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

Département d'Automatique, Télécommunication et d'Électronique

PROJET DE FIN D'ÉTUDES

Pour l'obtention du diplôme de Master

Filière : Automatique

Spécialité : Automatique et Informatique Industrielle

T H È M E

Détection d'objets d'intérieur par apprentissage profond

PRÉPARÉ PAR

- **ASSEF** Yanis
- **HANANE** Hicham

DIRIGÉ PAR

- Mr. **MENDIL** Boubekeur
- Mlle. **TENNICHE** Nesrine

EXAMINÉ PAR

- Mr. **SADJI** Moustapha
- Mr. **TIGHZERT** Lyes

ANNÉE UNIVERSITAIRE

2022/2023

DATE DE SOUTENANCE

Le : …/06/2023

REMERCIEMENTS

*Nous souhaitons exprimer notre profonde gratitude envers nos promoteurs, **Mr. MENDIL**, professeur en Automatique, et **Mlle. TENNICHE**, pour leur précieuse assistance, leurs conseils avisés et leur disponibilité tout au long de notre projet de fin d'études.*

*Nous tenons à remercier chaleureusement **Mr. SADJI** Moustapha, président du jury, d'avoir accepté de présider notre soutenance et de nous accorder cet honneur.*

*Nos sincères remerciements vont également à **Mr. TIGHZERT**, d'avoir accepté d'évaluer notre modeste travail.*

Nous voulons exprimer notre profonde reconnaissance envers nos chers parents pour leur soutien inconditionnel et les sacrifices consentis tout au long de notre parcours académique.

Enfin, nous souhaitons adresser nos sincères remerciements à nos amis et à toutes les personnes qui ont contribué à la réalisation de ce travail de fin d'études.

DÉDICACES

Je tiens tout d'abord à exprimer ma gratitude envers le Tout-Puissant, Dieu ALLAH, sans qui ce travail n'aurait pu voir le jour.

Je souhaite dédier humblement ce modeste travail aux personnes qui ont joué un rôle essentiel dans ma vie

À ma chère Maman, une âme unique et incomparable. Je souhaite la remercier chaleureusement pour ses innombrables sacrifices, sa compréhension sans faille et ses conseils précieux, qui ont été un soutien constant tout au long de ce parcours.

Je tiens également à exprimer un profond respect envers mon père, dont le dévouement total m'a guidé et inspiré.

À mon frère bien-aimé, Nacer, qui a été une source constante de motivation et de soutien.

À mes amis fidèles : Tarik, Amine, N'fissa, Mustapha, Rayane, Adel, Yacine, Yanis, Ryad, Kenza et Ziko, dont la présence et les encouragements m'ont poussé à aller de l'avant.

Je souhaite également dédier ce travail à mon binôme Hicham et à sa famille, qui ont été des compagnons de route précieux tout au long de ce projet

Enfin, je souhaite rendre hommage à toutes les années d'études, Je souhaite tout particulièrement rendre hommage à Mlle. TENNICHE Nesrine, qui a été un véritable pilier pendant notre projet. Sa présence, ses conseils précieux et son dévouement ont été essentiels à notre réussite.

À tous ceux mentionnés ici, je vous exprime ma reconnaissance infinie et vous dédie ce travail avec une profonde sincérité.

Yanis

DÉDICACES

À mes chers parents, je ne saurais exprimer suffisamment ma gratitude pour vos sacrifices et votre soutien inconditionnel tout au long de mon parcours. Votre amour et votre dévouement ont rendu possible ce travail. Je vous remercie du fond du cœur et espère pouvoir un jour vous rendre ce que vous avez fait pour moi. Que Dieu vous comble de bonheur et de longévité.

À mon frère, Adam.

À mes cousins Rachid, Redha, Mohamed, Aimen, Amine, Nassim.

À mes chers amis, Mohand, Zinou, Rayane, Wassim, Massensen, Yacine, Ryad, Moumouh, Mehrez et Zizou. Je dédie ce travail avec reconnaissance. Votre amitié et votre soutien ont été inestimables. Nous avons partagé des moments précieux qui m'ont encouragé tout au long de cette étape importante de ma vie.

Je tiens également à exprimer ma profonde gratitude envers mon binôme, Yanis, pour son sérieux et sa compréhension tout au long de la réalisation de ce modeste travail. Sa précieuse contribution a été d'une importance capitale, et je suis reconnaissant de pouvoir compter sur son aide et son implication.

Hicham

TABLE DES MATIÈRES

SOMMAIRE

TABLE DES MATIÈRES

Chapitre IV : Entrainement de YOLOv4 sur la base de données YAHINE

LISTE D'ABRÉVIATIONS

IA: Intelligence Artificielle.

DL: Deep learning.

CNN: Convolutional Neural Network.

FC: Fully connected.

RNN: Recurrent Neural Network.

AE: Auto encodeurs.

GAN: Generative Adversarial Networks.

R-CNN: Regions with Convolutional Neural Network.

SVM: Support Vector Machine.

ROIs: Regions of Interest.

HOG: Histogram of Oriented Gradients.

SSD: Single Shot Detector.

VGG-16: Visual Geometry Group 16.

IoU: Intersection over Union.

CSP: Cross Stage Partial.

SPP: Spatial Pyramid Pooling.

PAN: Path Aggregation Network.

RPN: Region Proposal Network.

E-RPN: Efficient Region Proposal Network.

GPU: Graphic Processing Unit.

CPU: Central Processing Unit.

MSCOCO: Microsoft Common Objects in Context.

Pascal VOC: Pascal Visual Object Classes.

KITTI: Karlsruhe Institute of Technology and Toyota Technological Institute.

SUN: Scene Understanding.

CUDA: Compute Unified Device Architecture.

LIBSO: Library Source.

OpenCV: Open-Source Computer Vision Library.

RVB : Rouge Vert Bleu.

MAP : Mean Average Pooling.

INTRODUCTION GÉNÉRALE

Grace aux progrès réalisés dans les domaines de la technologique et de l'informatique au fil des années, les algorithmes d'IA ont connu des avancées remarquables. Ces derniers ont révolutionné de nombreux domaines tels que la médecine, les services financiers, les jeux vidéo, les chatbots, les assistants vocaux et bien d'autres. Les algorithmes d'IA sont de plus en plus sophistiqués et sont utilisés pour résoudre des problèmes complexes qui étaient auparavant considérés comme insolubles par les méthodes traditionnelles [1].

Face à la croissance exponentielle de la demande de technologies intelligentes, notamment la reconnaissance et la détection d'objets, il est devenu crucial de développer des méthodes plus précises et plus rapides, indépendantes et capables de percevoir différents modèles de données. [1].

L'apprentissage automatique, qui consiste en la capacité des systèmes informatiques à trouver des solutions de manière autonome, est un domaine clé de l'IA, notamment avec l'utilisation de l'apprentissage profond, qui simule le fonctionnement des neurones du corps humain [2].

Notre travail s'inscrit dans ce contexte. Il s'agit d'exploiter les paradigmes de l'apprentissage profond pour résoudre un problème du quotidien, à savoir retrouver certains objets d'intérieur souvent égarés.

Pour ce faire, une nouvelle base de données a été créé. Celle-ci contient autour de 5000 images de cinq objets : téléphone, lunette, portefeuille, clés et télécommande. Cette conception est basée sur certaines étapes clés telles que l'acquisition des données dans des conditions diverses, la mise en forme et les annotations.

La suite du travail concerne le problème de détection dans des conditions réelles (diversité des formes des objets, conditions d'éclairage, distance, etc). L'apprentissage profond s'est présenté comme une solution naturelle à ce problème, vue ces capacités d'apprentissage et de généralisation. Des outils et des ressources logicielles open source ont été exploités pour la conception de notre modèle.

Le mémoire est structuré en trois chapitres qui ont pour but de présenter de manière progressive les fondements nécessaires centrés sur la détection d'objets, des architectures jusqu'au fonctionnement en profondeur.

- Le premier chapitre porte une introduction générale à l'IA et à l'apprentissage automatique, suivi d'une exploration approfondie de l'apprentissage profond. Nous aborderons en détail le fonctionnement de l'apprentissage profond, en mettant l'accent sur les réseaux de neurones convolutifs. De plus, nous examinerons l'application de l'apprentissage profond en vision artificielle, qui constitue l'objectif principal de notre travail.

- Le deuxième chapitre est dédié à l'étude approfondie des méthodes basées sur l'apprentissage profond. En commençant par retracer l'évolution de ces dernières et en présentant les fondements théoriques qui les sous-tendent. Ensuite, nous allons examiner les méthodes les plus populaires et les plus couramment utilisées pour la détection d'objets, en détaillant leurs architectures respectives, avec un accent sur YOLOv4, qui sera la méthode principale utilisée dans notre travail durant la partie conception.

- Le troisième chapitre a été consacré à l'exploration des bases de données pour la détection d'objets. Après avoir examiné les concepts fondamentaux, nous avons développé notre propre base de données appelée YAHINE, qui comprend une variété d'images représentant différentes catégories d'objets d'intérieur.

- Dans le quatrième chapitre, On a entamé l'entraînement d'un modèle pré-entrainé YOLOv4 sur notre base de données, réalisant ainsi deux expériences distinctes : une avec la base YAHINE directement et l'autre avec augmentation de données. Nous avons évalué les performances de détection en utilisant des mesures telles que la précision, le rappel et le score F1. Ce qui nous a permis de comparer les résultats obtenus par les deux approches et de formuler des suggestions d'améliorations potentielles pour des travaux futurs.

CHAPITRE 1

INTRODUCTION À L'APPRENTISSAGE PROFOND

Récapitulatif

Dans ce chapitre, nous allons définir d'une façon générale l'IA ainsi que l'apprentissage automatique, puis nous entrerons dans le cœur de ce dernier avec l'apprentissage profond tout en expliquant son fonctionnement, les réseaux de neurones convolutifs et l'application de l'apprentissage profond en vision artificielle qui est l'objectif de notre travail.

I.1. Introduction

De nos jours, L'IA a considérablement amélioré le paradigme et le spectre de la recherche avec une promesse d'applicabilité continue [1]. L'IA, la force motrice de la révolution technologique actuelle, reste le cas classique de toute démarche visant à apprendre à une machine des tâches humaines.

En exploitant les capacités de l'apprentissage profond, qui est une branche de l'IA, la vision artificielle a considérablement progressé ces dernières années, en particulier dans la détection d'objets, la reconnaissance de formes et la segmentation d'images. Cette évolution a permis la mise en place de multiples applications pratiques, telles que la reconnaissance faciale ou la reconnaissance d'objets comme la reconnaissance des plaques d'immatriculations, etc.

Dans ce chapitre, nous allons définir d'une façon générale l'IA ainsi que l'apprentissage automatique, puis entrer dans le cœur de ce dernier avec l'apprentissage profond tout en expliquant son fonctionnement, les réseaux de neurones convolutifs et l'application de l'apprentissage profond en vision artificielle qui est l'objectif de notre travail.

I.2. Intelligence artificielle

L'IA est un domaine qui traite de la conception, le développement et la mise en œuvre de divers systèmes capables d'imiter l'intelligence humaine [2]. Cette discipline repose sur la conception et la mise en œuvre d'algorithmes et de modèles mathématiques qui permettent à des systèmes informatiques de simuler des processus cognitifs comme la perception, la compréhension, l'apprentissage, le raisonnement et la prise de décision.

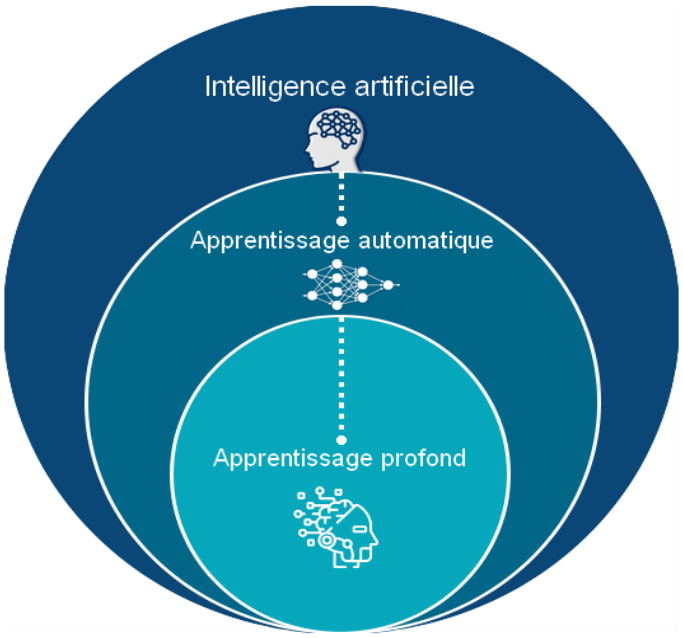


Figure I.1 : Schéma de la hiérarchie de l'IA

I.3. Apprentissage automatique

I.3.1. Définition

L'apprentissage automatique (machine learning en anglais), est une branche de l'IA qui permet à l'ordinateur d'apprendre à partir de données. Contrairement à la programmation classique où l'humain donne des instructions précises à l'ordinateur, l'apprentissage automatique utilise des algorithmes pour apprendre à partir d'exemples de données.

I.3.2. Processus d'apprentissage en apprentissage automatique

Le processus d'apprentissage consiste à entraîner un modèle mathématique sur des données d'entrée, afin de lui permettre de réaliser une tâche donnée. Par exemple, un modèle d'apprentissage automatique peut apprendre à identifier des images de chats à partir d'un ensemble de données d'images étiquetées comme présenté dans la figure I.2.

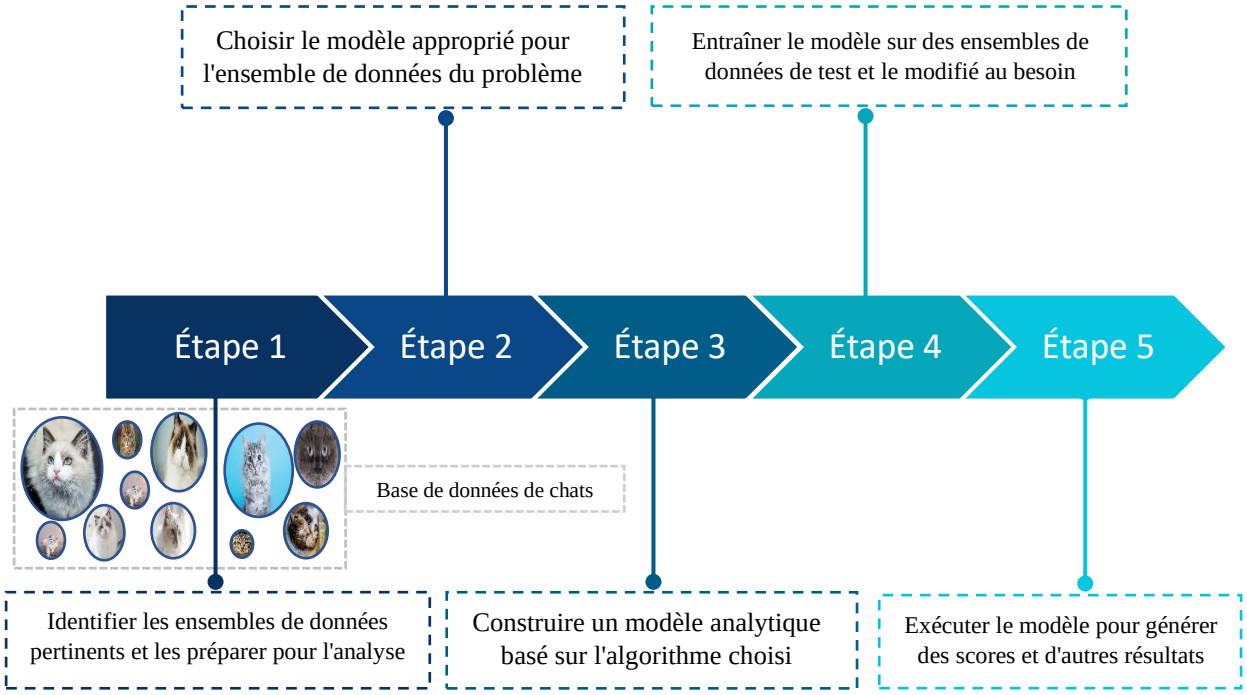


Figure I.2 : Illustration du processus d'apprentissage automatique

I.3.3. Différents types d'algorithmes de l'apprentissage automatique

L'apprentissage automatique comprend différents types d'algorithmes, tels que la régression linéaire, les machines à vecteurs de support (SVM), la répartition en K-moyennes (K-means), les arbres de décision [3], tel qu'illustré dans la figure (I.3).

Les algorithmes sont choisis en fonction de la tâche à accomplir et des caractéristiques des données d'entrée.

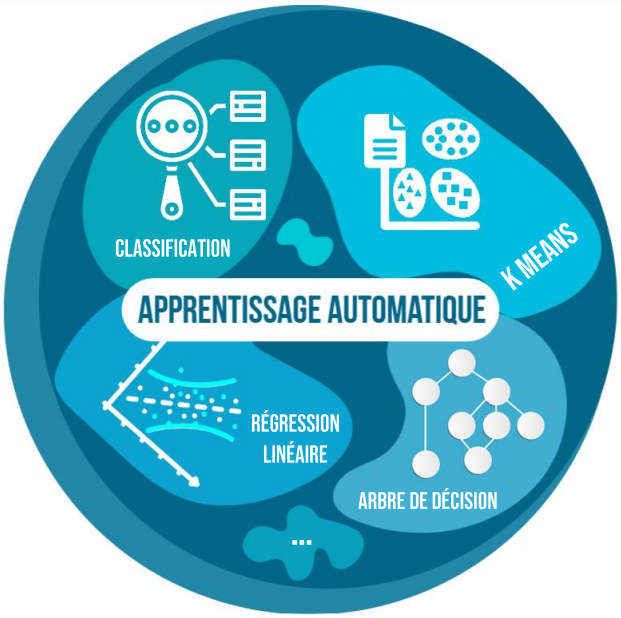


Figure I.3 : Types d'algorithmes de l'apprentissage automatique

I.4. Apprentissage profond

I.4.1. Définition

L'apprentissage profond (DL en anglais), une branche prédominante de l'IA, a été étendu avec des structures de réseau diversifiées et qui utilise des réseaux de neurones artificiels profonds pour apprendre à partir de données [4]. Contrairement aux réseaux de neurones peu profonds, qui ont une ou deux couches cachées, les réseaux de neurones profonds ont plusieurs couches cachées, ce qui leur permet d'apprendre des caractéristiques complexes à partir de données brutes.

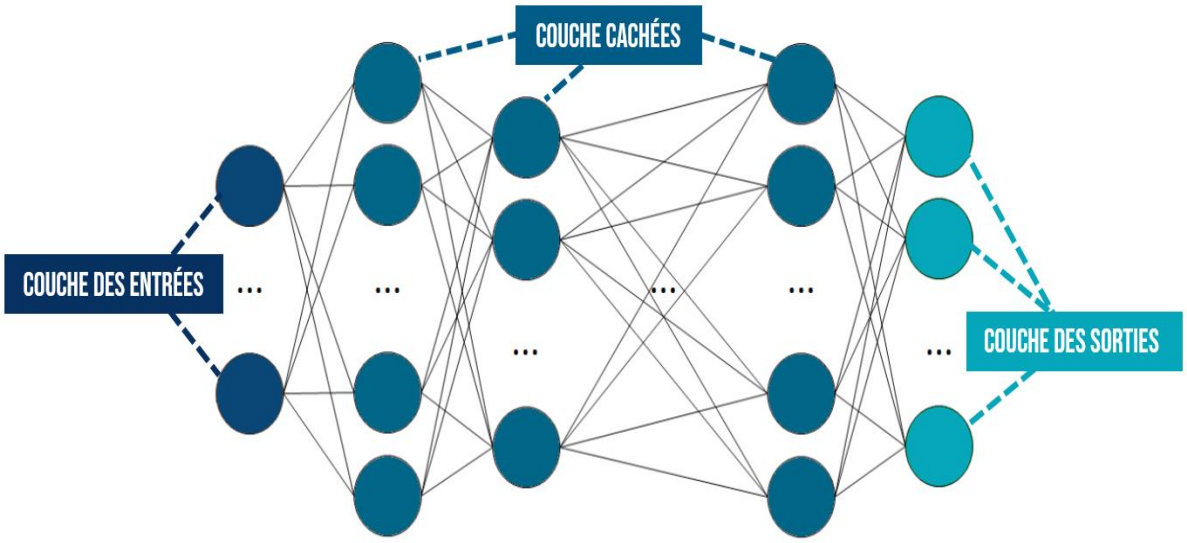


Figure I.4 : Réseau de neurone profond

I.4.2. Fonctionnement de l'apprentissage profond

L'apprentissage profond repose sur l'utilisation de réseaux de neurones artificiels profonds, qui sont composés de plusieurs couches de neurones interconnectées. Ces réseaux sont capables d'apprendre à partir de données brutes en effectuant des calculs sur les connexions entre les neurones.

Le processus d'apprentissage consiste à ajuster les poids des connexions entre les neurones pour minimiser l'erreur de prédiction. Cela se fait par rétropropagation de l'erreur, où l'erreur est propagée de la sortie du réseau vers l'entrée, en ajustant les poids des connexions à chaque étape.

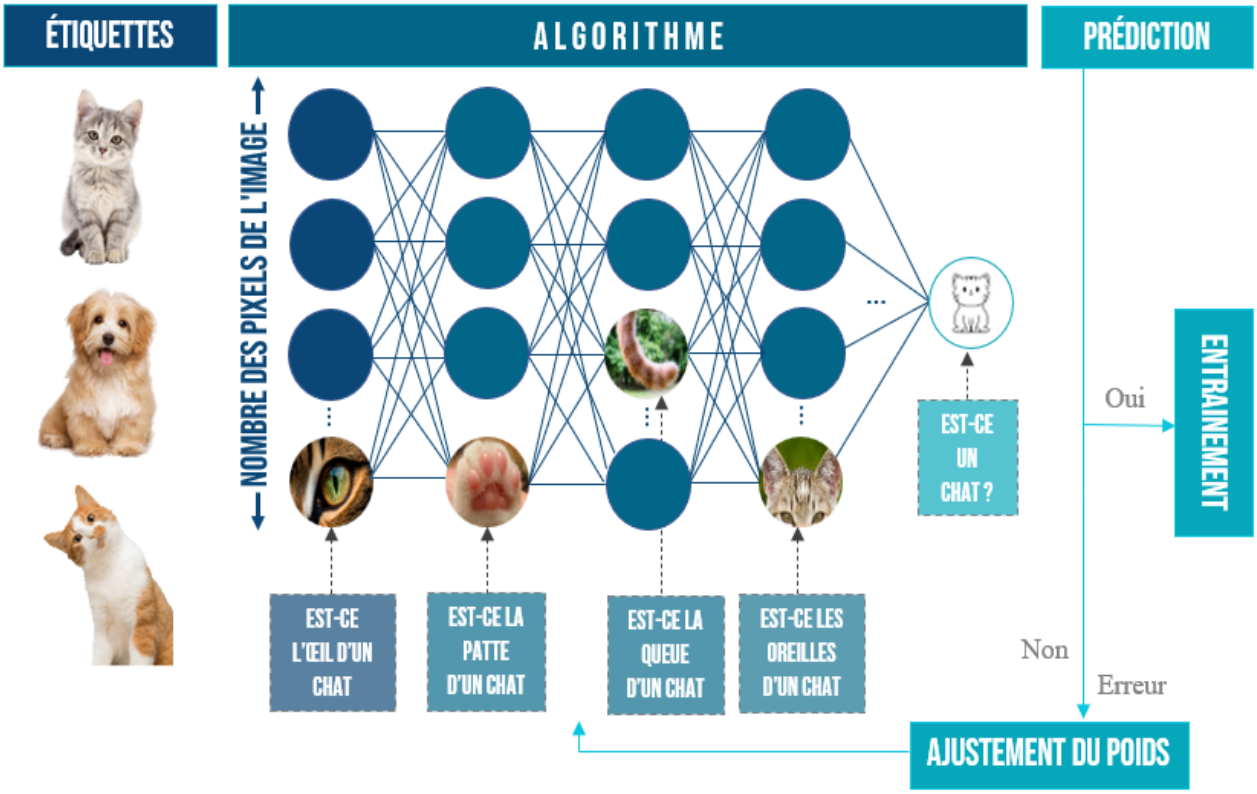


Figure I.5 : Fonctionnement de l'apprentissage profond

I.4.3. Types des réseaux neuronaux

Il existe plusieurs types de réseaux de neurones qui sont utilisés en apprentissage profond, chacun avec ses propres avantages et limites en fonction du type de problème à résoudre, les principaux types de réseaux neuronaux sont :

A) Les réseaux de neurones denses (FC).
B) Les réseaux de neurones récurrents (RNN).
C) Les réseaux de neurones convolutifs (CNN).
D) Les réseaux de neurones autoencodeurs (AE).
E) Les réseaux de neurones adversariaux (GAN).

