République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieure et de la Recherche Scientifique
Université Abderrahmane Mira de Béjaia
Faculté de Technologie

جامعة بجاية
Tasdawit n Bgayet
Université de Béjaïa

ATE Department

Département d'Automatique, Télécommunication et d'Electronique

Mémoire de Fin d'Etudes

En Vue De L'Obtention Du Diplôme de Master En Télécommunications

Option : Systèmes des Télécommunications

Thème

U-Net : Réseau Neuronal Convolutionnel pour la Segmentation des Images Biomédicales

Réalisé par :

Miss. MAIDOUCHE Ouardia
Miss. TABIA Tamazight

Soutenu le 19/ 06/ 2025, Devant le jury ci-dessous :

Président :	Mr MEKHMOUKH Abdenour
Examinateur :	Mr KASMI Reda
Encadrant :	Mrs GHENNAM Souhaila Souad

Promotion : 2024/2025

Dédicace

À la mémoire de mon père,
Que Dieu lui accorde Sa miséricorde et l'accueille dans Son vaste paradis. Son souvenir reste vivant dans mon cœur et m'accompagne à chaque étape de ma vie.

À ma mère,
L'essence même de ma vie, mon pilier, ma force et mon refuge. Merci pour ton amour, tes sacrifices et ton soutien inconditionnel.

À mes frères Syphax et Amazigh, et à ma sœur Tafath,
Merci pour votre amour, votre soutien et votre présence constante.

À l'oncle de mon amie et à sa famille,
Pour leur accueil chaleureux, leur générosité et leur bienveillance. Votre soutien a compté énormément dans mon parcours.

À mon voisin Omar Matous, à sa femme Malika, et à leurs enfants Rania et Hussein, pour leurs paroles réconfortantes et leur soutien moral.

À mes amis, Pour leur présence, leur aide, leurs encouragements et les moments partagés qui ont rendu ce chemin plus léger.

À vous tous,
Je dédie ce mémoire avec toute ma reconnaissance, mon respect et mon affection.

Tamazight

Dédicace

Au nom d'Allah, le Tout Miséricordieux, le Très Compatissant.

Ce mémoire représente l'aboutissement d'un long parcours fait d'efforts et de persévérance.

Je le dédie à mon père, modèle de générosité et de dignité, à ma mère, dont les prières sincères ont guidé mes pas, à ma sœur Rania et mon frère Seddik, piliers de mon quotidien, ainsi qu'à mes grands-parents Madjid et Louiza, symboles d'espoir et de sagesse.

Je rends également hommage à ma tante Fahima, pour son rôle essentiel dans mon orientation, à mon oncle Abdellah pour son appui concret, et à la mémoire de mes tantes Mebarka et Nacira, dont le soutien reste gravé dans mon cœur.

Mes remerciements s'étendent à tous les membres de ma famille, à mes amis, ainsi qu'à toutes les personnes ayant contribué, de près ou de loin, à la réalisation de ce travail.

Puisse Allah nous guider et nous accorder la réussite dans le bien.

Ouardia

Remerciements

Nous souhaitons, tout d'abord, exprimer notre profonde reconnaissance à Allah, le Tout-Puissant, pour nous avoir accordé la santé, la patience et la persévérance nécessaires à la réalisation de ce travail. Que Ses bénédictions et Sa guidance nous accompagnent dans chacun de nos projets futurs.

Nous tenons à adresser nos remerciements les plus respectueux à Madame GHENNAM, notre encadrante, pour la qualité de son encadrement, la clarté de ses orientations, ainsi que pour sa disponibilité et son engagement tout au long de ce mémoire. Ses remarques pertinentes et ses conseils avisés ont grandement enrichi notre réflexion et orienté notre démarche.

Nos remerciements s'adressent également aux membres du jury, pour l'intérêt qu'ils portent à notre travail, le temps qu'ils lui consacrent, ainsi que pour leurs observations constructives et éclairées qui ne manqueront pas de nourrir notre développement académique et professionnel.

Nous exprimons aussi notre gratitude à Monsieur KHELLIEL Hani et Monsieur FEKAIR Laid, nos encadrants en milieu professionnel, pour leur accueil, leur accompagnement méthodologique et technique, ainsi que pour l'environnement de travail stimulant qu'ils nous ont offert tout au long de notre stage.

Nous remercions chaleureusement nos familles, pour leur soutien moral indéfectible, leur confiance constante et leurs encouragements tout au long de notre parcours. Nos pensées reconnaissantes vont également à nos amis et collègues, pour leur présence bienveillante, leur patience et leur appui dans les différentes étapes de ce projet.

Enfin, nous adressons nos vifs remerciements à toutes les personnes, connues ou anonymes, qui ont contribué de près ou de loin à l'aboutissement de ce mémoire.

Merci du fond du cœur !

Table des matières

Liste des Figures

Liste des Tableaux

Liste des acronymes

IA	***I**ntelligence **A**rtificielle*
ML	***M**achine **L**earning (Apprentissage automatique)*
RN	***R**éseaux **N**euronaux*
DL	***D**eep **L**earning (Apprentissage profond)*
RNA	***R**éseau de **N**eurones **A**rtificiels*
SOM	***S**elf-**O**rganizing **M**aps (cartes auto-organisatrices)*
SVD	***S**ingular **V**alue **D**ecomposition (Décomposition en valeurs singulières)*
CAH	***C**lassification **A**scendante **H**iérarchique*
ReLU	***R**ectified **L**inear **U**nit (Unité Linéaire Rectifiée)*
CNN	***C**onvolutional **N**eural **N**etwork (Réseau de neurones convolutif)*
MLP	***M**ulti **L**ayer **P**erceptron (Perceptron multicouche)*
CONV	*La couche de convolution*
FC	***F**ully **C**onnected (Entièrement connecté)*
POOL	*La couche de pooling*
FCN	***F**ully **C**onvolutional **N**etworks (Réseaux entièrement convolutionnels)*
TP	***T**rue **P**ositive*
TN	***T**rue **N**egative*
FP	***F**alse **P**ositive (Faux positif)*
FN	***F**alse **N**egative (Faux négatif)*
SEN	***S**ensitivity (Sensibilité)*
SPE	***S**pecificity (Spécificité)*
ACC	***A**ccuracy (Précision)*
CPU	***C**entral **P**rocessing **U**nit (Unité centrale de traitement)*
RAM	***R**andom **A**ccess **M**emory (Mémoire vive)*
GPU	***G**raphical **P**rocessing **U**nit (Unité de traitement graphique)*
FOV	***F**ield **O**f **V**iew*

Introduction Générale

Depuis plusieurs décennies, l'intelligence artificielle (IA) n'a cessé de s'imposer comme un domaine stratégique dans la recherche scientifique et l'innovation technologique. Elle regroupe un ensemble de techniques visant à reproduire, simuler ou dépasser certains aspects de l'intelligence humaine. L'un des champs les plus prometteurs de l'IA est le Deep Learning ou apprentissage profond, une technique d'apprentissage automatique basée sur l'utilisation des réseaux de neurones. Grâce aux progrès matériels (GPU, mémoire), aux volumes massifs des données disponibles, et à la sophistication des algorithmes, le Deep Learning a atteint des performances spectaculaires dans plusieurs domaines.

Parmi les architectures les plus performantes du Deep Learning, les réseaux de neurones convolutifs (CNN) ont démontré une grande efficacité dans des tâches telles que la classification, la détection et surtout la segmentation d'image. La segmentation d'image est une étape clé dans de nombreuses applications comme la médecine, la robotique, la surveillance ou encore la conduite autonome. Elle représente un enjeu majeur pour une meilleure compréhension et exploitation des données visuelles. En particulier, dans le domaine médical, une segmentation précise des structures anatomiques peut considérablement faciliter le diagnostic, le suivi des pathologies et la planification des traitements.

Notre projet de fin d'étude s'inscrit dans cette perspective. Il a pour objectif d'aborder une architecture avancée de Deep Learning appelée U-Net, spécialement conçue pour la segmentation d'image biomédicale. Ce modèle se distingue par son architecture en forme de U qui permet une capture efficace des informations contextuelles et par ses performances qui ont été démontrées même sur des bases de données de petite taille. Toutes ces raisons ont motivé notre choix pour ce modèle pour accomplir la tâche de segmentation d'image.

Le mémoire débute par une introduction aux concepts fondamentaux de l'IA et du Deep Learning. Nous présenterons le CNN de manière exhaustive. Ensuite, nous aborderons sommairement les méthodes de segmentation classiques et les concepts récents s'y référant. Nous présenterons en détail l'architecture et le fonctionnement de l'U-Net, et nous l'appliquerons sur des images médicales, spécifiquement pour la segmentation des vaisseaux rétiniens à partir des images de la rétine.

L'enjeu de notre travail est double : d'une part, démontrer l'efficacité du modèle U-Net dans la segmentation d'image, et d'autre part, contribuer à la compréhension et à la vulgarisation de l'usage du Deep Learning dans le domaine médical.

Chapitre I

Généralités sur l'IA et le Deep Learning

I.1 Introduction

Depuis quelques années, les concepts liés à l'intelligence artificielle inondent particulièrement les articles scientifiques, et plus généralement notre quotidien. Lorsque nous parlons d'intelligence artificielle, nous faisons très souvent implicitement allusion aux technologies qui y sont associées à savoir le Machine Learning, Neural Network ou le Deep learning. Deux termes extrêmement utilisés avec des applications toujours plus nombreuses, mais généralement pas toujours bien définis. Nous nous attelons donc dans ce chapitre d'abord à définir et à distinguer ces trois concepts, puis nous nous attarderons sur le Deep Learning objet de ce chapitre.

I.2 Quelques définitions

I.2.1 Intelligence artificielle (IA)

L'Intelligence Artificielle est un champ de recherche qui regroupe l'ensemble des techniques et méthodes qui tendent à comprendre et à reproduire le fonctionnement du cerveau humain. Elle vise à la conception de machines intelligentes qui peuvent penser par elles-mêmes et prendre leur propre décision afin d'imiter le comportement ou l'intelligence humaine [1]. Marvin Minsky a défini l'intelligence artificielle comme étant « une science dont le but est de faire réaliser par une machine des tâches que l'homme accomplit en utilisant son intelligence » [2]. L'intelligence artificielle trouve application dans la résolution de problèmes à haute complexité logique ou algorithmique.

I.2.2 Machine Learning (ML)

Le Machine Learning, ou l'appellation francisé Apprentissage Automatique, est un sous-domaine de l'IA (figure I.1) qui consiste à donner la capacité aux machines (ou systèmes) d'apprendre automatiquement et de prendre des décisions à partir des données, mais aussi d'améliorer leurs performances sur une tâche spécifique sans être explicitement programmés [3]. Contrairement à la programmation qui consiste en l'exécution de règles prédéterminées. Les algorithmes du ML se basent principalement sur les calculs statistiques.

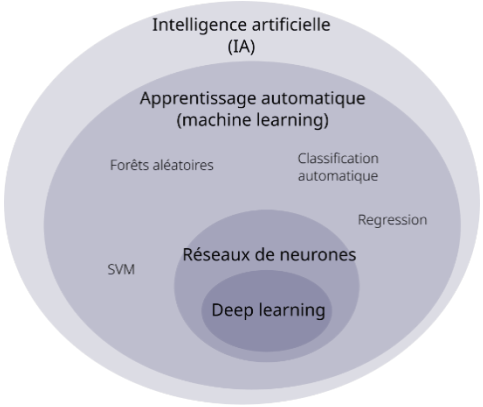


Figure I.1 – Relation entre IA, ML, RN et DL [4]

L'objectif principal qui était d'imiter l'intelligence humaine, s'est vu ralenti dans un premier temps en raison des limites de la théorie et de la technologie qui existaient. Machine Learning,

s'est vu donc réduite sur des tâches spécifiques, ainsi la plupart des algorithmes ML tels qu'ils existent aujourd'hui se concentrent sur l'optimisation des fonctions, et leur utilisation devient fréquemment un processus répétitif d'essais et d'erreurs, dans lequel le choix de l'algorithme parmi les problèmes donne des résultats de performances différents [5][6].

I.2.3 Réseaux Neuronaux

Les réseaux neuronaux artificiels RNA sont une catégorie du ML qui établissent la tâche d'apprentissage automatique en mimant les actions du cerveau humain. Tout comme le cerveau ces réseaux sont constitués de neurones artificiels interconnectés les uns aux autres et sont disposés en plusieurs couches (entrée, cachée, sortie). Ces neurones reçoivent des informations en entrée qui sont traitées et transmises de couche en couche jusqu'à générer des sorties et cela sans règles programmées, car essentiellement, un réseau neuronal résout les problèmes par essais et erreurs [6].

Les réseaux de neurones représentent une approche clé au sein de l'IA et du ML, largement utilisée aujourd'hui dans des domaines variés. Leur structure a servi de fondement au développement du deep learning [5].

I.2.4 Deep Learning

Le Deep Learning ou l'apprentissage profond, une technique du ML et de l'IA qui repose sur le modèle des réseaux neurones. C'est en effet une extension plus profonde et sophistiquée des réseaux neuronaux artificiels, composés de dizaines voire des centaines de couches de neurones, ce qui confère le caractère « profond » ou « deep » à l'apprentissage via ces réseaux neuronaux, dits "réseaux de neurones profonds". Plus il y'a de couches, plus l'apprentissage est profond. Cette extension a été rendue possible par les avancées technologiques récentes, dont deux faits particulièrement importants :

— L'émergence de grands volumes de données (big data) issus d'internet,

— Et la disponibilité d'une forte puissance de calcul rendant possible l'estimation de millions de paramètres lors de traitement de dizaines, voire, de centaines de couches de neurones aux propriétés complexes et variées. Le DL a la particularité d'être gourmand en puissance et en données.

Le Deep Learning est donc une sorte de sous-catégorie avancée du Machine Learning et qui contrairement à ce dernier, il n'a pas besoin d'aide humaine pour travailler avec de grandes quantités de données non structurées, car il peut détecter lui-même des représentations ou des fonctionnalités. De plus, plus il traite d'informations, plus les résultats qu'il propose sont raffinés [5].

I.3 Types d'apprentissage pour ML, RN et DL

L'apprentissage est un processus qui permet à un système à base d'un modèle ML, RNA ou DL de modifier son comportement en fonction des données qu'il reçoit dans le but d'améliorer ses performances sur une taches spécifique. La modification du comportement se fait en ajustant les paramètres du modèle pour minimiser l'erreur entre les prédictions et les valeurs réelles [3].

On distingue deux types d'apprentissage : Apprentissage supervisé et Apprentissage non supervisé :

I.3.1 Apprentissage Supervisé

En apprentissage supervisé, l'algorithme est guidé avec des connaissances préalables de ce que devraient être les valeurs de sortie du modèle. Par conséquent, le modèle ajuste ses paramètres de façon à diminuer l'écart entre les résultats obtenus et les résultats attendus. La marge d'erreur se réduit ainsi au fil des entraînements du modèle, afin d'être capable de l'appliquer à de nouveaux cas.

Parmi les algorithmes utilisés dans l'apprentissage supervisé, on site :

— Les Algorithmes de Régression : Gradient Descent, Arbre de decision[7].

— Algorithmes de Classification : Naives Bayes, K plus proches voisins (K-NN), Support Vector Machine [7].

