الجمهورية الجزائرية الديمقراطية الشعبية
Républiquealgériennedémocratiqueetpopulaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique
جامعة عين تموشنت بلحاج بوشعيب
Université d'Ain Temouchent - Belhadj Bouchaib
Faculté des Sciences et Technologie
Département de Mathématiques et Informatique

Projet de Fin d'Etudes
Pour l'obtention du diplôme de Master en : Informatique
Domaine : Mathématiques et Informatique
Filière : Informatique
Spécialité : Cyber Sécurité et Intelligence Artificielle

Thème

Segmentation d'images urbaines par Deep Learning

Soutenu le : 25 /06/2025

Présenté par :

- Melle: BOUAALAOUI CHAIMA
- Melle: SLATNA AICHA

Devant le jury composé de :

Mme BELGRANA F.Z	**MCA UAT.B.B (Ain Temouchent)**	**Président**
Mme BERRAKEM.F	**MAA UAT.B.B (Ain Temouchent)**	**Examinatrice**
Mme SAIDI S.	**MAA UAT.B.B (Ain Temouchent)**	**Encadrante**

Année Universitaire : 2024/2025

Remerciements

Avant tout, je rends grâce à Dieu Tout-Puissant, pour m'avoir accordé la force, la patience et la santé nécessaires pour mener à bien ce travail.

Je tiens à exprimer ma profonde gratitude à mon encadrant madame SAIDI Samira, pour sa disponibilité, ses conseils pertinents, sa rigueur scientifique et son accompagnement tout au long de ce mémoire. Ses orientations précieuses ont été d'un apport inestimable à la réalisation de ce travail.

Je remercie également les membres du jury, pour l'honneur qu'ils me font en acceptant d'évaluer ce travail, ainsi que pour leurs remarques constructives.

Je n'oublie pas mes collègues et amis, pour leur soutien moral, les échanges enrichissants et les bons moments partagés.

Enfin, j'exprime toute ma reconnaissance à ma famille, en particulier à mes parents, pour leur amour inconditionnel, leur soutien permanent et leurs encouragements sans faille tout au long de mon parcours.

Dedicace

À mon cher père, **SLATNA Said**

À ma tendre mère, **BENAMER Houaria**

À mon époux bien-aimé, **Mohamed**

À mes frères, **Ben Adel et Sidahmed**

À ma très chère sœur, **Yamina**

À mes belles-sœurs, **Fatima et Samira**

À mes ami(e)s fidèles

À toutes les personnes que je connais de près ou de loin

Et à tous les autres membres de ma famille.

Aicha

Dedicace

J'ai exprimé ma profonde gratitude et mes remerciements à mes parents et ma famille, pour leur soutien constant, leurs encouragements et leur patience, qui m'ont permis d'arriver là où je suis aujourd'hui.

Mes mots de remerciement s'adressent également à mes chères camarades : **Insaf**, **Chiraz**, **Nouhad**, **Khadija** et **Fatima**, pour leur esprit de collaboration, leur soutien et les beaux moments partagés durant nos années d'études.

Chaima

ملخّــص:

تمثّل التجزئة الدلالية أحد المحاور الرئيسية في مجال الرؤية الحاسوبية، حيث تهدف إلى تخصيص تسمية دقيقة لكل بكسل من الصورة، مما يساعد على تحقيق فهم عميق ومتكامل لمحتواها البصري. ومنذ الانتشار الواسع للتعلّم العميق، شهد هذا المجال تقدّمًا كبيرًا بفضل ظهور الشبكات العصبية التلافيفية(CNN) ، التي أظهرت كفاءة عالية في استخراج وتمثيل الخصائص البصرية.

ومن أبرز هذه النماذج نموذج **U-Net**، الذي طُوّر أصلاً لإجراء التجزئة الدقيقة في المجال الطبي، وتم اعتماده كنقطة انطلاق لهذا العمل. حيث أُدخلت تعديلات على بنيته، وأضيفت طبقات مخصصة، بهدف تعزيز قدرته على معالجة البيئات البصرية كثيفة التفاصيل، على غرار المشاهد الحضرية التي توفّرها قاعدة البيانات **Cityscapes**، المعروفة بثرائها ودقّة شروحاتها.

وقد ساهمت هذه التحسينات المقترحة في زيادة دقّة النموذج بشكل ملحوظ، وفي جعل عملية التعلّم أكثر استقرارًا، وأسرع، وأفضل تكيّفًا مع تنوّع البيئات الحضرية الحديثة. يمثّل هذا العمل بذلك أساسًا منهجيًا متميزًا يمكن الاعتماد عليه عند تطوير نماذج مخصّصة للتجزئة الدقيقة للبيئات البصرية المعقّدة.

الكلمات المفتاحية : التعلّم العميق، التجزئة الدلالية، U-Net، الشبكات العصبية التلافيفية، المشاهد الحضرية، Cityscapes.

Résumé :

La segmentation sémantique est un champ majeur de la vision par ordinateur, visant à attribuer une étiquette précise à chaque pixel d'une image, afin de permettre une compréhension détaillée de son contenu visuel. Depuis l'essor majeur du deep learning, ce domaine a connu des avancées significatives grâce à l'introduction des réseaux de neurones convolutionnels (CNN), qui démontrent une efficacité remarquable pour l'extraction et la représentation des caractéristiques visuelles.

Parmi les modèles les plus populaires figure le U-Net, initialement conçu pour la segmentation précise en contexte médical, que nous avons adopté comme base pour ce travail. Des modifications structurelles ainsi que l'ajout de nouvelles couches spécifiques ont été introduits afin d'améliorer sa capacité à segmenter des environnements visuels denses tels que les scènes urbaines proposées par la base de données Cityscapes, reconnue pour la richesse et la précision de ses annotations.

Les améliorations proposées ont permis d'augmenter significativement la précision du modèle tout en rendant le processus d'apprentissage plus stable, plus rapide et mieux adapté à la diversité des environnements urbains modernes. Ce travail fournit ainsi une base méthodologique utile pour la conception de modèles adaptés à la segmentation précise des environnements visuels complexes.

Mots-clés : Deep Learning, Segmentation sémantique, U-Net, CNN, Scènes urbaines,

Cityscapes.

Abstract :

Semantic segmentation is a major area of computer vision that aims to assign a precise label to every pixel of an image, thereby enabling a deep and comprehensive understanding of its visual content. Since the rise of deep learning, this field has witnessed significant advances with the introduction of Convolutional Neural Networks (CNNs), which have demonstrated remarkable efficiency in extracting and representing visual features.

Among the most popular models is U-Net, initially designed for precise segmentation in medical contexts, which we have adopted as the foundation of this work. Structural modifications and additional dedicated layers have been introduced to enhance its ability to segment dense visual environments, such as the urban scenes found in the Cityscapes dataset, renowned for its richness and precise annotations.

These improvements have significantly increased the model's precision, while making the learning process more stable, faster, and better suited to the diversity of modern urban environments. This work thus provides a solid methodological basis for designing models adapted to the precise segmentation of complex visual environments.

Keywords: Deep Learning, Semantic Segmentation, U-Net, CNN, Urban Scenes,

Cityscapes.

Table de matière:

Table de matière: 1
List des figures 4
Liste des abréviations 6
Introduction générale 8
I. Chapitre 1 : Segmentation des images 12
1. Introduction 12
2. Image numérique 12
2.1. Définition d'image 12
2.2. Définition d'une image numérique 13
1.1. Images matricielles et vectorielles 13
1.2. Types d'images 14
1.3. Caractéristiques de l'image numérique 17
2. Introduction à la segmentation 22
2.1. Les différents types de segmentation d'image 23
2.2. Différentes approches de la segmentation 25
3. Conclusion : 35
II. Chapitre 2 : Apprentissage profond 38
1. Introduction 38
2. Définition du Deep Learning : 38
3. Applications du Deep Learning : 38
4. Historique et évolution du Deep Learning : 39
4. Réseaux de neurones : 40
4.1. Qu'est-ce qu'un réseau de neurones ? 40
4.2. Neurone biologique : 40
4.3. Du neurone biologique au neurone artificiel : 41
4.4. Perceptron : 41
4.5. Perceptron multicouche (MLP) : 43
4.6. Les types de réseaux de neurones artificiels : 44
4.7. Introduction aux réseaux de neurones convolutifs (CNN) : 45

4.8. Quelques architectures CNN pour la segmentation : 52
4.9. Apprentissage par transfert (Transfer Learning) 56
4.10. Sur-ajustement et Sous-ajustement : 57
5. Segmentation d'images urbaines par Deep Learning : 58
5.1. Définition : 58
5.2. Historique et évolution des bases de données urbaines pour la segmentation sémantique 59
6.3 État de l'art 59
6. Conclusion : 60
III. Chapitre 3 : Implémentation et analyse expérimentale 63
1. Introduction : 63
2. Jeu de données utilisé : 63
3. Préparation et structuration du jeu de données : 64
4. Répartition des données : 64
5. Chargement dans Google Colab : 65
6. Pipeline de prétraitement : 65
7. Environnement de développement et outils utilisés : 67
7.1. Langage de programmation Python : 67
7.2 Environnement Google Colab : 67
7.3. Bibliothèques utilisées : 68
8. Modèle U-Net proposé : 72
8.1. Particularités de l'implémentation 73
8.2. Paramètres et taille du modèle : 74
9. Entraînement du modèle : 74
9.1. Hyperparamètres utilisés 75
9.2. Stratégies d'optimisation 75
10. Analyse et discussion des résultats : 76
11. Comparaison avec d'autres modèles de segmentation : 79
12. Limites et difficultés rencontrées : 80
12.1. Contraintes liées à l'environnement de développement : 80
12.2. Contraintes liées aux données utilisées : 81
13. Interface graphique développée avec Gradio : 81
14. Conclusion : 82
Conclusion générale : 84

Bibliographie.. 86

List des figures

Figure I-1: Représentation matricielle d'une image numérique.[3] 13
Figure I-2: la différence entre Image pixellisée vs image lissée. 14
Figure I-3: Illustration d'une image binaire en noir et blanc. [6] 15
Figure I-4: Illustration d'une image monochrome en niveaux de gris.[7] 16
Figure I-5: Illustration d'une image polychrome en espace RGB. [7] 16
Figure I-6: Visualisation 3D de l'espace RGB. 17
Figure I-7: Structure image en couleur. [7] 17
Figure I-8: Image présentant l'affichage des pixels. [8] 18
Figure I-9: Dimension d'une image, (a) la dimension 64*64, (b) la dimension 32*32. [9] 19
Figure I-10: La résolution d'une image, (a) pleine résolution, (b) (½) résolution, (c) ¼ résolution. [9] 19
Figure I-11 : Les trois primaires : Rouge, Vert, et Bleu. [9] 20
Figure I-12: Histogramme et palette associés à une image. [10] 21
Figure I-13: Effet du bruit sel et poivre sur une image du Mémorial du Martyr(Algérie). 21
Figure I-14: Comparaison entre l'image originale et son contour détecté. [7] 22
Figure I-15: Segmentation sémantique. 23
Figure I-16: Segmentation d'instance. 24
Figure I-17: Segmentation panoptique. 25
Figure I-18: Extension graduelle des régions. [7] 26
Figure I-19: Divisions progressives en blocs. [7] 27
Figure I-20: Fusion progressive des blocs présentant une similarité avec le bloc initial de l'image.[7] 28
Figure I-21: Seuillage binaire basé sur l'histogramme. 29
Figure I-22: Multi-seuillage basé sur l'histogramme. 29
Figure I-23: Détection des contours sur l'image de la chèvre. 31
Figure I-24: Contours extraits à l'aide du filtre Laplacien à partir de l'image originale. [7] 32
Figure I-25: Illustration de l'application du filtre de Canny. [7] 33
Figure I-26: Détection des contours et des dérivées associées. 33
Figure I-27: Détection des contours à l'aide des différents filtres. [7] 34
Figure II-1: Comparaison entre un neurone biologique et un neurone artificiel. [32] 41
Figure II-2: Structure et fonctionnement d'un perceptron simple. [33] 42
Figure II-3: Architecture d'un perceptron multicouche (MLP). [37] 43
Figure II-4: Architecture des réseaux neuronaux récurrents et variantes LSTM/GRU. [38] 44
Figure II-5: Structure générale d'un réseau de neurones convolutifs (CNN). [43] 46
Figure II-6: Fonctionnement d'une couche de convolution. [36] 47
Figure II-7: Illustration des opérations de pooling.[49] 49
Figure II-8: Exemples de fonctions d'activation utilisées dans les réseaux de neurones. 50
Figure II-9: Exemples d'architectures CNN utilisées pour la segmentation. [58] 53
Figure II-10: Architecture U-Net. [59] 54
Figure II-11: Architecture SegNet. [61] 55

Figure II-12: Architecture VGG (Visual Geometry Group). [63] 55
Figure II-13 : Schéma de l'architecture DeepLabv3. [64] 56
Figure II-14: Comparaison entre sur-ajustement (overfitting) et sous-ajustement (underfitting). [68] 58
Figure III-1 : Logo Python. [96] 67
Figure III-2 : Logo Google Colab. [97] **68**
Figure III-3 : Logo TensorFlow. [98] 68
Figure III-4 : Logo Keras. [99] 69
Figure III-5 : Logo NumPy. [100] 69
Figure III-6 : Logo Pillow (PIL). [101] 70
Figure III-7 : Logo Matplotlib. [102] 70
Figure III-8 : Logo Gradio. [103] 71
Figure III-9 : Logo Scikit-learn. [104] 71
Figure III-10 : Logo Seaborn. [105] 71
Figure III-11 : Logo OpenCV. 72
Figure III-12 : Répartition des données en ensembles d'entraînement, validation et test. 65
Figure III-13 : Montage du Google Drive dans l'environnement Google Colab. 65
Figure III-14 : Comparaison entre l'architecture U-Net standard et le modèle U-Net proposé. 74
Figure III-15 : Courbes de perte et de précision sur les ensembles d'entraînement et de validation. . 76
Figure III-16 : Matrice de confusion sur les prédictions du modèle. 77
Figure III-17 : Exemples de segmentation sur des images de test (de gauche à droite : image originale, masque réel, prédiction du modèle). 78
Figure III-18 : Interface avant prédiction. 81
Figure III-19 : Exemple de prédiction (à gauche : image originale, à droite : image segmentée). 82

Liste des abréviations

CNN : Convolutional Neural Network (Réseau de Neurones Convolutifs)

U-Net : Architecture CNN en forme de U, conçue pour la segmentation

FCN : Fully Convolutional Network

ReLU: Rectified Linear Unit (fonction d'activation)

BN: Batch Normalization

ASPP : Atrous Spatial Pyramid Pooling

RNN : Recurrent Neural Network (Réseau de Neurones Récurrent)

LSTM : Long Short-Term Memory

GRU : Gated Recurrent Unit

AE : Autoencoder

MAE : Mean Absolute Error (Erreur Absolue Moyenne)

MSE : Mean Squared Error (Erreur Quadratique Moyenne)

SGD : Stochastic Gradient Descent (Descente de Gradient Stochastique)

Adam : Adaptive Moment Estimation

RGB : Red, Green, Blue (rouge, vert, bleu)

MLP : Multilayer Perceptron (Perceptron Multi-couches)

ANN : Artificial Neural Network

DNN : Deep Neural Network

VGG : Visual Geometry Group

Cityscapes : Base de données d'images urbaines annotées

ML : Machine Learning (Apprentissage automatique)

DL : Deep Learning (Apprentissage profond)

IA : Intelligence Artificielle

GPU : Graphics Processing Unit (Processeur graphique)

Introduction générale

Introduction générale

Aujourd'hui, l'explosion des technologies numériques a profondément transformé notre manière de percevoir, capturer et traiter le monde qui nous entoure. Des milliards d'images sont générées quotidiennement par les smartphones, les caméras de surveillance, les drones, les satellites et d'autres dispositifs. Dans cet environnement saturé de données visuelles, comprendre automatiquement le contenu des images représente un défi majeur, aussi bien pour les applications industrielles que pour les besoins sociétaux. La segmentation d'images, et plus particulièrement la segmentation sémantique, constitue un élément clé de cette démarche, en permettant de décomposer une image en régions significatives afin de faciliter son analyse et son interprétation automatique.

Les méthodes classiques de segmentation, reposant sur les contours, les régions homogènes ou le seuillage, ont longtemps constitué le socle des approches en traitement d'images. Toutefois, ces techniques se sont révélées insuffisantes face à la complexité des scènes réelles, notamment dans les environnements urbains, où les variations d'échelle, les occlusions et la richesse visuelle exigent des solutions plus performantes. L'apparition du Deep Learning et, en particulier, des réseaux de neurones convolutifs (CNN) a marqué une avancée décisive dans ce domaine. Parmi ces architectures, U-Net, initialement conçue pour la segmentation d'images médicales, a démontré une grande efficacité pour traiter des scènes complexes telles que les paysages urbains.

Dans un contexte d'actualité où les véhicules autonomes, les systèmes de vidéosurveillance intelligents et les villes connectées deviennent de plus en plus répandus, la capacité à segmenter avec précision les images urbaines revêt une importance stratégique. Ces technologies nécessitent une compréhension fine, rapide et fiable de l'environnement pour garantir la sécurité et la qualité des décisions automatisées. De plus, la quantité massive de données visuelles générées chaque jour dans les espaces urbains crée un besoin croissant de solutions robustes pour leur traitement et leur exploitation.

Ce mémoire s'inscrit dans cette dynamique, en explorant l'utilisation d'une architecture U-Net adaptée à la segmentation d'images urbaines. L'objectif est d'améliorer les performances de segmentation en exploitant les capacités des réseaux convolutifs sur la base de données Cityscapes, reconnue pour la richesse et la complexité de ses scènes urbaines. À travers ce

travail, nous avons développé un modèle personnalisé, en tenant compte des défis spécifiques posés par la segmentation d'images de ce type.

L'étude de la segmentation d'images urbaines par Deep Learning présente un intérêt multiple. Sur le plan scientifique, elle contribue à l'évolution des méthodes de segmentation en enrichissant les architectures existantes. Sur le plan technologique, elle permet de développer des solutions applicables à des domaines concrets tels que la mobilité autonome, les villes intelligentes ou les systèmes avancés de surveillance. Enfin, elle offre un apport sociétal en contribuant à la mise en place de systèmes plus sûrs et plus performants, capables d'interpréter efficacement des environnements urbains de plus en plus complexes.

Dans quelle mesure l'amélioration d'une architecture U-Net permet-elle d'optimiser la segmentation sémantique des images urbaines face aux défis que posent la diversité des scènes et la densité des informations visuelles ? C'est autour de cette question que s'articule ce travail. L'approche adoptée repose sur le développement et l'évaluation d'un modèle adapté aux spécificités de la base Cityscapes, afin de répondre aux exigences des systèmes modernes de traitement d'images.

Notre champ d'étude est centré sur la segmentation d'images de scènes urbaines européennes issues de la base de données Cityscapes, en s'appuyant sur les avancées récentes en Deep Learning et en réseaux convolutifs. Le cadre temporel de cette recherche correspond à l'évolution actuelle des technologies dans le domaine de l'analyse d'images urbaines.

Ce mémoire est structuré de la manière suivante :

- Chapitre 1 : présentation des notions fondamentales relatives à l'image numérique et aux différentes techniques de segmentation, avec une mise en perspective des approches classiques.
- Chapitre 2 : exploration des apports du Deep Learning et des réseaux de neurones convolutifs dans le domaine de la segmentation, en mettant en lumière le rôle des architectures comme U-Net.
- Chapitre 3 : présentation de notre contribution à travers le développement et l'évaluation d'une architecture U-Net adaptée à la segmentation d'images urbaines sur la base Cityscapes.

Nous terminons ce rapport par une conclusion générale, dans laquelle nous précisons les résultats obtenus, les limites rencontrées et les perspectives offertes par ce travail.

Chapitre 1 :

Segmentation des images

Chapitre 1

I. Chapitre 1 : Segmentation des images

1. Introduction

La segmentation sémantique est une étape essentielle en vision par ordinateur. Elle permet de diviser une image en identifiant et en classifiant chaque région selon sa signification ou sa fonction. En attribuant une étiquette à chaque pixel selon la classe d'objet correspondante, cette technique facilite l'analyse automatique d'images complexes, notamment dans des contextes tels que les environnements urbains, la médecine ou la conduite autonome.

Ce chapitre présente les concepts fondamentaux liés à l'image numérique et à la segmentation. Nous décrivons la structure et les propriétés des images numériques, en distinguant les principaux types d'images (binaires, en niveaux de gris, en couleurs) et en examinant des éléments clés tels que la résolution, la taille, le contraste, le bruit et la distribution des intensités à travers l'histogramme.

Ensuite, nous présentons les principales catégories de segmentation d'images, en insistant sur la segmentation sémantique. Celle-ci est différenciée de la segmentation d'instance et de la segmentation panoptique, dont nous précisons les objectifs, les spécificités et les domaines d'application.

Enfin, nous passons en revue les méthodes courantes de segmentation : celles fondées sur les contours, sur les régions homogènes et sur les techniques d'apprentissage automatique, y compris l'apprentissage profond. Chaque approche sera présentée en termes de principes, d'avantages et de limites.

Cette introduction vise à établir les fondations indispensables à la compréhension des techniques modernes de segmentation sémantique, tout en mettant en évidence leur rôle clé dans les systèmes intelligents d'interprétation visuelle.

2. Image numérique

2.1. Définition d'image

Une image est une représentation visuelle d'une scène réelle ou artificielle. Elle peut être captée par un dispositif (caméra, scanner, appareil photo, satellite) ou générée par des moyens informatiques. Selon le mode d'acquisition, elle peut être analogique (comme une pellicule) ou numérique (comme un fichier).[1]

2.2. Définition d'une image numérique

Contrairement aux images dessinées sur papier ou capturées sur pellicule, les images traitées par un ordinateur sont numériques, c'est-à-dire représentées sous forme de suites de bits. Une image numérique est constituée d'une grille régulière d'éléments de taille fixe appelés pixels, chacun portant une valeur correspondant à un niveau de gris ou à une couleur. Cette valeur peut être soit prélevée directement dans une image réelle, soit calculée à partir d'une description interne de la scène à représenter.

La numérisation d'une image correspond au processus de conversion d'un signal analogique en une représentation numérique. Le résultat est une matrice bidimensionnelle de valeurs numériques notée *f(x, y)*, où :

- *x* et *y* représentent les coordonnées cartésiennes d'un point de l'image.[2]

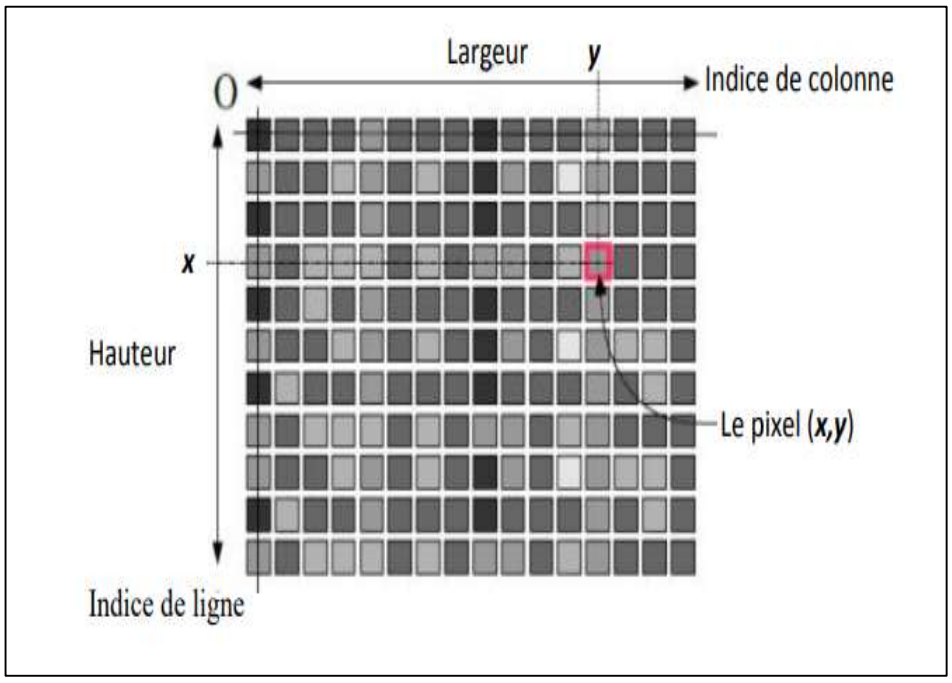


Figure I-1: Représentation matricielle d'une image numérique.[3]

1.1. Images matricielles et vectorielles

Les images numériques se classent en deux grandes catégories : les images matricielles (ou bitmap/raster) et les images vectorielles.

- **Image matricielle :** elle est constituée d'une matrice de pixels, chaque pixel représentant un point de couleur. Ce type d'image est adapté aux photographies et aux rendus détaillés, mais son redimensionnement entraîne souvent une perte de qualité (effet de pixellisation). Les formats courants sont: JPG, GIF, PNG, etc.
- **Image vectorielle :** elle est définie à l'aide d'équations mathématiques décrivant des formes géométriques (lignes, courbes, polygones). Contrairement aux images matricielles, elle peut être redimensionnée sans perte de qualité, ce qui la rend idéale pour les logos, les illustrations et les conceptions graphiques. Les formats vectoriels les plus utilisés sont : AI, EPS, PDF.

