الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

# Mémoire de Master

Filière : Électronique
Spécialité : Systèmes Embarqués

Présenté par

**Hamel Nadjet**

&

**Ziouane Ferial**

# Détection de codes à barre de type 1D par l'apprentissage profond, Application au décodage de code à barres

Dirigé par : Pr. NAMANE ABDERRAHMANE

Année Universitaire 2022-2023

# Remerciements

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ.

*Tout d'abord, nous exprimons notre gratitude la plus profonde à Allah, le Tout-Puissant, pour nous avoir accordé la force, la guidance et la sagesse tout au long de notre parcours de recherche et de la rédaction de notre thèse.*

*Nous tenons à exprimer notre profonde reconnaissance envers notre promoteur, le Pr. Abderrahmane Namane, pour son précieux encadrement, son soutien et son expertise approfondie ont joué un rôle essentiel dans l'orientation et la qualité de cette recherche. Nous sommes sincèrement reconnaissants de sa direction, qui a été déterminante dans notre croissance intellectuelle et la réussite de cette thèse.*

*Nous sommes redevables envers nos familles bien-aimées pour leur amour inconditionnel, leur encouragement et leur compréhension. Leur soutien constant, leur motivation et leurs sacrifices ont été les pierres angulaires de nos études. À nos parents, qui sont nos piliers de force, nous leur sommes éternellement reconnaissants pour leur confiance en nous et leurs prières incessantes.*

*Nous souhaitons exprimer notre sincère appréciation aux membres éminents du jury pour leur précieux temps, leur évaluation méticuleuse et leurs commentaires constructifs. Leur expertise et leurs suggestions avisées ont grandement enrichi ce travail et contribué à sa qualité globale. Nous sommes honorés d'avoir eu l'opportunité de présenter nos recherches devant des personnalités aussi distinguées.*

**ملخص:** الهدف من هذا المشروع هو تطوير نظام يجمع بين التعلم العميق وتقنيات معالجة الصور للكشف عن الرموز الشريطية من النوع D1 وتوطينها وفك تشفيرها. استخدمنا نموذج YOLO V5 ، المعروف على نطاق واسع لدقته وقدرته على اكتشاف الأشياء في الوقت الفعلي. طبقنا تقنيات معالجة الصور المتقدمة لتحديد موقع الرموز الشريطية بدقة في صورة ما وفك تشفيرها لاستخراج المعلومات. لقد طورنا أيضًا واجهة رسومية سهلة الاستخدام لتسهيل استخدام نظامنا.

**كلمات المفاتيح:** التعلم العميق، معالجة الصور 5V YOLO ,الرموز الشريطية.

**Résumé :** L'objectif de ce projet est de réaliser un système qui combine l'apprentissage profond et les techniques de traitement d'image pour la détection, la localisation et le décodage des codes-barres de type 1D. Nous avons utilisé le modèle YOLO V5, largement reconnu pour sa précision et sa capacité à détecter les objets en temps réel. Parallèlement, nous avons appliqué des techniques de traitement d'image avancées afin de localiser précisément les codes-barres dans une image et les déchiffrer pour extraire les informations encodées. Nous avons également développé une interface graphique conviviale pour faciliter l'utilisation de notre système.

**Mots clés :** YOLO V5, apprentissage profond, traitement d’image, code-barres.

**Abstract :** The objective of This Project is to realize a system that combines deep learning and image processing techniques for the detection, localization and decoding of 1D type barcodes. We used the YOLO V5 model, widely recognized for its accuracy and ability to detect objects in real time. At the same time, we applied advanced image processing techniques to precisely locate barcodes in an image and decipher them to extract the encoded information. We have also developed a user-friendly graphical interface to facilitate the use of our system.

**Keywords :** YOLO V5, Deep Learning, image processing , barcode.

# Listes des acronymes et abréviations

**1D :** une dimension

**ANN:** Artificial Neural Networks

**ASCII:** American Standard Code for Information Interchange

**CAB:** Code à Barres

**CNN:** Convolutional Neural Network

**CSP:** Cross-Stage Partial network

**CV:** Computer Vision

**DETR:** DEtection Transformer

**EAN:** European Article Number

**FCL:** Fully Connected Layer

**Fast-RCNN:** Fast Region-based Convolutional Neural Network

**Faster-RCNN:** Faster Region-based Convolutional Neural Network

**GPU:** Graphics Processing Unit

**HOG:** Histogram of Oriented Gradients

**ILVRC:** ImageNet Large Scale Visual Recognition Challenge

**IoU:** Intersection over Union

**ISBN:** International Standard Book Number

**PANet:** Path Aggregation Network

**QR:** Quick Response

**ReLU**: Rectified Linear Unit

**RNA:** Réseaux Neuronaux Artificiels

**R-CNN:** Region-based Convolutional Neural Network

**RVB:** Rouge Vert Bleu

**SSD:** Single Shot Detector

**USPS:** United States Postal Service

**UPC:** Universal Product Code

**VS Code:** Visual Studio Code

**YOLO:** You Only Look Once

**YOLOv5:** You Only Look Once version 5

# Table des matières

# Liste des figures

# Liste des tableaux

# Introduction générale

L'émergence de l'intelligence artificielle et son importance croissante en tant que technologie transformatrice dans de multiples industries ont mené à de grands progrès dans le domaine de la vision par ordinateur. En particulier, la reconnaissance d'objets utilisant l'apprentissage profond s'est avérée être une approche puissante dans le traitement d'image.

Nos recherches se concentrent sur la détection et le décodage des codes-barres 1D mettant à jour les capacités des techniques d'apprentissage profond, en particulier OpenCV (une bibliothèque de vision informatique open source). Nous avons constitué une base de données personnelle regroupant 1000 images, spécialement conçue pour servir de référence dans le cadre de notre projet. Cette base de données nous permet de former notre modèle de détection d'objets personnalisé de manière optimale.

En exploitant le potentiel de YOLOv5 (You Only Look Once), un algorithme de détection d'objets, notre objectif est d'identifier et de localiser avec précision les codes-barres 1D. De plus, nous prévoyons d'ajouter une étape de décodage en utilisant OpenCV, permettant ainsi d'obtenir les informations correspondantes aux codes-barres détectés.

Pour démontrer la praticité et l'utilité de notre projet, nous avons développé une interface web conviviale à l'aide de Tkinter. Cette interface sert de plate-forme pour présenter la mise en œuvre de notre système de détection de codes à barres, permettant aux utilisateurs de télécharger des images et de recevoir des résultats de détection en temps réel, et de visualiser facilement le code à barres détecté et ses informations correspondantes.

Cette approche inclusive d'identification et d'interprétation des codes à barres a des applications réelles dans des domaines comme la vente au détail, la logistique et la gestion des stocks, où le balayage rapide et précis des données de codes à barres est crucial.

Au cours de cette thèse, nous procéderons à une évaluation de la performance de notre modèle, nous évaluerons la précision de la détection ainsi que l'exactitude du décodage, et discuterons des difficultés et des possibilités futures pour améliorer la détection et le décodage des codes-barres à l'aide du deep learning et OpenCV. Notre recherche s'ajoute au domaine en expansion de la détection des objets et montre comment les techniques d'apprentissage profond peuvent résoudre efficacement les problèmes du monde réel, y compris la détection et le décodage des codes à barres de type 1D.

Le mémoire est organisé en trois chapitres suivis d'une conclusion générale :

**Chapitre 1** : Ce chapitre fournit un aperçu des codes-barres, y compris leurs types, leur objectif et leurs applications courantes. Il vise à donner une compréhension générale des codes-barres et de leur importance dans diverses industries.

**Chapitre 2 :** Le chapitre 2 aborde les bases théoriques de la recherche. Il discute des techniques de traitement d'image, et propose une brève introduction à l'apprentissage profond, y compris un aperçu des réseaux neuronaux et des principales architectures. Une attention particulière est accordée à l'architecture, aux types et au fonctionnement du modèle YOLO (You Only Look Once), utilisé dans le cadre du projet.

**Chapitre 3 :** Le troisième chapitre se concentre sur la mise en œuvre du projet. Il explique comment les techniques de traitement d'image et les méthodologies d'apprentissage profond ont été intégrées, en mettant l'accent sur le modèle YOLO. Le chapitre présente les résultats du projet, en mettant en valeur les performances du système et les résultats obtenus.

# Chapitre 1 Généralités sur la détection de code-barres

## 1.1 Introduction

Dans de nombreuses entreprises, la détection des codes à barres est essentielle pour automatiser l'identification et le suivi des biens et des documents. Des systèmes de détection des codes à barres précis et efficaces sont maintenant nécessaires en raison de la dépendance croissante à la technologie des codes à barres pour la gestion des stocks, la logistique, la vente au détail et d'autres fins [1]. Ce chapitre offre une introduction approfondie à la détection des codes à barres en examinant ces objectifs, stratégies, difficultés et utilisations.

## 1.2 Définition du code-barres

Un code-barres, ou code à barres (CAB), est la représentation d'une donnée numérique ou alphanumérique sous forme d'un symbole constitué de barres et d'espaces dont l'épaisseur varie en fonction de la symbologie utilisée et des données ainsi codées. Il existe des milliers de codes-barres différents ; ceux-ci sont destinés à une lecture automatisée par un capteur électronique, le lecteur de code-barres. Pour l'impression des codes-barres, les technologies les plus utilisées sont l'impression laser et le transfert thermique.[2]


**Figure 1.1:** Composition des codes à barres de type EAN13 Composition des codes à barres de type EAN13

Le code-barres est un standard de l'industrie économique et commerciale actuellement. Cela vient du nombre d'avantages que ce type d'identification offre :

- Un gain de temps en caisse pour le consommateur
- Une meilleure gestion des flux produits et des stocks
- Amélioration de la précision des informations sur les produits
- La traçabilité des produits
- Lutte contre la fraude
- Cartes fidélités et études statistiques

Bien que le code-barres soit un système très efficace, il présente néanmoins certains inconvénients, tout comme toute autre méthode de suivi des stocks.

Tout d'abord, les codes-barres ne peuvent être lus que par des lecteurs spécifiques conçus à cet effet, ce qui peut représenter un investissement financier important pour les petits commerçants.

De plus, la capacité de stockage d'informations reste limitée malgré les avantages du code-barres.

## 1.3 Origines du code-barres

En 1952, Joseph Woodland et Bernard Silver déposent le premier brevet pour un code-barres, cherchant à automatiser l'enregistrement des produits. Ils combinent le système de sonorisation de films avec le code morse pour créer des barres noires et des espaces blancs qui peuvent être scannés à l'aide d'une lumière. En 1966, le premier code-barres est utilisé commercialement.[3]

Dans les années 1970, George Laurer invente le code UPC (Universal Product Code), qui ajoute des chiffres sous les barres verticales pour faciliter l'identification des produits. Les codes-barres sont d'abord utilisés pour étiqueter les wagons de train, puis dans les supermarchés pour automatiser les caisses. Le premier produit scanné avec un code-barres en caisse est un paquet de chewing-gum en 1974, dans l'Ohio.

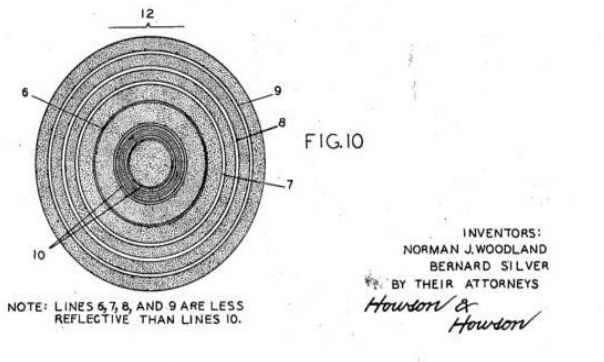



**Figure 1.2 :** Le premier code-barres

## 1.4 Types de code-barres

Les codes-barres sont des outils essentiels pour l'identification et le suivi des produits[4]. Il existe différents types de codes-barres, voir figure(1.3).

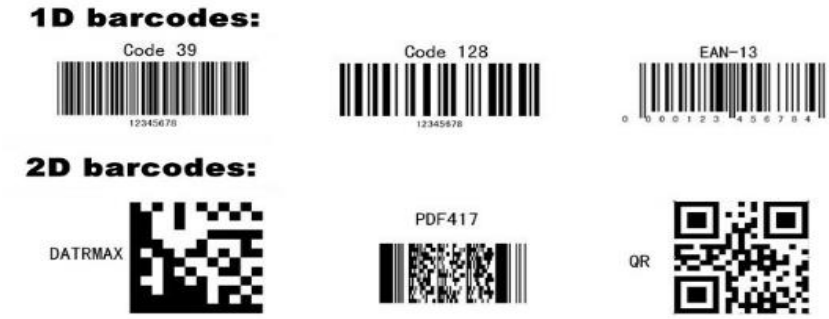



**Figure 1.3 :** Diffèrent type de code-barres

Chacun de ces types ayant ses propres caractéristiques et applications spécifiques, Voici les plus utilisés :

### 1.4.1 Code-barres 39

Il s'agit de l'un des codes-barres les plus anciens et c'est une symbologie courante dans les domaines de l'électronique, des soins de santé et du gouvernement. Il s'agit d'un code alphanumérique linéaire à une dimension (1D) avec la capacité d'inclure l'ensemble des 128 caractères ASCII et de s'étendre à n'importe quelle longueur, limitée uniquement par la taille de l'étiquette. Si l'espace est un problème, le code-barres 128 serait un meilleur choix à considérer.

### 1.4.2 Code-barres 128

Dérivé de l'ensemble de caractères ASCII 128 (0-9, a-z, A-Z et certains caractères spéciaux), ce code-barres compact est largement utilisé dans les applications d'emballage et d'expédition à travers le monde. Le code 128 dispose d'un réglage de commutation automatique qui permet aux utilisateurs de l'optimiser en fonction de la longueur du code-barres.

### 1.4.3 Codes universels de produits (UPC)

Présents sur presque tous les produits de vente au détail, ces codes-barres ont été initialement créés pour les épiceries afin de faciliter l'impression rapide des reçus et le suivi des stocks. Après avoir obtenu un numéro UPC, un fabricant recevra un numéro d'entreprise unique à combiner avec ses numéros de produit individuels.

### 1.4.4 Numéro international d'article (EAN)

Considérés comme une extension des codes UPC, ces codes-barres sont spécifiquement utilisés par les librairies, les bibliothèques, les universités et les grossistes pour la traçabilité des livres. Ces codes à 13 chiffres sont créés à partir des numéros internationaux normalisés du livre (ISBN) correspondant à chaque livre suivi. Tout comme les UPC, ils sont standardisés pour l'identification unique des éditeurs.[3]

### 1.4.5 PDF417

Ce code-barres linéaire 2D empilé peut être trouvé dans de nombreux types d'identification tels que votre permis de conduire. Il est également le standard choisi par l'USPS et le Département de la Sécurité intérieure en raison de ses capacités avancées, telles que l'encodage de liens vers plusieurs fichiers de données. Cependant, il peut être assez grand en taille 4 fois plus grand que d'autres codes-barres 2D tels que le Datamatrix et les QR Codes.

### 1.4.6 Datamatrix Code (Code Data Matrix)

Il est devenu l'un des codes-barres 2D les plus courants. Il s'agit d'un code carré capable d'encoder de grandes quantités d'informations - littéralement énormes - dans un espace très réduit ; c'est pourquoi il est très populaire dans l'industrie électronique et les soins de santé. Les codes 2D nécessitent des scanners sophistiqués, tels que les smartphones, pour "prendre une photo" et traduire l'image entière en une seule fois. Lorsque les entreprises ont besoin d'une plus grande capacité de stockage de codes-barres, les codes-barres 2D surpassent leurs homologues 1D. [3]

### 1.4.7 QR Code (Codes QR)

La dernière tendance en matière de codes-barres, les codes QR, gagnent en popularité en tant qu'outils de marketing pour lier à des informations en ligne. Moins compacts que le Data Matrix, on les trouve souvent sur des supports publicitaires et devantures de magasins, renvoyant à des promotions spéciales ou à des détails sur un produit particulier.[3]

## 1.5 Principes de Fonctionnement de la Détection des Codes-Barres

La détection d'un code-barres repose sur l'utilisation d'un dispositif de capture d'image, comme une caméra ou un scanner, et d'un logiciel de traitement d'image. L'image contenant le code-barres est capturé, puis prétraitée pour améliorer sa qualité. Ensuite, le logiciel analyse l'image à la recherche de motifs ressemblant à un code-barres, en identifiant les zones claires et sombres caractéristiques. Une fois le code-barres détecté, le logiciel extrait le motif du code-barres de l'image et le décode pour

obtenir les informations encodées. Les données extraites du code-barres peuvent ensuite être utilisées pour diverses applications, telles que l'identification des produits ou la gestion des stocks. [5]

**Figure1. 3:** Scanner un code à barres à l'aide d'un scanner

Les caméras utilisées pour la lecture des codes-barres fonctionnent de manière similaire aux lecteurs de codes-barres traditionnels, mais offrent l'avantage supplémentaire d'une évaluation plus précise de la qualité des codes-barres, allant au-delà d'une simple lecture binaire (lisible ou illisible).


**Figure 1.4:** Scanner un code à barres à l'aide d'une caméra

## 1.6 Études adoptées dans la détection des codes-barres

La détection d'objets est une tâche fondamentale en vision par ordinateur qui consiste à identifier et localiser des objets d'intérêt dans des images ou des séquences vidéo. Ce domaine n'est pas aussi récent qu'il n'y paraît. En fait, la détection d'objets a évolué au cours des 20 dernières années. Les progrès de la détection d'objets sont généralement séparés en deux périodes historiques distinctes (avant et après l'introduction de l'apprentissage profond). [6]

### 1.6.1 Approches initiales :

a **Algorithme de Viola-Jones :** En 2001, Viola et Jones ont introduit un algorithme révolutionnaire de détection d'objets utilisant des caractéristiques de type Haar et des classifieurs en cascade. Leur méthode était efficace et capable de détecter en temps réel les visages, ouvrant la voie à des avancées ultérieures dans la détection d'objets.

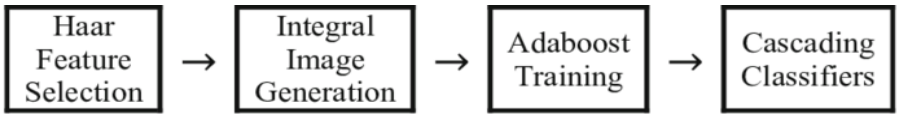



**Figure 1.5 :** Viola et Jones algorithme

b **Histogramme des gradients orientés (HOG)** *:* Introduit par Dalal et Triggs en 2005, HOG a révolutionné la détection des piétons. Il calculait les gradients de l'image pour capturer l'apparence locale des objets et les informations de forme, améliorant ainsi la précision de la détection.

