MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE ABBES LAGHROUR DE KHENCHELA
FACULTE DES SCIENCES ET DE LA TECHNOLOGIE

Département Mathématique et Informatique

N° de série : ……

Mémoire de fin d'études

Pour l'obtention du diplôme de Master (L.M.D)

Spécialité : Sécurité et Technologie Web

Apprentissage supervisé pour la détection et suivi d'objets sur des vidéos capturées à partir de drones

Réalisé par :
Mr. BENYEZZA Oussama Wail
Mlle. LAHRECHE Lina

Encadré par :
Dr. ABBAS Fayçal

Membres de jury :

Dr. Mahdi Mohamed MALIK : Président

Mme. Souad SAADI : Examinatrice

Promotion : 2021/2022

Résumé

Les solutions de détection d'objets basées sur les algorithmes d'apprentissage profond ont attiré toute l'attention ces dernières années. L'utilisation des drones pour des applications potentielles telles que la surveillance des forêts et territoires, la gestion de l'agriculture et l'administration urbaine. La détection et suivi d'objets en mouvement est une tâche complexe pour les techniques existantes, en effet cette complexité réside dans la diversité des objets (forme, texture, couleur), variations de luminosité, le problème d'occlusions.

Dans ce mémoire notre objectif est de lever les défis imposés par les techniques traditionnelles en se basant sur les approches de l'apprentissage en profondeur afin d'offrir une solution automatique en temps réel pour la détection et le suivi d'objets en mouvement sur des vidéos, capturées à partir de drones.

Dans ce travail nous proposons un modèle basée sur les réseaux de neurones convolutifs et le transfert d'apprentissage afin de détecter les arbres de palmiers dans des zones forestière denses. Notre idée consiste à utiliser un transfert de connaissances en se basant sur la dernière architecture de YOLO v5.

En effet nous avons modifié l'architecture du Backbone afin d'optimiser l'étape de détection et de reconnaissance des palmiers afin d'optimiser la tâche de l'extraction de caractéristiques et également de permettre à notre modèle de se focaliser sur les caractéristiques les plus pertinentes. Un jeu de donnée a été introduit manuellement afin d'évaluer les performances de notre solution. Notre modèle produit de bons résultats en termes de précision ainsi il permet d'offrir un comptage des palmiers en temps réel.

Mots clés : Apprentissage profond, Réseaux de neurones convolutifs, Détection d'objets, YOLOv5, Backbone, transfert de connaissance

---

ملخص

اجتذبت حلول اكتشاف الكائنات القائمة على خوارزميات التعلم العميق كل الاهتمام في السنوات الأخيرة. استخدام الطائرات بدون طيار للتطبيقات مثل مراقبة الغابات والأراضي وإدارة الزراعة والإدارة الحضرية. يعد اكتشاف وتتبع الأجسام المتحركة مهمة معقدة بالنسبة للتقنيات الحالية ، بل إن هذا التعقيد يكمن في تنوع الكائنات (الشكل ، والملمس ، واللون) ، والاختلافات في السطوع ، ومشكلة الانسداد.

في هذه المذكرة، هدفنا هو التغلب على التحديات التي تفرضها التقنيات التقليدية القائمة على مناهج التعلم العميق من أجل تقديم حل تلقائي في الوقت الفعلي لاكتشاف وتتبع الأجسام المتحركة على مقاطع الفيديو التي تم التقاطها من الطائرات بدون طيار.

في هذا العمل نقترح نموذجًا يعتمد على الشبكات العصبية التلافيفية وننقل التعلم لاكتشاف أشجار النخيل في مناطق الغابات الكثيفة. فكرتنا هي استخدام نقل المعرفة بناءً على أحدث نموذج YOLO v5

في الواقع ، لقد قمنا بتعديل بنية النموذج من أجل تحسين مرحلة اكتشاف أشجار النخيل والتعرف عليها من أجل تحسين مهمة استخراج الميزات وأيضًا للسماح لنموذجنا بالتركيز على أهم الميزات الأكثر صلة. تم تقديم مجموعة بيانات يدويًا لتقييم أداء الحل الذي نقدمه. ينتج نموذجنا نتائج جيدة من حيث الدقة ويسمح لنا بتوفير عدد من أشجار النخيل في الوقت الفعلي.

Abstract

Object detection solutions based on deep learning algorithms have attracted much attention in recent years. The use of drones for potential applications such as forest and land surveillance, agriculture management and urban administration. Detecting and tracking moving objects is a complex task for existing techniques, indeed this complexity lies in the diversity of objects (shape, texture, color), variations in brightness, the problem of occlusions.

In this memory our objective is to overcome the challenges imposed by traditional techniques by using deep learning approaches to provide an automatic real-time solution for the detection and tracking of moving objects on videos, captured from drones.

In this work we propose a model based on convolutional neural networks and transfer learning to detect palm trees in dense forest areas. Our idea is to use knowledge transfer based on the latest architecture of YOLO v5.

Actually, we modified the architecture of the Backbone in order to optimize the detection and recognition step of the palm trees in order to optimize the task of feature extraction and also to allow our model to focus on the most relevant features. A dataset has been introduced manually in order to evaluate the performance of our solution. Our model produces good results in terms of accuracy so it can offer a real time palm tree count.

Keywords : Deep learning , Object detection, Convolutional neural network, YOLOv5, knowledge transfer , Backbone

---

Dédicace

“

Je dédie ce modeste travail À :

*À ma très chère mère **”Farida”**, Quoi que je fasse ou que je dise, je ne saurai point te remercier comme il se doit. Ton affection me couvre, ta bienveillance me guide et ta présence à mes côtés a toujours été ma source de force pour affronter les différents obstacles*

*À mon très cher père **”Saci”**, Tu as toujours été à mes côtés pour me soutenir et m'encourager. Que ce travail traduit ma gratitude et mon affection.*

*À mes très chers frères **”Acim”** et **”Ayhem”** et mes chers sœurs **”Rana”** et **”Rama”** et **”Malek”***

*À mon bras droit **”Mehdi”**, Merci d'être l'épaule sur laquelle je peux toujours compter*

À tout mes amis, Merci pour votre support et soutien moral

À tous ceux qui me sont chers, à vous tous

Merci,

Puisse Dieu vous donne santé, bonheur, courage et surtout réussite.

”

- Oussama

Dédicace

“

D'un sentiment plein d'amour, de sincérité de fidélité, Je dédie ce modeste travail à :

À l'homme de ma vie, mon exemple éternel, mon soutien moral et source de joie et de bonheur, celui qui s'est toujours sacrifié pour me voir réussir, que dieu te protège de tout malheur, à toi papa ***"Hocine"*** *.*

À la lumière de mes jours, la source de mes efforts, la flamme de mon cœur, ma vie et mon bonheur , maman que j'adore ***"Nassima"*** *je ne saurai jamais comment te repayer pour tout ce que tu as fait pour moi.*

À mes frères ***"Ahmed"*** *et* ***"Hadil"****, et ma petite princesse (alaa).*

À mon binôme ***"Oussama"*** *merci pour ton aide et ton soutien dans cette longue aventure.*

À mes plus proches amies : ***"Aridj"*** *et* ***"Amina"****.*

À tous mes enseignants sans exception.

”

- Lina

Remerciements

“

En premier, nous aimerions remercier le bon Dieu le tout puissant de nous avoir donné le courage et la volonté de réaliser ce mémoire.

Nous tenons tout d'abord à manifester notre profonde gratitude envers notre encadreur de ce mémoire Dr. Fayçal ABBAS pour nous avoir fait profiter de son expérience dans la recherche et de la qualité de son encadrement pendant toute la durée du mémoire, sa disponibilité et ses conseils ont été très constructifs.

Nous désirons remercier nos chers parents qui nous ont soutenus et encouragé durant toute notre vie et pendant notre cursus d'étude.

Nos remerciements les plus chaleureux vont à tous nos amis pour leurs disponibilités et leurs très précieux conseils ainsi que leurs remarques qui nous ont permis d'améliorer la qualité de ce travail.

Nous tenons à exprimer toute notre grande gratitude aux membres de jury d'avoir accepté de juger ce travail.

”

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Les drones nous surprennent sous toutes leurs formes, ils sont de plus en plus nombreux, et commencent à occuper notre quotidien. Nous devenons donc progressivement familiers de ces machines qui peuvent être, selon leur usage, des appareils professionnels civils ou militaires, des objets connectés, des jouets, etc.

Ce projet de fin d'étude vise à apporter une solution de détection et de reconnaissance en temps réel sur des séquences vidéo prise par des drones. Nous nous concentrons sur la détection d'arbres (palmiers) à partir d'images aériennes. Les résultats de la détection peuvent aider dans diverses applications telles que l'agriculture, l'urbanisme et la modélisation 3D.

Plusieurs taches effectuer manuellement par des humains dans des zones forestiers ne sont pas envisageables et cela pour des raisons de cout et de sécurité . C'est dans ce contexte que nous créons un modèle de deep learning permettant de détecter et de suivre un objet dans une séquence vidéo prise par un drone.

Cette tâche est encore plus complexe lorsqu'il s'agit de détecter une espèce spécifique (dans notre cas palmier) dans une forêt dense. En effet, les arbres étant très similaires, il est difficile d'extraire des caractéristiques.

Dans ce mémoire notre but est de proposer une solution deep learning qui permet d'effectuer une détection et reconnaissance des objets à partir d'une image 2D ou sur une séquence vidéo prise par des drones. Dans un premier temps, nous créons une base de données afin d'entrainer et d'évaluer notre modèle. Notre solution basée sur les réseaux de neurones convolutifs et le transfert d'apprentissage afin de détecter les arbres de palmiers dans des zones forestière denses. Notre idée consiste à utiliser un transfert de connaissances en se basant sur la dernière architecture de YOLO v5.

Organisation du mémoire

Ce mémoire est organisé en cinq chapitres :

Le premier chapitre "**L'intelligence Artificielle**" Dans le premier chapitre, nous présenterons le principe de l'apprentissage en IA, pour cela nous avons défini la base associées à l'apprentissage, ainsi que les caractéristiques d'apprentissage et aussi les différentes classifications d'apprentissage automatique, pour les approches d'apprentissage en IA, il existe plusieurs types, nous montrons quelques approches, qui sont les plus connues dans le domaine de l'IA, et les réseaux de neurones convolutionnels qui seront utilisés tout au long de nos travaux.

Le deuxième chapitre "**Détection des objets**" Dans le deuxième chapitre, nous présenterons les différentes techniques pour la détection d'objet en basant sur les versions de notre modèle choisi "**YOLO v5**".

Le troisième chapitre "**État de l'art**" Dans le troisième chapitre, nous passons brièvement en revue certains travaux remarquables qui utilisent des méthodes de détection des palmiers pour obtenir un aperçu générale.

Le quatrième chapitre "**Implémentation**" Dans le quatrième chapitre, nous allons montrer la partie implémentation de notre travail.

Le cinquième chapitre "**Résultats et Validation**" Dans le cinquième chapitre, nous discuterons les différents résultats expérimentaux obtenus par notre modèle, et procédé à une discussion de ces derniers.

Enfin, nous terminerons ce travail par une conclusion générale et nous proposerons quelques perspectives pour des futurs travaux.

Chapitre I

L'intelligence Artificielle

1 Introduction

Une certaine partie de notre monde est envahi par les technologies de l'intelligence artificielle, depuis plus d'une décennie, l'intelligence artificielle (IA) vit une accélération dans son développement et son adoption dans plusieurs domaines, elle se concentre sur la résolution de problèmes liés aux activités humaines de différentes natures (prise de décision, perception, diagnostic, interprétation de données, conception, planification, compréhension du langage ...), donc les objectifs de l'intelligence artificielle sont la création ou la simulation de processus cognitifs similaires à ceux de l'humain. [1]

L'intelligence artificielle (IA) est définie comme un domaine de la science et de l'ingénierie qui s'intéresse à l'analyse et à la résolution de problèmes informatiques, depuis ses débuts comme domaine de recherche spécifique l'intelligence artificielle (IA) se développe encore, ce développement a explosé les stratégies potentielles d'utilisation des données pour créer des prévisions scientifiques. [2].

L'intelligence artificielle (IA) permet aux machines de reconnaître des images, de transcrire des sons d'une langue à une autre, de traduire du texte, de conduire des voitures autonomes ou de contrôler des processus industriels. [3]

L'apprentissage automatique (également connu sous le nom de Machine Learning) est l'une des extensions de l'intelligence artificielle, ce champ d'études a pour but d'améliorer les performances des machines à résoudre des tâches mais en permettant aux machines de répondre à la question grâce à ce qu'elle en a appris auparavant. [4]

L'apprentissage en profondeur(également connu sous le nom de Deep Learning) est une technique d'apprentissage automatique qui a considérablement amélioré les résultats dans de nombreux domaines tels que la vision par ordinateur, la reconnaissance vocale et la traduction automatique, les techniques d'apprentissage en profondeur permettent de résoudre de nombreux problèmes dans de nombreux domaines tels que la santé, les transports, le commerce, la finance et l'énergie... [5]

Dans ce chapitre, nous allons introduire les principes de l'apprentissage en intelligence artificielle est nous allons mettre l'accent sur ce qui est essentiel pour notre travail, et nous allons explique les réseaux de neurones convolutif qui est très important dans notre mémoire.

2 Définition de l'intelligence artificielle

L'intelligence artificielle n'a pas de définition universelle pour le moment, on peut identifier plusieurs façons d'expliquer l'intelligence artificielle, nous allons citer quelques définitions :

- « La faculté de connaître et comprendre, incluant la perception, l'apprentissage, l'intuition, le jugement et la conception. » (Petit Robert)
- « La faculté de connaître et de raisonner. » (Dictionnaire Américain Héritage).
- « Application de la connaissance à la résolution de problèmes. » (Newell et Simon).

La définition a peu près répondue est que l'intelligence artificielle est une discipline de l'informatique qui réunit des sciences théorie et technologie (y compris la logique mathématique, les statistiques, les probabilités, la neurobiologie computationnelle et l'informatique) [6], pour reproduire les processus cognitifs humains dans le but de réaliser des actions « intelligente » .[7]

3 Une courte histoire de l'intelligence artificielle

L'intelligence artificielle a passé par de nombreux événements, voici quelques étapes importantes de son histoire :

- 1943 : McCulloch et Pitts : Modèle de circuit booléen du cerveau.
- 1950 : Machines de Turing : Premier ordinateur électronique a déchiffré les codes allemands permettant le débarquement en Normandie.
- 1950 : Les premiers programmes d'intelligence artificielle, programme de dames de Samuel, Newell et Simon's Logic Theorist, le moteur de géométrie de Gelernter .
- 1956 : Dartmouth Conference : "Every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it."
- 1965 : L'algorithme complet de Robinson pour le raisonnement logique "Robinson's complete algorithm for logical reasoning".
- 1966 → 1973: L'intelligence artificielle découvre la complexité computationnelle la recherche sur les réseaux neuronaux disparaît presque "Al discovers computational complexityy Neural network résearch almost disappears".
- 1969 → 1979 : Développement précoce des systèmes basés sur la connaissance "Early development of knowledge-based systems" .
- 1978 : L'intelligence artificielle devient une science *.
- 1980 : L'intelligence artificielle devient une industrie.
- 1980 → 1989 : Système expert. En 1985, le marché d'un milliard de dollars. Projet ambitieux (ordinateur japonais de cinquième génération).
- 1986 : Les réseaux neuronaux reviennent en force
- 1995 : "The emergence of intelligent agents- "-bots"".
- 2000 → 2010 : Explosion des applications (augmentation de la puissance des ordinateurs, disponibilité de quantités massives de données et progrès de l'apprentissage automatique).

4 Les Techniques de l'intelligence artificielle

Dans tous les domaines d'activité, les techniques de l'intelligence artificielle tendent à élargir le champ d'action des machines, leur donnant la capacité de voir, d'entendre, de raisonner, de parler, d'agir, Il existe plusieurs technique en va expliquer quelques-unes :

4.1 Système Expert

Un système expert est un système qui utilise des connaissances humaines capturées dans un ordinateur pour résoudre des problèmes qui sont suffisamment difficiles, les systèmes experts recherchent et utilisent les informations pertinentes de leurs utilisateurs et des bases de connaissances disponibles afin de faire des recommandations, l'utilisateur peut interagir avec un ordinateur pour résoudre un certain problème, cela peut se produire parce que le système expert peut stocker des connaissances heuristiques, généralement pour développer un système expert, une méthode basée sur des règles est nécessaire pour analyser et calculer la base de connaissances [8]

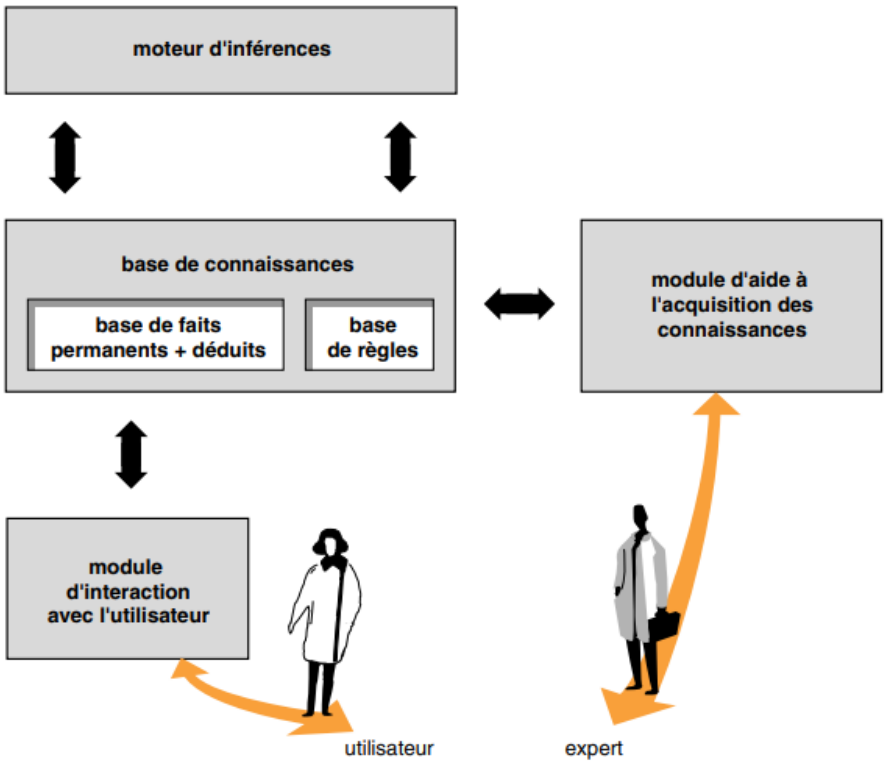


Fig. I.1 : Architecture d'un système expert [9]