La figure ci-dessous illustre les différences entre ces deux types d'images.[4]

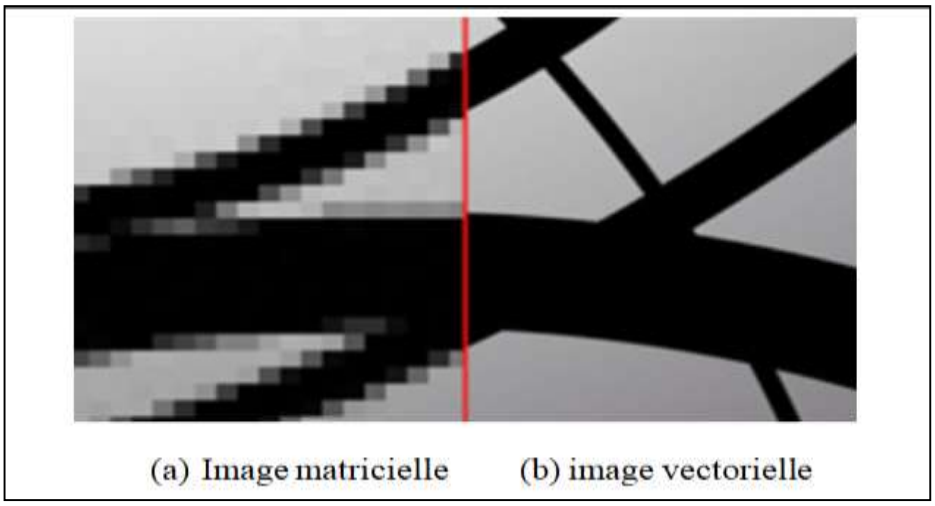


Figure I-2: la différence entre Image pixellisée vs image lissée.

1.2. Types d'images

On distingue trois principales catégories d'images numériques :

- **Image binaire** : composée uniquement de deux couleurs, généralement une pour l'arrière-plan et une pour l'avant-plan.
- **Image en niveaux de gris (monochrome)** : constituée de variations d'une même teinte, typiquement des nuances de gris.
- **Image en couleurs (polychrome)** : également appelée « vraies couleurs », elle repose sur la combinaison de plusieurs teintes**.**[3]

2.4.1. Images binaires (noir et blanc)

Une image binaire est une image dans laquelle chaque pixel peut prendre uniquement l'une des deux valeurs possibles : noir absolu ou blanc absolu. Ce type d'image est fréquemment utilisé pour les documents textuels numérisés, comme les scans de livres ou de manuscrits, où deux teintes suffisent à distinguer le contenu.[5]

Figure I-3: Illustration d'une image binaire en noir et blanc.**[6]**

2.4.2. Images en niveaux de gris (monochromes)

Les images en niveaux de gris sont composées de différentes teintes de gris, généralement 256 nuances. Par convention :

- La valeur **0** représente le noir absolu,
- La valeur **255** représente le blanc absolu.

Chaque niveau est codé sur 8 bits, mais d'autres codages existent : 16 bits pour une précision accrue, 2 bits pour un nombre réduit de nuances. Un codage sur 1 bit ne permet que deux niveaux (0 ou 1), ce qui correspond à une image binaire.[5]

Figure I-4: Illustration d'une image monochrome en niveaux de gris.[7]

2.4.3. Images en couleurs (Polychromes)

Les images en couleurs reposent sur la synthèse additive, où le mélange de couleurs permet d'en créer de nouvelles. La plupart sont basées sur trois couleurs primaires : Rouge, Vert et Bleu (RVB ou RGB).

Chaque composante est généralement codée sur 8 bits, ce qui donne : 3×8=24 bits par pixel, permettant une large gamme de couleurs (de 0 à 255 par canal).

Figure I-5: Illustration d'une image polychrome en espace RGB.[7]

L'espace couleur le plus utilisé en imagerie numérique est le RVB.[5]

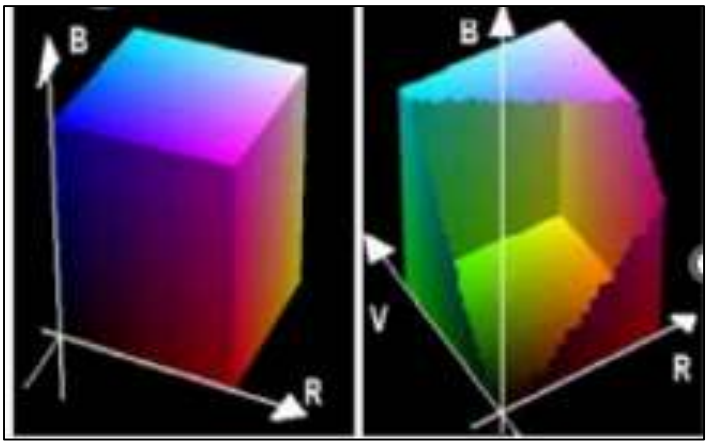


Figure I-6: Visualisation 3D de l'espace RGB.

Il existe plusieurs méthodes pour convertir une image RVB en niveaux de gris. La plus simple consiste à utiliser la moyenne arithmétique des trois composantes :

Gris = (Rouge + Vert + Bleu) / 3

Cette méthode attribue une même intensité aux trois canaux, produisant une image en niveaux de gris.

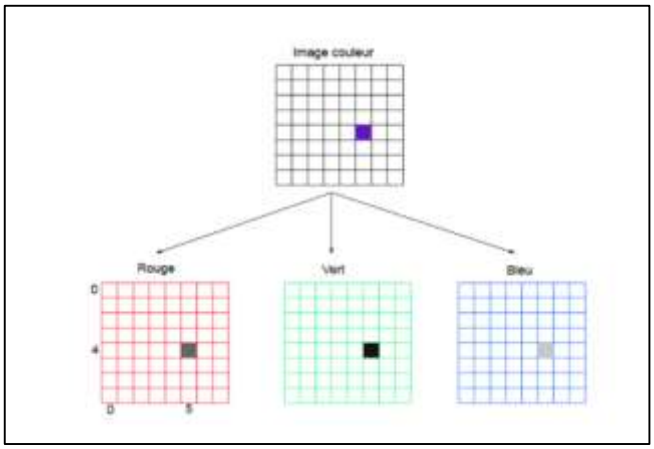


Figure I-7: Structure image en couleur.**[7]**

1.3. Caractéristiques de l'image numérique

Une image numérique est un ensemble structurant d'informations définies par plusieurs paramètres spécifiques qui permettent de la décrire, de la représenter et de la manipuler efficacement.

2.5.1. Le Pixel

Le mot "pixel" est issu de l'expression anglaise "Picture element" (élément d'image). Il représente le plus petit point composant une image numérique, traduisant une valeur correspondant à une intensité lumineuse.

Si le bit est l'unité d'information minimale d'un ordinateur, le pixel en est l'unité graphique de base, traitée par les logiciels et les matériels d'affichage. Par exemple, la lettre "A" peut être affichée sous forme d'un ensemble de pixels (voir figure ci-dessous).

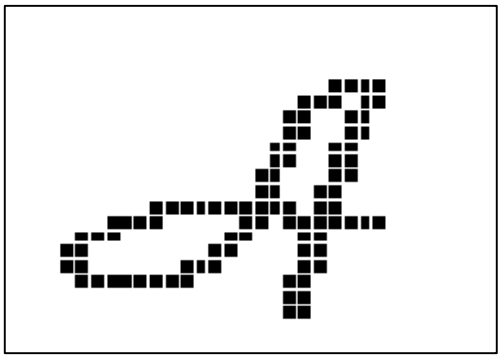


Figure I-8: Image présentant l'affichage des pixels.**[8]**

Un pixel peut avoir deux types de valeurs :

- Un **scalaire**, lorsqu'il représente un niveau de gris, soit une intensité unique.
- Un **vecteur**, lorsqu'il représente une couleur composée de plusieurs canaux (rouge, vert, bleu - RVB)[7].

2.5.2. La dimension d'une image

Une image numérique est représentée sous forme de matrice, composée de lignes (hauteur) et de colonnes (largeur) de pixels. Sa dimension correspond donc à ces deux valeurs : largeur × hauteur. Le nombre total de pixels est obtenu en multipliant ces deux dimensions.[8]

Figure I-9: Dimension d'une image, (a) la dimension 64*64, (b) la dimension 32*32.**[9]**

2.5.3. La résolution d'une image

La résolution d'une image définit son niveau de détail. Elle correspond au nombre de pixels par unité de longueur et s'exprime en dpi (dots per inch) ou ppp (points par pouce). Plus la résolution est élevée, plus l'image est précise et riche en détail[8].

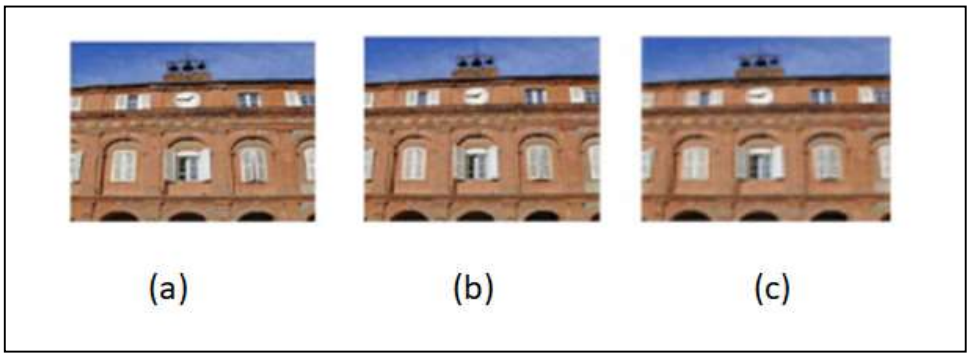


Figure I-10: La résolution d'une image, (a) pleine résolution, (b) (½) résolution, (c) ¼ résolution.[9]

Remarques :

- 1 pouce = 2,54 cm
- La correspondance entre pouce et pixels dépend de la résolution. Par exemple, à 100 dpi, 1 pouce correspond à 100 pixels.
- 1 inch (unité anglo-saxonne) est équivalent à 2,54 cm, soit 1 pouce[7].

2.5.4. La couleur

La couleur est l'un des éléments les plus distinctifs du contenu visuel d'une image. Elle joue un rôle essentiel dans l'analyse et la reconnaissance des objets. Plusieurs

descripteurs permettent de la caractériser, notamment les espaces de couleur et les histogrammes.

- **Espace de couleur :** Chaque pixel peut être représenté comme un point dans un espace de couleur tridimensionnel.
  - **RVB (Rouge, Vert, Bleu)** : Utilisé principalement pour l'affichage numérique.
  - **TSL (Teinte, Saturation, Luminosité)** : Pratique pour la manipulation des couleurs selon la perception humaine.

- **Modèle RVB (RGB)** : Il repose sur la synthèse additive des trois couleurs primaires. Il permet de générer une large palette de couleurs et constitue la base des systèmes de capture et d'affichage numériques.

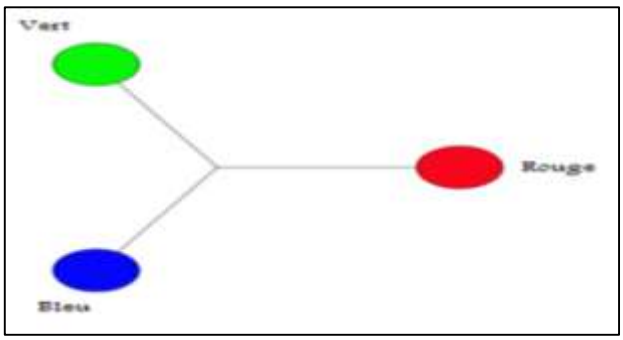


Figure I-11 : Les trois primaires : Rouge, Vert, et Bleu.**[9]**

2.5.5. Histogramme d'une image

L'histogramme des niveaux de gris constitue un outil fondamental en analyse d'image. Il offre une représentation graphique de la répartition des intensités, en indiquant la fréquence d'apparition de chaque niveau de gris dans l'image. Cette distribution permet d'évaluer la dynamique de l'image, c'est-à-dire la manière dont les intensités sont étalées sur l'ensemble des pixels.

On distingue généralement trois formes d'histogrammes :

- **Histogramme unimodal :** il présente un seul pic, représentant principalement soit l'objet, soit le fond.
- **Histogramme bimodal :** il comporte deux pics distincts, correspondant à un objet bien séparé de son fond.

- **Histogramme multimodal :** il présente plusieurs pics séparés par des creux, suggérant la présence de plusieurs objets ou zones d'intensité différente dans l'image.[5]

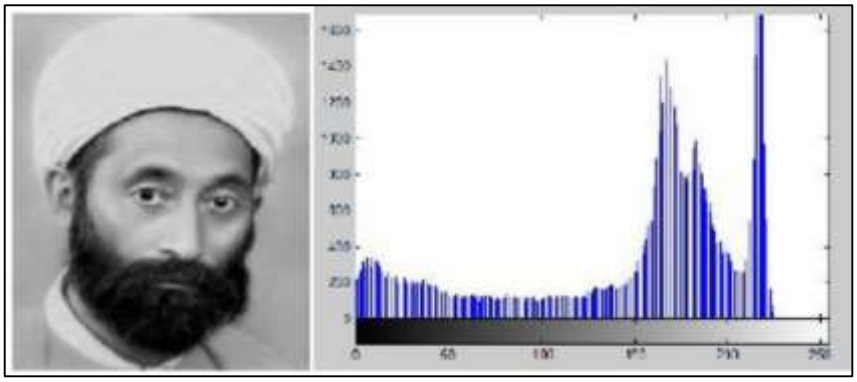


Figure I-12: Histogramme et palette associés à une image.**[10]**

2.5.6. Le bruit

Le bruit (ou parasite) dans une image se manifeste par des variations soudaines et anormales de l'intensité lumineuse d'un pixel par rapport à ses voisins. Ce phénomène est généralement causé par des perturbations physiques ou électroniques, notamment dues à l'éclairage, aux capteurs optiques, ou aux circuits de traitement d'image lors de la capture ou de la transmission.[11]

2.5.6.1. Types de bruit

Les types de bruit les plus courants sont :

- Bruit Gaussian (bruit blanc)
- Bruit impulsionnel (poivre et sel)[11]

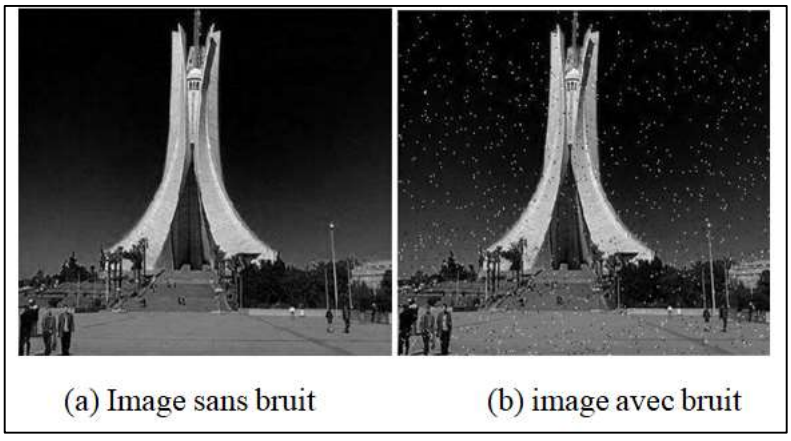


Figure I-13: Effet du bruit sel et poivre sur une image du Mémorial du Martyr(Algérie).

2.5.6.2. Le contour

L'extraction des contours est une étape essentielle dans de nombreuses applications d'analyse d'images. Ils constituent des repères clés permettant la délimitation, l'interprétation et la reconnaissance des objets présents dans une scène. Ils résultent généralement de :

- discontinuités de réflectance (variations de texture ou d'éclairage).
- discontinuités de profondeur (bords des objets).

Dans l'image, ces contours apparaissent sous forme de variations brusques de l'intensité lumineuse**.**[11]

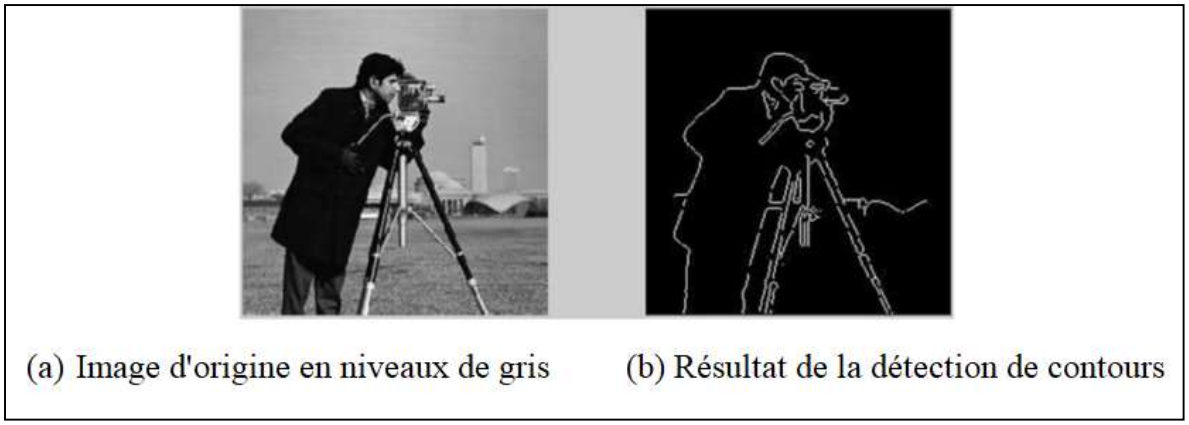


Figure I-14: Comparaison entre l'image originale et son contour détecté**.[7]**

2. Introduction à la segmentation

Selon Shapiro et Stockman, la segmentation consiste à diviser une image en plusieurs composantes distinctes afin d'en faciliter l'analyse. Ce processus peut être abordé de deux manières complémentaires :

- La première approche vise à détecter les contours qui séparent les différentes régions homogènes, en s'appuyant sur des discontinuités d'intensité ou de texture.
- La seconde repose sur le regroupement de pixels présentant des caractéristiques similaires (intensité, couleur, texture), afin de former des régions cohérentes.

Autrement dit, chaque pixel de l'image se voit attribuer une valeur ou un ensemble de caractéristiques, et les pixels similaires sont regroupés pour constituer des zones homogènes. En combinant ces zones segmentées, on obtient une représentation structurée de l'image, où les objets appartenant à une même catégorie sont clairement identifiés**.**[12]

2.1. Les différents types de segmentation d'image

La segmentation d'image peut être classée en trois grandes catégories : la segmentation sémantique, la segmentation d'instance et la segmentation panoptique. Chacune de ces approches présente des objectifs spécifiques, des méthodes particulières et des domaines d'application distincts.

2.1.1. Segmentation sémantique

La segmentation sémantique consiste à attribuer une étiquette de classe à chaque pixel de l'image, en regroupant les pixels partageant une même signification. Par exemple, tous les pixels représentant des arbres sont étiquetés comme "arbre", et tous ceux représentant des véhicules sont associés à la classe "véhicule", sans distinction entre les différentes instances.

Cette approche présente l'avantage d'être relativement simple à mettre en œuvre et peu coûteuse en calcul, car elle ne nécessite qu'une seule étiquette par pixel. Elle est particulièrement adaptée à la compréhension globale de scènes, comme l'étiquetage d'images, l'analyse de scènes urbaines ou la classification basée sur le contenu sémantique.

Parmi les techniques courantes, on retrouve les réseaux entièrement convolutionnels (FCN), U-Net et DeepLab. Toutefois, cette méthode peut montrer ses limites dans les contextes où une analyse fine des objets individuels est requise.

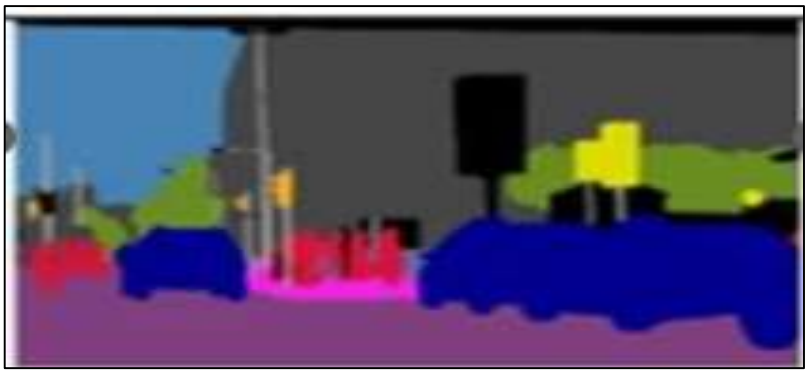


Figure I-15: Segmentation sémantique.

2.1.2. Segmentation d'instance

La segmentation d'instance va plus loin en distinguant non seulement les classes d'objets, mais aussi chaque instance individuelle au sein d'une même classe. Par exemple, dans une image contenant plusieurs voitures, cette méthode permet de

segmenter chaque voiture séparément, même si elles appartiennent toutes à la même classe "voiture".

Cette approche est plus complexe et plus exigeante en termes de calcul, car elle implique à la fois la détection et la segmentation fine de chaque objet. Elle est utile dans des applications comme le comptage d'objets, le suivi d'instances ou la segmentation sémantique à un niveau plus granulaire.

Les architectures les plus utilisées incluent Mask R-CNN, ou encore des FCN combinés à des modules de détection. Toutefois, cette méthode peut être mise en difficulté par des objets partiellement occultés ou fortement superposés.

Figure I-16: Segmentation d'instance.

2.1.3. Segmentation panoptique

La segmentation panoptique est une approche hybride qui combine les avantages de la segmentation sémantique et de la segmentation d'instance. Elle vise à fournir une compréhension complète d'une scène en segmentant à la fois les classes sémantiques (routes, bâtiments, trottoirs, etc.) et les instances individuelles (piétons, véhicules…).

C'est la méthode la plus avancée et la plus exigeante en termes de calcul et de labellisation, mais elle est idéale pour les projets nécessitant une compréhension globale et précise de l'environnement, comme les systèmes de conduite autonome.

Des techniques comme Panoptic FPN, Panoptic-DeepLab ou encore le Module de Segmentation Panoptique (PSM) sont utilisées dans ce domaine. L'annotation des données est complexe car elle doit inclure à la fois les aspects sémantiques et les instances spécifiques.[13]

Figure I-17: Segmentation panoptique.

2.2. Différentes approches de la segmentation

La segmentation constitue une étape essentielle du traitement d'image. De nombreuses méthodes ont été développées au fil des années, que l'on peut regrouper en trois grandes catégories **:**

- La segmentation basée sur les régions (region-based segmentation)
- La segmentation basée sur les contours (edge-based segmentation)
- La segmentation basée sur le seuillage (threshold-based segmentation)[14]

2.2.1. Approche par régions

L'approche par régions consiste à diviser une image en zones homogènes, en regroupant les pixels adjacents partageant des propriétés communes telles que la couleur, l'intensité ou la texture.

Le processus repose sur un critère d'homogénéité qui guide l'agrégation progressive des pixels jusqu'à ce que chaque pixel appartienne à une région donnée. Cette méthode s'appuie sur les caractéristiques internes des zones de l'image pour conduire la segmentation.Parmi les techniques associées, on retrouve la croissance de région, la division et la fusion de régions, qui seront détaillées dans les sous-sections suivantes. [15]

3.2.1.1. Croissance de région (RegionGrowing)

La croissance de région est une technique de segmentation qui étend progressivement des régions à partir de pixels initiaux appelés germes. Le processus commence par considérer chaque pixel comme une région indépendante, puis des regroupements sont effectués sur la base de deux critères : la similarité (par exemple des niveaux de gris) et l'adjacence spatiale.

Un critère de similarité couramment utilisé est la variance intra-régionnelle, qui doit rester inférieure à un certain seuil. La région s'étend tant que les pixels voisins respectent les critères d'homogénéité définis. Lorsqu'aucun pixel voisin ne satisfait ces conditions, un nouveau germe est sélectionné, et le processus recommence jusqu'à ce que l'ensemble de l'image soit segmenté. [16]

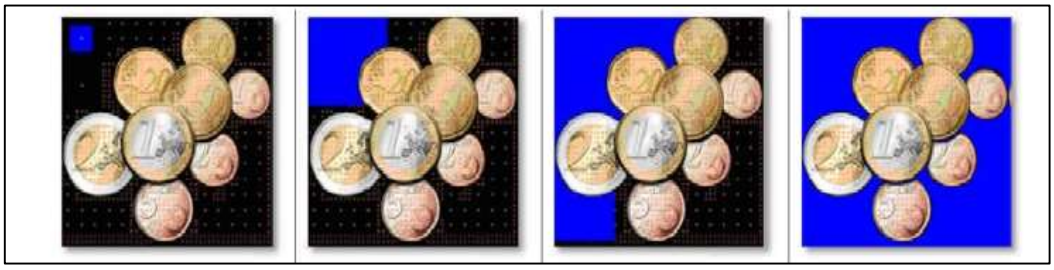


Figure I-18: Extension graduelle des régions.**[7]**

Avantages et limites de la croissance de région :

La croissance de région présente plusieurs avantages notables :

- Sa mise en œuvre est simple et rapide.
- Elle permet de segmenter des objets à la topologie complexe.
- Elle conserve fidèlement la forme des régions segmentées.

Toutefois, cette méthode comporte également certaines limites :

- Le résultat dépend fortement du choix initial des germes et du critère d'homogénéité.
- Une mauvaise sélection des germes ou un critère de similarité mal défini peut entraîner une sous-segmentation (régions trop vastes) ou une sur-segmentation (régions trop fragmentées).
- Certains pixels peuvent rester non classés s'ils ne satisfont aucun critère de regroupement.

3.2.1.2. Segmentation par fusion de régions (Merging)

Les méthodes de fusion (*regionmerging*) adoptent une approche ascendante, dans laquelle tous les pixels de l'image sont analysés. Pour chaque voisinage

d’un pixel, un prédicat P est évalué : si ce dernier est vérifié, les pixels concernés sont alors regroupés au sein d’une même région.