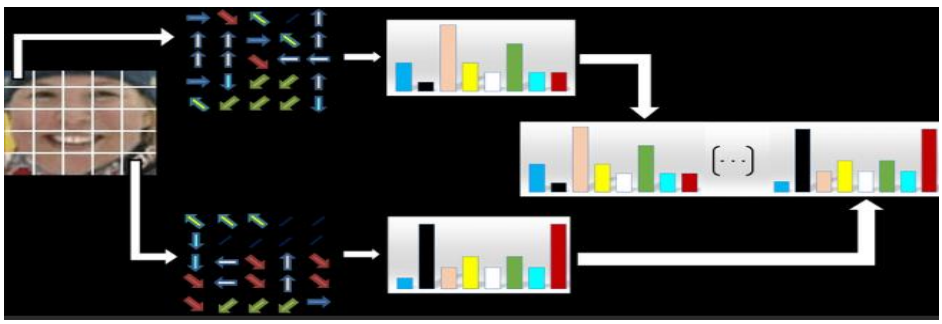


**Figure 1.6:** Calcule des Histogramme des gradients orientés

### 1.6.2 Regions with Convolutional Neural Network et ses variantes

**a** **Regions with CNN (R-CNN)** *:* En 2014, Girshick et al. ont proposé R-CNN, un travail fondamental qui combinait des propositions de régions avec des réseaux de neurones convolutionnels (CNN). Il a obtenu des résultats impressionnants en matière de détection d'objets en utilisant des caractéristiques CNN dans un pipeline multi-étapes. [7]

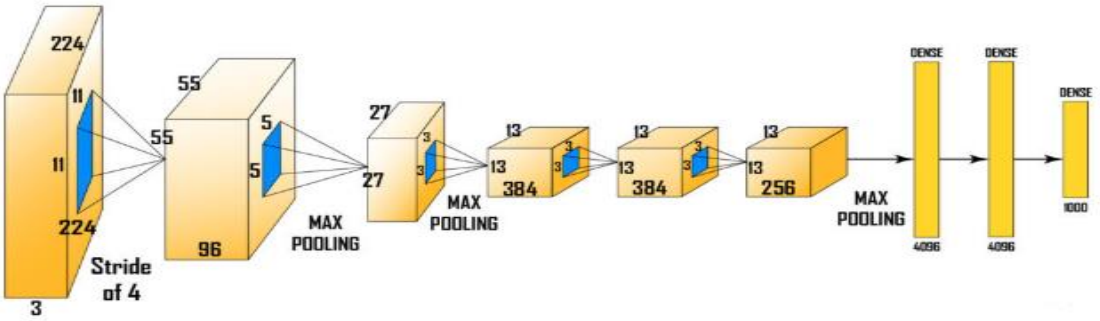



**Figure 1.7 :** Architecture de R-CNN

**b** **Fast R-CNN :** En 2015, Girshick a introduit Fast R-CNN, qui améliorait R-CNN en partageant les calculs entre les propositions de régions, ce qui permettait une vitesse d'inférence plus rapide. [7]

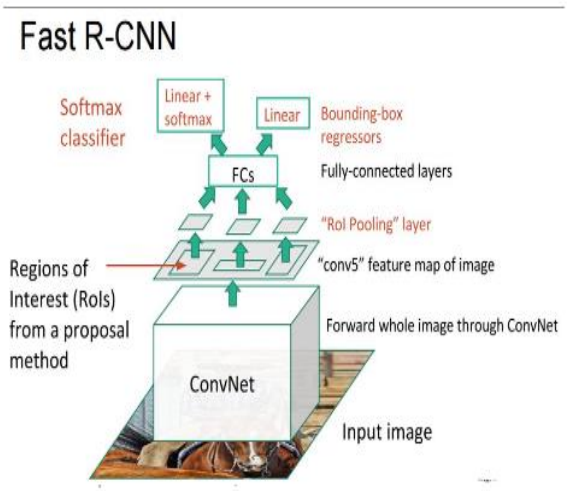



**Figure 1.8 :** Architecture de Fast R-CNN

c **Faster R-CNN :** En 2015, Ren et al. ont proposé Faster R-CNN, qui a introduit le Réseau de Proposition de Régions (RPN). RPN a permis l'apprentissage de bout en bout en générant directement des propositions de régions à partir de caractéristiques de convolution, éliminant ainsi le besoin de méthodes de proposition externes.[7]

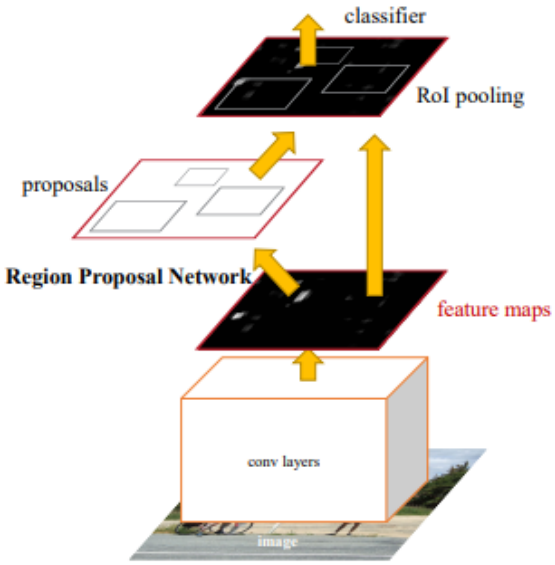



**Figure 1.9 :** Architecture de Faster R-CNN

### 1.6.3 Détecteurs en un seul passage (SSD)

a **Single Shot Detector (SSD) :** Liu et al. ont présenté SSD en 2016, une approche novatrice qui combinait des cartes de caractéristiques de haut et de bas niveau pour une détection d'objets efficace. SSD a atteint une détection en temps réel en prédisant directement les classes d'objets et les décalages de boîtes englobantes à partir de plusieurs cartes de caractéristiques.

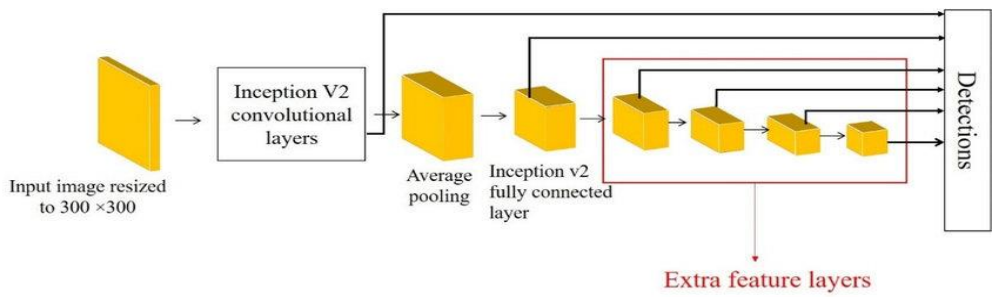



**Figure 1.10 :** Architecture du Single Shot Detector

b **YOLO (You Only Look Once) :** Redmon J et al. ont introduit YOLO en 2016, révolutionnant la détection d'objets en proposant un cadre de détection unifié. YOLO a atteint une vitesse impressionnante en effectuant la classification d'objets et la régression des boîtes englobantes (bounding boxes) en une seule passe à travers le réseau.[8]

### 1.6.4 Réseaux de pyramides de caractéristiques (FPN) :

a **Feature Pyramid Network (FPN) :** Lin et al. ont proposé FPN en 2017 pour relever le défi de la détection d'objets à différentes échelles. FPN a créé une architecture descendante qui fusionnait des cartes de caractéristiques à différentes résolutions, permettant une détection précise des objets à différentes échelles.[9]

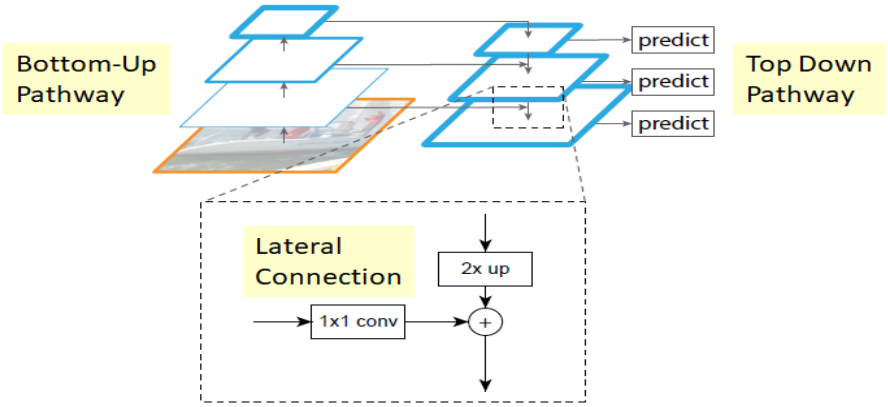



**Figure 1.11 :** Réseau de pyramides de caractéristiques (FPN)

### 1.6.5 Détecteurs efficaces

a **EfficientDet** : En 2019, Tan et al. ont introduit EfficientDet, qui optimisait les modèles de détection d'objets en termes de précision et d'efficacité. En tirant parti d'une mise à l'échelle composée et d'architectures de réseau efficaces, EfficientDet a atteint des performances de pointe avec beaucoup moins de paramètres.

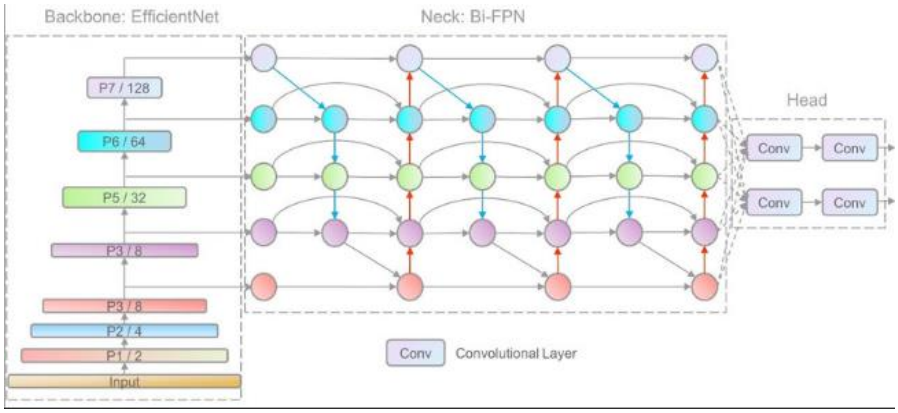



**Figure 1.12 :** Architecture du réseau EfficientDet

### 1.6.6 Détecteurs basés sur les transformateurs

**a** **DETR** : Carion N et al. ont présenté DETR en 2020, introduisant une architecture basée sur les transformateurs pour la détection d'objets. Contrairement à la majorité des méthodes de détection existantes, DETR utilisait un cadre de prédiction basé sur un ensemble et ne nécessitait pas de couches personnalisées. Ainsi, elle peut être reproduite facilement dans n'importe quel cadre contenant la norme CNN et les classes de transformateurs. DETR obtenait des résultats compétitifs tout en éliminant le besoin de boîtes d'ancrage conçues à la main.

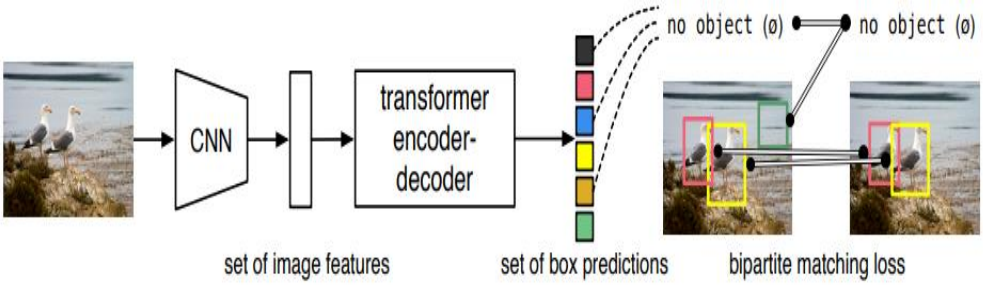



**Figure 1.13 :** DETR prédictions

## 1.7 Présentation à la base de données ArteLab

Dans cette section, nous présentons la base de données ArteLab, une référence que nous avons utilisée pour comparer nos propres résultats. Avant de plonger dans les détails de cette base de données, nous tenons à souligner l'importance de cette étude comparative pour notre recherche.

### 1.7.1 Base de données Artelab

La base de données ArteLab est un ensemble de données couramment utilisé dans le domaine de la vision par ordinateur pour évaluer les algorithmes de détection et de décodage de codes-barres.

Il a été introduit dans plusieurs articles. Cette base est composés d'images contenant différents types de codes-barres, notamment des codes-barres 1D et 2D, capturés dans des conditions variables telles que l'éclairage, la rotation et l'échelle. Le jeu de données fournit des annotations de vérité terrain (ground truth), notamment la position, la longueur et l'angle du code-barres, qui peuvent être utilisées pour entraîner et évaluer les algorithmes de détection de codes-barres. Les chercheurs utilisent souvent le jeu de données ArteLab comme référence pour tester les performances des systèmes de détection et de décodage de codes-barres.

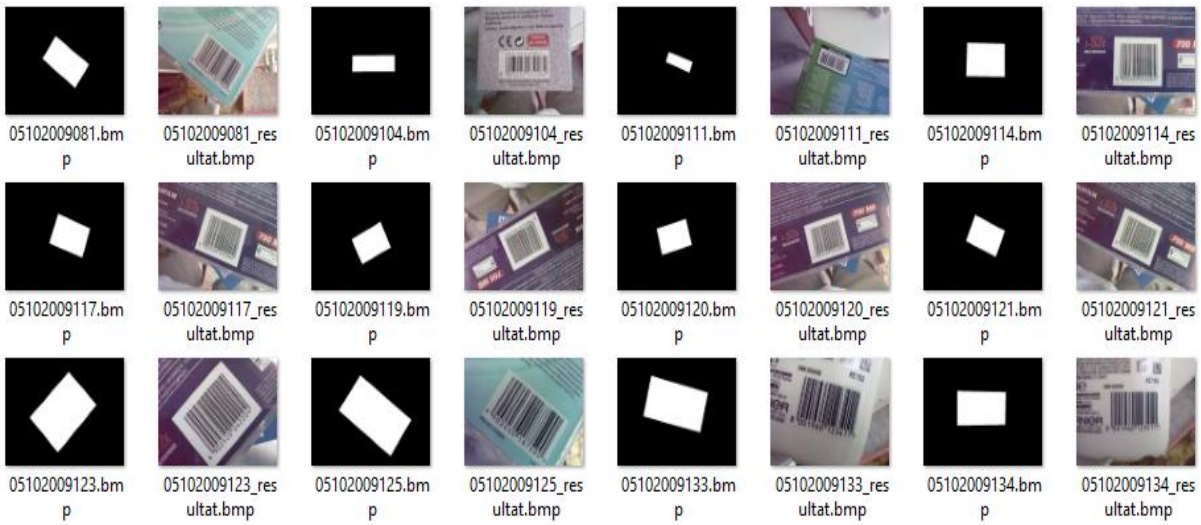



**Figure 1.14 :** Quelques images de la base de données ArteLab

### 1.7.2 Détection de codes-barres en temps réel

Afin de fournir des méthodes fiables et efficaces pour la reconnaissance des codes à barres, ArteLab a effectué des recherches approfondies dans le domaine de la détection des codes-barres. Leur travail repose sur l'utilisation d'algorithmes de vision informatique et de techniques d'apprentissage automatique pour reconnaître et décoder avec précision les codes-barres provenant de sources multiples, tels que les emballages de produits et les étiquettes.

ArteLab a étudié plusieurs symboles de codes à barres au cours de leurs recherches, notamment les codes-barres 1D tels que le Code 39 et les codes 2D comme le code QR et la matrice de données. Ils ont développé des algorithmes sophistiqués pour résoudre des problèmes tels que le bruit d'image, la distorsion de la perspective et les variations de taille et d'orientation des codes-barres.

Leur travail met en évidence l'importance des codes-barres dans de nombreux scénarios et ouvre de nouvelles perspectives pour leur utilisation dans divers domaines. En se concentrant sur la détection en temps réel des codes-barres à partir de flux vidéo, ArteLab contribue à améliorer la précision et la vitesse de reconnaissance par rapport aux techniques passives traditionnelles. Leurs recherches ouvrent également des possibilités d'applications commerciales basées sur un tel système de numérisation passive

C. Creusot and A. Munawar leur travail met en évidence la limitation du véritable potentiel des codes-barres dans de nombreux scénarios. Les auteurs présentent une technique en temps réel pour la détection de codes-barres dans des conditions réelles à partir de flux vidéo. Leur technique se démarque des techniques passives de pointe en termes de précision et de vitesse. De plus, les auteurs discutent des applications commerciales potentielles rendues possibles grâce à un tel système de numérisation passive. Cette recherche ouvre de nouvelles perspectives pour l'utilisation des codes-barres dans divers domaines[10].

**Figure 1.15 :** Exemples de résultats de détection sur l'ensemble de données ArTe-Lab

### 1.7.3 Détection et classification des codes-barres en temps réel à l'aide du Deep-Learning

Dr Daniel Kold Hansen, Kamal Nasrollahi, Christoffer B. Rasmusen and Thomas B. Moeslund décrivent comment ils ont adapté un détecteur de pointe basé sur l'apprentissage en profondeur, appelé You Only Look Once (YOLO), pour la détection rapide et fiable de codes-barres. Leur détecteur est capable de détecter à la fois des codes-barres 1D et des codes QR. Les résultats obtenus sur l'ensemble de données de référence Muenster BarcodeDB sont exceptionnels, avec un taux de détection de 0,991. Leur système développé est également capable de déterminer la rotation des codes-barres 1D et QR, ce qui permet d'ajuster la détection en conséquence, ce qui est bénéfique pour le processus de décodage. Les performances de détection et de prédiction de rotation démontrent une exécution en temps réel efficace du système[11].

## 1.8 Défis de la détection des codes-barres

La détection des codes-barres présente plusieurs défis auxquels les systèmes doivent faire face. [12]

Voici quelques-uns des problèmes courants :

a **Mauvaise qualité des images :** Les codes-barres peuvent être mal imprimés, flous ou endommagés, ce qui rend leur détection difficile.

**Figure 1.16 :** Code-barres mal imprimé

b **Occultation** : Les codes-barres peuvent être partiellement ou totalement cachés par des objets, des étiquettes superposées ou des saletés, ce qui empêche leur lecture.


**Figure 1.17 :** Code-barres encrassé

c **Distorsion de perspective :** Lorsque le code-barres est capturé à un angle oblique ou avec une déformation de perspective, il peut devenir déformé, ce qui rend difficile son interprétation.


**Figure 1.18 :** Distorsion de perspective d'un code-barres

d **Variations de symbologie :** Il existe différentes symbologies de codes-barres avec des structures et des arrangements spécifiques. Les systèmes doivent être capables de reconnaître et de lire différentes symbologies pour assurer une détection précise.

Pour surmonter ces défis, des stratégies et des techniques sont utilisées, telles que :

a **Amélioration d'images :** Des algorithmes de traitement d'image améliorent la qualité des images des codes-barres en réduisant le bruit, en améliorant le contraste et en renforçant les contours.

b **Algorithmes adaptatifs :** Pour maximiser les chances de détection et de décodage réussis, ces algorithmes s'adaptent aux variations de qualité d'image, de symbologie et de positionnement.

c **Algorithmes de décodage robustes :** Des algorithmes de décodage avancés, qui prennent en compte les erreurs de lecture et les variations potentielles, sont utilisés pour extraire les informations des codes-barres, même dans des conditions difficiles.

## 1.9 Conclusion

Le chapitre précédent a introduit la détection des codes-barres en présentant la définition des codes-barres et leurs différents types. Diverses approches utilisées dans les études sur la détection des codes-barres ont été discutées, telles que les méthodes basées sur les régions, les réseaux neuronaux convolutionnels et les techniques avancées comme SSD, FPN et les détecteurs basés sur les transformateurs. La base de données ArteLab pour la recherche sur la détection des codes-barres a été présentée, ainsi que les défis rencontrés dans cette tâche. Le prochain chapitre se concentrera sur les techniques de traitement d'image et d'apprentissage profond pour la détection d'objets, fournissant des bases théoriques pour la détection des codes-barres à l'aide de ces méthodes.