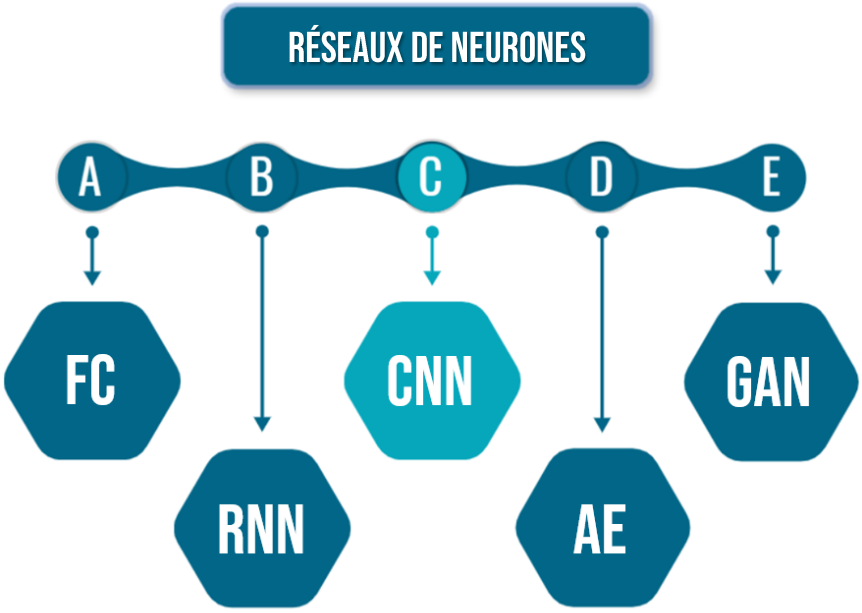


Figure I.6 : Types de réseaux de neurones de l'apprentissage profond

I.5. Réseaux de neurones convolutifs

Les réseaux de neurones convolutifs (CNN en anglais) sont une architecture de réseau de neurones spécialement conçue pour le traitement d'images et de vidéos. Les CNN sont capables d'extraire des caractéristiques importantes à partir des images, en utilisant une technique appelée convolution.

Les CNN sont composés de plusieurs couches de neurones, chacune effectuant une opération spécifique sur les données d'entrée. Les couches typiques d'un réseau de neurones convolutifs comprennent :

I.5.1. Couche de convolution

La couche de convolution est la pierre angulaire du CNN. Elle comporte la majeure partie de la charge de calcul du réseau. La convolution est une opération mathématique est d'appliquer un filtre à une image pour extraire ses caractéristiques telles que les bords, les coins et les textures.

Les filtres utilisés sont des matrices de nombres qui glissent sur l'entrée de la couche précédente, multipliant et additionnant les valeurs de l'entrée qui se trouvent sous chaque position du filtre en se déplaçant de gauche à droite et de haut en bas. Les valeurs résultantes sont ensuite stockées dans une carte de caractéristiques.

Si on prend une image d'entrée d'une taille $W \times W \times D$, La taille du volume de la sortie peut être directement calculer avec cette formule :

$$Wout = \frac{W-F+2P}{S} + 1 \qquad \text{(I.1)}$$

D est le nombre de canaux, F représente la taille du filtre, S est le pas de ce dernier et le paramètre P également appelé remplissage ou padding correspond à un nombre de lignes et de colonnes de pixels dont les valeurs sont fixées à zéro et qui sont ajoutées autour de l'image. Les figures (I.7) et (I.8) illustrent respectivement un exemple du processus de remplissage et de l'opération de convolution.

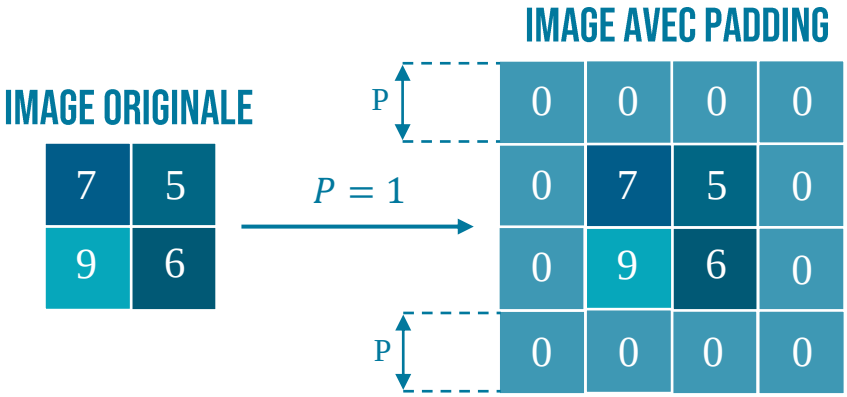


Figure I.7 : Exemple de padding d'une matrice 2x2

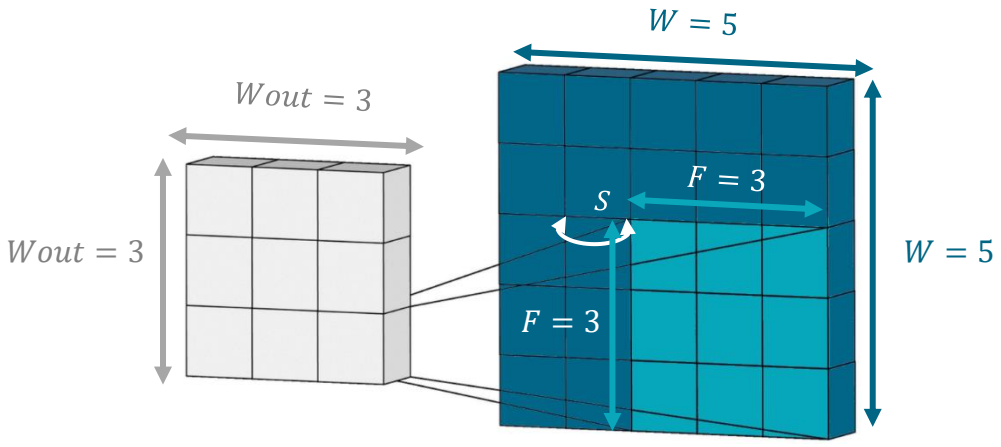


Figure I.8 : Illustration d'une opération de convolution avec un S = 1 & $P = 0$

I.5.2. Couche de pooling

La couche de convolution est suivie par une couche de pooling. Le but principal de cette couche est de réduire le nombre de paramètres et de calculs nécessaires dans le réseau, tout en conservant les caractéristiques les plus importantes de l'image en utilisant les opérations les plus courantes à savoir :

- Max Pooling : Prendre la valeur maximum de chaque région de la carte.
- Average Pooling (Mean Pooling) : Prendre la valeur moyenne de chaque région de la carte.

Si on prend une image d'entrée d'une taille $W \times W \times D$, On peut alors directement utiliser la formule afin de calculer la dimension de sa sortie :

$$Wout = \frac{W-F}{S} + 1 \qquad \text{(I.2)}$$

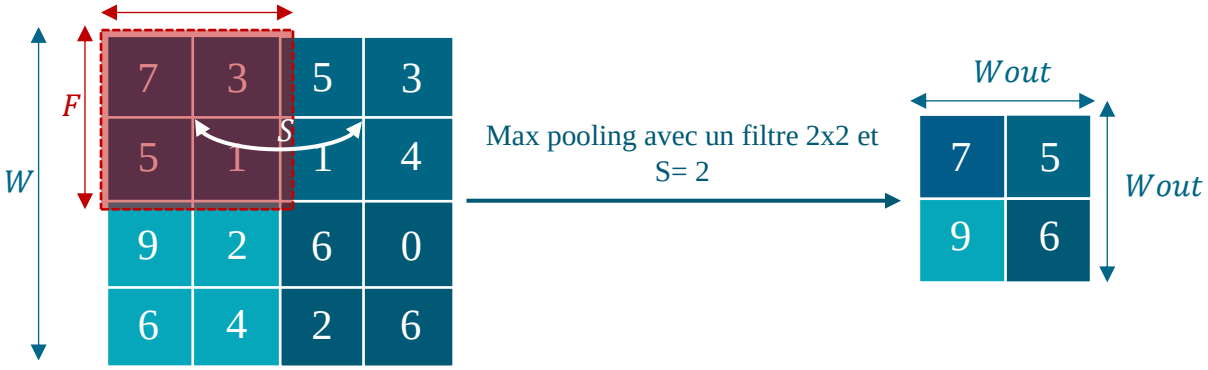


Figure I.9 : Opération de pooling d'une image 4x4 utilisant max pooling 2x2

I.5.3. Couche entièrement connectée

La couche entièrement connectée, également appelée couche de classification, c'est une couche qui relie toutes les sorties de la couche précédente à chaque entrée de la couche suivante, comme dans un réseau de neurones classique. Cette couche est souvent placée à la fin du réseau de neurones convolutifs pour effectuer la classification des caractéristiques extraites des images par les couches convolutives et les couches de Pooling.

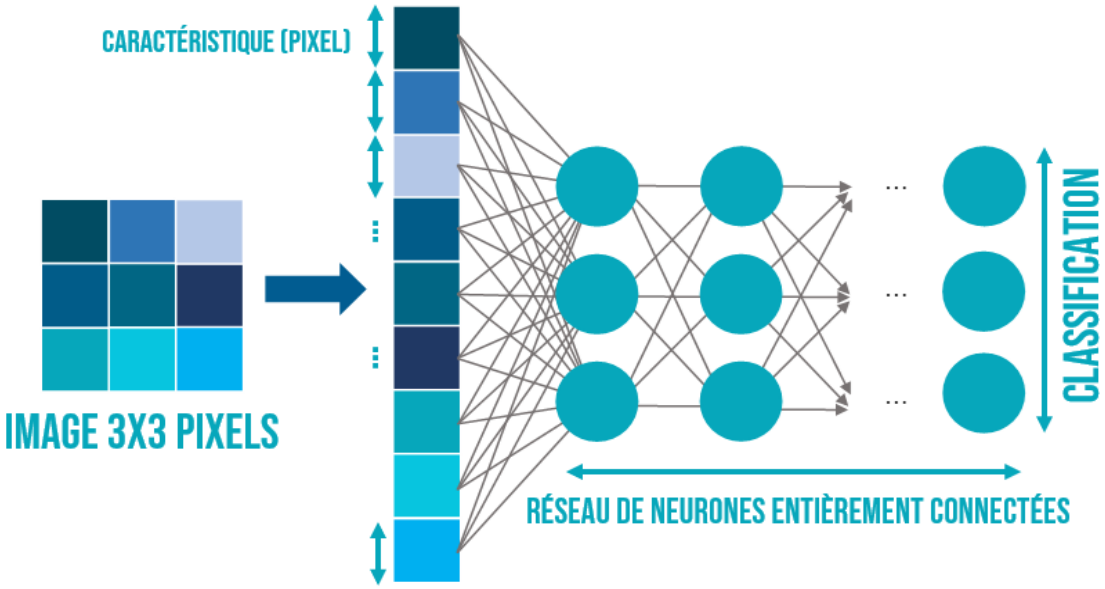


Figure I.10 : Constitution d'une couche entièrement connectée

Elle est constituée de neurones entièrement connectés, c'est-à-dire que chaque neurone de la couche précédente est connecté à chaque neurone de la couche qui la succède. Cette couche utilise généralement une fonction d'activation Softmax pour calculer les probabilités de chaque classe d'appartenance pour une entrée donnée, la sortie prédite quant à elle peut être comparée à la

vérité terrain pour calculer l'erreur de prédiction et ajuster les poids des neurones dans le réseau de neurones à l'aide de l'algorithme de rétropropagation du gradient afin d'améliorer les performances de classification.

Après avoir combiné toutes les couches du CNN, le fonctionnement final est illustré dans la figure (I.11).

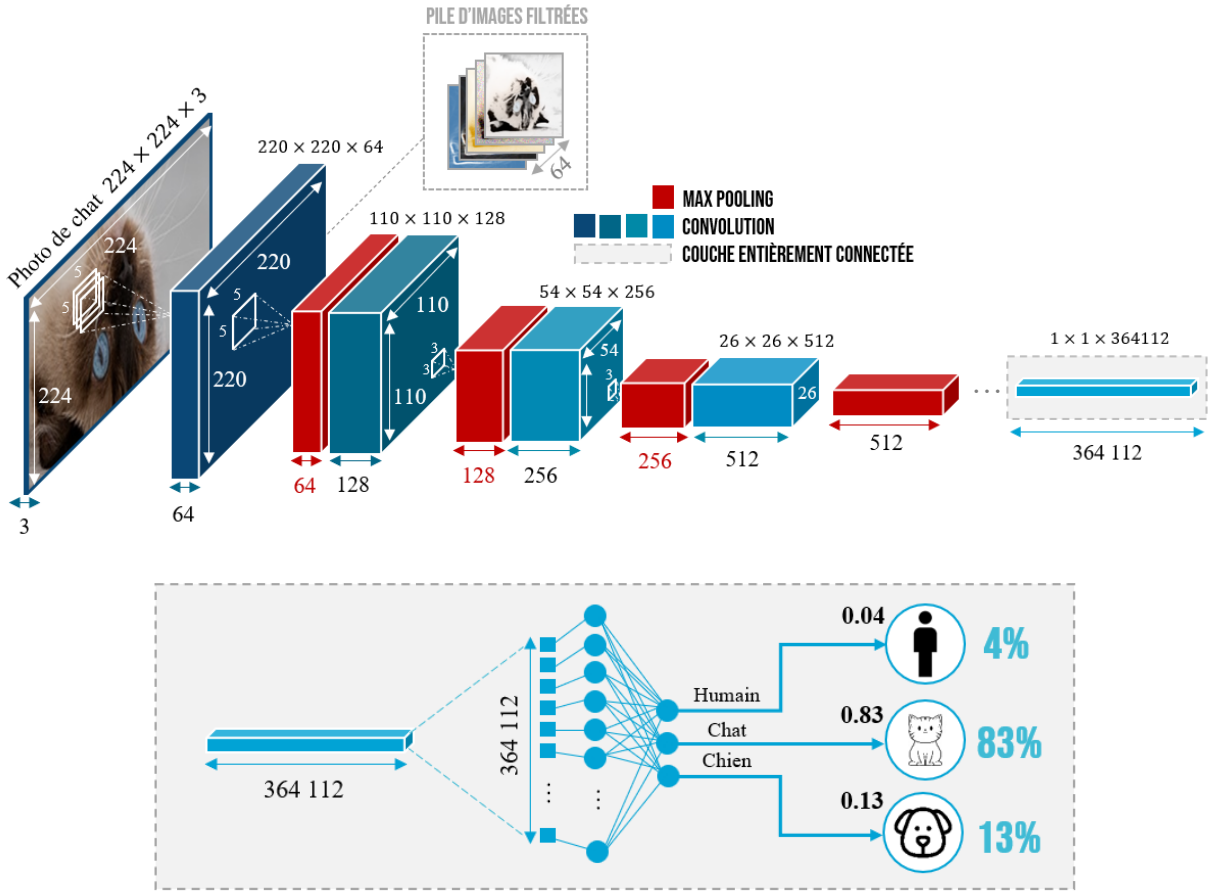


Figure I.11 : Fonctionnement d'un CNN

I.5.4 Avantages et applications du CNN

Les avantages des CNN sont leur capacité à apprendre automatiquement des représentations hiérarchiques de caractéristiques à partir des données, leur architecture profonde qui permet une meilleure expressivité, la possibilité d'optimiser plusieurs tâches liées simultanément et leur grande capacité d'apprentissage pour résoudre des problèmes de vision artificielles complexes. En d'autres termes, les CNN offrent une approche plus efficace et puissante pour l'analyse et la compréhension d'images que les méthodes traditionnelles [5].

En raison de ces avantages, les CNN ont été largement utilisés dans de nombreux domaines de recherche tels que la reconstruction de super-résolution d'images, la classification d'images, la recherche d'images, la reconnaissance faciale, la détection de piétons et l'analyse

vidéo. Les CNN sont donc des outils polyvalents pour de nombreuses tâches de traitement d'images [6].

I.6. Vision Artificielle

I.6.1. Définition

La vision artificielle, également appelée vision par ordinateur, est une branche essentielle de l'IA qui vise à concevoir des systèmes capables d'analyser et d'interpréter des images et des vidéos numériques à l'aide du filtrage. Ces systèmes utilisent des algorithmes et des modèles mathématiques pour extraire des informations pertinentes à partir des données visuelles. La vision artificielle permet ainsi de remplacer l'œil et le cerveau humain pour l'observation et le jugement, offrant ainsi de nombreuses possibilités pour la recherche et l'industrie [7].

I.6.2. Filtrage en vision artificielle

Le filtrage est une méthode essentielle en vision artificielle qui permet d'améliorer la qualité des images en réduisant les effets indésirables tels que le bruit, les distorsions et les artefacts. Il est couramment utilisé pour prétraiter les images avant d'effectuer des opérations de traitement d'images plus avancées telles que la détection d'objets, la segmentation d'images, la reconnaissance de formes, etc.

En utilisant différentes techniques de filtrage, les images peuvent être améliorées en augmentant la netteté, en renforçant les contours, en améliorant la luminosité et en réduisant le flou.

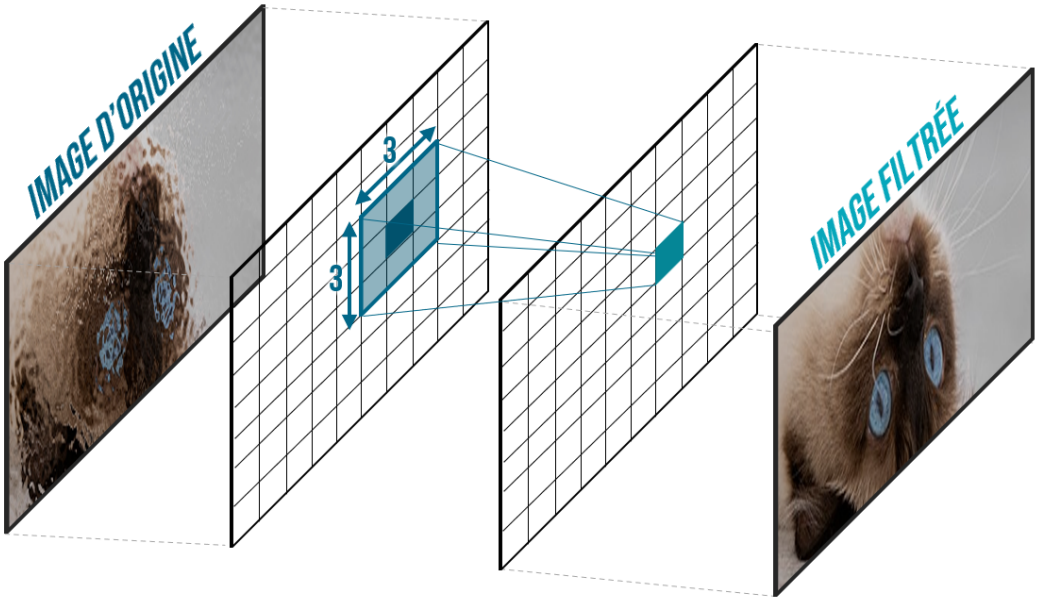


Figure I.12 : Filtrage d'image en utilisant la réduction de bruit

Le filtrage est également important pour faciliter l'analyse et la compréhension des images par les machines. En effet, le bruit et les distorsions peuvent rendre la tâche difficile pour les algorithmes de vision artificielle, car ils peuvent confondre les détails importants avec le bruit et les artefacts. En éliminant le bruit et en améliorant la qualité de l'image, les algorithmes de vision artificielle peuvent mieux détecter les objets, les formes et les structures dans les images, ce qui facilite leur analyse et leur compréhension.

I.6.3. Vision artificielle dans l'apprentissage profond

La vision artificielle est une application clé de l'apprentissage profond, qui permet aux réseaux de neurones de s'entraîner à partir de données visuelles telles que des images ou des vidéos. Les techniques de vision artificielle, telles que la classification, la détection d'objets et la segmentation, sont souvent utilisées comme tâches d'entraînement pour les réseaux de neurones profonds, leur permettant d'apprendre des caractéristiques visuelles hiérarchiques complexes. Cela permet de créer des systèmes intelligents capables de comprendre et d'interpréter des données visuelles, offrant de nombreuses applications dans des domaines tels que l'imagerie médicale, la reconnaissance faciale, la conduite autonome, etc.

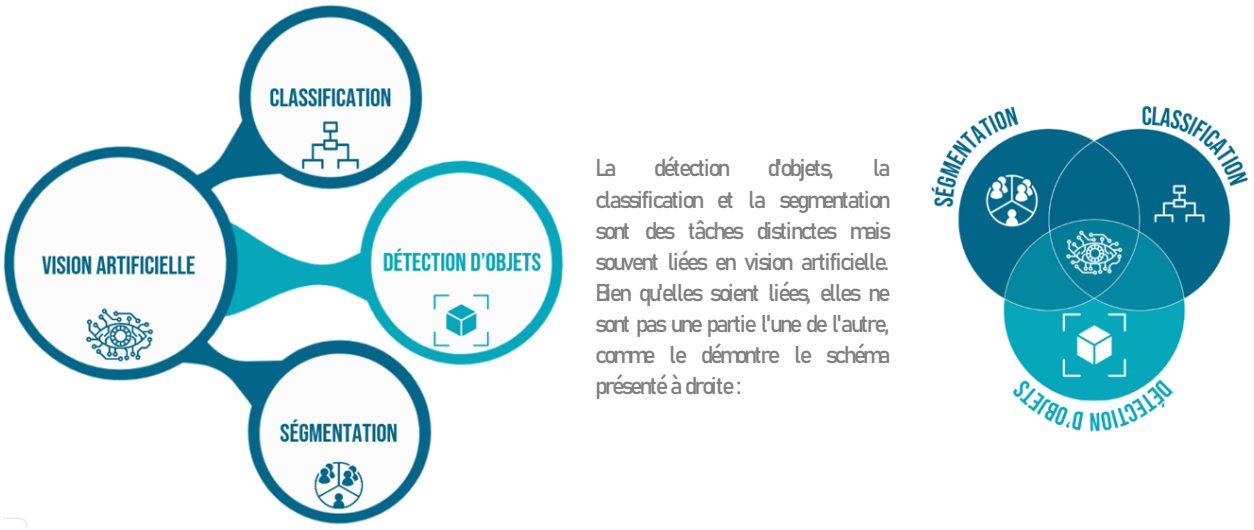


Figure I.13 : Techniques de la vision artificielle dans l'apprentissage profond

I.7. Conclusion

Ce chapitre nous a permis d'établir les fondements de l'IA et de l'apprentissage automatique, en se concentrant sur l'apprentissage profond et son utilisation courante dans la vision artificielle. Nous avons également mis en évidence l'importance des réseaux de neurones convolutifs, qui sont largement utilisés dans la vision artificielle, Avec un accent sur la détection d'objets, qui est l'objectif principal de notre travail. En combinant toutes ces notions, nous avons établi une base solide pour la poursuite de notre travail sur la détection d'objets en utilisant l'apprentissage profond dans la vision artificielle.

CHAPITRE 2

DÉTECTION D'OBJETS

Récapitulatif

Ce chapitre sera dédié à l'étude approfondie des méthodes basées sur l'apprentissage profond. En commençant par retracer l'évolution de ces dernières et en présentant les fondements théoriques qui les sous-tendent. Ensuite, On examinera les approches les plus répandues et couramment employées pour la détection d'objets, en mettant l'accent sur YOLOv4, qui constituera la méthode principale utilisée dans notre travail lors de la phase d'entrainement et de test.

II.1. Introduction

La détection d'objets demeure un des sujets les plus complexes dans le domaine de la vision par ordinateur. Ce domaine fait face à de nombreux défis tels que la variabilité des objets, les changements de posture, la présence d'occlusions, la variation d'éclairage et la diversité des environnements, etc. Au cours du siècle dernier, des algorithmes classiques de traitement de données ont été développés pour la détection d'objets, mais ont été peu à peu remplacés par des algorithmes intégrant l'apprentissage en profondeur basés sur des réseaux neuronaux convolutifs [8]. Ce chapitre se propose de passer en revue les fondamentaux de la détection d'objets, ses dernières avancées technologiques et les algorithmes utilisés.

II.2. Principes fondamentaux de la détection d'objets

II.2.1. Définition

La détection d'objets est une méthode de traitement d'image qui permet d'identifier et de localiser des objets dans une image ou une vidéo en utilisant des algorithmes de la vision artificielle. Cette tâche nécessite de combiner deux étapes essentielles :

- **Classification :** Attribution d'une étiquette à un objet détecté.
- **Localisation :** Détermination de la position exacte de l'objet dans l'image en dessinant une boîte de délimitation autour de l'objet.

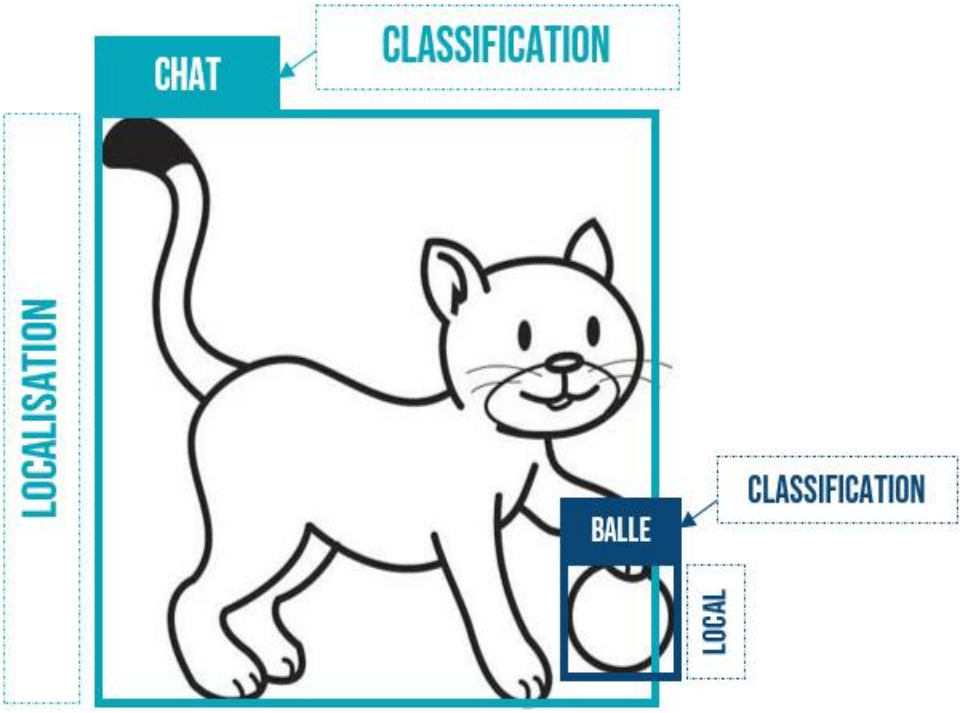


Figure II.1 : Exemple de reconnaissance d'objets

II.2.2. Domaines d'applications

La détection d'objets est une technologie prometteuse qui offre de nombreuses applications pratiques pour améliorer la sécurité, la navigation et la surveillance dans différents domaines telles que la surveillance de la sécurité publique, la reconnaissance de plaques d'immatriculation, la détection de panneaux de signalisation, la navigation autonome des voitures et la surveillance industrielle.