Malgré son efficacité, cette approche présente deux principaux inconvénients :

- Elle dépend fortement du critère de fusion, dont le mauvais paramétrage peut affecter la qualité du résultat.
- Elle peut entraîner une sous-segmentation, lorsque des régions distinctes sont fusionnées à tort.[10]

3.2.1.3. Segmentation par division de régions (Split)

La segmentation par division repose sur un processus descendant (top-down), qui consiste à subdiviser l’image en régions plus petites selon un critère d’homogénéité.

L’algorithme débute en considérant l’image entière comme une région initiale. Celle-ci est ensuite divisée en sous-régions. Chaque sous-région est évaluée à son tour, puis éventuellement redécoupée, jusqu’à ce que toutes les zones satisfassent le critère d’homogénéité défini.[17]

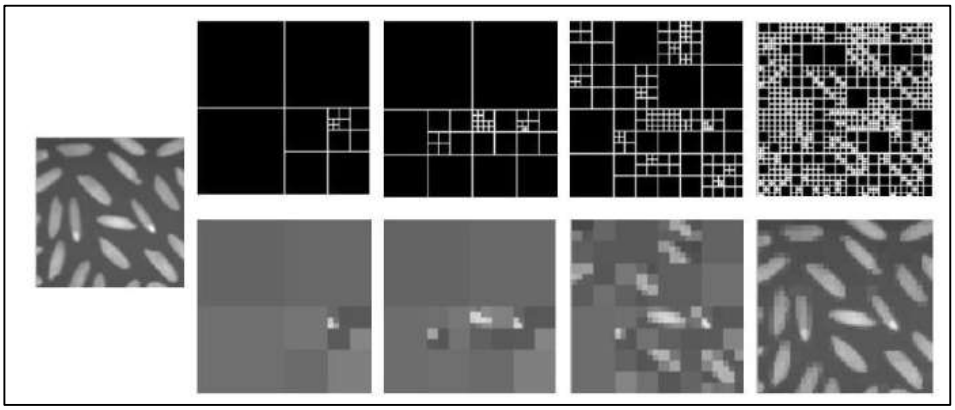


Figure I-19: Divisions progressives en blocs.**[7]**

3.2.1.4. Segmentation par division-fusion (Split and Merge)

Cette méthode combine les deux approches précédentes. L’image est d’abord divisée récursivement en régions élémentaires homogènes, puis les régions voisines présentant des caractéristiques similaires sont fusionnées selon un prédicat de regroupement.

Le processus débute généralement en traitant chaque pixel ou petit bloc comme une région distincte. Ensuite, les régions adjacentes sont progressivement

fusionnées si elles satisfont simultanément des critères tels que la proximité spatiale et la similarité d'intensité.

L'algorithme s'arrête lorsque plus aucune fusion n'est possible selon les règles établies.[15]

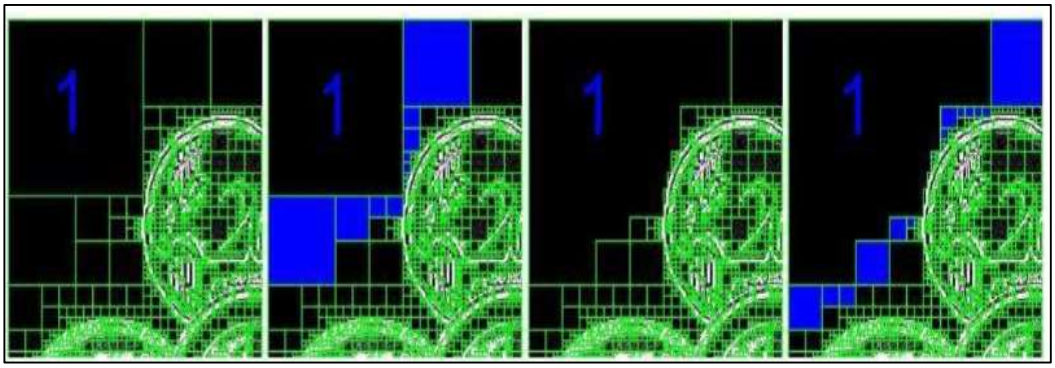


Figure I-20: Fusion progressive des blocs présentant une similarité avec le bloc initial de l'image.**[7]**

Limites de la méthode division-fusion :

Cette méthode présente plusieurs limitations, principalement à trois niveaux :

- Les régions segmentées ne correspondent pas toujours fidèlement aux objets réels présents dans l'image.
- Les contours obtenus sont souvent flous ou imprécis, et ne coïncident pas exactement avec les véritables frontières des objets.
- Il est souvent difficile de définir des critères pertinents pour regrouper les pixels, ou pour décider de la fusion ou de la division des régions.[15]

3.2.2. Segmentation par seuillage :

3.2.2.1. Définition du seuillage :

Le seuillage (thresholding en anglais) est une technique simple et largement utilisée en segmentation d'images, visant à séparer les objets de leur arrière-plan. Elle repose sur l'utilisation d'une ou plusieurs valeurs seuils, permettant de distinguer différentes zones de l'image en fonction de leur intensité lumineuse ou de leur niveau de gris.

Dans la majorité des cas, ces seuils sont choisis en optimisant une fonction objectif spécifique à la nature de l'image.

On distingue principalement deux types de seuillage :

- **Le seuillage simple (ou binaire)** : il divise l'image en deux classes (objet et fond), produisant une image binaire.

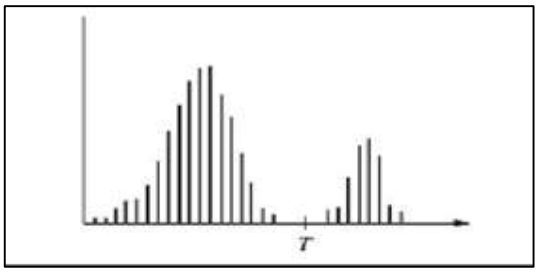


Figure I-21: Seuillage binaire basé sur l'histogramme.

- **Le multi-seuillage (ou seuillage multi-niveaux)** : utilisé pour les images comportant plusieurs objets ayant des niveaux de luminance différents. Dans ce cas, plusieurs seuils sont nécessaires, et le résultat est une image comportant *n+1* classes pour *n* seuils.

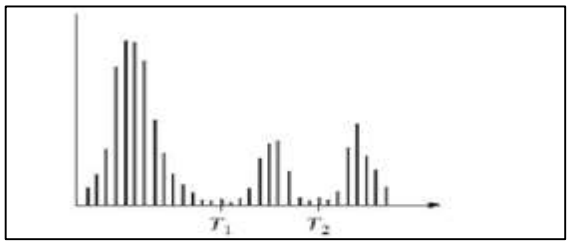


Figure I-22: Multi-seuillage basé sur l'histogramme.

La segmentation par seuillage appliquée à l'histogramme constitue un cas particulier de segmentation par classification. Elle consiste à répartir les pixels en différentes classes selon leurs niveaux de gris, les frontières entre les classes étant déterminées par les valeurs seuils.

Il existe plusieurs stratégies de seuillage, parmi lesquelles :

- Le seuillage global : un seul seuil est appliqué à l'ensemble de l'image
- Le seuillage local (ou adaptatif) : le seuil varie selon des zones spécifiques de l'image.

- Le seuillage manuel : les seuils sont définis directement par l'utilisateur.

a) Seuillage global :

Le seuillage global applique un seul seuil à l'ensemble de l'image. Il consiste à comparer le niveau de gris de chaque pixel xi avec un seuil global fixe T (par exemple 127). En fonction de cette comparaison, la nouvelle valeur du pixel bi est déterminée comme suit :

- $b_i=255$ si $x_i \geq T$
- $b_i=0$ si $x_i<T$

Ce procédé permet de transformer l'image en une image binaire, où les pixels au-dessus du seuil deviennent blancs (255) et ceux en dessous deviennent noirs (0).[10]

b) Seuillage local (ou adaptatif) :

Le seuillage local, également appelé seuillage adaptatif, utilise un seuil variable calculé pour chaque région de l'image. Contrairement au seuillage global, cette méthode prend en compte les valeurs des pixels voisins afin de déterminer les seuils locaux, ce qui la rend plus adaptée aux images présentant des variations d'éclairage ou de contraste. Elle permet une segmentation plus précise dans des conditions hétérogènes.[16]

c) Seuillage manuel :

Dans le cas du seuillage manuel, l'opérateur définit lui-même la valeur seuil à appliquer à l'image. Cette approche repose sur l'intervention humaine pour sélectionner les régions pertinentes, selon des critères visuels ou contextuels. Elle est généralement utilisée dans des applications simples, ou lorsque l'expertise de l'utilisateur est requise pour identifier avec précision les objets à extraire.

d) Le seuillage automatique :

Le seuillage n'est pas toujours une opération manuelle ; il peut également être réalisé automatiquement. Le seuillage automatique repose sur l'analyse de la distribution en fréquence de l'histogramme des niveaux de gris de l'image. L'objectif des méthodes automatiques de seuillage est d'identifier deux "groupes" distincts sur l'histogramme et de déterminer la valeur qui permet de les séparer de manière optimale.[1]

3.2.3. Segmentation basée sur les contours

La segmentation basée sur les contours constitue un domaine largement exploré en traitement et analyse d'images. Les contours correspondent généralement aux limites des objets dans une scène, détectées par des variations rapides d'intensité ou de couleur.

Les premières méthodes de segmentation se fondent sur la détection de ces discontinuités visuelles, en utilisant des filtres dérivateurs, capables de mettre en évidence les contours.

Parmi les techniques classiques, on retrouve les opérateurs de gradient, l'opérateur Laplacien, ainsi que des filtres tels que Sobel, Prewitt et Roberts. Ces filtres reposent sur le calcul de différences finies pour détecter les transitions abruptes dans l'image. D'autres méthodes plus avancées utilisent des critères d'optimalité, comme les filtres de Canny et Deriche, qui intègrent une meilleure gestion du bruit et une localisation plus précise des contours.

Toutefois, ces techniques peuvent présenter certaines limites : elles peuvent produire des contours non fermés, bruités, ou ne pas détecter certaines frontières visuelles.[1]

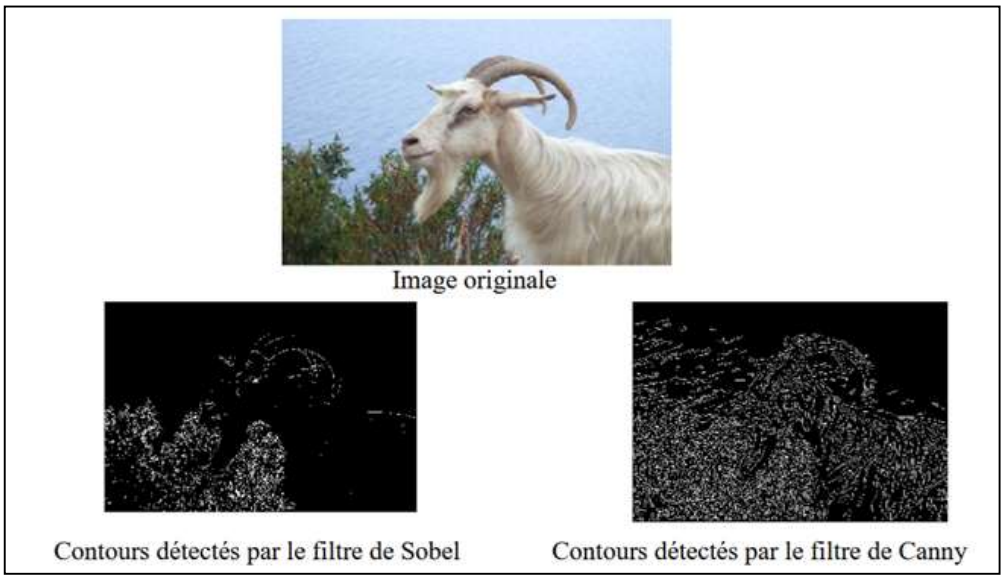


Figure I-23:Détection des contours sur l'image de la chèvre.

3.2.3.1. Le filtre Laplacien :

Le filtre Laplacien est un filtre de convolution conçu pour mettre en évidence les zones de l'image où l'intensité lumineuse varie rapidement. Il est particulièrement efficace pour faire ressortir les contours nets des objets, ce qui en

fait un outil privilégié dans les applications de reconnaissance de formes, d'abord dans le domaine militaire, puis dans des usages civils variés.[18]

Figure I-24: Contours extraits à l'aide du filtre Laplacien à partir de l'image originale.**[7]**

3.2.3.3. Approche de Canny et Deriche :

Le détecteur de Canny est l'un des algorithmes les plus utilisés pour la

détection de contours.

Il repose sur trois critères fondamentaux :

- **Robustesse à la détection** : garantir la détection des vrais contours malgré le bruit.
- **Précision de localisation** : positionner les contours au plus près de leur emplacement réel.
- **Unicité de la réponse** : éviter la détection multiple d'un même contour..[19]

À partir du filtre de Canny, Deriche a proposé une variante fondée sur des conditions initiales différentes. Son algorithme offre une implémentation plus efficace, tout en conservant les mêmes performances théoriques. Il est souvent privilégié pour des raisons de coût computationnel réduit.[20]

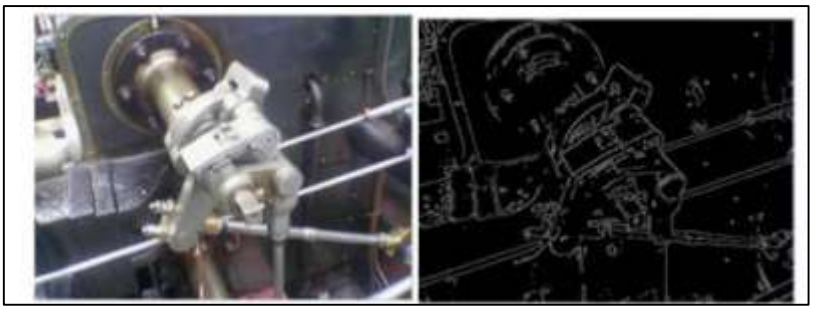


Figure I-25: Illustration de l'application du filtre de Canny.**[7]**

3.2.3.4. Les filtres de Prewitt, Sobel, Freeman et Kirsch :

Ces filtres, nommés d'après leurs concepteurs, ont tous été développés dans le même objectif : détecter avec précision les contours naturels présents dans une image, notamment les images issues de capteurs CCD. Initialement conçus pour des applications en vision nocturne, ces filtres se révèlent également précieux dans le domaine de l'astronomie, notamment pour l'analyse morphologique d'objets célestes comme les galaxies spirales.

Par exemple, le filtre de Sobel repose sur deux masques de convolution 3×3, l'un orienté selon l'axe horizontal (X) et l'autre selon l'axe vertical (Y). Ces deux noyaux, agissant comme des filtres de gradient, sont combinés pour produire une image finale mettant en évidence les contours.[20]

3.2.3.5. Méthodes dérivées :

Les méthodes dérivées sont couramment utilisées pour détecter les transitions d'intensité en appliquant des opérations de différenciation numérique, telles que la première et la deuxième dérivée. À chaque position de l'image, un opérateur est appliqué pour identifier les transitions significatives sur la base de l'attribut de discontinuité choisi. Le résultat est généralement une image binaire, dans laquelle les pixels appartenant à un contour sont nettement différenciés des autres. [15]

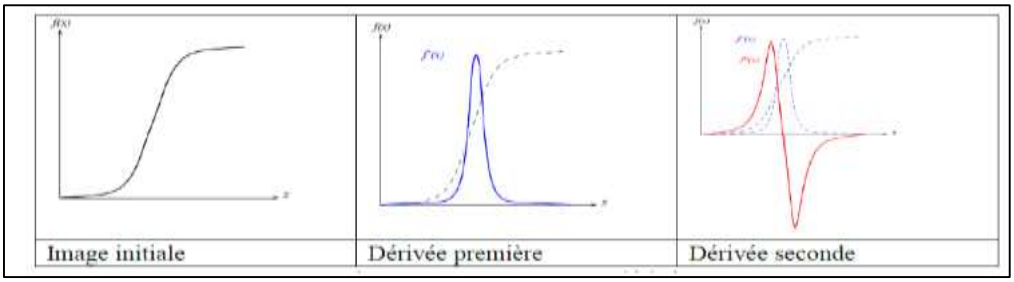


Figure I-26:Détection des contours et des dérivées associées.

3.2.3.6. L'approche du gradient :

Cette approche repose sur l'utilisation de la première dérivée pour calculer le gradient d'intensité à chaque pixel. Le gradient à un pixel donné est un vecteur caractérisé par son amplitude et sa direction. Plusieurs opérateurs permettent d'estimer ce gradient, parmi lesquels les masques de Roberts, Prewitt et Sobel sont les plus répandus.[15]

La figure suivante illustre les contours détectés à l'aide de ces filtres.[10]

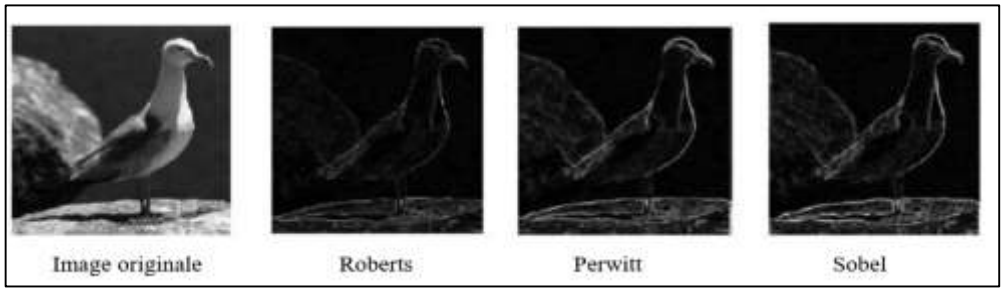


Figure I-27:Détection des contours à l'aide des différents filtres.**[7]**

2.2.3. Segmentation fondée sur la classification des pixels :

La classification consiste à regrouper les pixels d'une image en sous-ensembles homogènes selon un critère défini. Ce processus, également appelé partitionnement (clustering), permet d'identifier des groupes de pixels partageant des caractéristiques similaires. Certaines approches peuvent s'appuyer sur des principes issus de la logique floue.

- Cette méthode repose sur un apprentissage préalable à partir d'un ensemble de données annotées (base d'apprentissage). Elle vise à découper l'espace de représentation en classes distinctes à l'aide d'une fonction de discrimination, fondée sur une connaissance a priori de la structure de l'image.[21]

3.2.3.1. Classification supervisée des pixels :

Cette méthode repose sur un apprentissage préalable à partir d'un ensemble de données annotées (base d'apprentissage). Elle vise à découper l'espace de représentation en classes distinctes à l'aide d'une fonction de discrimination, fondée sur une connaissance a priori de la structure de l'image. Parmi les algorithmes couramment utilisés **:**

- k-plus proches voisins (K-NN).

- Classificateur bayésien.
- Machines à vecteurs de support (SVM).
- Réseaux de neurone smulticouches.

3.2.3.2. Classification non supervisée des pixels :

La classification non supervisée divise l'espace de représentation en zones homogènes sans connaissance préalable de l'image. Elle se base uniquement sur des critères de similarité entre les pixels.

Aucune phase d'apprentissage n'est nécessaire : la méthode fonctionne de manière dite « aveugle ».

Parmi les algorithmes utilisés pour cette classification :

- k-moyennes (K-means)
- C-moyennes floues (fuzzy c-means)
- Algorithme de Fisher [7]

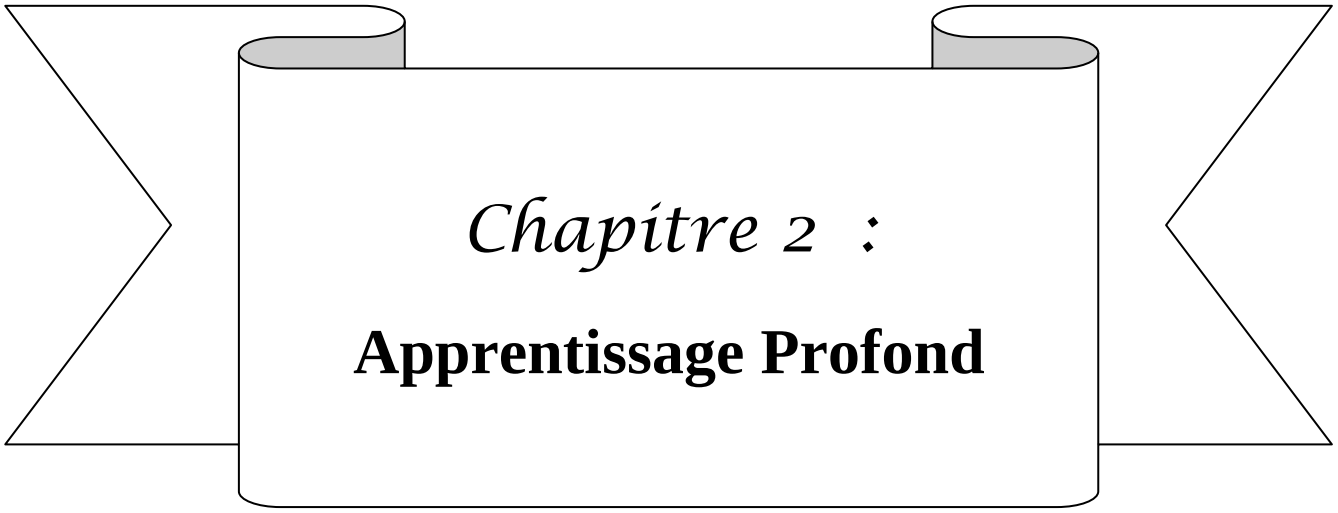
3. Conclusion :

La segmentation d'image constitue une composante fondamentale du traitement visuel et joue un rôle central dans de nombreuses applications de la vision artificielle. Qu'elles reposent sur l'analyse des contours, la cohérence des régions ou la classification des pixels, les méthodes de segmentation permettent d'extraire des structures significatives, facilitant ainsi l'interprétation automatique des images.

Une compréhension approfondie des principes fondamentaux de l'image numérique, ainsi que des différentes techniques de segmentation, est indispensable pour traiter efficacement les problématiques liées à l'analyse d'images. Ces connaissances offrent un socle solide pour appréhender les enjeux actuels du domaine, notamment en ce qui concerne le développement de systèmes capables de percevoir et d'interpréter leur environnement visuel de manière fine et contextualisée.

Dans cette perspective, le chapitre suivant sera consacré aux apports de l'apprentissage profond (Deep Learning) dans le domaine de la segmentation sémantique. Nous y mettrons en lumière les avancées méthodologiques qui ont permis d'atteindre des niveaux

de précision élevés, notamment dans le cadre complexe de la segmentation des scènes urbaines.



Chapitre 2 :

Apprentissage Profond

II. Chapitre 2 : Apprentissage profond

1. Introduction

Le Deep Learning, ou apprentissage profond, est aujourd'hui au cœur de nombreuses avancées dans le domaine de l'intelligence artificielle. Il s'appuie sur des réseaux de neurones artificiels capables d'apprendre automatiquement à partir de grandes quantités de données, ce qui le rend particulièrement performant dans les tâches complexes de traitement d'images.

Ce chapitre a pour objectif de présenter les concepts fondamentaux du Deep Learning qui serviront de base pour la suite du travail. Nous commencerons par introduire le fonctionnement général des réseaux de neurones artificiels, depuis l'inspiration biologique jusqu'aux modèles mathématiques utilisés dans les systèmes actuels. Nous aborderons ensuite des architectures plus avancées, telles que les réseaux convolutifs (CNN), les réseaux récurrents, ou encore les autoencodeurs, chacun adapté à des besoins spécifiques. Une attention particulière sera accordée aux architectures CNN utilisées en segmentation d'images, telles qu'U-Net, SegNet ou VGG, qui sont devenues des références dans le domaine. Ce chapitre traitera également des notions essentielles comme les fonctions d'activation, les fonctions de coût, les optimiseurs, ainsi que les techniques de régularisation et d'apprentissage par transfert. Nous consacrerons aussi une partie à la présentation des bases de données utilisées pour la segmentation d'images urbaines, ainsi qu'aux caractéristiques spécifiques de ces images (densité visuelle, variations d'échelle, occlusions, etc.), qui posent des défis particuliers pour les modèles d'apprentissage. Enfin, ce chapitre inclura une section consacrée à l'état de l'art, qui permettra de situer notre travail dans le contexte des recherches existantes et de comparer différentes approches en matière de segmentation par Deep Learning.