I.3.2 Apprentissage Non-Supervisé

En revanche, dans l'apprentissage non supervisé le modèle ne dispose pas les valeurs de sortie possibles du modèle, ce qu'on appelle les « données étiquetées ». Il est alors impossible à l'algorithme de calculer de façon certaine une erreur ou un score de réussite. L'algorithme doit donc découvrir par lui-même la structure sous-jacente des données, et de donc de déduire les regroupements présents dans ces données. Il existe deux principaux domaines de modèles dans l'apprentissage non-supervisées pour retrouver les regroupements :

- Les méthodes par partitionnement, tels que les algorithmes des k-means, les cartes auto-organisatrices SOM tel que le réseau neuronal de Kohnen, et la décomposition en valeurs singulières (SVD)[7].
- Les méthodes de regroupement hiérarchique, telle que la classification ascendante hiérarchique (CAH), le regroupement par le voisinage le plus proche [7].

I.4 Les réseaux de neurones

I.4.1 Neurone Artificiel

Comme nous l'avions précédemment dit, un RNA se compose d'au moins deux couches, chacune contenant plusieurs neurones dits également nœuds. D'une couche à l'autre, les neurones sont liés entre eux. En s'appuyant sur la figure I.2, à chaque neurone sont associées des données d'entrée $(X_1, X_2, X_3, \ldots, X_i, \ldots, X_N)$, un poids, un seuil b_j et une sortie y_j, dont la valeur est le résultat de la somme de ses entrées multipliées chacune par le poids de celle-ci W_{ji}. Cette somme pondérée est appliquée à une fonction d'activation à seuil : si la somme dépasse une certaine valeur, la sortie du neurone est 1, sinon elle vaut 0. Ce neurone activé envoie sa sortie aux neurones de la couche suivante, et ainsi de suite [6].

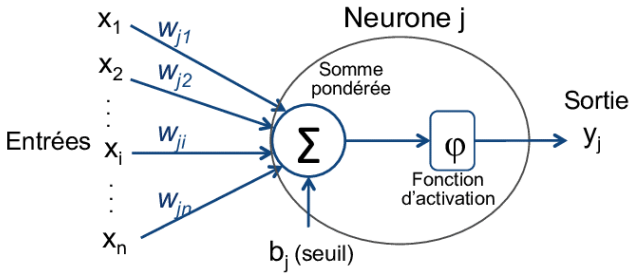


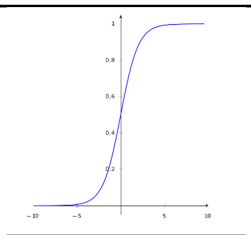
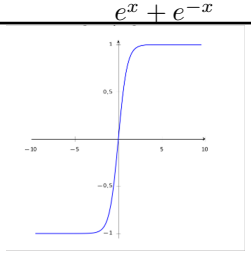
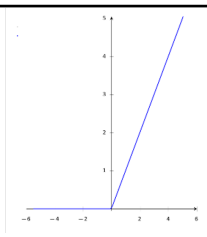
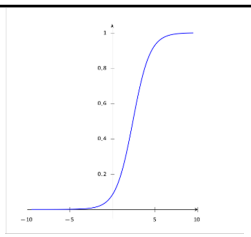
FIGURE I.2 – Neurone artificiel [8]

Au final l'expression de la sortie est :

$$y_j = \varphi\left(\sum_{i=1}^{n} W_{ji} X_i + b_j\right) \tag{I.1}$$

Il est important de signaler que la fonction d'activation est indispensable pour introduire de la non-linéarité et permettre l'apprentissage dans les réseaux profonds. Sans elle, le réseau serait équivalent à un modèle linéaire, même avec plusieurs couches, et ne pourrait pas s'adapter efficacement. Parmi les fonctions d'activation les plus utilisées, rapportées dans le tableau I.1, on retrouve : la sigmoïde logistique, la tangente hyperbolique, ReLU et Softmax [6].

TABLE I.1 – Exemples des fonctions d'activations [9].

Fonction sigmoïde	**Fonction Tangente**	**Fonction Relu**	**Fonction Softmax**
$\varphi(x) = \frac{1}{1+e^{-x}}$	$\varphi(x) = \tanh(x) = = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$	$\varphi(x) = \max(x, 0)$	$\varphi(x) = \frac{e^{x_i}}{\sum_j e^{x_j}}$
			

I.4.2 Types des réseaux de neurones

Selon l'architecture et le mode de circulation des données, il existe deux types essentiels de réseaux de neurones :

— Réseaux de Neurones Feedforward : transmission directe des données.

— Réseaux de Neurones Récurrents : réutilisation des résultats internes [10].

I.4.2.1 Les réseaux de neurones feed-forward

Feed-forward veut dire propagation-avant, ce qui signifie que la donnée traverse le réseau d'entrée à la sortie, dans une seule direction et sans retour en arrière de l'information. Typiquement, dans la famille des réseaux à propagation-avant (Feed-forward), on distingue les réseaux monocouches dits perceptrons simples et les réseaux multicouches appelés perceptrons

multicouche. Le premier contient une seule couche cachée et le deuxième en contient plusieurs, en plus de la couche d'entrée et de la couche de sortie[10].

(a) (b)

FIGURE I.3 – Réseau de Neurones FeedForward (a) monocouche et (b) multicouches [10]

C'est à base de cette architecture avec une certaine sophistication que sont construits les réseaux neuronaux profonds comme le réseau neuronal à convolution (Convolutional Neural Network CNN) et qui trouvent une grande émergence et grands intérêts. Ce réseau neuronal fera l'objet du chapitre suivant.

I.4.2.2 Les réseaux de neurones récurrents

Les réseaux récurrents ou RNN pour Recurrent Neural Network, sont des réseaux de neurones dans lesquels l'information peut se propager dans les deux sens, de l'entrée vers la sortie, ou de la sortie vers l'entrée, et y compris des couches profondes aux premières couches. En cela, ces réseaux sont plus proches du vrai fonctionnement du système nerveux, qui n'est pas à sens unique[11].

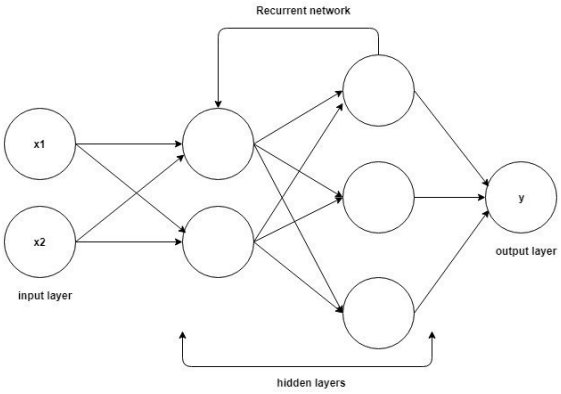


FIGURE I.4 – Réseau de neurones récurrent [11]

Grace aux connexions récurrentes qui conserve à un instant t un certain nombre d'états passés, on dit que les RNNs mémorisent des informations des étapes précédentes. Cela les rend adaptés pour des tâches comme la reconnaissance vocale car ils peuvent identifier des mots en tenant compte des sons précédents, la traduction automatique ou l'analyse de texte car ils établissent des relations entre les mots déjà analysés [11].

I.5 Conclusion

Ce chapitre était l'occasion de voir les différentes catégories de l'intelligence artificielle, d'introduire particulièrement le deep learning qui trouve de plus en plus intérêt dans la conception des applications IA, et d'aborder les réseaux neuronaux qui sont la base du DL. Nous pourrons maintenant aborder le prochain chapitre qui sera consacré à un réseau neuronal type du DL, en l'occurrence le réseau neuronal à convolution (Convolutional Neural Network CNN).

Chapitre II

Le réseau neuronal convolutif CNN

II.1 Introduction

Dans le cas de traitement d'informations plus complexes et très variés, la création de réseaux de neurones spécialisés doit-être envisagée. Les Convolutional Neural Network (CNN) ou Réseaux de Neurones Convolutifs en français sont conçus pour répondre à ces exigences. Ce sont les modèles les plus performants du Deep Learning, ils sont puissants et sont principalement utilisés pour traiter des données structurées en grille, comme les images, ce qui les rend extrêmement efficaces dans le domaine du traitement d'images, notamment dans la reconnaissance faciale et la classification des images. Ce chapitre sera consacré au CNN.

II.2 Avant propos sur les CNN

Le CNN désigne une sous-catégorie des réseaux de neurones, dont l'un des principaux usages est la classification d'image, d'ailleurs à ce jour un des modèles réputés être les plus performants. L'autre usage est le traitement naturel du langage, du fait que les CNN sont très efficaces pour l'analyse sémantique, la modélisation de phrase, la classification ou la traduction. Dans une moindre mesure, les CNN sont utilisés aussi pour l'analyse vidéo.

Par ailleurs, le CNN reçoit en entrée une image sous la forme d'une matrice de pixels. Celle-ci dispose de 3 dimensions :

— Deux dimensions désignant la taille de l'image, sa largueur et sa hauteur. La largeur est associée aux nombre de colonnes N et la hauteur aux nombres des lignes M,

— Une troisième dimension, qui correspond à la profondeur de la matrice image,

— qui est de 1 pour une image à niveau de gris,

— ou de 3 pour une image couleur. Chacune des couleurs fondamentales : Rouge, Vert, Bleu, est associé à une matrice de taille M×N [6][12].

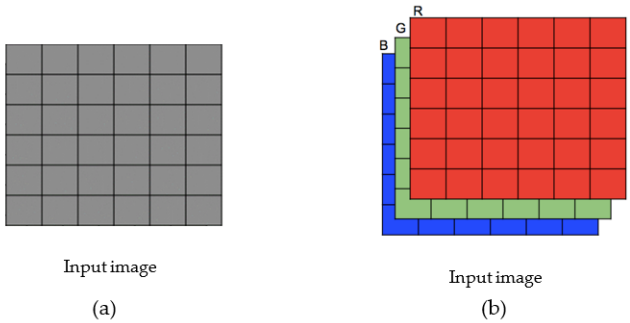


FIGURE II.1 – Entrée d'un CNN, (a) image à niveau de gris et (b) image en couleur [13].

II.3 Architecture d'un Convolutional Neural Network-CNN

Contrairement à un modèle à multicouche MLP (Multi Layers Perceptron) classique, constitué d'un ensemble de couches de neurones et dont la tâche principale et la classification, l'architecture du Convolutional Neural Network dispose également d'un MLP pour la classification

qu'on désignera « partie de classification », et en amont de laquelle se trouve « une partie convolutive », composée d'une multitude de couches de neurones, et qui fait la spécificité principale du CNN. Ce dernier comporte par conséquent deux parties bien distinctes :

- **Une partie convolutive :** dont l'objectif final est d'extraire les caractéristiques propres à chaque image. Cette extraction se fait par une succession de convolution avec des filtres ou noyaux de convolution, créant des nouvelles images appelées cartes de convolution. Chaque convolution est suivie par une compression de façon à réduire au fur et à mesure la taille des cartes. Au bout de cette partie, les cartes de convolutions obtenues sont concaténées dans un vecteur de caractéristiques appelé code CNN appelé également vecteur de caractéristiques.
- **Une partie classification :** Le code CNN obtenu en sortie de la partie convolutive est fourni en entrée d'une deuxième partie, qui est un perceptron multicouches MLP constituée de couches entièrement connectées, et dont le rôle est de combiner les caractéristiques du code CNN afin de classer l'image[12].

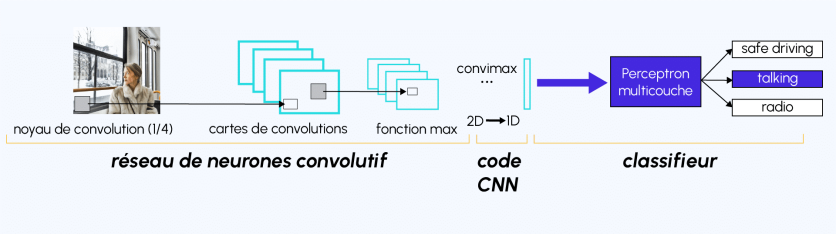


FIGURE II.2 – Schéma représentant l'architecture d'un CNN [12].

II.3.1 Partie convolutive

II.3.1.1 La convolution

La convolution est une opération mathématique dont l'effet sur une image s'assimile à un filtrage. En effet le filtrage d'une image consiste à la convolution de celle-ci avec un filtre. Le filtre en question généralement approxime la caractéristique pertinente (feature) de l'image qu'on veut extraire ou accentuer. Il existe des filtres de convolution fréquemment utilisés et permettant d'extraire des caractéristiques plus pertinentes comme la détection des bords (filtre dérivateur) ou la détection des forme géométriques. Le choix des filtres par ailleurs se fait automatiquement par le modèle CNN [6].

On explique dans la suite comment s'effectue la convolution dans le CNN :

— Dans un premier temps, on a la taille de la fenêtre du filtre, étant le filtre est définit automatiquement,

— La fenêtre du filtre, se déplace progressivement dans l'image de gauche vers droite et de haut en bas jusqu'à arriver au bout de l'image, comme illustrée sur la figure II.3. Le déplacement de la fenêtre se fait avec un certain pas dit « stride » qui peut être différent de 1, égale à un certain nombre de pixels. Ce pas est défini au préalable,

— À chaque portion d'image rencontrée, un calcul de convolution s'effectue. Plus cette portion ressemble au filtre, plus le résultat de la convolution est élevé. Une fois toute l'image

est, en sortie on obtient une carte de convolution où sont localisées les caractéristiques pertinentes dans l'image[12].

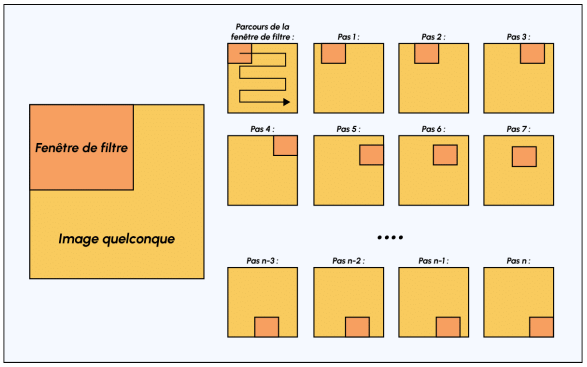


FIGURE II.3 – Schéma du parcours de la fenêtre de filtre sur l'image [12].

La convolution d'une image I de taille (M×N) avec un filtre h de taille (m×n), s'exprime par la relation suivante :

$$(I * h)_{(i,j)} = \sum_{k=-\frac{n-1}{2}}^{\frac{n-1}{2}} \sum_{l=-\frac{m-1}{2}}^{\frac{m-1}{2}} h(k,l) \cdot I(i-k, j-l) \tag{II.1}$$

Où pour chaque pixel I(i,j) avec i=1..M et j=1..N, on considère une portion de l'image de même taille que le filtre [m,n] et dont il est le centre. On fait la multiplication de tous les pixels de cette portion avec les éléments du filtre, puis on somme les produits obtenus pour ainsi résulter au produit de la convolution [6]. Cette explication est illustrée dans la figure II.4.