# Chapitre 2 Approches de pointe dans la détection de codes-barres

## 2.1 Introduction

Ce chapitre donne un aperçu approfondi des recherches et des avancées les plus récentes dans le domaine de la détection et de l'identification des codes-barres, en mettant l'accent sur les techniques d'apprentissage profond. Il explore les méthodes de pointe actuellement utilisées pour relever les défis de la détection des codes-barres et présente le modèle YOLOv5 comme un exemple marquant de ces approches.

## 2.2 Traitement d'images à l'aide de la vision par ordinateur

Cette section se concentre sur les techniques de traitement d'images utilisées dans notre étude. Nous aborderons les nombreuses méthodes et approches utilisées pour manipuler et analyser les images, ce qui nous permettra d'appréhender pleinement la méthodologie du traitement d'image utilisée dans le cadre de ce projet. Ces techniques visent à améliorer l'alignement, à extraire les caractéristiques des codes-barres, à extraire la boîte englobante (bounding box) et à évaluer la précision de la détection des codes-barres. Pour poser les bases de notre discussion sur les techniques de traitement d'images, nous devons d'abord définir ce qu'est une image numérique. Une image numérique est une cadre composée de pixels. Chaque pixel représente une petite partie de l'image et contient des informations sur sa couleur et sa luminosité. Chaque pixel est composé de trois couleurs : le rouge, le vert et le bleu. En les combinant, on obtient différentes couleurs. Une image complète est formée par la combinaison de plusieurs pixels. Ces pixels sont disposés en grille, formant une représentation bidimensionnelle de l'image. Les images numériques (voir figure 2.1 ) peuvent être stockées, manipulées et affichées à l'aide de systèmes informatiques.[13]

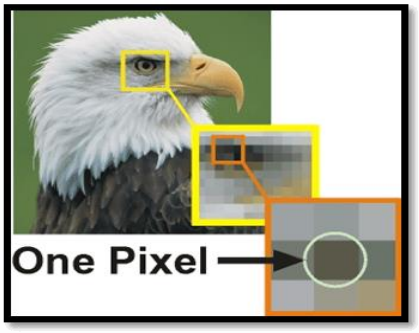



**Figure 2.1 :** Démonstration d'une image numérique

Dans la section suivante nous présentons les techniques spécifiques utilisées dans ce projet pour détecter et reconnaître les codes-barres.

### 2.2.1 Manipulation d'images

Ces méthodes sont utilisées pour améliorer l'alignement et la taille de l'image du code-barres avant son traitement ultérieur : [14]

a **Rotation de l'image :** L'image du code-barres est tournée de cette manière pour améliorer l'alignement et l'orientation. L'alignement horizontal ou vertical des barres du code-barres améliore la précision des procédures de traitement ultérieures.

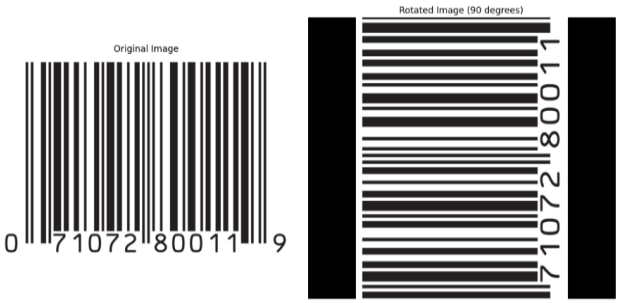



**Figure 2 .2 :** Rotation des images à l'aide d'OpenCV

b **Redimensionnement de l'image :** Le redimensionnement d'une image consiste à modifier ses dimensions tout en conservant le rapport hauteur/largeur. Il peut aider à normaliser la taille des images de codes-barres.

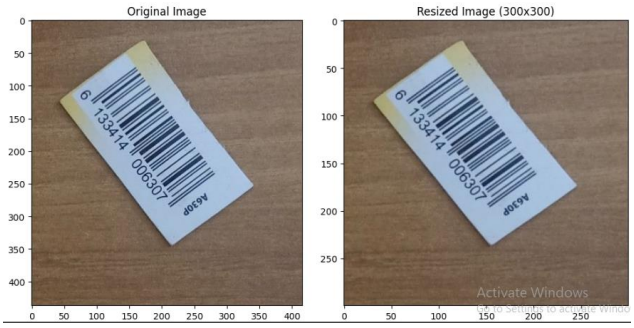



**Figure 2.3 :** Redimensionnement d'une image à l'aide d'OpenCV

c **Recadrage de l'image :** Cette technique consiste à découper de l'image originale une région d'intérêt spécifique contenant le code-barres. Cette étape se concentre sur le traitement de la zone du code-barres, en supprimant toutes les données d'arrière-plan inutiles.


**Figure 2.4 :** Recadrage d'une image à l'aide d'OpenCV

### 2.2.2 Détection des Bords

a **Flou gaussien :** Le flou gaussien est une technique de traitement d'image largement utilisée qui consiste à appliquer une distribution gaussienne aux valeurs des pixels d'une image. Cette technique permet de lisser ou de flouter efficacement l'image, en réduisant la présence de bruit à haute fréquence et de détails fins, tout en conservant la structure générale et les contours. Cette technique est une forme de filtrage passe-bas, souvent utilisée pour améliorer la qualité de l'image, réduire les artefacts de l'image ou préparer les images pour d'autres tâches d'analyse ou de traitement. OpenCV dispose d'une fonction intégrée permettant d'effectuer facilement un flou gaussien/un lissage sur les images. Tout ce que vous avez à spécifier est la taille du noyau gaussien avec lequel votre image doit être convoluée.

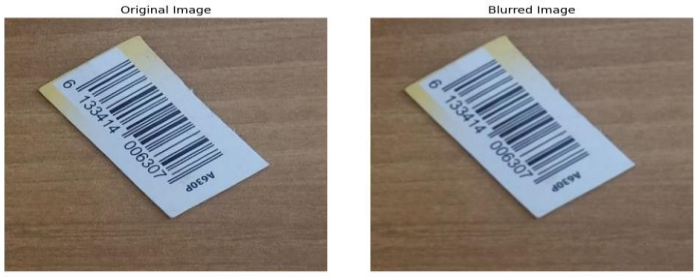



**Figure 2.5 :** Application d'un flou gaussien en utilisant OpenCV

**b** **L'opérateur Sobel :** L'opérateur de Sobel est une opération conjointe de lissage gaussien et de différenciation, il est donc plus résistant au bruit. Vous pouvez spécifier la direction des dérivées à prendre, verticale ou horizontale (par les arguments, ordre y et ordre x respectivement). Il calcule le gradient de l'intensité de l'image à chaque pixel, mettant en évidence les régions où les changements d'intensité sont rapides et qui correspondent aux bords.

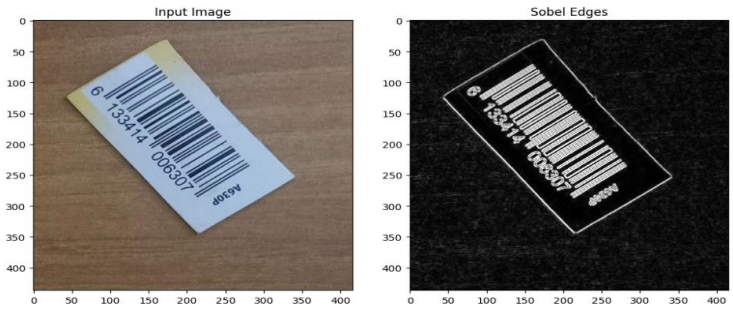



**Figure 2.6 :** Application du filtre de Sobel à l'aide d'OpenCV

### 2.2.3 Conversion de l'espace couleur

Il existe plus de 150 méthodes de conversion de l'espace couleur dans OpenCV. Nous n'en étudierons que deux :

**a** **RGB :** L'espace colorimétrique RVB (RGB) est représenté sous la forme d'un format d'image à trois canaux où chaque pixel possède trois canaux de couleur : Rouge, Vert et Bleu. Les valeurs de couleur pour chaque canal vont généralement de 0 à 255, représentant l'intensité de chaque composante de couleur. [15]

| | Nominal Range | White | Yellow | Cyan | Green | Magenta | Red | Blue | Black |
|---|---|---|---|---|---|---|---|---|---|
| R | 0 to 255 | 255 | 255 | 0 | 0 | 255 | 255 | 0 | 0 |
| G | 0 to 255 | 255 | 255 | 255 | 255 | 0 | 0 | 0 | 0 |
| B | 0 to 255 | 255 | 0 | 255 | 0 | 255 | 0 | 255 | 0 |

**Figure 2.7 :** Code RGB

**b** **Conversion en niveaux de gris :** réduit l'image à une représentation à un seul canal, ce qui simplifie les étapes de traitement ultérieures. Il s'agit de convertir l'image en niveaux de gris, où la valeur de chaque pixel représente son intensité.[16].

Pour passer d'une image couleur (RVB) vers une image à niveaux de gris ou noir et blanc, on doit exploiter une des deux équations du codage YUV ou YIQ qui sont identiques et qui est :

$$G = 0{,}299*R + 0{,}587*V + 0{,}114*B$$

Avec G est le niveau de gris du pixel, et R, V et B sont les composantes rouge, vert et bleue du pixel.

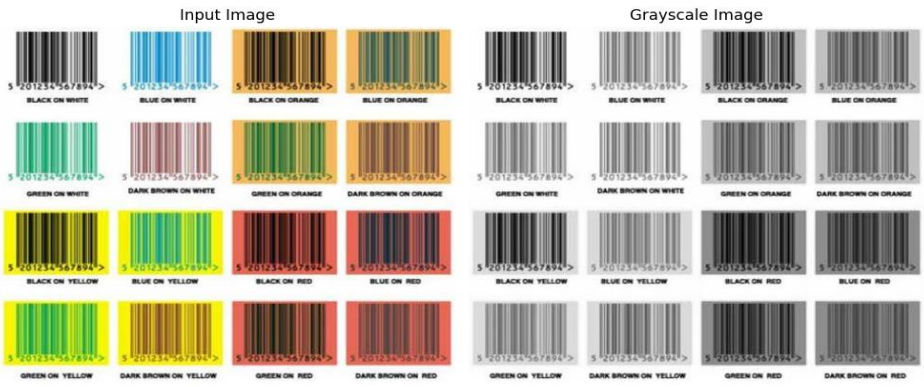


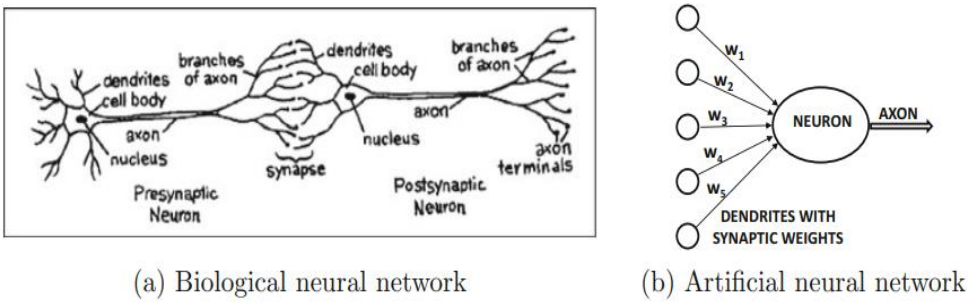

**Figure2.8:** Conversion d'images en niveaux de gris avec OpenCV

## 2.3 Aperçu de l'apprentissage profond :

L'apprentissage profond a commencé à avoir un impact considérable sur la vision par ordinateur en 2012, lorsque le groupe de M. Hinton a remporté le concours de reconnaissance visuelle à grande échelle ImageNet (ILSVRC) grâce à l'apprentissage profond. Aujourd'hui, de nombreuses startups émergent et travaillent sur des applications de vision artificielle avec des technologies d'apprentissage profond. La revue technologique du MIT a classé l'apprentissage profond parmi les dix technologies les plus innovantes en 2013. [17]

### 2.3.1 Réseaux neuronaux artificiels

Les réseaux neuronaux artificiels (RNA) ou ANN (Artificial neural networks) sont des techniques d'apprentissage automatique inspirées du mécanisme d'apprentissage des organismes biologiques. Dans le système nerveux humain, les neurones sont connectés par des synapses. Les RNA simulent ce processus à l'aide d'unités de calcul appelées neurones. Ces neurones sont reliés entre eux par des poids, qui déterminent la force des connexions (voir figure 2.9). [18]




(a) Biological neural network (b) Artificial neural network

**Figure 19 :** Neurone biologique et neurone artificiel

#### a Architecture du réseau neuronal

Chaque entrée d'un neurone est pondérée et le réseau calcule une fonction en propageant les valeurs calculées des neurones d'entrée aux neurones de sortie à l'aide de ces poids. Les poids sont ajustés en fonction des données d'apprentissage, qui consistent en des paires entrée-sortie. Sur la base des représentations d'entrée, le réseau effectue des prédictions et les compare aux

étiquettes de sortie annotées dans les données d'apprentissage. Les erreurs de prédiction entraînent l'ajustement des poids, ce qui permet d'affiner la fonction calculée au fil du temps. [19]

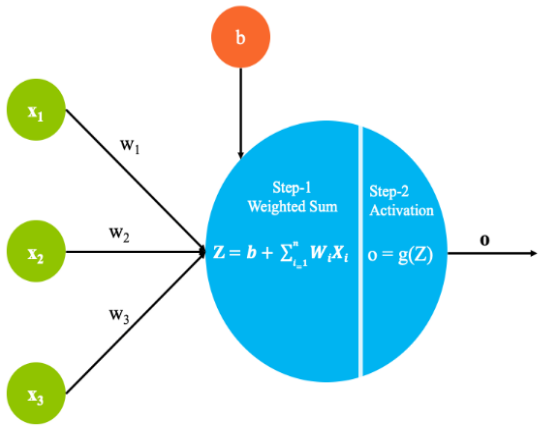



**Figure 2.10 :** Architecture de base du perceptron

- **Entrées x1 à xn :** Il s'agit des valeurs ou des caractéristiques extraites de l'ensemble de données. Chaque entrée de l'ensemble de données comprend n variables dépendantes qui fournissent des informations pertinentes pour le modèle.

- **Pondérations w1 à wn :** Il s'agit des paramètres associés à chaque variable d'entrée. Les poids déterminent l'importance ou la contribution de chaque variable d'entrée dans le calcul du modèle.

- **b :** Il s'agit d'un terme constant ajouté au modèle qui joue le rôle de biais. Il permet un niveau de flexibilité supplémentaire en tenant compte de tout flou/lissage gaussien facilement réalisable sur les variables d'entrée.

### b Structure du réseau neuronal

Les RNA contiennent des neurones artificiels. Il peut y avoir un nombre quelconque de neurones, en fonction des exigences de l'application. Ces neurones sont regroupés en couches. La structure ANN la plus couramment utilisée comprend une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie. Les neurones biologiques communiquent entre eux en envoyant des impulsions

électriques, tandis que les ANN transmettent l'information entre les couches et les nœuds. Au niveau des nœuds, l'importance du signal d'entrée est déterminée par l'association de poids. La valeur des poids peut être positive ou négative. Un neurone est actif si le poids est positif, tandis qu'un neurone devient inactif si le poids est négatif. Un neurone additionne toutes les entrées et multiplie ces entrées par le poids associé au nœud. [20]

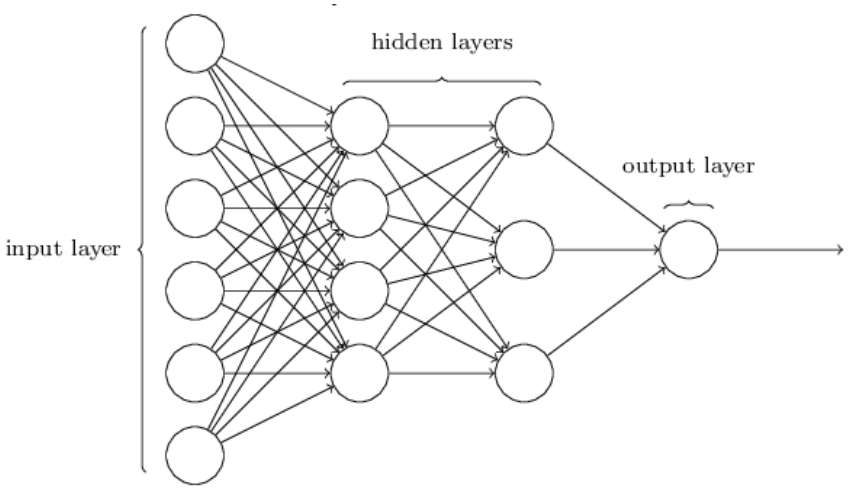



**Figure2.11:** Perceptrons multicouches

### 2.3.2 Réseaux neuronaux convolutifs

Les réseaux neuronaux convolutifs (CNN) sont des réseaux d'inspiration biologique utilisés en vision artificielle pour des tâches telles que la classification d'images et la détection d'objets. Les CNN ont une architecture tridimensionnelle dont l'étendue spatiale et la profondeur correspondent aux caractéristiques et aux canaux de couleur. Ils se composent de couches de convolution et de couches de sous-échantillonnage. Les couches de convolution utilisent des filtres pour cartographier les activations d'une couche à l'autre, capturant ainsi les relations spatiales. Les connexions dans les CNN sont peu nombreuses et les couches de niveau inférieur capturent des formes primitives tandis que les couches de niveau supérieur capturent des formes plus complexes. Les couches de sous-échantillonnage compriment les empreintes spatiales. Les CNN ont connu un grand succès dans les tâches de reconnaissance d'images et de détection d'objets, dépassant même les performances humaines dans certains cas. [21]

### a Architecture des Réseaux Neuronaux Convolutifs :

Le composant clé d'un CNN est la couche convolutive, voir figure 2.12, qui applique un ensemble de filtres ou de noyaux à l'image d'entrée. Chaque filtre détecte différents motifs ou caractéristiques de l'image, tels que les bords, les textures ou les formes. [22]

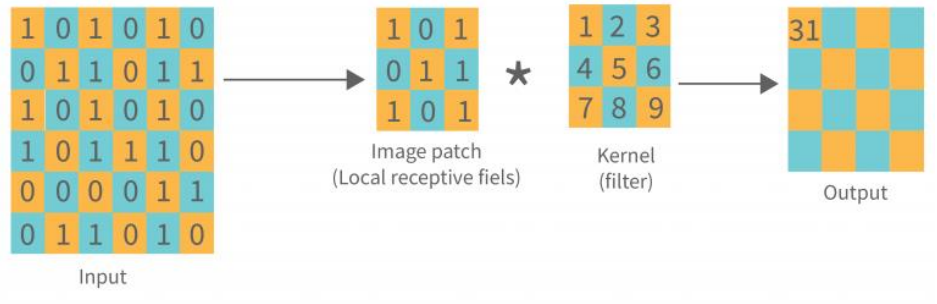



**Figure 2.12 :** Couche de convolution

Trois paramètres permettent de dimensionner le volume de la couche de convolution [23] :

- **Profondeur de la couche :** nombre de noyaux de convolution (nombre de neurones associés à un même champ récepteur).
- **Le pas :** contrôle le chevauchement des champs récepteurs. Plus le pas est petit, plus les champs récepteurs se chevauchent et plus le volume de sortie sera grand.