2. Définition du Deep Learning :

Le deep Learning, ou apprentissage profond, est une branche du Machine Learning qui repose sur l'utilisation de réseaux neuronaux profonds, inspirés du fonctionnement du cerveau humain, pour apprendre à partir de grandes quantités de données. Ces modèles permettent de modéliser des problèmes complexes, d'effectuer des prédictions et de prendre des décisions de manière autonome.[22]

3. Applications du Deep Learning :

Le deep Learning est appliqué dans de nombreux domaines. Les principaux sont :

- **Vision par ordinateur :** utilisée dans la reconnaissance faciale, la détection et la classification d'objets, ainsi que dans la perception visuelle des véhicules autonomes.
- **Santé et médecine : diagnostic** assisté, prédiction et détection de maladies, recherche pharmaceutique, analyse d'images médicales et segmentation.
- **Traitement du langage naturel (NLP) : utilisé** dans les traducteurs automatiques, les assistants vocaux et les chatbots, ou encore l'analyse de sentiments et la classification de textes.
- **Robotique, transport et automatisation intelligente.**[23]

4. Historique et évolution du Deep Learning :

L'histoire du Deep Learning trouve ses racines dans les recherches sur les réseaux de neurones artificiels, un concept introduit dès 1943 par Warren McCulloch et Walter Pitts. Ils ont développé un premier modèle mathématique inspiré du neurone biologique, posant ainsi les bases des réseaux de neurones artificiels. Leur théorie, selon laquelle l'activation des neurones constitue l'unité de base de l'activité cérébrale, a ouvert la voie à l'idée que ces réseaux pouvaient imiter le fonctionnement du cerveau humain.[24]

En 1957, Frank Rosenblatt conçoit le Perceptron, un algorithme capable de classer des données en fonction de leurs caractéristiques. Ce modèle marque une avancée majeure dans l'apprentissage automatique, mais il se heurte rapidement à des limitations, notamment son incapacité à traiter des problèmes complexes comme la reconnaissance d'images avec plusieurs classes.[25]

Dans les années 1980, des progrès significatifs permettent de surmonter certaines limitations des réseaux neuronaux grâce à l'introduction du Perceptron multicouche et de la rétropropagation du gradient, popularisée par David Rumelhart, Geoffrey Hinton et Yann LeCun. Cette avancée améliore l'apprentissage des réseaux profonds, bien que les contraintes computationnelles restent un frein à leur adoption massive.[26]

L'essor du Big Data et des avancées en calcul parallèle dans les années 2000 relance l'intérêt pour le Deep Learning. En 2012, la victoire du modèle AlexNet lors du concours ImageNet démontre la supériorité des réseaux neuronaux convolutifs (CNN) pour la reconnaissance d'images, marquant un tournant dans l'histoire du Deep Learning.[24]

Depuis, les architectures de réseaux neuronaux se multiplient avec des modèles comme LSTM (Long Short-Term Memory) pour le traitement du langage naturel, les Transformers à

partir de 2017, et plus récemment les grands modèles de langage (LLM). Le Deep Learning continue d'évoluer avec des applications toujours plus avancées en intelligence artificielle.[27]

4. Réseaux de neurones :

4.1. Qu'est-ce qu'un réseau de neurones ?

Un réseau de neurones artificiels (Artificiel Neural Networks ANN) est un modèle inspiré du cerveau humain, conçu pour reconnaître des motifs et traiter des informations de manière autonome.Il repose sur un ensemble de neurones interconnectés qui coopèrent en ajustant leurs connexions pondérées afin d'optimiser l'analyse des données et produire un résultat. Grâce à un processus d'apprentissage basé sur l'exemple, il adapte progressivement ses paramètres pour améliorer ses performances sur une tâche spécifique, comme la classification, la prédiction ou la reconnaissance d'images.[28][29]

4.2. Neurone biologique :

Un neurone biologique est une cellule spécialisée du système nerveux qui joue un rôle clé dans la transmission des signaux électriques dans le corps. Il estcomposé de trois principales structures :

- **Le corps cellulaire (ou soma) :** Il est le centre de contrôle du neurone, où les informations reçues par les dendrites sont traitées.Il contienté galement le noyau de la cellule.
- **Les dendrites :** Ce sont des extensions qui reçoivent les signaux d'autres neurones et les transmettent au corps cellulaire. Elles sont cruciales pour la réception des informations provenant de l'environnement ou d'autres neurones.
- **L'axone :** C'est une longue fibre qui transmet les signaux électriques du corps cellulaire vers d'autres neurones ou vers les muscles et glandes.

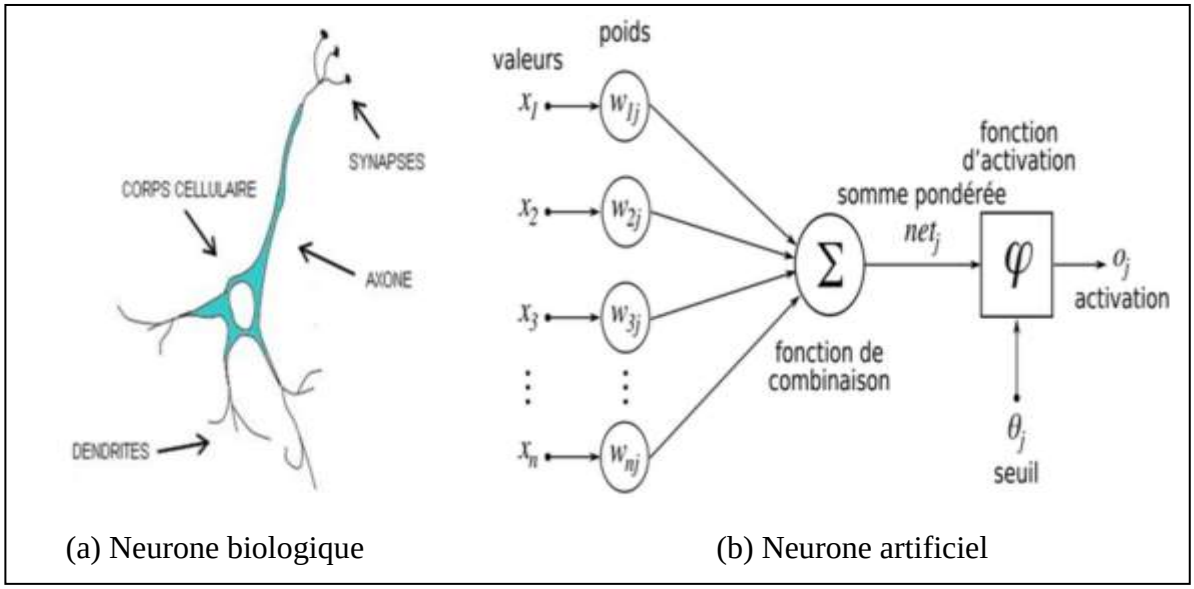
La transmission des signaux d'un neurone à l'autre se fait par des connexions appelées synapses, où les neurotransmetteurs chimiques permettent le passage de l'influx nerveux d'un neurone à un autre.[30]

Ces interactions sont essentielles pour des fonctions complexes comme la pensée, la mémorisation, et la coordination des mouvements. Le neurone fonctionne de manière collective avec d'autres neurones pour exécuter ces processus cognitifs.

4.3. Du neurone biologique au neurone artificiel :

Un neurone artificiel est un modèle inspiré des neurones biologiques, utilisé dans les réseaux de neurones artificiels (RNA). Il reçoit plusieurs entrées, les combine en fonction de poids associés à chaque connexion, puis passe le résultat à travers une fonction d'activation pour générer une sortie. Ce processus permet aux neurones artificiels de traiter et de transmettre des informations. Ils sont utilisés dans des applications variées, telles que la reconnaissance d'images, la traduction automatique et la prédiction.[31]

Le neurone artificiel imite le fonctionnement d'un neurone biologique en recevant, traitant et transmettant des informations. Bien que leur structure diffère, ils partagent le même principe fondamental de traitement des données.[28]



(a) Neurone biologique (b) Neurone artificiel

Figure II-1: Comparaison entre un neurone biologique et un neurone artificiel.**[32]**

4.4. Perceptron :

Le perceptron est un modèle d'algorithme d'apprentissage supervisé développé par Frank Rosenblatt en 1957, conçu pour la classification binaire. Il imite le comportement d'un neurone biologique en effectuant des calculs simples à partir des données d'entrée.

Un perceptron est un neurone artificiel qui reçoit plusieurs entrées, les pondère avec des poids, puis applique une fonction d'activation pour produire une sortie.[31]

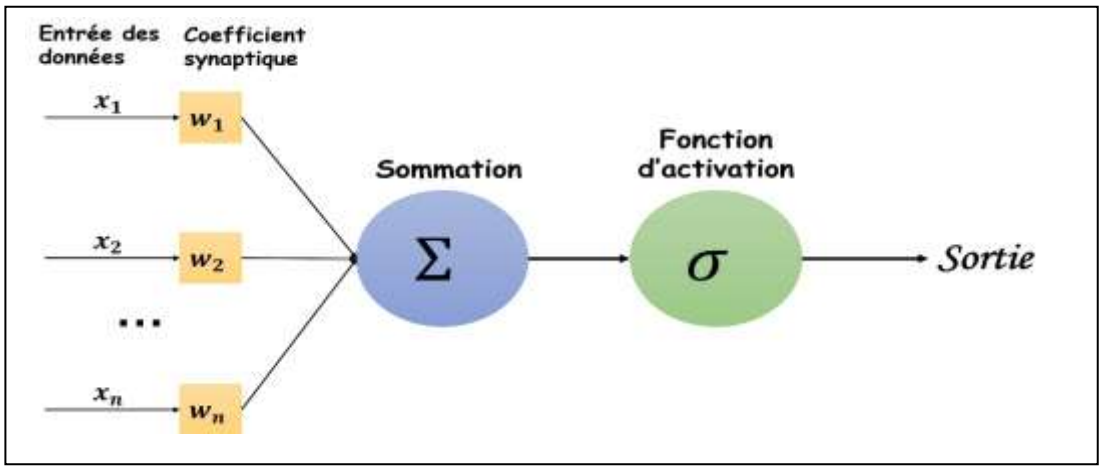


Figure II-2: Structure et fonctionnement d'un perceptron simple.**[33]**

La structure et le fonctionnement d'un perceptron reposent sur plusieurs étapes essentielles:

- **Entrées et poids :** Le perceptron reçoit plusieurs entrées, chaque entrée étant associée à un poids qui détermine son influence sur la sortie.
- **Somme pondérée :** Les entrées sont multipliées par leurs poids respectifs, puis additionnées, et un **biais** est ajouté à cette somme.Cela permet de prendre en compte une déviation qui affine la décision.[34]
- **Fonction d'activation** : Le résultat de cette somme pondérée est passé à travers une fonction d'activation, souvent une fonction seuil, qui renvoie 1 si la somme dépasse un certain seuil et 0 sinon.
- **Sortie :** La sortie finale du perceptron est alors déterminée par cette fonction d'activation.

Le perceptron ajuste ses poids en fonction de l'erreur entre la sortie prédite et la sortie réelle. Si la prédiction est incorrecte, les poids sont ajustés pour minimiser l'erreur. Cela se fait via une technique appelée mise à jour des poids lors de la phase d'apprentissage supervisé.

Le perceptron est limité lorsqu'il s'agit de résoudre des problèmes non linéaires. Il ne peut pas séparer des données qui ne sont pas linéairement séparables, comme c'est le cas pour le problème **XOR**. Cela a conduit au développement de réseaux de neurones multicouches pour traiter des problèmes plus complexes.[35]

Malgré sa simplicité, le perceptron a posé les bases de l'apprentissage automatique moderne. Il a inspiré la création de modèles plus avancés, tels que les réseaux de neurones

à couches profondes (**deeplearning**). Il reste un outil pédagogique fondamental pour comprendre comment les algorithmes d'apprentissage supervisé fonctionnent et est souvent utilisé pour la classification binaire dans des situations simples.

4.5. Perceptron multicouche (MLP) :

Le perceptron multicouche (MLP) est une extension du perceptron simple, conçue pour résoudre des problèmes non linéaires en intégrant une ou plusieurs couches cachées entre la couche d'entrée et la couche de sortie. Ces couches intermédiaires permettent d'extraire des caractéristiques plus complexes et d'améliorer la capacité du réseau à généraliser.[36]

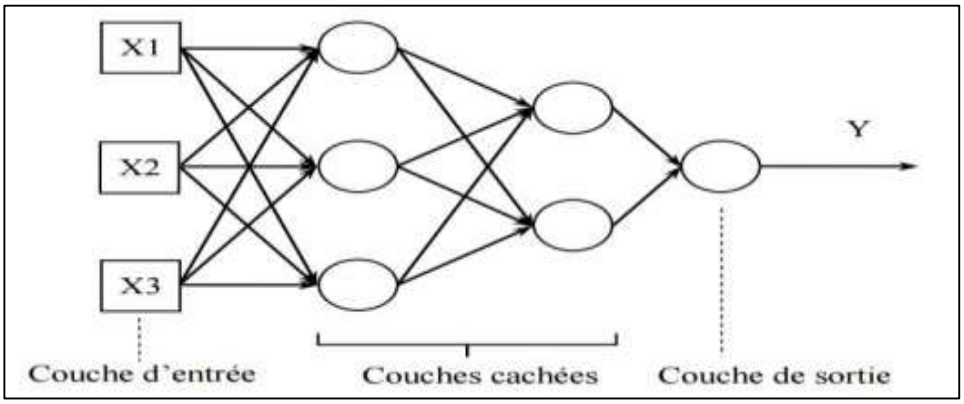


Figure II-3: Architecture d'un perceptron multicouche (MLP). **[37]**

Un MLP est constitué de neurones organisés en couches :

- **La couche d'entrée :** reçoit les données brutes du problème à traiter, chaque neurone correspondant à une caractéristique spécifique de ces données.
- **Une ou plusieurs couches cachées :** traitent et transforment les informations reçues de la couche précédente en appliquant des pondérations et des fonctions d'activation, permettant ainsi au réseau de modéliser des relations complexes et non linéaires dans les données.
- **La couche de sortie :** fournit le résultat final du réseau, que ce soit une prédiction de classe, une valeur numérique ou une autre sortie, en fonction de la tâche spécifique à accomplir.

Le choix de l'architecture du réseau, notamment le nombre de couches et de neurones par couche, est un paramètre crucial qui influence la capacité d'apprentissage et de généralisation du modèle. Un autre hyper-paramètre clé est la fonction

d'activation, qui introduit la non-linéarité nécessaire pour capturer des relations complexes dans les données.[28]

Le perceptron multicouche constitue la base des réseaux de neurones profonds (Deep Learning), qui intègrent plusieurs couches cachées et sont capables de résoudre des problèmes encore plus sophistiqués.

4.6. Les types de réseaux de neurones artificiels :

Les réseaux de neurones profonds (Deep Neural Networks, DNN) se déclinent en plusieurs types, chacun adapté à des tâches spécifiques. Voici les principaux types :

5.6.1. Réseaux de Neurones Récurrents :

Un réseau de neurones récurrents **(Recurrent Neural Networks, RNN)**est un type de réseau neuronal conçu pour traiter des données séquentielles, telles que du texte, de l'audio ou des séries temporelles. Contrairement aux réseaux neuronaux classiques, les RNN possèdent des connexions récurrentes qui leur permettent de conserver une mémoire des informations précédentes lors du traitement des nouvelles entrées. Cette caractéristique les rend particulièrement adaptés aux tâches où le contexte et l'ordre des données sont essentiels, comme la traduction automatique, la reconnaissance vocale ou la prédiction de séries temporelles.[38]

Les architectures **Long Short-Term Memory (LSTM)** et **GatedRecurrent Unit (GRU)** sont des variantes des réseaux de neurones récurrents (RNN) traditionnels, conçues pour améliorer la mémorisation et mieux capturer les dépendances à long terme dans les données séquentielles.

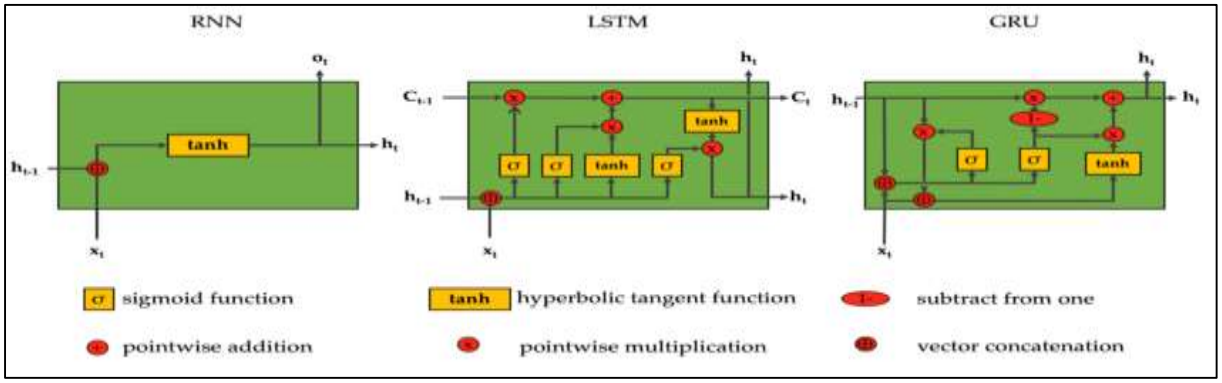


Figure II-4: Architecture des réseaux neuronaux récurrents et variantes LSTM/GRU. **[38]**

5.6.2. Autoencodeurs (AE) :

Les auto-encodeurs sont des réseaux de neurones artificiels utilisés en apprentissage non supervisé pour apprendre des représentations compactes et efficaces des données. Ils se composent de deuxparties:l'encodeur, qui compresse les données d'entrée en une représentation de dimension inférieure appelée **espace latent**

- l'encodeur, qui compresse les données d'entrée en une représentation de dimension inférieure appelée **espace latent**,
- le décodeur, qui reconstruit les données d'origine à partir de cette représentation.

L'objectif est d'apprendre une représentation dense des données, facilitant des tâches telles que la réduction de la dimensionnalité, le débruitage et la détection d'anomalies.[39][40]

5.6.3. Réseaux de neurones convolutifs (CNN) :

Les réseaux de neurones convolutifs (Convolutional Neural Networks, CNN) sont des architectures de réseaux neuronaux profonds conçues pour traiter des données structurées en grille, telles que les images. Ils sont particulièrement efficaces pour les tâches de reconnaissance et de classification d'images.[41]

4.7. Introduction aux réseaux de neurones convolutifs (CNN) :

Les CNN sont particulièrement efficaces pour le traitement et l'analyse des données visuelles. Inspirés par l'organisation du cortex visuel des animaux, les CNN sont composés de plusieurs types de couches qui travaillent conjointement pour extraire des caractéristiques pertinentes des images et effectuer des tâches telles que la classification ou la détection d'objets.[42]

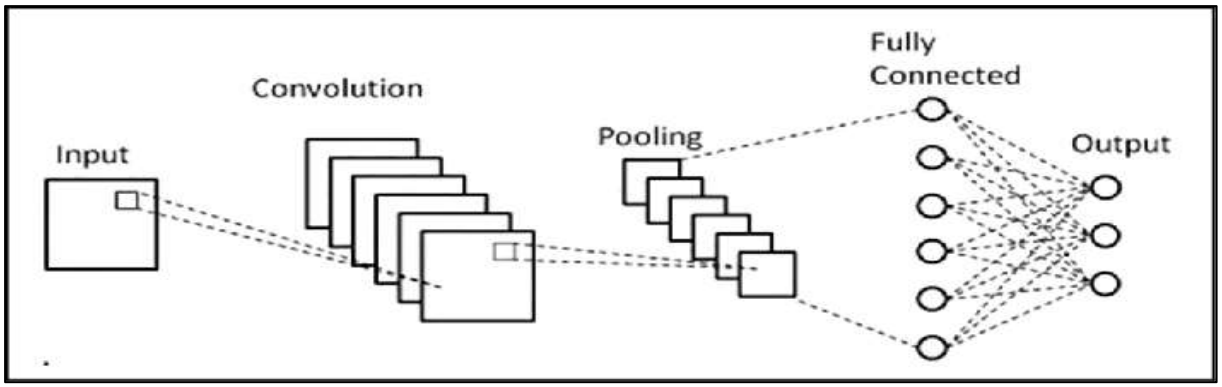


Figure II-5: Structure générale d'un réseau de neurones convolutifs (CNN).**[43]**

Les CNN sont également utilisés dans le traitement de données complexes comme la parole ou l'audio. Leur efficacité repose sur une structure composée de plusieurs types de couches, chacune dotée de paramètres et d'hyperparamètres spécifiques. Voici quelques notions fondamentales concernant les CNN :

a) Couche de convolution:

La couche de convolution est un élément fondamental des réseaux de neurones convolutifs (CNN). Elle a pour rôle d'extraire les caractéristiques pertinentes des images en entrée en appliquant une opération mathématique appelée convolution. Cette opération consiste à faire glisser un petit filtre (ou noyau) sur l'image d'entrée et à calculer le produit scalaire entre le filtre et la portion de l'image qu'il recouvre. Le résultat est une valeur unique qui représente la présence d'une caractéristique spécifique à cet endroit de l'image.

Les couches convolutives sont composées de vecteurs d'entrée, tels que l'image elle-même, de filtres, tels que des détecteurs de caractéristiques, et de vecteurs de sortie, appelés cartes de caractéristiques ou cartes d'activation. Après avoir traversé une couche convolutive, l'image est transformée en une carte de caractéristiques qui met en évidence les éléments pertinents détectés.[44]

Une caractéristique notable de ces couches est le partage des paramètres de filtrage entre les différents neurones d'un même noyau de convolution. Cette approche permet au réseau de neurones convolutifs d'être invariant au déplacement, c'est-à-dire que si un filtre est efficace pour détecter une caractéristique dans une partie de l'image, il le sera probablement également dans d'autres parties.

Après la convolution, il est courant d'appliquer une fonction d'activation pour introduire une non-linéarité dans le réseau. Cette étape est essentielle pour permettre au réseau d'apprendre des relations complexes entre les variables. La fonction ReLU (Rectified Linear Unit) est fréquemment utilisée à cet effet, car elle favorise un apprentissage plus rapide et plus efficace.[45]

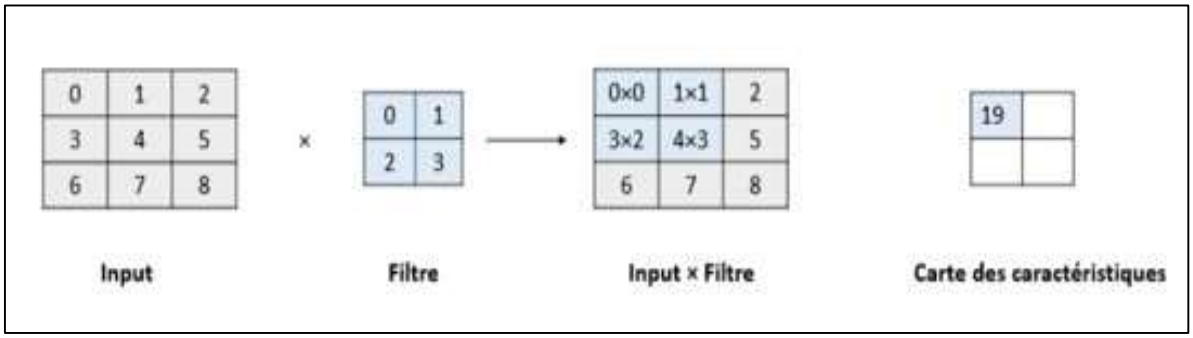


Figure II-6: Fonctionnement d'une couche de convolution. **[36]**

b) Choix des hyper paramètres :

Un hyper paramètre est une variable utilisée pour configurer un modèle de machine learning. Contrairement aux paramètres du modèle, qui sont appris durant l'entraînement, les hypers paramètres doivent être définis par l'utilisateur avant de commencer l'entraînement.[46]

Quatre hyper paramètres principaux caractérisent la couche de convolution :

- **K (Profondeur) :** nombre de filtres (kernels) appliqués.
- **F (Taille du filtre) :** dimensions du filtre (kernel), généralement F×F×D pixels.
- **S (Pas ou stride) :** déplacement du filtre sur l'image lors de la convolution. Un pas plus grand réduit la taille de la sortie, tandis qu'un pas plus petite conserve davantage de détails spatiaux.
- **P (Marge ou padding) :** ajout de pixels autour des bords de l'image d'entrée avant convolution, afin de contrôler la taille des sorties et préserver les informations aux frontières. Sans padding, les dimensions des cartes de caractéristiques diminuent après chaque couche, ce qui peut entraîner une perte rapide d'informations importantes.[47]

Pour une image d'entrée de dimensions (W × H × D), où :

- W : largeur en pixels (width)
- H : hauteur en pixels (height)
- D : profondeur (nombre de canaux, depth)

La carte de caractéristiques résultante aura comme dimensions (Wc x Hc x Dc), où:

Wc= [(W – F + 2P) / S] + 1 **Équation II-01**

Hc = [(H – F + 2P) / S] + 1 **Équation II-02**

Dc = K

Ces formules permettent de calculer les dimensions de la sortie en fonction des hyperparamètres choisis pour la couche de convolution.[34]

c) Couche de Pooling :

La couche de pooling est une composante essentielle des réseaux de neurones convolutifs (CNN), utilisée pour réduire la dimension spatiale des cartes de caractéristiques générées par les couches de convolution. Cette réduction diminue le nombre de paramètres et de calculs dans le réseau, améliorant ainsi son efficacité et aidant à prévenir le surapprentissage.[48]

L'opération de pooling consiste à parcourir la carte de caractéristiques avec une fenêtre (ou filtre) de taille définie et à appliquer une fonction d'agrégation sur les valeurs de chaque région couverte par cette fenêtre.