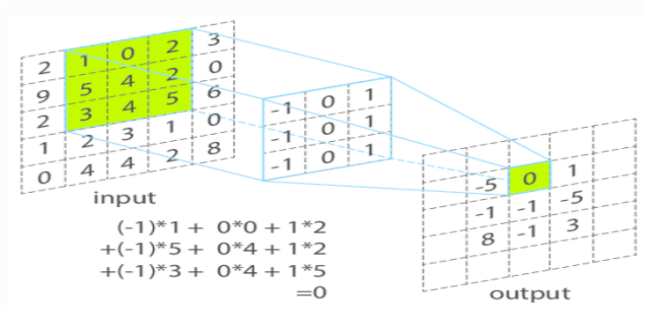


FIGURE II.4 – Calcul de la convolution [14].

Pendant la convolution, on est confronté au problème des bords. Pour pourvoir effectuer la convolution sur les pixels qui se trouvent sur les bords, deux solutions s'offrent à nous :

Remplir de zéros, sur $\frac{m-1}{2}$ lignes et $\frac{n-1}{2}$ colonnes, de part et d'autre de l'image. C'est ce qu'on appelle le « zero-padding ».

Ou dupliquer les pixels du bord, sur $\frac{m-1}{2}$ lignes et $\frac{n-1}{2}$ colonnes, de part et d'autre de l'image. Ceci est appelé effet miroir, et c'est la solution la plus adoptée.

La taille de l'image résultant de la convolution est $(N+n-1 \times M+m-1)$, si on considère le stride noté S, elle serait de [6] :

$$\left(\frac{N+n-1}{S} \times \frac{M+m-1}{S}\right)$$

.

II.3.1.2 Pooling

Le Pooling est une opération de sous-échantillonnage, elle est placée entre les couches de convolution, et son objectif est de réduire la dimension des couches de convolution au fur et à mesure du traitement. Le sous-échantillonnage est réalisé en remplaçant un bloc de pixels par une valeur unique selon un certain critère . Trois types de Pooling existent et sont présentées en figure II.5 où une entrée de taille (4×4) est réduite donc à (2×2), avec une fenêtre-pooling de taille (2×2) et un stride de 2, et au bout toute région de taille (2×2) est réduite à une seule valeur selon le type de Pooling utilisé :

- **Max pooling :**extrait la valeur maximale de la région.
- **Mean pooling :** calcule la moyenne des pixels de la région.
- **Sum pooling :** additionne les valeurs des pixels de la région sans diviser par leur nombre [15].

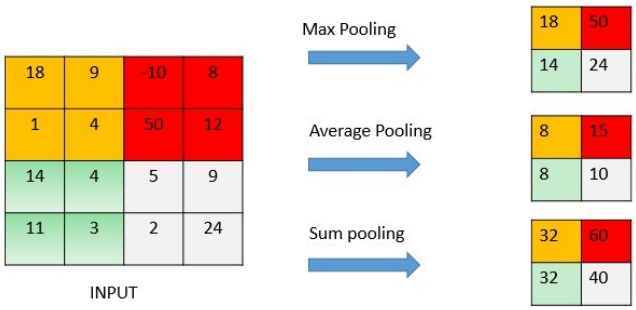


FIGURE II.5 – Différents types de Pooling, avec une fenêtre-Pooling de taille (2×2) et un stride de 2 [15].

La figure II.6 illustre l'effet du Pooling sur la couche de convolution dont la taille s'est réduite de moitié pour une fenêtre de taille (2×2). Généralement, pour une fenêtre-Pooling de taille (n×n), la taille de la couche de convolution est réduite de n fois.

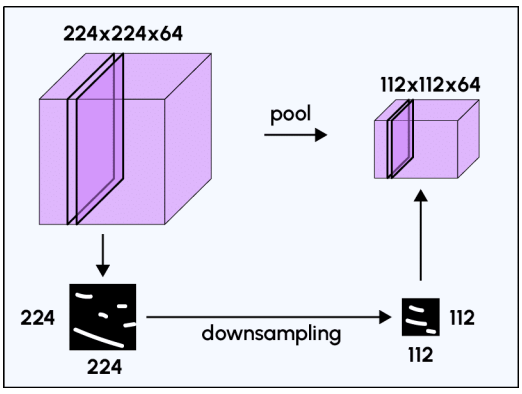


FIGURE II.6 – Effet du Pooling [12].

De plus, de l'intérêt du Pooling qui est de réduire le nombre de paramètres à apprendre, il réduit ainsi le coût de calcul, et il fournit aussi une invariance par petites translations, si une petite translation ne modifie pas la moyenne, le maximum ou la somme de la région balayée, la

moyenne ou le maximum ou le somme de chaque région resteront les mêmes et donc la nouvelle matrice créée restera identique.

Ceci dit au bout d'un nombre d'opérations de convolution et de pooling, on résulte à une carte de taille très réduite par rapport à l'image initiale, par contre de profondeur bien plus élevée. Cette carte comprend les caractéristiques les plus pertinentes de l'image. Elle sera ensuite concaténée en un vecteur (flatten vector) qu'on appelle « vecteur caractéristique » ou encore « le code CNN »[12].

II.3.2 Partie Classification

Après la partie convolutive d'un CNN, vient la partie classification qui se base sur le modèle du réseau de neurones classique feedforword, à savoir le modèle perceptron multicouche (MLP) vu en chapitre 1 et comme illustré en figure II.7. Souvent dans les CNN, le MLP utilisé à une ou deux couches cachées. Comme étant les neurones dans le MLP sont entièrement connectés, cette partie de classification est appelée « Couche entièrement connectée » au bien « Fully Connected Layer ». Elle reçoit en entrée le code CNN et pour objectif d'en attribuer une étiquette décrivant sa classe d'appartenance. Pour la mise à jour des poids du MLP, il est souvent utilisé le célèbre algorithme de descente de gradient [15].

En effet, le MLP, vu son fonctionnement, applique successivement une combinaison linéaire puis une fonction d'activation afin de classifier le vecteur code CNN à son entrée, et du moment que ce vecteur correspond aux caractéristiques pertinentes de l'image, il classifie en fait l'image à l'entrée du CNN. Le MLP renvoie ainsi en sa sortie un vecteur de taille correspondant au nombre de classes dans lequel chaque composante représente la probabilité d'appartenance de l'image à une classe [15][16].

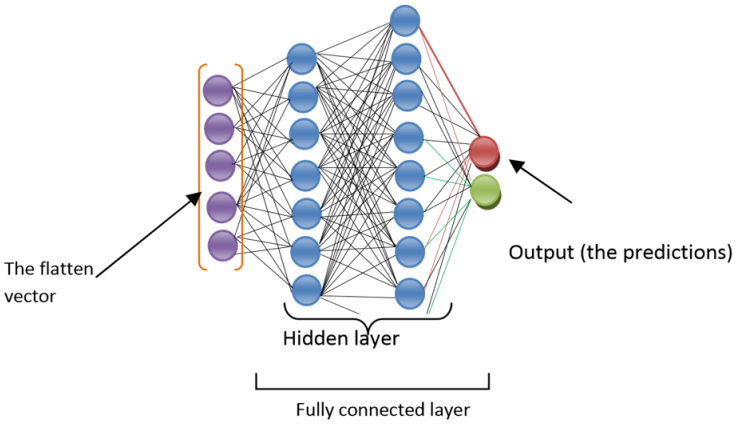


FIGURE II.7 – Couche entièrement connectée de la partie classification [16].

II.3.3 Architecture du CNN

Cette partie a pour but de synthétiser les étapes précédentes pour présenter l'architecture totale du CNN avec ses différentes couches. Eventuellement quelques couches de mise au forme sont nécessaires, qui seront ajoutées et expliquées.

Un Convolutional Neural Network est généralement constitué de :

1. Couche de convolution (CONV) : dont le but est d'extraire les caractéristiques pertinentes de l'image [16].

2. Couche d'activation ReLU (Rectified Linear Units) : Cette couche vient après chaque couche de convolution et a pour rôle de remettre à zéro toutes les valeurs négatives reçues en entrée et résultant de la convolution comme illustré par la figure II.8 . La fonction d'activation Relu associe à chaque valeur x, la fonction f(x)= max (0,x), comme ça a été vu en chapitre 1. L'intérêt de ces couches d'activation est de rendre le modèle non linéaire et de ce fait plus complexe [16].

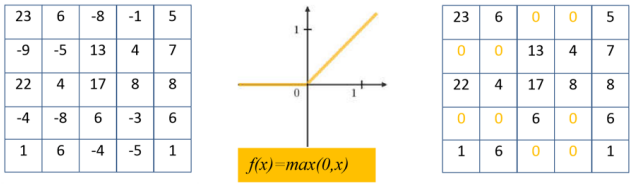


FIGURE II.8 – effet de la fonction d'activation ReLU [16].

3. Couche de Pooling (POOL) : dont le but est de comprimer la taille des images ou des couches [15].

4. Couche Fully Connected (FC) : constitué d'un MLP à une ou deux couches cachées et dont le rôle est la classification [15].

Toutes ces couches sont combinées bout à bout pour former le CNN représenté dans la figure II.9.

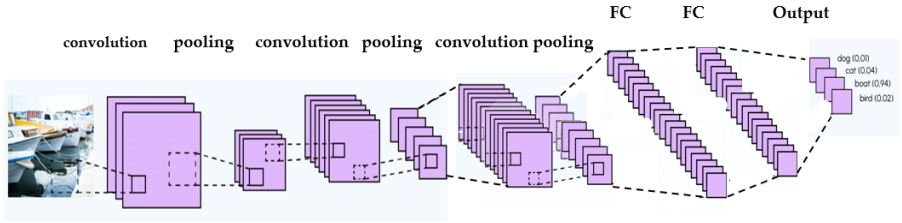


FIGURE II.9 – Architecture du CNN [12].

II.4 Paramètres du CNN et Ajustement

Le bon paramétrage d'un CNN est essentiel pour optimiser ses performances. En plus des paramètres standards des réseaux MLP, comme le nombre des couches internes, le nombre des neurones par couches, les CNN nécessitent des réglages spécifiques tels que :

— **Le nombre de filtres,** souvent réduit dans les premières couches et plus élevé dans les couches profondes,

— **La taille des filtres,** qui varie selon la nature des données. Exemple, (5×5) pour des images simples comme jusqu'à (15×15) pour des images plus complexes,

— **La configuration du pooling,** généralement en (2×2), bien que des tailles plus grandes soient possibles, avec un risque de perte d'information si la réduction est trop agressive [16].

II.5 Avantages et Inconvénients des CNN :

TABLE II.1 – Avantages et inconvénients des CNN

Les avantages	Les inconvénients
Les CNN apprennent automatiquement à partir des pixels bruts sans nécessiter de prétraitement, ce qui simplifie l'entrée et améliore les performances [17].	Les CNN nécessitent un grand volume de données étiquetées pour être efficaces, ce qui rend leur entraînement coûteux et chronophage [17].
Ils exploitent la structure spatiale des images pour construire des représentations hiérarchiques et généralisables [17].	Ils sont sensibles au surapprentissage et nécessitent des techniques de régularisation, ce qui augmente la complexité du modèle [17].
Les CNN sont particulièrement performants pour traiter des données visuelles comme les images et les vidéos [18].	Leur fonctionnement difficile à interpréter en fait des « boîtes noires », limitant leur utilisation dans des domaines critiques [17].

II.6 CONCLUSION

Ce chapitre a été consacré au CNN, détaillant son architecture et son fonctionnement. Ces différentes couches ont été largement décrites en termes de structure et de cheminement des données et d'opérations que subit l'image brut en entrée jusqu'à aboutir à sa classe en sortie. La principale application du CNN est la classification, et c'est ce qui l'a rendu célèbre du fait qu'il réalise les meilleures performances que tous les autres algorithmes existants.

Dans la communauté scientifique on s'est penché sur l'extension de ce réseau neuronal vers la segmentation proprement dit. Au lieu de résulter vers une classe, on résulte sur une image où les points sont répartis en plusieurs classes chacune représente une région de l'image à segmenter. C'est le défi relevé par l'U-Net et qui fera l'objet du chapitre 4, étant dans le chapitre 3 on donnera un état de l'art sur la segmentation.

Chapitre III

La Segmentation d'Image

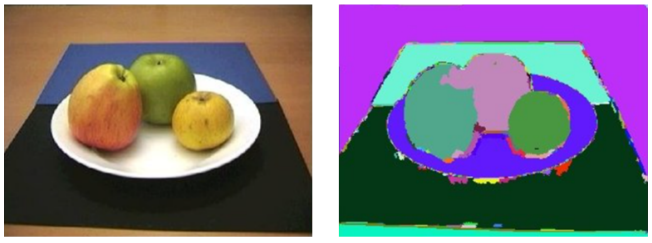
III.1 Introduction

La segmentation d'images est un domaine clé du traitement d'image et dernièrement de l'apprentissage profond qui lui a conféré d'autres concepts. La segmentation d'images trouve application dans nombreux domaines et suscite toujours de l'intérêt pour créer de nouveaux algorithmes avec plus de précision et efficacité. Parmi les domaines d'application, on cite l'aide au diagnostic en imagerie médicale, la conduite autonome et les voitures sans conducteur, l'automatisation de la locomotion en robotique, le suivi d'objet dans la surveillance, et l'identification d'objets d'intérêt dans les images satellite.

Dans ce chapitre on rappellera les méthodes de segmentation d'images classiques, et on présentera brièvement celles basées sur l'apprentissage profond.

III.2 Définition

La segmentation d'images consiste à diviser une image en plusieurs régions homogènes et distinctes. L'homogénéité est définie selon des critères, tels que des similarités entre les pixels par rapport au niveau du gris, à la couleur, à la textures, ... Souvent il y'a une labélisation dite également étiquetage qui vise à annoter aux pixels de chaque région une même étiquette [19].



(a) (b)

FIGURE III.1 – (a) image originale, (b) image segmentée selon le critère de couleur[20].

III.3 Les approches de la segmentation

Les approches de segmentation peuvent être classées en trois importantes catégorie :

— Approche contour,

— Approche région,

— Approche classification des pixels.

III.3.1 Segmentation d'image fondée sur les contours :

Un contour est défini comme la frontière séparant deux régions homogènes adjacentes d'une image . Ces méthodes de segmentation visent d'abord à détecter les frontières entre les régions, autrement dit, à détecter les contours des régions. Une fois que la région est délimitée, les pixels lui appartenant sont affiliés à une même étiquette. La fermeture du contour est une condition essentielle pour la réussite de la segmentation.

Parmi les détecteurs de contours, on site les filtres dérivateurs de Sobel, Prewitt, Roberts ou Canny-Deriche. Toutefois, ces filtres produisent souvent des contours non fermés, entrecoupés

ou bruités, limitant les performances de ces méthodes . Les contours actifs dits également modèles déformables, ou encore contours déformables, pallient ces limitent en assurant à coup sûr un contour fermés.

Un contour actif consiste en une courbe qui se déforme sous l'action de forces basées sur l'énergie interne inhérente au contour (son élasticité, sa courbure, ...) et l'énergie externe définie à partir de l'image. La déformation se fait sous la minimisation du total de ces énergies le long de cette courbe. La figure III.2 illustre la déformation du contour actif, sur l'image finale on voit la délimitation des régions (objet + arrière-plan), il s'en suivra l'opération d'étiquetage pour une achever le segmentation [19].