4.2 Système Multi-Agent

Un système multi-agent (MAS or "self-organized system") est un système informatisé composé de plusieurs agents (processus, bots, personnes) intelligents en interaction. Les systèmes multi-agents peuvent résoudre des problèmes qui sont difficiles ou impossible à résoudre pour un agent individuel ou un système monolithique. [10]

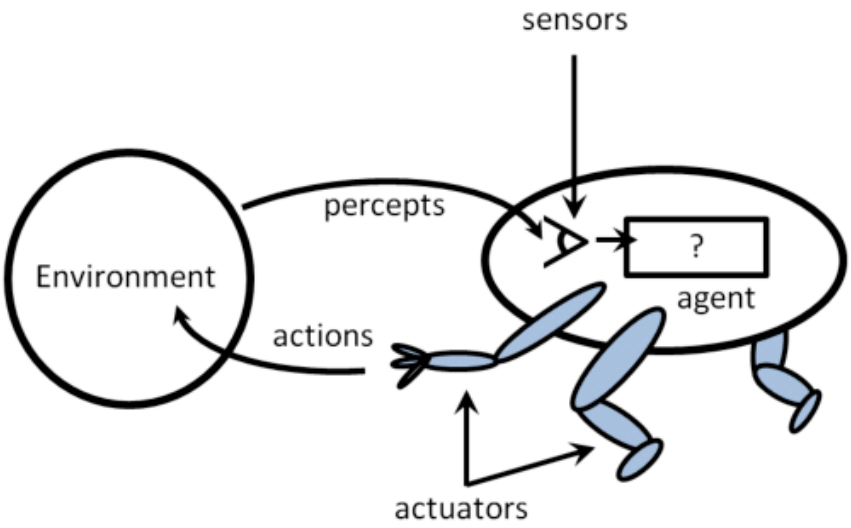


FIG. I.2 : Interaction de l'agent intelligent avec l'environnement [11]

4.3 Réseau de neurones

Les réseaux de neurones sont un ensemble d'algorithmes inspirés du cerveau humain le but de cette technologie est de simuler l'activité du cerveau humain, plus spécifiquement, la reconnaissance de motifs et le transfert d'informations entre différentes couches de connexions neuronales, les réseaux neuronaux peuvent s'adapter aux changements d'entrée, le réseau génère ainsi le meilleur résultat possible sans avoir à redéfinir les critères de sortie. [12]

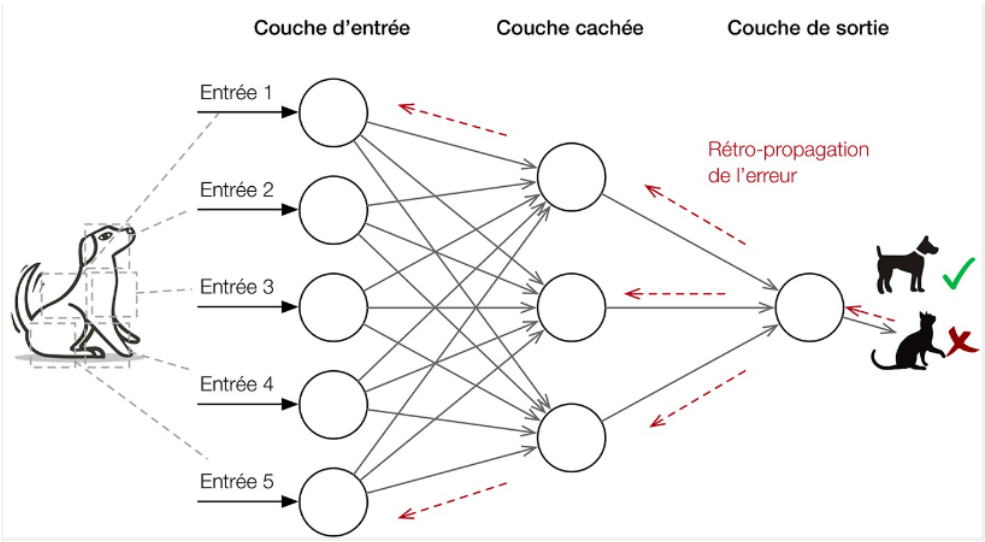


FIG. I.3 : Architecture et fonctionnement d'un réseau de neurones artificiel simple [13]

5 Les domaines de l'intelligence artificielle :

La recherche en intelligence Artificielle a permis de réaliser d'importants progrès dans la dernière décennie et ce dans différents domaines, et nous le trouvons dans des domaines beaucoup plus impressionnants à la limite de la science-fiction, comme la robotique ou les jeux. nous allons citer quelques exemples :

5.1 La sécurité/sûreté

La sûreté régalienne a été parmi les premiers domaines d'application de ces méthodes, elles sont par exemple utilisées pour détecter des comportements inhabituels dans des images de vidéosurveillance, ou pour reconnaître automatiquement des parties d'images, les caméras ou les drones de surveillance équipées d'intelligence artificielle sont l'une des technologies les plus populaires, ces vidéo offrent des capacités de surveillance à distance et des capacités avancées d'intelligence artificielle, leur système d'analyse envoie des alertes basées sur des scénarios prédéterminés ou des images de situations à haut risque, telles qu'un criminel identifié entrant dans un bâtiment ou un vandalisme présumé dans un guichet automatique. [14] [15]

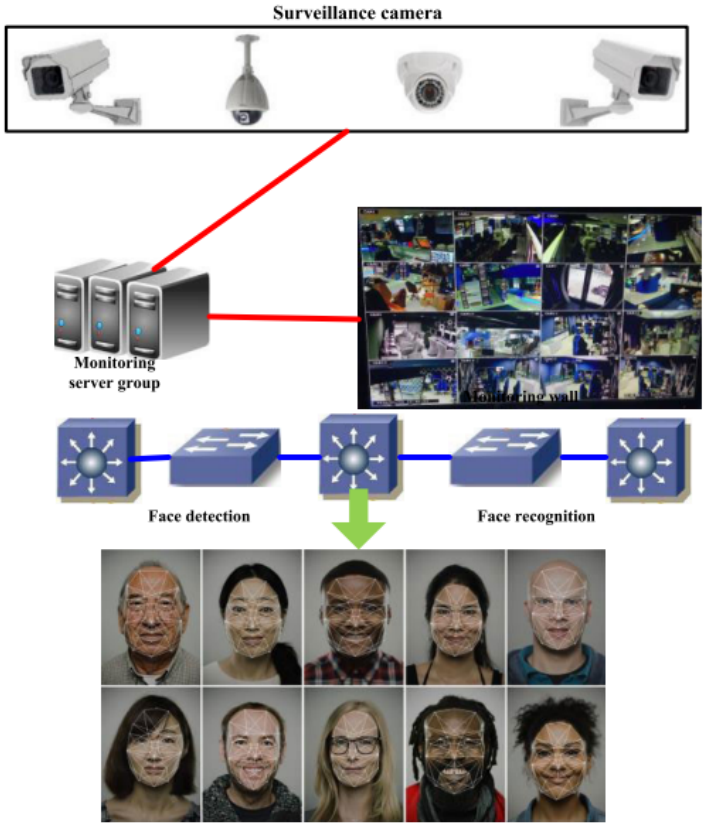


Fig. I.4 : caméra de surveillance utilisant l'IA [14]

5.2 Les voitures autonomes

L'intelligence artificielle a de nombreuses applications dans les voitures autonomes. Ils impliquent principalement l'apprentissage automatique de la conduite - apprendre à une voiture comment se comporter en cas d'accident, L'IA est également impliquée dans la vérification de la qualité des composants automobiles dans le cadre de la maintenance prédictive, la compréhension de l'environnement grâce aux données remontées par des capteurs, et aussi utilisée pour analyser le comportement du conducteur ou la cybersécurité pour surveiller l'état de la connectivité et empêcher tout piratage. [16]

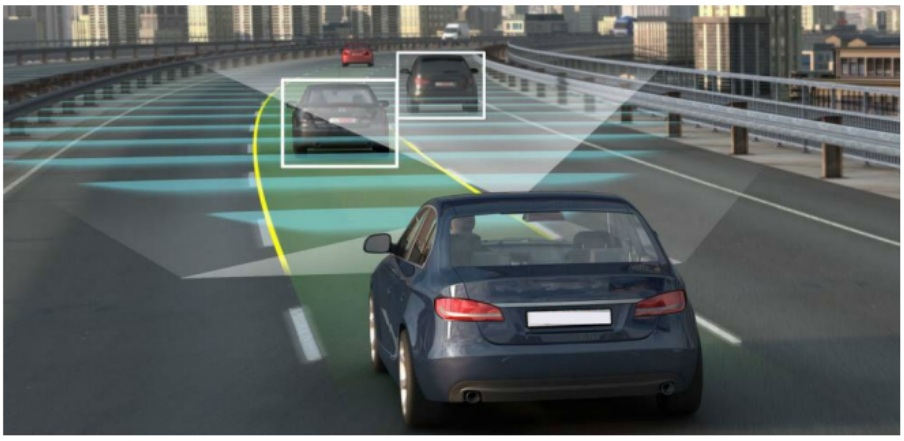


Fig. I.5 : Voiture Intelligente [16]

5.3 La santé

L'intelligence artificielle pénètre tous les aspects de l'industrie médicale, elle est déjà meilleure que les médecins pour détecter la nature cancéreuse du mélanome ou analyser les radiographies pulmonaires, le système Watson d'IBM aide à diagnostiquer et à prescrire, notamment en se basant sur le séquençage de l'ADN des tumeurs cancéreuses. [17] [18]

Fig. I.6 : L'intelligence artificiel dans le domaine médecine [19]

5.4 Commerce

Les acheteurs en ligne fournissent une mine de données que l'IA peut utiliser pour créer une « expérience client » sur mesure car, le comportement passé du consommateur permet de prévoir ses besoins, de ce fait, le e-commerce a multiplié les chatbots et les assistants virtuels personnalisés. Certains magasins physiques testent des systèmes de reconnaissance faciale pour suivre les parcours et la satisfaction des clients. [20]

FIG. I.7 : L'intelligence artificiel Chatbot dans le domaine commerce

6 Apprentissage automatique

L'apprentissage automatique consiste à extraire les connaissances à partir des données. Il s'agit d'un domaine de recherche à l'intersection des statistiques, de l'intelligence artificielle et de l'informatique et il est également connu sous le nom d'analyse prédictive ou d'apprentissage statistique. L'application des méthodes d'apprentissage automatique est devenue côte à côte avec notre vie quotidienne au cours des dernières années. Des recommandations automatiques des films à regarder, à ce que la nourriture à commander ou quels produits à acheter, à la radio en ligne personnalisée et la reconnaissance de vos amis dans vos photos, de nombreux sites Web et appareils modernes ont des algorithmes d'association de machines à leur cœur. Lorsque vous regardez un site web complexe comme Facebook, Amazon ou Netflix, il est très probable que chaque partie du site contient plusieurs modèles d'équipe de machines. En dehors des applications commerciales, l'apprentissage automatique a eu une influence énorme sur la façon dont la recherche sur les données est effectuée aujourd'hui. [21]

les principales composantes de l'apprentissage automatique sont :

- Une base de données.
- Une fonction de perte ou fonction d'objectif (la définition du but pour l'agent).
- Un modèle.

- Une méthode d'optimisation.

6.1 Les types d'apprentissage automatique

6.1.1 Apprentissage supervisé

Une méthode d'apprentissage automatique qui s'appuyant sur des données ou des exemples labellisés (étiquetés ou annotés) pour entraîner des modèles d'intelligence artificielle (IA) prédictifs, dans laquelle la machine peut apprendre une relation f : x → y qui relie x à y en ayant analysé des d'exemples d'associations x → y, [22], la classification et la régression sont des types d'apprentissage supervisé :

- La classification : Elle donne la prédiction de Oui ou Non, par exemple, "Cette tumeur est-elle cancéreuse ?", "Ce cookie répond-il à nos normes de qualité ?"
- La régression : Elle donne la réponse à "Combien".

Les principaux algorithmes d'apprentissage supervisé :

- Arbres de classification et de régression
- K-NN.
- Naïve Bayes Classifier.
- Régression linéaire.
- Régression logistique.

6.1.2 Apprentissage non supervisé

Contrairement a l'apprentissage supervisé il n'y a pas d'étiquetage ou de résultats corrects, la tâche consiste à découvrir la structure des données : par exemple, en regroupant des éléments similaires en "clusters", ou en réduisant les données à quelques "dimensions" importantes. [23]

Les principaux algorithmes d'apprentissage non supervisé :

- Méthode des K-moyennes.
- Clustering hiérarchique.
- Algorithme Apriori.
- Analyse en composantes principales (ACP).
- Décomposition en valeurs singulières (SVD).

6.1.3 Apprentissage semi-supervise

L'apprentissage semi-supervisé est une branche de l'apprentissage automatique qui vise à combiner l'apprentissage non supervisé et l'apprentissage supervisé, en général, les algorithmes d'apprentissage semi-supervisé tentent d'améliorer les performances de l'une de ces deux tâches en utilisant des informations généralement associées à l'autre, en ce sens, l'apprentissage semi-supervisé est à mi-chemin entre les apprentissages supervisé et non-supervisé : il cherche à exploiter les données non-étiquetées pour apprendre la relation entre les exemples et leur étiquette. [24]

7 Apprentissage profond

L'apprentissage profond est l'une des méthodes d'apprentissage automatique qui connaît des avancées importantes en intelligence artificielle dans les dernières années, et il utilise des réseaux neuronaux artificiels multicouches (Figure I.8), parmi les exemples récents où l'apprentissage profond a fait de grands progrès dans l'apprentissage automatique, citons la classification des images , traduction linguistique, reconnaissance vocale, entre autres. [25]

L'apprentissage profond est basé sur ce que l'on appelle des "réseaux de neurones artificiels", constitués de milliers d'unités ("neurones"), chacune effectuant de petites opérations simples, les résultats des "neurones" de la première couche servent d'entrée aux calculs de la deuxième couche, et ainsi de suite, par exemple, pour la reconnaissance visuelle, des premières couches d'unités identifient des lignes, des courbes, des angles… des couches supérieures identifient des formes, des combinaisons de formes, des objets, des contextes.

Des progrès dans l'apprentissage en profondeur ont été rendus possibles, notamment en augmentant la puissance des ordinateurs et le développement de grandes bases de données. [26]

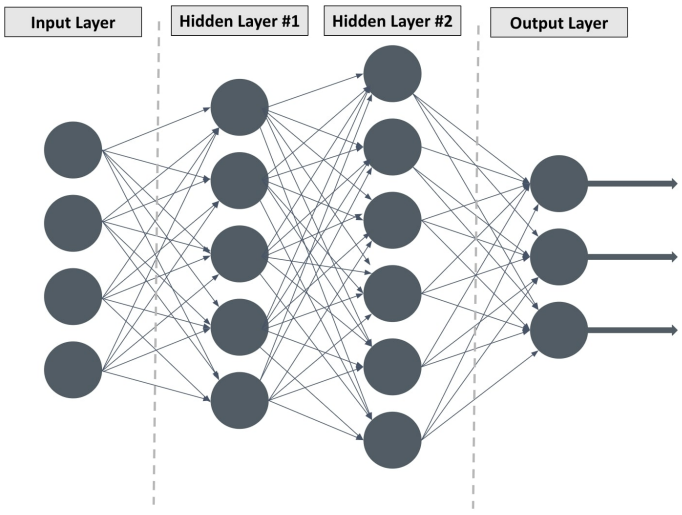


Fig. I.8 : Structure d'un réseaux de neurones profond

7.1 Pourquoi L'apprentissage profond ?

Ce qui différencie l'apprentissage profond de ces techniques d'apprentissage est d'utiliser plusieurs couches cachées, plus puissant que nous pouvons comprendre le véritable potentiel de l'apprentissage en profondeur. Une grande différence entre le deep learning et les algorithmes traditionnels de machine learning c'est-à-dire qu'il s'adapte bien, plus la quantité de données fournies est grande, meilleures sont les performances des algorithmes d'apprentissage profond sont meilleurs, contrairement à plusieurs algorithmes classiques du machine learning qui ont une limite supérieure sur la quantité de données qu'ils peuvent recevoir Parfois appelés "plate-forme de performance", les modèles d'apprentissage en profondeur n'ont pas un tel limites (théoriques), dépassant même la performance humaine traitement d'images, etc... [27]

Une autre différence entre les algorithmes ML traditionnels et les algorithmes d'apprentissage en profondeur c'est l'étape d'extraction des caractéristiques. Dans les algorithmes ML traditionnels L'extraction des caractéristiques se fait manuellement, ce qui est une étape difficile et coûteuse temps et besoin d'un expert dans le domaine, qu'en Deep Learning cette étape est exécutée automatiquement par l'algorithme. [28]

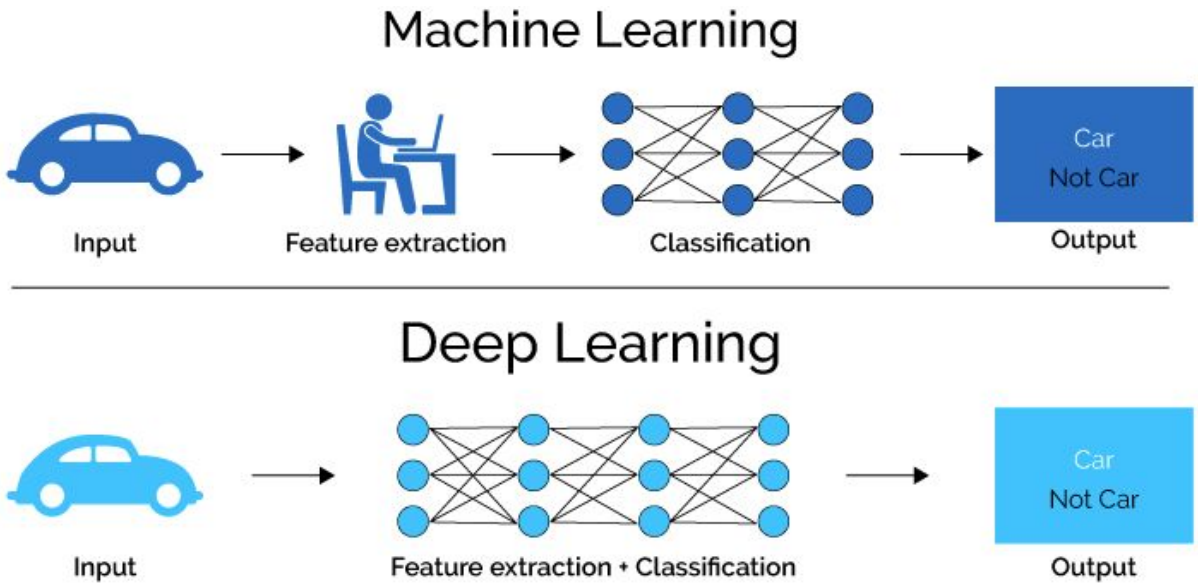


Fig. I.9 : Comparaison des processus Machine learning classiques et du Deep Learning [29]

7.2 Les algorithmes de l'apprentissage profond

Les différents algorithmes de Deep Learning ;

- Réseaux neuronaux convolutifs (CNN)
- Réseaux neuronaux récurrents (RNN)
- Réseaux de fonction de base radiale (RBFN)

- Réseaux de mémoire à long et court terme (LSTM)
- Réseaux adversariaux génératifs (GAN)
- Machines de Boltzmann restreintes (RBM)

les réseaux de neurones convolutionnels (CNN) ont une grand importance dans notre sujet donc nous allons l'expliquer :

7.3 Les réseaux de neurones convolutionnels

Un réseau de neurones convolutionnels (CNN) est un type de réseau de neurones artificiels utilisé dans la reconnaissance et le traitement des images, qui est spécifiquement conçu pour traiter les données de pixels, CNN est devenue dominante dans diverses tâches de vision par ordinateur. [30]

En entrée du réseau de neurones convolutionnels , l'image est fournie sous forme d'une matrice de pixels. cette dernière a deux dimensions en niveaux de gris (Largeur et Hauteur) par exemple : 112 × 112 pixels pour l'image, où la première valeur 112 représente la largeur de l'image et la seconde valeur représente la hauteur, la couleur est représentée par une troisième dimension de profondeur (3 couches de ces 112 × 112 pixels, chacune correspondante à une colleur : rouge , bleu et vert).