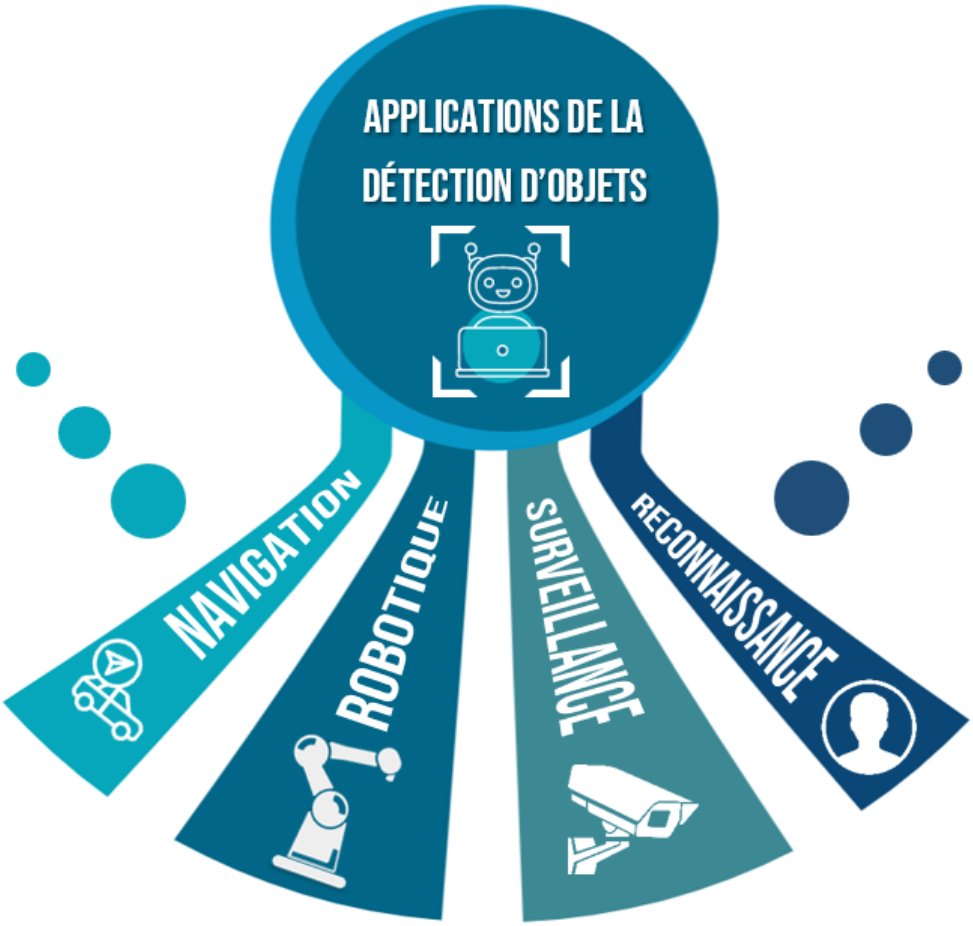


Figure II.2 : Applications de la détection d'objets

II.3. Évolution de la détection d'objets

L'évolution de la détection d'objets a été marquée par une transition des méthodes traditionnelles de traitement d'images aux approches d'apprentissage en profondeur plus avancées, permettant une précision accrue, une généralisation plus forte et une détection en temps réel plus rapide et plus précise et ce grâce aux avancées dans les techniques l'apprentissage profond, de l'apprentissage automatique et de la vision par ordinateur. Voici quelques étapes clés dans cette évolution :

II.3.1. Détection de caractéristiques

Les premières méthodes de détection d'objets se concentraient sur l'extraction de caractéristiques à partir d'images, telles que les coins, les bords, les textures, etc. Ces caractéristiques étaient ensuite utilisées pour identifier des régions d'intérêt qui pouvaient contenir des objets.

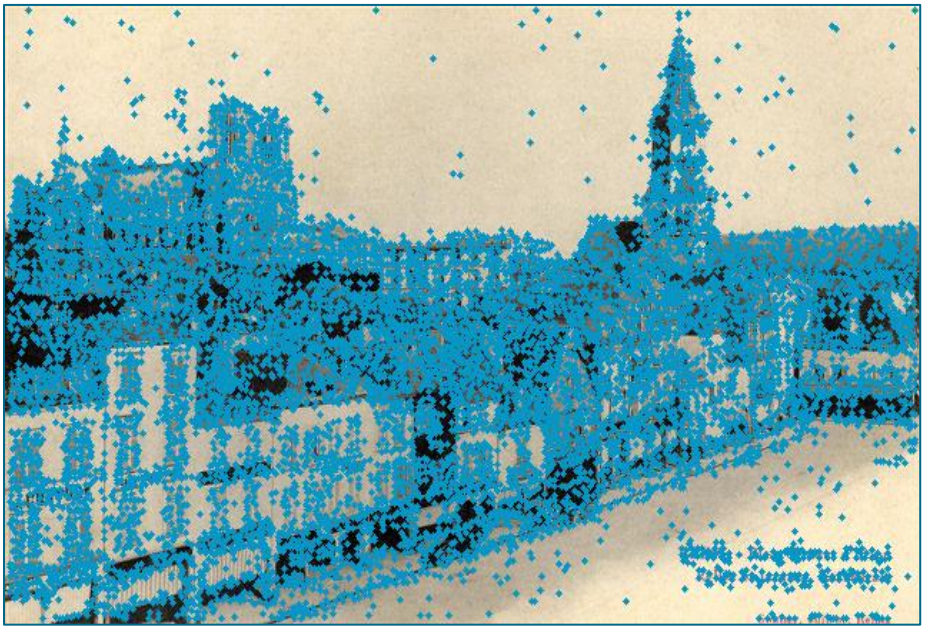


Figure II.3 : Image réelle illustrant la détection de caractéristiques

Il existe plusieurs techniques de détection de caractéristiques populaires en vision par ordinateur :

II.3.1.1. Haar Cascades

Haar Cascades est une technique qui a été développée par Viola et Jones en 2001. Elle utilise des classificateurs en cascade pour détecter des caractéristiques telles que des bords, des coins et des lignes dans une image, qui sont ensuite combinées pour former une description complète de l'objet. Cette technique est largement utilisée dans le domaine de la vision par ordinateur pour la détection d'objets tels que les visages, les yeux, les sourires, etc. [9]

II.3.1.2. HOG

HOG est une technique puissante pour la détection d'objets dans des images. Elle consiste à diviser une image en petits blocs, à calculer l'orientation des gradients pour chaque pixel dans chaque bloc, puis à créer un histogramme des orientations des gradients pour l'ensemble des pixels dans chaque bloc qui sont utilisés pour décrire la forme et les contours de l'objet dans l'image. [10]

La méthode HOG a été initialement proposée pour la détection de piétons dans des images de vidéosurveillance, mais elle peut également être utilisée pour détecter d'autres types d'objets tels que des voitures, des animaux, etc.

II.3.2. Régions proposées

Les méthodes de détection d'objets basées sur des régions proposées ont été développées pour améliorer la précision. Ces méthodes proposent des régions d'intérêt potentielles à partir de l'image, puis appliquent un classificateur pour déterminer si ces régions contiennent un objet ou non. Si on prend alors l'exemple cité dans la figure II.4, l'algorithme va retirer la région verte et conserver uniquement la région présentant des chances d'avoir des objets détectables.

Figure II.4 : Image réelle illustrant la détection de caractéristiques

II.3.3. Réseaux de neurones convolutifs

Les CNN ont été introduits pour la détection d'objets, permettant aux algorithmes de traiter directement les images brutes plutôt que de devoir extraire des caractéristiques manuellement. Ils ont permis d'améliorer considérablement la précision de la détection d'objets.

Les modèles de détection d'objets par les architectures CNN peuvent être divisés en deux types : la détection en deux coups et la détection à un coup.

II.3.3.1. Détection en deux coups

La détection en deux coups utilise généralement une approche basée sur la proposition de régions pour détecter des objets dans une image. Cette approche est divisée en deux étapes principales. Dans la première étape, des ROIs sont générés à partir de l'image en utilisant une méthode de proposition de régions [11].

Les ROIs sont ensuite redimensionnés pour être utilisées comme entrée dans un réseau de neurones convolutifs qui extrait les caractéristiques de chaque ROI. Dans la seconde étape, chaque

ROI est classifié et les décalages par rapport à l'emplacement et la taille des objets réels sont prédits. Cette étape est généralement réalisée à l'aide d'un réseau de neurones convolutifs qui produit en sortie les scores de classification et les décalages correspondants.

En résumé, la détection en deux coups avec la proposition de région utilise une méthode en deux étapes pour détecter des objets dans une image en se concentrant sur les régions les plus prometteuses. On peut retrouver dans cette catégorie plusieurs méthodes tel que :

II.3.3.1.1. R-CNN

R-CNN [12] est un modèle de détection d'objets basé sur la proposition de région. Combinant ainsi des régions proposées (même principe que celui présenté dans la figure II.4) avec des CNN. Cette méthode a considérablement amélioré les performances en termes de précision, mais était relativement lente.

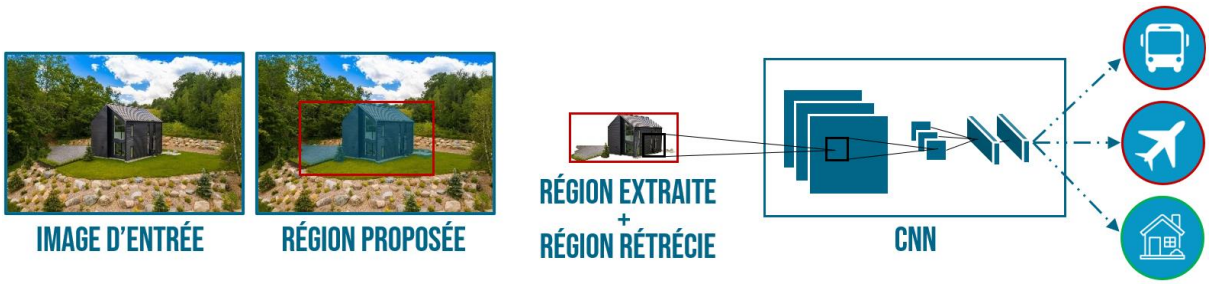


Figure II.5 : Architecture et fonctionnement du model R-CNN

II.3.3.1.2. Fast R-CNN

Fast R-CNN [13] a été développé améliorant la vitesse de R-CNN en utilisant une seule passe de CNN pour classifier les régions proposées.

II.3.3.1.3. Faster R-CNN

Faster R-CNN [14] a été développé en utilisant un réseau de neurones convolutifs pour générer des régions proposées de manière efficace. Cette méthode a permis de réduire encore le temps de traitement tout en améliorant la précision.

II.3.3.2. Détection en un coup

La détection en un coup est une approche de détection d'objets qui ne nécessite pas de proposition de régions. Cette approche utilise un réseau de neurones convolutifs pour prédire directement les boîtes englobantes et les classes des objets dans une seule passe sur l'image [11].

Dans un réseau de détection en un coup, l'image d'entrée est divisée en grilles et chaque grille prédit un ensemble de boîtes englobantes et de scores de confiance pour chaque classe d'objet possible. Ces boîtes englobantes sont ajustées en fonction des caractéristiques extraites de chaque grille, permettant ainsi de localiser précisément les objets d'intérêt.

En somme, la détection en un seul passage se sert d'une méthode de détection d'objets qui prédit simultanément les boîtes englobantes et les classes des objets, sans avoir besoin de proposer des régions préalables sur l'image. Bien que cette approche soit généralement plus rapide, elle peut être moins précise que la détection en deux coups. Parmi les méthodes de détection en un coup, on peut citer :

II.3.3.2.1. RetinaNet

RetinaNet [15] est une méthode de détection d'objets basée sur les réseaux de neurones convolutifs, inspiré de la rétine, qui est la couche sensible à la lumière de l'œil. De la même manière que la rétine de l'œil capte la lumière pour permettre la vision, RetinaNet "capture" les caractéristiques des objets dans une image pour permettre la détection d'objets.

Elle est conçue pour résoudre le problème de la classification déséquilibrée dans la détection d'objets, où la grande majorité des régions d'image ne contiennent pas d'objets d'intérêt, conduisant à un nombre écrasant d'échantillons négatifs. La particularité de cette méthode est qu'elle permet de détecter efficacement des objets de petite taille, qui peuvent être difficiles à détecter avec d'autres méthodes.

II.3.3.2.2. SSD

La méthode SSD [16] repose sur l'utilisation d'un réseau de neurones convolutifs feed-forward pour détecter des objets dans des images. Ce réseau produit des boîtes englobantes de taille fixe et des scores pour la présence d'instances de classes d'objets dans ces boîtes. Les premières couches du réseau sont basées sur l'architecture VGG-16 utilisée pour la classification d'images de haute qualité, mais tronquées avant toute couche de classification. Après la production des boîtes englobantes, une étape de suppression non maximale est effectuée pour produire les détections finales.

II.3.3.2.3. YOLO

Depuis sa première publication en 2015, YOLO [17] est devenu l'une des architectures de détection d'objets les plus populaires et les plus utilisées. Il a été créé par Joseph Redmon et Ali Farhadi et a été amélioré au fil des ans avec des versions proposées par les créateurs eux-mêmes, ainsi que par la communauté.

YOLO utilise une approche de détection d'objets en une seule passe, ce qui le rend plus rapide et plus efficace pour la détection d'objets en temps réel. Cette caractéristique en fait un choix populaire pour de nombreuses applications de vision artificielle où la détection d'objets en temps réel est essentielle. Il repose également sur un réseau de neurones profond qui peut être entraîné sur de grandes quantités de données pour améliorer les performances de détection. Tout comme le SSD, YOLO utilise des boîtes englobantes pour détecter et localiser des objets dans des images ou des vidéos. Ainsi que des techniques comme la suppression non-maximale et la régression pour améliorer la précision de la détection d'objets. De ce fait il a pris une place cruciale dans les domaines de la surveillance, de la sécurité et de la reconnaissance [18].

On peut résumer l'évolution de la détection d'objets en présentant un schéma final qui regroupe les étapes clés de cette évolution. Ce schéma, représenté dans la figure (II.6), met en évidence les progrès réalisés au fil du temps.

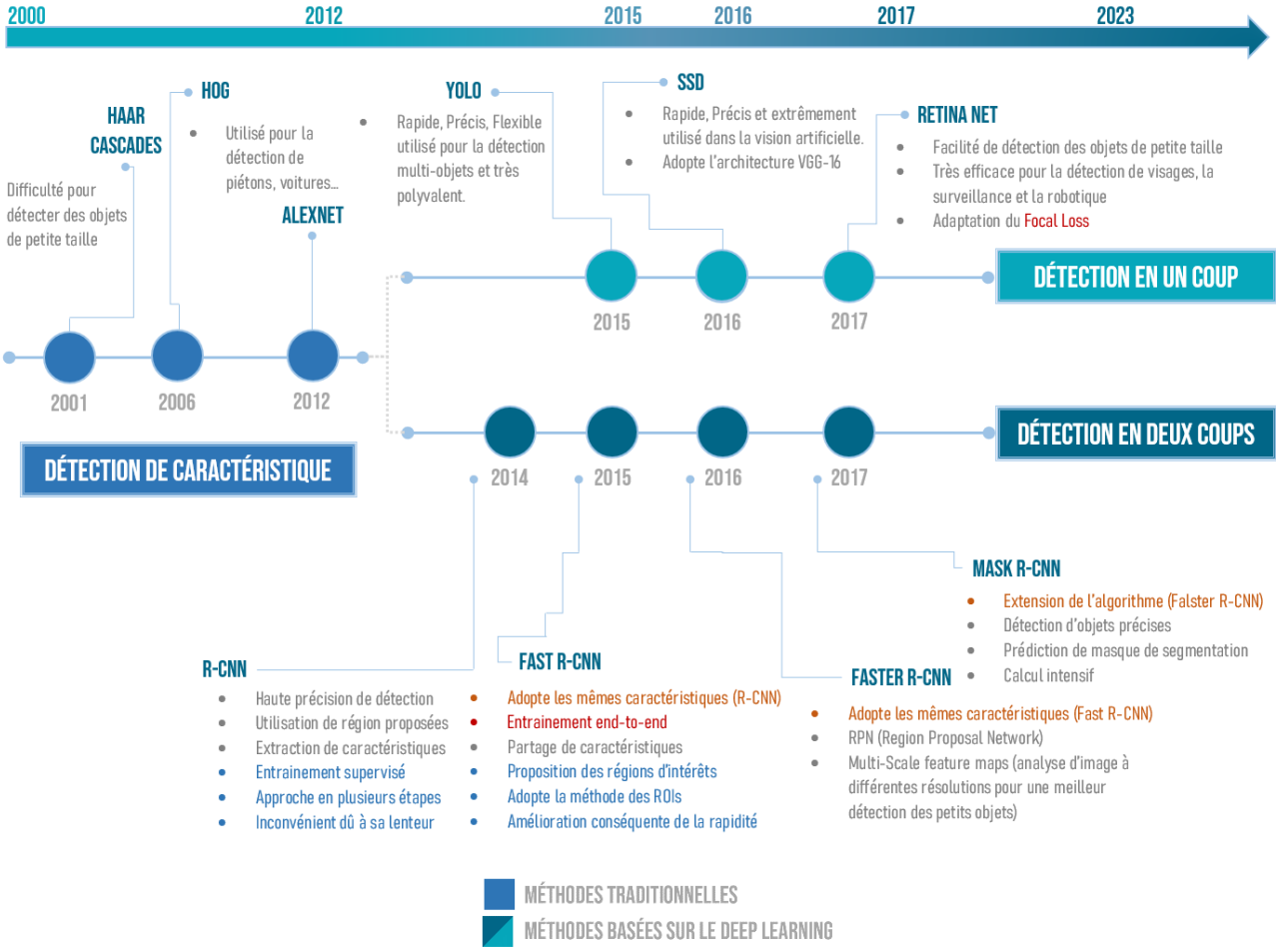


Figure II.6 : Schéma regroupant l'évolution de la détection d'objets

II.4. Fondements théoriques de YOLO

II.4.1. Fonctionnement de YOLO

Comme on l'a vu précédemment, YOLO est un modèle de détection d'objets en temps réel en vision artificielle, contrairement aux autres algorithmes de détection d'objets qui effectuent des recherches d'objets sur des régions de l'image. Ce modèle est fractionné en 3 étapes essentielles.

II.4.1.1. Division en grilles

La principale règle qu'adopte YOLO est de diviser l'image en petites zones de taille fixe, appelées "cellules". Chaque cellule est ensuite évaluée indépendamment des autres pour détecter la présence d'objets.

On peut prendre un exemple d'une image de 224×224 pixels, L'algorithme YOLO va alors la diviser en plusieurs cellules selon une taille de (S × S), S étant la longueur du côté du carré de la cellule, Si $S = 8$ pixels alors l'image sera coupée en 64 cellules de $28 \times 28 = 784$ pixels.

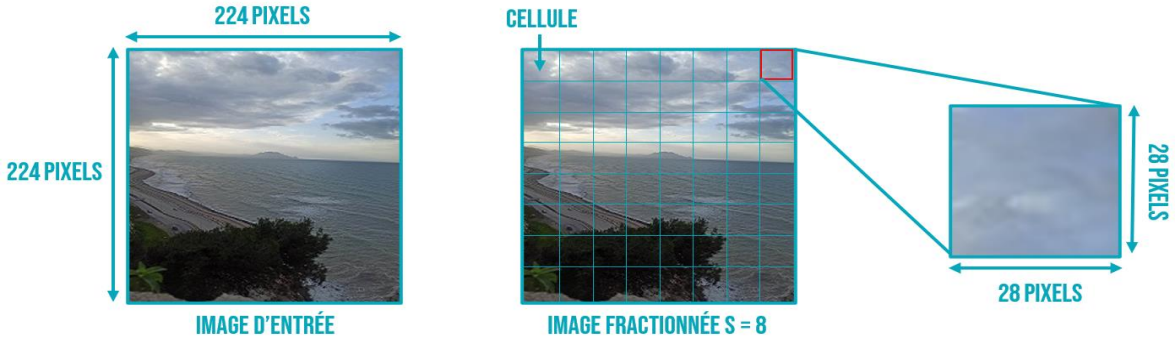


Figure II.7 : Fractionnement d'une image en 64 cellules

II.4.1.2. Prédiction des boites englobantes

La boîte englobante dans YOLO est une région rectangulaire qui encadre l'objet détecté dans une image. Cette boîte est représentée par quatre coordonnées : les coordonnées « x » et « y » du coin supérieur gauche de la boîte, ainsi que sa largeur « w » et sa hauteur « h ».

La prédiction des boîtes englobantes est la tâche qui vient après ce que l'algorithme YOLO ait diviser l'image en cellules, c'est un processus itératif. On prend chaque cellule puis le modèle commence par prédire des boîtes englobantes approximatives pour tous les objets de l'image.

Prenons l'exemple d'une image avec une grille de 3x3 cellules (S=3), où chaque cellule prédit une seule boîte englobante (B=1) pour des objets qui peuvent être soit des voitures (C=1) soit des humains (C=2). Dans ce cas, le réseau de neurones convolutifs prédit un vecteur K pour chaque cellule. La figure (II.8) illustre un schéma représentatif de l'exemple mentionné précédemment.

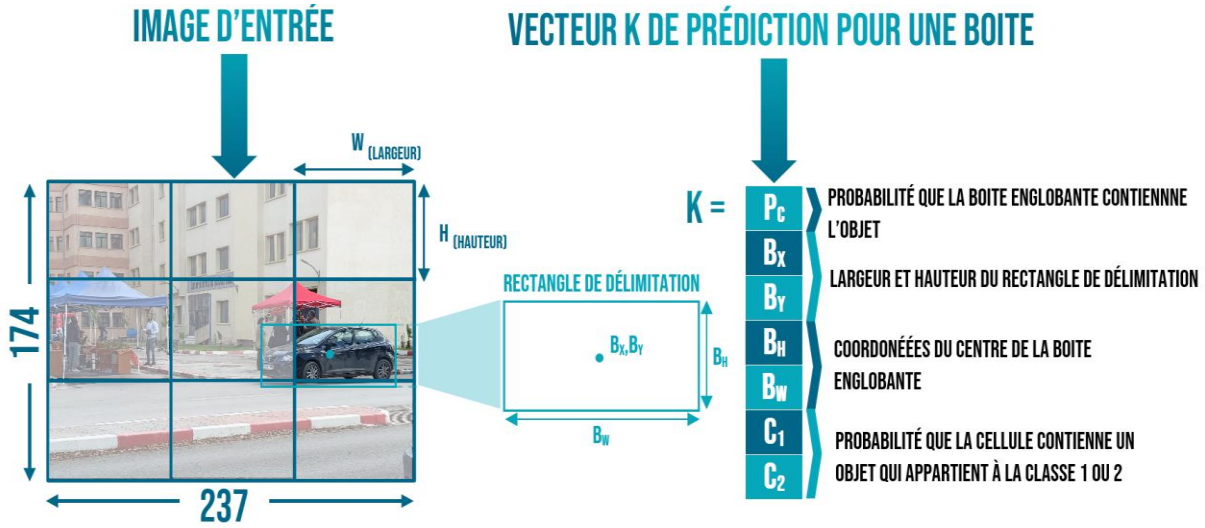


Figure II.8 : Affectation et analyse du vecteur de prédiction pour une seule boîte englobante

On peut calculer alors directement les éléments inconnus du vecteur de prédiction K au format YOLO avec les relations suivantes :

$$B_x = \frac{X-H}{H} \quad \text{(II.1)}$$

$$B_y = \frac{Y-W}{W} \quad \text{(II.2)}$$

$$B_H = \frac{H'}{174} \quad \text{(II.3)}$$

$$B_w = \frac{W'}{237} \quad \text{(II.4)}$$

Où X et Y sont les coordonnées cartésiennes du centre du rectangle de prédiction.

Ensuite, l'algorithme ajuste ces boîtes pour mieux s'adapter aux objets réels en utilisant des techniques, à savoir :

II.4.1.2.1. L'intersection sur union

L'intersection sur union (IoU en anglais) ou Indice de Jaccard est une mesure de la similarité entre deux boîtes englobantes. Elle est couramment utilisée pour déterminer la précision des prédictions en distinguant les vrais positifs des faux positifs. Pour évaluer les résultats en utilisant l'IoU, il est nécessaire de définir un seuil de précision [19]. Cette mesure permet de comparer la boîte prédite avec la boîte détectable et de calculer la surface de chevauchement entre ces deux boîtes. Elle est calculée en divisant la surface de l'intersection entre les deux boîtes englobantes par la surface de leur union. Cette mesure permet de déterminer si deux boîtes englobantes chevauchantes représentent le même objet ou non.