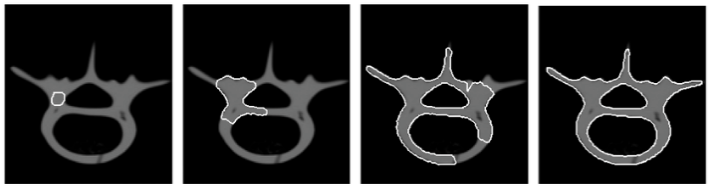


FIGURE III.2 – Déformation du contour actif[21].

III.3.2 Segmentation fondée sur les régions :

Ces méthodes considèrent les pixels au milieu de leurs voisinages, et les régions homogènes sont formées par des groupes de pixels connexes partageant les mêmes critères d'homogénéité spécifiés[19]. On distingue deux principales méthodes :

— La méthode croissance de région,

— La méthode division/fusion.

III.3.2.1 Méthode de croissance de région :

La segmentation par croissance de régions est une méthode ascendante. La croissance à partir d'un pixel initial appelé « germe », il peut être choisi aléatoirement ou manuellement. C'est le point de départ de la région à former. Les pixels connexes au germe respectant les critères d'homogénéité sont ajoutés à la région, ainsi la région croit, jusqu'à ce qu'il n'y ait plus de pixels aux voisinages qui vérifient les critères d'homogénéité (voir figure III.3). Ainsi de suite jusqu'à ce que toutes les régions sont détectées ou seules les régions d'intérêt sont détectées[19].

FIGURE III.3 – Principe de la méthode de Croissance de régions[20].

III.3.2.2 Méthode division/fusion :

C'est une méthode descendante et ascendante à la fois. Dans la première étape, d'abord l'image ensuite les régions obtenues, subissent une succession de division, généralement en

quatre, tant qu'un critère d'homogénéité n'est pas respectée, en l'occurrence une variance supérieure à un certain seuil. Une fois que se critère est respectée la division s'arrête, aboutissant à une image souvent sur-segmentée, été dont plusieurs régions sont homogènes entre elles et peuvent former une seule région. C'est là qu'intervient l'étape de fusion qui rassemble les régions connexes et similaires en une seule. Cette tâche est répétée jusqu'à ce qu'il n'y ait plus de régions connexes similaires[19], comme illustré sur la figure III.4.

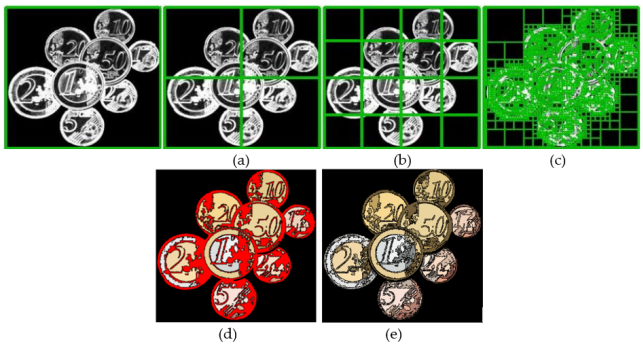


FIGURE III.4 – Segmentation par division/fusion. (a) image originale, (b) résultat intermédiaire de division, (c) division finale, (d) résultat intermédiaire de fusion, (e) fusion finale[22].

III.3.3 Segmentation par classification des pixels :

Ces méthodes regroupent les pixels partageant les mêmes attributs sans tenir compte de leur connexité. Les ensembles des pixels similaires sont appelés classes, et chaque classe correspond de ce fait à une région de l'image. Parmi ces méthodes, on distingue les techniques de seuillage et celles de classification [23][19].

III.3.3.1 Les méthodes de seuillage :

Le seuillage est basé sur l'histogramme est l'une des méthodes de segmentation les plus simples à mettre en œuvre. L'histogramme permet de représenter la distribution des niveaux de gris sur l'ensemble de pixels de l'image. Chaque pic identifié dans l'histogramme correspond à une zone de forte densité, c'est-à-dire, à une plage de niveau de gris partagée par un grand nombre de pixels correspondant à une région importante, voire un objet prédominant dans l'image. Il suffit de trouver les seuils qui séparent au mieux ces régions sur l'histogramme, ce qui n'est pas toujours une tâche aisée[23].

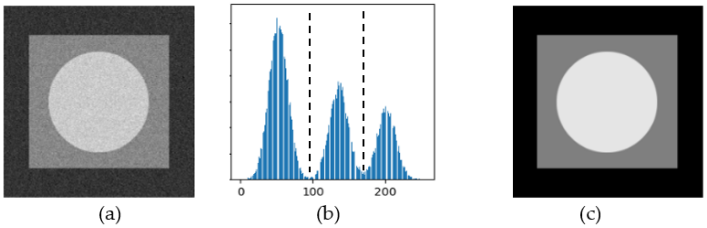


FIGURE III.5 – Segmentation par seuillage. (a) image originale, (b) histogramme et seuils, (c) segmentation par seuillage[24].

III.3.3.2 Méthodes de classification :

Nombreuses méthodes de classifications développées initialement dans le partitionnement des données dans l'apprentissage automatique (machine learning) trouvent application dans la segmentation d'image, tel que la méthode des K-means ou K-clusters. C'est une méthode d'apprentissage non supervisée, qui après un certain nombre d'itération, regroupe des données d'entraînement non étiquetées en groupes dits également clusters[23]. (voir figure III.6)

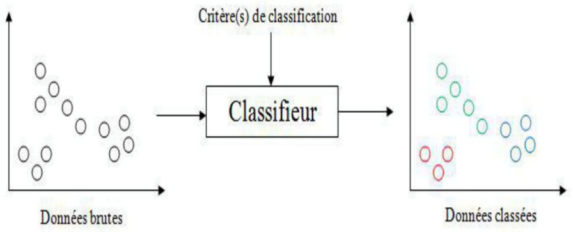


FIGURE III.6 – Classification par la méthode des K-means. (a) données initiales au nombre de 12, (b) classification en 3 classes[25].

Dans la segmentation d'image, la méthode des K-clusters permet donc de classer les pixels de l'image en des clusters correspondant chacun à une région homogène de l'image sans avoir au préalable l'image étiquetée. Toutefois les préalables de cette méthode sont le nombre de classe (ou clusters) K, et la nécessité d'une initialisation aléatoire des centres des classes.

A chaque itération, chaque pixel de l'image est associé au centre le plus proche pour former les K-clusters. Ensuite, les centres de classe sont mis à jour en calculant la moyenne des K-cluster formés. Ce processus se répète jusqu'à atteindre un certain nombre d'itérations ou jusqu'à ce que les centres de classes se stabilisent[23]. Un résultat de la segmentation au moyen de la méthode des K-Clusters est illustré dans la figure III.7.

FIGURE III.7 – Segmentation par la méthode des K-means (a) image originale, résultats de la segmentation pour (b) K=2 , (c) K=3, (d) K=10[26].

III.4 La segmentation basées sur le deep learning

Avec l'essor du deep learning et ses nombreuses applications en vision par ordinateur, la segmentation d'images a connu aussi une évolution et de nouveaux concepts sont apparus. En effet, pour répondre aux exigences spécifiques des applications de la DL en conduite autonome d'automobile et en médecine pour ne citer que cela, des nouvelles notions de segmentation sont nées, à savoir, la segmentation sémantique, la segmentation par instance, et la segmentation panoptique , avec toujours le but de diviser l'image en régions distinctes mais également significatives. Cette dernière caractéristique est permise grâce au potentiel inhérent de l'apprentissage profond. On va s'atteler dans la suite à définir ces nouveaux paradigmes.

III.4.1 La segmentation sémantique :

La segmentation sémantique associe une étiquette à chaque pixel contenu dans une image selon sa classe sémantique, c'est-à-dire selon le sens dans l'image[15]. Une classe sémantique peut être désignée par catégorie, comme le montre la figure III.8, où l'image contient quatre classes sémantiques ou quatre catégories :

— Humain, étiqueté en rouge,

— Arbre (tronc et feuillage), étiquetés en vert foncé,

— Herbe, étiqueté en vert clair,

— Ciel, étiqueté en bleu pour les pièces.

(a) (b)

FIGURE III.8 – Segmentation Sémantique : (a) image originale, (b) Segmentation en 4 classes sémantiques (Humain, Arbre, Herbe, Ciel)[27].

III.4.2 La segmentation par instance

Une classe sémantique peut contenir plusieurs objets, dit instances. La segmentation d'instance se concentre sur une même classe sémantique pour y détecter séparément chacune de ses instances. Le résultat de la détection se fait par un étiquetage différencié ou par un cadre entourant chaque instance[15].

Par rapport à l'image de la figure III.9, nous considérons la classe sémantique « Humain » et les instances sont les « Personnes », la segmentation par instance va donc détecter chaque personne séparément, et le résultat dans ce cas-ci est fait par un étiquetage différencié. On voit par ailleurs que seule la classe sémantique à intérêt à savoir « Humain » est représentée, car le modèle de segmentation par instance dans la pratique ne traite que cette classe.

FIGURE III.9 – Segmentation par instance. Les différentes « Personnes » (instances) dans la classe sémantique « Humains » sont détectées séparément[27].

III.4.3 La segmentation panoptique

La segmentation panoptique combine entre les deux types d'informations, sémantique et instance. C'est-à-dire, en plus de procéder à une segmentation sémantique, on détecte et segmentent chaque instance dans une classe sémantique. La segmentation panoramique signifie que toutes les catégories (classes) doivent être détectées et que différents objets (instances) de la même catégorie doivent être distingués[15].

Par rapport l'image de la figure III.10, il y a segmentation de toutes les classes sémantiques (Humain, Arbre, Herbe, Ciel), et aussi segmentation de toutes les instances dans la classe sémantique Humain.

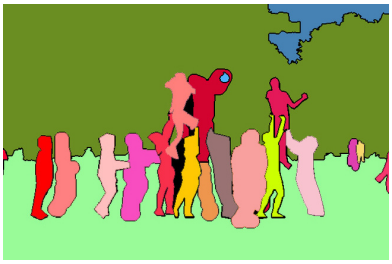


FIGURE III.10 – Segmentation Panoptique[27].

III.4.4 Modèles de DL pour segmentation d'image

On distingue nombreux modèles, et la figure III.11 présente un diagramme qui illustre la chronologie des travaux les plus populaires basés sur DL pour la segmentation sémantique, ainsi que la segmentation d'instance depuis 2014. L'article [Image Segmentation Using Deep Learning : A Survey, 2020] présente tous ces modèles en termes de fonctionnement, architecture et résultats de segmentation. Nous nous contentons de les lister quelques-uns :

— Fully Convolutional networks,

— Convolutional Models with Graphical Models,

— Encoder-Decoder Based Models : U-Net, V-Net,

— Multi-Scale and Pyramid Network Based Models,

— R-CNN Based Models (for Instance Segmentation),

— DeepLab Family and Dilated Convolutional Models : Développé par Google en 2015,

— Recurrent Neural Network Based Models,

— Attention-Based Models,

— Generative Models and Adversarial Training,

— CNN Models with Active Contour Models.

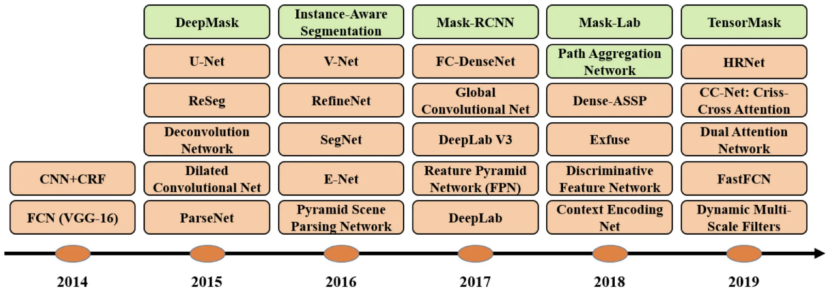


FIGURE III.11 – La chronologie des algorithmes de segmentation basés sur le DL. Les blocs orange et vert renvoient respectivement aux algorithmes de segmentation sémantique et d'instance[28].

Pour la segmentation des images médicales, les modèles qui ont été conçus dans ce but et qui sont les plus connus, on cite le FCN (Fully CNN) et l'U-Net. Ce dernier possède des caractéristiques particulières, d'où notre choix de l'étudier et ça fera l'objet des prochains chapitres[28].

III.4.5 Application des algorithmes de Segmentation basés sur le DL

On en peut finir cette partie sans aborder les applications types de segmentation la segmentation, notamment la sémantique, basée sur les modèle DL combinée avec l'IA.

Véhicules autonomes : Les voitures autonomes s'appuient sur la segmentation sémantique pour voir le monde qui les entoure et réagir en temps réel. La segmentation sémantique divise en classes ce que voit la voiture, comme la route, les autres véhicules, les panneaux, les feux de circulation, les intersections, les piétons... etc. Grâce aux informations qui lui sont fournies par le biais de la segmentation sémantique, la voiture peut circuler jusqu'à destination et est capable de réagir face aux événements inattendus [28].

Diagnostic médical : Bon nombre de procédures médicales courantes telles que le scanner, la radiographie et l'IRM, reposent sur l'analyse d'image. Traditionnellement cette tâche est réalisée par un professionnel de la santé, mais aujourd'hui, l'IA équipée d'une fonctionnalité de segmentation sémantique permet de faire cette analyse et de détecter les anomalies et même de suggérer des diagnostics. Il faut tout de même signaler que cette dernière revient toujours aux professionnels de la santé[28].

Photographie : Les filtres et les fonctionnalités les plus utilisés dans les applications comme Instagram et TikTok s'appuient sur la segmentation sémantique pour identifier les différents objets (têtes, yeux, nez, cheveux,... ou encore, voitures, bâtiments, animaux, etc.) et permettre l'application des filtres ou des effets sélectionnés[28].

Agriculture : En agriculture, l'IA combinée avec l'automatisation et la segmentation sémantique sont utilisées pour détecter les signes d'infestation dans les cultures, et même pour automatiser la pulvérisation des pesticides. La vision par ordinateur indique à l'agriculteur les parties du champ infectées ou à risque[28].

III.5 Conclusion

Ce chapitre a passé en revue les principales approches de segmentation d'images, allant des méthodes classiques basées sur les contours, les régions et la classification des pixels, jusqu'aux

techniques reposant sur le DL et les nouveaux paradigmes de la segmentation. Le prochain chapitre sera consacré à l'étude de l'un des modèles DL conçu spécifiquement pour la segmentation d'images médicales, à savoir l'U-Net.

Chapitre IV

L'U-Net pour la segmentation d'image

IV.1 Introduction

La segmentation d'images médicales est cruciale dans le diagnostic clinique, et les méthodes basées sur les réseaux de neurones convolutionnels (CNN) ont montré de bonnes performances. Parmi elles, U-Net s'est imposé comme une architecture de référence pour la segmentation médicale, grâce à sa capacité à produire des segmentations fines, même avec un nombre limité d'images d'entrainement.

IV.2 Définition

U-Net est une architecture de réseau de neurones convolutionnels (CNN) spécialement conçue pour la segmentation d'images. Son appellation U-Net est en lien à sa forme distinctive en "U", qui est aussi la clé centrale de son fonctionnement. Initialement développée pour la segmentation d'images biomédicales, il s'est rapidement imposé dans de nombreux d'autres domaines grâce à sa robustesse, notamment dans les situations où les données d'entrainement sont peu nombreuses[29].