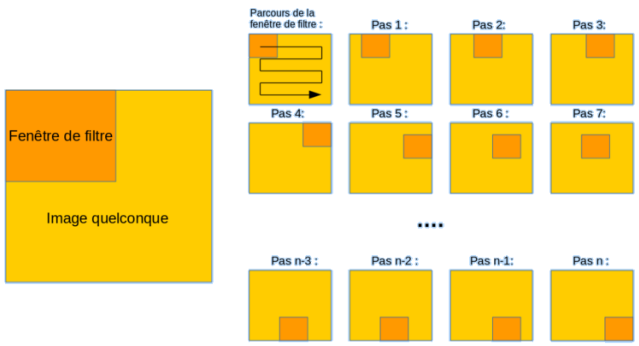



**Figure 2.13 :** Schéma du parcours de la fenêtre de filtre sur l'image

- **La marge (à 0) ou zero padding :** parfois, il est commode de mettre des zéros à la frontière du volume d'entrée. La taille de ce 'zero-padding' est le troisième hyper paramètre. Cette marge permet de contrôler la dimension spatiale du volume de sortie. En particulier, il est parfois souhaitable de conserver la même surface que celle du volume d'entrée.

Après les couches convolutives, les CNN comprennent généralement des couches de mise en commun (Pooling Layer), qui réduisent l'échantillonnage des cartes de caractéristiques en réduisant leurs dimensions spatiales. Cela permet de capturer les informations les plus importantes tout en réduisant la complexité des calculs. [22]

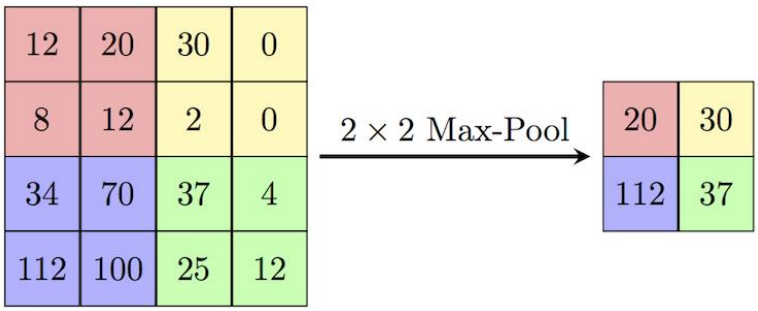



**Figure2.14:** Exemple D'Argumentation maximale

La sortie des couches convolutives et de mise en commun est ensuite aplatie et introduite dans une ou plusieurs couches entièrement connectées (Fully Connected Layer : FCL), voir figure 2.15, où chaque neurone est connecté à tous les neurones de la couche précédente. Les couches entièrement connectées effectuent des tâches de classification ou de régression en apprenant des relations complexes entre les caractéristiques extraites et la sortie cible. Pour faire des prédictions, le CNN utilise une fonction d'activation après chaque couche pour introduire une non-linéarité et augmenter la capacité de modélisation du réseau.

Les poids du réseau sont d'abord aléatoires, puis ajustés au cours du processus d'apprentissage.

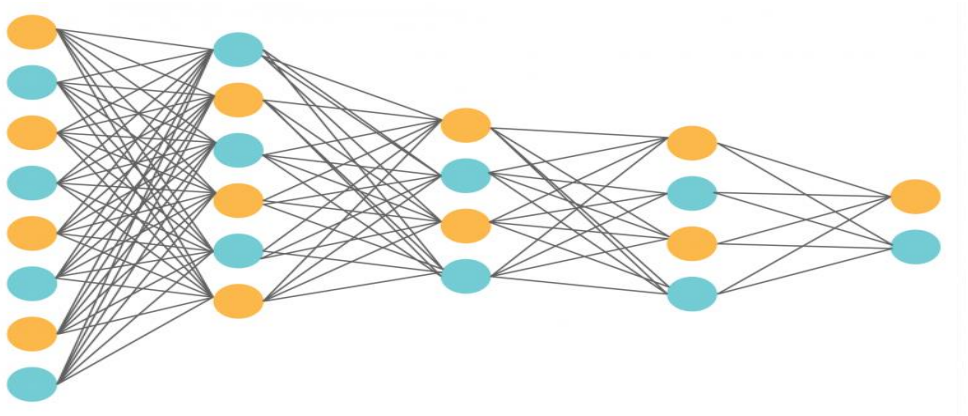


**Figure 2.15 :** la couche FCL

Voici une représentation simplifiée dans la figure 36 qui illustre le déroulement global et l'architecture d'un réseau de neurones convolutifs (CNN).

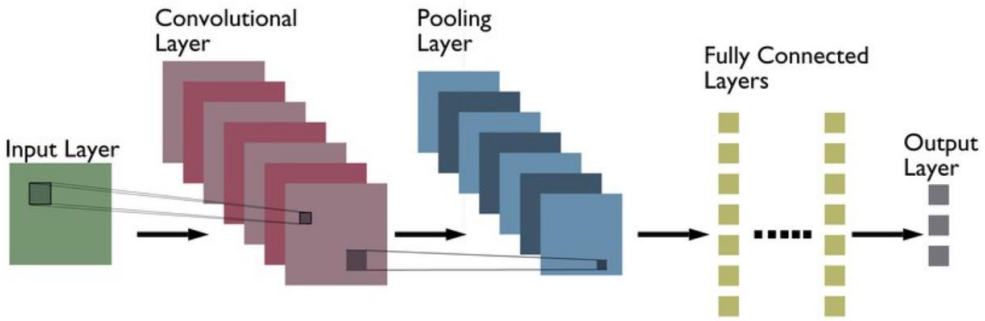



**Figure 2.16 :** Architecture basique du CNN

### 2.3.3 Fonctions d'activation

**a** **Fonction sigmoïde** : elle est couramment utilisée dans la couche de sortie pour les tâches de classification binaire, car elle produit des valeurs comprises entre 0 et 1. Toutefois, elle présente l'inconvénient de ralentir le processus d'apprentissage lorsque les valeurs de sortie sont très grandes ou très petites, en raison de sa pente plate à ces extrêmes. [24]

**b** **Fonction d'activation tanh :** est une amélioration de la fonction sigmoïde car elle offre une meilleure non-linéarité. Elle est comprise entre -1 et 1 et est une version mise à l'échelle de la fonction sigmoïde. Elle peut remplacer la fonction sigmoïde dans la plupart des cas, sauf lors de la classification binaire où les valeurs de sortie sont spécifiquement 0 ou 1.

c **ReLU (Rectified Linear Unit)** : c'est la fonction d'activation la plus utilisée car elle offre de meilleurs gradients, même pour de grandes valeurs de l'entrée. Cela signifie qu'elle ne souffre pas du problème de ralentissement de l'apprentissage rencontré par sigmoïde et tanh.

d **Leaky ReLU** : est une extension de ReLU qui traite le problème des gradients nuls pour les valeurs d'entrée négatives. Elle introduit une petite pente non nulle pour les valeurs négatives, bien qu'en pratique son impact sur les performances ne soit pas significatif. Par conséquent, ReLU reste la fonction d'activation préférée dans de nombreuses applications

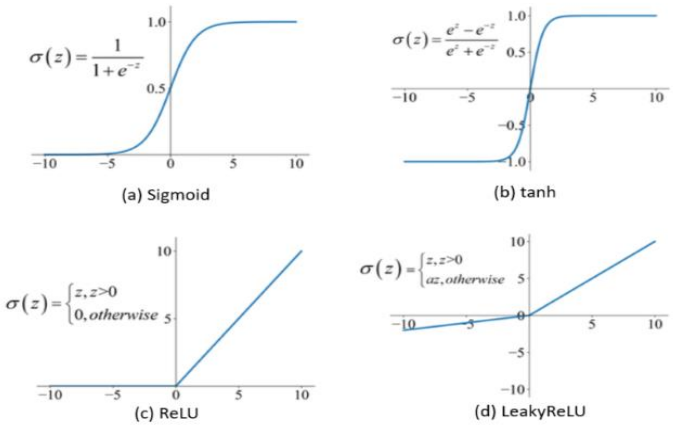



**Figure 2.17 :** Fonctions d'activation les plus courantes dans les réseaux neuronaux

### 2.3.4 Apprentissage en deep learning

L'apprentissage est le processus par lequel les paramètres libres d'un réseau de neurones s'adaptent, en utilisant un processus de stimulation à leur environnement. Le type d'apprentissage est déterminé par la façon dont ces adaptations ont lieu. Un algorithme d'apprentissage est un ensemble de règles bien définies qui résolvent un certain souci d'apprentissage[25] .

Il existe trois principales catégories d'apprentissage :

a **Apprentissage supervisé**

L'apprentissage supervisé est la tâche d'apprentissage automatique la plus simple et la plus connue. Il est basé sur un certain nombre d'exemples pré classifiés, dans lesquels

est connu à priori la catégorie à laquelle appartient chacune des entrées utilisées comme exemples. L'algorithme le plus connu pour l'apprentissage supervisé est le gradient, qui est par définition, pour un neurone, l'erreur relative à ce neurone. Il peut en quelque sorte être vu comme la contribution du neurone à l'erreur globale. A chaque rétro propagation, on calcule le gradient de chaque neurone, en commençant par ceux de la couche de sortie (les gradients des couches inférieurs se calculent à partir des gradients des couches supérieurs).[26]

**b Apprentissage non supervisé**

C'est de l'apprentissage par exploration où l'algorithme d'apprentissage ajuste les poids des liens entre neurones pour maximiser la qualité de classification des entrées.

**c Apprentissage par renforcement**

Les sorties idéales ne sont pas connues directement. Aussi, les poids sont ajustés de façons aléatoires et la modification est conservé si l'impact est positif ou non.

### 2.3.5 Cadres d'apprentissage profond

Frameworks offrent des blocs de construction pour la conception, l'entraînement et la validation de réseaux neuronaux profonds par le biais d'une interface de programmation de haut niveau. Voici quelques-uns des d'apprentissage profond les plus populaires, couramment utilisés pour la détection d'objets :

a **TensorFlow :** TensorFlow est un framework open-source développé par Google. Il fournit un écosystème complet pour construire et déployer des modèles d'apprentissage automatique, y compris des réseaux neuronaux profonds, avec un support pour les tâches de détection d'objets.

b **PyTorch :** PyTorch est un cadre d'apprentissage profond open-source développé par le laboratoire de recherche en IA de Facebook. Il est connu pour son graphe de calcul dynamique, qui permet un développement flexible des modèles et un débogage facile. PyTorch a gagné en popularité dans la communauté de l'apprentissage profond et offre des outils puissants pour la détection d'objets.

c **Caffe :** Caffe est un cadre d'apprentissage profond développé par le Berkeley Vision and Learning Center (BVLC). Il est largement utilisé pour diverses tâches de vision par ordinateur, notamment la détection d'objets. Caffe a une architecture simple et efficace, ce qui le rend adapté à un prototypage et à un déploiement rapides.

d **Keras :** Keras est un cadre d'apprentissage profond de haut niveau qui s'exécute au-dessus de TensorFlow ou d'autres moteurs dorsaux tels que Theano ou CNTK. Il fournit une interface conviviale et intuitive pour construire des réseaux neuronaux profonds, y compris des modèles pour la détection d'objets.

Ces cadres d'apprentissage profond prennent en charge à la fois le CPU et le GPU pour l'entraînement et l'inférence, offrant ainsi des options pour l'utilisation de différentes ressources matérielles. L'accélération GPU est particulièrement bénéfique pour les tâches d'apprentissage profond en raison de l'intensité de calcul qu'elles impliquent.[27]

## 2.4 Détection d'objets à l'aide de YOLO (You Only Look Once)

### 2.4.1 Algorithme YOLO

Grâce aux techniques avancées de vision par ordinateur, les objets présents dans les images peuvent être identifiés en quelques secondes avec une grande précision. Il existe aujourd'hui de nombreux algorithmes qui permettent d'effectuer une détection rapide des objets. YOLO est l'une de ces méthodes populaires de détection d'objets. La détection d'objets consiste à déterminer la position des objets dans une image et à classer ces objets. Les méthodes traditionnelles, telles que R-CNN, Fast-RCNN et Faster-RCNN, présentent des limitations en termes de vitesse et d'optimisation. [28]

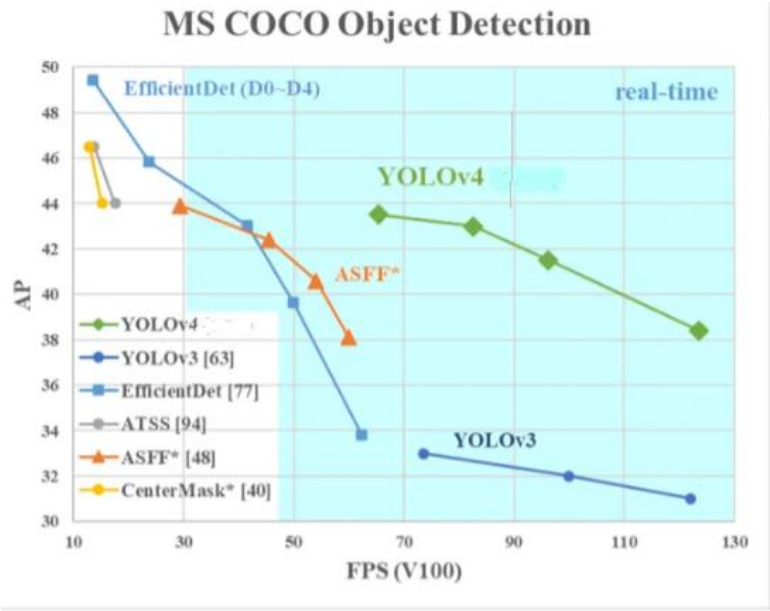



**Figure2.18:** Comparaison de la vitesse et de la précision de différents détecteurs d'objets

YOLO est un réseau neuronal convolutif (CNN) qui détecte les objets en temps réel. Il utilise un seul réseau neuronal pour traiter l'ensemble de l'image en entrée. L'image est ensuite divisée en régions, et YOLO prédit des boîtes de délimitation et des probabilités pour chacune d'entre elles. Les boîtes englobantes prédites sont pondérées en fonction de leur probabilité de contenir un objet. Pour notre projet, nous avons mis en œuvre l'algorithme YOLO version 5 (YOLOv5).[28]

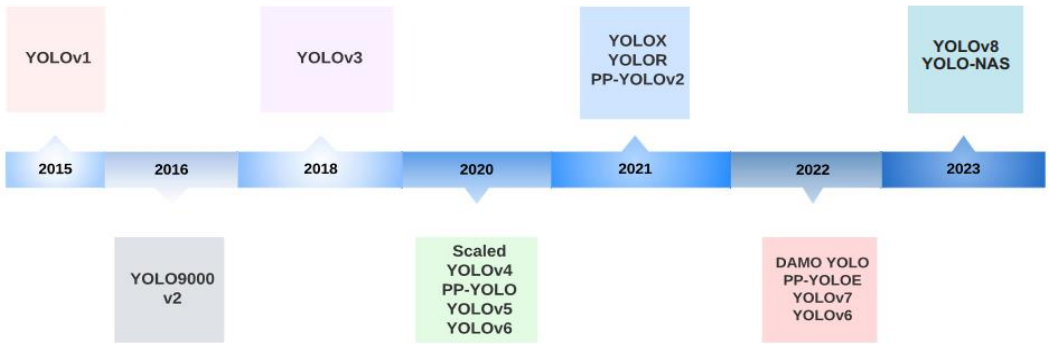



**Figure 2.19 :** Chronologie des versions de YOLO

### 2.4.2 Détails du modèle YOLO

L'algorithme YOLO utilise plusieurs techniques pour parvenir à une détection efficace et précise des objets, notamment les cellules de la grille, les boîtes de délimitation, les boîtes d'ancrage et les calculs d'intersection sur l'union (Intersection over Union : IoU). Ces techniques constituent la base de l'approche unique du YOLO en matière de détection d'objets dans les images. Dans ce qui suit, nous présentons ces techniques. [29]

**a Blocs résiduels Délimitation :**

Tout d'abord, l'image est divisée en plusieurs grilles. Chaque grille a une dimension de SxS. La figure (40) ci-dessous montre comment une image d'entrée est divisée en grilles. En réalité, une image peut être découpée en 19 x 19 cellules de grille. Pour cet exemple, nous utiliserons une grille de 3 x 3. Toutes les cellules de la grille sont de même dimension. Chaque cellule de la grille détectera les objets qui apparaissent en son sein. Par exemple, si un centre d'objet apparaît dans une certaine cellule de la grille, cette cellule sera chargée de le détecter.

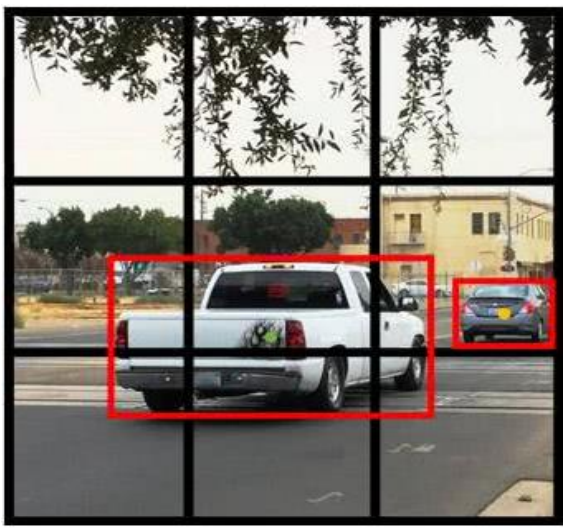


**Figure 2.20 :** Deux voitures détectées avec des boîtes englobantes surlignées en rouge

**b Bounding Box**

Une boîte englobante est une délimitation qui met en évidence un objet dans une image. Il s'agit d'un rectangle qui entoure un objet et qui spécifie sa position, sa classe (par exemple : voiture, personne) et sa confiance. Chaque boîte englobante d'une image se compose des attributs suivants :

- Largeur (bw)
- Hauteur (bh)
- Classe - (c)
- Centre de la boîte de délimitation (bx,by)

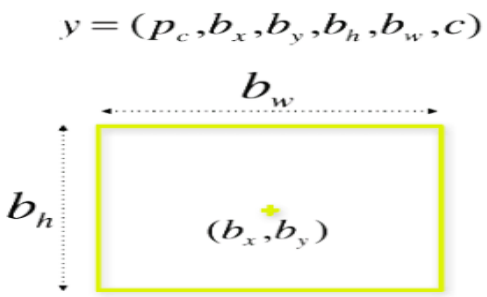



**Figure2.21:** Représentation de la boîte englobante

La figure 2.22 montre un exemple de coordonnées d'une boîte de délimitation. (La boîte englobante est représentée par un rectangle rouge). [30]

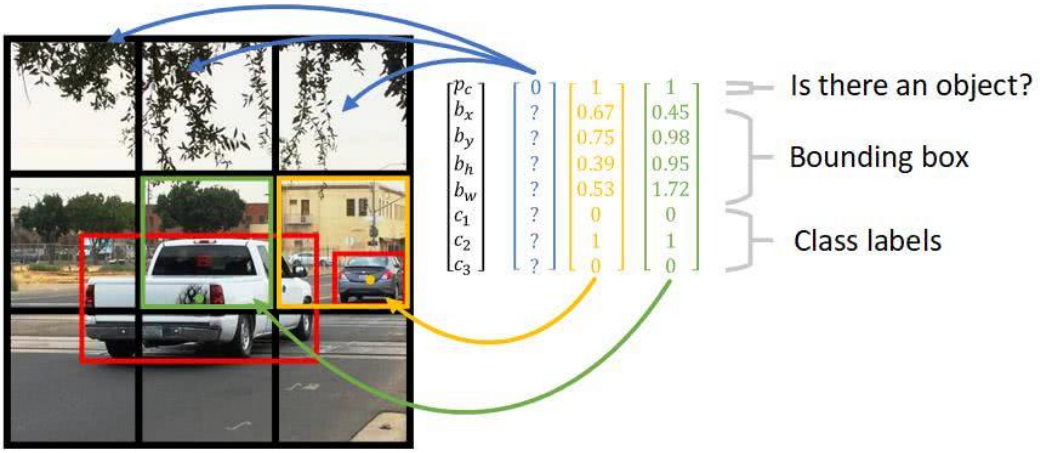



**Figure 2.22 :** Représentation de la sortie dans le cas du YOLO

## c Boite d’ancre

La boîte d'ancrage permet à l'algorithme YOLO de détecter plusieurs objets centrés dans une cellule de la grille. Comme le montre la figure 2.23, la voiture et le piéton sont tous deux centrés dans la cellule centrale de la grille.