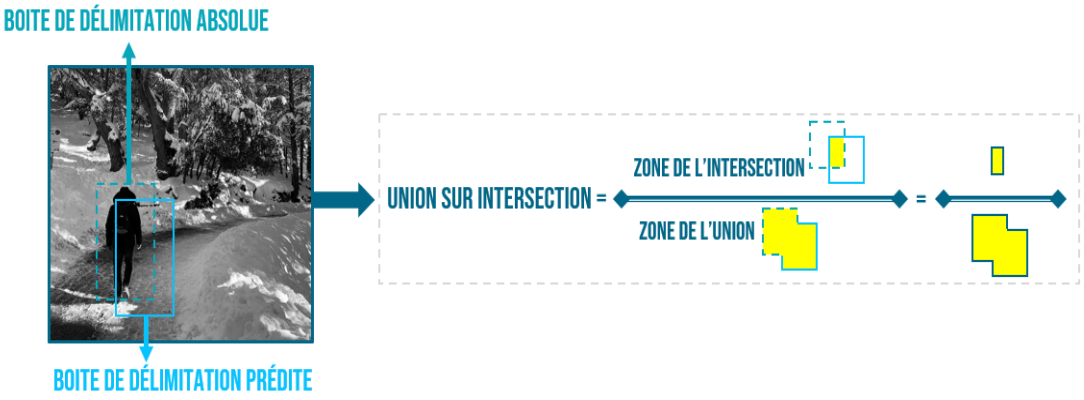


Figure II.9 : Schéma démonstratif de l'union sur intersection

Il peut y'avoir par la suite plusieurs cas de figure de ces dernières et on les note avec un score d'IoU. On dit alors que le score est plus élevé plus le chevauchement entre les boites est meilleur. Les prédictions avec des scores d'IoU plus élevés sont considérées comme étant plus précises car elles correspondent davantage aux boîtes englobantes réelles. À l'inverse, les

prédictions avec des scores d'IoU plus faibles sont moins précises et nécessitent des ajustements pour améliorer les performances du modèle.

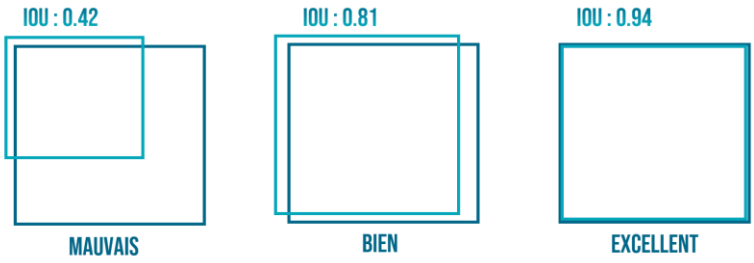


Figure II.10 : Différences entre les scores d'IoU

Une fois que les boîtes englobantes ont été détectées pour chaque objet, le modèle doit ajuster ces boîtes englobantes pour qu'elles s'alignent plus précisément avec les objets réels. Cela est accompli à l'aide de techniques de régression qui cherchent à minimiser la différence entre les boîtes englobantes prédites et les boîtes englobantes réelles, tout en maximisant l'intersection sur union entre ces deux boîtes.

En plus des boîtes englobantes, le modèle de détection d'objet doit également déterminer la classe de chaque objet détecté. Pour ce faire, le modèle utilise un classificateur qui attribue une probabilité à chaque classe possible pour chaque boîte englobante prédite. Les classes sont souvent représentées par des catégories d'objets, telles que "voiture", "personne", "chat", etc (B).

En plus des prédictions de classe, le modèle de détection d'objet peut également fournir des prédictions de position en 2D pour chaque boîte englobante prédite. Ces prédictions de position sont souvent appelées prédictions de régression 2D et permettent de raffiner la position et la taille de chaque boîte englobante prédite (C).

Finalement, chaque boîte englobante prédite est associée à une classe et à des prédictions de régression 2D, ce qui permet de fournir des prédictions complètes de boîte englobante et de classe pour chaque objet détecté dans l'image. Ces informations peuvent être utilisées pour identifier et suivre les objets dans une séquence d'images, ou pour d'autres tâches liées à la détection d'objet, telles que la segmentation sémantique ou la reconnaissance de gestes (D).

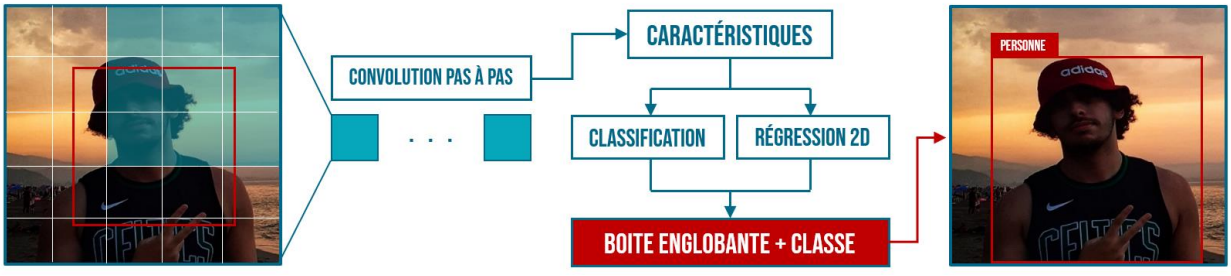


Figure II.11 : Ajustement des boites englobantes

II.4.1.2.2. Boîte d'enclenchement

La boîte d'enclenchement est une technique qui consiste à prédéfinir plusieurs tailles et formes de boîtes englobantes candidates, appelées ancres (anchors), pour chaque classe d'objet. YOLO utilise ensuite ces ancres pour prédire des boîtes englobantes qui correspondent mieux aux objets réels. Les ancres sont définies en fonction de la taille et de la forme des objets dans le jeu de données d'entraînement. Pendant l'entraînement, l'algorithme de YOLO apprend à prédire des boîtes englobantes en combinant les informations des ancres prédéfinies avec les caractéristiques de l'image analysée.

Si plusieurs boîtes se trouvent dans la même cellule, en prenant l'exemple cité dans (II.4.1.2) le vecteur correspondant à cette cellule est augmenté comme montré dans la figure (II.12).

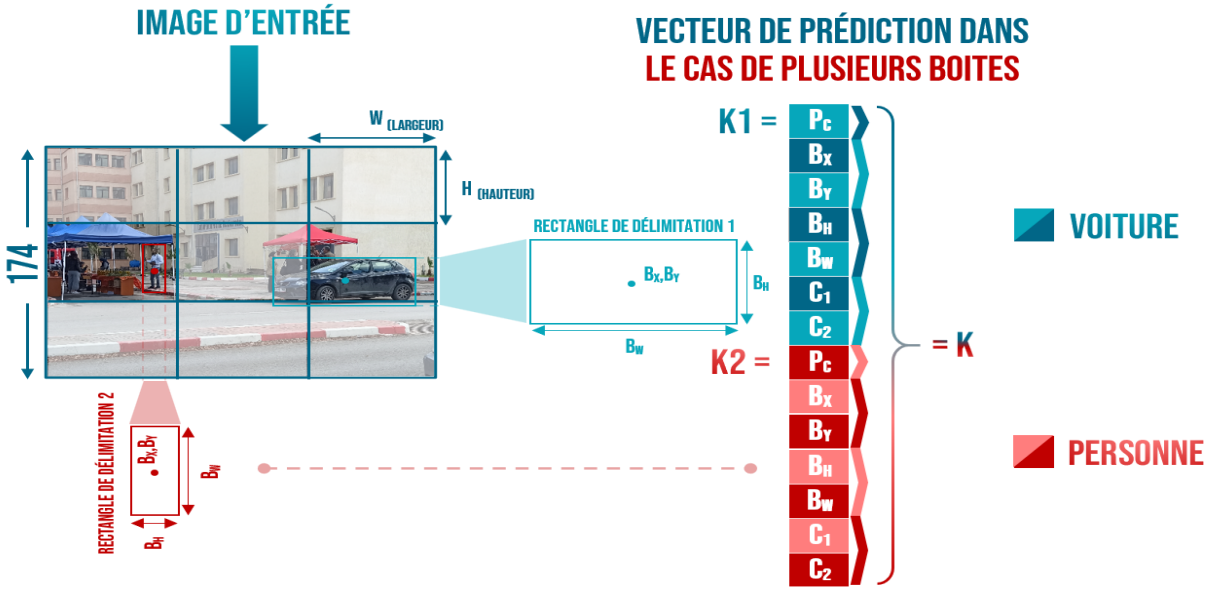


Figure II.12 : Affectation et analyse du vecteur de prédiction pour plusieurs boîtes englobantes

II.4.1.2.3. Classification

Après avoir prédit les boîtes englobantes pour les objets détectés dans l'image, YOLO utilise un score pour évaluer la qualité de chaque boîte englobante. Ce score est calculé en combinant deux mesures :

1- **La confiance de l'algorithme** dans la détection de l'objet dans cette boîte englobante (c'est-à-dire à quel point l'objet ressemble à la classe d'objets prédite)
2- **La précision de la boîte englobante** (c'est-à-dire à quel point la boîte englobante est proche de l'objet réel).

Le score final pour chaque boîte englobante est alors calculé en multipliant la confiance par la précision.

$$Score_{finale} = Confiance \times Précision \quad \text{(II.15)}$$

Le score de confiance ou (Confiance) est généralement compris entre 0 et 1, où 1 indique une confiance totale dans la prédiction de l'objet dans la boîte englobante. Si le score de confiance est inférieur à un certain seuil prédéfini, la boîte englobante est considérée comme étant une fausse détection et est donc ignorée.

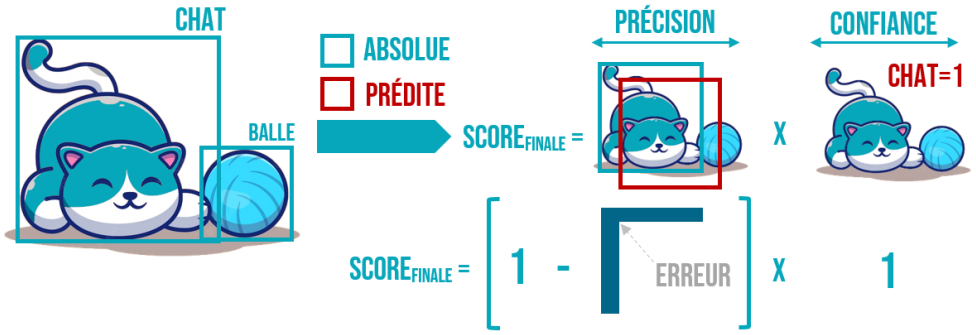


Figure II.13 : Évaluation du score finale d'une boite englobante

YOLO utilise ensuite un seuil de score pour sélectionner les boîtes englobantes qui représentent les meilleures prédictions pour chaque objet dans l'image en utilisant des techniques comme la suppression non maximale qui est couramment utilisée dans les algorithmes de détection d'objet pour éliminer les doublons et améliorer la précision de la détection.

Figure II.14 : Les étapes générales du processus de suppression non maximale

Un exemple de suppression non maximale peut être illustré dans la figure (II.15).

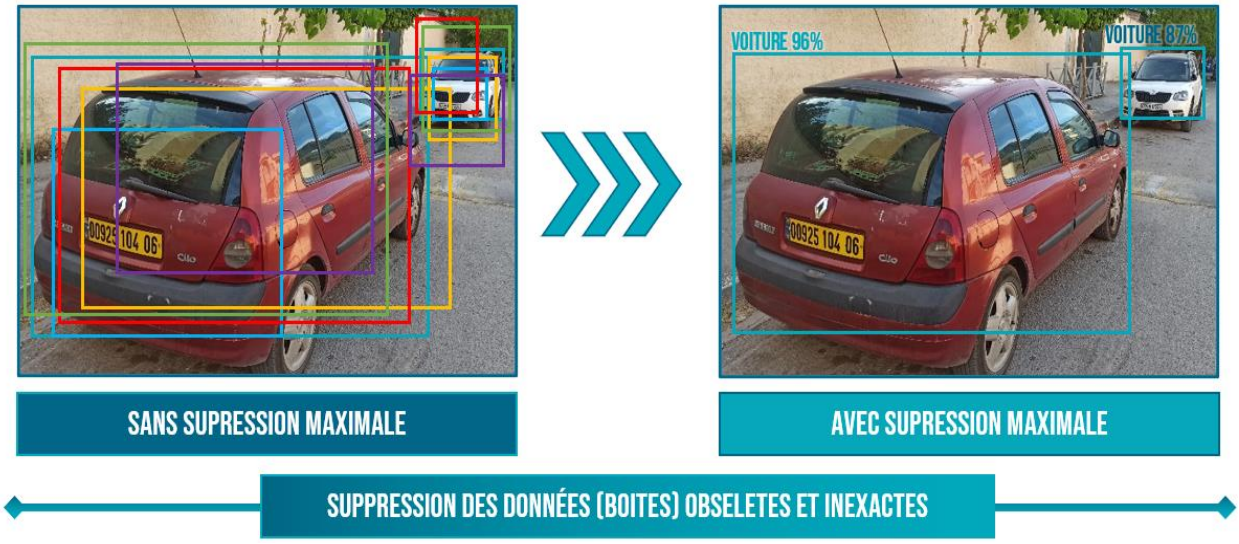


Figure II.15 : Application de la suppression maximale

II.4.2. Variations de YOLO

Après la création de YOLO en 2015, ses créateurs ont travaillé à améliorer l'algorithme et ont sorti plusieurs versions améliorées.

YOLOv2 [20] est sorti en 2016, il avait des performances supérieures à YOLOv1 et utilisait des techniques telles que la batch normalisation et les cellules résiduelles pour améliorer la précision.

YOLOv3 [21] qui a été publié en 2018 et qui a apporté de nouvelles améliorations telles que la détection multi-échelle, une meilleure gestion des fausses détections et l'utilisation de l'apprentissage en profondeur pour améliorer les performances.

YOLOv4 [22] est finalement sorti en 2020, il s'agit de la version la plus avancée à ce jour. Elle utilise une architecture plus complexe, avec des couches résiduelles CSP, des méthodes d'augmentation de données innovantes et une fonction d'erreur modifiée pour améliorer les performances de détection.

D'autres versions ultérieures, telles que **YOLOv5**, **YOLOv6**, **YOLOv7** et **YOLOv8**, ont été développées par d'autres individus, mais elles ne sont pas officiellement reconnues sous la licence YOLO.

Le tableau (II.1) récapitule les distinctions entre toutes les versions de YOLO connues à ce jour.

Tableau II.1 : Tableau comparatif des versions de YOLO

Version	Créateur(s)	Date de création	Nombre de couche de convolution	Framework
YOLOv1	Joseph Redmon	2015	24	Darknet
YOLOv2	Joseph Redmon & Ali Farhadi	2016	19	Darknet
YOLOv3	Joseph Redmon & Ali Farhadi	2018	53	Darknet
YOLOv4	Alexey Bochkovskiy, Chien-Yao Wang & Hong-Yuan Mark Liao	2020	53	Darknet
YOLOv5	Ultralytics	2020		PyTorch
YOLOv6	WONG Kin Ho	2021		PyTorch
YOLOv7	Nicholas Renotte	2021		PyTorch
YOLOv8	Mingxing Tan & Quoc V	2022		TensorFlow

Remarques : Pour les versions de 6 à 8, il n'y a pas d'informations officielles disponible, mais nous avons recueilli des données généralement partagées sur les forums spécialisés. Par conséquent, ces informations ne sont pas fiables à 100%. Quant au nombre de couches de ses versions, il dépend de l'architecture utilisée. En raison des nombreuses versions publiées, chaque version adopte un nombre de couches qui lui est propre. Par exemple : Pour YOLOv5, la version YOLOv5s compte 9 couches, tandis que la version YOLOv5x compte 19 couches…, etc.

Il y'a eu entre temps aussi des versions antérieures tel que YOLO9000 qui est une version plus récente de YOLOv2 qui a été publiée en 2017. Elle intègre les fonctionnalités de YOLOv2 tout en augmentant le nombre de classes d'objets détectables à plus de 9 000 d'où elle tire son nom [23].

II.4.3. Frameworks de YOLO

Les frameworks dans l'apprentissage profond, en particulier dans YOLO, sont utilisés pour plusieurs raisons, on peut citer :

1- Fournit une interface facile à utiliser pour concevoir, entraîner et déployer des modèles de détection d'objets. Ils offrent également des fonctionnalités avancées telles que l'autodifférenciation et la parallélisation pour améliorer l'efficacité de l'entraînement et du déploiement de modèles.
2- Permet une personnalisation facile des modèles de détection d'objets, ce qui peut être important pour des applications spécifiques nécessitant une précision et une performance accrues.
3- Les frameworks permettent également de bénéficier de la contribution de la communauté et de la disponibilité des modèles pré-entrainés pour faciliter le processus de développement et d'entraînement.

On trouve généralement trois types de frameworks qui sont récurrents dans toutes les versions de YOLO, et ce sont :

II.4.3.1. Darknet

Darknet est un framework open source écrit en C et CUDA qui a été développé spécifiquement pour YOLO. Darknet permet de concevoir, entraîner et déployer des modèles de détection d'objets, y compris des modèles YOLO. Il est largement utilisé pour entraîner des modèles de détection d'objets à grande échelle, car il peut être optimisé pour fonctionner rapidement sur des GPU.

II.4.3.2. PyTorch

PyTorch est un framework open source développée par Facebook qui est devenu populaire ces dernières années. PyTorch permet de créer et d'entraîner des modèles de détection d'objets de manière efficace, en utilisant des fonctionnalités avancées telles que l'autodifférenciation et des modules pré-entrainés. Il est également compatible avec YOLO et d'autres modèles de détection d'objets.

II.4.3.3. TensorFlow

TensorFlow est un framework populaire, développé par Google. Tout comme PyTorch et Darknet, TensorFlow peut être utilisé pour concevoir, entraîner et déployer des modèles de détection d'objets tels que YOLO. TensorFlow est également optimisé pour fonctionner sur des processeurs graphiques (GPU), ce qui permet d'accélérer le processus d'entraînement des modèles. TensorFlow propose également des fonctionnalités avancées telles que l'autodifférenciation et des modules pré-entrainés pour faciliter la création et l'entraînement de modèles de détection d'objets.

II.4.4. Architecture de YOLO

Chaque version de l'algorithme YOLO adopte une architecture de réseaux de neurones convolutifs différente, toutes basées sur l'architecture Darknet.

YOLOv1 est basée sur le réseau de neurones convolutifs inspirée du modèle GoogLeNet, son réseau contient 24 couches de convolution et 2 couches entièrement connectées.

YOLOv2 utilise une architecture appelée Darknet-19, qui intègre des connexions résiduelles pour améliorer la performance et la vitesse de l'algorithme. Ce réseau est composé de 19 couches de convolution et 5 couches de maxpooling.

YOLOv3, quant à lui, repose sur l'architecture appelée Darknet-53, qui intègre des blocs résiduels et des connexions en skip pour améliorer à la fois la précision et la performance de l'algorithme. Ce réseau est composé de 53 couches de convolution.

YOLOv4 utilise une version améliorée de Darknet-53 avec plusieurs améliorations pour améliorer la performance et la précision de l'algorithme.

	Type	Filtres	Taille	Sortie
	Convolution	32	3 × 3	256 × 256
	Convolution	64	3 × 3 / 2	128 × 128
1×	Convolution	32	1 × 1	
	Convolution	64	3 × 3	
	Résiduel			128 × 128
	Convolution	128	3 × 3 / 2	64 × 64
2×	Convolution	64	1 × 1	
	Convolution	128	3 × 3	
	Résiduel			64 × 64
	Convolution	256	3 × 3 / 2	32 × 32
8×	Convolution	128	1 × 1	
	Convolution	256	3 × 3	
	Résiduel			32 × 32
	Convolution	512	3 × 3 / 2	16 × 16
8×	Convolution	256	1 × 1	
	Convolution	512	3 × 3	
	Résiduel			16 × 16
	Convolution	1024	3 × 3 / 2	8 × 8
4×	Convolution	512	1 × 1	
	Convolution	1024	3 × 3	
	Résiduel			8 × 8
	Average pooling		Global	
	Connectées		1000	
	SoftMax			

Figure II.16 : Architecture de Darknet53 [21]

Le réseau de YOLOv4 est composé de trois parties principales, chacune divisée en plusieurs blocs [23].

II.4.4.1. Épine dorsale CSPDarknet53

CSPDarknet53 est une architecture de réseaux de neurones convolutifs, YOLOv4 l'utilise comme base pour extraire des caractéristiques à partir de l'image d'entrée, qui sont ensuite utilisées pour prédire les positions et les classes des objets présents dans l'image.

CSPDarknet53 est divisé en plusieurs blocs qui sont ensuite utilisés pour améliorer les performances de la détection d'objets. Chaque bloc CSP contient deux branches parallèles, avec des couches de convolution, de normalisation et de non-linéarité. Les deux branches sont ensuite fusionnées en utilisant une opération de concaténation.

En utilisant CSPDarknet53, YOLOv4 est capable d'obtenir des performances de détection d'objets supérieures à celles de ses prédécesseurs. La combinaison de CSPDarknet53 avec d'autres

techniques de pointe, telles que l'augmentation de données, permet à YOLOv4 d'obtenir des résultats impressionnants sur les ensembles de données de détection d'objets les plus courants, tels que COCO et Pascal VOC.

II.4.4.2. Coup

II.4.4.2.1. SPP

SPP est une technique utilisée dans YOLOv4 pour améliorer la capacité de la méthode à détecter des objets à différentes échelles dans l'image. Il permet de prendre en compte les caractéristiques d'une image à différentes résolutions sans avoir à redimensionner l'image elle-même.

SPP prend la sortie du backbone de YOLOv4 et la divise en différentes zones de tailles variées. Ensuite, un max-pooling est appliqué à chaque zone afin d'obtenir une représentation de chaque zone à une échelle fixe. Ces représentations sont ensuite concaténées pour former un vecteur de caractéristiques global, qui est utilisé pour prédire les positions et les classes des objets présents dans l'image.

De là, on peut dire que l'utilisation du SPP est un élément important de l'architecture de YOLOv4 qui contribue à sa capacité à détecter des objets avec une grande précision et à s'adapter à des situations réelles diverses.

II.4.4.2.2. PAN

PAN quant à lui est une technique utilisée dans YOLOv4 pour améliorer la précision de la détection d'objets en permettant la fusion de caractéristiques à différentes échelles spatiales. Il est utilisé pour agréger des caractéristiques provenant de différentes couches du réseau de neurones et pour les fusionner afin d'obtenir une représentation globale de l'image.

PAN utilise une architecture de pyramide de caractéristiques pour agréger les caractéristiques de différentes échelles spatiales. Les caractéristiques sont agrégées à partir de plusieurs niveaux de la pyramide, puis fusionnées en une seule représentation globale de l'image. Cette représentation est ensuite utilisée pour prédire les positions et les classes des objets dans l'image.

De ce fait, PAN permet à YOLOv4 de prendre en compte des caractéristiques à différentes échelles spatiales, ce qui améliore la précision de la détection d'objets, notamment pour les objets de petite taille ou éloignés. PAN permet également de réduire les faux positifs en utilisant des caractéristiques de différentes échelles spatiales pour confirmer la présence réelle d'un objet dans l'image.

II.4.4.3. Tête

La "tête" de YOLOv4 fait référence à la dernière partie de l'architecture du réseau de neurones, qui est responsable de la prédiction des positions et des classes des objets dans l'image.

Elle est composée de plusieurs couches entièrement connectées qui transforment les caractéristiques extraites par le backbone du réseau de neurones en prédictions de boîtes englobantes et de classes d'objets correspondantes. Elle utilise des techniques comme le E-RPN qui est une technique utilisée pour améliorer la vitesse et la précision de la détection d'objets. E-RPN est une version améliorée du RPN utilisé dans les modèles de détection d'objets basés sur les réseaux de neurones convolutifs.

E-RPN est utilisé pour générer des propositions des ROIs dans l'image, qui sont ensuite utilisées pour détecter des objets. La différence principale entre E-RPN et le RPN classique est que E-RPN utilise une architecture pyramidale pour extraire des caractéristiques à différentes échelles spatiales, ce qui permet d'améliorer la précision de la génération de ROIs.

Dans YOLOv4, E-RPN est utilisé dans la tête du réseau pour générer des propositions de ROIs, qui sont ensuite utilisées pour prédire les positions et les classes des objets dans l'image. L'utilisation de E-RPN permet à YOLOv4 d'améliorer la précision de la détection d'objets tout en maintenant une haute vitesse de traitement des images, ce qui est important pour des applications en temps réel.

La figure II.17 présente un schéma qui permet de récapituler l'architecture globale de YOLOv4 en combinant les trois parties mentionnées précédemment.