IV.3 Structure de l'UNet

Le réseau U-Net comporte deux chemins, un premier chemin contractant appelé aussi encodeur ou chemin descendant ou encore chemin de sous-échantillonnage et un deuxième chemin symétrique dit chemin d'expansion appelé également décodeur ou chemin ascendant ou encore chemin de sur-échantillonnage.

L'encodeur (chemin de contraction) réduit les dimensions spatiales (hauteur et largeur de l'image) et extrait des informations de haut niveau à chaque étape, tandis que le décodeur (chemin expansif) utilise ces informations pour reconstruire une carte de segmentation détaillée de l'image.

Nous donnerons à présent une décrire chaque chemin, en plus d'autres aspects du modèle pour enfin proposer son architecture[29].

IV.3.1 Chemin Contractant ou Encodeur

Il est construit sur le modèle du CNN, et est constitué de plusieurs sous blocs encodeurs dont le but de chacun est de réduire par 2 la taille de la carte en entrée et d'augmenter par 2 le nombre des canaux (ou la profondeur). Autrement dit, une carte de taille [l,h,p] en entrée, à la sortie du bloc elle sera de taille [l/2 ,h/2 ,2×p].

Dans chaque bloc sont effectuées :

- Une première Convolution avec des filtres de taille 3×3 suivi d'une activation Relu.Cette convolution est effectuée sur les p canaux à l'entrée du sous-bloc,
- une deuxième Convolution de taille 3×3 avec activation Relu toujours sur les p canaux,
- un MaxPool de stride 2×2, qui a pour effet de diviser la taille de la carte par 2.

Ainsi le chemin retractant, comme le CNN, permet au bout de quelques sous-blocs encodeur de diminuer le nombre de pixels, grâce au MaxPooling, et d'augmenter le nombre d'informations

de chaque pixel grâce au calcul de convolution. Sauf qu'ici, deux convolutions sont effectuées, et à la différence du CNN, et qu'il n'y a pas de phase de concaténation ni de classification qui en suivent comme dans le CNN[29].

IV.3.2 Chemin d'expansion ou Décodeur

Il se compose d'une succession de plusieurs sous-blocs décodeurs, dont chacun a pour rôle exactement l'inverse de sous bloc encodeur. En effet, il multiplie par 2 la taille de la carte en entrée et réduit par 2 le nombre des canaux. Autrement dit, si une carte en entrée est de taille [l,h,p], à la sortie elle sera de [2×l, 2×h, p/2].

Chaque sous-bloc décodeur, est connecté au sous bloc encodeur du chemin contractant se trouvant au même niveau, via une connexion dite de concaténation ou connexion résiduelle ou encore saut de connexion. Le but étant de prendre une copie de la carte à la sortie du sous bloc encodeur pour la fusionner avec la carte actuelle du sous bloc décodeur. Cela permet de combiner les informations importantes, dites globales obtenues au bout du décodage et qui sont préservées dans les phases du chemin d'expansion, avec les informations perdues lors de l'encodage (sous-échantillonnage), dits locales et qui sont généralement les détails et les structures fines.

Cette symétrie entre encodeur et décodeur permet de restituer les détails fins tout en utilisant les informations contextuelles globales extraites.

Dans chaque sous bloc décodeur sont effectués :

- Un UpSample (sur-échantillonnage) de stride 2×2, qui permet d'augmenter la taille de la carte par 2,
- Une concaténation de la carte actuelle avec celle provenant de bloc downSample se trouvant au même niveau, via la connexion résiduelle. Les deux cartes ont la même taille [l,h,p],
- un calcul de Convolution dont le noyau est de taille (2×2)ou (3×3) suivie d'une activation Relu. Le calcul s'effectue sur p/2 canaux, où p est le nombre de canaux en entrée du sous-bloc décodeur. Cette convolution permet de pour raffiner les cartes caractéristiques.

Le chemin d'expansion, permet au bout de quelques sous blocs décodeurs d'augmenter le nombre des pixels et de diminuer le nombre d'informations de chaque pixel[29].

IV.3.3 Goulot d'étranglement

Entre les deux chemins contractant et à expansion, il est inséré un bloc de transition appelé Goulot d'étranglement dans le but est de ressortir davantage les caractéristiques les plus abstraites (profondes) capturées par le codeur avant d'être envoyées vers l'encodage.

Dans ce bloc sont effectuées deux Convolutions successives avec des filtres de taille 3×3 suivies d'une activation Relu. Cette convolution est effectuée sur les p canaux à l'entrée du bloc [29].

IV.3.4 Architecture de l'UNet

Nous pouvons à présent présenter l'architecture de U-Net (voir figure IV.1).

1. Chemin contractant constitué de N sous blocs encodeurs, où N est un entier indiquant la profondeur du réseau. Dans la figure IV.1, on a présenté un exemple pour N=3. Cette partie constitue le chemin de contraction, c'est la branche gauche du U.

2. Chemin d'expansion, constitué de N sous blocs décodeurs. Cette partie constitue le chemin d'expansion, c'est la branche droite du U.

3. Connexions résiduelles liant les sous bloc décodeurs aux sous blocs encodeur, deux par deux.

4. Entre les deux chemins contractant et à expansion, il est inséré un bloc de transition appelé « goulot d'étranglement » dans le but est de ressortir davantage les caractéristiques les plus abstraites (profondes). Elle est constituée de deux couches de convolution suivie d'activation ReLu.

L'image à l'entrée et la carte à la sortie, peuvent être soumises à des convolutions pour ajuster la taille ou affiner le contenu.

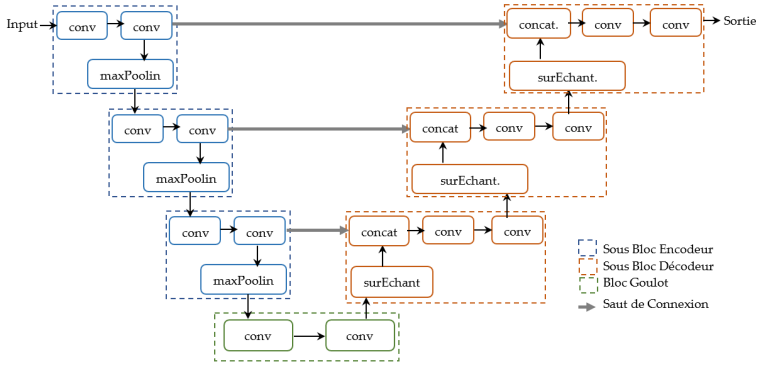


FIGURE IV.1 – Schémas représentatif de l'architecture U-Net.

La structure en "U" est non seulement symétrique mais aussi fonctionnelle, car elle permet la fusion des caractéristiques de contexte à partir du chemin contractant avec des informations de localisation plus détaillées à partir du chemin d'expansion, grâce aux connexions résiduelles (sauts de connexion). Cette combinaison de contexte et de localisation est cruciale pour effectuer une segmentation précise à l'échelle des pixels.

La figure IV.2 représente l'UNet tel qu'il est proposé par Olaf et son équipe dans [Ronneberger et al,2015] que nous utiliserons pour segmenter les images médicales en chapitre 5.

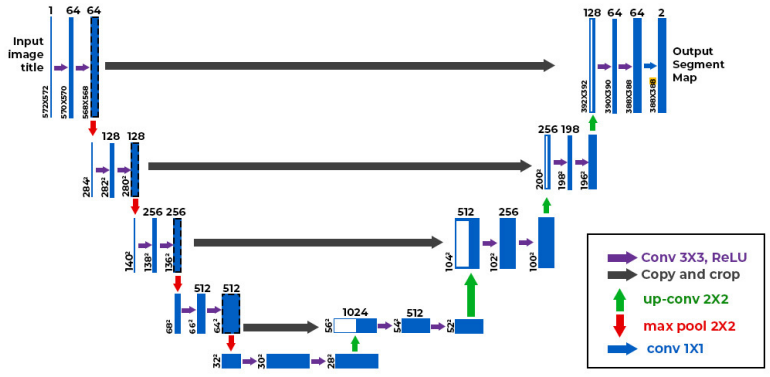


FIGURE IV.2 – l'architecture UNet d'Olaf [29].

IV.4 Fonction de Coût et Optimisation

La fonction de coût est un élément fondamental dans le processus d'apprentissage du modèle UNet, c'est elle qui le guide dans la tâche de segmentation.

Comme la segmentation consiste à classifier chaque pixel de l'image comme appartenant à une classe ou non, et Comme dans le commun des segmentations médicales, il s'agit de classer les pixels en deux classes : objet (organe, tumeur,..) et arrière-plan, il est donc utilisé une fonction de perte adaptée à ce type de segmentation, dite binaire, et qui est la Binary Cross-Entropy (BCE). Cette fonction mesure l'écart entre les valeurs prédites par le modèle et les valeurs réelles du masque binaire (0 ou 1). Formellement, la BCE est définie par :

$$BCE = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)] \tag{IV.1}$$

Où N : nombre de pixels, y_i est l'étiquette binaire (0 ou 1) dans l'image binaire de référence du $i^{\text{ème}}$ pixel, et p_i est la probabilité prédite de ce $i^{\text{ème}}$ pixel d'appartenir à la classe 1.

L'entropie croisée binaire mesure la distance entre les étiquettes de classe de référence (0 ou 1) et les probabilités prédites produites par le modèle. Plus la valeur binaire de l'entropie croisée est faible, meilleures seront les prédictions du modèle pour s'alignent sur les étiquettes vraies. Cette fonction de coût permet de pénaliser fortement les erreurs de prédiction et de ce fait permet une détection précise des pixels de l'objet à segmenter[6][29].

Durant l'entraînement, cette fonction de coût est minimisée à l'aide de l'algorithme d'optimisation Adam (Adaptive Moment Estimation), qui combine les avantages de deux extensions de la descente du gradient : le « Momentum » et le « Root Mean Square Propagation » noté RSMpropo. Les deux méthodes ont pour but d'accélérer la descente du gradient en évitant les grandes variations qui entrainent la divergence du gradient de la fonction du coût ou les très faibles variations, notamment au début, qui entrainent à un minimum local. La descente du gradient avec momentum manie la moyenne du gradient alors que RSMprop la variance du gradient. L'algorithme ADAM qui combine entre les deux méthodes, manie à la fois la moyenne et la variance du gradient, été ceci dans le but d'assurer la convergence de la descente du gradient vers un minimum global de manière rapide et stable[6].

Concrètement, à chaque itération d'entraînement, le modèle effectue une passe avant pour produire une prédiction, puis une rétropropagation (backpropagation) de l'erreur calculée par la fonction de coût BCE est effectuée pour mettre à jour les paramètres du réseau. Ce processus est répété pendant plusieurs itérations, permettant au modèle d'apprendre progressivement à extraire des caractéristiques discriminantes et à améliorer sa capacité de segmentation[29].

Nous présentons dans la suite les étapes de l'algorithme ADAM :
1. Estimation de la moyenne des gradients, notée m_t, et est calculée comme suit :

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{IV.2}$$

Où $g_t = \frac{\partial(\text{BCE})}{\partial t}$ est le gradient à l'instant t, β_1 est un hyperparamètre qui contrôle le taux de décroissance de la moyenne du gradient, généralement proche de 0,9.

2. Estimation de la variance des gradients, notée v_t et est calculée comme suit :

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{IV.3}$$

Où $g_t = \frac{\partial(\text{BCE})}{\partial t}$ est le gradient à l'instant t, β_2 est un hyperparamètre qui contrôle le taux de décroissance de la moyenne des carrés du gradient, généralement proche de 0,999.

3. Correction des biais, puisque la moyenne m_t et la variance v_t sont initialisées à zéro, elles ont tendance à être biaisées vers zéro surtout pendant les étapes initiales. Pour corriger ces biais, on calcule des estimations corrigées de la moyenne et de la variance suivant les expressions suivantes :

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{IV.4}$$

4. Mise à jour des paramètres, notés w_t suivant l'expression suivante :

$$w_t = w_{t-1} - \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \alpha \tag{IV.5}$$

Où α est le taux d'apprentissage, généralement égal à 0,001 et ε est une constante positive, de l'ordre de 10^{-8}, insérée pour éviter la division par zéro [30].

IV.5 Les avantages et les inconvénients de U-Net

IV.5.1 Les avantages de U-Net

- **Haute précision avec des données limitées** : U-Net atteint d'excellentes performances même avec des petits ensembles de données de formation grâce à son architecture et à ses stratégies d'augmentation des données[31].
- **Localisation précise** : la combinaison de fonctionnalités de bas et de haut niveau via des connexions de saut permet une localisation précise des limites des objets[31].
- **Rapide et efficace :** l'architecture entièrement convolutive d'U-Net permet un traitement efficace d'images volumineuses avec des vitesses de segmentation rapides [31].
- **Architecture simple et modulaire :** :basée sur des auto encodeurs et couches de convolution, facile à implémenter et à comprendre[31].
- **Très utilisée dans le domaine médical :** excellente performance pour détecter les anomalies, tumeurs, etc[31].
- **Segmentation précise au niveau de pixel :** permet une analyse fine des images [31].

IV.5.2 Les inconvénients de U-Net

- **Consommation élevée de ressources :** demande beaucoup de calculs et de mémoire, difficilement applicable en temps réel[31].
- **Sensibilité au surapprentissage (overfitting) :** comme beaucoup de modèles de deep learning, il peut trop s'ajuster aux données d'entrainement s'il n'est pas bien régularisé[31].

IV.6 Conclusion

Ce chapitre a été consacré à l'U-Net, sa structure et son fonctionnement ont été largement détaillés. L'architecture du modèle d'Olaf Ronneberger et al, conçue pour la segmentation d'image médicale a été présentée, son implémentation et application sur l'objet du chapitre suivant.

Chapitre V

IMPLEMENTATION ET RESULTATS EXPERIMENTAUX

V.1 Introduction

Dans ce chapitre, nous allons aborder l'implémentation sous Python du modèle UNet. Nous l'appliquerons sur deux bases de données d'images biomédicales et nous évaluerons ses performances. Mais avant cela nous décrirons l'environnement de travail et exposerons les bibliothèques Python nécessaires pour le fonctionnement du modèle.

V.2 L'environnement de travail

Nous présentons dans ce qui suit l'environnement matériel (hardware) et logiciel nécessaire à la réalisation de notre travail.

V.2.1 Hardware

Le matériel utilisé pour l'exécution de ce travail est un ordinateur bureautique ALFATRON des caractéristiques suivantes :

— CPU : Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz.

— RAM : 64 Gb.

— Le GPU NVIDIA Quadro P5000 avec 16 Go de mémoire, qui nous a été fourni par SONATRACH à Hassi Messaoud lors de notre stage chez aux.

V.2.2 Software

V.2.2.1 Langage de programmation

Python est le langage de programmation le plus largement utilisé dans le domaine de l'intelligence artificielle, et en particulier dans l'apprentissage profond, car il contient les bibliothèques les plus fiables et utiles pour l'utilisation des réseaux de neurones et de la vision artificielle. Nous avons utilisé la version 3.9.0 / 64 bits de python[25].

V.2.2.2 Visual Studio Code

Visual Studio Code est un éditeur de code gratuit et open source de Microsoft, compatible avec Windows, MacOs et Linux. Léger, personnalisable et extensible, il prend en charge plusieurs langages de programmation comme « Python » et convient aussi bien aux débutants qu'aux développeurs expérimentés[25].