Les types de pooling les plus courants sont le max pooling, qui sélectionne la valeur maximale dans chaque région couverte par la fenêtre, conservant ainsi les caractéristiques lesplus dominantes de l'image, et l'averagepooling, qui calcule la moyenne des valeurs dans chaque région couverte par la fenêtre, offrant une représentation plus lissée des caractéristiques.[45]

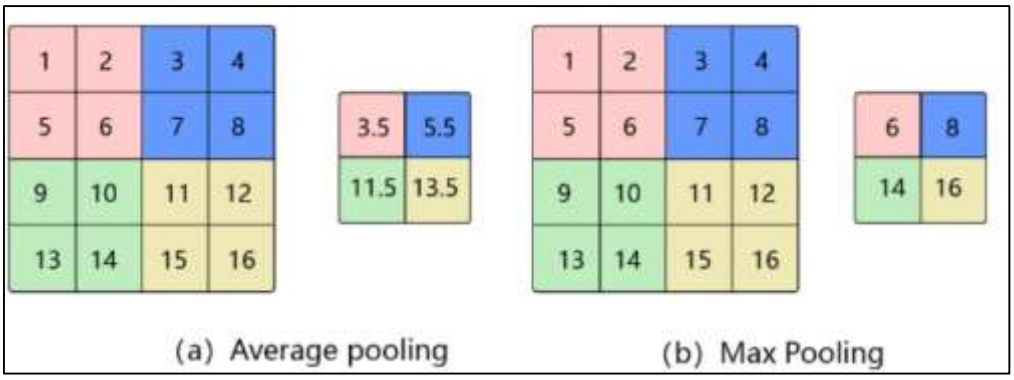


Figure II-7: Illustration des opérations de pooling.**[49]**

d) Couche entièrement connectée :

La couche entièrement connectée, aussi appelée **FullyConnected (FC)**, est généralement placée à la fin d'un réseau de neurones convolutifs (CNN). Chaque neurone y est relié à tous ceux de la couche précédente, permettant ainsi de combiner les caractéristiques extraites par les couches de convolution et de pooling. Elle transforme ces caractéristiques en un vecteur de sortie, dont chaque composante représente la probabilité d'appartenance de l'image d'entrée à une classe spécifique. Cette couche réalise donc la classification finale, en utilisant souvent la fonction d'activation **Softmax**, adaptée pour générer des probabilités. Contrairement aux couches convolutives qui capturent des motifs locaux, la couche FC intègre une vision globale des données et joue un rôle crucial dans la prise de décision du modèle.[50]

e) Couche dropout:

La couche Dropout est une technique de régularisation utilisée pour réduire le sur-apprentissage lors de l'entraînement des réseaux de neurones. Elle consiste à désactiver aléatoirement certains neurones, ainsi que leurs connexions, pendant chaque itération d'apprentissage, avec une probabilité prédéfinie. Cette approche empêche le réseau de devenir trop dépendant de certains neurones, favorisant ainsi une meilleure généralisation sur de nouvelles données.[51]

f) Fonction d'activation :

Les fonctions d'activation sont des éléments essentiels dans les réseaux de neurones artificiels, introduisant des non-linéarités qui permettent au réseau d'apprendre des relations complexes. Sans elles, un réseau de neurones, quelle que soit sa profondeur, se comporterait comme un simple modèle linéaire, limitant ainsi sa capacité à résoudre des problèmes complexes.

Voici quelques-unes des fonctions d'activation les plus couramment utilisées :

- **Sigmoïde (Sigmoid) :** La fonction sigmoïde, également appelée fonction logistique, est une fonction d'activation couramment utilisée dans les réseaux de neurones. Elle transforme une valeur d'entrée x en une sortie comprise entre 0 et 1.[39]
- **Rectified Linear Unit (ReLU)** : La fonction d'activation ReLU est largement utilisée dans les réseaux de neurones, notamment en vision par ordinateur. Elle renvoie zéro pour les entrées négatives et la valeur d'entrée elle-même pour les entrées positives.[48]
- **Tangente hyperbolique (Tanh)** : La fonction Tanh transforme les entrées en sorties comprises entre -1 et 1, centrées autour de zéro. Cette caractéristique facilite la modélisation de relations complexes, car elle permet au réseau de traiter efficacement des valeurs négatives, positives ou neutres. [39]
- **Softmax** : La fonction Softmax est généralement appliquée à la couche de sortie pour les tâches de classification multi-classes. Elle transforme un vecteur de scores bruts (logits) en une distribution de probabilité, où chaque sortie est comprise entre 0 et 1, et la somme de toutes les sorties est égale à 1. Chaque composant de cette sortie représente la probabilité que l'entrée appartienne à une classe spécifique.[52]

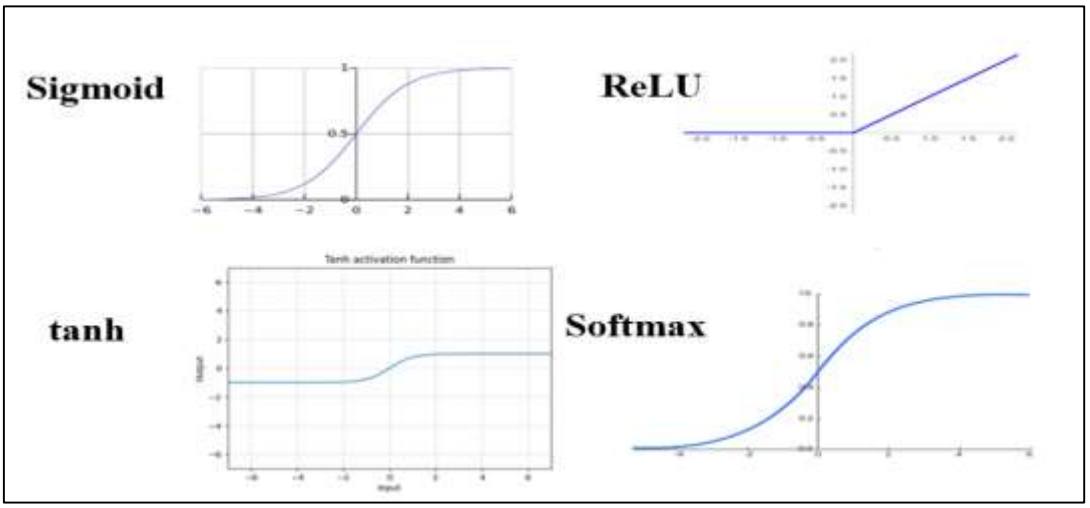


Figure II-8: Exemples de fonctions d'activation utilisées dans les réseaux de neurones.

g) Fonction de coût :

Également appelée fonction d'erreur, fonction de perte ou fonction objectif, constitue une mesure quantitative de l'écart entre les prédictions effectuées par un modèle d'apprentissage automatique et les valeurs réelles attendues. Elle joue un rôle fondamental dans le processus d'apprentissage, en orientant l'ajustement des paramètres du modèle afin de minimiser cette différence. Cette minimisation permet d'améliorer progressivement la qualité des prédictions.[53]

Le choix de la fonction de coût dépend étroitement de la nature du problème étudié, notamment s'il s'agit d'une tâche de régression ou de classification.

Dans les problèmes de **régression**, où l'objectif est de prédire des valeurs continues, deux fonctions de coût sont couramment utilisées :

- **Erreur Quadratique Moyenne (Mean Squared Error - MSE)** : elle calcule la moyenne des carrés des écarts entre les valeurs prédites et les valeurs réelles. Cette approche pénalise significativement les grandes erreurs en les élevant au carré. Sa formule mathématique est la suivante : [53]

$$\mathbf{MSE} = \frac{1}{n}\sum_{i} \|\hat{y}\mathrm{i} - \mathrm{yi}\|^2 \qquad \textbf{Équation II-03}$$

n : Nombre d'exemples dans l'ensemble de données.

ŷi: La valeur prédite par le modèle.

yi: La vraie valeur (étiquette).

- **Erreur Absolue Moyenne (MeanAbsoluteError - MAE)** : est une mesure qui évalue la moyenne des différences absolues entre les valeurs prédites par un modèle et les valeurs réelles observées. Sa formule mathématique est la suivante :[53]

$$\mathbf{MAE} = \frac{1}{n}\sum_{i} |\hat{y}\mathrm{i} - \mathrm{yi}| \qquad \textbf{Équation II-04}$$

Pour les tâches de classification, où l'objectif est d'assigner des catégories aux observations, la fonction de coût fréquemment utilisée est :

- **La perte logarithmique (Log Loss) :** évalue la performance des classificateurs probabilistes en tenant compte de l'incertitude des prédictions. Elle pénalise davantage les modèles qui attribuent une forte probabilité à une classe incorrecte, reflétant ainsi une mauvaise performance. [54]

h) Epoch : une époque correspond à un passage complet de l'ensemble des données d'entraînement à travers le réseau de neurones.[55]

i) Batche (lot) : désigne un sous-ensemble des données d'entraînement qui est traité ensemble lors d'un passage avant (forwardpass) et d'un passage arrière (backwardpass) du réseau de neurones.[55]

j) Batch size : La taille du lot est un hyper paramètre qui définit le nombre d'exemples d'entraînement inclus dans un lot. Elle détermine la quantité de données traitées avant que le modèle ne mette à jour ses poids.[55]

k) Les optimiseurs : Les optimiseurs ajustent les paramètres d'un modèle, comme les poids d'un réseau de neurones, pour minimiser la fonction de perte. Cela se fait via la rétropropagation, un processus clé de l'apprentissage profond, visant à trouver un minimum de la fonction de perte en ajustant les paramètres du modèle.Parmi les optimiseurs les plus utilisés, on peut citer :[56]

- **Stochastic Gradient Descent (SGD) :** met à jour les poids après chaque échantillon d'entraînement, ce qui peut rendre la courbe de perte bruyante. Pour réduire ce bruit, elle utilise un échantillon aléatoire à chaque itération pour calculer le gradient.[56]
- **Adam (Adaptive Moment Estimation) :** est un optimiseur largement utilisé en apprentissage profond. Il combine les avantages de Momentum et RMSProp, ce qui lui permet de gérer efficacement des gradients bruités et des données complexes. Adam ajuste automatiquement le taux d'apprentissage pour chaque paramètre, ce qui le rend particulièrement efficace et stable. En pratique, c'est souvent le choix par défaut pour l'entraînement des réseaux de neurones.[57]

4.8. Quelques architectures CNN pour la segmentation :

Au fil du temps, plusieurs architectures CNN ont été développées, chacune avec des structures propres, permettant d'améliorer les performances en vision par

ordinateur. Leurs succès sont souvent évalués à travers des compétitions telles qu'ImageNet. Grâce à leur efficacité dans l'extraction des caractéristiques visuelles, les CNN sont devenus des piliers de l'intelligence artificielle. Certaines architectures ont marqué le domaine et servent désormais de références dans la recherche en deeplearning.

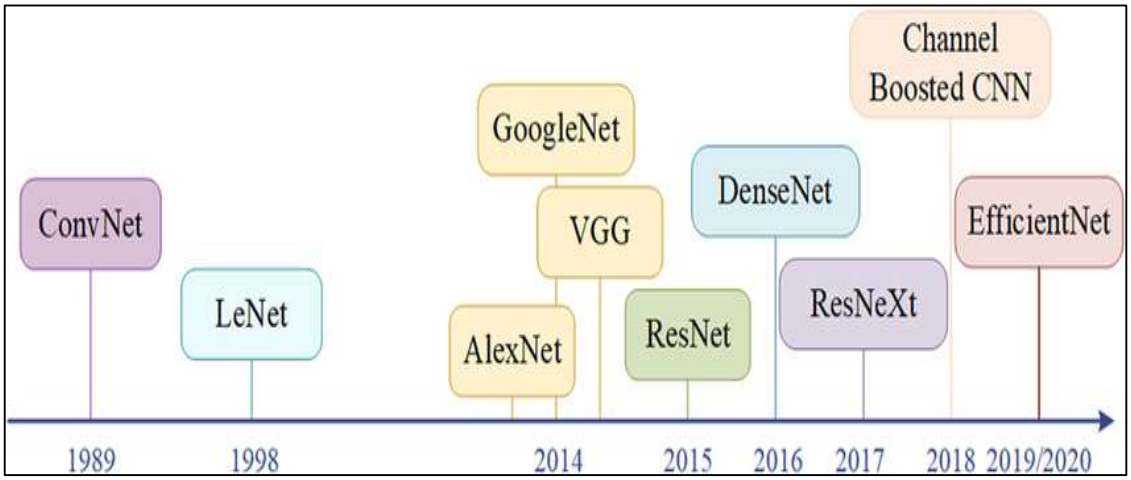


Figure II-9: Exemples d'architectures CNN utilisées pour la segmentation.**[58]**

5.8.1. U-Net :

U-Net est une architecture CNN conçue pour la segmentation d'images, notamment biomédicales. Proposé par Olaf Ronneberger et al. à l'université de Fribourg, ce modèle s'est rapidement imposé comme une référence majeure.[28]

Sa structure en forme de U repose sur deux chemins principaux :

- **Le chemin de contraction (encodeur) :** Il extrait les caractéristiques importantes de l'image via une succession de convolutions, d'activations ReLU et de max pooling, tout en réduisant progressivement la dimension spatiale.
- **Le chemin d'expansion (décodeur) :** Il reconstruit la carte de segmentation en combinant les caractéristiques extraites avec des informations de localisation précises grâce à des convolutions transposées et de connexions de saut (skip connections).[34]

Cette conception permet à U-Net de localiser précisément les objets dans une image, en prédisant la classe de chaque pixel. L'une de ses forces majeures est sa capacité à

fonctionner efficacement même avec un nombre limité d'images d'entraînement, ce qui en fait un choix idéal dans les domaines où les données sont rares, comme en médecine.[28]

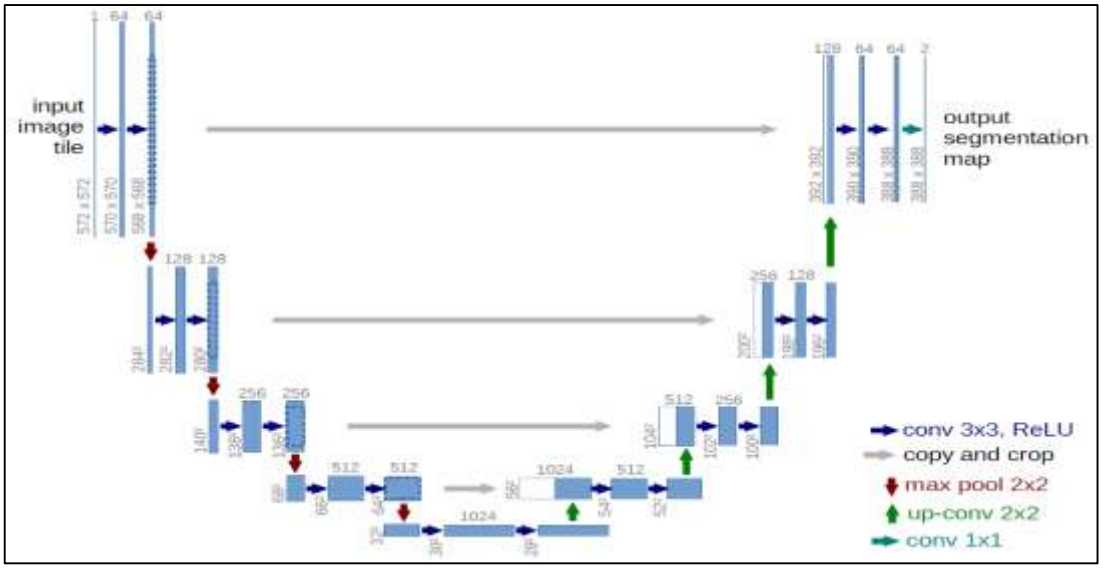


Figure II-10: Architecture U-Net.**[59]**

5.8.2. SegNet :

SegNet est un modèle CNN conçu pour la segmentation sémantique, c'est-à-dire la classification de chaque pixel d'une image selon une catégorie (route, bâtiment, véhicule, etc.). Il repose sur une architecture encodeur-décodeur symétrique basée sur les couches convolutionnelles du célèbre modèle VGG-16.

- **Encodeur :** succession de couches de convolution, normalisation par batch (BN), activation ReLU et max pooling.
- **Décodeur :** structure symétrique à l'encodeur, utilisant les indices de max pooling de l'encodeur pour effectuer un sur-échantillonnage précis sans devoir réapprendre les poids d'upsampling.
- **Sortie :** une carte de segmentation, où chaque pixel est étiqueté selon la classe à laquelle il appartient.[60]

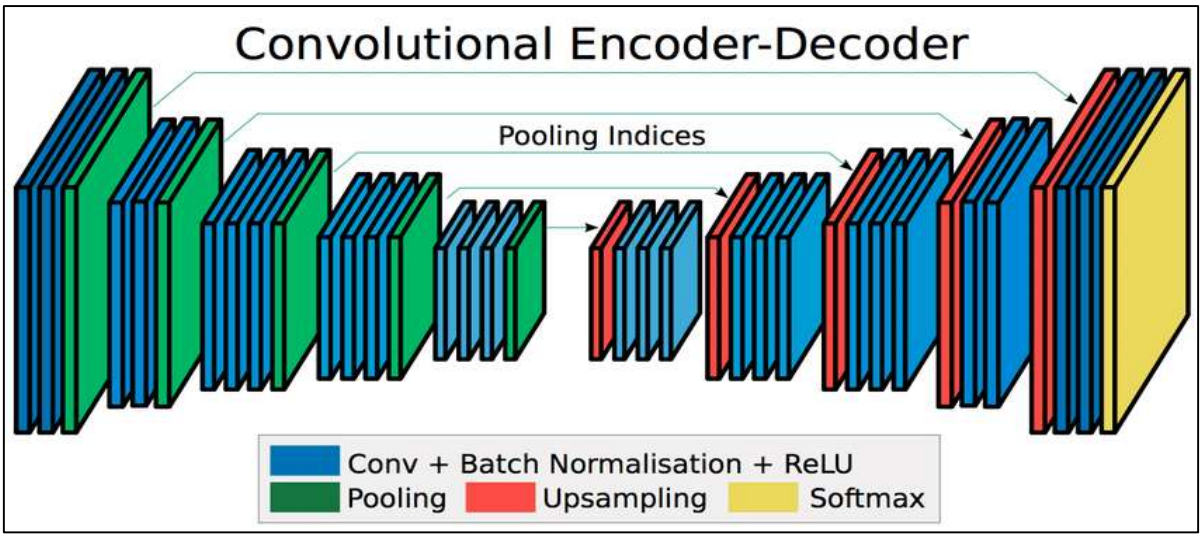


Figure II-11: Architecture SegNet.[61]

5.8.3. VGG :

VGG (Visual Geometry Group) est une architecture CNN développée par l'Université d'Oxford. Les variantes les plus connues sont VGG-16 et VGG-19, contenant respectivement 16 et 19 couches. Elle est reconnue pour sa simplicité et son efficacité, utilisant uniquement des filtres 3×3 et des couches de pooling pour extraire les caractéristiques.

Pour la segmentation d'images, VGG-16 est souvent utilisée comme backbone dans des modèles plus complexes (comme SegNet ou FCN), offrant un bon compromis entre performance et complexité.[62]

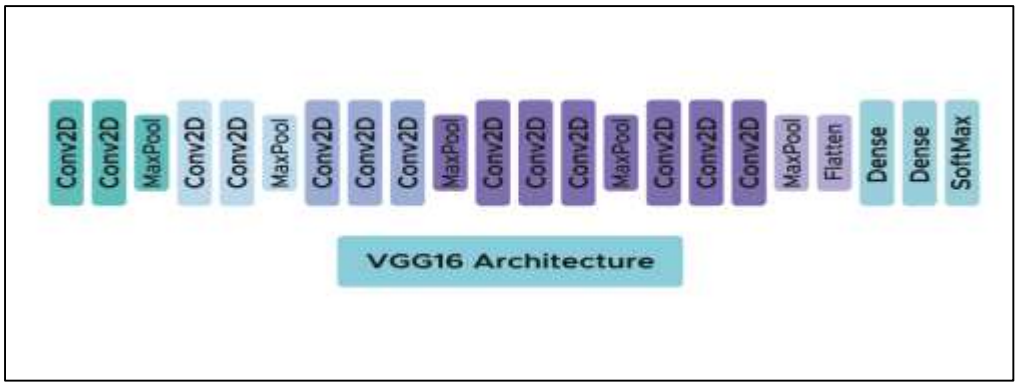


Figure II-12: Architecture VGG (Visual Geometry Group).**[63]**

5.8.4. DeepLabv3 :

DeepLabv3 est une architecture de segmentation sémantique conçue pour améliorer la détection d'objets à différentes échelles en contexte complexe. Elle repose sur l'utilisation de convolutions dilatées (atrous convolutions), appliquées en cascade ou en parallèle, permettant de capturer des informations à plusieurs résolutions sans perte de précision spatiale. L'un des composants clés de DeepLabv3 est le module Atrous Spatial Pyramid Pooling (ASPP), qui combine plusieurs niveaux de dilatation avec un pooling global pour renforcer la représentation du contexte global de l'image. Cette combinaison améliore considérablement les performances sur des scènes à forte variabilité spatiale.

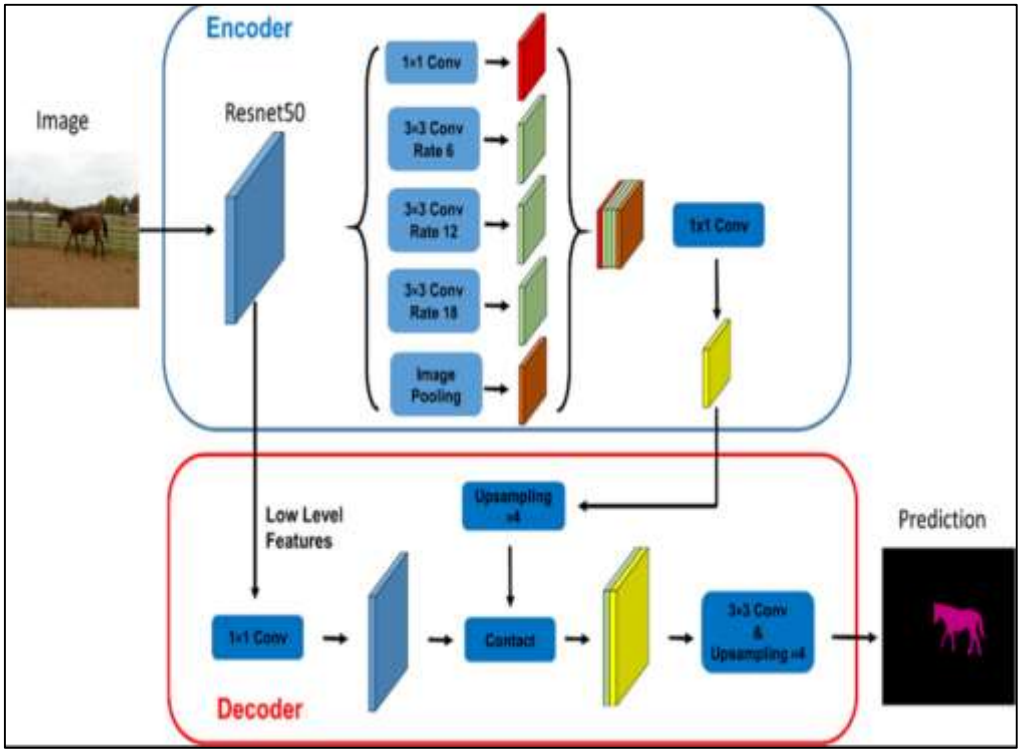


Figure II-13 : Schéma de l'architecture DeepLabv3.**[64]**

4.9. Apprentissage par transfert (Transfer Learning)

L'apprentissage par transfert est une approche du machine learning qui consiste à réutiliser les connaissances acquises par un modèle préalablement entraîné sur une tâche source, pour les adapter à une tâche cible similaire. Cette méthode est particulièrement utile lorsque la quantité de données disponibles pour la nouvelle tâche est limitée.

Dans le domaine du deeplearning, elle permet d'exploiter des architectures de réseaux neuronaux pré-entraînées sur de grands ensembles de données (comme ImageNet), en ne modifiant que certaines couches du modèle. Les premières couches, qui extraient des

caractéristiques génériques, sont souvent conservées, tandis que les dernières sont ajustées (fine-tuning) pour s'adapter à la nouvelle tâche.

Cette technique permet de réduire le temps d'entraînement, limiter les besoins en ressources, et améliorer les performances dans des domaines exigeants, comme la segmentation d'images, où les données annotées sont rares ou coûteuses à obtenir.[65]

4.10. Sur-ajustement et Sous-ajustement :

4.10.1. Surajustement (Overfitting) :

Le surajustement se produit lorsqu'un modèle apprend trop bien les détails et les particularités des données d'entraînement, y compris le bruit et les anomalies, au point de perdre sa capacité à généraliser à de nouvelles données. Il donne donc de très bons résultats sur l'ensemble d'entraînement, mais échoue à bien prédire sur les données jamais vues.

Comment l'éviter :

- **Arrêt anticipé (early stopping) :** interrompre l'entraînement lorsque les performances sur le jeu de validation cessent de s'améliorer.
- **Régularisation (L1, L2) :** pénaliser les poids trop importants pour éviter que le modèle devienne trop complexe.
- **Augmentation des données :** générer de nouvelles instances à partir des données existantes pour enrichir l'entraînement.
- **Réduction de la complexité du modèle** : simplifier l'architecture ou réduire le nombre de paramètres.
- **Validation croisée :** évaluer le modèle sur plusieurs sous-ensembles des données pour vérifier sa robustesse.
- **Méthodes d'ensemble :** combiner plusieurs modèles pour obtenir une prédiction plus stable (ex. bagging, boosting).[66]

4.10.2. Sous-ajustement (Underfitting)

Le sous-ajustement apparaît lorsqu'un modèle est trop simple ou mal entraîné, et ne parvient pas à capturer les relations essentielles dans les données. Il donne de mauvais résultats aussi bien sur les données d'entraînement que de test.

Comment l'éviter :

- **Augmenter la complexité du modèle :** utiliser des modèles plus profonds ou plus riches si le problème est complexe.
- **Allonger la durée d'entraînement :** laisser le modèle s'entraîner plus longtemps si nécessaire.
- **Améliorer les caractéristiques :** enrichir les entrées ou effectuer une ingénierie des variables plus adaptée.
- **Réduire le taux de régularisation :** trop de régularisation peut empêcher le modèle d'apprendre suffisamment.
- **Réviser le prétraitement :** une mauvaise normalisation ou un filtrage trop sévère peuvent masquer les patterns utiles.

