017)

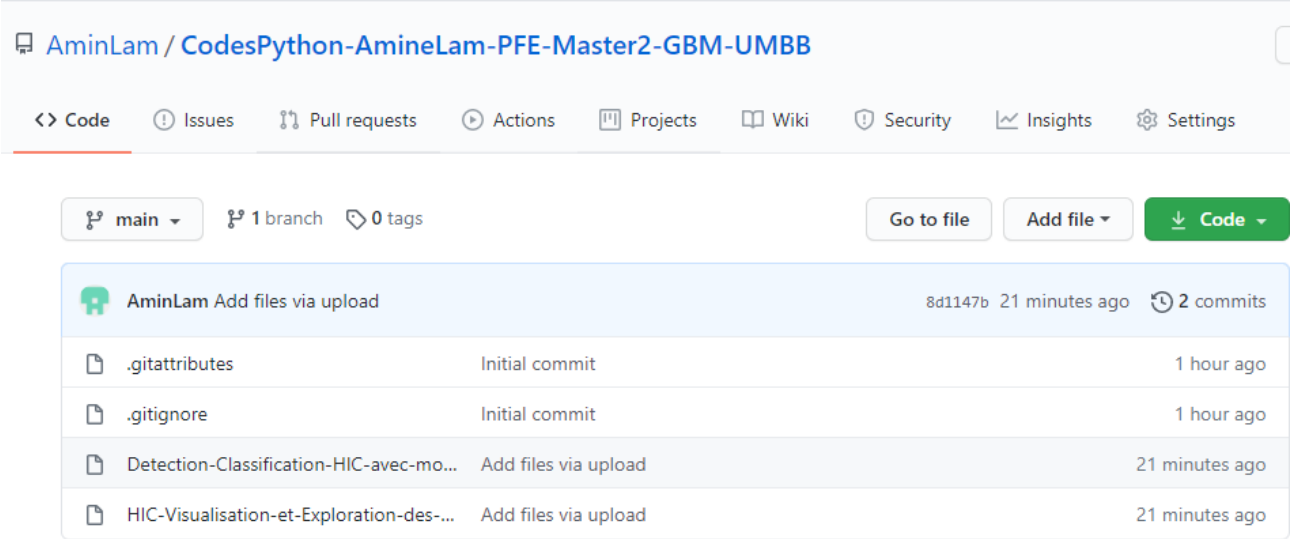
[49] https://sti-biotechnologies-pedagogie.web.ac-grenoble.fr/content/fichiers-dicom-format-dcm-en-imagerie-medicale#:~:text=en%20informations%20textuelles-,Outre%20les%20images%20num%C3%A9riques%20issues%20des%20examens%20m%C3%A9dicaux%2C%20les%20fichiers,'acquisition%2C%20le%20praticien%20etc

ANNEXES

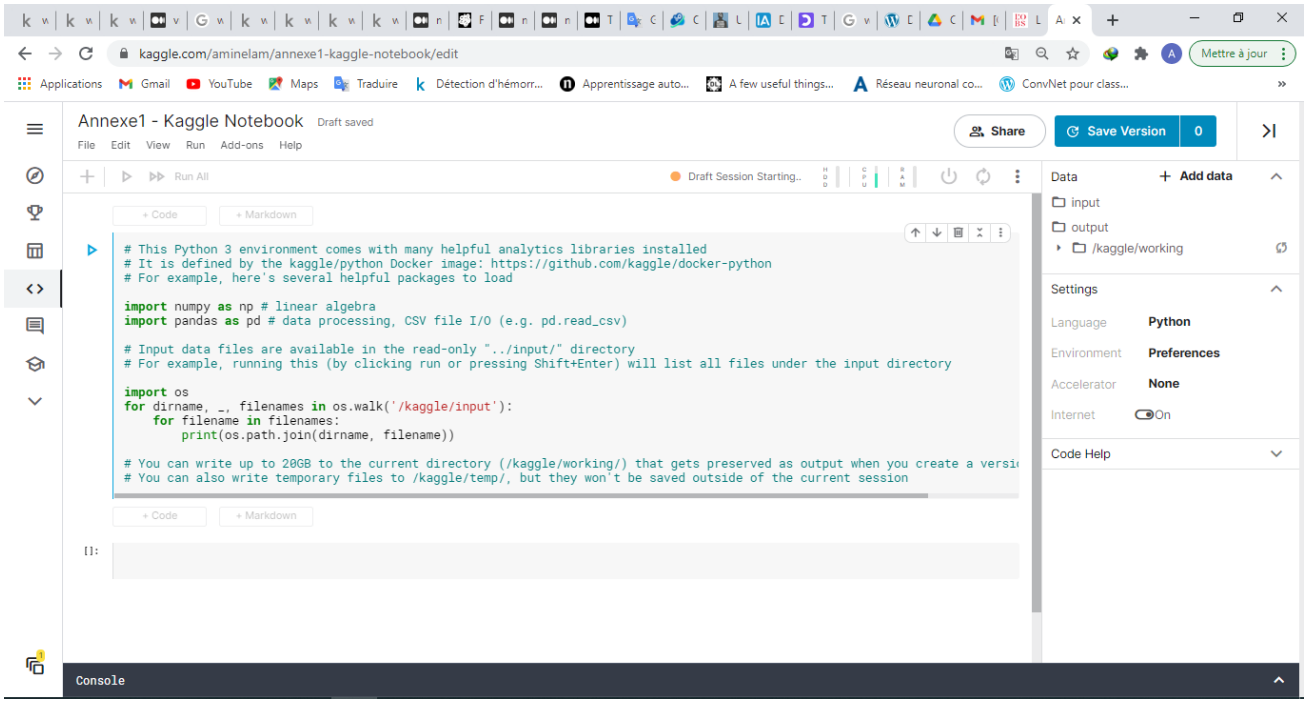
Annexes

Annexe 1 : Pour exploiter les codes que nous avons utilisé dans notre présent travail, veuillez suivre les étapes suivantes :