7.3.1 Les composants de base d'un réseau de neurones convolutionnels

Les réseaux de neurones convolutionnels sont généralement composés des couches suivantes :

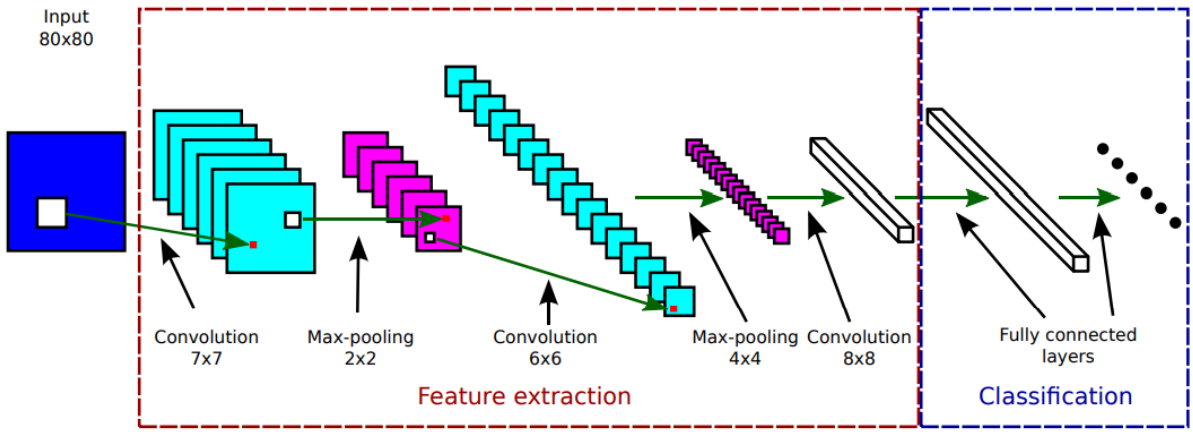


FIG. I.10 : Exemple d'un réseaux de neurones convolutionnels [31]

- **Convolution** :La couche convolutionnelle utilise des filtres qui scannent l'entrée, suivant ses dimensions en effectuant des opérations de convolution, elle peut être réglée en ajustant la taille du filtre (Kernel) et le stride, la sortie de cette opération est appelée feature map ou activation map, trois hyperparamètres sont utilisés pour déterminer le volume de la couche convolutive (volume de sortie). [32] :

– Dimensions et nombre des filtres (Kernels) : par exemple un filtre de taille F × F appliqué à une entrée contenant C canaux est un volume de taille F × F × C qui effectue des convolutions sur une entrée de taille I × I × C et qui produit un feature map de sortie de taille S × S × 1.

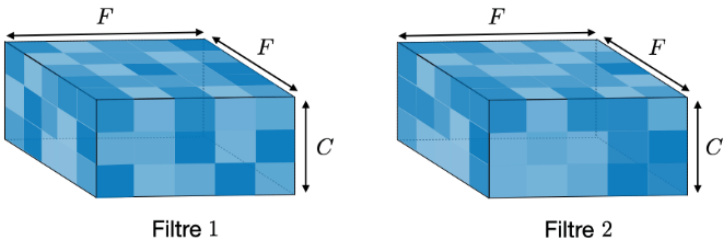


FIG. I.11 : Les Dimensions des filtres

– Le pas (Stride) : le stride est un paramètre qui représente le nombre de pixels que la fenêtre déplace après chaque opération, le pas contrôle le chevauchement des champs récepteurs, plus le pas est petit, plus les champs récepteurs se chevauchent et plus le volume de sortie sera grand.

FIG. I.12 : L'opération de pas

– **Padding** : Il est parfois pratique de placer des zéros aux limites du volume d'entrée, la marge contrôle la dimension spatiale du volume de sortie, en particulier, il est parfois nécessaire de conserver la même surface que le volume d'entrée.

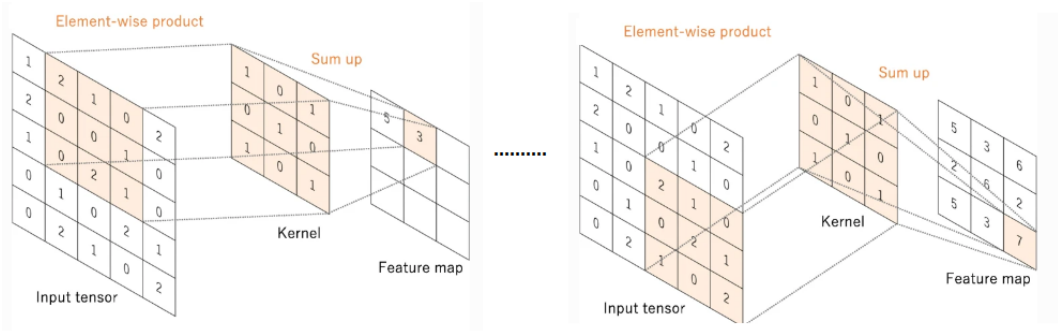


FIG. I.13 : Exemple explicative d'une convolution [30]

- **Max pooling :** L'objectif principale d'une couche Pooling est de réduire la taille d'une image en appliquant une opération de sous-échantillonnage à chaque canal d'entrée, en préservent les caractéristiques (features) de l'image, pour ce faire, on divisez l'image en cellules régulières et on conserve la valeur maximale dans chaque cellule.

Ainsi, la sortie après la couche de max-pooling serait une carte contenant les caractéristiques les plus importantes de la précédente carte (feature map). [32]

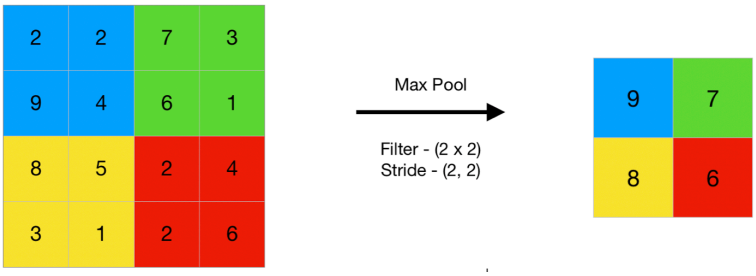


Fig. I.14 : Exemple de fonctionnement d'une couche de Max-Pooling

8 Conclusion

L'intelligence artificielle est un domaine encore jeune mais qui reste très vaste, nous avons consacré ce chapitre à la présentation des notions importants en apprentissage automatique, évidemment, il existe d'autres concepts et d'autres technologies d'apprentissage, nous avons pris soin d'expliquer ce que est essentiel pour notre travail.

Ce chapitre a juste été une introduction vers le deuxième chapitre de notre thème intitulé la détection des objets.

Chapitre II

Détection des objets

1 Introduction

Récemment, il y a eu de nombreuses avancées dans le domaine de l'intelligence artificielle grâce à l'apprentissage en profond et au traitement d'images, la reconnaissance d'images et même la recherche d'objets dans les images sont désormais possibles à l'aide de GPU (Processeur graphique) standard, le deep learning et notamment les modèles de CNN, ont permis d'atteindre des niveaux performances inégalés en computer vision.

La détection d'objet est une tâche de vision par ordinateur importante, qui traite la détection des instances d'objets visuels d'une certaine classe (comme les humains, les animaux ou les voitures) dans des images numériques, l'objectif de la détection d'objets est de développer des modèles informatiques qui fournissent des informations fondamentales dont ont besoin les applications de vision par ordinateur : "Quels objets, sont où ?". [33]

Dans ce chapitre l'accent sera mis sur : qu'est-ce que la détection d'objets et comment elle a évolué dans les dernières années, les techniques de détection d'objet, les méthodes de détection d'objets de vision par ordinateur, nous listons des exemples, et des cas d'utilisations, ainsi les algorithmes de détection d'objets les plus récents, Et finalement nous allons présenter les nouveaux algorithmes de reconnaissance d'objets.

2 Définition de la détection d'objet

La détection d'objet est la procédure qui consiste à déterminer l'instance de la classe à laquelle l'objet appartient et d'estimer l'emplacement de l'objet en produisant la boîte de délimitation autour de l'objet, la détection d'une seule instance de l'image est appelée détection d'objet de classe unique, alors que la détection des classes de tous les objets présents dans l'image est connue sous le nom de détection d'objets multi-classes. [34]

3 Pourquoi la détection d'objets est-elle importante ?

La détection d'objets est l'un des problèmes fondamentaux de la vision par ordinateur, il constitue la base de nombreuses autres tâches de vision par ordinateur, tels, la segmentation, le sous-titrage d'image, le suivi des objets, et plus encore, les applications de détection d'objets spécifiques comprennent la détection des piétons, le comptage des personnes, la détection de visages, la détection des voiture, la détection de textes ou la reconnaissance de plaques d'immatriculation. [33]

4 Les modèles de détection les plus performants par CNN

Il existe deux types de modèles de détection d'objets pour les architectures CNN [35] :

- **Modèle de détection a une étape** : Les détecteurs à une étape prédisent des

boîtes de délimitation sur les images sans l'étape de proposition de région, et comprend des modèles tels que : yolo, SDD, RetinaNet , yolov3, yolov4, yolov5, yoloR ...

Ce processus consomme moins de temps et peut donc être utilisé dans des applications en temps réel.

- **Modèle de détection en deux étapes** : Au contraire, ce modèle propose la région suivi de la classification des objets en fonction des caractéristiques extraites de la région proposée avec régression par boîte limitative, et comprend des modèles tels que : Fast RCNN, Faster RCNN, Mask R-CNN, RCNN, SPPNet, Pyramid Networks/FPN, G-RCNN ...

  Les méthodes à deux étapes atteignent la plus grande précision de détection, mais sont généralement plus lentes, en raison des nombreuses étapes.

Le principal avantage des modèles de détection en une étape est que ces algorithmes sont généralement plus rapides que les détecteurs à plusieurs étapes et structurellement plus simples. [35]

Il existe plusieurs modèles de détection d'objet nous avons présenter l'histoire des modèles de détection des objets dans cette image (figure II.1) dans un ordre chronologique [33] :

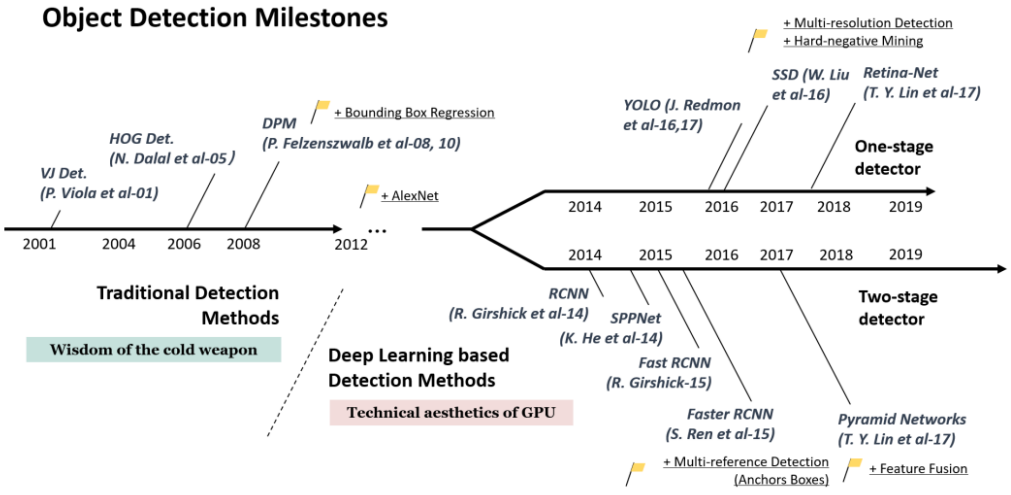


Fig. II.1 : Histoire de la détection des objets [33]

4.1 R-CNN

Pour contourner le problème de la sélection d'un grand nombre de régions, R-CNN a été proposé par Ross Girshick et en 2014 [36] dont le principe consiste à effectuer une recherche sélective pour extraire seulement 2000 régions de l'image et il les a appelées propositions de région.

Pour traiter le problème de la localisation des objets, cet algorithme va suivre les étapes suivants : dans la première étape le réseau de neurones de convolution prend en

entrée une image de taille quelconque ensuite il va produire des régions d'intérêts dans lesquelles les R-CNN pourraient les détecter, ces régions seront envoyées au second réseau afin d'identifier la présence de l'objet dans cette région (Figure II.2)

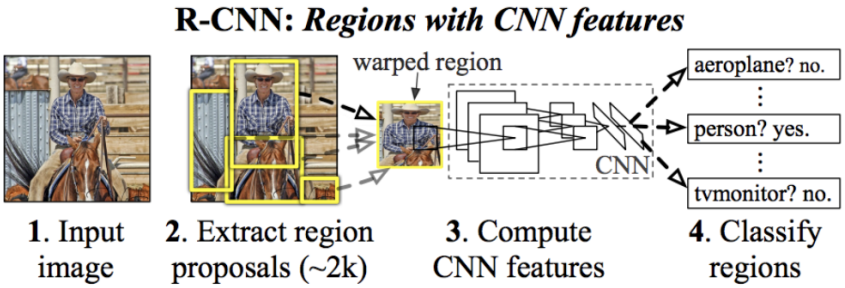


Fig. II.2 : Architecture de R-CNN [36]

4.2 FAST R-CNN

Le même auteur de la définition précédente (Ross Girshick | R-CNN) a résolu certains des inconvénients de R-CNN pour créer un algorithme de détection d'objets plus rapide, c'est Fast R-CNN. [37]

L'idée est similaire à l'algorithme R-CNN. Mais, au lieu de transmettre les propositions de région au CNN, ils transmettent l'image d'entrée au CNN pour générer une carte de caractéristique convolutionnelles.

Ensuite, la carte de caractéristiques convolutives générée après ces couches est introduite dans une couche de mise en commun du RoI qui extrait un vecteur de caractéristiques de longueur x fixe afin qu'elle puisse être introduite dans une séquence de couches entièrement connectée (FC) qui sont connectées à deux couches de sortie :

- Une couche Softmax qui produit des estimations de probabilité pour les classes d'objet.
- Une couche à valeurs réelles qui produit les coordonnées des boîtes de délimitation calculées par régression.

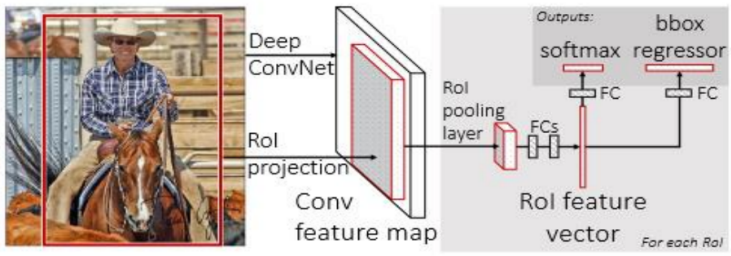


Fig. II.3 : Architecture de Fast R-CNN [37]

4.3 Faster R-CNN

Faster R-CNN est une extension de Fast R-CNN, Comme son nom l'indique, Faster R-CNN est plus rapide que Fast R-CNN grâce au réseau de proposition de région (RPN) [38], il se décompose en deux modules principaux :

- Le premier module est un réseau convolutionnel profond qui crée des cartes de caractéristiques convolutions utilisables par le module RPN (Region Proposal Network) cette carte (de n'importe quelle taille) et générer un ensemble d'objets rectangulaires, chacune avec un score de précision.
- Le deuxième module est le détecteur Fast R-CNN, qui utilise la région proposée comme l'architecture Fast R-CNN précédente.