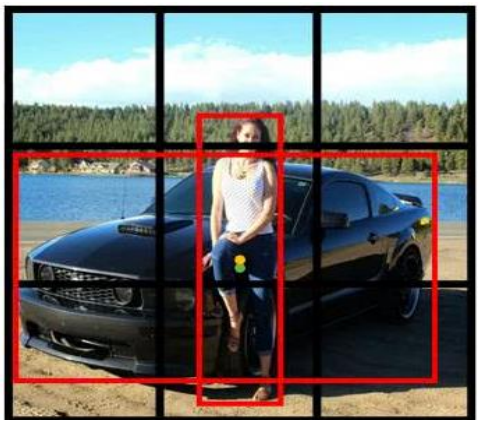


**Figure 2.23 :** Deux objets détectés dans la même grille

L'idée d'une boîte d'ancrage ajoute une "dimension" supplémentaire aux étiquettes de sortie en prédéfinissant plusieurs boîtes d'ancrage. Ainsi, nous pourrons assigner un objet à chaque boîte d'ancrage. À des fins d'illustration, nous choisirons deux boîtes d'ancrage de deux formes différentes, comme le montre la figure (2.24). [30]

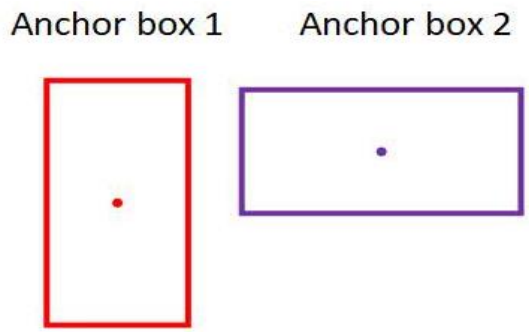



**Figure2.24:** Boîtes d'ancrage de différentes tailles

En utilisant deux boîtes d'ancrage de formes différentes pour chaque cellule de la grille, nous pouvons capturer les variations de formes des objets et améliorer la précision de la détection des objets. Le choix des boîtes d'ancrage nous permet d'associer des objets à des boîtes d'ancrage ayant des formes similaires. Par exemple, si un objet a une forme haute, il sera associé à la boîte d'ancrage qui a également une forme haute. Cela permet

de prédire avec précision les boîtes d'encerclement d'objets de formes différentes. Avec l'introduction de deux boîtes d'ancrage, la sortie de chaque cellule de la grille est étendue pour inclure des informations pour les deux boîtes d'ancrage. Dans l'exemple donné, la cellule centrale de la grille a maintenant un total de 16 étiquettes de sortie (8 étiquettes pour chaque boîte d'ancrage) comme le montre la figure (2.25). Cette expansion permet au modèle d'apprendre et de prédire les coordonnées, les scores d'objectivité et les probabilités de classe pour plusieurs objets à l'intérieur d'une seule cellule de la grille. [29]

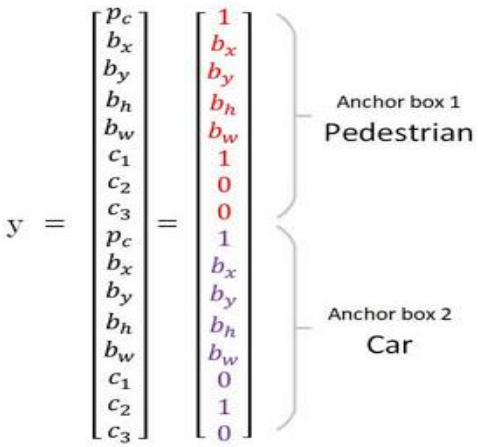



**Figure2.25:** Sortie du YOLO dans le cas de deux ancres dans la meme grille

En prenant en compte différentes formes de boîtes d'ancrage, l'algorithme YOLO améliore sa capacité à détecter et à localiser des objets de formes et de tailles variées, améliorant ainsi les performances globales du système de détection d'objets.

### d Intersection Over Union :

L'intersection par rapport à l'union (IOU) est une opération de détection d'objets qui décrit la manière dont les boîtes se chevauchent. YOLO utilise l'IoU pour fournir une boîte de sortie qui entoure parfaitement les objets. Chaque cellule de la grille est responsable de la prédiction des boîtes englobantes et de leurs scores de confiance. Ce mécanisme élimine les boîtes de délimitation qui ne sont pas égales à la boîte réelle. Il s'agit également d'une mesure couramment utilisée pour évaluer les performances de la détection d'objets en comparant la boîte de délimitation de la vérité terrain (ground truth) à la boîte de délimitation prédite, voir figure(2.26). [31]

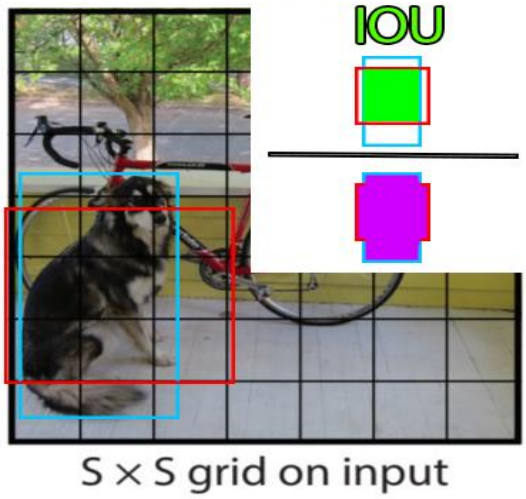



**Figure 2.26 :** Illustration du calcul de l'intersection over union

L'IoU est égale à 1 si la boîte de délimitation prédite est la même que la boîte réelle. En réalité, il est extrêmement improbable que les coordonnées (x, y) de notre boîte de délimitation prédite correspondent exactement aux coordonnées (x, y) de la boîte de délimitation réelle.

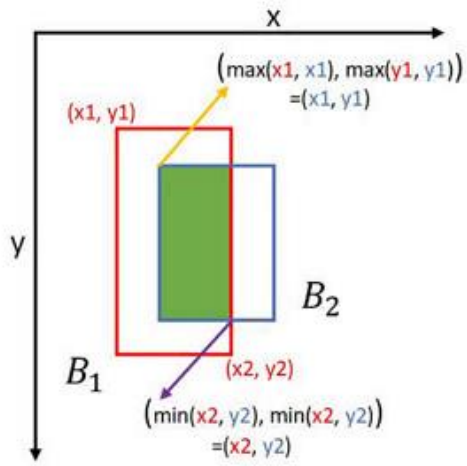



**Figure 2.27 :** Prédiction des coordonnées et de la taille de la boite englobante

### 2.4.3 Types de YOLOv5

Comparé à d'autres cadres de détection d'objets, YOLOv5 est extrêmement facile à utiliser pour mettre en œuvre des technologies de vision par ordinateur dans une application. Nous classons ces mises à jour conformément à la qualité de détection dans les catégories suivantes [32] :

- Facile à installer : YOLOv5 ne nécessite que l'installation de torch et de quelques bibliothèques Python légères.

- Formation rapide : Les modèles YOLOv5 s'entraînent très rapidement, ce qui permet de réduire les coûts d'expérimentation lors de l'élaboration de notre modèle.

- Des ports d'inférence qui fonctionnent : YOLOv5 vous permet d'effectuer des inférences sur des images individuelles, des lots d'images, des flux vidéo ou des ports de webcam.

- Structure intuitive du système de fichiers de données : La structure des dossiers de fichiers est intuitive et il est facile de naviguer pendant le développement.

- Facile à appliquer sur les appareils mobiles : Vous pouvez facilement traduire YOLOv5 de PyTorch weights à ONXX weights à CoreML à IOS.

YOLOv5 dispose de plusieurs variétés de modèles pré-entraînés comme nous pouvons le voir dans la figure (48) ci-dessous. La différence entre eux est le compromis entre la taille du modèle et le temps d'inférence. La version légère du modèle YOLOv5s ne pèse que 14 Mo mais n'est pas très précise. De l'autre côté du spectre, nous avons YOLOv5x dont la taille est de 168 Mo mais qui est la version la plus précise de sa famille. [33]


**Figure 2.28 :** Comparaison des modèles YOLOv5

YOLOv5 se décline en quatre versions principales : petite (s), moyenne (m), grande (l) et très grande (x), comme le montre la figure (2.29) ci-dessus. Chacune offre des taux de précision progressivement plus élevés. Chaque variante nécessite également un temps de formation différent. [34]

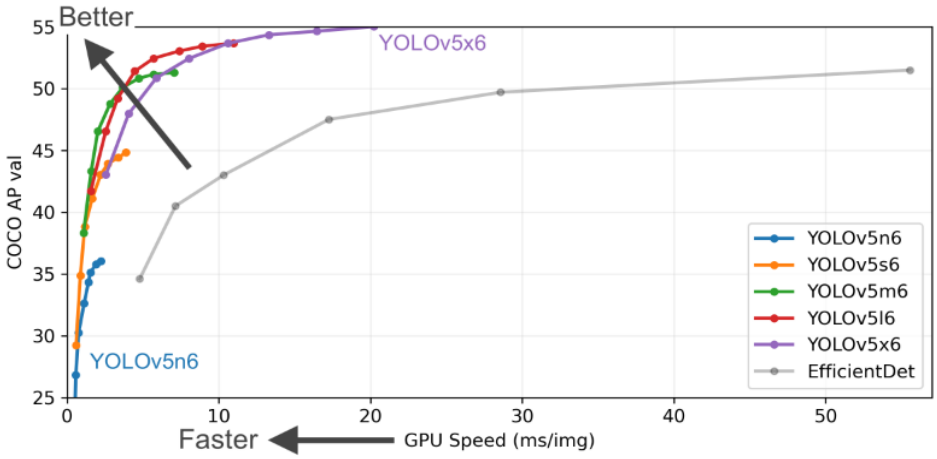



**Figure2.29**: Performance de YOLOv5 modèles pour différentes tailles

### 2.4.4 Architecture de YOLOv5

L'architecture YOLOv5 est un modèle de détection d'objets de pointe qui vise à assurer une détection rapide et précise des objets en temps réel. Elle s'appuie sur les versions précédentes de YOLO et introduit plusieurs améliorations clés.

L'architecture du YOLOv5 peut être divisée en trois composants principaux : backbone, neck et head. [35]

- **Backbone :** Il utilise CSPDarknet comme colonne vertébrale pour l'extraction de caractéristiques à partir d'images composées de réseaux partiels à plusieurs niveaux. Il est initialement entraîné sur un ensemble de données de classification à une résolution inférieure.
- **Neck :** Il utilise PANet (Path Agregation Network) pour générer un réseau de pyramides de caractéristiques afin d'effectuer une agrégation sur les caractéristiques et de les transmettre à Head pour la prédiction.
- **Head :** Couches qui génèrent des prédictions à partir des boîtes d'ancrage pour la détection d'objets

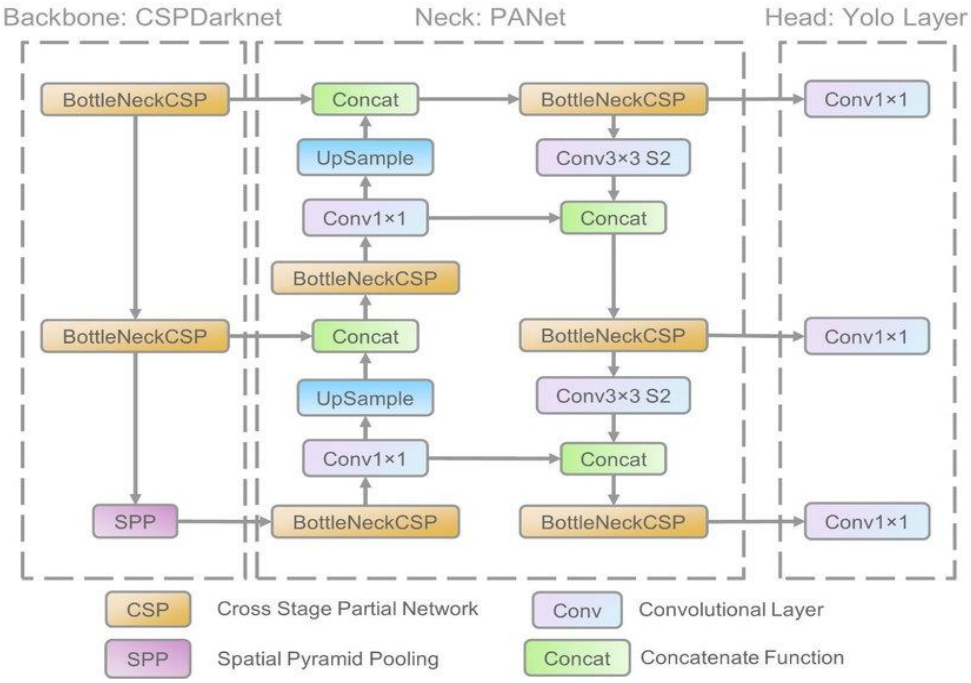



**Figure2.30:** Architecture du réseau Yolov5

## 2.5 Détection d'objets et métriques d'évaluation

Les métriques de détection d'objets servent à évaluer les performances du modèle dans le cadre d'une tâche de détection d'objets. Elles nous permettent également de comparer objectivement plusieurs systèmes de détection ou de les comparer à un point de référence. En conséquence, des concours importants tels que PASCAL VOC et MSCOCO fournissent des métriques prédéfinies pour évaluer les performances de différents algorithmes de détection d'objets sur leurs ensembles de données. Il existe de nombreuses mesures pour évaluer les modèles d'apprentissage automatique. Chacune d'entre elles présente des avantages et des inconvénients[36]. Voici quelques mesures d'évaluation couramment utilisées :

### 2.5.1 Intersection over Union

L'intersection sur l'union (IoU) est une mesure qui quantifie le degré de chevauchement entre deux régions. Comme nous l'avons déjà expliqué plus haut dans ce chapitre, ce concept permet d'évaluer la précision d'une prédiction et fournit une valeur comprise entre 0 et 1 [37]. Voici un récapitulatif de l'illustration de l'intersection sur l'union dans la figure (2.40) .

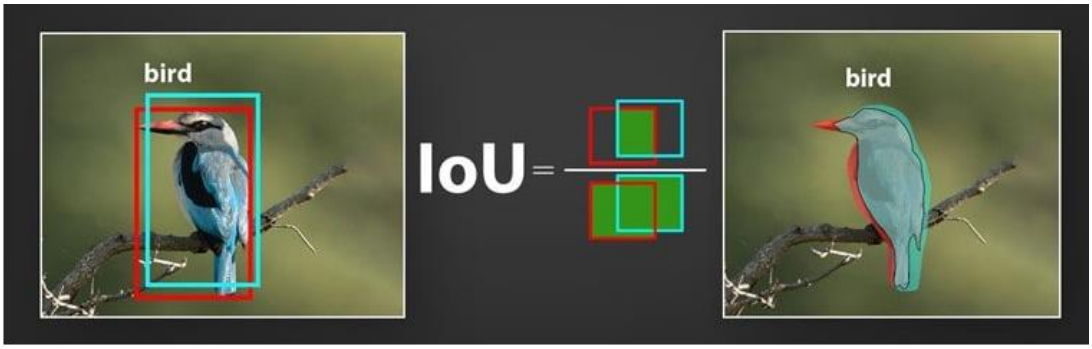



**Figure 2.40 :** Visualisation de l'intersection over union

L'IoU est calculé en divisant le chevauchement entre l'annotation prédite et l'annotation de vérité terrain par l'union de ces deux annotations :

$$IoU(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

La figure (2.41) ci-dessous présente quelques résultats du calcul de l'indice d'utilité (IoU(A,B)):


**Figure 2.41 :** Exemple de calcul de l'intersection sur les unions pour différentes boîtes englobantes

### 2.5.2 Matrice de Confusion

Après avoir appliqué un seuil d'utilité, nous pouvons classer une prédiction comme vraie positive, fausse positive ou fausse négative. Il convient de noter que ces termes forment collectivement la matrice de confusion [38].

Nous les comparons généralement à la vérité de terrain et les répartissons en quatre groupes :

- **Vrai positif [True Positive TP]:** le modèle détecte correctement un objet.
- **Faux positif [False Positive FP]:** un objet ne figurant pas sur l'image est détecté.
- **Faux positif [False Positive FN]:** un objet n'est pas détecté dans la réalité du terrain.
- **Vrai négatif [True Negative TN]:** le modèle prédit correctement la classe négative.

Les quatre groupes forment ce que l'on appelle la matrice de confusion, présentée dans l'illustration de figure (2.42).

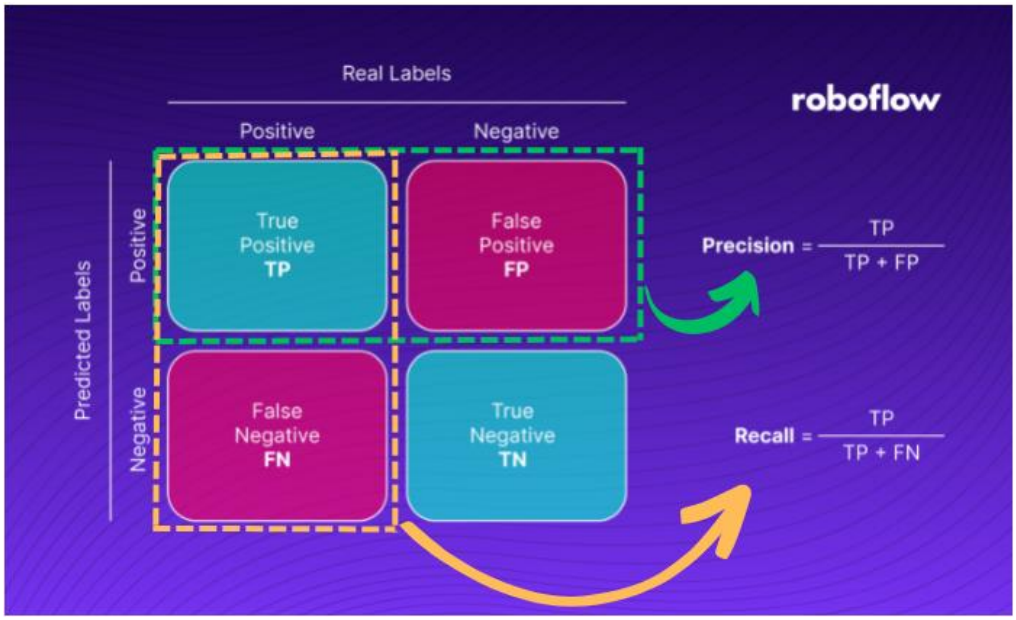



**Figure 2.42 :** Illustration de la matrice de confusion

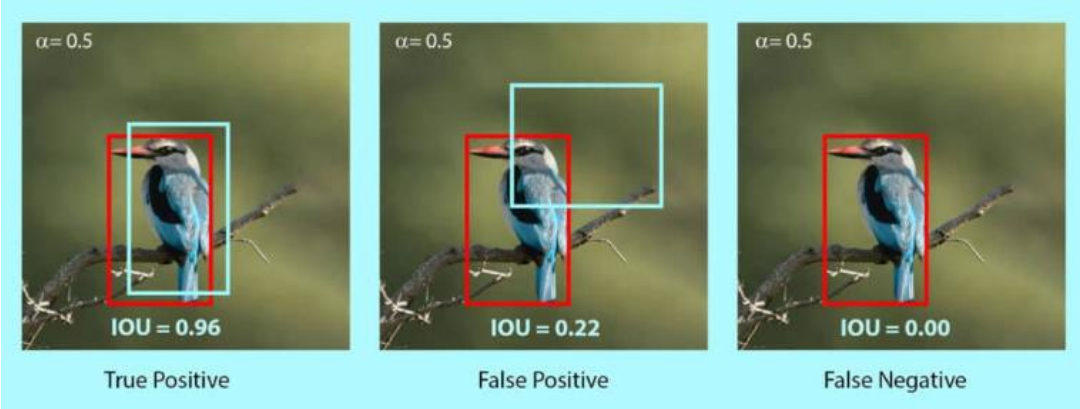



**Figure 2.43 :** Résultats de détection basée sur le calcul de l'IoU

Dans la détection d'objets, la justesse de la prédiction (TP, FP ou FN) est déterminée à l'aide du seuil de l'IoU. La vérité de terrain désigne les objets connus.