V.3 Préparation de l'environnement Software

Nous présentons les bibliothèques de Phyton nécessaires pour l'implémentation et le déroulement du modèle U-Net et pour son application dans la segmentation des images.

— **TensorFlow :** est un framework open source de calcul numérique développé par Google et publié en novembre 2015. Il est rapidement devenu l'un des outils les plus utilisés pour le deep learning, notamment grâce à sa capacité à manipuler des tensors (structures de données multidimensionnelles). De nombreuses applications de Google, comme Gmail, Google Photos ou la reconnaissance vocale, reposent aujourd'hui sur TensorFlow[11].

— **Keras :** est une API de haut niveau en Python qui permet de créer facilement des modèles de deep learning. Elle est simple, flexible, et s'appuie principalement sur TensorFlow comme back-end[11].

— **NumPy :** est une bibliothèque pour langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux[32].

— **OpenCV :** est une bibliothèque graphique libre, initialement développée par Intel spécialisée dans le traitement d'images en temps réel. La société de robotique Willow Garage et la société ItSeez se sont succédé au support de cette bibliothèque[32].

— **Matplotlib :** est une bibliothèque du langage de programmation Python destinée à tracer et visualiser des données sous formes de graphiques. Elle peut être combinée avec les bibliothèques Python de calcul scientifique NumPy et SciPy[32].

— **PIL :** la bibliothèque Python Imaging Library ajoute des capacités de traitement d'image à votre interpréteur Python. Cette bibliothèque offre une prise en charge étendue des formats de fichier, une représentation interne efficace et des capacités de traitement d'image assez puissantes. La bibliothèque d'image de base est conçue pour un accès rapide aux données stockées dans quelques formats de pixel de base[32].

— **Le module os :** est un module fourni par Python dont le but est d'interagir avec le système d'exploitation. Il permet ainsi de gérer l'arborescence des fichiers, de fournir des informations sur le système d'exploitation, processus, variables, systèmes, ainsi que de nombreuses fonctionnalités du système. Le module os peut être chargé simplement avec la commande : import os[33].

— **Tifffle :** est une bibliothèque Python permettant de lire et d'écrire des fichiers TIFF, notamment utilisés en bioimagerie, en stockant des tableaux NumPy et des métadonnées[34].

— **Scikit-learn (ou Sklearn) :** est une bibliothèque open source de machine learning en Python. Elle permet d'effectuer des tâches comme la classification, la régression, le clustering et le prétraitement des données. Elle s'appuie sur les bibliothèques NumPy et Matplotlib, essentielles pour l'analyse et la visualisation des données[32].

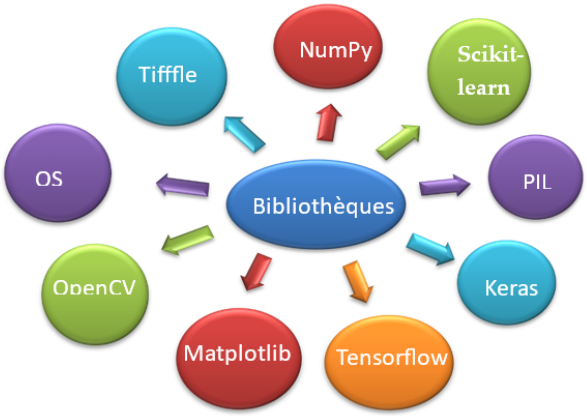


FIGURE V.1 – ensemble des bibliothèques utilisées

V.4 Les Bases de Données

Nous avons fait nos tests sur deux bases de données que nous nous sommes procurés à partir de la plateforme « Kaggle » qui est une plateforme en ligne, filiale de Google, dédiée aux data scientists et aux spécialistes du machine learning. Elle permet de partager des jeux de données, de collaborer, de créer des modèles et de participer à des concours. Kaggle propose aussi un environnement cloud, appelé Kernels, pour coder et analyser en Python.

V.4.1 Base de données Chest-ct-segmentation

La base de données Chest-CT-Segmentation est couramment utilisée pour la segmentation des images thoraciques issues de scanners (CT scans), notamment dans les travaux récents portant sur l'identification des lésions pulmonaires. Elle contient environ 16 708 paires d'images et de masques annotés, réparties entre l'entraînement et le test. Chaque image est de taille 512×512 pixels, en niveaux de gris. Les masques associés comportent trois canaux distincts, chacun correspondant à une structure anatomique spécifique : les poumons, le cœur et la trachée. Grace a son volume important et ses annotations précise, ce dataset permet l'entrainement de modèles de segmentation profonds et performants, contrairement à d'autres bases plus restreintes.

V.4.2 Base de données DRIVE

La base de données DRIVE est largement utilisé pour la segmentation des vaisseaux rétiniens, notamment dans les études récentes qui évaluent les performances des modèles. Il contient 20 images d'entraînement et 20 de test, chacune de taille 584×565 pixels, représentant différents cas pathologiques. Cependant, le nombre limité d'images rend difficile l'obtention de modèles très performants.

V.5 Prétraitement des images

Le prétraitement des images est une étape essentielle dans le cadre de la segmentation d'images médicales. Il permet de préparer les images brutes afin de les rendre mieux exploitables par le modèle et permettre d'améliorer ses performances.

L'ensemble des prétraitements effectués sur les images des bases de données sont :

— Conversion des images couleur en des images niveaux de gris.

— Amélioration du contraste localement en appliquant une égalisation d'histogramme sur l'image par des petites portions.

— Normalisation centrée réduite suivie d'une remise à l'échelle. La normalisation de l'intensité des images de la base des données se fait en soustrayant la moyenne puis en divisant par l'écart type du dataset. Puis une remise à l'échelle entre 0 et 255 est accomplie. Cette étape est importante car elle permet de stabiliser et d'accélérer l'apprentissage du réseau.

— Transformation gamma pour ajuster la luminance des images. Un gamma supérieur à 1 fait d'éclaircir l'image, ce qui aide à faire ressortir les structures peu visibles comme les vaisseaux sanguins.

Toutes ces opérations de prétraitement sont réalisées par la fonction « `my_PreProc()` » en annexe 1.

V.6 Construction de l'U-Net

Nous allons implémenter le modèle U-Net de la figure IV.2. Les images en entrée, après prétraitement, sont de dimensions 584×560×1, correspondant à des images en niveaux de gris.

- **Partie encodeur :** constitué de 3 blocs :
  - Un premier bloc applique :
    - o deux convolutions avec 16 filtres 3×3 , et chacune suivie d'une activation ReLU.
    - o un MaxPooling 2×2, réduisant la taille à 292×280×16.
  - Le deuxième bloc utilise 32 filtres, et applique la même séquence : deux convolutions suivie d'une activation Relu et d'un MaxPooling, pour une sortie de 146×140×32.
  - Le troisième bloc augmente encore la profondeur avec 64 filtres, pour une sortie de 73×70×64, en réalisant la même séquence.
- **Partie goulot (bottleneck)** : Au point le plus bas du réseau, les deux couches convolutives utilisent 128 filtres chacune, en sortie la carte est de dimension 73×70.
- **Partie décodeur (chemin ascendant) :** par analogie avec la partie encodeur, il est aussi constitué de trois blocs :
  - Un premier bloc applique :
    - o un Sur échantillonnage 2×2, augmentant la taille de 73×70×128 à 146×140×64,
    - o Une concaténation avec la carte récupérée, via la connexion de concaténation,
    - o deux convolutions avec 64 filtres 3×3, et chacune suivie d'une activation ReLU, donnant une sortie de taille 146×140×64.
  - Le deuxième bloc utilise 32 filtres, et applique la même séquence : un sur-échantillonnage, suivi d'une concaténation puis deux convolutions suivie d'une activation Relu, pour une sortie de 292×280×32.
  - Le troisième bloc augmente la taille et diminue la profondeur avec 16 filtres , pour une sortie de 292×280×16, en réalisant la même séquence.
- **Couche de sortie :** où est effectuée une convolution 1×1 avec une fonction d'activation sigmoïde, qui produit une carte de probabilité binaire pour chaque pixel de l'image. Cette carte indique la probabilité d'appartenance d'un pixel à la classe d'intérêt. En sortie on obtient une carte de dimension 584×560×1.

Le code Python d'implémentation de l'UNet sur trouve en Annexe 2.

V.7 Entrainement de l'U-Net

Le modèle est compilé avec l'optimiseur Adam et la fonction de perte Binary Cross Entropy. L'entraînement du modèle U-Net se fait progressivement sur un lot de 8 images, dit batch, cette taille est fixée à l'avance. Pour chaque lot, le modèle est entrainé. Au bout d'une itération,

l'ensemble des données d'entrainement sont injectées au modèle. Le nombre d'itération est aussi fixé à l'avance, dans notre programme, il est de 500 itérations. L'entrainement se fait par la fonction « model.fit () » en annexe 3.

Le tableau V.1, résume les couches du modèle et les paramètres entrainables à chaque itération. Il indique que le modèle contient un total de 481 745 paramètres entraînables, et aucun paramètre non entraînable, ce qui signifie que toutes les couches participent à l'optimisation durant l'entraînement. La taille mémoire estimée du modèle est de 1,84 MB, ce qui reste raisonnable pour des applications biomédicales, qu'on pourrait donc implémenter sur des GPU grand public.

TABLE V.1 – Construction de l'U-Net

Layer (type)	Type complet	Output Shape	Paramètres	Connecté à
Input_2	Input(shape=(584, 560, 1))	(None, 584, 560, 1)	0	[]
Conv2D_1	Conv2D(16, 3, relu, same)	(None, 584, 560, 16)	160	Input_2
Conv2D_2	Conv2D(16, 3, relu, same)	(None, 584, 560, 16)	2320	Conv2D_1
Max_pooling2D_1	MaxPooling2D()	(None, 292, 280, 32)	0	Conv2D_2
Conv2D_3	Conv2D(32, 3, relu, same)	(None, 292, 280, 32)	4640	Max_Pooling2D_1
Conv2D_4	Conv2D(32, 3, relu, same)	(None, 292, 280, 32)	9248	Conv2D_3
Max_pooling2D_4	MaxPooling2D()	(None, 146, 140, 32)	0	Conv2D_4
Conv2D_5	Conv2D(64, 3, relu, same)	(None, 146, 140, 64)	18496	Max_pooling2D_2
Conv2D_6	Conv2D(64, 3, relu, same)	(None, 146, 140, 64)	36928	Conv2D_5
Max_pooling2D_"	MaxPooling2D()	(None, 73, 70, 64)	0	Conv2D_6
Conv2D_7	Conv2D(128, 3, relu, same)	(None, 73, 70, 128)	73856	Max_pooling2D_3
Conv2D_8	Conv2D(128, 3, relu, same)	(None, 73, 70, 128)	147584	Conv2D_7
Conv2D_transpose_1	Conv2DTranspose(64, 2, stride=2,same)	(None, 146, 140, 64)	32832	Conv2D_8
Concat_1	Concatenate()	(None, 146, 140, 128)	0	[Conv2D_transpose_1 + Conv2D_6]
Conv2D_9	Conv2D(64, 3, relu, same)	(None, 146, 140, 64)	73792	Concat_1
Conv2D_10	Conv2D(64, 3, relu, same)	(None, 146, 140, 64)	36928	Conv2D_9
Conv2D_Transpose_2	Conv2DTranspose(32, 2, stride=2, same)	(None, 292, 280, 32)	8224	Conv2D_10
Concat_2	Concatenate()	(None, 292, 280, 64)	0	[Conv2D_Transpose_2 + Conv2D_4]
Conv2D_11	Conv2D(32, 3, relu, same)	(None, 292, 280, 32)	18464	Concat_2
Conv2D_12	Conv2D(32, 3, relu, same)	(None, 292, 280, 32)	9248	Conv2D_11
Conv2D_Transpose_3	Conv2DTranspose(16, 2, stride=2, same)	(None, 584, 560, 32)	2064	Conv2D_12
Concat_3	Concatenate()	(None, 584, 560, 32)	0	[Conv2D_Transpose_3 + Conv2D_2]
Conv2D_13	Conv2D(16, 3, relu, same)	(None, 584, 560, 16)	4624	Concat_3
Conv2D_14	Conv2D(16, 3, relu, same)	(None, 584, 560, 16)	2320	Conv2D_13
Output	Conv2D(1, 1, sigmoid)	(None, 584, 560, 1)	17	Conv2D_14

V.8 Résultats

Dans les figures V.2et V.3, on présente deux résultats de segmentation pour chacune des bases de données Chest-ct-segmentation et DRIVE.

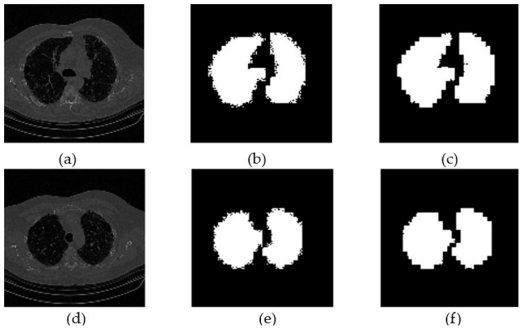


FIGURE V.2 – Résultats de la segmentation des images de scanner thoracique (CT scan). (a)(d) Image originale, (b)(e) vérité terrain, (c)(f) segmentation obtenue par l'UNet.

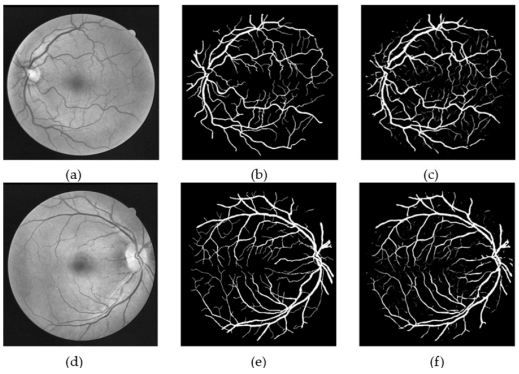


FIGURE V.3 – Résultats de la segmentation des images rétiniennes. (a) Image originale, (b) vérité terrain, (c) segmentation obtenue par l'UNet.

Nous ne commenterons par la suite que les résultats de la base DRIVE. Le modèle U-Net a démontré une bonne capacité à segmenter les vaisseaux sanguins rétiniens, en particulier les troncs principaux ainsi qu'un grand nombre de branches secondaires. La comparaison entre le masque réel (vérité terrain) et le masque prédit montre une correspondance globale satisfaisante dans la structure des vaisseaux.

On note toutefois que les vaisseaux très fins (capillaires), bien visibles dans les annotations manuelles, sont parfois mal détectés ou absents dans la prédiction, ce qui met en évidence une sensibilité encore limitée du modèle pour les structures de petite taille.

Par ailleurs, le modèle génère très peu de fausses détections en dehors de la zone vasculaire, ce qui traduit une bonne gestion du bruit de fond.

V.9 Evaluation des performances du Modèle : Entrainement et Test

V.9.1 Evaluation pendant l'entrainement

Les métriques utilisées pour évaluer les performances du modèle pendant l'entrainement sont : l'accuracy (précision globale), la sensitivité (capacité à détecter les vaisseaux) et la spécificité (capacité à ignorer les faux vaisseaux). Ces métriques d'évaluations sont calculées à partir des indicateurs suivants : les vrais positifs (TP), les vrais négatifs (TN), les faux positifs (FP), les faux négatifs (FN). Les expressions de calcul sont en annexe 4, et la fonction python qui permet de les calculer est confusion_matrix() et elle est en Annexe 4.