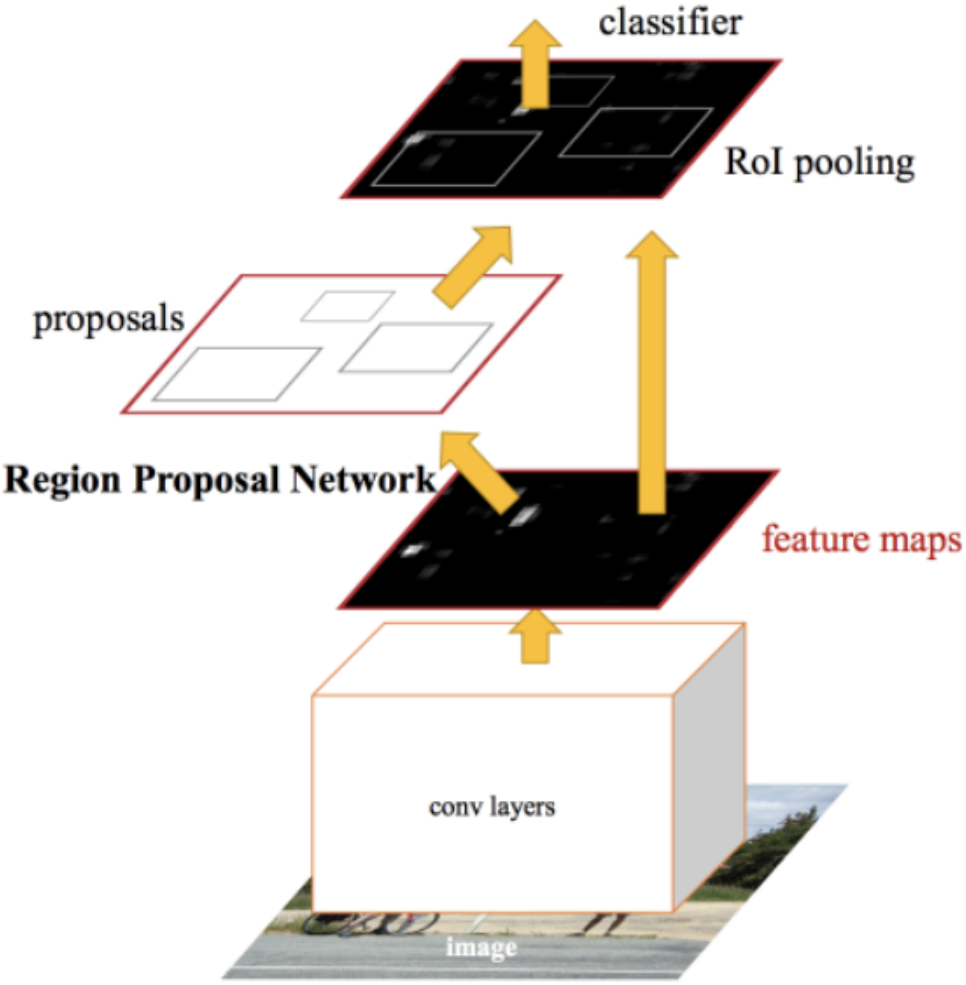


Fig. II.4 : Architecture de Faster R-CNN [38]

4.4 Mask R-CNN

Le masque R-CNN est une extension du faster R-CNN, faster R-CNN prédit les boîtes englobantes et le Mask R-CNN ajoute essentiellement une autre branche pour prédire un

masque d'objet en parallèle. [39]

La couche masque est un petit FCN (Fully Convolutional Network) appliqué à chaque RoI (Region of Interest), prédisant un masque de segmentation d'une manière pixel à pixel, ce principe d'alignement pixel à pixel c'est la pièce manquante du Fast/Faster R-CNN.

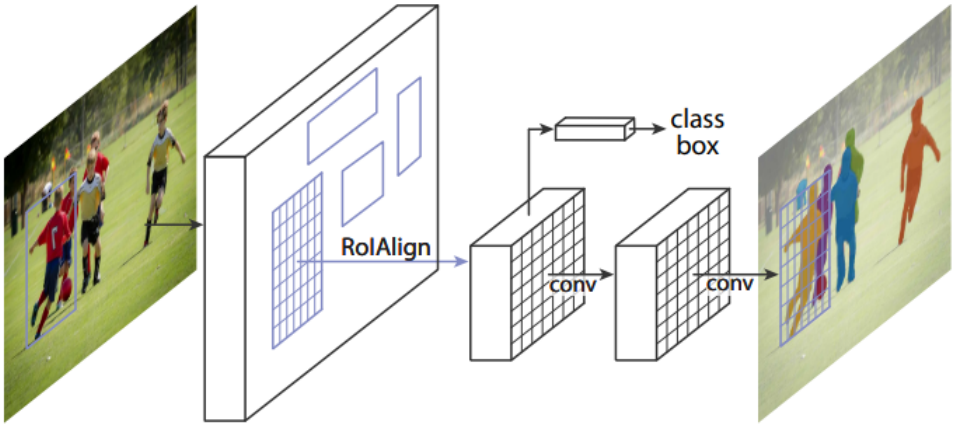


Fig. II.5 : Architecture de Mask R-CNN [39]

4.5 SSD : Single Shot MultiBox Detector

SSD : Single Shot MultiBox Detector (par C. Szegedy et al.) l'article a été publié le Novembre 2016 [40], SSD C'est aussi un réseau convolutionnel qui a divisé sa phase de convolution en deux opérations plus simples et rapides.

Il a établi de nouveaux records de performance et de précision avec plus de 74 % mAP (précision moyenne) avec 59 images par seconde sur des ensembles de données standard comme PascalVOC et COCO.

Afin pour mieux comprendre le SSD (Figure II.6) :

- Single Shot : cela signifie que les tâches de localisation et de classification d'objets se font en un seul passage vers le réseau.
- MultiBox : c'est le nom d'une technique de régression de boîte de délimitation développée par Szegedy et al [40].
- Détecteur : Un détecteur d'objets qui classe également les objets détectés.

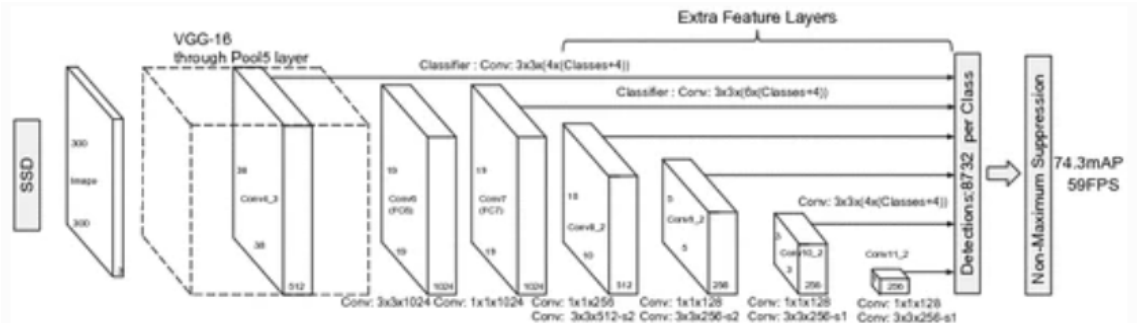


Fig. II.6 : Architecture de SSD : Single Shot MultiBox Detector [40]

4.6 YOLO

YOLO qui est une abréviation de You Only Look Once, YOLO est l'un des algorithmes de détection d'objets les plus populaires utilisé par les chercheurs du monde entier, il a été décrit pour la première fois en 2015 dans l'article de Joseph Redmon. [41]

Ce réseau utilise les caractéristique de l'image d'entrée pour prédire les boites englobantes (Bounding box en anglais) de touts les classes de l'image simultanément, YOLO raisonne globalement sur l'ensemble de l'image et sur tout les objets qu'elle contient, la conception YOLO permet un apprentissage de bout en bout et des vitesses en temps réel (avec une vitesse jusqu'à 45 images/seconde) tout en maintenant une précision moyenne élevée. [41]

Le principe du modèle repose sur l'utilisation d'une série de couches convolution (figure II.7) avec différentes tailles et nombre de filtres, l'architecture du YOLO est composée de 24 couches de convolution suivi de deux couches entièrement connectées (fully-connected layers) (figure II.7), la dimension des images d'entrée est de 448x448. [41]

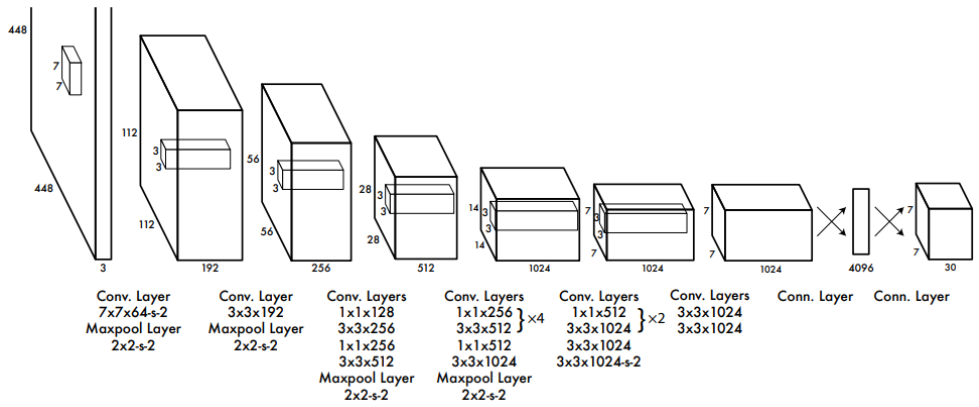


Fig. II.7 : Architecture de YOLO (You Only Look Once) [41]

Les sorties de l'architecture YOLO sont les coordonnées des zones englobantes (bounding boxes) et les probabilités des classes de chaque zone, la figure (II.8) montre un exemple de reconnaissance d'objets par le modèle YOLO.

FIG. II.8 : Exemple de détection d'objets par YOLO

Les étapes de fonctionnement de YOLO :

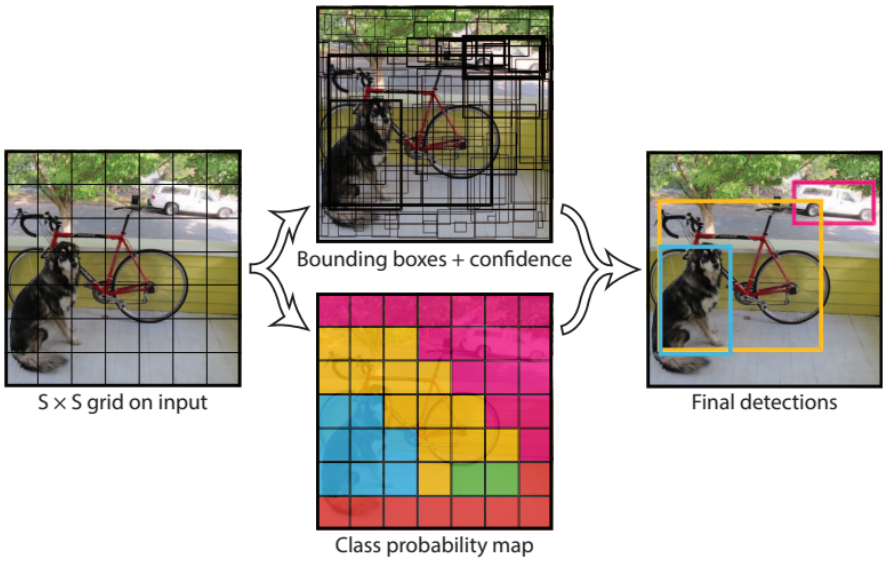


FIG. II.9 : Principe de fonctionnement de l'architecture YOLO [41]

- **Diviser l'image en cellules avec une taille S x S :** le modèle fonctionne en divisant d'abord l'image d'entrée en une grille de cellules par exemple dans la Figure (II.10 | S x S = 7 x 7) ce qui donne N (dans l'exemple N = 49) cellule de sortie.

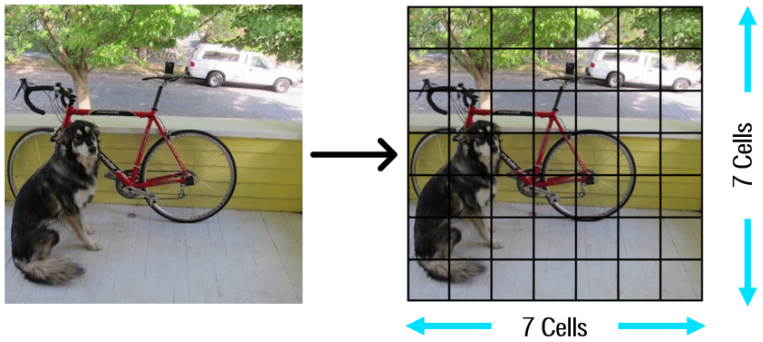


FIG. II.10 : Exemple diviser l'image en (S*S) grille [41]

- **Chaque cellule prédit B cadres de délimitation :** Après la divisions de l'image à N cellule, chaque cellule est responsable de la prédiction des cadres de délimitation avec un score de confiance de chaque cadre, la prédiction d'une classe est également basée sur chaque cellule, chaque cadre est représenter par cinq variables (X , Y , W, H , C) et aussi le score de confiance de la boite englobante [41], ces variables représente :
  - (X , Y) : Ces coordonnées prédites représente le centre du cadre (boîte englobante) par rapport aux limites de la cellule de grille.
  - W : Cette variable représente la largeur prédite de la boite englobante par rapport à l'image entière.
  - H : Représente la hauteur prédite de la boite englobante par rapport à l'image entière.
  - C : La prédiction de la classe de l'objet (par exemple : personne, chat, arbre, voiture ...).
  - S : Représente la prédiction de confiance représente le IoU entre la boîte prédite et toute boîte de vérité terrain.

  Exemple :

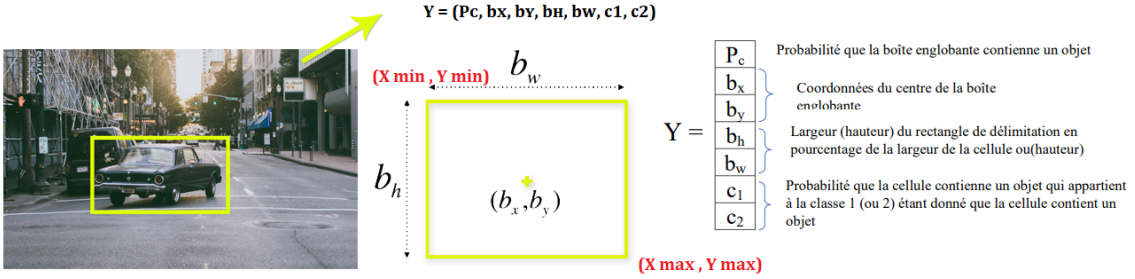


FIG. II.11 : Exemple d'un vecteur prédit dans le cas d'une seule boite englobante

On a la taille de l'image (412 x 224), pour calculer les valeurs de vecteur Y avec format YOLO :

- bx = ((X max + X min)/2) / 412
- by = ((Y max + Y min)/2) / 224

- bh = (Y max - Y min)/224
- bw = (X max - X min)/412

Pc= la prédiction de confiance représente le IoU entre la boîte prédite et la boîte de vérité terrain.

a) **Intersection sur Union (IoU)** : est une façon de mesurer la qualité de la détection d'un objet, qui est utilisée pour éliminer plusieurs boîtes qui entourent le même objet, en fonction de la boîte qui a une plus grande confiance. elle est utilisée pour déterminer les vrais positifs et les faux positifs dans un ensemble de prédictions, l'IoU est le rapport entre la surface de l'intersection des boite englobantes considérées et la surface de l'union des ensembles (Figure II.12). [42]

Fig. II.12 : Intersection sur Union (IoU)

Il y a des exemples de bons et de mauvais scores Intersect sur Union dans l'image ci-dessus :

Fig. II.13 : Exemple intersection sur Union (IoU)

Comme vous pouvez le voir, les cadres de délimitation (boîtes englobantes) prédites qui se chevauchent fortement ont des scores plus élevés que celles qui se chevauchent moins, cela fait de l'intersection sur l'union une excellente mesure pour évaluer les détecteurs d'objets. [42]

b) **Boîte d'ancrage (Anchor Box) :** S'il y a plusieurs boîtes dans la même cellule, dans ce cas les boîtes d'ancrage jouent un rôle important qui sépare les objets.

Nous avons vu la représentation d'un seul vecteur dans un cellule, dans le cas où il y a plusieurs cadres dans une cellule en représente le vecteur comme suite :

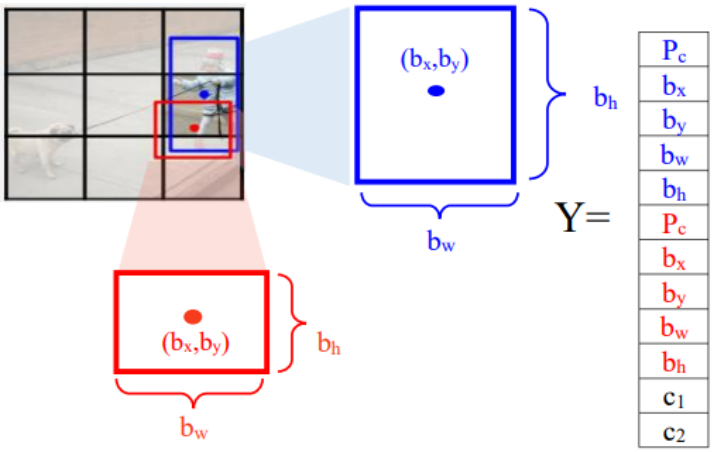


FIG. II.14 : le vecteur prédit dans le cas de plusieurs cadres dans une cellule

Donc, l'image qui est divisée en une grille S x S, chaque cellule de cette grille prédit B cadres de délimitation, un score de confiance pour chaque cadre et la classe C de l'objet [42], toutes ces prédictions sont représentées dans une forme d'un tenseur comme montre l'image suivante (Figure II.15) :

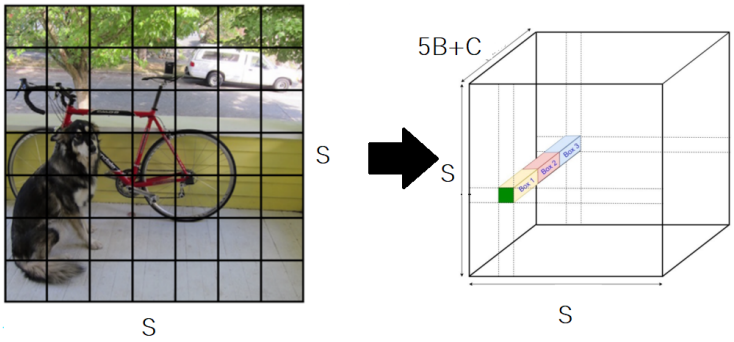


FIG. II.15 : La spécification des emplacements de la boîte englobante et probabilités de classe avec un tenseur

La représentation mathématique du tensor : $S \times S \times (B * 5 + C)$

c) **Suppression non maximale :** les objets présents dans l'image peuvent être de tailles et de formes différentes, pour capturer parfaitement chacun d'entre eux, les algorithmes de détection d'objets créent plusieurs boîtes englobantes, mais pour chaque objet de l'image, nous devons avoir une seule boîte englobante.

Nous comprenons donc maintenant pourquoi nous avons besoin de la suppression non maximale et à quoi il sert, plus précisément l'objectif de cette technique est de sélectionner la meilleure boîte englobante pour un objet et de rejeter ou de " supprimer " toutes les autres boîtes englobantes [43], donc les étapes de cette technique sont :

- Sélectionner la boîte avec le score le plus élevé.
- Ensuite, compare le chevauchement (intersection sur union) de cette boîte avec d'autres boîtes.
- Supprimez les boîtes englobantes dont le chevauchement (intersection sur union) est >50
- Passez ensuite au score d'objectivité le plus élevé suivant.
- Enfin, répétez les étapes 2 à 4 jusqu'à fini tous les objets dans l'image.