### 2.5.3 Précision

La précision quantifie l'exactitude des prédictions positives en évaluant la proportion de prédictions positives qui sont effectivement correctes. Pour calculer la précision, il suffit de diviser le nombre de vrais positifs par le nombre total de détections [39]. Mathématiquement, la précision est définie comme suit :

$$\boldsymbol{P = \frac{TP}{(TP + FP)} = \frac{TP}{toutes\ les\ detections}}$$

La valeur est comprise entre 0 et 1.

### 2.5.4 Rappel (Recall)

Le rappel mesure la proportion de résultats positifs réels qui ont été prédits correctement. Il s'agit des vrais positifs parmi toutes les vérités de terrain. Mathématiquement, il est défini comme suit :

$$\boldsymbol{R = \frac{TP}{(TP + FN)} = \frac{TP}{toutes\ les\ vérités\ de\ terrain}}$$

Comme pour la précision, la valeur du rappel est également comprise entre 0 et 1.

### 2.5.5 Moyenne de la précision moyenne

La moyenne de la précision moyenne (mAP ; mean Average Precision) est particulièrement utile pour évaluer les modèles qui traitent plusieurs classes ou requêtes. La mAP est calculée en calculant d'abord la précision moyenne (AP) pour chaque classe ou requête. La précision moyenne mesure la précision à différents niveaux de rappel et résume la courbe précision-rappel pour une classe ou une requête spécifique. Il quantifie l'efficacité avec laquelle le modèle classe les instances pertinentes par rapport aux instances non pertinentes [40]. Afin de réduire l'impact des fluctuations de la courbe, nous interpolons d'abord la précision à plusieurs niveaux de rappel avant de calculer la PA. La précision interpolée $Pinterp$ à un certain niveau de rappel r est définie comme la précision la plus élevée trouvée pour tout niveau de rappel r ′ ≥ r :

$$P_{interp}(r) = \max_{r\prime \geq r} p(r')$$

L'AP peut alors être défini comme l'aire sous la courbe de précision-recours interpolée, qui peut être calculée à l'aide de la formule suivante :

$$\boldsymbol{AP} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n-1}} (\boldsymbol{r_{i+1}} - \boldsymbol{r_i}) \boldsymbol{P_{interp}}(\boldsymbol{r_{i+1}})$$

D'où $r_1$ , $r_2$ , . . . , $r_n$ sont les niveaux de rappel (dans un ordre croissant) auxquels la précision est interpolée pour la première fois. Étant donné qu'il existe généralement plus d'une classe (notée K) dans la détection d'objets, la précision moyenne (mAP) est introduite. La mAP est définie comme la moyenne des valeurs AP calculées pour chaque classe individuelle parmi les K classes. Cela permet une évaluation complète des performances de toutes les classes dans la détection d'objets. Elle est définie comme suit :

$$mAP = \frac{\sum_{i=1}^{K} AP_i}{K}$$

## 2.6 Conclusion

Ce chapitre a abordé des sujets clés de la vision par ordinateur, de l'apprentissage profond et de la détection d'objets. Nous avons exploré les principes fondamentaux du traitement d'images, donné un aperçu des techniques d'apprentissage profond et approfondi le concept de détection d'objets à l'aide de YOLO. En outre, nous avons discuté de l'importance des métriques d'évaluation pour évaluer la performance des modèles. Ces sujets sont essentiels dans divers domaines et ouvrent la voie à de futures avancées dans les technologies de vision par ordinateur et de détection d'objets.

Dans le prochain chapitre, nous avons mis en pratique les connaissances que nous avons acquises dans le cadre de ce projet, en appliquant les concepts du traitement d'images, l'apprentissage profond et de détection d'objets.

# Chapitre 3 Méthodologie et Implémentation

## 3.1 Introduction

Dans ce chapitre, nous nous plongerons dans la mise en œuvre de notre système qui utilise des algorithmes d'apprentissage profond pour la détection et le décodage des codes-barres 1D. Notre étude se concentre sur l'évaluation de l'efficacité de l'algorithme YOLOv5 pour entraîner un modèle capable de détecter avec précision différents types de codes-barres 1D, et tester ses performances dans des scénarios réels. Nous utilisons des outils tels que Roboflow pour annoter et étiqueter les données d'entraînement, et Google Colab pour l'entraînement et le test du modèle. Notre objectif ultime est de développer un système fiable et efficace pour lire les codes-barres dans des environnements diversifiés.

## 3.2 Environnement de travail

### 3.2.1 L'ordinateur portable

Nous avons travaillé sur notre projet avec un ordinateur portable avec ces capacités :

- HP I7
- Processeur : Intel(R) Core(TM) i7-4600U CPU @ 2.10GHz 2.70 GHz
- Mémoire vive : 8,00 Go

### 3.2.2 Caméra

Pour capturer les images des codes-barres, nous avons utilisé soit la caméra intégrée à notre ordinateur, soit la caméra de notre smartphone. Cela nous a permis d'obtenir des images claires et de haute qualité pour le traitement ultérieur.

## 3.3 Logiciels utilisés

### 3.3.1 Google Collab

Google Colab est une plateforme basée sur le cloud qui offre un accès gratuit à un environnement de notebook Jupyter avec une prise en charge intégrée de l'exécution du code Python et de l'utilisation de différents frameworks d'apprentissage en profondeur. Il propose des ressources GPU et TPU gratuites pour l'entraînement des modèles d'apprentissage en profondeur, ce qui en fait un choix populaire pour la recherche et l'expérimentation dans le domaine de la vision par ordinateur[41].

### 3.3.2 Roboflow

Roboflow est une plateforme en ligne qui fournit des outils pour annoter, gérer et préparer les données d'entraînement pour les modèles de vision par ordinateur. Elle propose une variété de formats d'annotation, tels que les boîtes englobantes, les masques de segmentation et les points clés, et prend en charge l'intégration avec les frameworks d'apprentissage en profondeur populaires[52].

### 3.3.3 Visual Studio Code

Visual Studio Code (VS Code) est un éditeur de code source populaire développé par Microsoft, connu pour sa légèreté et sa personnalisation. Il offre une plateforme polyvalente aux développeurs pour écrire, modifier et déboguer du code dans divers langages de programmation. Nous avons utilisé Python avec Visual Studio Code pour notre projet.

## 3.4 Bibliothèques utilisées

**Tableau 1.1 :** Table regroupant les bibliothèques utilisées

| Bibliothèque | Utilitaire |
|---|---|
| OpenCV | Bibliothèque de liaisons Python conçue pour résoudre des problèmes de vision par ordinateur. Elle offre une large gamme de fonctionnalités pour le traitement d'images, la détection d'objets, la reconnaissance faciale, la segmentation d'images et bien plus encore. OpenCV-Python est largement utilisée dans le domaine de la vision par ordinateur pour développer des applications et des systèmes de traitement d'images[43]. |
| Pyzbar | Bibliothèque Python populaire qui offre des fonctionnalités de décodage de codes-barres. Elle permet de décoder différents types de codes-barres, y compris les codes-barres 1D (comme les codes EAN et UPC) et les codes-barres 2D (comme les codes QR et les codes DataMatrix)[44]. |
| Torch | Bibliothèque open source d'apprentissage automatique basée sur la bibliothèque Torch. Elle est utilisée pour des applications telles que la vision par ordinateur et le traitement du langage naturel.PyTorch offre une interface conviviale pour la création et l'entraînement de modèles d'apprentissage profond, avec une prise en charge intégrée du calcul sur GPU. La flexibilité et la facilité d'utilisation de PyTorch en font un choix populaire parmi les chercheurs et les praticiens de l'apprentissage automatique[45]. |
| Numpy | Bibliothèque pour le langage de programmation Python, qui ajoute la prise en charge des tableaux et matrices multidimensionnels de grande taille, ainsi qu'une vaste collection de fonctions mathématiques de haut niveau pour effectuer des opérations sur ces tableaux [46]. |
| Tkinter | Le tkinterpackage ("Tk interface") est l'interface Python standard de la boîte à outilsTcl/Tk GUI. Tk et tkintersont disponibles sur la plupart des plates-formes Unix, y compris macOS, ainsi que sur les systèmes Windows [47]. |

## 3.5 Implémentation du système proposé

Dans ce système, nous divisons notre tâche en deux parties principales : la détection des codes-barres à l'aide de YOLOv5 et le décodage après l'application de techniques de traitement d'images comme illustré dans la Figure 3.1.

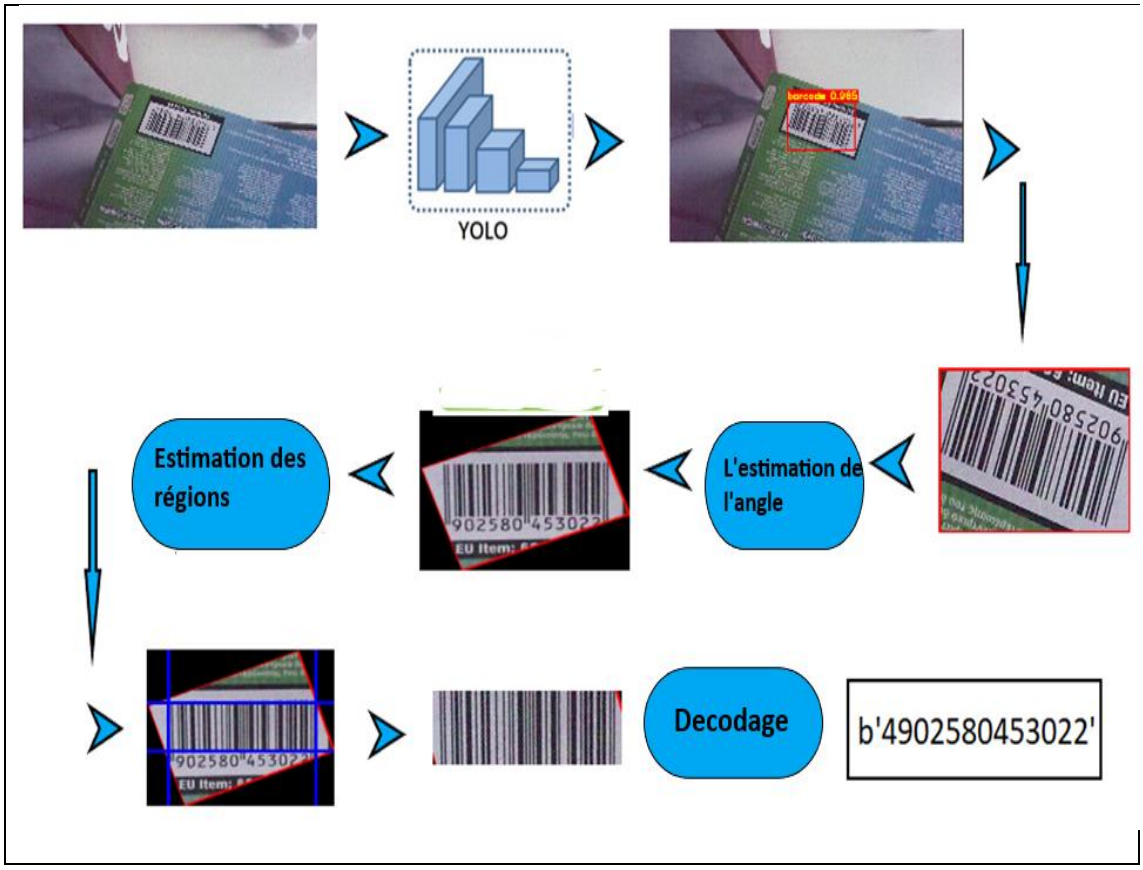



**Figure3.1:** Présentation du système proposé

### 3.5.1 Détection

Dans cette partie, nous allons créer une base de données, entraîner notre modèle YOLOv5 et enfin effectuer des tests.

#### a Création de base de données

Pour le jeu de données, nous avons collecté un total de 500 images à l'aide d'un smartphone. Nous avons délibérément inclus des codes-barres de différents types, tailles et orientations. Nous avons veillé à capturer des images dans différentes conditions d'éclairage et d'angles pour rendre le jeu

de données plus robuste. Cette collection complète d'images constitue une base solide pour l'entraînement et l'évaluation de notre modèle YOLOv5 pour la détection des codes-barres 1D.

**Figure3.2:** Echantillons d'images de la base de données créée

L'étape suivante consiste à étiqueter les images pour obtenir les données nécessaires à chaque image, à savoir les coordonnées des boîtes englobantes et les classes, car c'est exactement ce que YOLOv5 requiert pour entraîner le jeu de données. Pour étiqueter les images, nous utilisons le logiciel Roboflow. Roboflow offre une interface conviviale qui nous permet d'annoter les images en dessinant des boîtes englobantes autour des codes-barres.

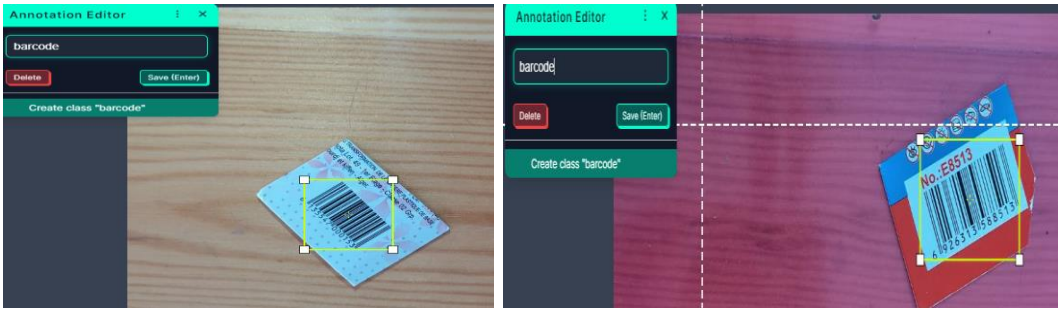



**Figure 3 .3 :** L'annotation du code-barres sur Roboflow

Une fois que la base de données a été étiquetée, nous avons utilisé des techniques d'augmentation de données pour améliorer la robustesse du modèle, comme illustré dans la figure (3.4). L'objectif de l'augmentation de données est d'accroître la diversité des données d'entraînement, ce qui peut améliorer la capacité du modèle à généraliser à de nouvelles données. Cela augmente la taille de l'ensemble d'entraînement en apportant de légères modifications aux images d'entraînement.

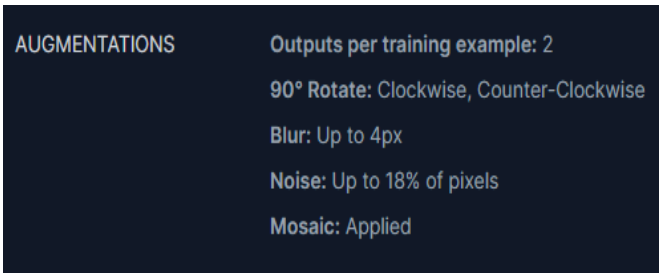



**Figure3.4:** Types des augmentations

Après l'augmentation de la base des données, nous avons divisé celle-ci en ensembles d'entraînement, de validation et de test. L'ensemble d'entraînement a été utilisé pour entraîner le modèle d'apprentissage profond, tandis que l'ensemble de validation a été utilisé pour évaluer les performances du modèle pendant l'entraînement. La séparation des données en ensembles d'entraînement, de validation et de test vise à éviter que le modèle ne sur ajuste et à évaluer précisément notre modèle.

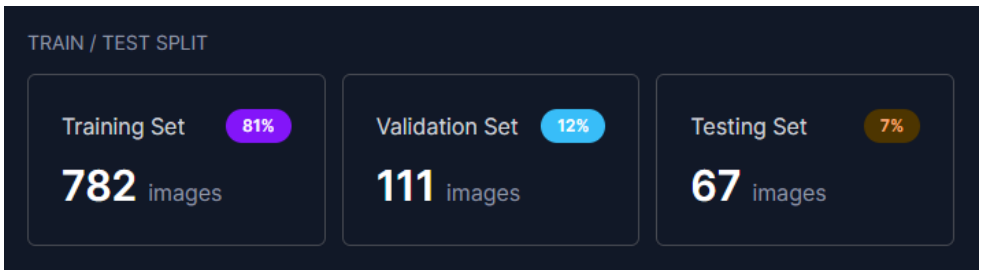



**Figure3.5:** Division de la base de données en train/ valid/ test.

### b Entrainement du réseau sur Yolov5

Pendant la phase d'entraînement de notre réseau YOLOv5, nous avons utilisé Google Colab pour rendre le processus d'entraînement plus facile et plus rapide en exploitant le GPU fourni.

Dans la première étape, nous installons le dépôt YOLOv5 ainsi que les dépendances nécessaires (3.6). Nous pouvons utiliser l'un des fichiers de configuration prédéfinis, disponibles dans le dépôt YOLOv5, qui inclut l'architecture du modèle, les hyper paramètres et d'autres réglages, pour définir la configuration d'entraînement.


**Figure 3.6 :** Installation du Yolov5

Nous avons entraîné le modèle YOLOv5 en utilisant le script train.py du dépôt YOLOv5. Nous devons fournir le chemin vers le fichier de configuration d'entraînement et le chemin vers l'ensemble de données d'entraînement. Nous avons utilisé la commande suivante pour entraîner le modèle :

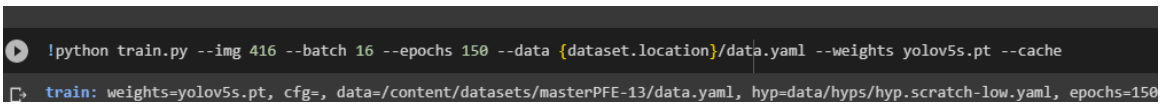



**Figure 3.7 :** Instruction d'apprentissage

- **--img 416 :** définit la taille de l'image d'entrée à 416x416 pixels. Nous avons choisi une taille d'image relativement petite pour des raisons de performances. Une taille d'image plus grande aurait nécessité plus de ressources de calcul et aurait ralenti le processus d'entraînement. De plus, cette taille d'image est suffisamment grande pour capturer les détails importants des codes-barres tout en restant raisonnable en termes de temps de traitement.