Il faut signaler que ces métriques sont calculées sur une image segmentée choisie aléatoirement, et ce n'est pas une moyenne, et qu'elles sont calculées uniquement sur la zone utile de l'image, définie par le masque FOV. Les métriques de la base DRIVE sont listées dans le Tableau V.2, et nous les commentons dans la suite :

- **TP (Vrais positifs) = 21 655**

  Ce chiffre élevé traduit une bonne capacité de détection des structures vasculaires principales et secondaires. Cela montre que le modèle réussit à reproduire fidèlement la majorité des segments vasculaires présents dans les annotations de référence, en particulier les troncs et branches bien définis. Ce résultat est essentiel dans le contexte médical, car il reflète la fiabilité du modèle pour repérer les zones cliniquement pertinentes.

- **FP (Faux positifs) = 6 610**

  Il s'agit des pixels prédits comme vaisseaux alors qu'ils ne le sont pas. Ce nombre, bien que non négligeable, reste relativement faible par rapport au nombre total des pixels, ce qui témoigne d'un bon contrôle du bruit de fond.

- **FN (Faux négatifs) = 6 563**

  Ce sont les vaisseaux que le modèle a manqués. On peut supposer que ce sont souvent les petits vaisseaux très fins ou peu visibles. C'est ici que le modèle montre ses limites : il a un peu plus de mal à détecter les détails très fins.

- **TN (Vrais négatifs) = 292 212**

  Là, le score est impressionnant. Plus de 292 000 pixels ont bien été reconnus comme "non-vaisseaux". Cela montre que le modèle est très fiable pour ne pas confondre le fond avec les vaisseaux, ce qui est essentiel en médecine.

- **Accuracy (Précision globale) = 0,9597 (95,97 %)**

  Un taux de précision globale (Accuracy) de 95,97 % indique que le modèle U-Net parvient à bien différencier les pixels vasculaires des pixels de fond dans la grande majorité des cas.Ce résultat reflète une bonne performance générale. Néanmoins, comme les images médicales sont très déséquilibrées (avec une forte dominance de pixels appartenant au fond), cette métrique peut parfois masquer les erreurs de détection sur les vaisseaux, notamment les plus fins. C'est pourquoi il est important de compléter cette mesure avec d'autres indicateurs, comme la sensibilité et la spécificité, pour obtenir une évaluation plus fiable du modèle.

- **Sensitivité = 0,7674 (76,74 %)**

  Le score de sensibilité de 76,74 % montre que le modèle détecte bien la majorité des vaisseaux sanguins, en particulier les plus visibles. Toutefois, environ un quart des vaisseaux, souvent les plus fins ou peu contrastés, ne sont pas détectés. Cela traduit une bonne performance globale, mais avec une marge d'amélioration pour capter les détails plus subtils.

- **Spécificité = 0,9779 (97,79 %)**

  Un score de spécificité de 97,79 % indique que le modèle U-Net est très efficace pour distinguer les zones sans vaisseaux (le fond) des structures vasculaires. Cela signifie qu'il commet très peu de faux positifs, c'est-à-dire qu'il évite de confondre des zones non vasculaires avec des vaisseaux. Cette capacité à ne pas sur-segmenter est essentielle dans le contexte mé-

dical, car elle réduit les erreurs de diagnostic potentielles en garantissant que seules les structures réellement présentes soient détectées comme telles.

TABLE V.2 – Performances du modèle UNet pendant l'entrainement sur la base DIVE

Performances de segmentation dans le FOV	**Résultats**
TP	21655.0000
FP	6610.0000
FN	6563.0000
TN	292212.0000
Accuracy	0.9597
Sensitivity	0.7674
Specificity	0.9779

Le modèle U-Net démontre une excellente performance générale dans la segmentation des vaisseaux rétiniens. Il réussit à détecter la majorité des structures vasculaires tout en évitant de générer trop de faux positifs. La sensibilité légèrement inférieure révèle cependant une difficulté à capter les plus petits capillaires, typique dans ce type de tâche. La spécificité élevée prouve que le modèle fait très peu d'erreurs sur les zones non vasculaires, ce qui est crucial pour des applications cliniques fiables.

Nous rapportons à titre indicatif dans le tableau V.3 les performances du modèle UNet sur la base des données Chest-CT-Segmentation sans pour autant les commenter.

TABLE V.3 – Performances du modèle

Performances de la segmentation	**Résultats**
TP	358120.0000
FP	21812.0000
TN	3297224.0000
FN	9244.0000
Accuracy	0.9916
Sensitivity	0.9748
Specificity	0.9934

V.9.2 Évaluation du modèle sur les données de test

Après l'entraînement du modèle U-Net, une évaluation sur les données de test est effectuée. Cette étape permet de quantifier la performance finale du modèle sur des données qu'il n'a jamais vues auparavant, ce qui est crucial pour estimer sa capacité de généralisation. Les métriques considérées sont : Accuracy (précision) et Perte (Loss). Cette dernière représente l'erreur globale entre les prédictions du modèle et les véritables annotations des images (les vérités terrain), plus cette valeur est faible, plus le modèle est performant. La fonction Python chargée de cette opération est « model.evaluate () » et est en Annexe 6.

TABLE V.4 – Performances du modèle sur les données Test

Les performances	**Les résultats**
Test Loss	0.1085
Test Accuracy	0.9632

Le tableau V.4 résume ces résultats, que nous commentons dans la suite :

- **Loss (perte) = 0,1085**
  Une valeur de 0,1085 est considérée comme relativement faible, suggérant une bonne qualité de prédiction des structures segmentées.
- **Accuracy (précision) = 0,9632**
  Cela signifie que 96,32 % des pixels ont été correctement classifiés par le modèle (comme appartenant ou non aux vaisseaux sanguins dans le contexte d'un dataset comme DRIVE). Cette métrique globale indique que le modèle a appris à bien différencier les régions d'intérêt des autres parties de l'image.

  Donc, ces résultats montrent que le modèle est capable de produire une segmentation précise avec un faible taux d'erreur, ce qui témoigne de la robustesse de l'architecture U-Net appliquée à la segmentation des images médicales.

V.9.3 Courbes d'apprentissage

Les courbes d'apprentissage permettent de visualiser l'évolution de la perte (loss) et de la précision (accuracy) du modèle U-Net au cours de l'entraînement. Elles comparent les performances sur tous les ensembles d'entraînement et de validation (test).

Pendant l'entrainement, ces courbes aident à évaluer la qualité de l'apprentissage, détecter un éventuel surapprentissage et guider les ajustements du modèle. Elles sont essentielles pour comprendre la dynamique de l'apprentissage et optimiser les performances du réseau.

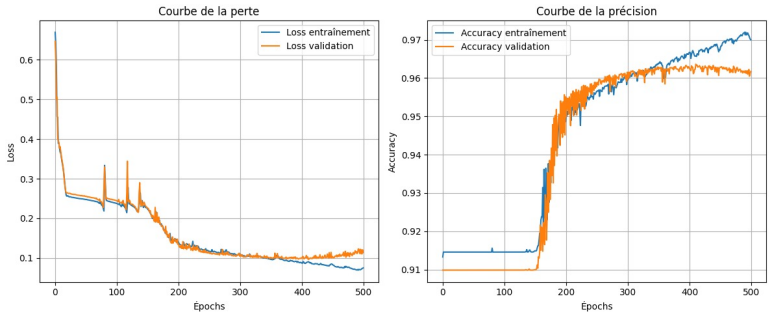


FIGURE V.4 – Les courbes d'apprentissage (perte et précision)

La figure ci-dessus présente les courbes de la perte (loss) et de la précision (accuracy) pour l'entraînement et la validation d'un modèle sur 500 époques. On observe que la perte diminue rapidement dans les premières phases, ce qui indique un bon apprentissage dès les premières itérations. Cependant, à partir d'environ 200 époques, la perte de validation cesse de diminuer et commence même à légèrement augmenter quand le nombre d'époque s'approche de 500, tandis que la précision de validation stagne. En revanche, les performances sur les données d'entraînement continuent à s'améliorer, en perte et en précision. Cela révèle un phénomène de surapprentissage (overfitting) au-delà de 200-250 époques. Pour améliorer la généralisation du modèle, une stratégie d'arrêt anticipé (early stopping) aurait été bénéfique. Le modèle atteint tout de même une précision de validation satisfaisante autour de 96 %.

Nous indiquons qu'il nous faut exactement 10 minutes pour entrainer le modèle sur 500 itérations avec le GPU NVIDIA Quadro P5000.

V.10 Conclusion

Ce chapitre a présenté la mise en œuvre du modèle U-Net pour la segmentation des vaisseaux rétiniens à partir du jeu de données DRIVE. Les étapes de prétraitement, d'entraînement et d'évaluation ont permis d'obtenir des résultats globalement satisfaisants. Le modèle s'est montré performant dans la détection des structures vasculaires principales, tout en révélant quelques limites dans la détection des vaisseaux très fins. Ces résultats confirment l'efficacité et la pertinence de l'architecture U-Net pour des applications en imagerie médicale.

Conclusion Générale

À travers ce travail, nous avons étudié une architecture de réseau neuronal convolutif, l'U-Net, pour assurer la segmentation des images médicales. Après avoir abordé son architecture et son fonctionnement, nous avons dédié tout un chapitre à son implémentation sous Python, son application sur des images médicales provenant de deux bases de données, DRIVE et Chest-ct-segmentation, ainsi qu'à l'évaluation de ses performances.

Nous avons consacré une attention particulière à l'analyse des résultats obtenus sur la base DRIVE, et nous affirmons sans conteste que cette architecture offre des performances remarquables dans la segmentation des vaisseaux rétiniens, une tâche cruciale dans le diagnostic ophtalmologique, malgré la petite taille de la base de données. Une précision atteignant 96 % et une fonction de perte faible, témoignent de la capacité du modèle, d'un côté à bien généraliser sur des images jamais vues, et d'un autre côté, à bien apprendre et à capturer les informations contextuelles à partir d'une petite base de données.

Toutefois, certaines limitations persistent, notamment dans la détection des capillaires très fins, souvent mal identifiés ou absents dans les prédictions obtenues. Nous expliquons cette limitation par la sensibilité du modèle au surapprentissage, mais éventuellement à la taille relativement réduite de la base des données. L'extension de cette base pour augmenter sa taille par des outils qui existent pourrait remédier à ces faiblesses.

En somme, ce mémoire confirme que l'intelligence artificielle appliquée à la vision par ordinateur représente une avancée majeure pour la segmentation d'images médicales, et que des solutions comme l'UNet peuvent jouer un rôle essentiel dans l'aide au diagnostic et à la décision clinique.

Bibliographie

[1] John McCarthy. *What is Artificial Intelligence ?* http://jmc.stanford.edu/articles/whatisai/whatisai.pdf. 2007.

[2] F. Alexandre et M. Minsky. *Intelligence artificielle débraillée*. Éditions Odile Jacob, 2016.

[3] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[4] Devotics. *Présentation de l'intelligence artificielle*. https://devotics.fr/presentation-intelligence-artificielle/. 2020.

[5] Yann LeCun, Yoshua Bengio et Geoffrey Hinton. "Deep Learning". In : *Nature* 521.7553 (2015), p. 436-444. doi : 10.1038/nature14539.

[6] Ian Goodfellow, Yoshua Bengio et Aaron Courville. *Deep Learning*. Disponible en ligne. MIT Press, 2016. url : http://alvarestech.com/temp/deep/Deep%20Learning%20by%20Ian%20Goodfellow,%20Yoshua%20Bengio,%20Aaron%20Courville%20(z-lib.org).pdf.

[7] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. New York : Springer, 2006.

[8] Safae Laqrichi. "Approche pour la construction de modèles d'estimation réaliste de l'effort/coût de projet dans un environnement incertain : application au domaine du développement logiciel". Soutenue le 17 décembre 2015, École doctorale EDSYS (Génie Industriel 4200046). Thèse de doctorat. France : Université de Toulouse, École Nationale Supérieure des Mines d'Albi-Carmaux conjointement avec l'INSA Toulouse, déc. 2015. url : https://hal.science/tel-01320110/.

[9] Mohamed Ilyes Smahi. *Estimation de la QoS dans les services Web par apprentissage profond*. Thèse de doctorat, Université Abou-Bekr Belkaid - Tlemcen. Soutenue le 12 mars 2022. Mars 2022.

[10] Sina Technologie. *Présentation des réseaux de neurones artificiels*. https://blog.sinatechnologie.com/presentation-des-reseaux-de-neurones-artificiels. 2024.

[11] A. M. Sehil. "Image Classification using Deep Learning". Thèse de doctorat. University of Ghardaïa, 2019.

[12] Jérémy Robert. *Convolutional Neural Network : Tout ce qu'il y a à savoir*. DataScientest. juin 2020. url : https://datascientest.com/convolutional-neural-network.

[13] Cezanne Camacho. *Convolutional Neural Networks*. cezannec.github.io. 2020. url : https://cezannec.github.io/Convolutional_Neural_Networks/.

[14] Bernard Perret. *Introduction aux réseaux de neurones convolutionnels*. https://perso.esiee.fr/~perretb/I5FM/TAI/convolution/index.html. s.d.

[15] KH. Amine. "Vers une reconnaissance intelligente des données biomédicales par apprentissage en profondeur". Soutenue le 14 septembre 2021. Thèse de doctorat. Algérie : Université Aboubakr Belkaïd – Tlemcen, sept. 2021.

[16] Radia TOUAHRI. "L'apprentissage profond pour la classification et l'interprétation d'images". Présentée le 14 septembre 2022. Thèse de doctorat. Annaba, Algérie : Université Badji Mokhtar – Annaba, Faculté des Sciences de l'Ingénierie, Département d'Informatique, 2022. URL : https://biblio.univ-annaba.dz/wp-content/uploads/2023/03/These-Touahri-Radia.pdf.

[17] LINKEDIN. *Quels sont les avantages et les inconvénients de l'utilisation des réseaux de neurones convolutionnels (CNN) ?* 2024. URL : https://fr.linkedin.com/advice/0/what-advantages-disadvantages-using-convolutional?lang=fr&lang=fr.

[18] Alex KRIZHEVSKY, Ilya SUTSKEVER et Geoffrey E. HINTON. "ImageNet Classification with Deep Convolutional Neural Networks". In : *Advances in Neural Information Processing Systems*. T. 25. 2012.

[19] Aicha BELLADGHAM. "Segmentation d'images et morphologie mathématique : Application à l'imagerie médicale de l'abdomen". Soutenue en décembre 2014, Département de Génie Électrique et Électronique. Thèse de doctorat. Université Abou-Bekr Belkaïd – Tlemcen, déc. 2014.

[20] Jean-Baptiste COURBOT et al. "Arbres de Markov triplets pour la segmentation d'images". Thèse de doctorat / HDR. Strasbourg, France : Université de Strasbourg – ICube / CNRS, 2016. URL : https://images.icube.unistra.fr/img_auth_namespace.php/6/6f/3-Segmentation.pdf.