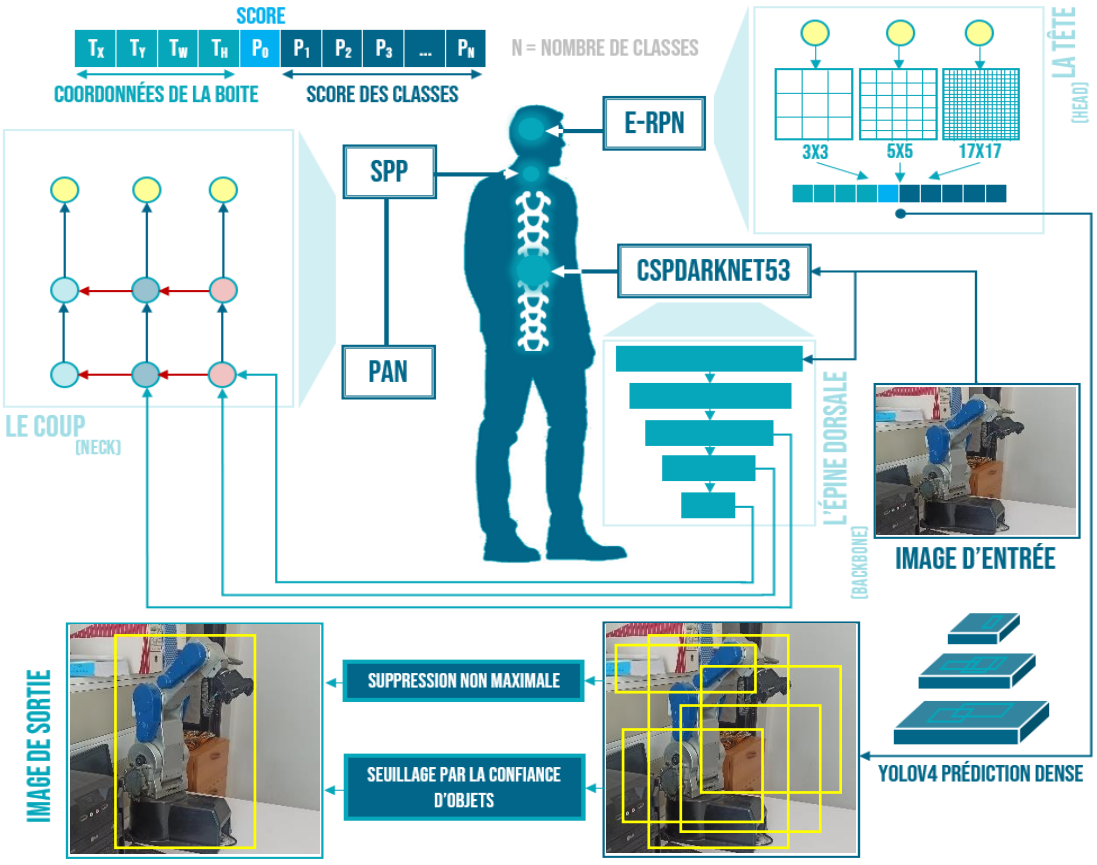


Figure II.17 : Architecture générale de YOLOV4.

II.5. Conclusion

Au cours de ce chapitre consacré à la détection d'objets, nous avons examiné diverses méthodes qui s'appuient sur l'apprentissage en profondeur. En commençant de l'évolution de ces méthodes jusqu'au fondements théoriques de ses dernières en passant les revues des méthodes les plus populaires et les plus couramment utilisées pour la détection d'objets, en détaillant leurs architectures respectives. Nous avons ensuite examiné en particulier la méthode YOLO, en présentant son algorithme détaillé ainsi toutes les versions de YOLO (officielles et non-officielles) et leurs différences. Nous avons privilégié la version YOLOv4 qui sera la méthode principale de notre travail pour la partie conception.

CHAPITRE 3

CRÉATION DE LA BASE DE DONNÉES YAHINE

Récapitulatif

Ce chapitre se concentre sur la création de la base de données YAHINE, qui forme le socle de notre étude. Cette base de données a été spécialement développée pour capturer des images représentatives d'objets d'intérieur dans diverses situations. Nous avons décrit en détail les étapes essentiels, telles que la collecte des images, la sélection des scènes, la création des annotations, etc.

III.1. Introduction

Dans ce chapitre, nous détaillerons la création de la base de données YAHINE, qui constitue le fondement de notre étude. Cette base de données a été spécialement conçue pour capturer des images représentatives d'objets d'intérieur prises dans différentes situations, Nous détaillerons les étapes de mise en forme et d'annotations qui ont contribué à la création de cette dernière.

III.2. Base de données

Une base de données est un ensemble organisé et structuré de données qui sont stockées de manière cohérente et accessible. Elle est conçue pour permettre la gestion, le stockage, la recherche et la récupération efficace des informations. Une base de données peut contenir différents types de données, tels que des textes, des chiffres, des images, de graphes des fichiers multimédias, etc.

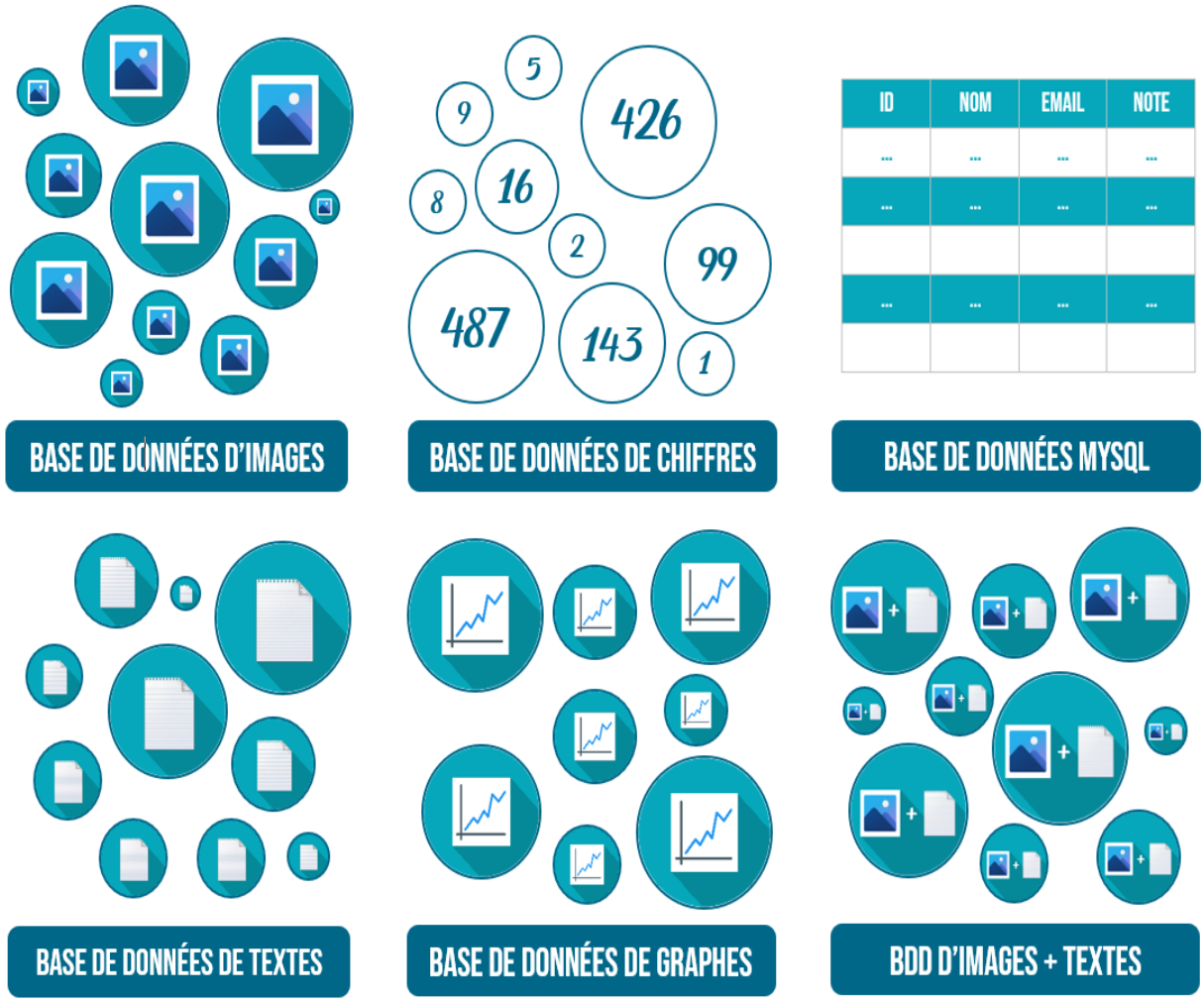


Figure III.1 : Différents types de bases de données

III.2.1. Bases de données clés pour la détection d'objets

Les bases de données jouent un rôle crucial dans le domaine de la détection d'objets. Elles fournissent parfois des jeux de données annotés qui peuvent servir de référence pour l'entraînement et l'évaluation des modèles de détection d'objets. Parmi les bases de données les plus utilisées et les plus influentes, on peut citer ls suivantes.

III.2.1.1. MSCOCO

MSCOCO [24] est largement reconnue comme l'une des bases de données de référence pour la détection d'objets. Elle comprend des images annotées avec des étiquettes d'objets couvrant plusieurs catégories différentes, ainsi que des informations sur les contours des objets et les relations spatiales.

III.2.1.2. Pascal VOC

Pascal VOC [25] est aussi une base de données incontournable dans le domaine de la détection d'objets. Elle offre un ensemble diversifié d'images annotées avec des informations sur différentes classes d'objets, notamment des personnes, des voitures, des animaux, etc.

III.2.1.3. ImageNet

ImageNet [26] est une base de données emblématique dans le domaine de la vision par ordinateur et de la détection d'objets. Elle contient des millions d'images annotées représentant des milliers de catégories d'objets. Elle a été essentielle pour les progrès de l'apprentissage en profondeur, permettant le développement de modèles de détection d'objets extrêmement précis.

III.2.1.4. KITTI

KITTI [27] est reconnue pour sa précision et sa pertinence dans le domaine de la détection d'objets liés à la conduite automobile. Elle comprend des séquences vidéo, des données lidar, des données GPS qui sont une ressource précieuse pour l'évaluation des algorithmes pour les tâches de localisation et détection d'objets.

III.2.1.5. SUN

La base de données SUN [28] constitue une ressource importante pour la recherche sur la compréhension des scènes et la perception visuelle globale. Elle a été utilisée pour des tâches telles que la compréhension des scènes, la navigation autonome, la reconnaissance d'activités et la prédiction de l'esthétique des images.

III.2.1.6. Open Images

Open Images est une base de données d'images en ligne et librement accessible, développée par Google. Elle contient un vaste ensemble d'images provenant de diverses sources, avec des annotations détaillées sur les objets présents dans les images.

III.2.1.7. Wider Face

La base de données Wider Face [29] est une base de données largement utilisée dans le domaine de la détection de visages. Elle est principalement utilisée pour l'évaluation et la comparaison des algorithmes de détection de visages.

Le tableau (III.1) permet de représenter les différences entre les bases de données cités précédemment. Ainsi, il offre une comparaison visuelle des caractéristiques et des spécificités de chaque base de données permettant de mettre en évidence leurs points forts et leurs différences distinctives.

Tableau III.1 : Tableau comparatif des bases de données clefs de la détection d'objets

Base de données	Nombre d'images	Catégories	Annotations	Domaine d'application
MSCOCO	328 000	80	Oui	Détection d'objets en général
Pascal VOC	~ 20 000	20	Oui	Détection d'objets en général
ImageNet	14 197 122	21 841	Non	Détection d'objets en général
KITTI	~ 7 500		Oui	Perception pour les véhicules autonomes
SUN	131 067		Non	Reconnaissance de scènes
Open Images	+ 9 000 000	19 958	Oui	Détection d'objets en général
Wider Face	+ 32 000		Oui	Détection de visages

III.2.2. Conception de la base de données YAHINE

Dans le cadre de notre projet de fin d'études, nous avons pris la décision de créer une base de données novatrice que nous avons nommée YAHINE.

Cette base de données représente l'aboutissement de nos efforts, de nos recherches et de nos analyses approfondies. Notre objectif est de construire une ressource complète et précieuse qui servira de référence dans notre projet de fin d'études.

III.2.2.1. Objectifs de création de YAHINE

Notre base de données de détection d'objets peut être considérée comme moins fiable par rapport aux bases de données clés existantes, mais elle revêt une importance capitale en tant que sujet d'étude. Car, sa création présente plusieurs objectifs et offre de nombreux avantages.

1. La création de cette base de données nous permettra de mieux comprendre les défis et les nuances associés à la collecte et à l'annotation de données pour la détection d'objets. Nous

pourrons ainsi acquérir une expérience pratique précieuse dans le processus de création d'une base de données.

2. En créant notre propre base de données, nous pouvons personnaliser les catégories d'objets, les annotations et les formats selon nos besoins spécifiques. Cela nous permettra de mieux cibler et de créer une base de données adaptée à notre objectif qui est là détection des objets égarés d'intérieur.
3. Avec cette base de données, nous pourrons valider et évaluer l'efficacité de nos méthodes de détection d'objets, et ainsi personnaliser à notre guise pour corriger les potentiels manques liées à cette dernière, et avec ceci acquérir une compréhension approfondie des principaux défis et des subtilités inhérents à la collecte et à l'annotation de données pour la détection d'objets pour mieux appréhender les complexités du processus et d'améliorer notre expertise dans ce domaine.
4. Notre base de données bien qu'elle ne soit pas très performante n'empêche qu'elle peut toujours apporter une contribution significative à la communauté de recherche, en partageant nos résultats et en rendant notre base de données accessible, d'autres chercheurs pourront s'en servir comme point de référence, mener des comparaisons et enrichir les connaissances collectives.

III.2.2.2. Processus de collecte d'images

Dans le processus de collecte d'images pour notre base de données, nous avons principalement utilisé deux smartphones avec une résolution d'images de 4624×2136 pixels et 4160×3120 pixels, ainsi qu'une webcam d'ordinateur portable offrant une résolution de 1280×720 pixels. Pour garantir une base de données compacte et facilement exploitable, toutes les images ont été redimensionnées en une taille uniforme de 500×375 pixels. Cette étape de redimensionnement nous permet de créer une base de données qui est à la fois compacte et peu encombrante, tout en préservant les informations essentielles pour la détection d'objets.

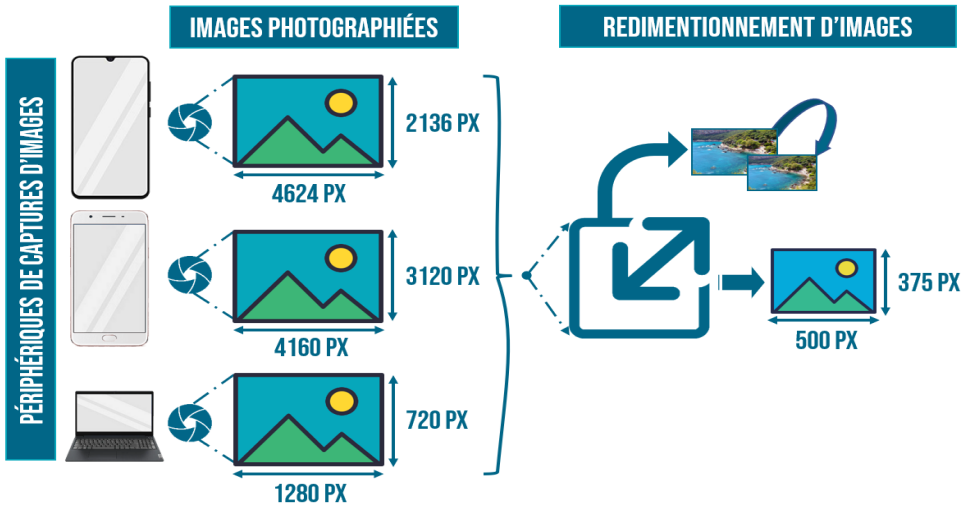


Figure III.2 : Illustration du processus de collection et redimensionnement d'images

III.2.2.3. Annotations d'images

Comme mentionné précédemment, nous avons créé une base de données axée sur les objets mobiles que l'on a tendance à égarer fréquemment. Notre objectif principal était de développer un algorithme capable de détecter ces objets spécifiques. Les classes d'objets que nous avons incluses dans notre base de données sont alors.

- **Téléphone** : En raison de son utilisation régulière, il est facile de le poser quelque part sans se souvenir exactement de l'endroit où nous l'avons laissé.
- **Lunettes** : Il est facile de les oublier ou de les poser dans des endroits peu évidents. De plus, elles peuvent ainsi se perdre facilement lorsqu'elles sont retirées et posées négligemment.
- **Clés** : En raison de leur petite taille et de notre routine quotidienne souvent chargée, il est courant de les poser négligemment ou de les égarer dans des endroits inattendus.
- **Portefeuille** : De par son contenu précieux, il est souvent manipulé et peut être posé ou égaré dans différents endroits de plus il peut facilement se glisser dans des endroits ou se mélanger à d'autres objets.
- **Télécommande** : Il est courant de les égarer dans les coussins du canapé, sous les coussins ou de les poser dans des endroits peu évidents où elles peuvent se perdre facilement

En rassemblant les images prises et annotées de ces objets, nous avons créé un ensemble de données spécifique à notre domaine d'intérêt, facilitant ainsi l'entraînement et l'évaluation de notre algorithme de détection d'objets ciblés.

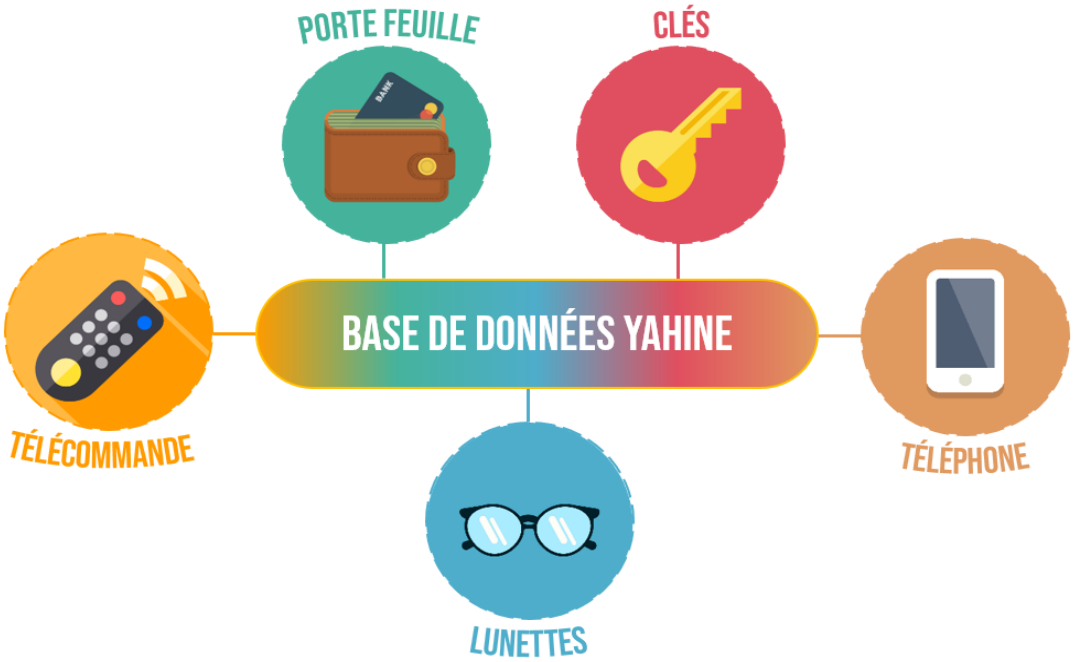


Figure III.3 : Illustration de la base de données YAHINE

Nous avons choisi d'utiliser le logiciel LabelImg afin d'assigner les annotations à chaque image.

III.2.2.3.1. LabelImg

LabelImg [30] est un outil open source utilisé pour annoter des objets dans des images. Il offre aux utilisateurs la possibilité de créer des boîtes de délimitation autour des objets d'intérêt présents dans une image et d'assigner des étiquettes à ces boîtes.

L'interface de LabelImg facilite l'annotation de manière intuitive et conviviale. Elle est illustrée visuellement par Figure III.4. Elle se compose de plusieurs outils qui ont un rôle spécifique dans le processus d'annotation.

1. Choix du format des annotations (YOLO, PascalVOC, etc).
2. Ajout d'une nouvelle zone de délimitation.
3. Modification des zones de délimitation (largeur, hauteur et emplacement) d'une boite déjà existante.
4. Vue d'ensemble des objets et leurs annotions présentes dans l'image.
5. Aperçu des images du dossier sélectionné pour l'annotation.
6. Données d'annotation incluant les dimensions (largeur et hauteur) et la position (coordonnées cartésiennes) de la boite de délimitation dans l'image.
7. Boite de délimitation de l'objet (Téléphone pour cet exemple).

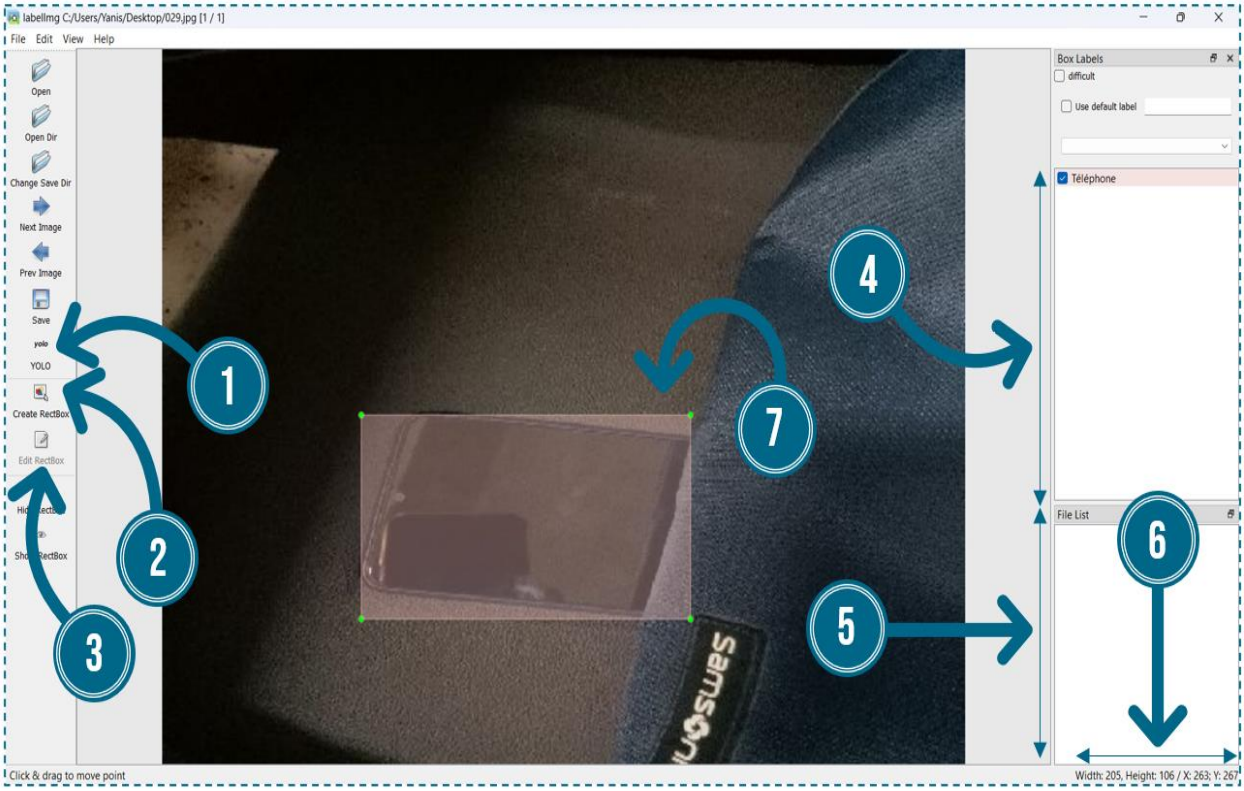


Figure III.4 : Interface d'utilisation de LabelImg

III.2.2.3.2. Format YOLO

Après avoir sélectionné le format YOLO dans LabelImg et annoté une image, puis une fois qu'on enregistrera l'image, le logiciel générera automatiquement un fichier .txt portant le même nom que l'image (par exemple, si l'image est nommée 029.jpg, le fichier d'annotation sera nommé 029.txt). À l'intérieur du fichier texte, les annotations seront au format YOLO.

Dans ce format chaque annotation est représentée par une ligne dans un fichier texte. Chaque ligne contient des informations séparées par des espaces :

[Classe_de_l'objet] [Centre_X] [Centre_Y] [Largeur_de_la_boîte] [Hauteur_de_la_boîte]

- **Classe de l'objet** : Il s'agit de l'indice de la classe dans le répertoire, pour notre base de données c'est réparti de la manière suivante :
  - 0 : Téléphone
  - 1 : Lunettes
  - 2 : Portefeuille
  - 3 : Clés
  - 4 : Télécommande
- **Centre X** : Ce sont les coordonnées x du centre de la boîte de délimitation de l'objet exprimée en proportions par rapport à la largeur et la hauteur de l'image (normalisées).
- **Centre Y** : Ce sont les coordonnées y normalisées du centre de la boîte de délimitation de l'objet.
- **Largeur de la boite** : C'est la valeur de la largeur de la boite normalisée.
- **Hauteur de la boite** : C'est la valeur de la hauteur de la boite normalisée.

Pour calculer les valeurs normalisées de l'image présente dans la figure (III.4) il nous suffit simplement de prendre les dimensions de notre image, dans notre base de données nous avons utilisées des images de taille 500×375 pixels, Alors $x_{max} = 500$ et $y_{max} = 375$.