- --**Batch 16 :** définit la taille du lot (batch size) à 16 images. Nous avons défini la taille du lot à 16 images. Cela signifie que pendant chaque itération d'entraînement, le modèle reçoit 16 images à la fois. Un lot de taille appropriée peut aider à tirer parti des capacités de calcul parallèle du GPU, ce qui accélère le processus d'entraînement. Nous avons choisi

une taille de lot de 16 en tenant compte des ressources disponibles et en optimisant l'équilibre entre l'efficacité et les performances.

- **--Epochs 150 :** définit le nombre d'époques à 150. Une époque correspond à une itération complète sur l'ensemble des données d'entraînement. Le choix du nombre d'époques dépend de la complexité du modèle et de la taille de l'ensemble de données. En choisissant 150 époques, nous avons donné suffisamment de temps au modèle pour apprendre les caractéristiques des codes-barres et converger vers une solution optimale.

- **--Weights yolov5s.pt :** spécifie le chemin vers les poids du modèle YOLOv5s pré-entraîné Nous avons utilisé les poids pré-entraînés du modèle YOLOv5s comme point de départ pour notre entraînement. Ces poids pré-entraînés contiennent les connaissances générales du modèle sur la détection d'objets à partir d'un ensemble de données plus vaste. En initialisant notre modèle avec ces poids, nous avons facilité et accéléré le processus d'apprentissage, car le modèle avait déjà appris certaines caractéristiques de la détection d'objets.

- **--Cache :** active la mise en cache des images pour accélérer l'entraînement. Nous avons activé la mise en cache des images pendant l'entraînement. Cela signifie que les images sont stockées en mémoire pour éviter de les recharger à chaque itération d'entraînement. Cela améliore les performances en réduisant le temps de chargement des données et en accélérant le processus d'entraînement.

Résultats de l'apprentissage :

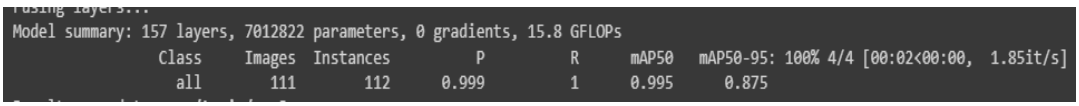



**Figure3.8:** Résultats de l'apprentissage

L'entraînement du réseau n'a pris que 15 minutes. Cette réduction significative

du temps d'entraînement peut être attribuée aux capacités de calcul parallèle du GPU, qui ont permis des calculs plus rapides et des optimisations lors des itérations d'entraînement. En exploitant la puissance de traitement du GPU, nous avons pu entraîner notre réseau YOLOv5 de manière plus efficace, permettant une convergence plus rapide du modèle et une expérimentation plus rapide avec différentes configurations. Ce temps d'entraînement accéléré a finalement accéléré le développement et le perfectionnement de notre système de détection de codes-barres, ce qui a permis de gagner un temps précieux et des ressources.

**c Performance**

Dans les résultats, nous avons obtenu une précision impressionnante de 0,99, Rappel de 1 et une précision moyenne (mAP) de 0,995. Ces métriques indiquent les performances élevées et la précision de notre modèle YOLOv5 pour la détection des codes-barres.

- Une précision de 0,99 signifie que sur l'ensemble des codes-barres prédits, 99 % d'entre eux sont correctement identifiés. Cela indique un taux de faux positifs très faible, montrant que notre modèle est précis dans l'identification des vrais positifs.

- Un Rappel de 1 signifie que le modèle a réussi à détecter tous les codes-barres pertinents dans l'ensemble de données, sans aucun faux négatif. Cela indique un niveau élevé de sensibilité dans la capture de toutes les instances de codes-barres présentes.

- La précision moyenne moyenne (mAP) de 0,995 démontre les performances globales du modèle en termes de précision et de rappel, en tenant compte de différentes classes d'objets et de seuils de détection. Cette valeur élevée indique que notre modèle atteint un excellent compromis entre précision et rappel et se comporte de manière constante sur l'ensemble de données.

- La perte (loss): Une perte plus faible indique que le modèle s'ajuste bien aux données d'entraînement.

Ces résultats exceptionnels mettent en évidence l'efficacité de notre modèle pour détecter et localiser avec précision les codes-barres, et démontrent son potentiel pour des applications de détection de codes-barres dans le monde réel.

La figure ci-dessous représente les résultats obtenus.

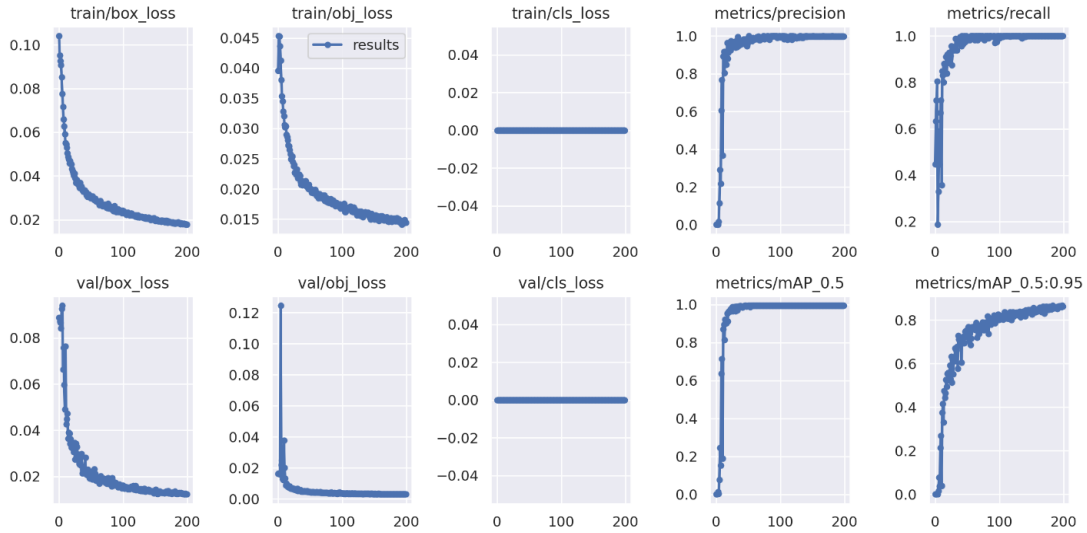



**Figure3.9:** Les résultats que nous avons obtenus

### d Tests

Pour évaluer les performances de notre modèle entraîné, nous avons développé un script Python. Nous avons chargé les meilleurs poids obtenus lors de la phase d'entraînement dans le script. Lorsqu'une image d'entrée est fournie, le script utilise le modèle pour détecter la présence d'un code-barres. Si un code-barres est détecté, le script dessine une boîte englobante autour de celui-ci, permettant d'identifier visuellement l'emplacement du code-barres. De plus, le script affiche le score de probabilité associé à la détection, fournissant une indication du niveau de confiance de la détection. Les coordonnées de la boîte englobante sont également fournies, permettant une

localisation précise du code-barres dans l'image. Enfin, le script mesure le temps d'exécution.

Voici quelques résultats de tests obtenus en exécutant le script :

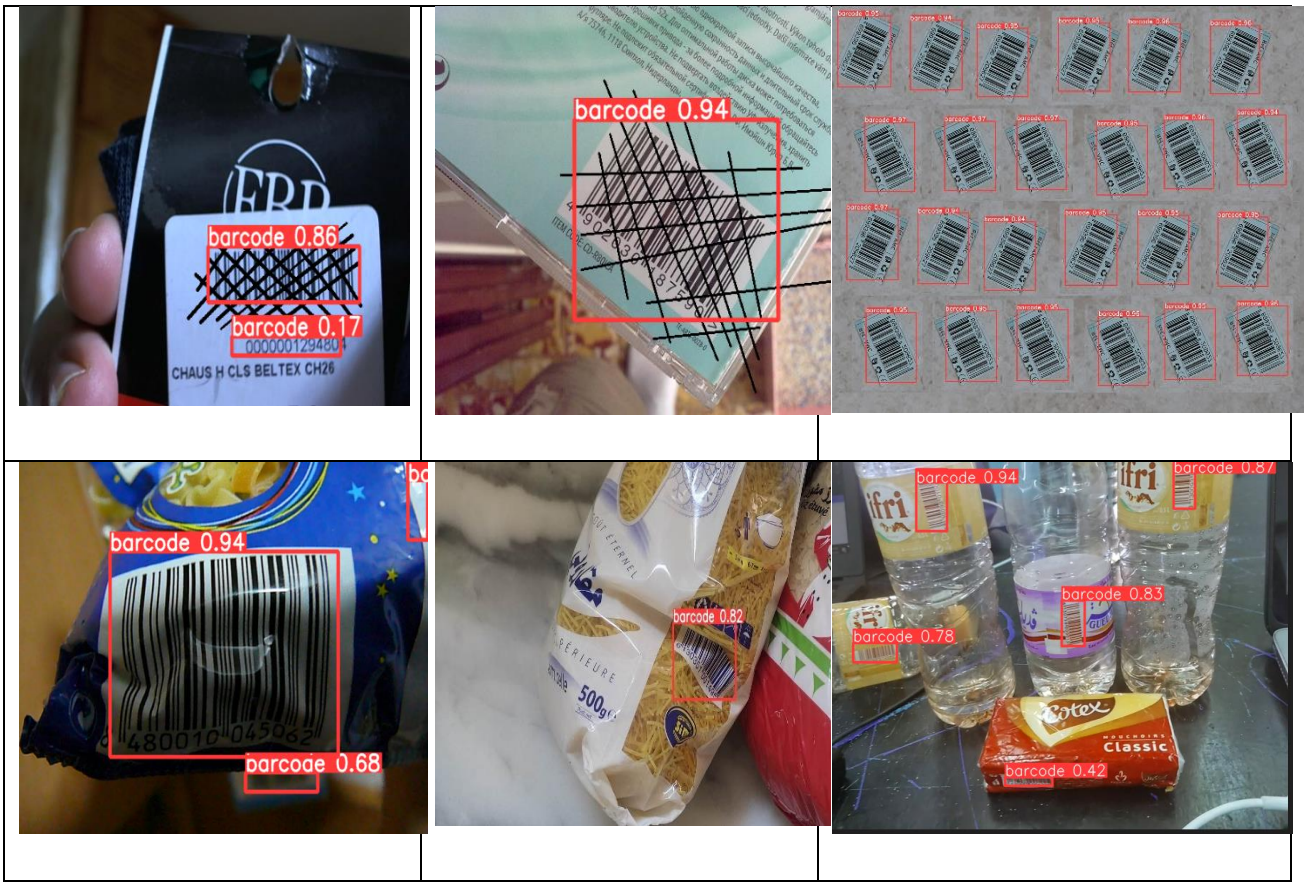



**Figure 3.10 :** Résultats de test

### 3.5.2 Décodage

Pour décoder un code-barres, il est important qu'il soit horizontal et bien défini. Pour cela, nous avons suivi les étapes suivantes :

#### a Recadrer l'image

La première étape consiste à extraire la région d'intérêt correspondant au code-barres détecté en utilisant les coordonnées du rectangle englobant obtenues lors du processus de détection. Cela nous permet de concentrer nos étapes de traitement et de décodage ultérieures spécifiquement sur la région du code-barres, optimisant ainsi la précision et l'efficacité de la reconnaissance et du décodage du code-barres.

**Figure 3.11 :** Image détectée

**Figure 3.12 :** Image-recadrée

### b Estimation de l'angle

L'estimation de l'angle vise à déterminer l'orientation du code-barres par rapport à l'axe horizontal. Cela devient nécessaire si le code-barres n'est pas parfaitement aligné et nécessite une correction d'orientation pour une détection et un décodage précis. Pour estimer l'angle, nous pouvons appliquer un filtre gaussien pour réduire le bruit et améliorer la qualité de l'image. Ensuite, nous utilisons l'opérateur de Sobel pour détecter les contours horizontaux et verticaux dans la région du code-barres.


**Figure 3.13** : l'angle affichée par le système

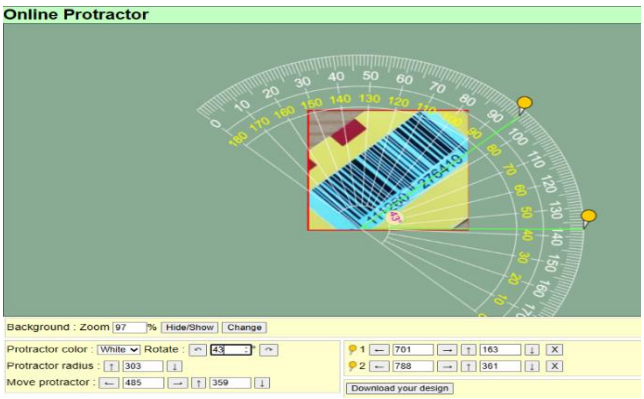



**Figure 3.14 :** Vérification de l'angle affichée

### c Orientation

Après avoir estimé l'angle d'inclinaison, l'étape suivante consiste à appliquer une rotation à la région du code-barres afin de l'aligner horizontalement. Cette rotation garantit que le code-barres est correctement orienté pour un décodage précis.


**Figure 3.15 :** Code-barres orienté par notre système

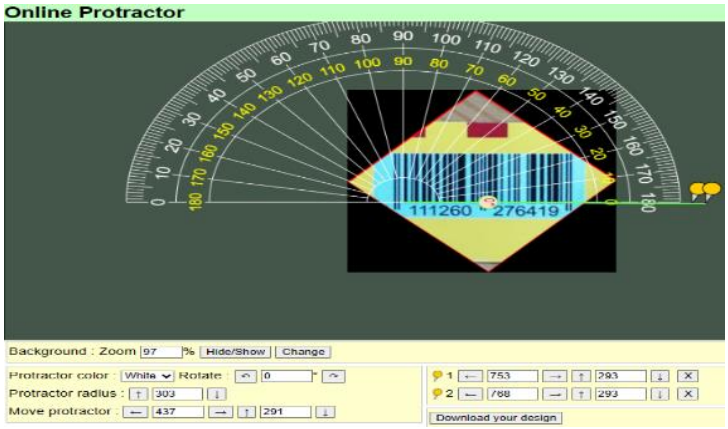



**Figure3.16:** Vérification de l'orientation de code à barres

### d Estimation des regions

L'estimation des régions ou le calcul des profils est essentiel pour analyser les variations d'intensité des pixels dans la région du code-barres, ce qui permet une détection précise des transitions et une caractérisation des éléments du code-barres. Ces informations sont d'une importance capitale pour extraire les données encodées lors de l'étape de décodage ultérieure.


**Figure 3.17 :** Estimation des régions

**e Décodage**

Dans la dernière étape de notre processus, nous utilisons les informations extraites du code-barres, y compris les transitions détectées et les caractéristiques des barres et des espaces, pour effectuer le décodage à l'aide de pyzbar.

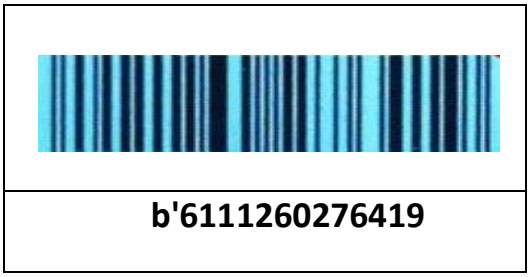



**Figure3.18:** Décodage de code-barres

**f Localisation du code à barres**

Les quatre paramètres spécifiques du code-barres, notamment la largeur, la hauteur, l'angleet le centre sont déterminés par l'utilisation d'une étape supplémentaire basée sur le traitement d'images. Ces paramètres nous permettent effectuer une localisation précise du code-barres, comme illustré dans la figure (3.19).

**Figure 3.19 :** Résultat final de localisation

## 3.6 Discussions des résultats

Après avoir testé le système, nous pouvons affirmer que notre système présente une bonne précision et zéro erreur de décodage. Les résultats montrent les performances et la fiabilité du système dans les tâches de détection et de décodage des codes-barres.

La précision de notre système témoigne de sa capacité à détecter et interpréter avec précision les codes-barres. Cette précision est essentielle dans les applications où l'identification correcte des codes-barres est primordiale.

L'absence d'erreurs de décodage est la preuve de la robustesse et de l'exactitude du système. Cela signifie que le système extrait de manière fiable les données encodées des codes-barres sans aucune erreur ou mauvaise interprétation. Ce haut niveau de précision inspire confiance en la capacité du système à fournir des informations précises et fiables sur les codes-barres.

En combinant la précision et l'absence d'erreurs de décodage, notre système est bien équipé pour gérer différents types de codes-barres et des conditions difficiles. Il établit une base solide pour son intégration dans diverses applications où la détection et le décodage précis, rapides et sans erreurs des codes-barres sont essentiels.

## 3.7 Evaluation

Dans cette section, nous avons mesuré les résultats sur la base de données Arte-Lab avec les vérités terrain provenant de (Zamberletti et al., 2013)[11].

Pour chaque image de la base de données, nous commençons par tracer la vérité terrain, qui comprend les caractéristiques réelles du code-barres telles que la position du centre, la longueur et l'angle.

Ensuite, nous comparons la vérité terrain obtenu (G) à partir de notre algorithme avec le deuxième vérité terrain (F) pour les résultats de détection trouvés. L'indice de Jaccard entre eux est défini comme étant le rapport d'intersection sur l'union, également connu sous le nom d'indice de Jaccard ou IoU .

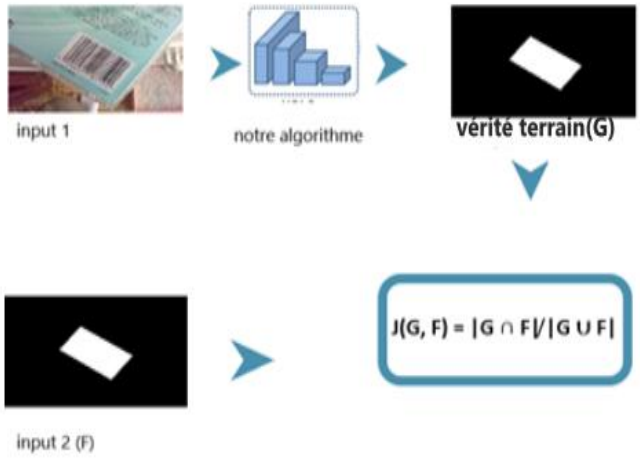



**Figure 3 .20 :** Présentation de calcul de l'indice de Jaccard

Un taux de détection pour un seuil de IoU donné T est défini comme la fraction d'images dans l'ensemble de données pour lesquelles le IoU est supérieur à ce seuil.

$$DT = P\ i \in S\ I(J(G, F) \geq T))\ |S$$

Où S est l'ensemble d'images dans l'ensemble de données et I est une fonction indicatrice .

Voici les résultats obtenus pour différentes seuils de l'indice de Jaccard.