Donc, il faut selectionner une seul boite (La meilleure boite)

FIG. II.16 : exemple d'une suppression non maximale [43]

Ces étapes de fonctionnement rend le modèle YOLO très rapide, 1000 fois plus rapide que le R-CNN et 100 fois que fast R-CNN, cette rapidité a permis YOLO d'être exécutable en temps réel. [44]

Plusieurs versions du YOLO ont été proposées à ce jour, on trouve : YOLO V1, YOLO V2, YOLO V3, YOLO V4, chaque version a pour but d'améliorer les performances du modèle en matière de rapidité et taux de classification.

Tous les modèles de base de YOLO ont été entraînées sur la base de données de COCO, mais il existe dans la littérature d'autres modèles entraînés sur d'autres bases de données telles que ImageNet de Google.

On va voire d'autre versions plus améliorer de la famille YOLO :

4.6.1 YOLO v2

YOLO v2 utilise quelques astuces pour améliorer l'entraînement et augmenter les performances, voila les nouveau changement dans ce modèle :

- **Batch Normalization :** l'inclusion de couches de normalisation par lots après chaque couche de convolution, en ajoutant la normalisation par lots aux couches convolutives dans l'architecture, la MAP (précision moyenne) a été améliorée de 2 %, cela a également aidé le modèle à se régulariser et le surajustement (Overfitting) a été réduit. [45]

- **L'augmentation de la résolution d'image d'entrée :** la taille d'entrée dans yolo v2 a été augmentée de 224*224 à 448*448, l'augmentation de la taille d'entrée de l'image a amélioré le MAP (précision moyenne moyenne) jusqu'à 4 % [45]

- **Les petit caractéristiques :** l'un des principaux problèmes à résoudre dans le yolo v1 est la détection d'objets plus petits sur l'image, cela a été résolu dans le yolo v2 qui divise l'image en 13*13 cellules de grille, ce qui est plus petit par rapport à sa version précédente. Cela permet au yolo v2 d'identifier ou de localiser les objets plus petits dans l'image et aussi efficace avec les objets plus grands.[45]

- **multi-échelles (Multi-Scale Training :**)sur yolo v1, la détection d'objets avec différentes tailles d'entrée est faible, il a des problèmes pour détecter le même objet sur une image de plus grande taille, cela a été résolu dans une large mesure dans yolo v2 où il est formé avec des images aléatoires avec différentes dimensions allant de 320*320 à 608*608, cela permet au réseau d'apprendre et de prédire avec précision les objets à partir de différentes dimensions d'entrée. [45]

- **Darknet 19** : yolo v2 utilise l'architecture Darknet 19 avec 19 couches convolutives et 5 couches de regroupement maximum et une couche softmax pour les objets de classification, l'architecture du Darknet 19 est présentée ci-dessous (Figure II.17), darknet est un framework de réseau neuronal écrit en Clanguage et CUDA, c'est vraiment rapide dans la détection d'objets, ce qui est très important pour la prédiction en temps réel. [45]

Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 2/2	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Maxpool		2 × 2/2	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	1000	1 × 1	7 × 7
Avgpool		Global	1000
Softmax			

Fig. II.17 : Architecture de Darknet-19 [45]

4.6.2 YOLO v3

Joseph Redmon et Ali Farhadi en 2018 [46], on aussi fait une augmentation progressive, cela a causé aussi l'augmentation de la complexité de l'architecture de YOLO v3, mais cette complexité a été échangée contre des améliorations de précision dans YOLO v3.

Ici, nous verrons quelles sont les améliorations de YOLO v3 :

- **Bounding Box Predictions :** dans YOLO v3, donne le score des objets pour chaque boite englobante, il utilise la régression logistique pour prédire le score de confiance.[46]
- **Prédictions de classe :** dans YOLO v3, il utilise des classificateurs logistiques pour chaque classe au lieu de softmax qui a été utilisé dans le précédent YOLO v2. [46]
- **Feature Pyramid Networks (FPN) :** YOLO v3 fait des prédictions similaires au FPN où 3 prédictions sont faites pour chaque emplacement où l'image d'entrée et les caractéristiques sont extraites de chaque prédiction . [46]
- **Darknet-53** : le prédécesseur YOLO v2 utilisait Darknet-19 comme extracteur (Backbone) de caractéristiques et YOLO v3 utilise le réseau Darknet-53 pour l'extracteur de caractéristiques qui a 53 couches convolutionnelles, il est beaucoup plus profond que le YOLO, Darknet-53 se compose principalement de filtres 3x3 et 1x1 avec des connexions de raccourci. [46]

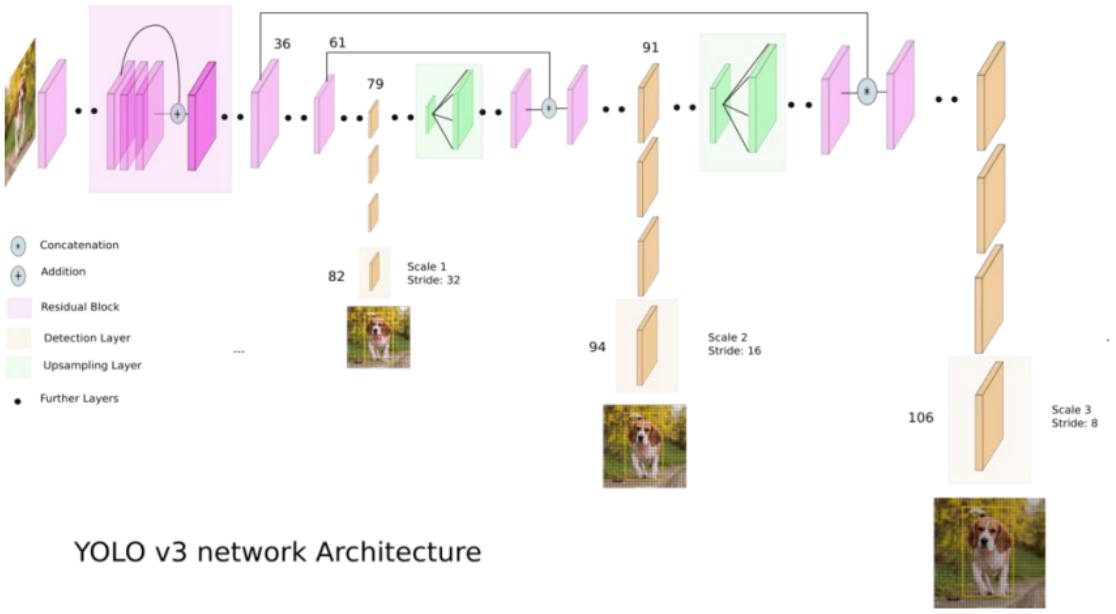


Fig. II.18 : Architecture de YOLO v3 [47]

4.6.3 YOLO v4

YOLO v4 est une amélioration de l'algorithme Yolov3 en améliorant la précision moyenne (mAP) jusqu'à 10 % et le nombre d'images par seconde de 12 %.

L'architecture YOLO v4 comporte 4 blocs distincts, Backbone, Neck, la prédiction dense (dense prediction) et la prédiction clairsemée (sparse prediction). [48]

Plus précisément YOLO v4 se compose de :

Backbone : CSPDarknet53, CSPDarkNet53 se compose de deux blocs :

- **Couche de base convolutive :** La couche de base convolutive se compose de la carte de caractéristiques entière
- **Bloc Cross Stage Partial (CSP) :** La stratégie Cross Stage Partial divise la carte des caractéristique dans la couche de base en deux parties une moitié sera envoyée à travers le bloc dense et l'autre moitié sera acheminée directement vers l'étape suivante sans aucun traitement pour les fusionne à l'aide de la hiérarchie Cross-stage, ce qui permet à plus de gradient de circuler à travers les couches et diminuer ainsi le problème de "Vanishing Gradient" [49]

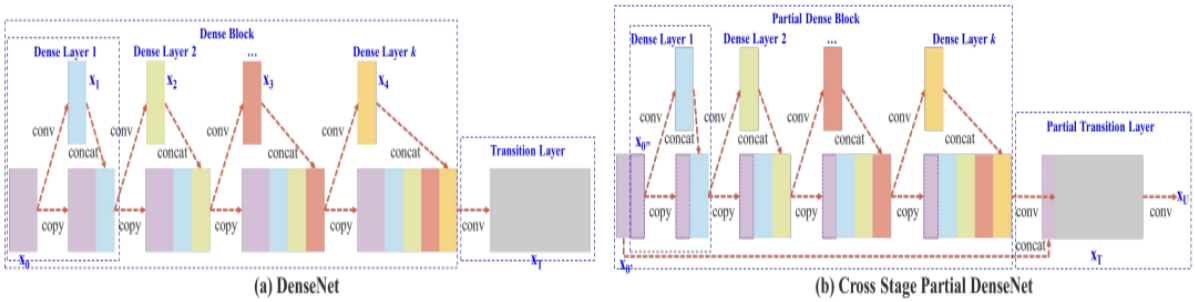


FIG. II.19 : L'architecture de la stratégie Cross Stage Partial (CSP) [49]

Neck : A pour rôle d'extraire les features pertinentes de l'ensemble des couches de la backbone, et de les combiner en features utiles à notre tâche de détection, plus précisément le cou (Neck) permet d'intégrer et combiner des features de différentes résolutions et complexités, le cou de yolo v4 ce compose de :

- **SPP - Bloc supplémentaire :** Un bloc supplémentaire appelé SPP (Spatial Pyramid Pooling) est ajouté entre le backbone CSPDarkNet53 et le réseau d'agrégation de fonctionnalités (PANet), ceci est fait pour augmenter le champ réceptif et séparer les caractéristiques contextuelles les plus importantes et n'a presque aucun effet sur la vitesse de fonctionnement du réseau. Il est connecté aux couches finales de CSPDarkNet. [50]
- **PANet (réseau d'agrégation de chemins) :** il correspond à une amélioration pour yolo, qui permet d'obtenir de meilleures prédictions, ceci est réalisé grâce à un travail sur la façon dont l'information se propage au sein du réseau, PANet a été conçu dans un premier temps pour réaliser une tâche de segmentation, il a donc été

légèrement adapté par les auteurs de yolov4 pour pouvoir réaliser de la détection d'objet. [51]

Head : yolov3, la tête est responsable de la décision finale du réseau, a partir des informations fournies par le cou, celle-ci décide où sont les éléments à détecter pour les localiser avec des boiter englobante

Yolov4 utilise :

- Sac de gratuité (Bag of Freebies, BoF) pour le backbone : CutMix et Mosaic, régularisation DropBlock, lissage des étiquettes de classe. [48]
- Sac de spécialités (BoS) pour le backbone : Activation de Mish, connexions partielles inter-étapes (CSP), connexions résiduelles pondérées à entrées multiples (MiWRC). [48]
- Sac de gratuité (BoF) pour le détecteur : Perte de CIoU, CmBN, régularisation DropBlock, augmentation des données en mosaïque, formation auto-adversaire, élimination de la sensibilité de la grille, utilisation de plusieurs ancrages pour les détecteurs, sensibilité de la grille, utilisation de plusieurs ancrages pour une seule vérité de base, planificateur de recuit en cosinus, hyperparamètres optimaux, formes d'entraînement aléatoires. [48]

yolov4 est yolov5 sont presque similaires, yolov5 va être utilisé dans notre implémentation donc on va voir plus de détails de l'architecture dont le chapitre suivant

5 Architecture d'un détecteur d'objet

Avant de voir le modèle en détail nous allons voir l'architecture d'un détecteur d'objets ordinaires il se compose de plusieurs parties (Input , Backbone, Neck, Head) :

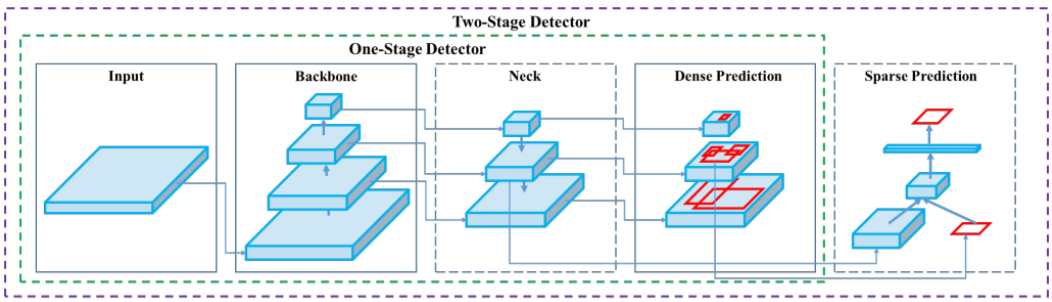


Fig. II.20 : Détecteur Architecture [48]

- **Entrée (Input)** : Image, patches, pyramide d'image
- **Backbone** : le backbone est un réseau neuronal convolutif (CNN) profond comme : VGG16 [52], ResNet-50 [53], SpineNet [54], EfficientNet-B0/B7 [55], CSPResNeXt50 [49], CSPDarknet53 [49], ce réseau est utilisé pour extraire des caractéristiques

visuelles de haut niveau à partir de l'image d'entrée, le backbone est généralement pré-entraîné sur un jeu de données de classification, généralement ImageNet [56], mais dans notre cas nous allons entraîner le réseau à zéro par ce que notre classe n'a jamais été entraîné avant dans les jeu de données populaire comme : ImageNet, Coco Dataset, PASCAL VOC, OpenImage ...

- **Cout (Neck)** : la structure placée entre la tête (Head) et le backbone dont l'objectif est de rassembler autant d'informations extraites par le Backbone que possible avant de les transmettre à la tête (Head), cette structure joue un rôle majeur dans le transfert des informations sur les petits objets en évitant qu'elles ne soient perdues vers des niveaux d'abstraction plus élevés, les blocs d'agrégation de chemins couramment utilisés dans le cou (Neck) sont : FPN [57], PAnet [51], NAS-FPN [58], BiFPN [59], ASFF [60], SFAM [61], Le point commun de ces méthodes est d'utiliser de façon répétée diverses méthodes de montée et de descente.
- **Tête (Head)** : Utilisé pour prédire les classes et les boîtes englobantes des objets, elle est généralement classée en deux catégories le détecteur d'objets à une étape et le détecteur d'objets à deux étapes :
  - **Prédiction dense (une étape)** :
    - RPN [38], SSD[40], YOLO [41], RetinaNet [62](basé sur l'ancrage)
    - CornerNet [63], CenterNet [64], MatrixNet[65], FCOS [66] (sans ancrage)
  - **Prédiction éparse (En deux étapes) :**
    - R-CNN plus rapide [38], R-FCN [67], Mask RCNN [39] (avec ancrage)
    - RepPoints [68] (sans ancre)

6 La mesure de la précision et rappel dans les algorithmes de détection d'objets :

La plupart des algorithmes de détection d'objets fonctionnent en deux étapes (combinées ou non) : la proposition de zones d'objets (zones qui pourraient contenir un objet) et la classification de ces zones, on va voir une méthode de mesure populaire pour mesurer la précision et le rappel des détecteurs d'objets, mais il faut savoir quelque indicateurs pour mesurer la performance d'un détecteur d'objet :

Ici on appelle donnée pertinente, une proposition qui contient réellement un objet de classe C et qui a été prédite avec la même classe C (Vrai Positif) ou une proposition qui ne contient pas la classe C et qui n'a pas été prédite comme contenant cette classe (Faux Négatif).

Par opposition, une proposition non pertinente désignera soit une proposition ne contenant pas une classe donnée C mais qui a été prédite comme contenant cette classe (Faux Positif) ou le contraire (Vrai Négatif).

Exemple :

Vrai positif Prédiction : Oussama présent Résultat attendu : oussama présent	**Faux positif** Prédiction : Oussama présent Résultat attendu : Oussama n'est pas présent
Faux négatif Prédiction : Oussama n'est pas présent Résultat attendu : Oussama présent	**Vrai négatif** Prédiction : Oussama n'est pas présent Résultat attendu : Oussama n'est pas présent

FIG. II.21 : Exemple si notre modèle prédit la présence de Oussama (personne) dans une image

- Précision : mesure à quel point les résultats sont précis par rapport à ce qui est proposé comme zone :

$$\text{Precision} = \frac{Vrai\ positif}{Vrai\ positif + Faux\ positif}$$

- Rappel : mesure à quel point des prédictions pertinentes sont proposées (combien y a t-il de propositions pertinentes parmi les zones proposées ?) :

$$\text{Recall} = \frac{Vrai\ positif}{Vrai\ positif + faux\ négatif}$$

- Accuracy : il indique le pourcentage de bonnes prédictions :

$$\text{Accuracy} = \frac{Vrai\ positif + Vrai\ négatif}{Total}$$

7 Conclusion

Les algorithmes et les modèles sont continuellement améliorés pour des meilleures performances dans la détection d'objets de façon générale.

Nous avons présenté dans ce chapitre un aperçu des méthodes de détection d'objets basées sur l'apprentissage en profondeur, nous avons vu les modelés de détection les plus connus et les plus performants avec ses architectures, nous avons basés sur les modèles de yolo qu'ils vont être important dans notre conception et nous avons vu aussi les étapes d'évolutions de ces modèles, finalement nous avons expliqué comment mesurer les performances d'un modèle de détection.

Dans le chapitre suivant, nous présenterons le modèle propos et nous expliqueront chaque module en détail.

Chapitre III

Les travaux antérieurs

De nombreux travaux ont été menés dans le domaine du traitement d'images, notamment dans le domaine de l'apprentissage profond. Cependant, très peu de travaux ont été réalisés dans notre domaine (détection d'objets), nous passons brièvement en revue certains travaux remarquables qui utilisent des méthodes de détection des palmiers pour obtenir un aperçu générale.

1 Travaux qui utilisent des méthodes d'intelligence artificielle pour la détection

Dans cet atricle [69] dans laquelle ils proposons une nouvelle méthode automatique basée sur l'apprentissage profond (DL) pour détecter et compter des palmiers à partir d'images obtenues par un drone (UAV). leur images qu'ils ont acquises ont d'abord été recadrées et échantillonnées en sous-images de petite taille qui ont été divisées en un ensemble d'entraînement, un ensemble de validation et un ensemble de test. ils ont utilisé un algorithme basé sur le Faster-RCNN (Figure III.1) pour construire le modèle et extraire les caractéristiques des images et identifie les palmiers à huile.