L'objectif est de construire un modèle ni trop simple (biais élevé), ni trop complexe (variance élevée). Un bon modèle est celui qui généralise bien, c'est-à-dire qui a un bon équilibre entre performance sur les données d'apprentissage et sur les données de validation/test.[67]

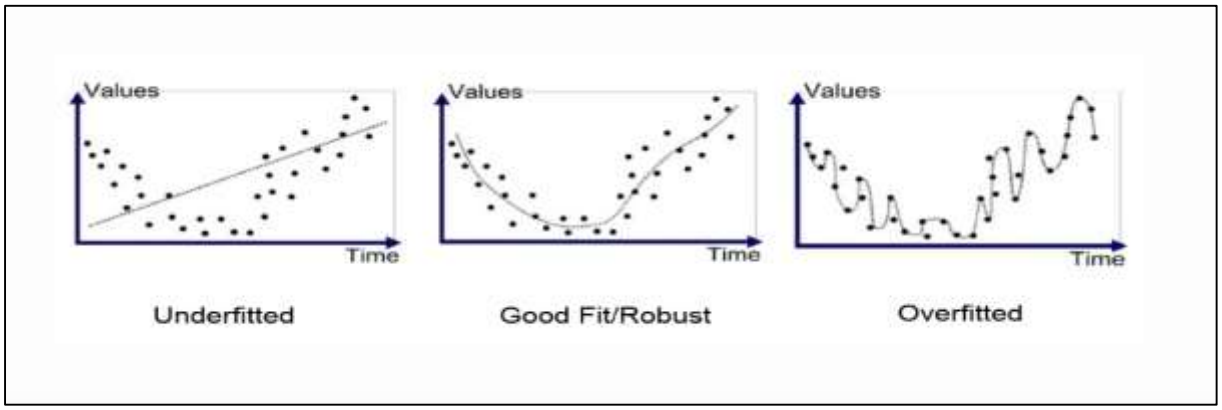


Figure II-14: Comparaison entre sur-ajustement (overfitting) et sous-ajustement (underfitting).**[68]**

5. Segmentation d'images urbaines par Deep Learning :

5.1. Définition :

Dans le cadre de la segmentation d'images urbaines, il est essentiel d'utiliser des bases de données représentatives des scènes réelles rencontrées en milieu urbain. Ces images présentent généralement une grande diversité d'objets, tels que les routes, bâtiments,

piétons, véhicules, arbres ou panneaux de signalisation. Cette diversité rend la tâche de segmentation particulièrement complexe, en raison notamment de la densité visuelle, des occlusions fréquentes, des variations d'éclairage et des différences d'échelle entre les objets.[69]

5.2. Historique et évolution des bases de données urbaines pour la segmentation sémantique

Depuis le début des années 2010, le besoin de données annotées pour la vision par ordinateur appliquée à la conduite autonome a conduit au développement de nombreuses bases de données d'images urbaines. En 2012, la base KITTI a été l'une des premières à fournir des données collectées en conditions réelles à partir de véhicules équipés de capteurs, ouvrant la voie à l'entraînement de modèles pour des tâches telles que la détection d'objets et la segmentation. En 2016, Cityscapes a marqué une avancée majeure en offrant des annotations pixel par pixel de scènes urbaines européennes, devenant une référence pour la segmentation sémantique. Dans la même période, face à la diversité limitée et au coût élevé des annotations manuelles, des bases synthétiques comme SYNTHIA (2016) et GTA5 ont été introduites pour simuler des environnements urbains variés et fournir des annotations précises à grande échelle. Par la suite, des bases comme MapillaryVistas (2017) et BDD100K (2020) ont élargi les perspectives en intégrant une plus grande diversité géographique, météorologique et temporelle. Aujourd'hui, ces bases de données réelles et synthétiques sont largement utilisées pour entraîner, évaluer et améliorer les modèles de segmentation sémantique, en particulier dans des contextes de généralisation, de transfert de domaine et d'apprentissage multitâche.[69][70][71]

6.3 État de l'art

La segmentation sémantique représente un domaine clé de la vision par ordinateur, Avec l'émergence des méthodes d'apprentissage profond, en particulier les réseaux de neurones convolutifs (CNN), ce domaine a connu des avancées majeures en termes de précision et de robustesse. Ainsi, un grand nombre de chercheurs se sont intéressés à cette thématique, proposant des architectures de plus en plus sophistiquées adaptées à diverses applications.

V. Nekrasov et al. [72] Ont proposé une méthode de segmentation sémantique en temps réel reposant sur la sparsité spatiale, en n'encodant que 25 % de l'image afin de réduire les

coûts de calcul. Évaluée sur le jeu de données Cityscapes, leur approche atteint un mIoU de 66 % tout en maintenant une vitesse élevée de 17,4 FPS, ce qui la rend particulièrement adaptée aux systèmes embarqués.

Dans les travaux de S. Khemiri et al. [73], l'architecture U-Net est employée pour la segmentation d'images de drones à haute résolution provenant de DroneDeploy. Grâce à sa structure en encodeur-décodeur dotée de connexions de saut, le modèle atteint une précision de 94 %, capturant efficacement les structures fines, ce qui le rend pertinent pour des applications telles que la télédétection

Baroudi et al. [6] Présentent d'autres travaux basés sur U-Net, démontrant une performance élevée dans des scènes urbaines. Une version modifiée de l'architecture a permis une segmentation efficace du trafic routier, atteignant une précision de 87,03 % sur le jeu de données Cityscapes, surpassant ainsi d'autres architectures CNN.

La série DeepLab, développée par L.-C. Chen et al[74],introduit des convolutions dilatées ainsi que des modèles CRF pour améliorer la précision des contours et capturer le contexte global de l'image. DeepLabv2 atteint un score de 79.7 % de mIoU sur le benchmark PASCAL VOC 2012 et environ 70.4 % sur Cityscapes, avec une précision pixel supérieure à 90 %, ce qui en fait une méthode particulièrement robuste pour la segmentation sémantique.

Mask2Former, proposé par B. Cheng et al.[75], est une architecture unifiée de type Transformer conçue pour la segmentation sémantique, d'instance et panoptique. Elle repose sur un mécanisme d'attention guidée par les masques, permettant une prédiction plus précise. Le modèle atteint un score de 81.3 % de mIoU sur Cityscapes et 57.8 % de PanopticQuality (PQ) sur COCO, démontrant ainsi une excellente capacité de généralisation sur des scènes urbaines complexes.

6. Conclusion :

Ce chapitre a permis d'explorer les bases théoriques nécessaires pour comprendre le fonctionnement et les atouts du Deep Learning dans le domaine du traitement d'images. En mettant l'accent sur les réseaux de neurones, les principales architectures utilisées pour la segmentation, ainsi que sur les particularités des images urbaines et les jeux de données

disponibles, nous avons rassemblé les éléments fondamentaux pour aborder la suite du travail.

Dans ce dernier chapitre, nous allons présenter les résultats et la discussion en détaillant le modèle proposé, les langages utilisés, puis en analysant les résultats obtenus.

Chapitre 3 :

Implémentation et analyse expérimentale

III. Chapitre 3 : Implémentation et analyse expérimentale

1. Introduction :

Ce chapitre décrit les étapes techniques mises en œuvre pour développer un modèle de segmentation sémantique appliqué à des scènes urbaines. Il débute par la présentation de l'environnement de développement utilisé, notamment Google Colab et les principales bibliothèques Python mobilisées. Le jeu de données Cityscapes est ensuite préparé à travers différentes opérations de structuration, de prétraitement et d'encodage.

L'architecture U-Net proposée est décrite, en mettant en évidence les choix réalisés pour l'adapter aux contraintes du projet. La procédure d'apprentissage est présentée, avec les hyperparamètres retenus et les mécanismes d'optimisation mis en œuvre. Enfin, les performances du modèle sont évaluées à l'aide de métriques quantitatives et de résultats visuels, avant de proposer une interface interactive avec Gradio permettant une expérimentation utilisateur simple et intuitive.

2. Jeu de données utilisé :

Le jeu de données Cityscapes est un ensemble de données public largement utilisé en vision par ordinateur, spécifiquement conçu pour la segmentation sémantique de scènes urbaines. Il a été développé pour fournir un environnement réaliste et complexe, permettant l'analyse d'images capturées en conditions réelles dans 50 villes européennes, principalement en Allemagne.

Le dataset se compose de 5000 images en haute résolution (2048 × 1024 pixels), annotées au niveau pixel selon 30 classes sémantiques telles que piétons, routes, bâtiments, trottoirs, véhicules, panneaux de signalisation, etc. Les images sont réparties en trois ensembles : train, val, et test, chacun organisé par ville. Les annotations sont fournies sous plusieurs formats, notamment labelIds (entiers) et color (images colorées), afin de s'adapter à différents pipelines de traitement.

Le choix d'utiliser Cityscapes dans ce projet se justifie par plusieurs atouts majeurs : la précision des annotations, la résolution élevée des images, la diversité des scènes urbaines, ainsi que la structure bien organisée des fichiers. Ces qualités en font un dataset adapté pour entraîner et évaluer un modèle de segmentation sémantique tel qu'UNet. Le jeu de données est disponible sur le site officiel du projet Cityscape.[90]

3. Préparation et structuration du jeu de données :

Pour ce projet, le jeu de données **Cityscapes** a été téléchargé depuis sa source officielle. Deux fichiers principaux ont été récupérés :

- **leftImg8bit_trainvaltest.zip :** contient les images RGB organisées par ville, dans les sous-dossiers train, val, et test.
- **gtFine_trainvaltest.zip :** contient les masques d'annotations fines correspondants, avec la même structure.

Après extraction des archives, il a été constaté que le sous-dossier test ne comporte pas d'annotations exploitables. En effet, les fichiers du répertoire gtFine/test sont entièrement noirs, c'est-à-dire que tous les pixels sont à zéro, ce qui indique qu'aucune classe annotée n'y est représentée. Cela reflète le choix des créateurs de Cityscapes de ne pas fournir les annotations de test pour un usage local, celles-ci étant réservées à l'évaluation officielle sur leur plateforme.

Par conséquent, le dossier test a été supprimé, et seuls les ensembles train et val ont été conservés pour la suite du travail.

La base de données résultante contient 3475 images annotées de manière fine et exploitable.

4. Répartition des données :

Les images annotées issues du jeu de données préparé ont été réparties en trois sous-ensembles distincts : 80 % pour l'entraînement, 10 % pour la validation et 10 % pour le test. Cette division a été réalisée de manière aléatoire dans le but de séparer les données utilisées pendant l'apprentissage de celles destinées à l'évaluation, assurant ainsi une mesure fiable des performances du modèle.

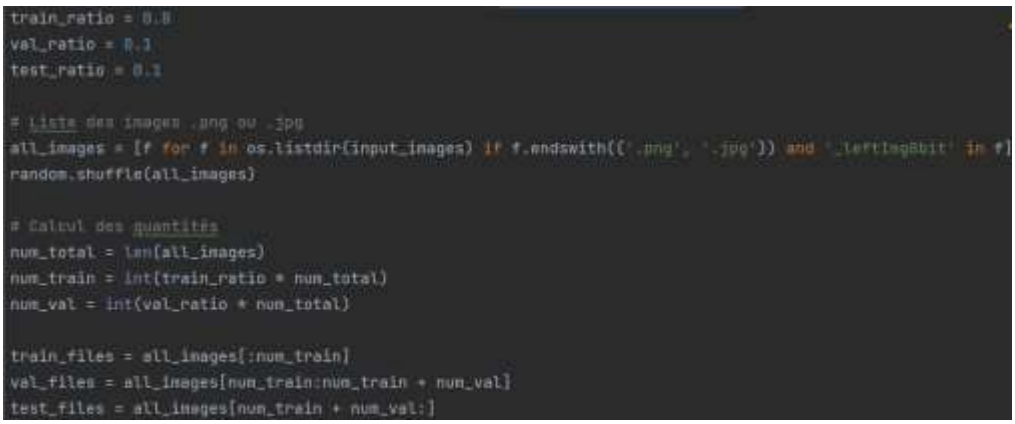


Figure III-1 : Répartition des données en ensembles d'entraînement, validation et test.

5. Chargement dans Google Colab :

Après la réorganisation locale du dataset, celui-ci a été transféré dans Google Drive afin d'être accessible depuis Google Colab. Le montage du Drive a permis d'accéder directement aux sous-dossiers contenant les images et les masques, facilitant ainsi le chargement du jeu de données pour l'entraînement du modèle.

Figure III-2 : Montage du Google Drive dans l'environnement Google Colab.

6. Pipeline de prétraitement :

Avant d'être utilisées pour l'entraînement du modèle UNet, les images et les masques ont été soumis à une série d'opérations de prétraitement. Ces étapes permettent de garantir la compatibilité des données avec l'architecture du modèle, tout en assurant un apprentissage stable et efficace.

- **Redimensionnement des images et masques :**

Toutes les images d'entrée et leurs masques associés ont été redimensionnés à une taille uniforme de 192 × 192 pixels, afin de s'adapter à l'architecture UNet et de limiter la consommation mémoire pendant l'entraînement. Cette taille a été choisie comme compromis entre niveau de détail et performance d'exécution.

- **Normalisation des images :**

Les images ont été **normalisées** en divisant chaque valeur de pixel par 255. Cela ramène les valeurs dans l'intervalle [0, 1], ce qui améliore la stabilité de l'entraînement du réseau de neurones convolutionnels.

- **Encodage des masques de classes :**

Les masques d'annotation, initialement en couleurs RGB, ont été convertis en entiers (*trainId*) représentant les classes. Cette opération consiste à comparer chaque pixel avec la couleur de référence de chaque classe (stockée dans le dictionnaire trainId2color) afin d'assigner l'étiquette correspondante. Une fonction vectorisée (find_closest_labels_vectorized) a été utilisée pour optimiser ce processus.
Le format obtenu est requis pour l'entraînement du modèle avec la fonction de perte *Sparse Categorical Crossentropy*, qui attend des labels codés sous forme d'entiers.

- **Conversion en tenseurs et mise en forme :**

Les images et les masques encodés ont été convertis en tableaux NumPy, puis transformés en tenseurs de type float32, compatibles avec l'environnement TensorFlow/Keras. Cette étape permet d'alimenter efficacement le modèle en données numériques structurées, prêtes à être utilisées pour l'entraînement.

La fonction np.stack() a été utilisée pour regrouper les lots d'images et de masques dans des tableaux multidimensionnels.

- **Vérification de la cohérence des données :**

Avant l'entraînement, une vérification des dimensions et des valeurs uniques a été effectuée sur les masques encodés pour s'assurer que toutes les classes attendues sont bien présentes. Cela garantit la validité des données et la cohérence de l'évaluation.

Ce pipeline de prétraitement a permis de structurer les données de manière optimale pour l'entraînement du modèle U-Net, tout en tenant compte des contraintes techniques propres à l'environnement Google Colab et au format du dataset Cityscapes.

7. Environnement de développement et outils utilisés :

7.1. Langage de programmation Python :

Python est un langage de programmation interprété, orienté objet à la syntaxe claire et aux structures de données modernes, faisant de lui une solution idéale pour le développement rapide et le scripting. Son écosystème riche, avec des bibliothèques comme TensorFlow, et PyTorch, en a fait le langage phare du deep learning, car il permet de concevoir des modèles complexes implémentés dans les produits et systèmes à destination des utilisateurs comme les chercheurs ou les développeurs. En effet, sa flexibilité, sa lisibilité et le large éventail de sa communauté contribuent à en faire un environnement de codage de choix.[77][78]

Figure III-3 : Logo Python.**[96]**

7.2 Environnement Google Colab :

Google Colab est un service cloud gratuit et accessible fondé sur Jupyter Notebook, offrant la

possibilitéd'exécuter du code directement à travers un simple navigateur, le tout sans avoir à les configurer aupréalable. Cet outil est tout particulièrement prisé par les professionnels spécialisés dans le deep learning etla science des données puisqu'il propose un accès libre à de puissantes ressources de calcul, quicomprennent des GPU et TPU, ce qui constitue un réel plus pour l'entraînement des modèles complexes.[79]

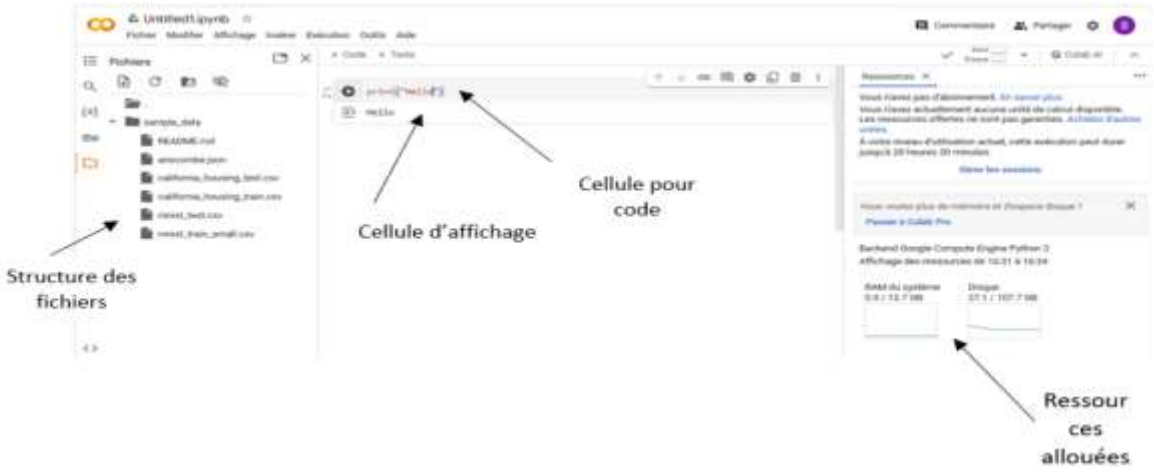


Figure III-4 : Interface de Google colab

7.3. Bibliothèques utilisées :

- **Tensorflow :**

TensorFlow est une plateforme open source globale pour le machine learning et Deep learning. Elle offre un écosystème complet et flexible d'outils, bibliothèques et ressources communautaires qui permettent aux chercheurs d'explorer le ML et DL, et aux développeurs d'implémenter et de déployer simplement des applications d'IA.[80]

Figure III-5 : Logo TensorFlow.**[98]**

- **Keras :**

Keras est une API Python dédiée au deep learning qui est unique par sa simplicité et sa flexibilité, qui peut être exécutée sur TensorFlow ou PyTorch, et ayant vocation à assurer une montée en complexité fluide. Keras permet à un néophyte de mettre en œuvre au plus vite des modèles en utilisant une syntaxe intuitive mais permet également à l'expert de garder la main grâce au recours à des fonctionnalités avancées.

Figure III-6 : Logo Keras.**[99]**

- **NumPy :**

NumPy constitue le paquet de base pour le calcul scientifique en Python. Il fournit un objet tableau multidimensionnel ainsi que des objets dérivés (tableaux masqués et matrices) et des fonctions rapides sur ces tableaux (opérations mathématiques, logiques, de reformatage de forme, trier, sélectionner, E/S, transformées de Fourier discrètes, algèbre linéaire de base, opérations statistiques de base, simulation aléatoire, etc.).[81]

Figure III-7 : Logo NumPy.**[100]**

- **PIL :**

Pillow, de son acronyme indicatif, “Python Imaging Library”, est une puissante bibliothèque Python pour le traitement d’images qui supporte de très nombreux formats d’échanges (jpeg, png, etc.), réalise assez rapidement des opérations de manipulation requises, et sait bien gérer les données pixels. Idéal pour passer d’une manipulation simple à la plus complexe, elle est l’un des outils à avoir dans son panier à outils pour tout projet Python d’application sur des images.[82]

Figure III-8 : Logo Pillow (PIL).**[101]**

- **Matplotlib :**

Matplotlib est une bibliothèque complète en Python, pouvant réaliser des visualisations statiques, animées et interactives. On peut faire du simple (pour les besoins modérés) ou du complexe (pour les besoins plus poussés), pour générer de beaux graphiques professionnels et personnalisables, ce qui est particulièrement adapté à l’écosystème scientifique de Python.[83]

Figure III-9 : Logo Matplotlib.**[102]**

- **Gradio :**

Gradio se présente comme une bibliothèque Python open-source, pratique pour le déploiement d’interfaces web interactives, notamment pour l’emploi des modèles d’intelligence artificielle et de machine learning pour les rendre accessibles à tous.

Conçu pour un accès démocratique aux modèles les plus complexes, elle permet de réaliser sa propre démonstration exploitable en quelques lignes de code et sans connaissances particulières en développement web.[84]

Figure III-10 : Logo Gradio.**[103]**

- **Scikit learn :**

Scikit-learn est un outil de machine learning d'apprentissage supervisé et non supervisé dont le code source est ouvert. Il offre, en outre, une multitude d'outils pour modèle, ajustement, préprocessing, sélection et évaluation de modèle mais aussi de bien d'autres choses.[85]

Figure III-11 : Logo Scikit-learn.**[104]**

- **Seaborn :**

Seaborn est une bibliothèque de visualisation de données avancée en Python, conçue pour faciliter le graphisme des données statistiques. Elle allie esthétisme et lisibilité à une grande simplicité d'utilisation, tout en reposant sur Matplotlib.[86]

Figure III-12 : Logo Seaborn.**[105]**

- **Tqdm :**

tqdm (de l'arabe « taqadum » ; « progrès ») est une bibliothèque Python à la fois minimaliste, puissante et hautement personnalisable destiné à ajouter facilement des barres de progression interactives aux boucles d'itération, tout en alliant (simple) usage et (forte) capacité de personnalisation.[87]

- **Shutil :**

Le module 'shutil' (abréviation de "shell utilities") est un module standard de la librairie Python fournissant des fonctions de haut niveau pour la gestion des fichiers et des répertoires : copie, suppression, archivage, etc. Il complète les fonctionnalités déjà fournies par le module os, en proposant d'autres opérations plus intéressantes.[88]

- **Opencv :**

OpenCV, acronyme d'Open Source Computer Vision Library, désigne une bibliothèque open source particulièrement célèbre dans plusieurs domaines comme la vision par ordinateur, le traitement d'images, et l'apprentissage machine.[89]

Figure III-13 : Logo OpenCV.

8. Modèle U-Net proposé :

Le modèle que nous avons implémenté repose sur une architecture inspirée d'U-Net, une référence dans la segmentation d'images, notamment médicales et urbaines. Conformément à la description théorique présentée au chapitre II, notre version conserve la structure encodeur–décodeur symétrique avec connexions de saut, tout en intégrant plusieurs ajustements visant à améliorer les performances sur les images complexes du jeu de données Cityscapes.

Pour cela, nous avons conçu une version modifiée d'U-Net, que nous appellerons ici notre modèle U-Net proposé. Cette version reprend la structure générale du modèle original, mais y

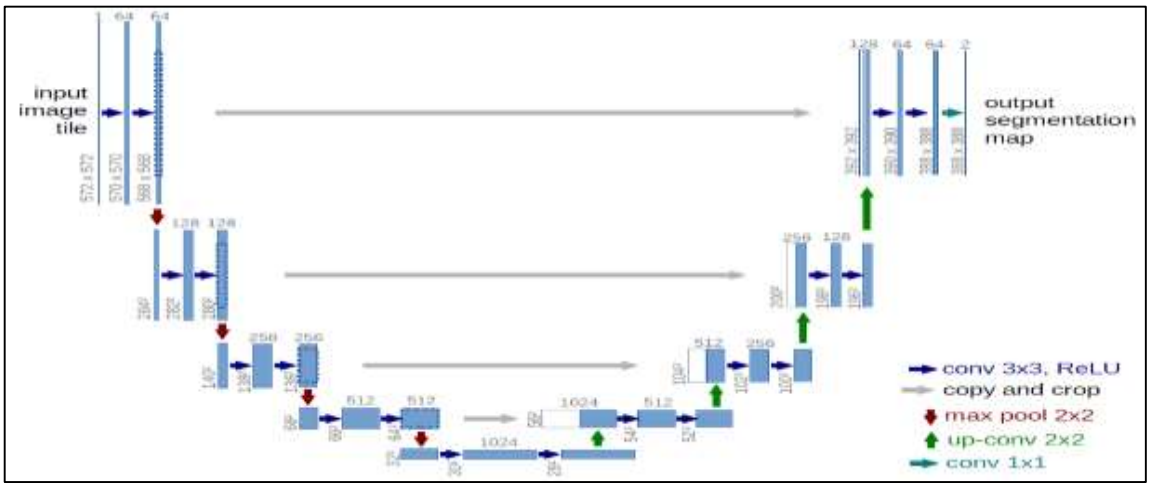
ajoute plusieurs ajustements spécifiques, destinés à améliorer la stabilité de l'apprentissage, la capacité de généralisation, ainsi que la précision des segmentations en contexte urbain.