**Tableau 3.2 :**Taux de détection pour différents seuils d'indice Jaccard

| D0.1 | D0.2 | D0.3 | D0.4 | D0.5 | D0.6 | D0.7 | D0.8 | D0.9 | D1 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 0.99 | 0.94 | 0.84 | 0.42 | 0 |

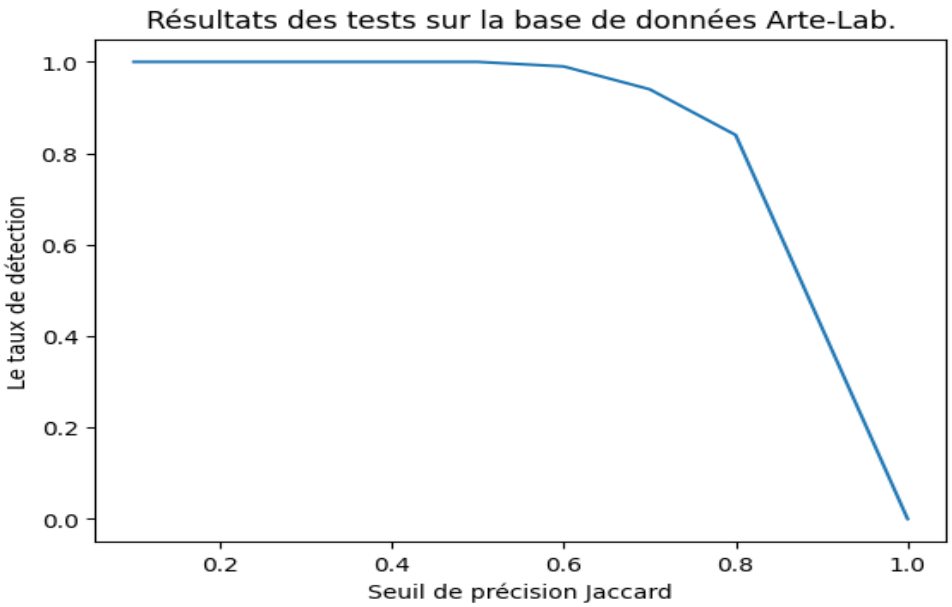



**Figure 3.22 :** Taux de détection pour différents seuils d'indice Jaccard

Nous comparons nos résultats avec Creusot Munawar 2015[19]et Hansen et al 2017[20].

**Tableau 3.3 :** Table de comparaison des résultats

| Approch | La moyenne JK | DETECTION RATE D0.5 |
|---|---|---|
| Creusot Munawar 2015 | 0.763 | 0.963 |
| Hansen et al 2017 | 0.816 | 0.991 |
| Nos résultats | 0.864 | 1 |

La moyenne du coefficient de Jaccard mesure le degré de chevauchement entre les résultats de détection obtenus par notre méthode et les vérités terrains. Notre méthode a affiché une moyenne du coefficient de Jaccard plus élevée, ce qui indique une meilleure correspondance.

De plus, un taux de détection de 0,5 signifie que notre méthode est capable de détecter la moitié des codes-barres présents, cette performance élevée est un indicateur clair de l'efficacité de notre approche.

Les résultats obtenus, mettent en évidence la performance atteint par notre méthode par rapport aux travaux précédents dans le domaine de la détection de codes-barres. Ils soulignent également la fiabilité et l'efficacité de notre approche pour la détection et localisation de codes-barres dans des scénarios réels.

## 3.8 Créations de l'interface graphique

La fin de notre projet, nous avons créé une interface graphique pour notre système de Détection et de décodage de codes-barres. Nous avons utilisé la bibliothèque Tkinter .Cette interface graphique offre une expérience utilisateur intuitive et permet aux utilisateurs d'interagir facilement avec notre système.

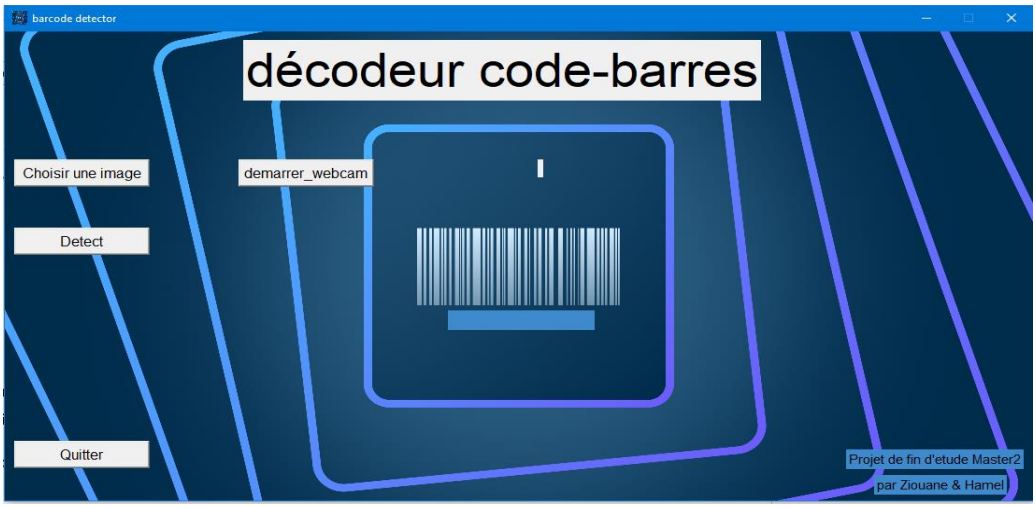



**Figure 3.23 :** Page d'accueil de L'interface graphique de notre system

L'interface graphique présente plusieurs fonctionnalités. Tout d'abord, elle permet aux utilisateurs de charger des images à partir de leur ordinateur. si on  click sur « Choisir une image » il faut choisis une photo sur le pc (voir figure 3.24 ).

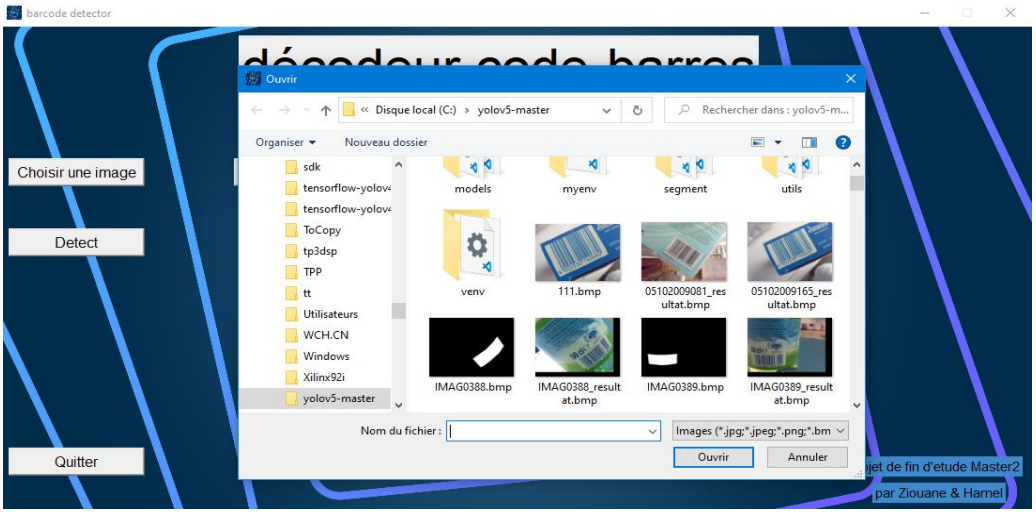



**Figure 3.24 :** Choisis une photo pour faire la détection

la 2eme option est la détection si on clique sur «Detect » notre system fait la détection et affiche la probabilité (voir figure 3.25).

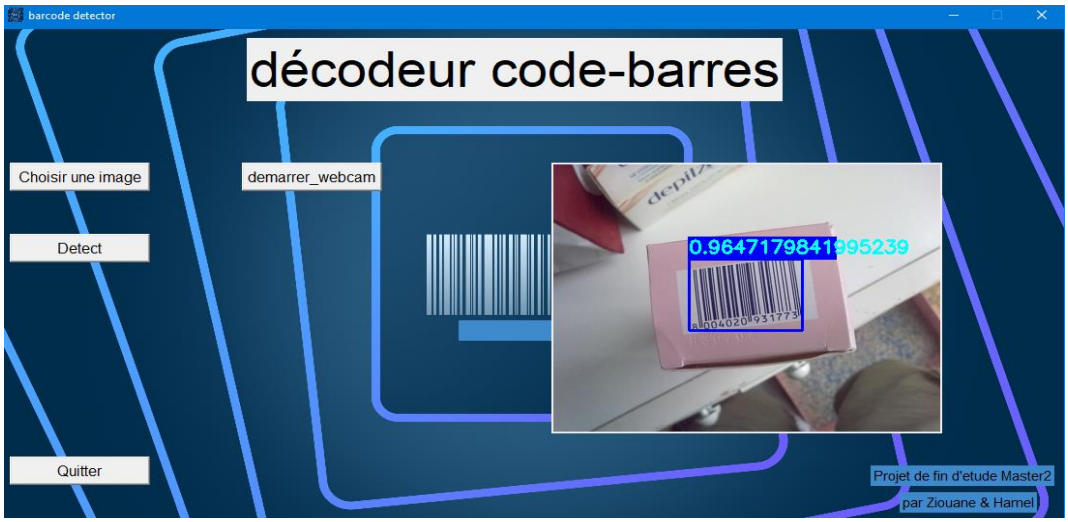



**Figure 3.25 :** résultat de détection en image

On peut faire aussi la détection en temps réel si en clique sur «demarrer_webcam »


**Figure 3.26 :** détection en temps réel

Finalement le bouton « Quitter » pour quitter le programme.

## 3.9 Conclusion:

Ce chapitre a décrit en détail l'implémentation de notre système de détection et de décodage de codes-barres. Nous avons présenté les différentes étapes de notre processus, de la préparation des données à l'évaluation des performances. Grâce à l'utilisation de techniques avancées telles que l'apprentissage en profondeur avec YOLOv5, l'estimation de l'angle et la rotation de l'image. Nous avons réussi à développer un système précis, fiable et efficace.

# Conclusion générale

Dans le cadre de ce projet, nous avons présenté le développement d'un système de détection et de décodage de codes-barres, pour objectif d'améliorer la précision et l'efficacité de cette technologie essentielle dans de nombreux domaines.

Nous avons utilisé l'architecture YOLOv5 pour l'apprentissage de notre modèle de détection de codes-barres. Nous avons également introduit deux procédures telles que ; l'estimation de l'orientation, la hauteur et la largeur du code à barres pour un décodage précis. La combinaison entre ces techniques (Yolov5 et traitement d'images) nous permis d'obtenir de très bons résultats du point de vue précision et robustesse.

Nous avons réalisé que la détection et le décodage des codes-barres sont des tâches complexes qui nécessitent une compréhension approfondie des techniques de vision par ordinateur et d'apprentissage automatique. Nous avons appris à traiter les images, à utiliser des algorithmes de décodage pour extraire les informations encodées dans les codes-barres. Nous avons également pris conscience de l'importance de la qualité des données d'entraînement, nous sommes optimistes au développement futur de notre travail.

Une direction possible pour l'amélioration de notre système tout d'abord, il faut enrichir la base de données de codes-barres en y ajoutant des échantillons tels que des codes-barres qui sont dominant par rapport au fond, une partie dédiée à la réduction du bruit. De plus, une partie de restauration des codes-barres endommagés.

# Bibliographie


1. Clerc, Sophie. "L'information et sa maîtrise : le cas du code-barres." Mémoire de DEA d'économie de l'industrie et des services, Université de Paris 1, 1991, 104 p.

2. MASDANIEL, Daniel. "Code-barres : Définition, Guide Complet & Solutions." Publié le 14 août 2020.

3. “Barcode Labeling - Getting Started - The Label Experts”. Consulté JUIN 2023.

barcode-labels.com/getting-started/barcodes/types/.

4. Parikh, Devi et al. "Localization and Segmentation of A 2D High Capacity Color Barcode." 2008 IEEE Workshop on Applications of Computer Vision.

5. "Camera Barcode Scanner Symbologies." Microsoft, 5 avril 2023, Consulté JUIN 2023.

learn.microsoft.com/en-us/windows/uwp/devices-sensors/pos-camerabarcode-symbologies.

6. Lin, S. and Wang, P. "Design of a barcode identification system." 2014 IEEE International Conference on Consumer Electronics - Taiwan, 2014. Consulté JUIN 2023

7. GeeksforGeeks. "R-CNN vs Fast R-CNN vs Faster R-CNN." GeeksforGeeks, 13 juillet 2017, Cnsulté JUIN 2023.

https://www.geeksforgeeks.org/r-cnn-vs-fast-r-cnn-vs-faster-r-cnn-ml/.

8. Hartiwi, Yessi et al. "Sistem Manajemen Absensi dengan Fitur Pengenalan Wajah dan GPS Menggunakan YOLO pada Platform Android." 2020,Consulté JUIN 2023.

9. Lin, Tsung-Yi et al. "Feature Pyramid Networks for Object Detection." 2017 IEEE Conference on Computer Vision and Pattern Recognition, 2016,Consulté JUIN 2023.

10. Creusot, C. and Munawar, A. "Real-time barcode detection in the wild." Proceedings of the 2015 IEEE Winter Conference on Applications of Computer Vision, Waikoloa, HI, USA, 5–9 January 2015, pp. 239–245.Consulté JUIN 2023.

11. Hansen, Daniel Kold, et al. "Real-Time Barcode Detection and Classification Using Deep Learning." Proceedings of the 9th International Joint Conference on Computational Intelligence - Volume 1: SCITEPRESS Digital Library, 2017, pp. 321-327.Consulté JUIN 2023

12. Xu, Yong et al. "An Effective Method of 1-D Bar Code Image Identification." 2013 IEEE 16th International Conference on Computational Science and Engineering, 2013,Consulté JUIN 2023.

13. Bevington, P. R. and Robinson, D. K. “Data Reduction and Error Analysis for the Physical Sciences”. McGraw-Hill Education, 2002.Consulté JUIN 2023.

14. "Uploadcare Documentation: Image Resize and Crop." Uploadcare, n.d., Consulté JUIN 2023.

uploadcare.com/docs/transformations/image/resize-crop.

15. Rosebrock, A. "OpenCV Color Spaces (cv2.cvtColor)." PyImageSearch, 28 avril 2021.Consulté JUIN 2023.

16. Fitriyah, Hurriyatul et al. "An Analysis of RGB, Hue and Grayscale under Various Illuminations." 2018 International Conference on Sustainable Information Engineering and Technology, 2018.Consulté JUIN 2023.

17. Andres, Eliot. “Deep Learning for Object Detection and Recognition”, 2020. Consulté JUIN 2023.

18. Aggarwal, Charu C. “Neural Networks and Deep Learning”. Determination Press, 2015. Consulté JUIN 2023.

19. Madan, P. and Madhavan, S. "An introduction to deep learning." IBM Developer, 2 mars 2020. Consulté JUIN 2023.

20. " Neural Network Basic Concepts." Neural Network 101,2023. Consulté JUIN 2023.

https://neuralnetwork101.com/

21. Krizhevsky, A., Sutskever, I. and Hinton, G. "Imagenet classification with deep convolutional neural networks." NIPS Conference, 2012, pp. 1097-1105.Consulté JUIN 2023.

22. Krizhevsky, A., Sutskever, I. and Hinton, G. E. "ImageNet Classification with Deep Convolutional Neural Networks." Advances in Neural Information Processing Systems 25, 2012. Consulté JUIN 2023

23. CS231n,”Convolutional Neural Networks for Visual Recognition”, Consulté JUIN 2023.

cs231n.github.io/convolutional-networks/

24. Punn, N. S. "Overview of Deep Learning Basics - I." Punn's Deep Learning Blog, 5 avril 2021.Consulté JUIN 2023.

25. Marc,P. “Réseaux de neurones“. Automne 2004, Université Laval.Consulté JUIN 2023.

26. Pat, N. “Neural networks and deep learning: deep learning explained to your granny”, 2018. Consulté JUIN 2023.

27. NVIDIA Developer. "Deep Learning Frameworks." NVIDIA Developer, 1 décembre 2022, Consulté JUIN 2023.

developer.nvidia.com/.

28. Bochkovskiy, A., Wang, C.-Y. and Liao, H.-Y.M. "YOLOv4: Optimal Speed and Accuracy of Object Detection." Institute of Information Science, Academia Sinica, Taiwan, 2020.Consulté JUIN 2023.

29. Karimi, G. "Introduction to YOLO Algorithm for Object Detection." 15 avril 2021. Consulté JUIN 2023

30. Chengwei. "Gentle guide on how YOLO Object Localization works with Keras (Part 2)." 2018. Consulté JUIN 2023

31. Rosebrock, A. "Intersection over Union (IoU) for object detection." 7 novembre 2016. Retrieved 30 April 2022. Consulté JUIN 2023

32. Solawetz, Jacob. "What is YOLOv5? A Guide for Beginners." Roboflow, 29 juin 2020. Consulté JUIN 2023

33. Maindola, G. "Introduction to YOLOv5 Object Detection." 21 juin 2021. Consulté JUIN 2023

34. Ultralytics. “YOLOv5: YOLOv5 in PyTorch. Version 5.0.0”, référentiel GitHub, 2021, Consulté JUIN 2023

https://github.com/ultralytics/yolov5.

35. Li, X. et al. "Oriented-YOLOv5: A Real-time Oriented Detector Based on YOLOv5." 2022 7th International Conference on Computer and Communication Systems, 2022, Consulté JUIN 2023.

36. Kong, Xuan et al. "Infrared Small Target Detection via Nonconvex Tensor Fibered Rank Approximation." IEEE Transactions on Geoscience and Remote Sensing, 2021, Consulté JUIN 2023.

37. "IoU (Intersection over Union)." Hasty, 17 Mai 2023.cosulté JUIN 2023. hasty.ai/docs/mp-wiki/metrics/iou-intersection-over-union.

.

38. Solawetz, J. "Qu'est-ce que la précision moyenne moyenne (mAP) dans la détection d'objets ?" 6 mai 2020. Retrieved 5 juin 2023.Consulté JUIN 2023.

39. Kukil. "Mean Average Precision (mAP) in Object Detection." 9 août 2022.Consulté JUIN 2023.

40. Zenggyu. "An Introduction to Evaluation Metrics for Object Detection." 16 décembre 2018.Consulté JUIN 2023.

41."FAQ-Colaboratory."Google Research, Consulté JUIN 2023.

research.google.com/colaboratory/faq.html.

42. matthewbrems. "Roboflow." GitHub, 2021, Consulté JUIN 2023.

github.com/opencv-ai/roboflow

43. "À propos d'OpenCV." OpenCV.org, consulté Juin 2023.

opencv.org/about/

44. "Pyzbar." PyPI, consulté JUIN 2023.

pypi.org/project/pyzbar.

45. “Torch”. Towards Data Science, consulté JUIN 2023. towardsdatascience.com/introduction-to-py-torch-.

46. "Introduction à NumPy." w3schools.com, consulté Juin 2023.

47. "tkinter - Interface Python pour Tcl/Tk." La bibliothèque standard - Documentation, Mis à jour le 16 juil. 2023. consulté Juin 2023.

docs.python.org/fr/3/library/tkinter.html

48. Zamberletti, A., Gallo, I. et Albertini, S. "Robust angle invariant 1d barcode detection." Proceedings of the 2013 2nd IAPR Asian Conference on Pattern Recognition, 5-8 novembre 2013, Naha, Japon, pp. 160-164.Consulté JUIN 2023.

49. Xiao, Yunzhe et Ming, Zhong. "1D Barcode Detection via Integrated Deep-Learning and Geometric Approach." Reçu le 29 juin 2019 ; Accepté le 6 août 2019 ; Publié le 9 août 2019.