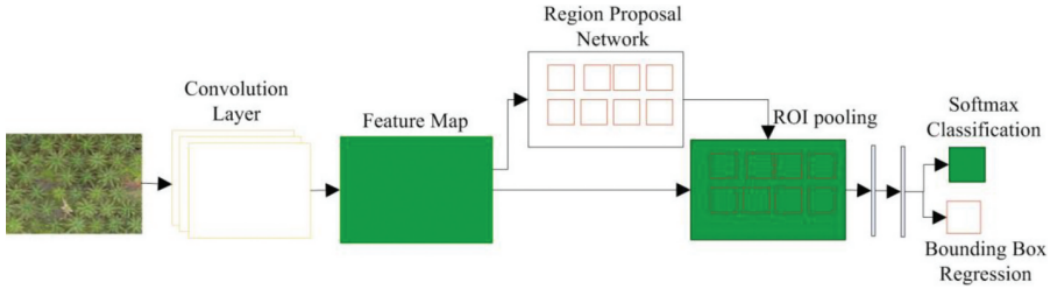


Fig. III.1 : Le processus d'entraînement de leur modèle (Faster RCNN) qu'elles ont utilisé [69]

Le modèle a ensuite été entraîné et utilisé pour détecter les palmiers à huile individuels sur l'ensemble des images de test. La précision globale de la détection des palmiers à huile a été mesurée sur trois sites différents avec 97,06 %, 96,58 % et 97,79 % de détection correcte des palmiers. Cette étude propose une nouvelle approche pour détecter les palmiers à huile à partir d'images de drones, ils ont utilisé Faster RCNN et une méthode de prétraitement d'image pour détecter les palmiers à huile à partir d'images de drones de la région de l'océan Indien. Les performances de la méthode proposée ont été comparées à celles des méthodes traditionnelles d'apprentissage automatique ANN et SVM sur trois sites différents. La méthode proposée est plus performante que les deux autres méthodes d'apprentissage automatique en termes de précision et d'exactitude globale. La détection des objets a échoué lorsque la taille des palmiers à huile dans les images était trop petite, il y a un chevauchement de l'arrière-plan et une faible résolution de l'objet. Néanmoins, sur la base du modèle pré-entraîné dans cet article, il faut 1,5 heure pour détecter et compter les 22 hectares de plantation de palmiers à huile dans cette étude, les résultats sont prometteurs, ce qui suggère qu'il a le potentiel d'être utilisé dans des applications pratiques.

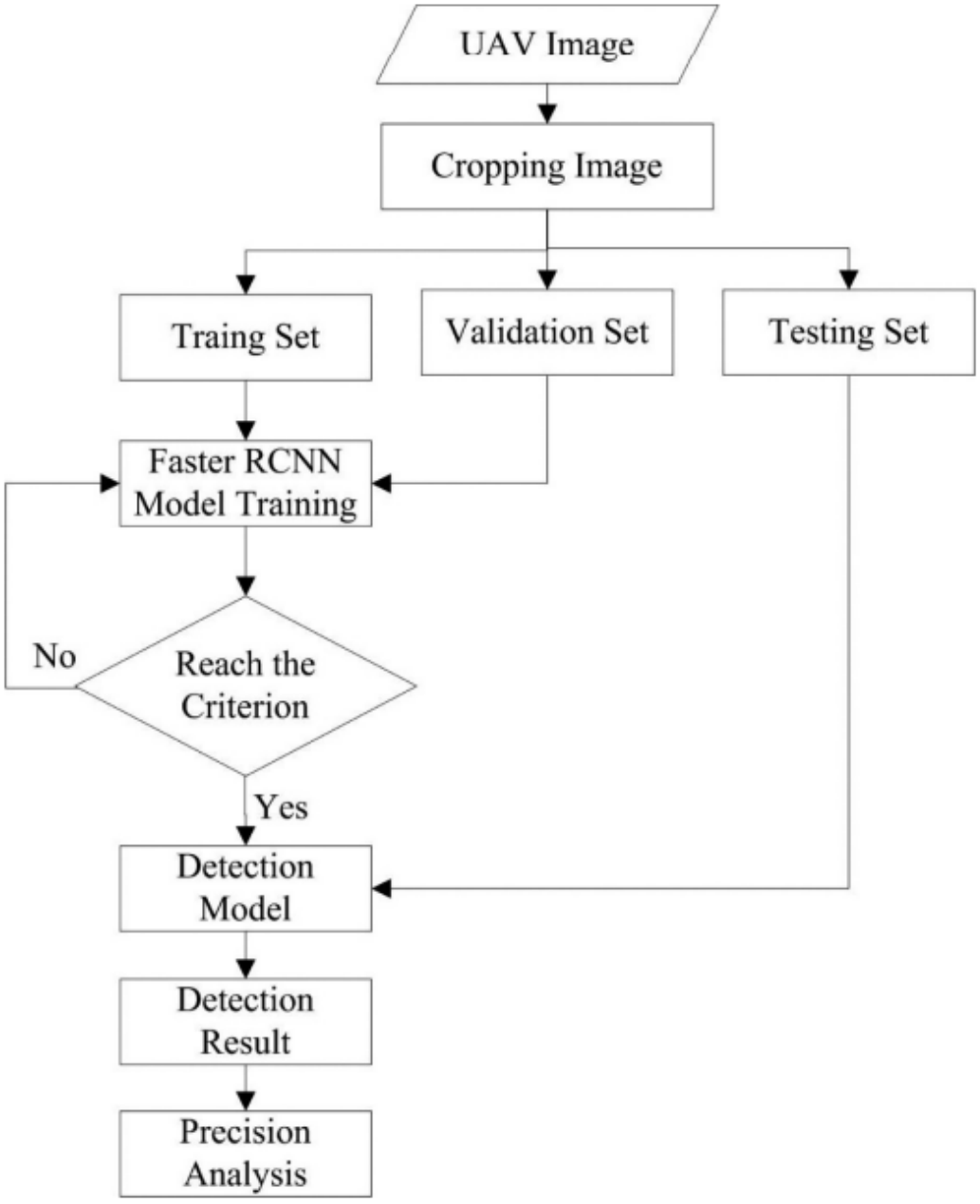


Fig. III.2 : L'organigramme de leur méthode proposée [69]

Dans un autre article [70], ils ont proposé un cadre basé sur l'apprentissage profond pour la détection et le comptage de palmiers à huile et ils utilisant des images de télédétection à haute résolution pour la Malaisie, ils ont utilisé un certain nombre d'échantillons interprétés manuellement pour entraîner et optimiser le réseau de neurones convolutifs (CNN), et prédire les étiquettes pour tous les échantillons dans un ensemble de données d'images collectées par la technique de la fenêtre coulissante (sliding window). Ensuite, ils ont fusionné les coordonnées prédites de la palme correspondant au même palmier en une seule coordonnée et ils ont obtenu les résultats finaux de la détection du palmier, grâce à la méthode qu'ils ont proposé, plus de 96 % des palmiers à huile de leurs zone d'étude peuvent être détectés correctement par rapport à la vérité du terrain interprétée manuellement, la différence entre le nombre prévu de palmiers et le nombre réel de palmiers est

inférieur à 4 % pour chaque région.

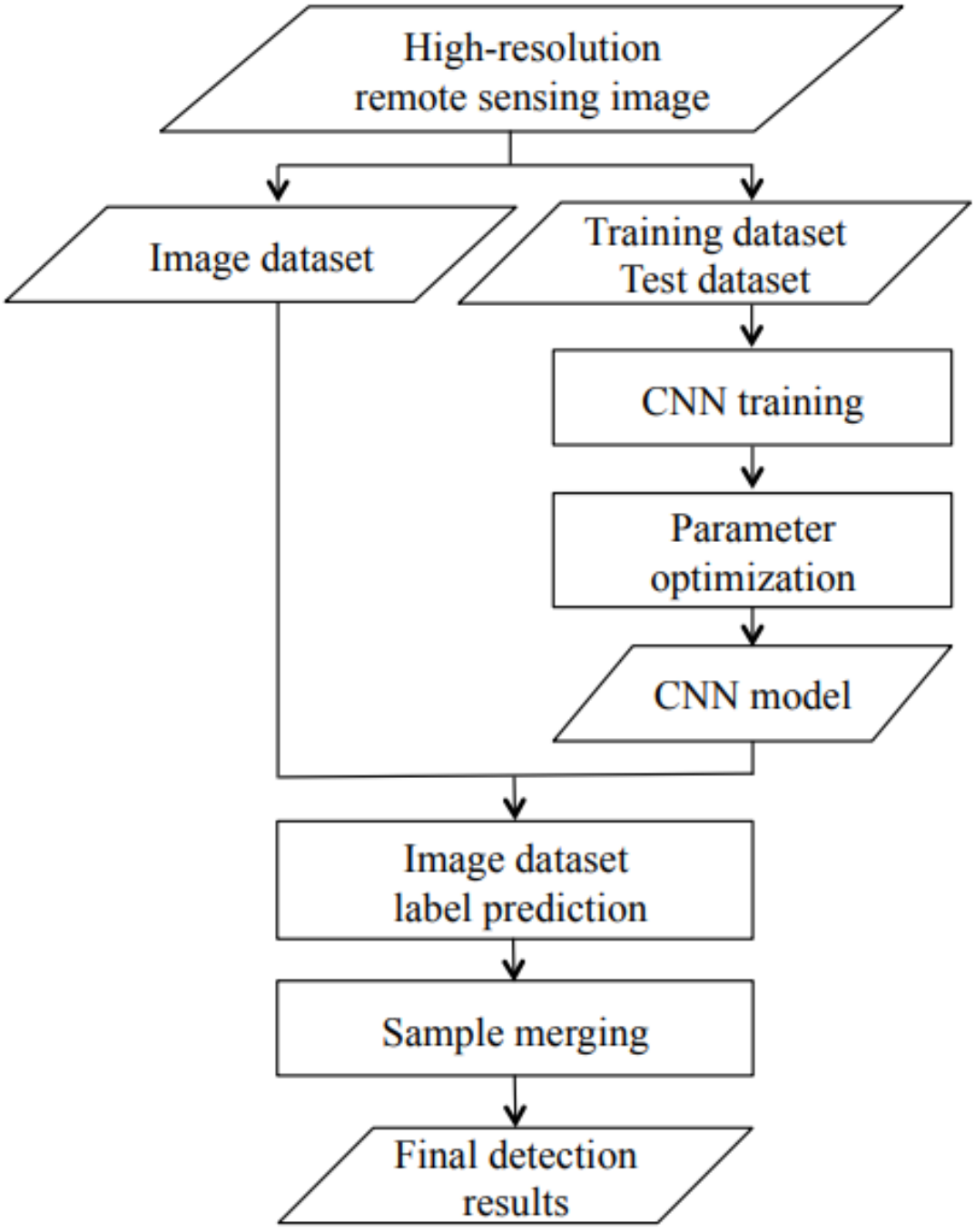


Fig. III.3 : L'architecture de leur méthode proposée [70]

Nous avons aussi dans d'autre travaux [71], ils proposons un framework original de deep learning pour le comptage et la géolocalisation automatique de palmiers à partir d'images aériennes, ils utilisant des réseaux de neurones convolutifs, pour cela ils ont collecté des images aériennes de deux régions différentes d'Arabie Saoudite, ils utilisant deux drones DJI et ils avons construit un ensemble de données d'environ 11 000 exemples de palmiers. Ensuite, ils avons appliqué plusieurs modèles récents de réseaux de neurones convolutifs

(Faster R-CNN, YOLOv3, YOLOv4 et EfficientDet), pour détecter les palmiers et autres arbres, et ils avons effectué une évaluation comparative complète en termes de de précision moyenne et de vitesse d'inférence, YOLOv4 et EfficientDet-D5 ont donné le meilleur compromis entre précision et vitesse (jusqu'à 99 % de précision moyenne et 7,4 FPS) figure(III.4).

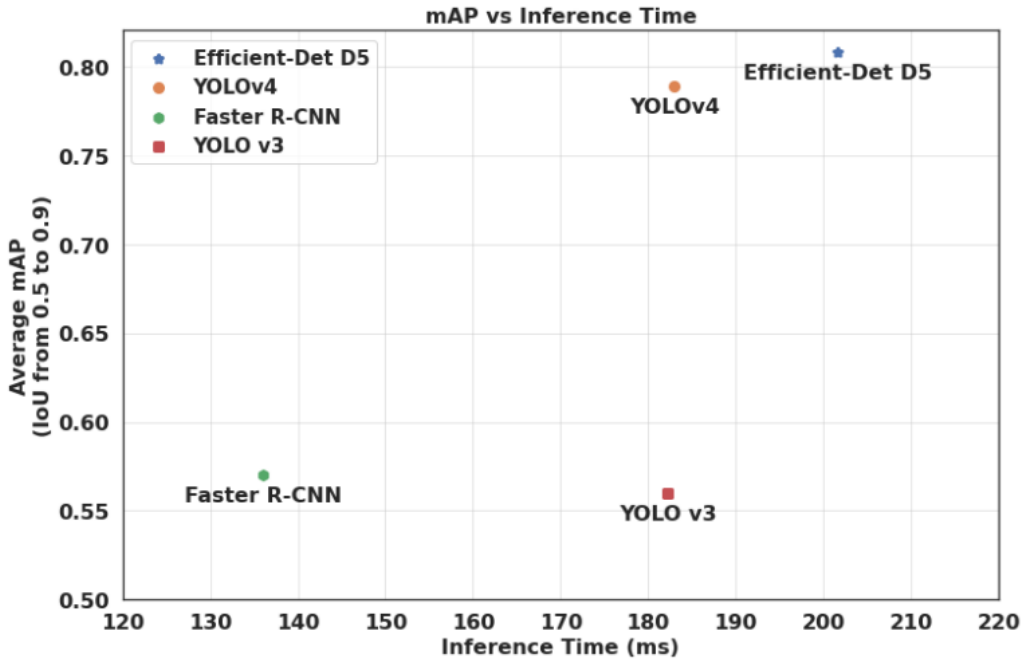


Fig. III.4 : Leur comparaison des quatre algorithmes en termes de mAP (moyenne pour des valeurs d'IoU entre 0,5 et 0,9 par pas de 0,1) et le temps d'inférence [71]

En plus, ils ont utilisé les métadonnées géolocalisées des images aériennes, ils avons utilisé les concepts de la photogrammétrie et les corrections de distance pour détecter automatiquement l'emplacement géographique des palmiers détectés. Cette technique de géolocalisation a été testée sur deux types de drones différents (DJI Mavic Pro et Phantom 4 pro) et a été évaluée comme fournissant une précision de géolocalisation moyenne de 1,6 m. Ce marquage GPS va leur permet d'identifier les palmiers de manière unique et de compter leur nombre à partir d'une série d'images de drones, ainsi leur géolocalisation peut être généralisée à tout autre objet dans les images de drones.

Enfin nous présentons ici cet article [72], ils proposons une méthode (Figure III.5) permettant de combiner plusieurs méthodes de traitement d'images avec un réseau neuronal convolutif (CNN) pour effectuer la détection et le comptage des palmiers. Cet article se concentre sur l'imagerie par drone, qui possède une haute résolution d'image et est largement déployée dans l'industrie des plantations. L'analyse des images de drones est un défi en raison des altitudes de vol différentes des drones, ce qui entraîne des différences de taille des arbres sur les images capturées. Le comptage par correspondance de modèles ou par la méthode de la taille fixe de la fenêtre coulissante produit souvent un comptage inexact. Au lieu de cela, leur méthode utilise l'analyse du domaine de fréquence pour estimer la taille des arbres avant le CNN. La méthode est testée sur deux images, de quelques milliers à quelques centaines de milliers d'arbres par image. L'expérience est réalisée avec l'image de drone, ils illustrant que le résultat est avec un score F1 de 98,6 % et 98,54 %.

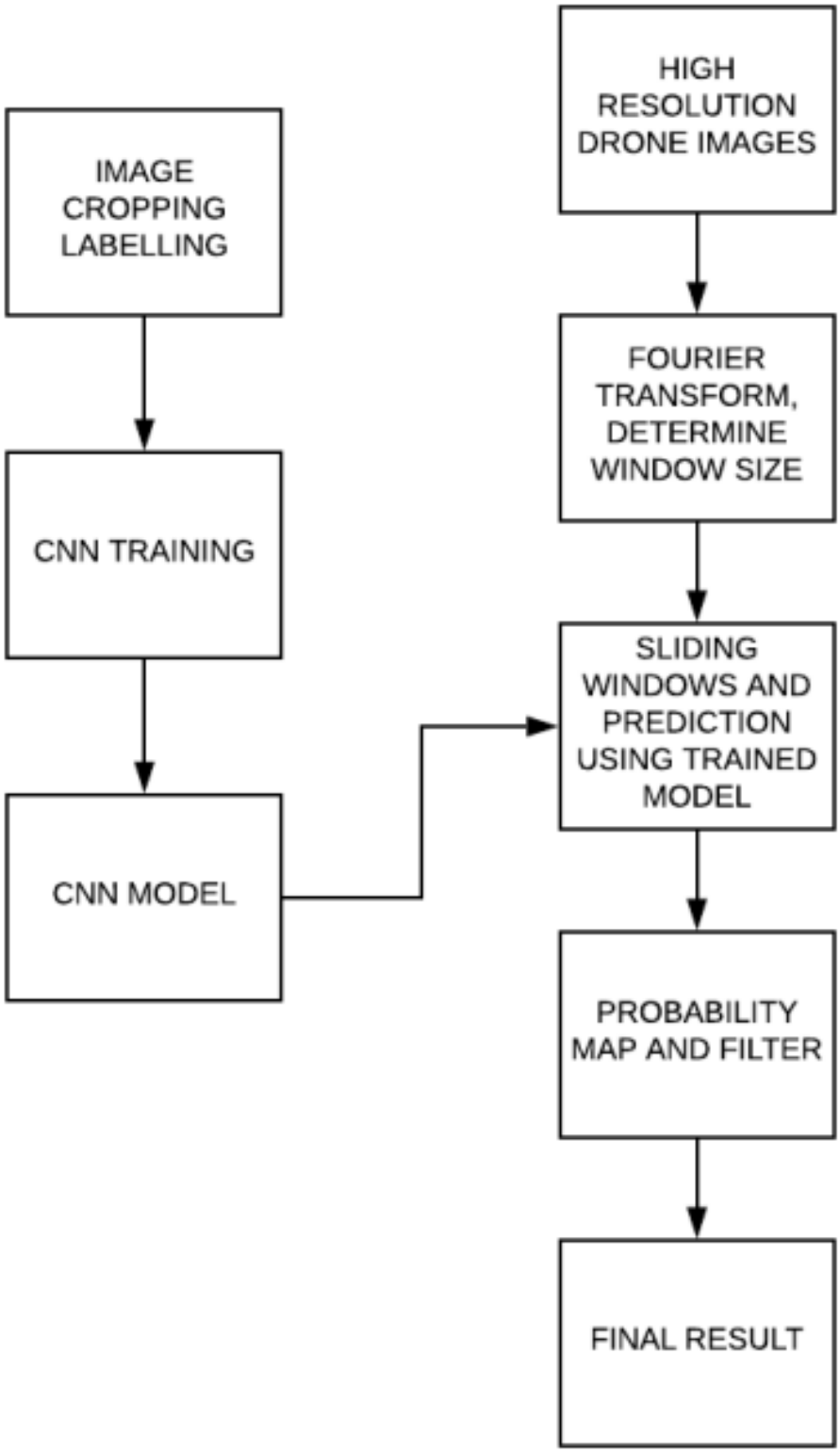


Fig. III.5 : Architecture de leur méthode proposée [72]

Chapitre IV

Implémentation

1 Introduction

Dans ce chapitre, nous allons expliquer la méthodologie mise en place pour la création d'un système de détection d'objet dans des images et des vidéos capturés depuis des drones.