Pour obtenir les coordonnées du centre du rectangle de délimitation en format YOLO, nous allons diviser les valeurs du point ($x_{centre} = 224$ et $y_{centre} = 249$) par les dimensions de l'image. Ainsi, nous obtenons le résultat suivant :

$$Centre_X = \frac{x_{centre}}{x_{totale}} = \frac{224}{500} = 0.448 \qquad Centre_Y = \frac{y_{centre}}{y_{totale}} = \frac{249}{375} = 0.664 \qquad \text{(III.1)}$$

Quant à la largeur et hauteur de la boîte en format YOLO, c'est quasiment la même chose que le centre, nous allons prendre les dimensions du rectangle ($hauteur = 113$ et $largeur = 204$) et le divisé sur les dimensions de l'image, ce qui nous donnera le résultat suivant :

$$largeur_{YOLO} = \frac{largeur}{x_{totale}} = \frac{204}{500} = 0.408 \qquad hauteur_{YOLO} = \frac{largeur}{y_{totale}} = \frac{113}{375} = 0.304 \qquad \text{(III.2)}$$

Dans le fichier texte associé à cette image, nous trouverons une ligne directe qui rassemblera toutes les valeurs calculées précédemment comme ceci : 0 0.448 0.664 0.408 0.304.

III.2.2.4. Caractéristiques de la base de données YAHINE

La base de données YAHINE est une base de données d'annotations et d'images qui ont été prises par nous et soigneusement sélectionnées pour représenter les divers objets. Les caractéristiques clés de la base de données sont résumées dans le tableau (III.2).

Tableau III.2 : Caractéristiques de la base de données YAHINE

Caractéristique	Valeur
Nombre d'images	~ 5200
Nombre de classes	5
Classes	Téléphone, lunettes, clés, portefeuille et télécommande
Répartition par classe	~ 1000
Taille des images	500 × 375 Pixels
Format des fichiers	JPG pour les images, TXT pour les annotations
Canaux de couleurs	RVB (Rouge, Vert, Bleu)
Source des images	Nos propres photos
Qualité des images	40% de la qualité originale

III.3. Conclusion

A travers ce chapitre, On a suscité l'intérêt quant aux caractéristiques de notre base de données spécialement conçue pour capturer des images représentatives d'environnements intérieurs à travers différentes étapes de collecte de données. Ces étapes sont cruciales pour la conception d'une base de données suffisante pour mener à bien l'entraînement et l'évaluation de notre modèle de détection. L'objectif est de proposer une solution pour retrouver des objets du quotidien souvent égarés.

CHAPITRE 4

ENTRAINEMENT DE YOLOV4 SUR LA BASE DE DONNÉES YAHINE

Récapitulatif

Ce chapitre explore les outils logiciels et les plateformes dédiés au traitement d'images pour créer un modèle de détection et de reconnaissance d'objets d'intérieur. Nous utilisons le modèle pré-entraîné YOLOv4 sur notre base de données YAHINE. Les résultats sont analysés en comparant l'approche avec et sans augmentation de données pour évaluer l'efficacité et la précision du modèle.

IV.1. Introduction

Dans le présent chapitre, nous avons d'abord exploré les outils logiciels et les plateformes dédiés au traitement d'images. L'objectif est de créer un modèle de détection et de reconnaissance d'objets de l'intérieur. Dans cette optique, nous avons utilisé le modèle pré-entrainé de YOLOv4 qui repose sur le concept de l'apprentissage profond et des réseaux de neurones convolutifs sur notre base de données YAHINE. On a procédé ensuite à l'analyse des résultats obtenus en comparant deux approches, l'une avec et l'autre sans ajout de données, afin d'évaluer l'efficacité et la précision de notre modèle.

IV.2. Présentation des outils et plateformes

Avant de procéder à l'entraînement de notre algorithme, il est essentiel de passer en revue les différents outils et plateformes qui seront indispensables pour mener à bien cette tâche.

IV.2.1. GitHub

GitHub est un portail en ligne qui se présente sous la forme d'une plateforme de développement collaborative et d'un hébergeur de projets logiciels. Il permet aux développeurs de télécharger et de partager leurs programmes directement via la plateforme.

Cette plateforme sera extrêmement bénéfique pour notre projet, car c'est là que les développeurs de YOLOv4 ont mis à disposition leur algorithme officiel. Par conséquent, nous pourrons simplement le télécharger à partir de cette plateforme.

IV.2.2. Google Drive

Google Drive est un service en ligne offert par Google qui permet aux utilisateurs de sauvegarder, partager et accéder à leurs fichiers et documents depuis n'importe quel appareil connecté à Internet. Il offre une solution pratique pour la conservation sécurisée des données, la collaboration en temps réel et la synchronisation de fichiers sur différents appareils.

Dans notre cas, Google Drive sera utilisé en tant qu'espace de stockage pour l'entièreté de notre projet, à savoir du modèle YOLOv4, de notre base de données, du résultat de notre entrainement, etc.

IV.2.3. Python

Python est un langage de programmation interprété et polyvalent, crée en 1991 et réputé pour sa lisibilité et sa simplicité syntaxique. Il est largement utilisé dans le domaine de la détection d'objets en raison de ses bibliothèques riches et de sa popularité croissante dans le domaine de l'apprentissage automatique et de la vision artificielle.

Il dispose de fonctions et fonctionnalités capable facilement d'accéder à darknet qui est un framework populaire pour la détection d'objets, notamment grâce à son implémentation de l'algorithme YOLO.

Python est principalement utilisé pour charger des modèles pré-entraînés, effectuer des prédictions sur des images ou des vidéos, et récupérer les résultats de détection d'objets, ainsi que pour **le prétraitement** des données avant de les passer à Darknet, et finalement aussi le **post-traitement** et **la visualisation** des résultats de détection.

IV.2.4. Google Colab

Google Colab est un environnement de développement et de calcul basé sur le cloud et sur Jupyter Notebook. Il est destiné à la formation et à la recherche dans l'apprentissage automatique et permet aux utilisateurs d'exécuter du code Python directement depuis leur navigateur web. Il fournit un accès gratuit à une machine virtuelle avec une puissance de calcul décente, ainsi qu'à certains frameworks et bibliothèques populaires, tels que Darknet, OpenCV, etc.

Il est facilement accessible avec un compte Gmail. On y accéder en directement via le lien officiel de Google Colab : https://colab.research.google.com. Puis une fois connecté au Gmail, on peut alors accéder un espace de travail et commencer la programmation.

Nous avons délibérément choisi d'utiliser Google Colab pour notre projet en raison de ses avantages en termes de performances. Cependant, on peut aussi exécuter l'algorithme YOLOv4 via notre propre ordinateur, mais il est recommandé d'avoir une GPU NVIDIA GeForce RTX série 20 ou un autre GPU plus performant. Tandis qu'avec Colab, on peut directement utilisés ses performances sans se soucier de la puissance de notre carte graphique. Les caractéristiques mis à disposition de Colab [31] connues à ce jour pour la version gratuite sont représentées dans le tableau (IV.4).

Tableau IV.1 : Caractéristiques de Google Colab

	CPU	GPU	Nombre de coeurs	VRAM	Mémoire Graphique	Stockage Interne	Logo
Google Colab	Intel Xeon	NVIDIA Tesla K80 (Repartie sur 2 GPU GK210)	4492 (2496×2)	24 Go (12×2)	GDDR5	75 Go	

Google Colab permet aux utilisateurs d'accéder aux GPU NVIDIA, ce qui accélère les calculs pour les tâches exigeant une grande puissance de calcul, dans notre projet.

IV.2.4.1. CUDNN

CuDNN est bibliothèque logicielle développée par NVIDIA et joue un rôle essentiel dans l'utilisation de CUDA dans le domaine de l'apprentissage automatique lors de l'entraînement de réseaux de neurones profonds [32].

IV.2.4.2. CUDA

CUDA est une plateforme de calcul parallèle développée par NVIDIA. Elle permet d'utiliser la puissance de calcul des GPU NVIDIA pour exécuter des tâches de calcul intensif de manière hautement parallèle.

CuDNN s'intègre à CUDA grâce à LIBSO pour accélérer les calculs nécessaires aux opérations matricielles et aux calculs intensifs effectués dans ces réseaux profonds en exploitant les fonctionnalités de calcul parallèle des GPU NVIDIA [32].

IV.2.4.3. LIBSO

LIBSO est un fichier binaire contenant le code compilé d'une bibliothèque logicielle. Il est utilisé pour lier la bibliothèque à d'autres applications ou projets. Il permet ainsi de faire la liaison entre CuDNN et CUDA permettant ainsi aux développeurs d'accéder aux fonctionnalités de calcul parallèle des GPU NVIDIA via cuDNN dans leurs applications d'apprentissage profond afin de pouvoir faire appel à OpenCV pour lire, manipuler, afficher et effectuer des opérations de traitement d'images en temps réel avec une faible latence.

IV.2.4.4. OpenCV

OpenCV est une bibliothèque open source populaire utilisée pour le traitement d'images et la vision par ordinateur. Elle offre une large gamme de fonctionnalités pour la manipulation d'images, le traitement d'images, la détection d'objets, la reconnaissance faciale, la vision artificielle. Les bibliothèques OpenCV peuvent également être compilées en tant que bibliothèques partagées (LIBSO) pour une utilisation efficace dans les applications qui requièrent des opérations de traitement d'images rapides et optimisées [33].

OpenCV met à disposition dans le domaine du traitement d'images et de la vision artificielle une large gamme de fonctionnalités et de capacités. Le tableau (IV.5) représente une petite sélection des caractéristiques qu'elle offre.

Tableau IV.2 : Caractéristiques d'OpenCV

	Nombre d'algorithmes	Nombre de fonctions	Domaines d'application	Compatibilité des formats	Logo Officiel
OpenCV	2500 Algorithmes de vision artificielle	500 Fonctions d'opérations sur les images (filtrage, transformation, redimensionnement, la segmentation)	Robotique, réalité augmentée, détection d'objets, traitement d'images médicales, etc.	JPEG, PNG, BMP, AVI, MPEG…	OpenCV

IV.3. Création du modèle YOLOV4 basé sur YAHINE

Avant d'entamer la configuration de notre modèle YOLOv4, il est essentiel de se connecter à Google Colab en utilisant notre compte Google Drive personnel et sélectionner l'option **GPU** sous **Accélérateur matériel** dans **Matériel d'exécution** qui se trouve dans situé l'onglet **Durée**.

Ensuite, nous allons faire importer notre drive à l'aide de la bibliothèque « **Drive** » de python puis cloner le dépôt du framework darknet dans ce dernier. Par la suite on a importé notre base de données comportant les images et annotations associés ainsi que fichier « *classes.txt* » pour spécifier les noms des classes d'objets que le modèle de détection doit reconnaître. Chaque ligne du fichier correspond à un nom de classe unique dans le dossier data se trouvant à l'intérieur de Darknet et qui fera office de répertoire utilisé pour stocker les données nécessaires à l'entraînement et à l'évaluation des modèles de détection d'objets et que l'on nommera « *custom_data* ». Après cela s'en suivra la configuration minutieuse de l'algorithme.

IV.3.1. Prétraitement de l'algorithme

Nous allons commencer le prétraitement en modifiant le fichier Makefile qui est fourni avec le code source de Darknet. Nous allons modifier les lignes contenant OpenCV, GPU, CUDNN, CUDNN_HALF et LIBSO en les initialisant à 1, ainsi nous passerons de leur état désactivé à leur état activé.

Ensuite, on exécute la commande « *!make* » pour lire le fichier Makefile fourni avec le code source et exécuter les instructions nécessaires pour compiler le code et générer un exécutable.

Après avoir vérifié que les annotations sont au bon format et qu'elles sont cohérentes avec les images, nous procéderons à la division de nos images en deux ensembles distincts. Un premier ensemble comprendra 75% du total des images et sera destiné à l'entraînement, tandis que le second ensemble, contenant les 25% restants, sera réservé aux tests.

Les images contenues dans la partie entrainement sont utilisées pour ajuster les paramètres du modèle afin qu'il puisse apprendre à reconnaître et à détecter les objets de notre étude. Quant à la partie test, les images seront utilisées pour évaluer les performances du modèle après son entraînement. Les performances du modèle sur l'ensemble de test fournissent des informations sur sa précision qui sera très utile pour voir la fiabilité de notre modèle.

Une fois la division effectuée, nous attribuerons les chemins des images à deux fichiers distincts. Le premier fichier, nommé « *train.txt* », contiendra les chemins des images destinées à l'entraînement. Le deuxième fichier, nommé « *test.txt* », contiendra les chemins des images destinées aux tests. Ensuite, nous placerons ces fichiers dans le dossier « *custom_data* ».

Ensuite, on crée le fichier « *classes.names* » qui est exactement le même fichier que « **classes.txt** » mais sous un format différent qui est spécifique à YOLO et est utilisé pour spécifier les noms des classes d'objets dans le cadre de l'architecture YOLOv4.

Il nous restera alors que la création de « *labelled_data.data* » qui sert essentiellement de fichier de configuration centralisé pour spécifier les détails nécessaires à l'entraînement du modèle. Il contient des informations importantes telles que les chemins d'accès vers le fichier d'entraînement, test, classes, le chemin où seront sauvegarder les poids, ainsi que le nombre de classes.

Toutes les étapes mentionnées précédemment, y compris la création du dossier "custom", peuvent être visualisées avec une vue d'ensemble du contenu de ce dossier à l'aide de l'illustration de Figure IV.1. Cela permet d'avoir une vision complète de toutes les étapes réalisées jusqu'à présent, ainsi que des fichiers et des répertoires présents dans le dossier « *custom_data* ».

Une fois que toutes les étapes précédentes ont été finalisées, la prochaine étape consistera à entraîner notre modèle afin de créer le fichier de poids. Ces poids seront utilisés ultérieurement pour évaluer les performances de notre modèle telles que la précision de la détection d'objets.

L'entraînement du modèle est une étape cruciale où le modèle apprend à reconnaître et à détecter les objets spécifiques que nous avons définis dans notre jeu de données.

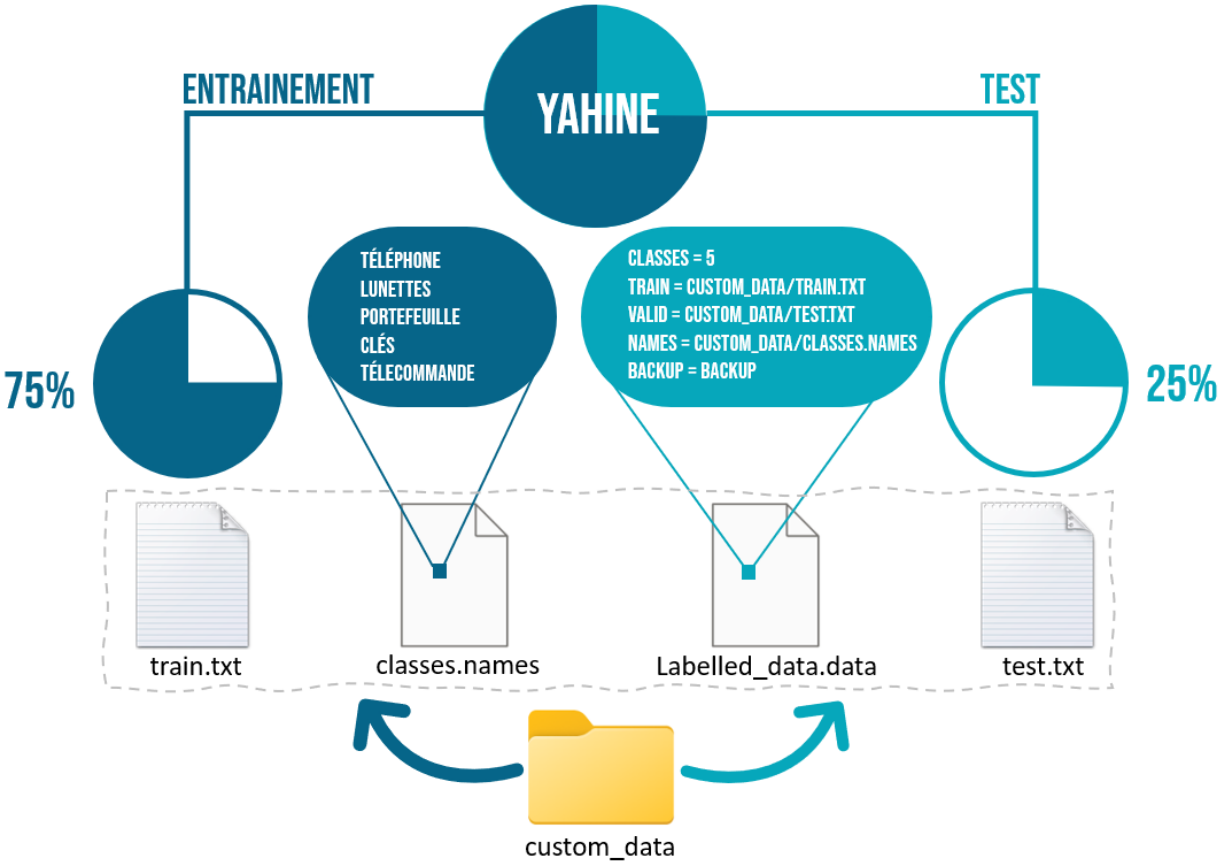


Figure IV.1 : Constitution du fichier custom_data

IV.3.2. Entrainement de l'algorithme YOLOV4

Nous allons commencer cette partie par la configuration du fichier « *yolov4-custom.cfg* » qui se trouve dans le dossier « *cfg* », les étapes générales pour configurer le fichier yolov4-custom.cfg sont :

IV.3.2.1. Configuration des paramètres du réseau de neurones

- **Batch** : Un batch est un groupe d'échantillons prises de la partie entraînement et sont utilisés pour mettre à jour les poids du modèle. Dans notre fichier, il indique la taille du lot prit lors de l'entraînement. Le cas idéal utilisé pour notre modèle est de prendre chaque lot de 64 images, c'est-à-dire on va mettre à jour le fichier poids après avoir calculé les gradients moyens sur un groupe de 64 échantillons d'entraînement à la fois.
- **Subdivisions** : il est courant de diviser le lot (batch) en mini-lots plus petits afin d'optimiser l'utilisation de la mémoire et d'accélérer le processus d'entraînement. Dans notre fichier, il définit le nombre de subdivisions dans lesquelles le lot est divisé. Ici, notre lot de 64 images sera divisé en 16 mini-lots, chacun contient 4 images.
- **Width** : Correspond à la largeur des images d'entrée lors de l'entrainement. On peut choisir de redimensionner l'entièreté des images pour réduire le temps de calculs. La valeur de la nouvelle largeur doit être un multiple de 32. Dans notre cas nous nous sommes penchés vers la valeur de 416.
- **Height** : Correspond à la hauteur des images d'entrée lors de l'entraînement du modèle. Même cas que le width, nous avons aussi utilisé un multiple de 32, soit 416.
- **Channels** : C'est le nombre de canaux de couleur de l'image d'entrée. Du fait que nous avons utilisés des images en couleurs RVB standard, la valeur de channels doit être de 3.
- **Momentum** : Momentum est une technique d'optimisation couramment utilisée pour accélérer la convergence du modèle pendant l'entraînement, Dans le fichier configure la valeur du paramètre momentum varie entre 0 et 1. Une valeur plus proche de 1 donnera plus d'importance aux gradients précédents, ce qui peut aider à accélérer la convergence du modèle, pour notre modèle on a opté pour une valeur de 0.949.
- **Decay** : Le paramètre "decay" régule l'ampleur de la régularisation L2 en déterminant le niveau de pénalisation appliqué aux poids du modèle. Lorsque la valeur de "decay" est élevée, la pénalisation des poids est plus prononcée, tandis qu'une valeur plus faible réduit l'effet de la régularisation. La valeur que l'on a utilisée est de 0.0005.

IV.3.2.2. Configuration des paramètres pour l'augmentation de données

- **Angle :** C'est le degré des rotations qui seront appliquées aux images pendant l'entrainement. Nous avons utilisé la valeur de 10 qui correspond à 10 ° car il est important de trouver un équilibre pour éviter des rotations excessives qui pourraient altérer la qualité et la pertinence des données d'entraînement.

- **Saturation :** C'est un paramètre qui permet de modifier l'apparence et la vivacité des couleurs des images pendant l'entraînement. Il permet d'augmenter la variabilité des données d'entraînement. Ce qui peut améliorer la robustesse du modèle face à des variations de couleurs dans les images réelles. Une valeur supérieure à 1 augmentera la saturation des couleurs, ce qui intensifiera les teintes présentes dans l'image. En revanche, une valeur inférieure à 1 réduira la saturation des couleurs, rendant les couleurs plus pâles ou plus grises. On fixe une valeur de 1.5.
- **Exposure :** C'est l'exposition d'une image à la lumière, Il représente la quantité de lumière qui y est présente. En ajustant l'exposition, on peut rendre les images plus lumineuses ou plus sombres. Une valeur plus grande que 1 augmentera l'exposition des images aux lumières les rendant plus lumineuse et une valeur inférieure à 1 diminuera l'exposition, ce qui assombrira les images. Dans notre cas, on a mis 1.2 afin d'améliorer la capacité du modèle à s'adapter à des conditions d'éclairage variables dans des scénarios réels.
- **Hue** : C'est un paramètre qui permet de modifier la teinte des couleurs dans les images pendant l'entraînement, Il contrôle l'amplitude des ajustements de teinte appliquée aux images pendant la data augmentation. Il spécifie l'intervalle de variation de la teinte en pourcentage, On a adopté pour une valeur de 0.1 soit 10% car la teinte représente la caractéristique de couleur dominante d'une image. En ajustant la teinte, on peut modifier les couleurs présentes dans une image afin d'améliorer la capacité du modèle à généraliser et à détecter des objets dans différentes nuances de couleurs.

IV.3.2.3. Configuration des paramètres d'optimisation

- **Learning_rate** : Il représente Le taux d'apprentissage. Un taux élevé permet des modifications importantes, tandis qu'un taux bas permet des ajustements plus fins. On a opté pour une valeur faible de 0.001 afin de ne pas trop avoir des sauts brusques dans les valeurs des poids du modèle.
- **Burn_in** : Il s'agit du nombre d'itérations initiales pendant lesquelles le taux d'apprentissage sera augmenté progressivement jusqu'à atteindre sa valeur désirée. On a délibérément choisi une valeur de 1000. Car cela permet d'assurer une convergence plus rapide du modèle au début de l'entraînement.
- **Max_batches** : C'est le nombre total d'itérations d'entraînement que le modèle effectuera. Une itération correspond à la présentation d'un batch d'images au modèle pour effectuer une mise à jour des poids. Il existe une formule pour le calculer :

$$Max_{batches} = nombre\ de\ classes \times 2000 = 5 \times 2000 = 10000 \qquad (IV.1)$$

- **Policy** : indique la politique de variation du taux d'apprentissage. Ici, la politique utilisée est basée sur des étapes.

- **Steps** : Ce sont les étapes spécifiques auxquelles le taux d'apprentissage sera ajusté. L'idéal est de choisir 80% et 90% de la valeur du « **max_batches** ». Alors, dans notre modèle, le taux d'apprentissage sera réduit à l'étape 8000 et sera ensuite réduit à l'étape 9000.
- **Scales :** Ce sont les facteurs de réduction du taux d'apprentissage à chaque étape spécifiée. On a choisi la valeur de 0,1 qui indique que le taux d'apprentissage sera réduit de 10% à chaque étape (8000 et 9000).

IV.3.2.4. Configuration des paramètres de convolution

- **Batch_normalize** : indique si oui ou non on veut activer la normalisation par lot dans notre modèle (1 pour l'activer et 0 pour la désactiver). Dans notre cas, elle est activée donc 1.
- **Filters** : Ce paramètre spécifie le nombre de filtres utilisés dans la couche. Chaque couche de convolution possède un nombre spécifique de filtres qui est déterminé par la conception du réseau. Dans notre cas, nous l'avons laissé comme il était de base soit : 32,64, …,1024. il est recommandé de choisir le nombre de filtres pour la dernière couche de façon à ce que la valeur

$$Filtres = (nombre\ de\ classes + 5) \times 3 = 10 \times 3 = 30 \qquad (IV.2)$$

- **Size** : fait référence à la taille du noyau de convolution utilisé dans chaque couche spécifique du modèle. Pour notre modèle, On a fixé un noyau de taille 3×3.
- **Stride** : indique le décalage entre les positions du noyau lors de la convolution. Le stride de notre modèle est défini sur 1. Cela signifie que le noyau se déplace d'un pas d'une unité à la fois lors de la convolution. Autrement dit, le noyau se déplace horizontalement et verticalement en se décalant d'un pixel à chaque itération.
- **Pad** : indique le nombre de zéros ajoutés autour de l'image d'entrée. Le pad de notre modèle est de 1. Cela signifie qu'un Padding d'une unité est appliqué de chaque côté de l'image d'entrée. Cela signifie qu'une ligne et une colonne de zéros sont ajoutées autour de l'image. Ce qui augmente la taille de l'image de 2 unités à la fois en hauteur et en largeur.

Ainsi il nous restera plus qu'à modifier le nombre classes à 5 dans la partie [yolo] situé à la ligne 1058 du fichier « *yolov4-custom.cfg* ».