8.1. Particularités de l'implémentation

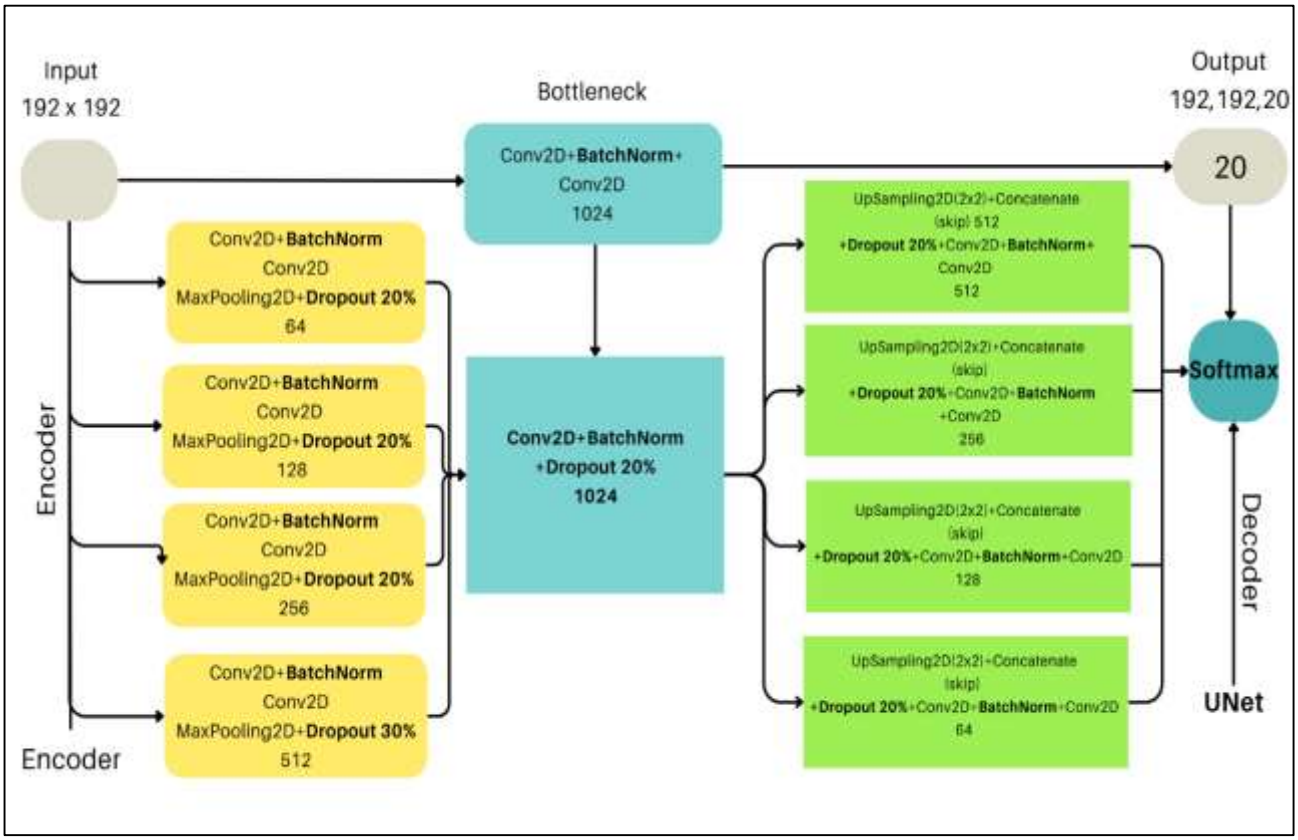
Comparée à l'U-Net original, notre version comporte plusieurs améliorations notables :

- **Ajout de Batch Normalization** après chaque convolution, pour accélérer l'apprentissage et stabiliser la convergence ;
- **Application de Dropout**de manière régulière pour renforcer la généralisation ;
- Ajout **d'uneskip connection interne au goulot** (concatenation profonde) pour enrichir les représentations ;
- **Ajout d'un skip connection interne au goulot** (concaténation profonde), pour enrichir les représentations extraites avant la reconstruction ;

La figure suivante compare l'architecture U-Net standard (schéma a) à notre modèle U-Net proposé (schéma b), intégrant l'ensemble des améliorations structurelles décrites ci-dessus.



(a) : Schéma de l'architecture du modèle U-Net.



(b) : Schéma de l'architecture du modèle U-Net proposé.

Figure III-14 : Comparaison entre l'architecture U-Net standard et le modèle U-Net proposé.

8.2.Paramètres et taille du modèle :

Le modèle compte environ 55 millions de paramètres, dont la quasi-totalité est entraînable. Cette taille relativement importante s'explique par l'utilisation d'un grand nombre de filtres convolutifs tout au long du réseau, par l'empilement intensif de couches dans la partie centrale (goulot), ainsi que par l'emploi de connexions de saut qui introduisent des concaténations de grande dimension entre les couches d'encodage et de décodage.

9. Entraînement du modèle :

L'apprentissage du modèle repose sur une approche empirique, consistant à ajuster progressivement les paramètres afin d'optimiser la qualité de la segmentation sémantique sur

les scènes urbaines. L'objectif principal était de garantir une bonne capacité de généralisation tout en assurant la stabilité de l'entraînement.

9.1. Hyperparamètres utilisés

Plusieurs hyperparamètres ont été définis de manière expérimentale après plusieurs essais :

Optimiseur : *Adam*, a été sélectionné pour sa rapidité de convergence et sa gestion dynamique du taux d'apprentissage.

Fonction de perte : *Sparse Categorical Crossentropy*, Cette fonction est particulièrement adaptée à la segmentation sémantique multi-classes, où chaque pixel est associé à une étiquette entière correspondant à une classe.

Époques : 50, nombre suffisant pour permettre au modèle d'apprendre efficacement tout en évitant le surajustement.

Taille de batch : 16, offrant un bon compromis entre stabilité d'apprentissage et consommation mémoire.

9.2. Stratégies d'optimisation

Afin de renforcer la robustesse du modèle et de prévenir le surapprentissage, nous avons mis en œuvre plusieurs stratégies d'optimisation reconnues pour leur efficacité dans les tâches de deep learning :

L'augmentation de données : Nous avons appliqué des transformations aléatoires sur les images d'entraînement, telles que des flips horizontaux, des rotations, des zooms, ou encore des variations de luminosité. Ces techniques ont pour but d'accroître artificiellement la diversité du jeu de données sans nécessiter de nouvelles annotations, ce qui améliore la capacité de généralisation du modèle. Cette approche est couramment utilisée dans la littérature pour la segmentation sémantique.[91]

Model Checkpoint : Cette technique a été utilisée pour sauvegarder automatiquement la version du modèle offrant la meilleure performance sur les données de validation, évitant ainsi de retenir un modèle surentraîné.

Early Stopping : Ce mécanisme permet d'arrêter automatiquement l'entraînement lorsque la performance sur les données de validation cesse de s'améliorer pendant un certain nombre d'époques. Il permet ainsi d'éviter le surapprentissage et de limiter le temps de calcul inutile.

Learning Rate Scheduler : Cette technique permet d'ajuster dynamiquement le taux d'apprentissage au fil des époques, afin de favoriser une convergence progressive et plus stable du modèl.

10.Analyse et discussion des résultats :

Cette section présente les résultats obtenus à l'issue de l'entraînement du modèle de segmentation sémantique. L'évaluation repose à la fois sur des indicateurs quantitatifs, issus de la phase d'apprentissage, et sur des résultats visuels permettant d'apprécier la qualité des prédictions.

Concernant les courbes d'apprentissage : La figure ci-dessous illustre l'évolution des principales métriques d'évaluation, à savoir la fonction de perte (loss) et la précision (accuracy), sur les ensembles d'entraînement et de validation.

La précision (accuracy) représente le rapport entre le nombre de prédictions correctes et le nombre total de prédictions, et s'exprime par la formule suivante [22] :

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \qquad \textbf{Équation III-01}$$

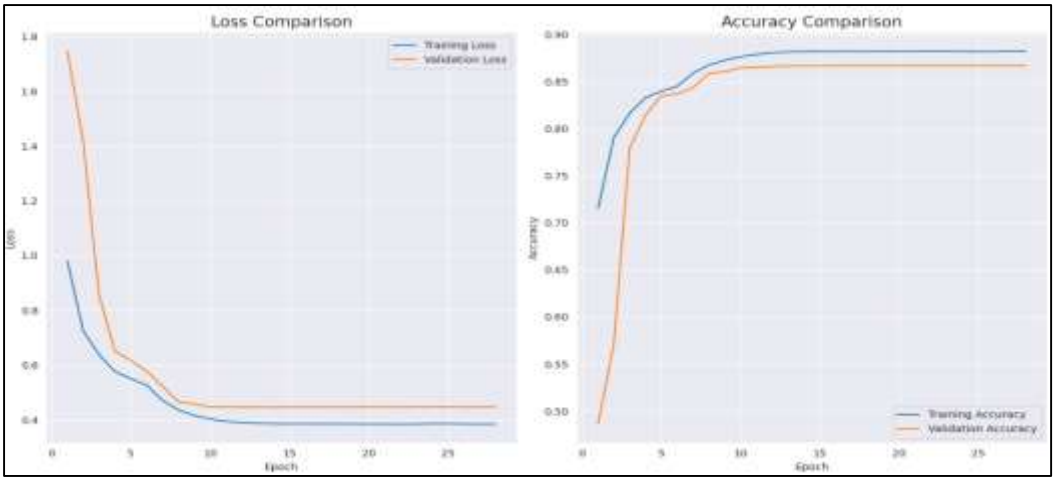


Figure III-15 : Courbes de perte et de précision sur les ensembles d'entraînement et de validation.

L'analyse de ces courbes montre une diminution régulière de la perte et une amélioration continue de la précision. Le modèle atteint en fin d'entraînement une

précision de 88,32 % avec une perte de 0,38 sur l'ensemble d'apprentissage, et une précision de 86,71 % avec une perte de 0,45 sur l'ensemble de validation.

La stabilité entre les courbes d'entraînement et de validation témoigne d'une bonne capacité de généralisation, sans signe notable de sur apprentissage.

S'agissant de la performance par classe, une matrice de confusion a été générée à partir des prédictions effectuées sur l'ensemble de test, en considérant 19 classes sémantiques principales.

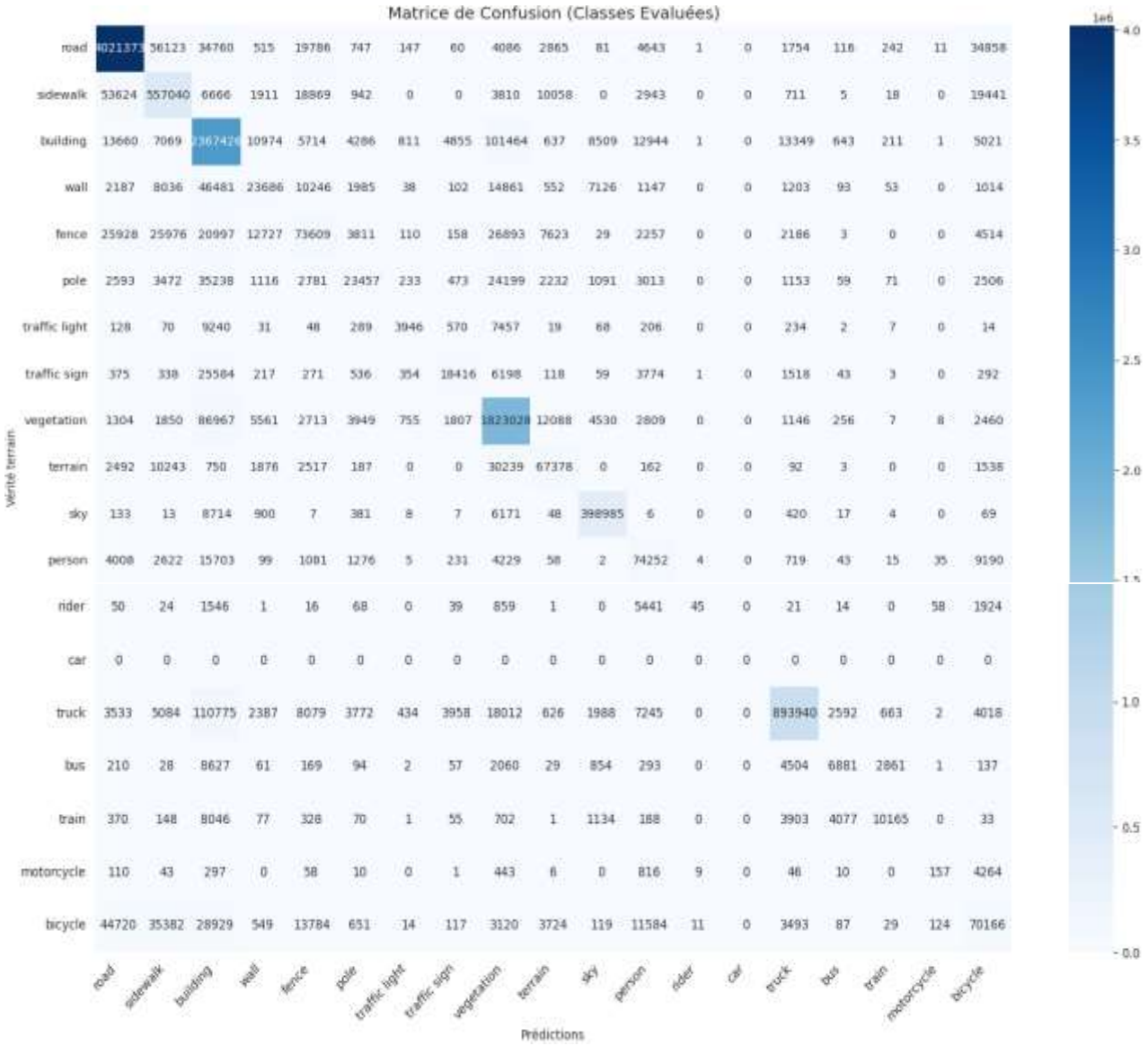


Figure III-16 : Matrice de confusion sur les prédictions du modèle.

Cette matrice permet d'observer la correspondance entre les étiquettes réelles et les classes prédites par le modèle. Les valeurs diagonales indiquent les prédictions correctes, tandis que les valeurs hors-diagonale traduisent les erreurs de classification.

L'analyse montre que les performances sont élevées pour les classes les plus fréquentes et visuellement dominantes, tandis que les classes rares ou visuellement similaires restent plus difficiles à segmenter.

Enfin, pour illustrer concrètement la qualité des prédictions réalisées, cette section présente plusieurs exemples visuels issus du jeu de test. Chaque exemple juxtapose l'image originale, le masque annoté (vérité terrain) et le masque généré par le modèle.

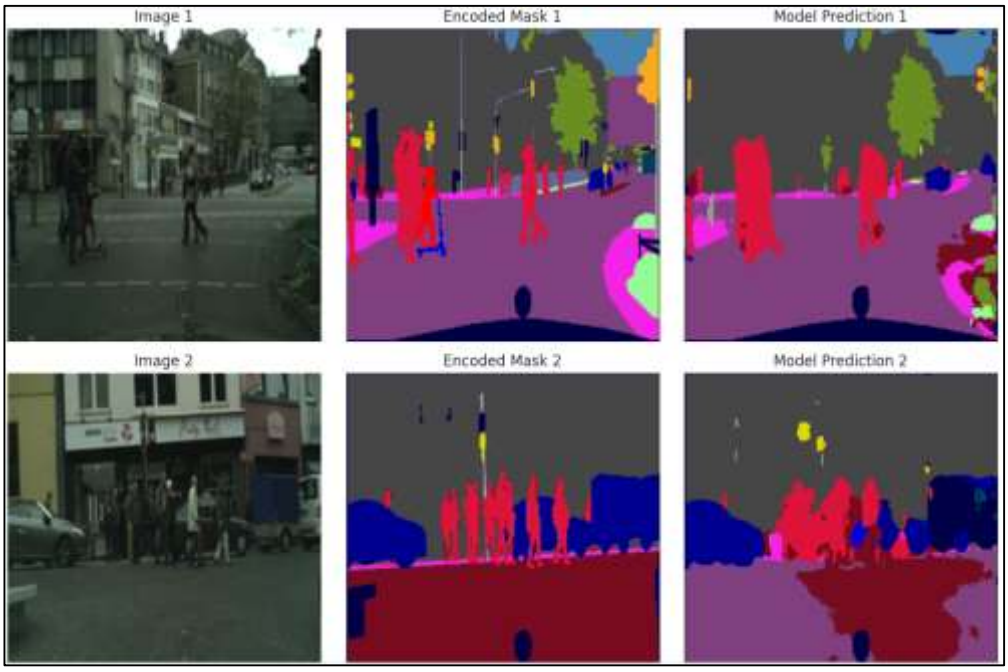


Figure III-17 : Exemples de segmentation sur des images de test (de gauche à droite image originale, masque réel, prédiction du modèle).

Dans l'ensemble, les résultats obtenus mettent en évidence la capacité du modèle à produire une segmentation fiable et cohérente des scènes urbaines. Certaines limites persistent néanmoins sur les classes peu représentées ou visuellement similaires, ouvrant des pistes d'amélioration pour les travaux futurs.

Chapitre 3

11.Comparaison avec d'autres modèles de segmentation :

Au cours de ce travail, plusieurs architectures ont été explorées dans le but d'améliorer la segmentation sémantique des scènes urbaines. Outre le modèle final U-Net proposé, deux autres configurations ont été testées :

- une architecture inspirée de DeepLabV3, reposant sur un backbone ResNet50 pré-entraîné sur ImageNet et un module ASPP (Atrous Spatial Pyramid Pooling) réimplémenté dans un format adapté au projet,
- une version d'U-Net intégrant un encodeur VGG16 également pré-entraîné sur ImageNet, dans une logique de transfert learning.

Ces modèles ont été évalués sur le même ensemble de test, à l'aide des métriques de précision (accuracy) et de fonction de perte (loss). Le tableau ci-dessous synthétise les résultats les plus significatifs :

Table 1 : Comparaison des performances des modèles testés.

Modèle	Accuracy (%)	Loss
U-Net (proposé)	**87.56**	**0.44**
DeepLabV3 (ResNet50)	83.50	0.57
U-Net avec VGG16	81.60	0.65

Un modèle U-Net simple a également été expérimenté en début de projet. Cependant, ses performances nettement inférieures (accuracy < 79 %) et l'absence de mécanismes avancés ont conduit à son exclusion du tableau comparatif. Il a été rapidement écarté au profit de solutions plus robustes.

L'analyse des résultats met en évidence la supériorité du modèle U-Net proposé, qui offre le meilleur compromis entre simplicité architecturale et performances. L'architecture DeepLabV3, bien que plus complexe, n'a pas permis d'atteindre des résultats équivalents. Quant au modèle U-Net avec VGG16, il présente une légère tendance au surapprentissage et une précision moindre, ce qui a motivé son exclusion comme solution finale.

Afin de situer notre modèle dans le contexte des approches existantes, nous avons également confronté ses résultats à ceux du modèle U-Net modifié de Baroudi et al., issu

de l'état de l'art sur la segmentation des scènes urbaines. Le tableau suivant présente cette comparaison :

Tableau 2 : Comparaison avec un modèle de l'état de l'art.

Modèle	Accuracy (%)	Loss
Notre modèle U-Net (proposé)	**87.56**	**0.44**
U-Net modifié – Baroudi et al.[6]	87.03	0.55

Notre modèle présente ainsi des performances légèrement supérieures en termes de précision, tout en affichant une fonction de perte plus faible, ce qui traduit une meilleure convergence et une stabilité accrue de l'apprentissage.

12.Limites et difficultés rencontrées :

Au cours de cette étude, plusieurs limitations ont été rencontrées, qu'elles soient d'ordre technique, méthodologique ou liées aux ressources disponibles. Les identifier permet de mieux évaluer la portée des résultats obtenus.

12.1. Contraintes liées à l'environnement de développement :

L'utilisation de Google Colab a imposé plusieurs contraintes, notamment une mémoire RAM limitée et une durée restreinte des sessions. Il a ainsi été nécessaire de redimensionner les images à 192×192 pixels au lieu de leur résolution native (1024×1024), ce qui a pu réduire la précision des segmentations. De plus, les crashs fréquents ont rendu difficile l'entraînement de modèles plus complexes sur de longues périodes.

L'utilisation de Google Colab comme environnement principal a imposé plusieurs restrictions notables :

- Une mémoire RAM limitée **.**
- Une durée de session restreinte, provoquant des interruptions fréquentes .
- L'impossibilité de déployer des architectures très profondes ou de travailler avec des images de grande résolution.

Ces limitations ont conduit au redimensionnement des images à 192×192 pixels, au lieu de leur taille native (1024×1024), ce qui a probablement affecté la précision de la segmentation, en particulier sur les objets fins. De plus, l'instabilité de l'environnement a rendu difficile l'entraînement de modèles complexes sur des périodes prolongées.

12.2. Contraintes liées aux données utilisées :

Une première série d'expériences a été menée sur une version du dataset Cityscapes obtenue via Kaggle, contenant 3475 images dans un format particulier, où chaque fichier combinait l'image et son masque côte à côte. Après séparation, cette base a permis d'entraîner un premier modèle.

Ensuite, le dataset officiel, plus riche et mieux structuré, a été utilisé. L'absence d'annotations pour l'ensemble test a cependant contraint l'expérimentation aux seuls ensembles d'entraînement et de validation, ramenant le volume exploité à 3475 images également. Néanmoins, la qualité supérieure des images a permis d'obtenir des résultats plus convaincants.

Enfin, une tentative d'utilisation d'un dataset externe de plus de 9000 images a dû être abandonnée en raison des limitations mémoire de Google Colab.[92]

13.Interface graphique développée avec Gradio :

Une interface simple a été développée à l'aide de la bibliothèque Gradio afin de faciliter l'utilisation du modèle de segmentation. Elle permet de téléverser une image et de visualiser instantanément le résultat produit par le modèle U-Net.

Figure III-18 : Interface avant prédiction.

L'utilisateur sélectionne une image dans la partie gauche de l'interface (Figure III-18), puis clique sur Submit pour lancer la segmentation. Le bouton Clear permet de réinitialiser l'entrée.

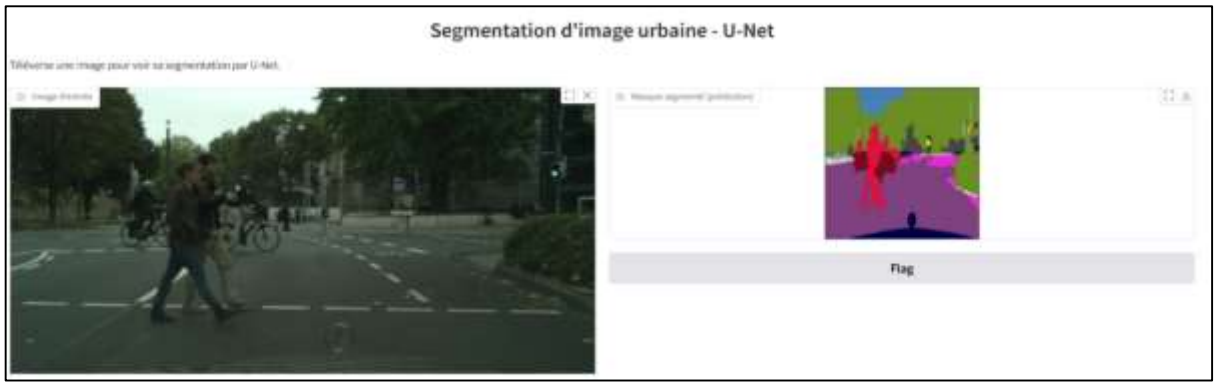


Figure III-19 : Exemple de prédiction (à gauche : image originale, à droite : image segmentée).

Une fois la prédiction effectuée, l'image originale et son masque segmenté sont affichés côte à côte (Figure III-19). Le masque visualise les différentes classes sémantiques selon les couleurs définies dans *trainId2color*.

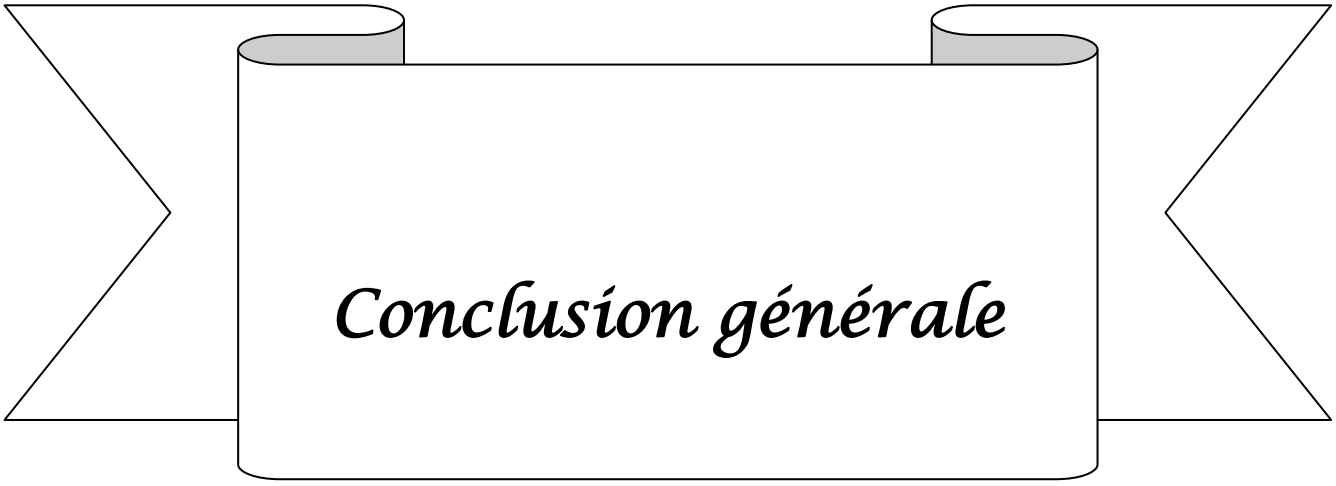
Cette interface rend le modèle interactif et facilement exploitable, tout en fournissant une évaluation visuelle immédiate de ses performances.

14.Conclusion :

L'implémentation du modèle de segmentation a permis de mettre en application les concepts théoriques abordés précédemment. Grâce à une préparation rigoureuse des données, à la conception d'une architecture U-Net adaptée et à un entraînement soigneusement paramétré, des résultats satisfaisants ont été obtenus sur des scènes urbaines complexes.