Nous avons créé une méthode basé sur le modèle YOLOv5 que nous avons modifié selon nos besoins, en effet nous avons vu dans les chapitres président que la détection d'objet dans un flux vidéo pour les applications de temps réel nécessite un réseau de neurones plus profond.

Notre idée consiste à utiliser un transfère de connaissances en se basant sur la dernière architecture de YOLO v5 en modifiant une portion de l'architecture dont la tache est l'extraction de caractéristiques afin de permette à notre modèle de se focaliser sur les caractéristiques les plus pertinentes. Afin de prouver l'efficacité de notre modèle, notre travail se focalise sur la détection d'un objet spécifique palmier dans forêt composée de millier d'arbre. Dans ce chapitre nous allons commencer par décrire l'architecture de notre modèle ainsi la création et l'intégration de l'objet palmier dans notre jeu de données afin d'entraîner et de valider notre modèle.

2 La conception et la mise en œuvre

Dans cette section nous allons présenter les différentes étapes composants notre modèle (voir la figure IV.1) à savoir le pré-traitement de données, l'entraînement et la détection. En effet nous allons expliquer chaque étape en détail.

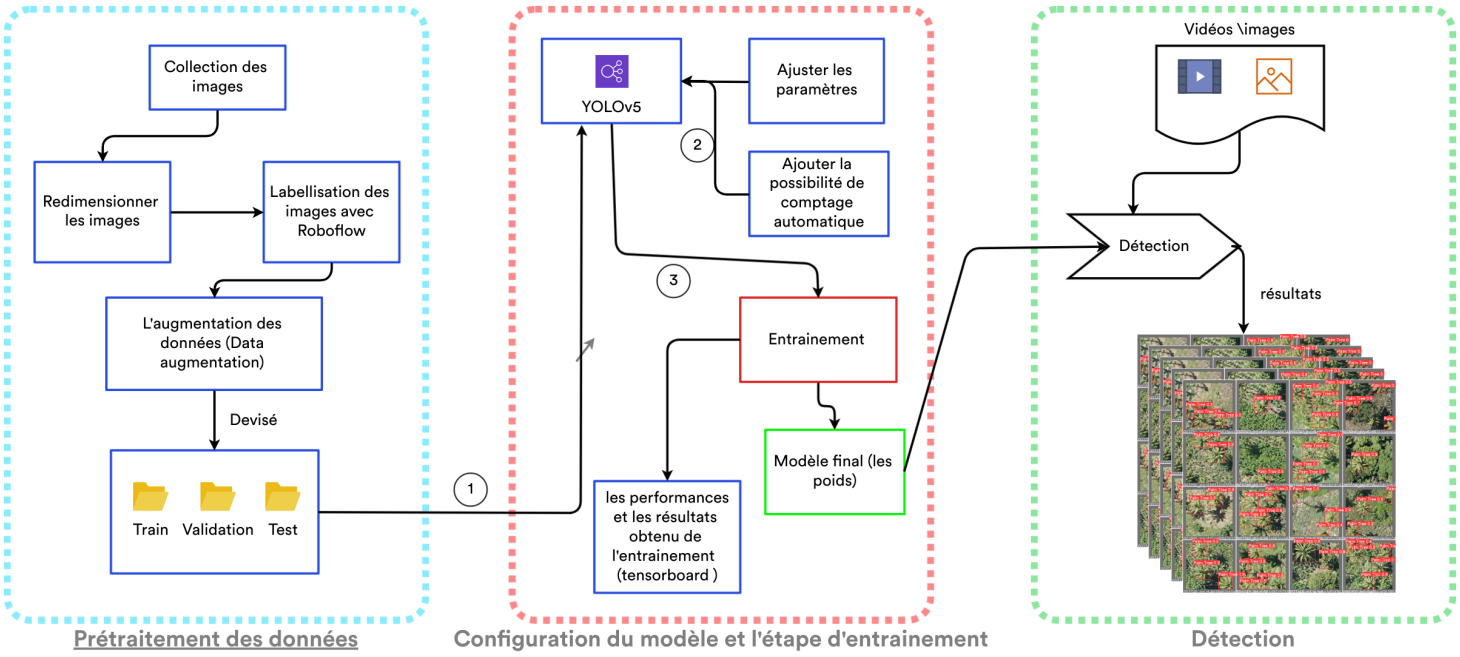


FIG. IV.1 : L'architecture des différentes étapes de notre travail

2.1 Constitution de jeu de données (dataset) d'entrainement

Construire un ensemble d'images ou de jeux de données n'est pas facile, en effet la qualité de jeu de données affectera directement les résultats, alors plusieurs centaines d'images sont nécessaires pour l'entraînement, de toute façon nous devons collecter et étiqueter manuellement des images avec des nombreuses caractéristiques.

2.1.1 Collection des images

Tout d'abord, nous avons recueilli depuis le web des images de palmiers prises avec drone environ 1993 images, puis les redimensionner en (224x224), notre entraînement ne portera donc que sur une seule classe.

2.1.2 Annotation du jeu de données

Après l'étape de collecte de données, il va falloir préciser la position de l'ensemble des pixels présentant l'objet à détecter dans notre cas, c'est ce que l'on appelle l'étape de labellisation, malheureusement ce n'est pas l'étape la plus sympathique car elle est longue et rébarbative.

Pour chaque image le but est très simple, il faut identifier et annoter la zone d'image contenant un ou plusieurs palmiers, il faut faire ce travail sur chaque photo et manuellement, il existe de nombreux outils disponibles que nous pouvons utiliser pour annoter les images gratuitement, l'un de c'est outil est Roboflow [73], mais avant de commencer à étiqueter les données, nous devons comprendre qu'il existe différents formats des boîtes englobantes, Roboflow utilise les formats d'annotation les plus courants, notamment JSON, XML, CSV et TXT.

La Figure IV.2 montre l'une de nos images étiquetées avec l'outil Roboflow

Fig. IV.2 : Exemple de labellisation de nos images avec Roboflow

Chaque modèle de détection utilise sa propre représentation des coordonnées, YOLOv5 dans PyTorch lit les données des boîtes englobantes dans des fichiers texte (.txt) dans lequel sont stockées les informations relatives aux étiquettes de l'image.

Ainsi, les données de toutes les boîtes englobantes d'une image doivent être regroupées et écrites dans un fichier texte qui correspond à cette image (chaque image a un fichier

texte), dans le fichier texte le nombre de lignes indique le nombre d'objets présents dans une image, et chaque ligne comporte cinq paramètres :

- L'index de la classe d'objets
- Coordonnées (x,y) qui représentent le centre de la boîte englobante
- Largeur de la boîte englobante
- Hauteur de la boîte englobante.

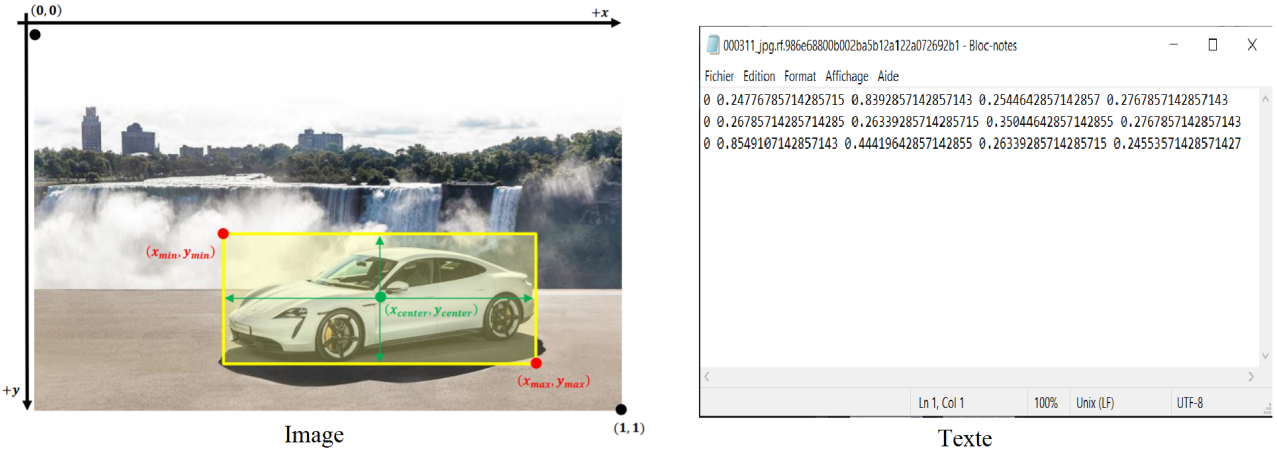


Image Texte

Fig. IV.3 : La représentation des boîtes englobantes dans une format de YOLO

L'étape suivant est de diviser l'ensemble de données en trois parties, 1590 images (train) pour la phase d'entraînement, 291 images (valid) pour mesurer la performance de l'entraînement , et 112 images (test) pour tester la détection sur des images, ces fichiers sont générés avec Roboflow [73] en format 'YOLOv5 pytorch' dans un fichier 'zip' qui contient aussi un fichier 'data.yaml'

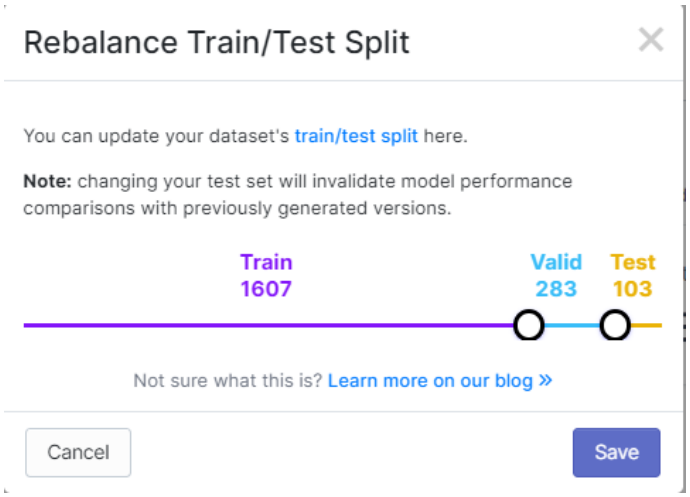


Fig. IV.4 : La division des données étiqueter avec Roboflow [73]

Le fichier "data.yaml" c'est un fichier de configuration des données qui spécifier :

- Où se trouvent nos données de formation et de validation.
- Le nombre des classes qu'on veut détecter.
- Et les noms qui correspondant à ces classes.

Notre propre fichier "data.yaml" contient une seule classe "Palm Tree" comme montre la figure (IV.5)

```
yolov5 > data > ! data.yaml
1   train: ../yolov5/DataSet/Data_Aug/train/images
2   val: ../yolov5/DataSet/Data_Aug/valid/images
3
4   nc: 1
5   names: ['palm tree']
```

Fig. IV.5 : Fichier data.yaml utilisé

2.1.3 L'augmentation des données

L'augmentation des données est une technique très importante pour générer des données plus utiles (augmenter le volume de données et préserve les étiquettes) (Figure IV.6) à partir de données existantes pour l'entraînement de CNNs pratiques et généraux, les méthodes existantes d'augmentation des données peuvent être grossièrement divisées en trois catégories : Transformation spatiale, la distorsion des couleurs, l'élimination des informations.[74] [75] :

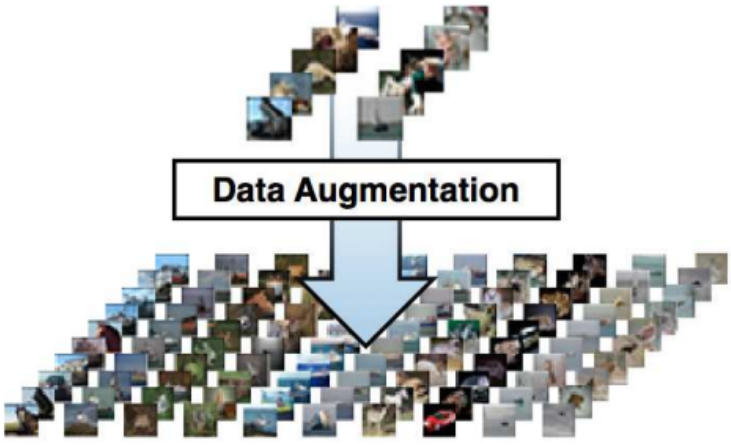


Fig. IV.6 : L'augmentation de données [75]

Dans notre cas, nous avons utilisé la transformation partielle et la distorsion des couleurs : rotation, contraste, luminosité, contours, netteté, flou ...

Après l'augmentation des données, notre jeu de données augmente avec un pourcentage de 160 % (environ 3200 image de plus) comme montre la figure (IV.7) :

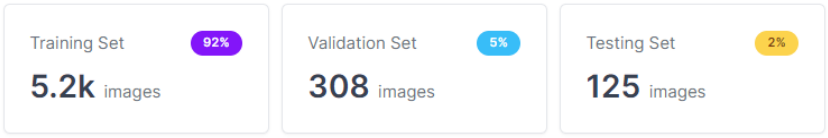


FIG. IV.7 : Le jeu de données après l'augmentation des données

l'outil utilisé pour cette augmentation est Roboflow [73] (Figure IV.8) :

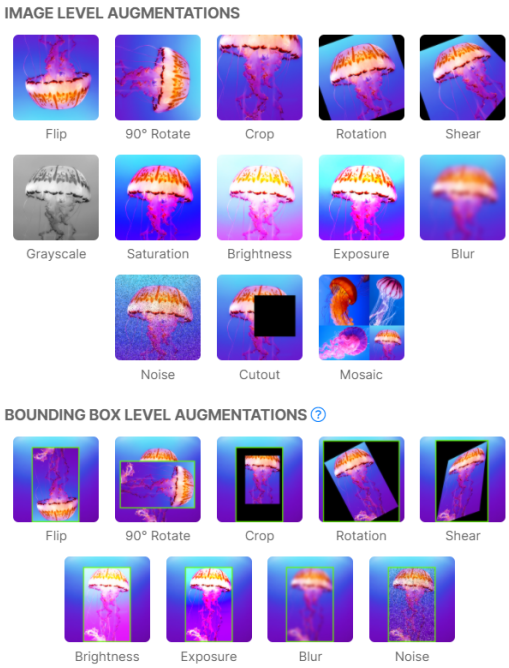


FIG. IV.8 : Augmentations des donnnéés avec Roboflow [73]

2.2 Entraînement et évaluation du modèle de détection

2.2.1 Architecture du modèle YOLO v5

Le modèle est constitué de :

- **Backbone** : CSPDarknet53 (Figure IV.10) le réseau convolutif le plus avancé à cette époque, CSPDarknet53 du modèle YOLOv5 a été choisie parmi 3 options : CSPResNext53, CSPDarknet53 et EfficientNet-B3, ainsi sur la base d'une justification théorique et de nombreuses expériences, le réseau neuronal CSP Darknet53 a été déterminé comme étant le modèle le plus optimal. [76]

Backbone model	Input network resolution	Receptive field size	Parameters	Average size of layer output (WxHxC)	BFLOPs (512x512 network resolution)	FPS (GPU RTX 2070)
CSPResNext50	512x512	425x425	20.6 M	**1058 K**	31 (15.5 FMA)	62
CSPDarknet53	512x512	725x725	**27.6 M**	950 K	52 (26.0 FMA)	**66**
EfficientNet-B3 (ours)	512x512	**1311x1311**	12.0 M	668 K	11 (5.5 FMA)	26

FIG. IV.9 : Tableau comparatif des 3 réseaux de Backbone [76]

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

FIG. IV.10 : Architecture de darknet53 [46]

CSPDarknet53 utilise la stratégie CSPNet [49] pour diviser la carte des caractéristiques de la couche de base en deux parties, puis les fusionner selon une hiérarchie horizontale. L'utilisation de cette stratégie permettra un flux de gradient plus important à travers le réseau. [49]

- **Neck** : PAnet, l'image d'entrée après avoir été transmise par le backbone, les caractéristiques de l'image sont transformées en caractéristiques sémantiques (ou caractéristiques apprises), en d'autres termes, depuis les couches de bas niveau, plus l'image d'entrée est traversée, la complexité des caractéristiques sémantiques augmentera tandis que la résolution spatiale des cartes de caractéristiques diminuera en raison du sous-échantillonnage. Cela conduit à une perte d'informations spatiales ainsi que de caractéristiques à grain fin, a fin de préserver ces caractéristiques à grain fin, Joseph Redmon a appliqué l'idée de l'architecture Feature Pyramid Network (FPN)

[57]

L'architecture FPN met en œuvre une démarche descendante pour transférer les caractéristiques sémantiques (de la couche de haut niveau), puis de les concaténer à des caractéristiques à grain fin (de la couche de bas niveau dans le Backbone) pour prédire les petits objets dans le détecteur à grande échelle.