Il nous reste plus qu'à donner l'autorisation d'exécuter le fichier exécutable « *darknet* » avec la commande :

```
!chmod +x ./darknet
```

Après avoir terminé ces étapes on a procédé au téléchargement des poids du modèle YOLOv4, « *yolov4.conv.137* », pré-entrainé avec la base de données MS COCO, l'exécution du programme permet de commencer l'entraînement de notre modèle. Nous n'avons plus qu'à configurer les emplacements de nos fichiers en utilisant la commande prédéfinie spécifique à cet effet.

```
!./darknet detector train ./data/custom_data/labelled_data.data
cfg/yolov4-custom.cfg yolov4.conv.137 -map -dont show
```

Une fois l'entraînement lancé, un texte s'affichera progressivement. Il nous indiquera la progression en temps réel de l'entraînement, y compris la perte, le taux d'apprentissage, le temps écoulé, le nombre d'images utilisées, ainsi que des statistiques spécifiques à chaque région du modèle, qui sont répartis de la manière suivante :

```
2: 1820.553223, 1819.150879 avg loss, 0.000000 rate, 5.425747 seconds, 128 images, 24.458235 hours left

Loaded: 0.000073 seconds

v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.000000), count: 1, class_loss
= 4425.576660, iou_loss = 0.000000, total_loss = 4425.576660

v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.000000), count: 1, class_loss
= 745.354614, iou_loss = 0.000000, total_loss = 745.354614

v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.341182), count: 8, class_loss
= 242.461472, iou_loss = 0.061218, total_loss = 242.522690

 total_bbox = 1594, rewritten_bbox = 0.125471 %
```

Figure IV.2 : Visualisation de l'évolution de l'apprentissage

L'information fournie par l'algorithme corresponds à :

- **2** : indique le numéro d'itération en cours.
- **1820.553223** : représente la perte moyenne de l'itération actuelle.
- **1819.150879 avg loss** : indique la perte moyenne cumulée jusqu'à l'itération actuelle.
- **0.000000 rate** : correspond au taux d'apprentissage utilisé à cette itération.
- **5.425747 seconds** : indique le temps écoulé pour cette itération en secondes.
- **128 images** : représente le nombre d'images utilisées pour cette itération.
- **24.458235 hours left** : estime le temps restant en heures pour terminer l'entraînement.

Ensuite, on retrouve plusieurs lignes commençant par "v3", qui donnent des informations spécifiques sur chaque couche du modèle :

- **avg (IOU : 0.000000)** : indique la moyenne de l'indice de Jaccard pour les prédictions de cette couche.
- **count** : le nombre d'objets détectés dans cette couche.
- **class_loss** : la perte due à la classification des objets.
- **iou_loss** : la perte due à la localisation des objets.
- **total_loss** : la perte totale pour cette couche.

Enfin, les statistiques globales des boîtes de délimitation calculées jusqu'à présent sont affichées, en prenant en compte et en rajoutant les calculs des autres itérations.

- **total_bbox :** le nombre total des boîtes de délimitation générées par le modèle.
- **rewritten_bbox** : le pourcentage des boîtes de délimitations qui ont été réécrites.

IV.3.3. Evaluation des performances

Afin d'évaluer les performances de notre modèle YOLOv4, il faut élaborer une approche combinant des évaluations qualitatives et quantitatives. Ces deux méthodes complémentaires nous permettent d'obtenir une évaluation complète du modèle. Elles fonctionnent de manière synergique pour nous donner une vision globale des performances du modèle.

IV.3.3.1. Evaluation quantitatives

Les évaluations quantitatives consistent à utiliser des mesures numériques et des métriques pour évaluer de manière objective les performances du modèle en évaluant son efficacité globale. Elles sont déterminées à partir d'une matrice de confiance contenant des valeurs numériques telles que :

- **Vrais positifs :** nombre d'objets présents détectés.
- **Vrais négatifs :** nombre d'objets non présents et non détectées
- **Faux positifs :** nombre d'objets non-présents détectés.
- **Faux négatifs :** nombre d'objets réels présents mais pas détectés.

Tableau IV.3 : Mesures de performances

Situations	Objet présent	Objet non-présent
Objet détecté	Vrais positif	Faux positif
Objet non-détecté	Faux négatif	Vrais négatif

Ces indicateurs de performance nous permettent d'avoir une vision globale de l'efficacité de notre modèle impliquant l'utilisation de métriques spécifiques, telles que :

IV.3.3.1.1. Précision

La précision c'est la capacité du modèle à détecter correctement les objets dans une image. Elle mesure le nombre de vrais positifs par rapport à la somme des vrais positifs et des faux positifs (nombre total d'objets détectés). Cela permet d'évaluer la capacité du modèle à produire des résultats précis en minimisant les erreurs de fausses détections. Elle peut directement être calculée par l'équation IV.2.

$$Précision = \frac{Vrais\ positifs}{Vrais\ positifs + Faux\ positifs} \qquad (IV.3)$$

IV.3.3.1.2. Rappel

Le rappel est défini comme le rapport entre le nombre de vrais positifs et la somme des vrais positifs et des faux négatifs (nombre total d'objets présents). Elle peut être calculée directement via l'équation IV.3.

$$Rappel = \frac{Vrais\ positifs}{Vrais\ positifs + Faux\ négatifs} \quad (IV.4)$$

On peut résumer ses deux indicateurs principaux directement avec le schéma de la figure IV.3.

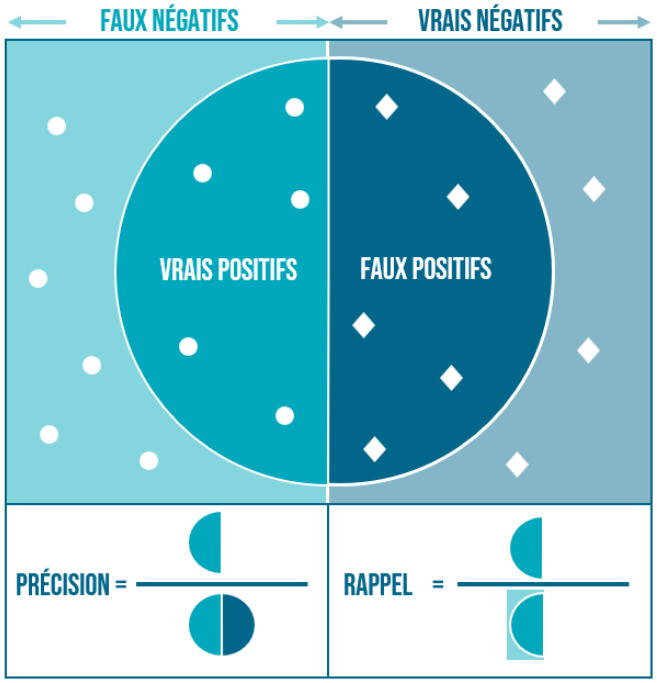
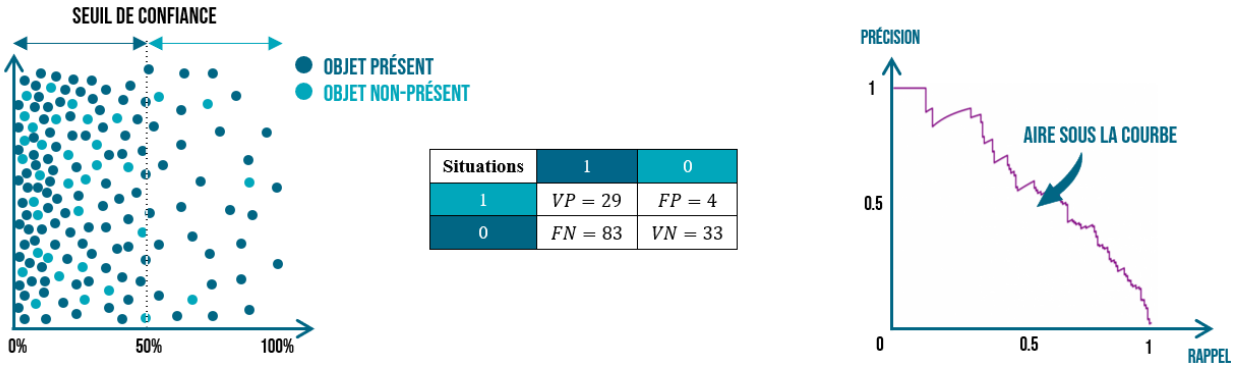


Figure IV.3 : Illustration des indicateurs de performances : Précision et Rappel

IV.3.3.1.3. Précision moyenne (map)

La précision moyenne évalue à la fois la précision et le rappel du modèle sur plusieurs classes d'objets. Elle est calculée en prenant la moyenne des précisions moyennes (AP) pour chaque classe d'objet. Pour calculer l'AP, on trace une courbe de précision-rappel en fonction des seuils de confiance pour chaque classe. Ensuite, l'AP est calculée comme l'aire sous cette courbe.



Situations	1	0
1	$VP = 29$	$FP = 4$
0	$FN = 83$	$VN = 33$

Figure IV.4 : Courbe de précision-rappel pour le calcul de l'AP

La précision moyenne est souvent calculée à l'aide de différentes variantes, telles que mAP@0.5, qui utilise un seuil d'IoU de 0,5 pour considérer une détection comme correcte. Cela permet de fixer un critère de seuil spécifique pour évaluer la précision moyenne.

IV.3.3.1.4. F-Score

Le F-score (aussi appelé F1-score) est utilisé pour évaluer la performance d'un modèle en tenant compte à la fois des vrais positifs, des faux positifs et des faux négatifs**.**

$$F - score = 2 \times \frac{Précision \times Rappel}{Précision + Rappel} \quad \text{(IV. 5)}$$

Le F-Score est calculé à partir de la précision et du rappel du modèle. Il évolue entre 0 à 1, une valeur de 1 représente une performance parfaite du modèle, tandis qu'une valeur de 0 indique une performance très faible**.**

IV.3.3.1.5. Average IoU

L'IoU mesure le degré de chevauchement entre une prédiction de boîte de délimitation et aux annotations de référence qui décrivent la localisation. Elle se calcule, comme son nom l'indique, en divisant l'air de l'intersection sur l'aire de l'union.

IV.3.3.2. Evaluation qualitatifs

Les évaluations qualitatives dans YOLOv4 se basent sur des critères tels que la qualité visuelle des images détectées, la capacité du modèle à détecter des objets de différentes tailles, formes et orientations, la robustesse du modèle face à des conditions d'éclairage et de bruit variées, et la généralisation du modèle à des domaines inconnus ou peu fréquents. Elles sont importantes pour évaluer la performance du modèle dans des situations réelles et complexes qui ne peuvent pas être entièrement capturées par des mesures quantitatives. Elles permettent également d'identifier les points forts et les points faibles du modèle et de proposer des pistes d'amélioration.

On peut résumer tout ce que'on a vu jusqu'à présent dans cette partie avec la figure IV.5 qui explique en détail le dérouler de notre algorithme du début de l'entrainement jusqu'à l'obtention des poids finaux.

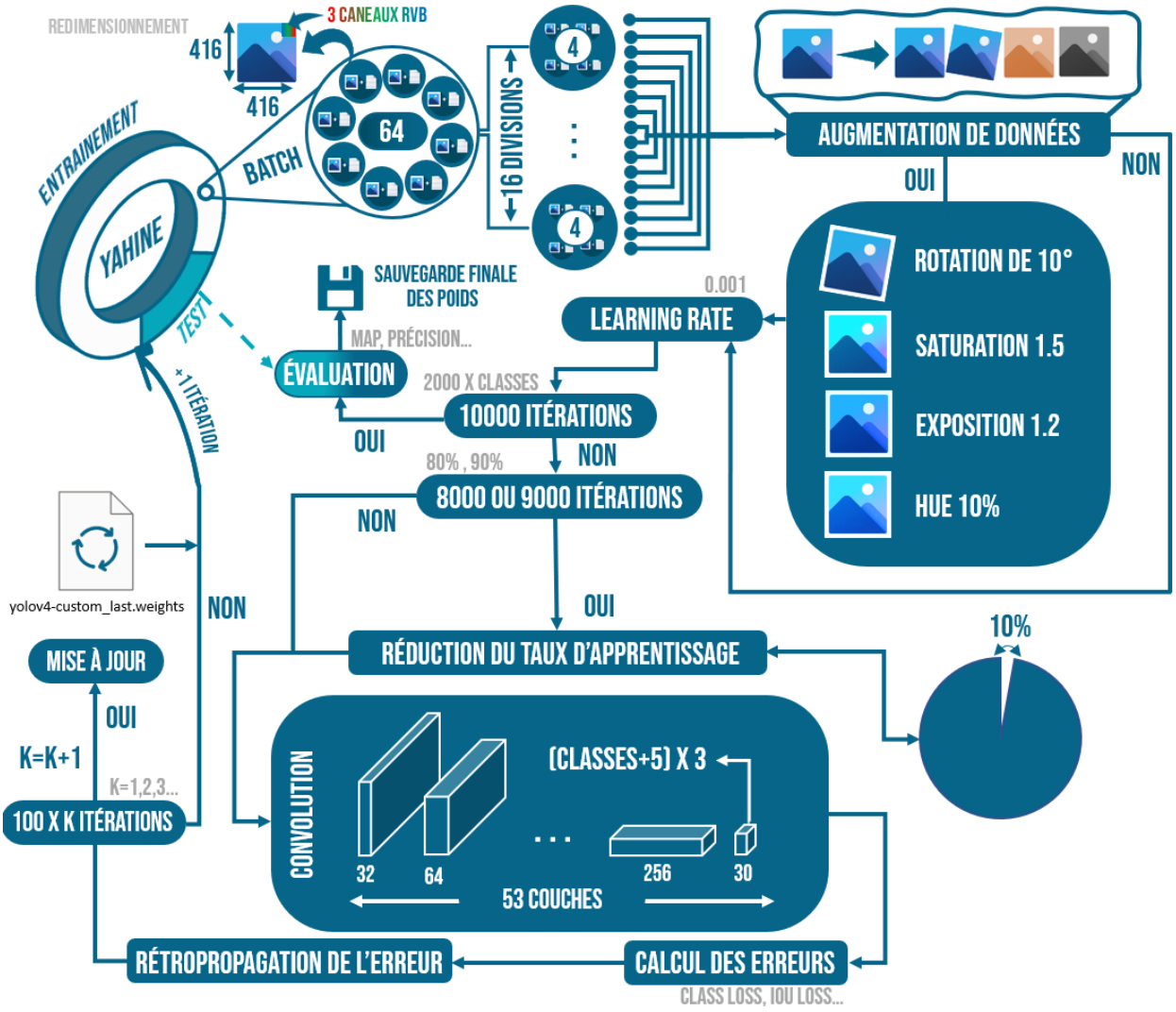


Figure IV.5 : Schématisation globale du processus d'apprentissage

IV.4. Résultats et simulations

Dans cette partie, nous allons simuler tous les résultats que nous avons acquis directement via notre conception. Ensuite, nous allons utiliser la même conception avec l'augmentation de données pour voir si, dans notre cas, il sera bénéfique pour le biais de notre base de données. Donc cette partie sera divisé en 2 parties, une partie sans augmentation qu'on nommera « **Première conception** » et une autre partie avec notre base de données augmentée qu'on appellera « **Deuxième conception** ». On fera par la suite des tests sur des images réels, ainsi que des tableaux comparatifs entre les deux cas, afin de voir de manière explicite la validité et l'efficacité des deux méthodes.

IV.4.1. Première Conception : à base de YAHINE réelle

Dans cette conception comme nous l'avons cité précédemment nous allons directement lancer notre algorithme de YAHINE réelle qui représente la version sans augmentation de données.

Sachant qu'avec 10000 itérations ainsi que les 5200 images de notre base de données et un batch de 64 images, notre algorithme prendra $\frac{5200}{64} = 82$ itérations afin de la parcourir entièrement. Donc avec 10000 itérations l'algorithme fera en moyenne 123 fois les images de notre base de données. On notera alors les résultats après simulations dans les tableaux.

Tableau IV.4 : Statistique des boites de délimitation durant l'apprentissage

Itérations	100	300	500	1000	5000	10000
Images traités	6400	19200	32000	64000	320000	640000
loss	255.89	7.3955	7.04	3.675841	0.993538	0.61541
Avg loss	438.48	7.3955	5.69	3.384097	1.169885	0.874123
Total_bbox	69379	132090	270715	612989	3510820	7421320
Rewritten_bbox	0.21188%	0.195321%	0.195778%	0.176023%	0.171223%	0.1589%

Le tableau IV.4 affiche les statistiques globales des boîtes de délimitation. Il est remarquable de constater une diminution significative de la réécriture des boîtes (Rewritten_bbox) ainsi que de leurs erreurs moyennes. Cette amélioration est due à l'application de l'apprentissage via des réseaux de neurones profonds. Grâce à cette approche, notre algorithme est maintenant capable de distinguer nos objets de manière autonome avec une erreur de plus en plus faible.

Tableau IV.5 : Comparatif des statistiques d'apprentissage de chaque classe

map@0.5 / Itérations	100	300	500	1000	5000	10000
Téléphone	0.7%	4.2%	15.67%	60.79%	90.15%	90.26%
Lunettes	0.05%	0.66%	5.61%	66.88%	88.95%	89.99%
Portefeuille	1.2%	5.75%	23.18%	68.72%	95.30%	93.24%
Clés	0.2%	0.95%	15.20%	63.34%	79.83%	80.86%
Télécommande	0.7%	2.05%	20.02%	69.44%	92.90%	93.04%
MAP total (0.5)	**0.57%**	**2.72%**	**15.94%**	**65.84%**	**89.43%**	**89.48%**

Dans le tableau IV.5, une progression notable est observée entre 500 et 1000 itérations, jusqu'à atteindre environ 5000, où l'augmentation devient presque négligeable. Cela est dû à la

limitation des images dans notre base de données et au fait que notre algorithme a déjà appris tout ce qu'il avait besoin d'apprendre, se perfectionnant uniquement à partir de ce stade.

Tableau IV.6 : Vrais et faux positifs de chaque classe - Conception 1

Indicateurs de performances	Images par classe	map@0.5	Vrais positifs	Faux positifs
Téléphone	1099	90.26%	436	51
Lunettes	1083	89.99%	427	50
Portefeuille	1084	93.24%	320	42
Clés	1039	80.86%	296	41
Télécommande	1018	93.04%	449	39

Dans le tableau IV.6, on constate de meilleurs résultats pour les classes Télécommande, Téléphone et ce grâce à leurs formes assez facile à mémoriser pour un algorithme d'apprentissage profond. On trouve la plus basse valeur pour les clés à cause de leurs formes complexes et qui varie d'image en image. Car, ils constituent une boite englobante en eux même de la même classe à savoir une clé, Donc notre algorithme aura du mal à détecter une seule clé là où il aura plus de facilité à détecter un groupe de clés ou bien une clé avec un porte-clés.

Tableau IV.7 : Indicateurs de performances de la base de données YAHINE - Conception 1

Indicateurs de performances	map@0.5	Précision	Rappel	F_Score	Average IoU
Valeurs	89.48%	90%	87%	88%	79.64%

Le tableau IV.7 représente les indicateurs de performances calculés après 10000 itérations, où on peut voir que les résultats sont assez fiables avec des valeurs supérieures à 80% que l'on peut qualifier de bons résultats. Pour améliorer ses résultats, il nous faudra augmenter le nombre d'images jusqu'à environs 2000 par classe afin de pouvoir avoir une précision fiable qui pourra être utilisé pour des applications d'industries ou autres.

Nous allons maintenant simuler l'algorithme de notre modèle sur des images prises, présentées dans la figure IV.6, qui illustrent les résultats obtenus de manière visuelle, en complément des données chiffrées des tableaux précédents.

Figure IV.6 : Résultats de simulation avec YAHINE réelle – Conception 1

A partir de figure IV.6 et de l'image 1, on constate les limites de notre modèle qui permet de détecter toutes les classes. Dès que deux objets se superposent, il aura du mal à détecter les 2 objets distinctement.

En se référant à l'image 2 de la figure IV.6, il est observable que notre modèle n'a pas accès à toute la base de données des portefeuilles, d'où il a du mal à distinguer le portefeuille en haut de l'image compte tenu de la diversité des modèles présents dans cette catégorie. Quant aux clés comme cité précédemment dans l'analyse des tableaux, ils forment une boîte englobante en eux-mêmes. Par conséquent, notre algorithme éprouve des difficultés à détecter une seule clé isolée, tandis qu'il peut trouver plus facilement un groupe de clés ou une clé associée à un porte-clés, comme montré dans la l'image 3 de la même figure

Pour conclure, l'image 4 de la figure IV.6 illustre l'évaluation de notre modèle d'objets dans un environnement moyennement éclairé. Dans cette situation, l'appréciation de ce dernier demeure adéquate et fiable dans une mesure satisfaisante.

IV.4.2. Deuxième Conception : à base de YAHINE augmentée

Dans cette deuxième conception, nous utilisons l'augmentation de données pour élargir notre base de données, en ajoutant des images filtrées et modifiées. Ce qui nous donne les résultats notés dans les tableaux IV.8-9-10.

Tableau IV.8 : Comparatif des statistiques d'apprentissage de chaque classe - Conception 2

Itérations	100	300	500	1000	5000	10000
Téléphone	0.4%	1.92%	16.71%	61.7%	91.72%	92.28%
Lunettes	0.13%	0.87%	4.99%	71.59%	89.41%	90.21%
Portefeuille	1.05%	4.70%	27.35%	84.96%	93.41%	94.2%
Clés	0.23%	1.08%	15.21%	73.54%	80.55%	80.74%
Télécommande	0.37%	1.29%	14.92%	73,20%	92.77%	94.5%
MAP total (0.5)	**0.44%**	**1.97%**	**15.84%**	**73%**	**89.57%**	**90.38%**

Tableau IV.9 : Vrais et faux positifs de chaque classe - Conception 2

Indicateurs de performances	map@0.5	Vrais positifs	Faux positifs
Téléphone	92.28%	423	53
Lunettes	90.21%	418	48
Portefeuille	94.2%	322	35
Clés	80.74%	293	36
Télécommande	94.5%	450	35

Tableau IV.10 : Indicateurs de performances de la base de données YAHINE - Conception 2

Indicateurs de performances	map@0.5	Précision	Rappel	F_Score	Average IoU
Valeurs	90.38%	0.9	0.88	0.89	78.78%

L'analyse des tableaux IV.8-9-10 révèle une tendance intéressante. Les résultats obtenus avec la deuxième conception sont légèrement meilleurs pour l'ensemble des classes étudiées, et donc on a une amélioration globale du map.

La figure IV.7 illustre les résultats obtenus avec les données précédentes lorsque l'on a pratiqué des tests sur des images.

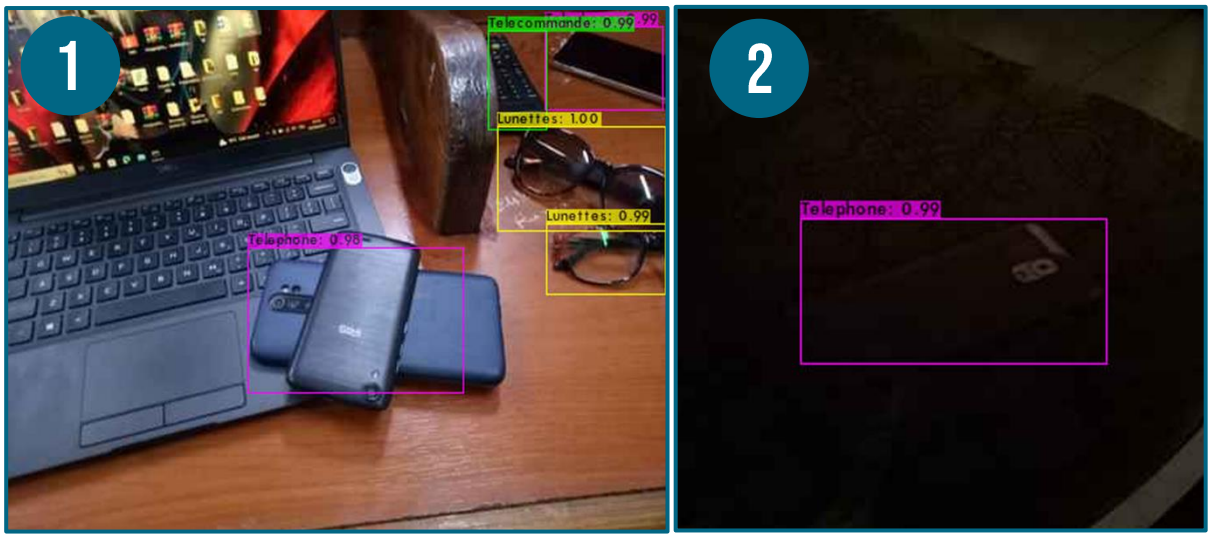


Figure IV.7 : Résultats de simulation avec YAHINE augmentée – Conception 2

On constate à partir de première de la figure IV.7 qu'avec ou sans augmentation de données l'algorithme a toujours du mal à détecter une pile d'objets, de même classe ou de différentes classes.

On remarque aussi à partir de l'image 2 de la figure IV.7 que grâce à l'augmentation de données, on discerne une amélioration des performances en termes de détection d'objets dans des environnements faiblement éclairés. Cette amélioration est due à l'application de divers filtres pour extraire les informations, ce qui permet une meilleure évaluation dans les zones d'ombre. Nous allons aborder cet aspect en profondeur.

IV.4.3. Etudes comparatifs approfondie

Afin de réaliser une comparaison complète entre nos deux conceptions, il est essentiel de mener une étude approfondie. Dans cette optique, il serait bénéfique d'approfondir notre analyse en utilisant des résultats graphiques, ce qui permettrait une meilleure visualisation des données. En examinant attentivement le graphe de la figure IV.8, nous pouvons observer plusieurs tendances intéressantes.