Les performances du modèle ont été validées à la fois par des indicateurs quantitatifs et des résultats visuels cohérents. L'utilisation d'une interface Gradio a permis de rendre le modèle accessible et facilement exploitable pour une démonstration interactive. Malgré certaines limitations techniques, les expérimentations menées ont mis en évidence la robustesse du modèle proposé, tout en soulignant des pistes d'amélioration pour de futurs travaux.

Ce chapitre constitue ainsi la base expérimentale sur laquelle repose l'analyse critique des résultats et la discussion des perspectives, développées dans le chapitre suivant.


Conclusion générale

Conclusion générale :

Au terme de ce travail consacré à la segmentation sémantique des images urbaines par apprentissage profond, nous avons pu démontrer l'apport des réseaux de neurones convolutifs, et plus particulièrement de l'architecture U-Net, pour répondre aux besoins croissants des systèmes modernes d'analyse d'images. Dans un monde où les flux d'images urbaines se multiplient de manière exponentielle, la capacité à segmenter avec précision ces scènes constitue un enjeu majeur pour des applications telles que les véhicules autonomes, les villes intelligentes et les systèmes avancés de surveillance.

La problématique qui a guidé ce travail portait sur la capacité d'une architecture U-Net améliorée à optimiser la segmentation d'images urbaines face aux défis posés par la richesse visuelle des scènes et la complexité des objets représentés. Pour y répondre, nous avons adopté une démarche méthodologique progressive, en nous appuyant d'abord sur les bases théoriques du traitement d'images et du Deep Learning, avant de concevoir et d'évaluer notre propre modèle sur la base Cityscapes. Le choix de cette base nous a permis de confronter notre approche à des scènes réalistes et variées, en adéquation avec les exigences des applications concrètes.

Les résultats obtenus confirment que l'architecture U-Net, adaptée aux spécificités des environnements urbains, permet d'atteindre des performances satisfaisantes en matière de segmentation sémantique. Notre modèle a su identifier de manière précise les différentes classes d'objets présentes dans les images, en surmontant certains des défis liés aux variations d'échelle, aux occlusions et à la complexité des scènes. Ces résultats illustrent le potentiel des approches basées sur le Deep Learning pour enrichir les capacités d'analyse visuelle des systèmes intelligents et automatisés.

Cependant, notre travail comporte également des limites. Certaines classes d'objets moins représentées dans la base de données, ou présentant des formes très variées, demeurent plus difficiles à segmenter. Par ailleurs, l'optimisation du modèle pour un usage en temps réel, indispensable pour certaines applications embarquées, reste un axe de développement essentiel. De plus, bien que les performances obtenues soient encourageantes, l'adaptabilité du modèle à d'autres contextes géographiques ou à des environnements urbains différents de ceux présents dans la base Cityscapes reste à renforcer.

Plusieurs pistes peuvent être envisagées pour surmonter ces limitations et enrichir les résultats obtenus. L'utilisation de ressources matérielles plus puissantes, telles que des GPU

de dernière génération ou des plateformes cloud plus robustes, permettrait d'explorer des architectures de réseaux plus profondes et de traiter des images en haute résolution. L'enrichissement des données, en intégrant un nombre plus important d'images ou en adoptant des techniques de data augmentation plus avancée, contribuerait à améliorer la diversité du jeu d'entraînement et la robustesse du modèle. L'exploration de nouvelles architectures intégrant des mécanismes d'attention ou de réseaux de type Vision Transformers pourrait permettre une meilleure prise en compte des objets de petite taille ou visuellement ambigus. Enfin, un effort ciblé sur le traitement des classes minoritaires, par le biais de stratégies de rééquilibrage des classes ou d'une annotation renforcée, permettrait d'améliorer la qualité de la segmentation sur les catégories les moins représentées.

Ce travail ouvre ainsi des perspectives de recherche prometteuses. L'intégration de ces améliorations méthodologiques, combinée à l'évolution rapide des techniques d'apprentissage profond, devrait permettre de développer des solutions toujours plus performantes pour la segmentation d'images urbaines. En conclusion, cette recherche apporte une contribution modeste mais significative à l'amélioration des approches actuelles. Elle confirme l'intérêt des architectures de type U-Net pour relever les défis complexes posés par l'analyse automatique des scènes urbaines modernes, et ouvre la voie à de nouveaux travaux dans ce domaine en constante évolution.

Bibliographie

[1] M. Amina et F. Fadia, «Segmentation des Images par Contours Actifs : Application sur les Images Satellitaires à Haute Résolutions,» Université Abou Bekr Belkaïd – Tlemcen, Tlemcen, Algérie, 2012.

[2] A. HAMANA et Y. GOSSA, «Stéganographie d'une image numérique,» Université Kasdi Merbah – Ouargla, Ouargla, Algérie, 2021.

[3] M. SANDELI, «Traitement d'images par des approches bio-inspirées : Application à la segmentation d'images,» Université Constantine 2, Constantine, Algérie, 2014.

[4] Bâches-Publicitaires.com, «Quel est la différence entre une image matricielle ou vectorielle ?,» 14 08 2014. [En ligne]. Available: https://www.baches-publicitaires.com/blog/actualites/vectorisation-cest/. [Accès le 10 03 2025].

[5] Y. Tamdrari et D. Bakiri, «Segmentation d'images en niveaux de gris par modélisation basée sur les niveaux de gris ordonnés,» Université Mouloud Mammeri de Tizi-Ouzou (UMMTO), 2011.

[6] R. BAROUDI et Z. BENCHIHA, «Approche d'apprentissage profond pour la segmentation sémantique d'images,» Université d'Ain Temouchent - Belhadj Bouchaib, Ain Temouchent, 2024.

[7] J.-m. M. Biringanine, «La liaison automatique des plusieurs images perçues sur un scanner,» 2008.

[8] M. &. D. D. Taibaoui, «La découverte des concepts sémantiques cachés avec plusieurs niveaux d'abstraction pour la recherche d'images,» UNIVERSITE KASDI MERBAH OUARGLA, OUARGLA, 2013.

[9] S. &. H. A. Benfriha, «Segmentation d'image par Coopération région-contours,» Ouargla, Algérie, 2016.

[10] I. Khenfer, E. A. Bettayeb et D. E. Salhi, «Filtrage Adaptatif 2-D pour la Restauration d'images perturbées par du Bruit Impulsionnel,» UNIVERSITÉ KASDI MERBAH OUARGLA, Ouargla, Algérie, 2020.

[11] P. Proulx, «Segmentation de captures d'images aériennes appliquée à la navigation visuelle assistée de drones,» Université du Québec en Outaouais, Département d'informatique et ingénierie, Gatineau, Québec, Canada, 2024.

[12] L. G. D. Services, «An Extensive Overview: 3 types of image segmentation,» 2023. [En ligne]. Available: https://www.gdsonline.tech/3-types-of-image-segmentation/. [Accès le 20 03 2025].

[13] C. Coello Coello, A. Christiansen et A. Hernández Aguirre, «Multiobjective design optimization of counterweight balancing of a robot arm using genetic algorithms,» chez *Proceedings of the 7th International Conference on Tools with Artificial Intelligence*, Herndon, Virginia, USA,

1995.

[14] M. Kass, A. Witkin et D. Terzopoulos, «Snakes: Active contour models,» *International Journal of Computer Vision*, vol. 1, n° %14, p. 321–331, 1988.

[15] N. P. Tiilikainen, «A Comparative Study of Active Contour Snakes,» Copenhagen, 2007.

[16] G. Michel-Claude et G. Colette, «Traitement des données de télédétection,» Dunod, 1999.

[17] R. C. Gonzalez et R. E. Woods, Digital Image Processing, Pearson, Éd., New York, USA: Pearson Global Edition, 2018.

[18] J. Canny, «A computational approach to edge detection,» IEEE Computer Society, New York, 1986.

[19] R. Deriche, «Optimal edge detection using recursive filtering,» *International Journal of Computer Vision*, vol. 2, p. 167, 1987.

[20] A. N. Benaichouche, «Conception de métaheuristiques d'optimisation pour la segmentation d'images : application aux images IRM du cerveau et aux images de tomographie par émission de positons,» Université Paris-Est, Paris France, 2014.

[21] L. A. Zadeh, «Fuzzy sets,» *Information and Control*, vol. 8, n° %13, p. 338–353, 1965.

[22] I. Goodfellow, Y. Bengio et A. Courville, Deep Learning, Cambridge: The MIT Press, 2016, p. 800.

[23] E. P. B. School, «Deep Learning : Définition, applications et avantages,» EDC Paris, 13 février 2024. [En ligne]. Available: https://www.edcparis.edu/fr/blog/deep-learning-definition-applications-et-avantages. [Accès le 02 04 2025].

[24] LeBigData, «Réseau de Neurones Artificiels : Définition, Fonctionnement, Exemples,» LeBigData, 9 novembre 2021. [En ligne]. Available: https://www.lebigdata.fr/reseau-de-neurones-artificiels-definition. [Accès le 02 04 2025].

[25] Praedictia, «L'histoire des réseaux de neurones,» Praedictia, 20 04 2021. [En ligne]. Available: https://praedictia.com/page/reseaux-de-neurones/lhistoire-des-reseaux-de-neurones.html. [Accès le 04 04 2025].

[26] F. SEO, «L'évolution des réseaux de neurones : du Perceptron à l'apprentissage profond,» Forge SEO, 05 01 2024. [En ligne]. Available: https://forge-seo.com/levolution-des-reseaux-de-neurones-de-la-perceptron-a-lapprentissage-profond-1707177972/. [Accès le 05 04 2025].

[27] T. Nicolas et W. Christian , «Histoire des réseaux de neurones et du deep learning en traitement de signaux et des images,» Grenoble, France, 2023.

[28] Z. Abdi et D. Mahia, «Développement d'un modèle deep learning pour la segmentation et la classification d'images pulmonaires,» Université Ibn Khaldoun - Tiaret, Tiaret, 2023.

[29] I. Aissougui et T. N. Dahemechi, «Modélisation des micromachines/capteurs en utilisant les réseaux de neurones artificiels,» Université 8 Mai 1945, Guelma, 2023.

[30] S. E. Ghamri, «Le neurone biologique,» 29 mai 2016. [En ligne]. Available: https://www.researchgate.net/figure/Le-neurone-biologique-Le-corps-cellulaire-ou-soma-assure-les-fonctions-de-soutien-et-de_fig48_303639668. [Accès le 10 04 2025].

[31] Clevy, «Introduction aux réseaux de neurones (2/3) : neurone biologique et neurone formel,» Clevy Blog, 04 06 2019. [En ligne]. Available: https://blog.clevy.io/fr/introduction-aux-reseaux-de-neurones-2-3-neurone-biologique-et-neurone-formel/. [Accès le 13 04 2025].

[32] T. Akter, M. F. Hossain, M. S. Ullah et R. Akter, «Mortality Prediction in COVID-19 Using Time Series and Machine Learning Techniques,» *Computational and Mathematical Methods*, vol. 2024, n° %110.1155/2024/5891177, p. 17, 2024.

[33] ScienceDirect, «Perceptron,» 2022. [En ligne]. Available: https://www.sciencedirect.com/topics/engineering/perceptron. [Accès le 17 04 2025].

[34] A. Sahli, « Segmentation des images médicales par apprentissage profond,» Université de Tébessa, Tébessa, 2021.

[35] Free-Work, «Qu'est-ce qu'un perceptron et à quoi sert-il ?,» Free-Work, 28 octobre octobre. [En ligne]. Available: https://www.free-work.com/fr/tech-it/blog/actualites-informatiques/quest-ce-quun-perceptron-et-a-quoi-sert-il. [Accès le 19 04 2025].

[36] H. e. D. H. Benameur, « Segmentation et interprétation d'images médicales par apprentissage automatique et système multi-agents,» Université Belhadj Bouchaib, Ain Témouchent, 2022.

[37] A. H. Sabry, «Le modèle de perceptrons multicouches (MLP),» 2015. [En ligne]. Available: https://www.researchgate.net/figure/Le-modele-de-perceptrons-multicouches-MLP_fig1_286404296. [Accès le 22 04 2025].

[38] A. Mezaache, «Classification Des Données Basée Sur Les Réseaux De Neurones Récurrents (RNN),» Université 8 mai 1945 – Guelma, Guelma, 2024.

[39] L. Biri et Y. Guemat, «Implémentation et évaluation des modèles de recommandation basés sur le machine et deep learning,» Université Mouloud MAMMERI , Tizi-Ouzou , 2020.

[40] D. Team, «Perceptrons multicouches dans l'apprentissage automatique : Un guide complet,» DataCamp, 14 11 2024. [En ligne]. Available: https://www.datacamp.com/fr/tutorial/multilayer-perceptrons-in-machine-learning. [Accès le 25 04 2025].

[41] D. Bird, «Réseaux de neurones : Définition et les différents types de réseaux,» Data Bird, 12 06 2024. [En ligne]. Available: https://www.data-bird.co/blog/reseaux-de-neurones. [Accès le 22 04 2025].

[42] M. Ben Lazreg, «Recherche de l'information dans les réseaux de neurones convolutifs pré-entraînés,» École de technologie supérieure, Montréal, Montréal, 2020.

[43] V. H. Phung et E. J. Rhee, «A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets,» *Applied Sciences*, vol. 9, n° %121, p. 4500, 10 2019.

[44] S. Saadoune, «Application de l'Intelligence Artificielle pour la reconnaissance des caractères manuscrits,» Université Belhadj Bouchaib – Aïn Témouchent, Aïn Témouchent, 2022.

[45] K. I. E. W. Saadi et K. Bendahi , «Segmentation D'Image Médicale Via Non superviser Réseau de neurones convolutif,» Université Mohamed Boudiaf de M'sila, M'sila, 2022.

[46] DataScientest, «Hyperparamètres : tout savoir,» DataScientest, 19 05 2023. [En ligne]. Available: https://datascientest.com/hyperparametres-tout-savoir. [Accès le 17 04 2025].

[47] S. BOUBAYA et D. BERBIT , «DEEP LEARNING POUR LA SEGMENTATION,» UNIVERSITE MOHAMED BOUDIAF de M'SILA, M'SILA, 2021.

[48] O.-H. NOUKAS, «Etude Comparative des CNNs et de L'algorithme K-NN en mammographie,» Université Ibn Khaldoun – Tiaret, Tiaret, 2023.

[49] nlpers, «Pooling in CNN – PaddlePedia,» 2021. [En ligne]. Available: https://paddlepedia.readthedocs.io/en/latest/tutorials/CNN/Pooling.html. [Accès le 14 04 2025].

[50] I. I. Group, «Qu'est-ce qu'un réseau de neurones convolutifs ?,» IBM (International Business Machines), 2024. [En ligne]. Available: https://www.ibm.com/fr-fr/think/topics/convolutional-neural-networks. [Accès le 21 04 2025].

[51] I. M. Learning, «Le Dropout : c'est quoi ? Deep Learning — explication rapide,» Inside Machine Learning, 28 11 2022. [En ligne]. Available: https://inside-machinelearning.com/le-dropout-cest-quoi-deep-learning-explication-rapide/. [Accès le 21 04 2025].

[52] S. Saxena, «SoftMax Activation Function for Neural Network,» 01 04 2021. [En ligne]. Available: https://www.analyticsvidhya.com/blog/2021/04/introduction-to-softmax-for-neural-network/. [Accès le 25 04 2025].

[53] R. Kassel, «La fonction de coût en IA : tout ce qu'il faut savoir,» 14 01 2025. [En ligne]. Available: https://datascientest.com/fonction-de-cout-tout-savoir. [Accès le 16 04 2025].

[54] K. Rai, «Understanding Log Loss: A Comprehensive Guide with Code Examples,» 01 09 2024. [En ligne]. Available: https://koshurai.medium.com/understanding-log-loss-a-comprehensive-guide-with-code-examples-c79cf5411426. [Accès le 25 04 2025].

[55] A. Verma, «Epochs, Batch and Iterations in Deep Learning,» 18 01 2019. [En ligne]. Available: https://medium.com/@akankshaverma136/epochs-batch-and-iterations-in-deep-learning-ed319565e85e. [Accès le 27 04 2025].

[56] H. BELLAHMER, «Implémentation et évaluation d'un modèle d'apprentissage automatique pour l'estimation de la valeur marchande de propriétés immobilières,» Université Mouloud Mammeri de Tizi-Ouzou, Tizi-Ouzou, 2020.

[57] M. L. i. P. English, «Deep Learning Course — Lesson 7.4: ADAM (Adaptive Moment Estimation),» Medium, 02 06 2023. [En ligne]. Available: https://medium.com/@nerdjock/deep-learning-course-lesson-7-4-adam-adaptive-moment-estimation-e23434850bfc. [Accès le 27 04 2025].

[58] J. Wang, S. Wang et Y. Zhang, «Deep learning on medical image analysis,» *CAAI Transactions on Intelligence Technology,* vol. 10, p. 1–35, 28 12 2023.

[59] O. Ronneberger, P. Fischer et T. Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, N. Navab, J. Hornegger, W. M. Wells et A. F. Frangi, Éds., Munich, Allemagne, Lecture Notes in Computer Science, vol 9351: Springer International Publishing, 2015, p. 234–241.

[60] Vijay Badrinarayanan, Alex Kendall et Roberto Cipolla, «SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation,» *CoRR (arXiv preprint),* n° %1abs/1511.00561, 02 11 2015.

[61] M. Trokielewicz et A. Czajka, «Data-Driven Segmentation of Post-mortem Iris Images,» chez *2018 Sixth International Workshop on Biometrics and Forensics (IWBF)*, 2018.

[62] K. Simonyan et A. Zisserman, «Very Deep Convolutional Networks for Large-Scale Image Recognition,» chez *International Conference on Learning Representations (ICLR)*, Oxford, 2015.

[63] DataScientest, «Qu'est-ce que le modèle VGG ?,» 27 04 2021. [En ligne]. Available: https://datascientest.com/quest-ce-que-le-modele-vgg. [Accès le 07 04 2025].

[64] N. Balasooriya, B. Dowden, J. Chen, O. De Silva et W. Huang, «In-situ sea ice detection using DeepLabv3 semantic segmentation,» chez *OCEANS 2021: San Diego–Porto*, San Diego–Porto, septembre 2021.

[65] Nanobaly, «Découvrez le Transfer Learning : quand l'IA ne part pas de zéro,» 17 11 2024. [En ligne]. Available: https://www.innovatiana.com/post/transfer-learning-101. [Accès le 23 04 2025].

[66] A. W. Services, «Qu'est-ce que le surajustement ?,» 2023. [En ligne]. Available: https://aws.amazon.com/fr/what-is/overfitting/. [Accès le 02 05 2025].

[67] DataScientest, «Underfitting ou le sous-ajustement en Machine Learning,» 20 12 2022. [En ligne]. Available: https://datascientest.com/underfitting-tout-savoir. [Accès le 02 05 2025].

[68] M. Hu, «Utilizing Machine Learning to Predict the Malignancy of a Breast Tumor,» 2024.

[69] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweile et R. Benenson, «The Cityscapes Dataset for Semantic Urban Scene Understanding,» chez *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* Las Vegas, NV, USA, 2016.

[70] A. Geiger, P. Lenz et R. Urtasun, «Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite,» chez *IEEE Conference on Computer Vision and Pattern Recognition*

(CVPR), Providence, RI, USA, 2012.

[71] G. Ros, L. Sellart, J. Materzynska, D. Vazquez et A. M. Lopez, «The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes,» chez *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016.

[72] V. Nekrasov, C. Shen et I. Reid, «Fast Semantic Segmentation with Spatial Sparsity,» chez *British Machine Vision Conference (BMVC)*, Newcastle, UK, 2018.

[73] R. Khettache et I. Berrahmoune, «Segmentation sémantique des images de drones par le modèle U-Net,» Université Badji Mokhtar – Annaba, Annaba, 2022.

[74] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy et Alan L. Yuille, «DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs,» *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, n° %14, p. 834–848, 12 05 2017.

[75] B. Cheng, I. Misra, A. Schwing, A. Kirillov et R. Girdhar, «Masked-attention Mask Transformer for Universal Image Segmentation,» 2022.

[76] Feng Li, Hao Zhang, Huaizhe Xu, Shilong Liu, Lei Zhang, Lionel M. Ni et Heung-Yeung Shum, «Mask DINO: Towards a Unified Transformer-Based Framework for Object Detection and Segmentation,» chez *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Vancouver, Canada, 2022.

[77] G. v. Rossum, «Python Programming Language – The Python Language Reference Essay,» Python Software Foundation, 2001 . [En ligne]. Available: https://www.python.org/doc/essays/blurb/. [Accès le 01 05 2025].

[78] P. S. Foundation, «Applications for Python,» Python Software Foundation, 2001. [En ligne]. Available: https://www.python.org/about/apps/. [Accès le 01 05 2025].

[79] «Colaboratory – Frequently Asked Questions,» 22 05 2024. [En ligne]. Available: https://research.google.com/colaboratory/faq.html#whats-colaboratory. [Accès le 01 05 2025].

[80] G. O. Source, «TensorFlow,» 16 11 2024. [En ligne]. Available: https://opensource.google/projects/tensorflow. [Accès le 01 05 2025].

[81] N. Developers, «What is NumPy?,» 2025. [En ligne]. Available: https://numpy.org/devdocs//user/whatisnumpy.html. [Accès le 01 05 2025].

[82] P. Developers, «Pillow (PIL Fork) Documentation,» 2024. [En ligne]. Available: https://pillow.readthedocs.io/en/stable/. [Accès le 03 05 2025].

[83] M. Developers, «Matplotlib: Visualization with Python,» 2024. [En ligne]. Available: https://matplotlib.org/. [Accès le 03 05 2025].

[84] G. Team, «Gradio,» 2024. [En ligne]. Available: https://www.gradio.app/. [Accès le 03 05

2025].

[85] S.-l. Developers, «About us – scikit-learn developers,» 2024. [En ligne]. Available: https://scikit-learn.org/stable/about.html#active-core-contributors. [Accès le 04 05 2025].

[86] M. Waskom, «Introduction to Seaborn,» 2024. [En ligne]. Available: https://seaborn.pydata.org/tutorial/introduction.html. [Accès le 14 05 2025].

[87] t. contributors, «tqdm - A Fast, Extensible Progress Bar for Python and CLI,» 2024. [En ligne]. Available: https://tqdm.github.io/. [Accès le 12 05 2025].

[88] P. S. Foundation, «shutil — High-level file operations,» 2024. [En ligne]. Available: https://docs.python.org/3/library/shutil.html. [Accès le 13 05 2025].

[89] O. Team, «About OpenCV,» 2024. [En ligne]. Available: https://opencv.org/about/. [Accès le 17 05 2025].

[90] C. D. Team, «Cityscapes – Semantic Understanding of Urban Street Scenes,» Cityscapes Dataset, 2016. [En ligne]. Available: https://www.cityscapes-dataset.com/. [Accès le 06 04 2025].

[91] C. &. K. T. M. Shorten, «A survey on Image Data Augmentation for Deep Learning,» *Journal of Big Data*, vol. 6, p. 60, 2019.

[92] V. d. Montréal, «Images annotées pour la segmentation sémantique – Caméras de circulation,» 2023. [En ligne]. Available: https://donnees.montreal.ca/dataset/images-annotees-cameras-circulation. [Accès le 02 05 2025].

[93] S. Neuhold, T. Ollmann, M. R. Bulò et P. Kontschieder, «The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes,» chez *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, 2017.

[94] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu et T. Darrell, «BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning,» chez *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, 2020.

[95] G. Ros, L. Sellart, J. Materzynska, D. Vazquez et A. M. Lopez, «The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes,» Las Vegas, NV, USA, 2016.

[96] Resotel, «Langage de programmation Python,» [En ligne]. Available: https://resotel.net.ma/recherche/afficher/langage-de-programmation-python. [Accès le 20 05 2025].

[97] HWLibre, «Qu’est-ce que le Colaboratoire Google ?,» 14 12 2021. [En ligne]. Available: https://fr.hwlibre.com/colaboratoire-google/. [Accès le 04 05 2025].

[98] DataScientest, «Formation TensorFlow : Où apprendre à manier le framework ?,» 17 01 2021.

[En ligne]. Available: https://datascientest.com/formation-tensorflow. [Accès le 03 05 2025].

[99] K. Team, «About Keras 3,» [En ligne]. Available: https://keras.io/getting_started/about/. [Accès le 01 05 2025].

[100] P. P. I. (PyPI), «NumPy,» 2024. [En ligne]. Available: https://pypi.org/project/numpy/. [Accès le 02 05 2025].

[101] M. Fa'alFard, «7 Popular Image Operations Using Python Pillow,» 19 02 2024. [En ligne]. Available: https://www.linkedin.com/pulse/7-popular-image-operations-using-python-pillow-mohammad-fa-alfard-apzff. [Accès le 01 05 2025].

[102] P. P. I. (PyPI), «Matplotlib,» 2024. [En ligne]. Available: https://pypi.org/project/matplotlib/. [Accès le 14 05 2025].

[103] P. P. I. (PyPI), «gradio,» 20025. [En ligne]. Available: https://pypi.org/project/gradio/. [Accès le 05 06 2025].

[104] T. D. Scientist, «Scikit-Learn 101: Exploring Important Functions,» 2023. [En ligne]. Available: https://thedatascientist.com/scikit-learn-101-exploring-important-functions/. [Accès le 16 04 2025].

[105] P. P. I. (PyPI), «Seaborn,» 24 01 2024. [En ligne]. Available: https://pypi.org/project/seaborn/. [Accès le 12 04 2025].