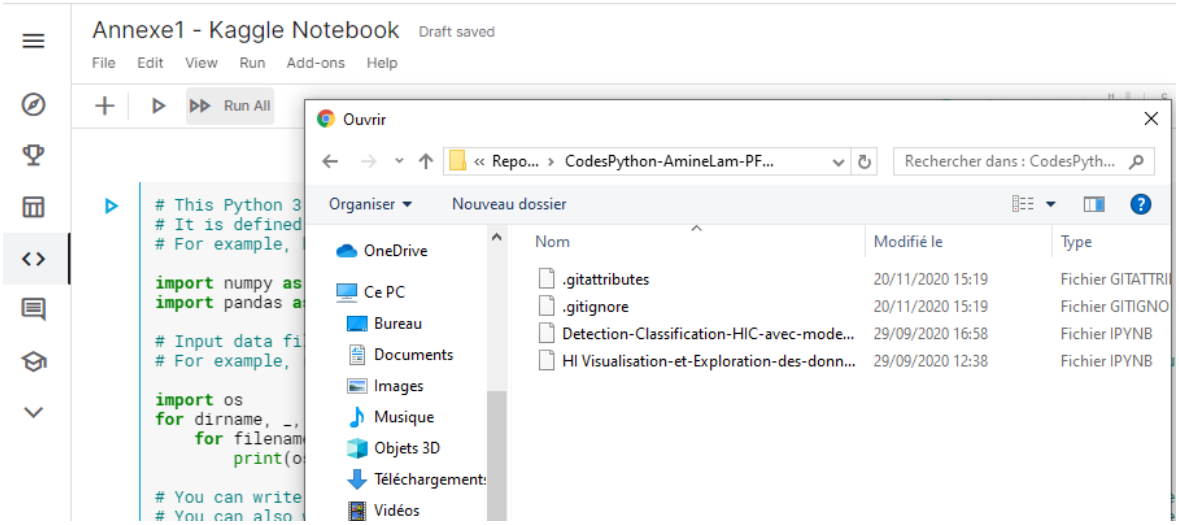
1- Télécharger les codes à partir de mon compte GitHub en utilisant ce lien : https://github.com/AminLam/CodesPython-AmineLam-PFE-Master2-GBM-UMBB.git



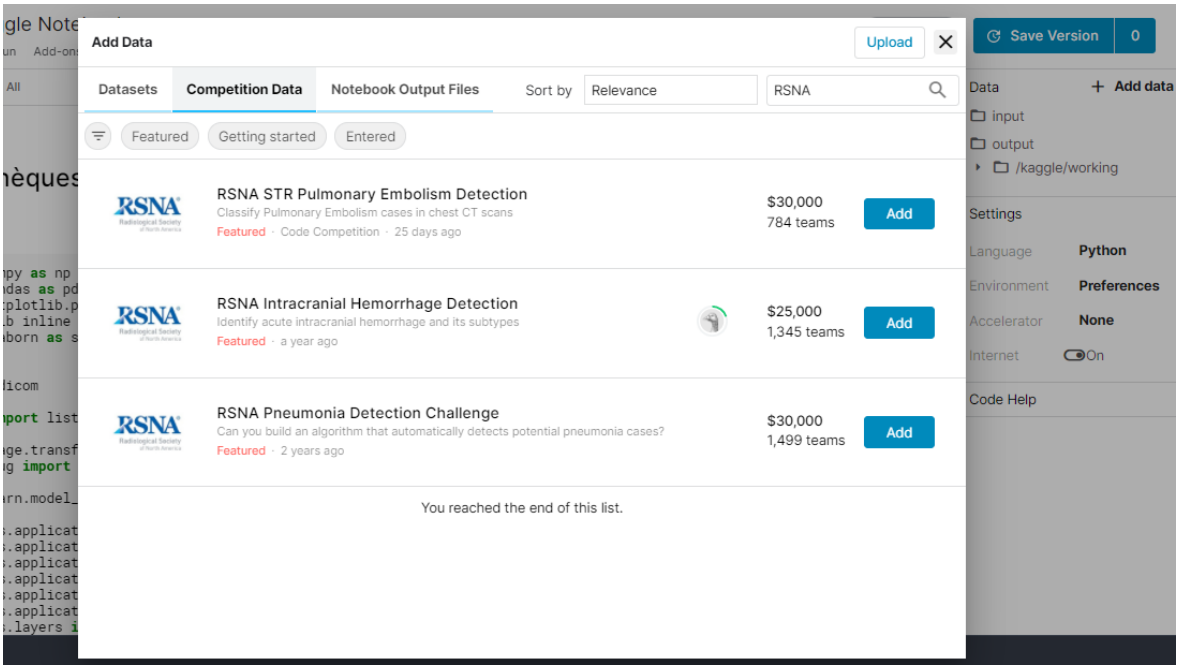
2- Ouvrir une nouvelle fenêtre Kaggle Notebook sur n'importe quel navigateur web



3- Charger dans la fenêtre Kaggle un des codes téléchargés : File > Upload > (‘Chemin du code’).



4- Enfin charger la base de données à partir de : Add data > Competition Data > RSNA Intracranial Hemorrhage Detection > Add.



Annexe 2 :

Vous pouvez télécharger la base de données que nous avons utilisée sur :

RSNA Intracranial Hemorrhage Detection | Kaggle