Tout d'abord, il est clair que l'augmentation de données présente initialement des difficultés, avec des performances faibles au départ. Cependant, après environ 650 itérations, nous remarquons une évolution drastique de l'apprentissage. Les performances s'améliorent de manière significative, indiquant que l'augmentation de données commence à jouer un rôle positif dans le

processus d'apprentissage. Cette amélioration continue tout au long de l'entraînement et reste supérieure par rapport à la conception sans augmentation de données avec une différence finale de 0.9%. Ce qui suggère que l'ajout d'augmentation de données a un impact positif sur les performances.

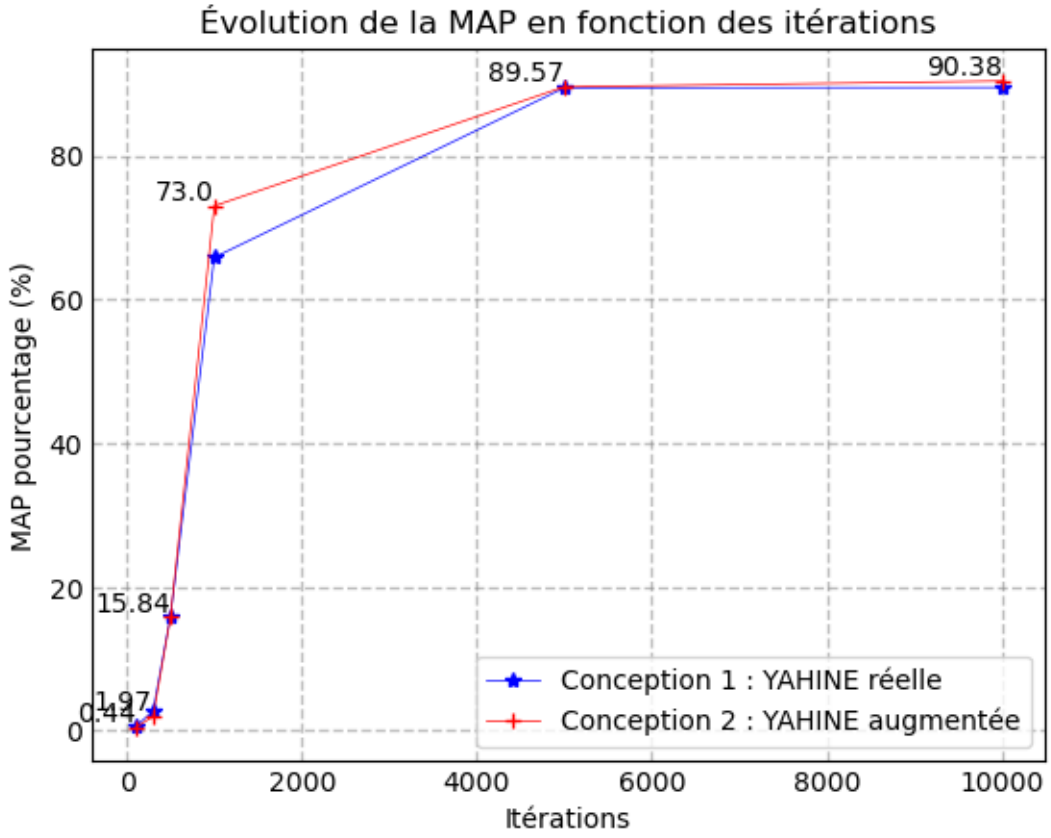


Figure IV.8 : Courbe comparatif entre l'apprentissage YAHINE Réelle et YAHINE augmentée

Se reposer exclusivement sur les résultats théoriques des tableaux et graphes pour évaluer la validité de notre modèle est insuffisant. Par conséquent, nous avons réalisé des tests pratiques approfondis afin d'extraire les caractéristiques essentielles et d'obtenir une comparaison exhaustive et approfondie. Cette approche nous permet de déterminer si notre modèle peut être utilisé dans des conditions spécifiques. Dans cette optique, nous avons soumis notre modèle à des tests dans des conditions de faible visibilité en raison de l'éclairage, variation de distance.

IV.4.3.1. Test en conditions de faible visibilité

IV.4.3.1.1. Luminosité

Pour ce test on fait varier la luminosité de notre image afin de simuler des endroits à faible concentration de lumière pour ensuite comparer entre les deux approches proposées, avec et sans augmentation de données, on note les résultats dans la figure IV.9.

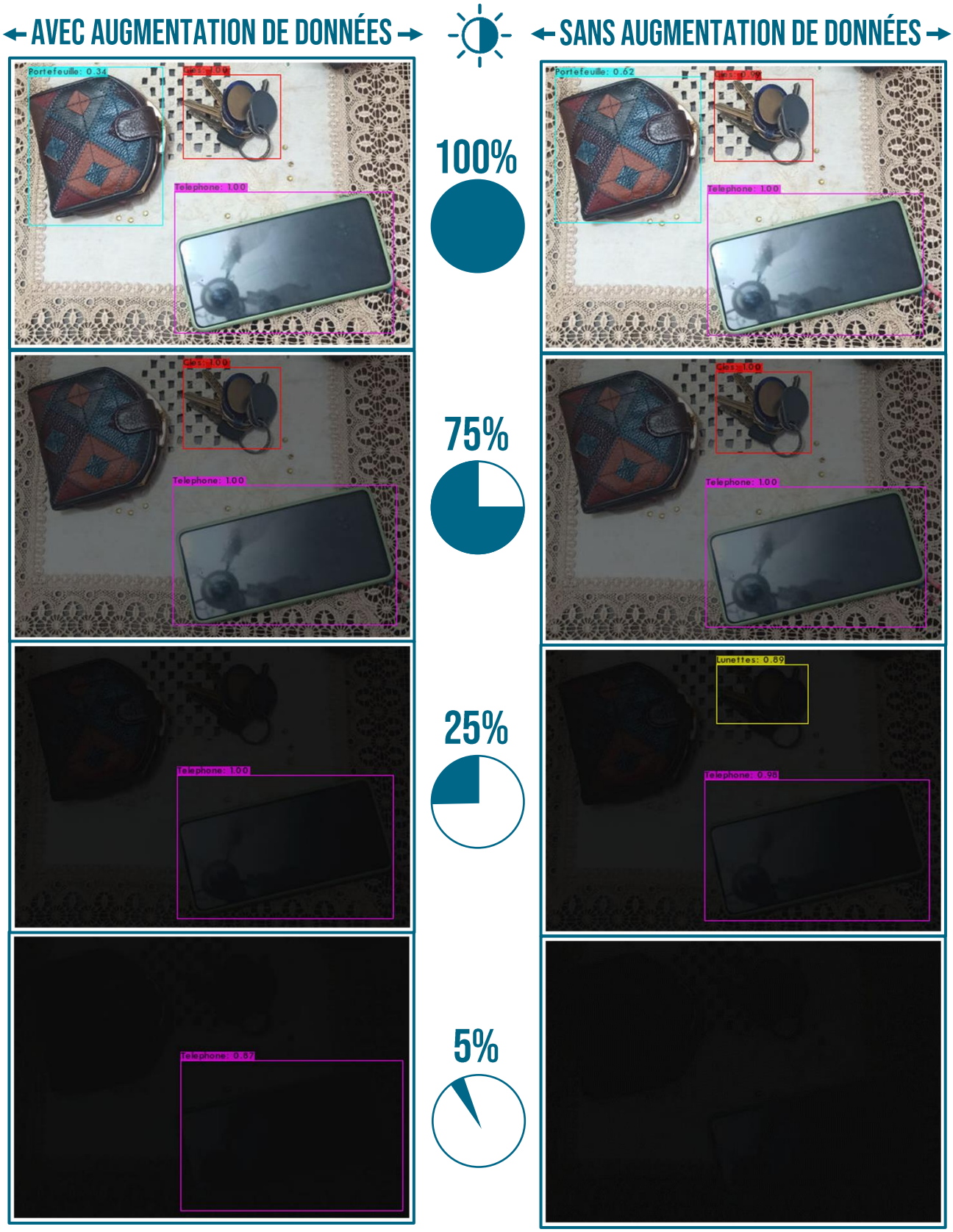


Figure IV.9 : Comparaison des performances de détection d'objets avec et sans augmentation de données en fonction de l'intensité lumineuse

On remarque à partir de la figure IV.9 que l'approche utilisée avec l'augmentation de données offre de meilleurs résultats à faible intensité lumineuse. Les résultats qu'elle offre reste fiable, peu importe la luminosité, contrairement à la conception 1 qui nous a donné un faux résultat avec une luminosité de 25%, en confondant les classes « lunettes » et « clés », là où avec l'augmentation de données le programme n'a rien détecté et reste alors plus fiable, ainsi même pour une luminosité inférieure à 5% il arrive encore à détecter la classe « téléphone ».

On va faire le même test pour chaque classe afin de déterminer à quel pourcentage de luminosité on arrive approximativement à détecter chaque objet, on notera les résultats dans les deux tableaux IV.11-12.

Tableau IV.11 : Pourcentage de luminosité pour la détection de chaque classe - Conception 1

Pourcentage / Classe	100%	75%	50%	25%	5%
Téléphone	Détecté	Détecté	Détecté	Détecté	
Lunettes	Détectée	Détectée			
Portefeuille	Détecté				
Clés	Détectées	Détectées			
Télécommande	Détectée	Détectée	Détectée		

Tableau IV.12 : Pourcentage de luminosité pour la détection de chaque classe - Conception 2

Pourcentage / Classe	100%	75%	50%	25%	5%
Téléphone	Détecté	Détecté	Détecté	Détecté	Détecté
Lunettes	Détectée	Détectée	Détectée		
Portefeuille	Détecté				
Clés	Détectée	Détectée	Détectée		
Télécommande	Détectée	Détectée	Détectée	Détectée	

On peut en déduire, en prenant en compte la remarque précédente, ainsi que des tableaux IV.11-12 qu'en l'absence de lumière, l'augmentation de données offre une meilleure appréciation en termes de résultats. Ces observations renforcent l'idée que l'augmentation de données peut jouer un rôle significatif dans l'amélioration des performances dans des conditions de faible luminosité.

Ainsi, il devient plus facile de tirer des conclusions sur l'efficacité de cette méthode dans de telles situations.

IV.4.3.1.2. Distance

Pour ce test, nous avons varié la distance de notre image afin de simuler des scénarios où les objets seront éloignés de l'objectif de la caméra. Les résultats sont enregistrés dans les tableaux IV.13-14, où nous pouvons observer les performances de chaque conception en fonction de la distance.

Tableau IV.13 : Performances de détection d'objets en fonction de la distance - Conception 2

Distance en mètre / Classe	0.5	1	1.5	2	2.5
Téléphone	Détecté	Détecté	Détecté	Détecté	
Lunettes	Détectées	Détectées	Détectées		
Portefeuille	Détecté	Détecté	Détecté	Détecté	
Clés	Détectées				
Télécommande	Détectée	Détectée	Détectée	Détectée	

Tableau IV.14 : Performances de détection d'objets en fonction de la distance - Conception 1

Distance en mètre / Classe	0.5	1	1.5	2	2.5
Téléphone	Détecté	Détecté	Détecté	Détecté	
Lunettes	Détectées	Détectées	Détectées	Détectées	
Portefeuille	Détecté	Détecté	Détecté		
Clés	Détectées				
Télécommande	Détectée	Détectée	Détectée	Détectée	

En se basant sur les deux tableaux IV.13-14, on observe qu'il n'y a pas de changement significatif en ce qui concerne l'utilisation de l'augmentation de données avec ses filtres. Les résultats varient de manière fluctuante, parfois meilleurs avec l'augmentation de données et parfois meilleurs sans. Par conséquent, il n'est pas possible de tirer des conclusions sur l'efficacité de l'augmentation de données pour améliorer l'appréciation sur une distance plus longue.

IV.4.3.2. Temps de détection

Dans cette étude, nous avons réalisé un test visant à évaluer la durée d'exécution de notre modèle. L'objectif principal était de mesurer le temps nécessaire pour effectuer des prédictions sur différentes images de test. Pour cela, nous avons examiné trois scénarios distincts. Dans le premier scénario, nous avons utilisé des images ne contenant qu'un seul objet appartenant à une classe spécifique. Dans le deuxième scénario, nous avons utilisé des images comportant plusieurs objets d'une même classe. Enfin, dans le troisième scénario, nous avons utilisé des images contenant plusieurs objets appartenant à différentes classes. Par la suite, nous avons calculé la moyenne de chaque scénario et les avons consignées dans le tableau IV.15.

Tableau IV.15 : Temps de prédiction des boites de délimitations avec et sans augmentation de données

Scénario / Conception	YAHINE réelle	YAHINE augmentée
Premier scénario	890.96 ms	764.221 ms
Deuxième scénario	891.688 ms	691.297 ms
Troisième scénario	868.239 ms	883.804 ms

À partir des observations du tableau IV.15, il est évident que l'augmentation des données entraîne une réduction significative du temps de calcul pour les scénarios où nous utilisons une seule classe. Cependant, dès que nous introduisons plusieurs classes, l'algorithme nécessite plus de temps pour effectuer les prédictions des boîtes de délimitation.

On constate aussi que dans les différentes conceptions nous avons une vitesse moyenne de détection de moins d'une seconde par image ce qui nous donne une bonne appréciation pour certaines applications. Mais elle reste insuffisante pour certaines situations telles que dans Navigation autonome de drones, la robotique industrielle avancée où des temps de réaction très courts sont nécessaires pour garantir la sécurité des opérations et la coordination des robots.

IV.4.4. Points forts et points faibles

En analysant attentivement les résultats des tests précédents, nous avons acquis une compréhension plus approfondie des points forts et des points faibles de notre modèle. Ces observations nous permettent désormais d'identifier et de mettre en évidence directement les aspects les plus significatifs de notre approche.

IV.4.4.1. Points fort

- ✓ Le modèle offre une excellente appréciation de la luminosité, en particulier dans des environnements faiblement éclairés. L'utilisation de l'augmentation de données et de divers

filtres permet une amélioration significative des performances de détection d'objets dans de telles conditions.

- ✓ Le modèle démontre une grande polyvalence en termes de détection d'images sur divers formats et avec différentes qualités.
- ✓ Les résultats obtenus avec l'approche utilisant l'augmentation de données sont remarquablement fiables, même lorsque la luminosité est très faible. Le modèle parvient à détecter la classe "téléphone" même lorsque la luminosité est inférieure à 5%, ce qui témoigne de sa robustesse et de son efficacité.
- ✓ Très bonne précision des boites de délimitations prédites en temps réel.
- ✓ Le modèle démontre parfois une bonne précision dans la génération des boîtes englobantes pour la classe « *clés* ».
- ✓ Vitesse de prédiction de moins d'une seconde cela permettra une réponse quasi instantanée à certaines requêtes.

IV.4.4.2. Points faibles

- × Difficulté à détecter efficacement une pile d'objets, qu'ils soient de la même classe ou de différentes classes à l'exception de la classe « *clés* ».
- × Fluctuation des résultats entre l'utilisation de l'augmentation de données et son absence sur l'amélioration des performances sur une distance plus longue, là où normalement nous aurons un résultat plus avantageux pour l'augmentation de données.
- × Le modèle rencontre des défis lors de la détection simultanée de plusieurs objets appartenant à des classes différentes, ce qui peut entraîner des confusions entre les classes en raison de la charge intensive de calcul à laquelle l'algorithme est soumis.
- × Limitation d'utilisation GPU de Google Colab liée à certaines pertes de données et un temps d'attente colossal.
- × Redimensionnement des images pour avoir des images peu encombrantes afin de répondre à la contrainte de stockage du compte drive « 12 GO » et d'éviter le dépassement de ce seuil, mais, en contrepartie, nous aurons une perte de données.
- × Vitesse moyenne de prédiction proche d'une seconde par image ce qui est insuffisant pour des situations en temps réel nécessitant un temps de réaction très court.

IV.5 Conclusion

Au cours de ce chapitre, nous avons présenté en détail notre étude portant sur la détection d'objets d'intérieur en utilisant notre propre base de données. Les résultats obtenus ont été très prometteurs, démontrant l'efficacité et la précision de notre modèle. Cependant, nous avons également identifié certaines limites et points faibles de notre système, qui soulignent les défis et les opportunités des améliorations futures et des nouvelles investigations dans ce domaine.

CONCLUSION GÉNÉRALE

Au cours de ce projet de fin d'études, nous avons mené une exploration approfondie du domaine de l'intelligence artificielle et de la détection d'objets, en mettant l'accent sur l'application de l'apprentissage profond en vision artificielle, notamment dans des environnements intérieurs.

Une étape clé de notre travail a été la création de notre propre base de données, que nous avons nommée YAHINE, regroupant plus de 5000 images annotées des objets les plus fréquemment égarés. Cette base de données a servi pour évaluer notre modèle de détection d'objets d'intérieur.

À partir de cette base de données, nous avons entraîné un algorithme YOLOv4 en réalisant deux expériences distinctes : l'une avec la base de données YAHINE réelle et l'autre augmentation de données. Une évaluation des performances de détection a été faite à l'aide de certains indicateurs.

Les résultats obtenus lors de l'évaluation et des tests ont mis en évidence plusieurs points forts de notre modèle. Cependant, il reste loin d'être parfait. Ce qui ouvre de nouvelles perspectives et possibilités d'améliorations pour l'avenir, ainsi que ses horizons d'applications. Dans notre, nous avons utilisé deux smartphones et une webcam. Une implémentation sur des robots mobiles ou en utilisant les caméras de surveillance sera plus intéressante. Il présente aussi un potentiel intéressant pour localiser et identifier efficacement des objets égarés dans différents contextes, tels que les lieux publics, les transports en commun, les centres commerciaux.

Enfin, ce projet nous a permis de découvrir un autre horizon et une immersion dans le domaine de l'intelligence artificielle et de la reconnaissance automatique en général. Ce qui constitue un très bon complément pour notre formation d'automaticiens.

RÉFÉRENCES

[1] Hasan, M. M., Islam, M. U., Sadeq, M. J., Fung, W. K., & Uddin, J, "Review on the Evaluation and Development of Artificial Intelligence for COVID-19 Containment", *Sensors*, vol. 23, no.1, pp. 527, 2023.

[2] TOMAR, Ravi, HINA, Manolo Dulva, ZITOUNI, Rafik, et al, "Innovative trends in computational intelligence", *Springer*, 2022.

[3] B. T JIJO, A.M ABDULAZEEZ, "Classification based on decision tree algorithm for machine learning", *Evaluation,* vol. 6, pp. 7, 2021.

[4] CHAI, Junyi, ZENG, Hao, LI, Anming, et al, "Deep learning in computer vision: A critical review of emerging techniques and application scenarios", *Machine Learning with Applications*, vol.5, 2021.

[5] Zhao, Zhong-Qiu, et al, "Object detection with deep learning*", Journal of latex class files*, vol. 14, no.8, pp. 1-22, 2017.

[6] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," *In 2010 IEEE Computer Society Conference on computer vision and pattern recognition*, pp. 2528-2535, 2010.

[7] Guo, Meng-Hao, et al, "Attention mechanisms in computer vision: A survey", *Computational Visual Media,* vol.8, no.3, pp. 331-368, 2022.

[8] L. Li, Z. Zhu et Z. Liu, "Improved Algorithm for Road Multi-target Tracking Based on YOLO", *IEEE 6th Advanced Information Technology, Electronic and Automation Control Conference* (IAEAC), pp. 1603-1609, 2022.

[9] P. Goel and S. Agarwal, "Hybrid Approach of Haar Cascade Classifiers and Geometrical Properties of Facial Features Applied to Illumination Invariant Gender Classification System," *2012 International Conference on Computing Sciences, Phagwara, India, 2012*, pp. 132-136, doi: 10.1109/ICCS.2012.40.

[10] M. Hofmann and G. Rigoll, "Exploiting gradient histograms for gait-based person identification," *2013 IEEE International Conference on Image Processing, Melbourne, VIC, Australia, 2013*, pp. 4171-4175, doi: 10.1109/ICIP.2013.6738859.

[11] M. M. M. Al-Khalidy, A. Mohammed, R. Nasser and Z. Zuhair, "Impact of performance: SSD vs YOLOV3 machine learning for mask detection and temperature sensing device," *4th Smart Cities Symposium (SCS 2021)*, pp. 361-367, 2021.

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation*", In Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580-587, 2014.

[13] R. Girshick, "Fast r-cnn", In *Proceedings of the IEEE international conference on computer vision*, pp. 1440-1448, 2015.

[14] S. Ren, K. He, K. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks", *Advances in neural information processing systems*, vol. 28,2015

[15] T.Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection", In *Proceedings of the IEEE international conference on computer vision*, pp. 2980-2988, 2017.

[16] W. Lie, D. Anguelov, D, Erhan, C. Szegedy, S. Reed, C.Y. Fu, and A.C. Berg, "SSD: Single Shot Multibox Detector", In *Computer Vision–ECCV 2016: 14th European CoProceedings, Part I 14*, pp. 21-37, 2016.

[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, realtime object detection", 2015.

[18] C. Liu, Y. Tao, J. Liang, K. Li and Y. Chen, "Object Detection Based on YOLO Network," *2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC)*, pp. 799-803, 2018.

[19] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), pages 659, 2019.

[20] Redmon, J., & Farhadi, A, "YOLO9000: better, faster, stronger," *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263-7271, 2016.

[21] J. Redmon, A. Ferhadi, "Yolov3: An incremental Improvement", *arXiv preprint arXiv, pp. 1804-02767*,2018.

[22] A. Bochkovskiy, C.Y. Wang, and H.Y.M. Liao, "Yolov4: Optimal speed and accuracy of object detection", *arXiv preprint arXiv*, pp. 2004-10934, 2020.

[23] X. Wang, Z. Chen, B. Wei and M. Ling, "Application of Pruning Yolo-V4 with Center Loss in Mask Wearing Recognition for Gymnasiums and Sports Grounds of Colleges and Universities," *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*, pp. 1373-1377, 2020.

[24] T.Y. Lin, M. Maire, S. Belongir, J. Hays, P. Perona, D. Ramanan, and C.L. Zitnick, "Microsoft coco: Common objects in context", In *Computer Vision–ECCV 2014: 13th European Conference, pp. 740-755, 2014.*

[25] M. Everingham, S.A. Eslami, L. Van Gool, C.K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes challenge: A retrospective”, *International journal of computer vision*,pp. 98-136, 2015.

[26] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database”, In IEEE *conference on computer vision and pattern recognition*, pp. 248-255, *2009.*

[27] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving”, the kitti vision benchmark suite”, In IEEE *conference on computer vision and pattern recognition*, pp. 3354-3361, 2012.

[28] B. Zhou, H. Zhou, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, “Scene parsing through ade20k dataset”, In *Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 633-641, 2017.

[29] S. Yang, P. Luo, C.C. Loy, and X. Tang, “Wider face: A face detection benchmark”, In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5525-5533, 2016.

[30] https://github.com/tzutalin/labelImg

[31] T. Carneiro, R. V. Medeiros Da NóBrega, T. Nepomuceno, G. -B. Bian, V. H. C. De Albuquerque and P. P. R. Filho, "Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications," *in IEEE Access*, vol. 6, pp. 61677-61685, 2018, doi: 10.1109/ACCESS.2018.2874767.

[32] S. Singh, A. Paul and M. Arun, "Parallelization of digit recognition system using Deep Convolutional Neural Network on CUDA," *2017 Third International Conference on Sensing, Signal Processing and Security (ICSSS)*, Chennai, India, 2017, pp. 379-383, doi: 10.1109/SSPS.2017.8071623.

[33] M. Ponika, K. Jahnavi, P. S. V. S. Sridhar and K. Veena, "Developing a YOLO based Object Detection Application using OpenCV," *2023 7th International Conference on Computing Methodologies and Communication (ICCMC)*, Erode, India, 2023, pp. 662-668, doi: 10.1109/ICCMC56507.2023.1008407.

Résumé

Dans ce travail, un système de détection en temps réel a été développé pour identifier et classer avec précision les objets d'intérieur. Pour cela, une base de données personnalisée, appelée YAHINE, a été créé comprenant une variété d'images représentant quelques objets d'intérieur souvent égarés. Ensuite, l'algorithme YOLOv4, basé sur le transfert d'apprentissage, a été entraîné en utilisant deux expériences distinctes : la première en utilisant la base de données YAHINE, et l'autre en utilisant l'augmentation de données. Les performances de détection ont été évaluées en utilisant des mesures telles que la précision, le rappel et le score F1 pour comparer les résultats des deux approches. Les résultats obtenus démontrent l'efficacité du système de détection d'objets proposé.

Abstract

In this work, a real time detection system was developed that accurately identifies and classifies indoor objects. Hence, a custom dataset, called YAHINE; was created, which includes a variety of images representing different indoors usually lost. Then, YOLOv4 algorithm, based on transfer learning; was trained using two distinct experiments: The first using the real YAHINE dataset and the other involving data augmentation. The detection performance was evaluated using metrics such as precision, recall, and F1 score to compare the results of the two approaches. The obtained results show the effectiveness of the proposed system.

ملخص

تم في هذا العمل تطوير نظام كشف فوري يتعرف ويصنف بدقة الأشياء الداخلية .ولذلك تم إنشاء مجموعة بيانات مخصصة تسمى ياهين, تحتوي على مجموعة متنوعة من الصور تمثل الأشياء الداخلية المفقودة عادة تم تدريب خوارزمية يولوالإصدار 4 باستخدام تقنية التعلم النقلي، من خلال تنفيذ تجربتين متميزتين :التجربة الأولى باستخدام مجموعة بيانات ياهين الفعلية والأخرى تشمل زيادة البيانات .تم تقييم أداء الكشف باستخدام مقاييس مثل الدقة والاسترجاع لمقارنة نتائج النهجين .أظهرت النتائج المحصل عليها فعالية النظام المقترح