[21] Scott E. UMBAUGH. *Snakes and Deformable Curves : Tutorial*. 2004. URL : https://www.siue.edu/~sumbaug/RetinalProjectPapers/Snakes%20and%20Deformable%20Curves%20TUTORIAL.pdf.

[22] Xavier PHILIPPEAU. *Segmentationen régions*. Segmentation en régions. Jan. 2008. URL : https://xphilipp.developpez.com/articles/segmentation/regions/?page=page_3.

[23] Arnaud CAPRI. "Caractérisation des objets dans une image en vue d'une aide à l'interprétation et d'une compression adaptée au contenu : application aux images échographiques". Soutenue en mai 2007. Thèse de doctorat. Université d'Orléans, mai 2007. URL : https://helios2.mi.parisdescartes.fr/~vincent/siten/Publications/theses/pdf/Capri.pdf.

[24] Lakhwinder KAUR, Renu VIG et Jaswinder KAUR. "Edge Detection Techniques : Evaluations and Comparisons". In : *Computer and Electronics Engineering Faculty Publications* 87 (2008). URL : https://digitalcommons.unl.edu/cgi/viewcontent.cgi?article=1087&context=computerelectronicfacpub.

[25] Roufaida ELEZAAR et HadjHammou FAKHAR. *Un réseau neuronal convolutif pour la segmentation d'images IRM cérébrale*. Mémoire universitaire. juillet 2023.

[26] Analytics VIDHYA. *Image Segmentation using K-means*. Juill. 2021. URL : https://medium.com/analytics-vidhya/image-segmentation-using-k-means-4bd79e801785.

[27] Ahmed MANDOUR et OpenGenus Tech Review TEAM. *Panoptic Quality (PQ), Segmentation Quality (SQ) and Recognition Quality (RQ)*. https://iq.opengenus.org/pq-sq-rq/. OpenGenus IQ. 2024.

[28] Zongwei ZHOU et al. "Models Genesis". In : *arXiv preprint arXiv :2001.05566* (2020). URL : https://arxiv.org/abs/2001.05566.

[29] Olaf RONNEBERGER, Philipp FISCHER et Thomas BROX. "U-Net : Convolutional Networks for Biomedical Image Segmentation". In : *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. T. 9351. Lecture Notes in Computer Science. Springer, 2015, p. 234-241. DOI : 10.1007/978-3-319-24574-4_28. URL : https://arxiv.org/abs/1505.04597.

[30] Diederik P. KINGMA et Jimmy BA. "Adam : A Method for Stochastic Optimization". In : *arXiv preprint arXiv :1412.6980* (2014). URL : https://arxiv.org/abs/1412.6980.

[31] IA SCHOOL. *U-Net : le réseau de neurones par vision par ordinateur*. 2025. URL : https://www.intelligence-artificielle-school.com/ecole/technologies/u-net-le-reseau-de-neurones-par-vision-ordinateur/.

[32] Hind Selma GHENNANI et Nihel Loubna GHENNANI. "Apprentissage profond pour une classification histopathologique précise dans le cancer du sein". Présenté le 27 Juin 2024 devant la commission composée de CHOUTTI Sidi Mohammed, BENAZZOUZ Mourtada, et AMGHAR Djaizia. Mémoire de Master. Tlemcen, Algérie : Université Abou Bekr Belkaid – Tlemcen, Faculté des Sciences, Département d'Informatique, 2024.

[33] TRESFACILE.NET. *Le module OS en Python*. 2022. URL : https://www.tresfacile.net/le-module-os-en-python/.

[34] Christoph GOHLKE. *tifffile*. Python Package Index. 2025. URL : https://pypi.org/project/tifffile/.

[35] Chetan L. SRINIDHI, P. APARNA et Jeny RAJAN. "Recent Advancements in Retinal Vessel Segmentation". In : *Journal of Medical Systems* 41.4 (2017). Image & Signal Processing. Received : 6 Jan 2017 / Accepted : 1 Mar 2017 / Published online : 11 Mar 2017, p. 70. DOI : 10.1007/s10916-017-0719-2. URL : https://doi.org/10.1007/s10916-017-0719-2.

Annexe A

Interface graphique

A.1 Annexe 1 : Les fonctions de prétraitement

```python
def rgb2gray(images):
    gray_imgs = np.dot(images[...,:3], [0.2989, 0.5870, 0.1140])
    gray_imgs = gray_imgs[:, np.newaxis, :, :]  # (N, 1, H, W)
    return gray_imgs
```

```python
def clahe_equalized(imgs):
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    imgs_equalized = np.empty(imgs.shape)
    for i in range(imgs.shape[0]):
        imgs_equalized[i, 0] = clahe.apply(imgs[i, 0].astype(np.uint8))
    return imgs_equalized
```

```python
def dataset_normalized(imgs):
    imgs_normalized = np.empty(imgs.shape)
    imgs_std = np.std(imgs)
    imgs_mean = np.mean(imgs)
    imgs_normalized = (imgs - imgs_mean) / imgs_std
    for i in range(imgs.shape[0]):
        imgs_normalized[i] = ((imgs_normalized[i] - np.min(imgs_normalized[i])) /
                              (np.max(imgs_normalized[i]) - np.min(imgs_normalized[i]))) * 255
    return imgs_normalized
```

```python
def adjust_gamma(imgs, gamma=1.2):
    invGamma = 1.0 / gamma
    table = np.array([(i / 255.0) ** invGamma * 255 for i in np.arange(0, 256)]).astype("uint8")
    new_imgs = np.empty(imgs.shape)
    for i in range(imgs.shape[0]):
        new_imgs[i, 0] = cv2.LUT(imgs[i, 0].astype(np.uint8), table)
    return new_imgs
```

```python
def my_PreProc(data):
    gray_imgs = rgb2gray(data)
    gray_imgs = dataset_normalized(gray_imgs)
    gray_imgs = clahe_equalized(gray_imgs)
    gray_imgs = adjust_gamma(gray_imgs)
    gray_imgs = gray_imgs / 255.
    return gray_imgs
```

A.2 Annexe 2 : Le code d'implémentation de l'UNet

```python
# --- U-Net Model ---
def unet_model(input_size=(IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS)):
    inputs = layers.Input(input_size)

    # Encoder
    c1 = layers.Conv2D(16, 3, activation='relu', padding='same')(inputs)
    c1 = layers.Conv2D(16, 3, activation='relu', padding='same')(c1)
    p1 = layers.MaxPooling2D()(c1)

    c2 = layers.Conv2D(32, 3, activation='relu', padding='same')(p1)
    c2 = layers.Conv2D(32, 3, activation='relu', padding='same')(c2)
    p2 = layers.MaxPooling2D()(c2)

    c3 = layers.Conv2D(64, 3, activation='relu', padding='same')(p2)
    c3 = layers.Conv2D(64, 3, activation='relu', padding='same')(c3)
    p3 = layers.MaxPooling2D()(c3)

    c4 = layers.Conv2D(128, 3, activation='relu', padding='same')(p3)
    c4 = layers.Conv2D(128, 3, activation='relu', padding='same')(c4)
```

```python
    # Decoder
    u5 = layers.Conv2DTranspose(64, 2, strides=2, padding='same')(c4)
    u5 = layers.concatenate([u5, c3])
    c5 = layers.Conv2D(64, 3, activation='relu', padding='same')(u5)
    c5 = layers.Conv2D(64, 3, activation='relu', padding='same')(c5)

    u6 = layers.Conv2DTranspose(32, 2, strides=2, padding='same')(c5)
    u6 = layers.concatenate([u6, c2])
    c6 = layers.Conv2D(32, 3, activation='relu', padding='same')(u6)
    c6 = layers.Conv2D(32, 3, activation='relu', padding='same')(c6)

    u7 = layers.Conv2DTranspose(16, 2, strides=2, padding='same')(c6)
    u7 = layers.concatenate([u7, c1])
    c7 = layers.Conv2D(16, 3, activation='relu', padding='same')(u7)
    c7 = layers.Conv2D(16, 3, activation='relu', padding='same')(c7)

    outputs = layers.Conv2D(1, 1, activation='sigmoid')(c7)

    return models.Model(inputs, outputs)
```

A.3 Annexe 3 : Entraînement du modèle U-Net

L'entraînement du modèle a été effectué en utilisant la fonction fit() fournie par la bibliothèque Keras. Cette méthode constitue l'élément central du processus d'apprentissage dans TensorFlow/Keras. Elle permet d'ajuster les poids du modèle en le faisant apprendre à partir d'un ensemble de données d'entraînement (X_train, y_train) au cours de plusieurs époques. À chaque itération, les poids sont mis à jour de manière à minimiser l'erreur entre les prédictions générées et les valeurs réelles (étiquettes).

Voici le programme correspondant à cette fonction :

```
history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=EPOCHS,
    batch_size=BATCH_SIZE
)
```

A.4 Annexe 4 : calcul des métriques à partir de la matrice de confusion

Pour obtenir les valeurs des Vrais Positifs (TP), Faux Positifs (FP), Faux Négatifs (FN) et Vrais Négatifs (TN), nous avons utilisé la fonction confusion_matrix de la bibliothèque scikit-learn. Le code suivant a été utilisé :

```
# Calcul de la matrice de confusion
cm = confusion_matrix(y_true_flat, y_pred_flat)

# Extraction des éléments TN, FP, FN, TP
TN, FP, FN, TP = cm.ravel()
```

Cette fonction prend en entrée :

- y_true_flat : les étiquettes réelles (aplaties),
- y_pred_flat : les prédictions binaires du modèle (aplaties également),

Et retourne une matrice 2x2 sous la forme : [[TN, FP], [FN,TP]]

L'utilisation de .ravel() permet de "déplier" cette matrice pour récupérer directement les valeurs dans l'ordre : TN, FP, FN, TP.

A.5 Annexe 5 : Métriques d'évaluation de performance

A.5.1 Matrice de confusion :

La matrice de confusion ou tableau de contingence est un outil utilisé pour mesurer les performances d'un modèle d'apprentissage automatique en vérifiant spécifiquement l'exactitude de son problème de prédiction et de classification par rapport à la situation réelle. C'est une matrice de taille égale au nombre de classes qui permet la génération des performances de chaque classe prise individuellement[16].

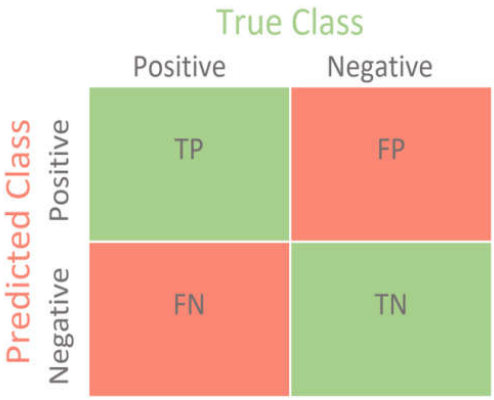


FIGURE A.1 – Matrice de confusion [16].

- TP (True Positive) signifie le pixel de vaisseau est correctement identifié[35].

$$TP = \frac{\text{number of pixels correctly as vessel pixel}}{\text{number of pixels actually in vessel region}} \tag{A.1}$$

- TN (True Négative) signifie le pixel non-vaisseau est aussi correctement identifié[35].

$$FP = \frac{\text{number of pixels wrongly detected as vessel pixel}}{\text{number of pixels actually in vessel region}} \tag{A.2}$$

- FP (faux positif) signifie le pixel de vaisseau mal identifié[35].

$$FP = \frac{\text{number of pixels wrongly detected as vessel pixel}}{\text{number of pixels actually in vessel region.}} \tag{A.3}$$

- FN (faux négatif) signifie le pixel non-vaisseau mal identifié[35].

$$FN = \frac{\text{number of pixels wrongly detected as non-vessel pixel}}{\text{number of pixels actually in non-vessel region}} \tag{A.4}$$

De ces quatre événements, il ressort les mesures suivantes :

— **La Précision[35] :**

$$ACC = \frac{number\ of\ correctly\ classified\ pixel}{number\ of\ pixel\ in\ image\ FOV} = \frac{TP + TN}{TP + TN + FP + FN} \tag{A.5}$$

— **La Sensibilité[35] :** Mesure la capacité à détecter les pixels des vaisseaux et indique les performances d'un bon algorithme pour classer correctement les pixels des vaisseaux.

$$SEN = \frac{TP}{TP + TN} \tag{A.6}$$

— **Spécificité [35] :** Elle mesure la capacité à détecter les pixels non-vaisseaux.

$$SPE = \frac{TN}{TP + TN} \tag{A.7}$$

Résumé

Ce mémoire explore l'application des techniques d'apprentissage profond—en particulier les réseaux de neurones convolutionnels (CNN)—à la segmentation d'images médicales. L'accent principal est mis sur la mise en œuvre d'un modèle U-Net pour la segmentation automatique des vaisseaux sanguins rétiniens à partir du jeu de données DRIVE. Après avoir exploré les fondements du deep learning et les méthodes classiques de segmentation, l'étude détaille l'architecture U-Net, son entraînement, et son évaluation. Le modèle atteint une précision de 96 %, montrant une capacité robuste à détecter les structures vasculaires principales, bien que des limites subsistent pour les capillaires fins. Ce travail confirme l'efficacité de U-Net dans les tâches de segmentation médicale et propose des pistes pour améliorer ses performances futures.

Mots-clés : Apprentissage profond, CNN, U-Net, Segmentation d'images médicales, Vaisseaux sanguins rétiniens, DRIVE, Précision, Évaluation de modèle.

Abstract

This thesis explores the application of deep learning techniques—particularly convolutional neural networks (CNNs)—to medical image segmentation. The main focus is on implementing a U-Net model for the automatic segmentation of retinal blood vessels using the DRIVE dataset. After examining the fundamentals of deep learning and classical segmentation methods, the study details the U-Net architecture, its training process, and its evaluation. The model achieves an accuracy of 96%, demonstrating strong capability in detecting major vascular structures, although some limitations remain with fine capillaries. This work confirms the effectiveness of U-Net in medical segmentation tasks and suggests avenues for improving its future performance.

Keywords : Deep Learning, CNNs, U-Net, Medical Image Segmentation, Retinal Blood Vessels, DRIVE Dataset, Accuracy, Model Evaluation.

ملخص

هذا البحث يستكشف تطبيق تقنيات التعلم العميق— و خاصة الشبكات العصبية الالتفافية (CNN)— في عملية تقسيم الصور الطبية . يركز العمل بشكل أساسي على تنفيذ نموذج U-Net للتقسيم التلقائي للأوعية الدموية في شبكية العين باستخدام قاعدة البيانات DRIVE

بعد استعراض أسس التعلم العميق و الأساليب التقليدية للتقسيم، يفصل البحث بنية نموذج U-Net ،تدريبه و تقييمه . و قد حقق النموذج دقة بلغت 96% ،مما يدل على قدرة قوية في كشف البنى الوعائية الرئيسية ،رغم إستمرار بعض تحديات في تحديد الشعيرات الدموية الدقيقة . تأكد هذه الدراسة فعالية نموذج U-Net في مهام تقسيم الصور الطبية و تقترح سبلا لتحسين أدائه مستقبلا.

الكلمات المفتاحية: التعلم العميق،CNN ، U-Net ،تقسيم الصور الطبية ،الأوعية الدموية الشبكية ، DRIVE ،الدقة ، تقييم النموذج.