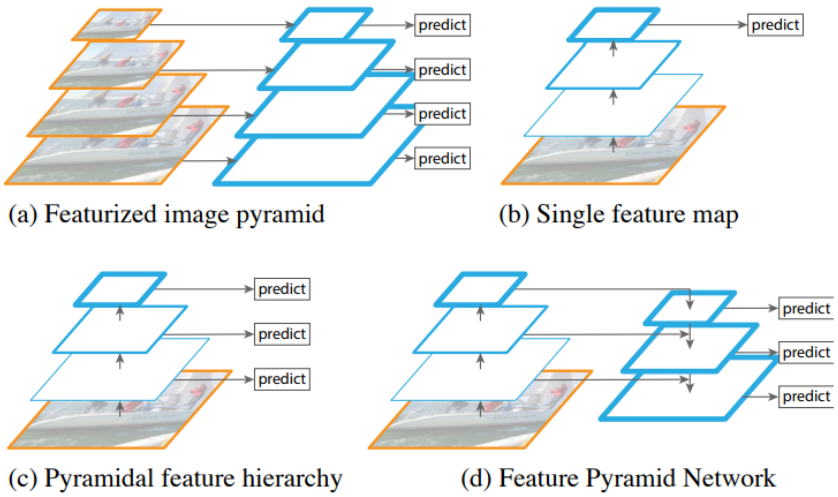


FIG. IV.11 : FPN architecture [57]

Le Path Aggregation Network (PAN) est une version avancée du FPN qui fonctionne de la même manière que FPN, mais ils ajoutent un chemin ascendant augmenté, comme illustré à la Figure (IV.12), afin que les réponses de texture fortes à partir de bas niveaux puissent être directement fusionnées avec réponses sémantiquement riches présentes dans N5 en utilisant un chemin de raccourci.

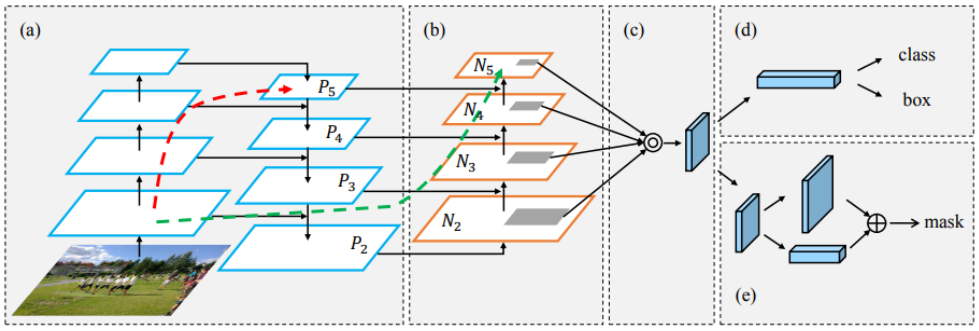


FIG. IV.12 : Path Aggregation Network (PAN) Architecture [51]

- **Head** : YOLO, dans le cas d'un détecteur à un stade, la fonction de la tête est d'effectuer une prédiction finale composée d'un vecteur contenant les coordonnées prédites des boîtes englobantes (centre, hauteur, largeur), le score de confiance de la prédiction et les classes de probabilité

Dans le schéma suivant (figure IV.13), nous allons détailler plus l'architecture de Yolo v5 :

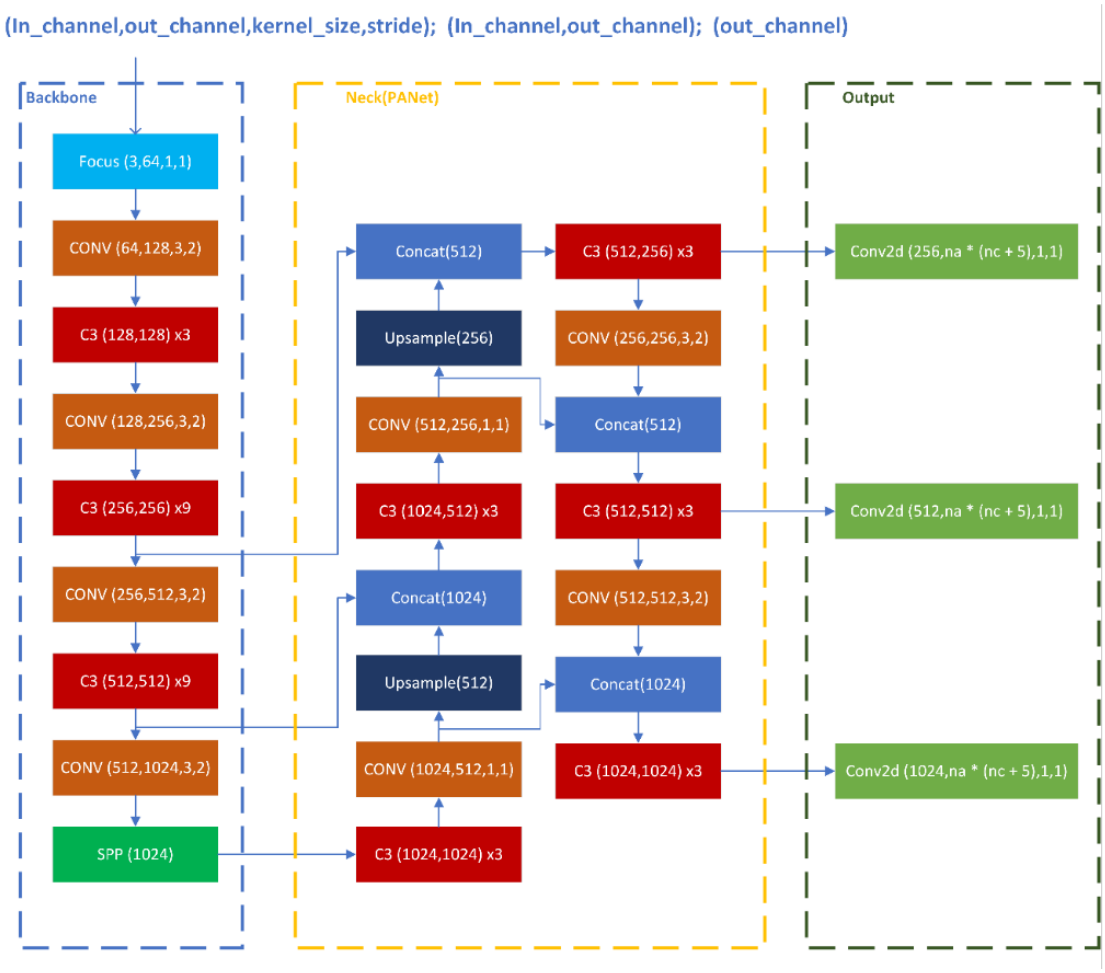


FIG. IV.13 : YOLOv5s network structure

La Couche Focus : cette couche réduit les calculs du modèle et accélère la vitesse de l'entraînement, ses fonctions sont les suivantes : premièrement, l'image d'entrée à 3 canaux (la taille de l'image d'entrée par défaut d'entrée de l'architecture est de 3 × 640 × 640) a été segmentée en quatre tranches de 3 × 320 × 320 par tranche, à l'aide d'un algorithme de segmentation, ensuite, l'opération Concat va relier les quatre sections en profondeur, la taille de la carte des caractéristiques de sortie étant de 12 × 320 × 320, Puis passe à travers la couche convolutive composée de 32 noyaux de convolution, la carte de caractéristiques de sortie de taille 32 × 320 × 320 a été générée. Enfin, à travers la couche BN (Batch Normalization) et les fonctions d'activation de SiLU, les résultats sont transmis à la couche suivante.

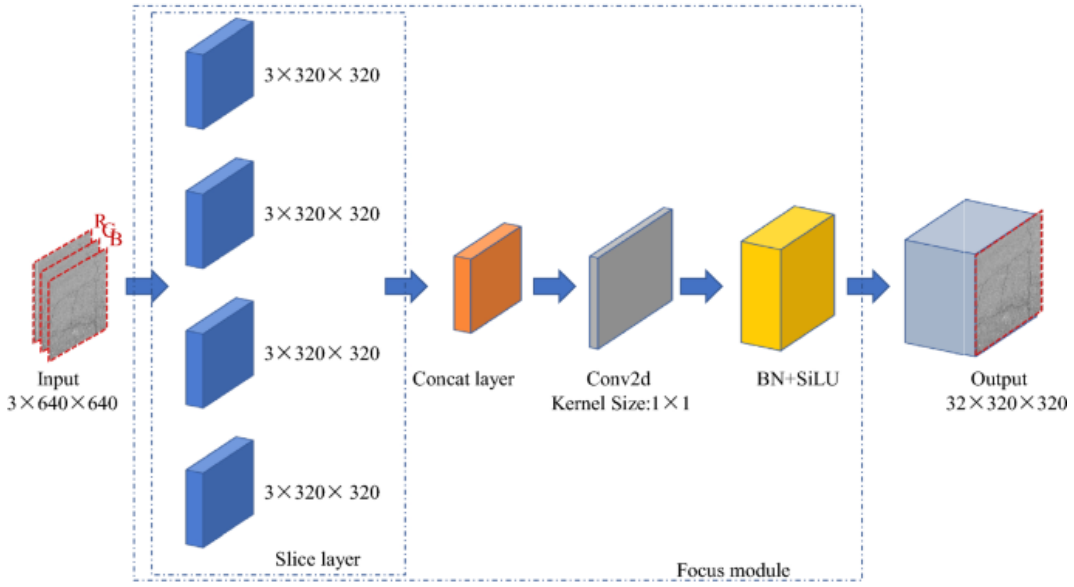


FIG. IV.14 : Structure du module Focus

Conv : cette couche ce compose de

- Couche convolutive standard
- Batch-normalization (BN) : cette fonction est utilisée pour normaliser et mettre à l'échelle les entrées de la couche précédente, c'est-à-dire appliquer une transformation qui maintient l'activation moyenne proche de 0 et l'écart type d'activation proche de 1
- La fonction d'activation SiLU : la fonction Sigmoid Linear Unit (SiLU) est une amélioration de ReLU car elle permet d'avoir des valeurs lisses lorsque x $<$ 0 , une moyenne plus proches de zéro permet un apprentissage plus rapide car cela rapprochent le gradient calculé du gradient naturel [77], il existe d'autre fonctions d'activation comme (Relu , Sigmoïde , dSiLU ...)

C3: La couche du réseau yolov5 est le module BottleneckCSP (Figure IV.16 (A)), mais dans notre cas on a utilisé la couche C3 (Figure IV.16 (B)) qui est conçu pour mieux extraire les caractéristiques profondes de l'image. Le module C3 est principalement composé d'un module Bottleneck (Figure IV.15 et IV.16 (B)), qui est une architecture de réseau résiduelle reliant une couche convolutive standard (Conv2d + BN + fonction d'activation SiLU) dont la taille du noyau de convolution est de 1 × 1, à une couche convolutive dont la taille du noyau de convolution est de 3 × 3, la sortie finale du module Bottleneck est l'addition de la sortie de cette partie et de l'entrée initiale.

L'entrée initiale du module C3 (Figure IV.16 (B)) est introduite dans deux branches, et le nombre de canaux des cartes de caractéristiques est divisé dans deux couches convolutive, la sortie de la premier couche convolutive passe sur la couche Bottleneck, ensuite après la sortie des cartes de caractéristiques du Bottleneck l'opération concat va relier les deux sections en profondeur. Enfin, la carte des caractéristiques de sortie du module est obtenue après avoir traversé successivement la couche de convolution (Conv2d + BN +

SiLU) successivement, et la taille de cette carte de caractéristiques est la même que celle de l'entrée du module C3.

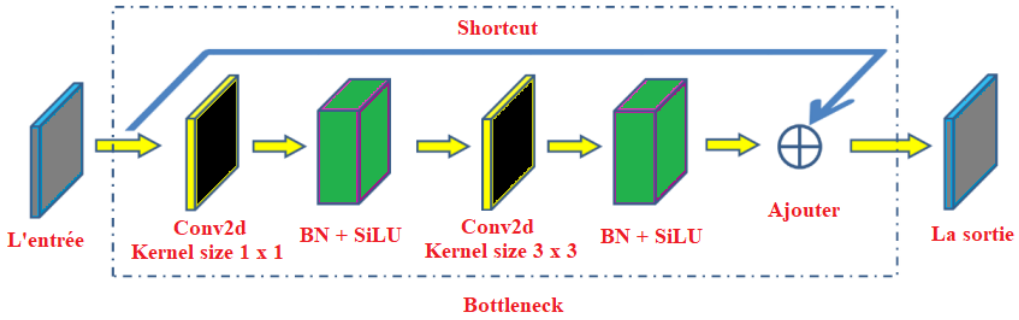


FIG. IV.15 : Bottleneck réseau structure

La différence entre le module C3 et le module BottleneckCSP est que le module Conv après la sortie résiduelle est supprimé, et que la fonction d'activation dans le module de convolution standard après concat est également modifiée de LeakyRelu à SiLU, la figure (IV.16) montre la différence entre la couche BottleneckCSP que nous avons remplacée et la couche C3 que nous avons mis.

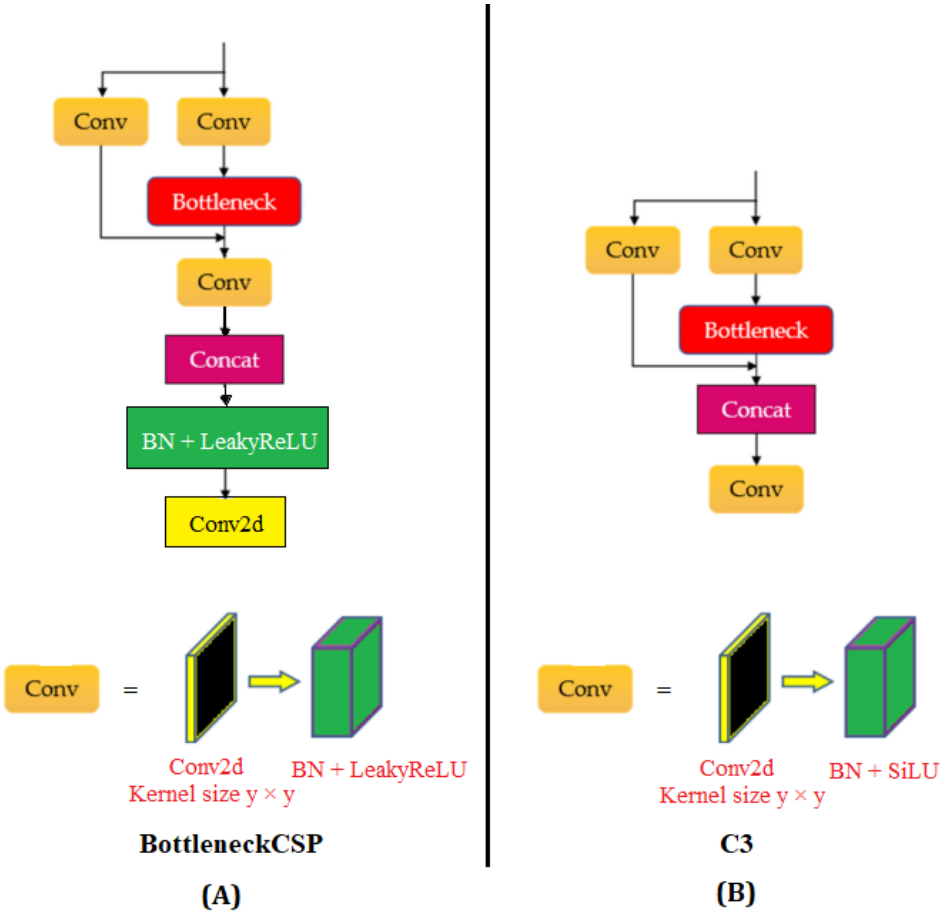


FIG. IV.16 : La différence entre la couche C3 et BottleneckCSP

SPP : La neuvième couche du réseau Backbone est le module SPP (spatial pyramid

pooling figure IV.17), qui est conçu pour améliorer le champ réceptif du réseau pour le passer dans le Neck, la taille de la carte de caractéristiques d'entrée du module SPP qui fait partie de YOLOv5s est 1024 × 20 × 20. Tout d'abord, la carte de caractéristiques d'une taille de 512 × 20 × 20 est produite après un passage dans la couche convolutive dont la taille du noyau de convolution est de 1 × 1. Ensuite, cette carte de caractéristiques et la carte de caractéristiques de sortie passe dans trois couches parallèles de Maxpooling (couche de mise en commun maximale) sont connectés en profondeur, et la taille de la carte de caractéristiques de sortie est de 2048 × 20 × 20. Enfin, la carte finale des caractéristiques de sortie, d'une taille de 1024 × 20 × 20, est obtenue après un passage dans la couche convolutive.

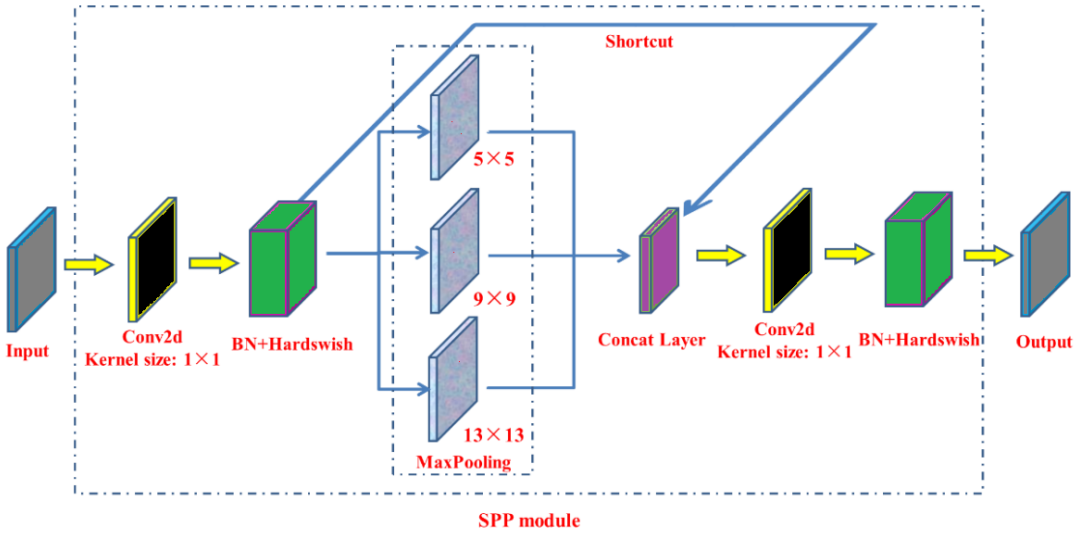


Fig. IV.17 : L'architecture de SPP

2.2.2 Ajouter la possibilité de comptage automatique dans notre système de détection

On a ajouté le comptage automatique dans le système avec les étapes suivantes :

- Extraire les informations générées dans l'étape de détection, parmi ces informations nous avons le nombre des boîtes englobantes prédites (le nombre des palmiers détecté) en temps réel, ensuite nous précisons quelle information nous volons extraire (le nombre des boites englobantes prédites) et nous avons affecté le nombre dans une variable (l).

```
    'image 1/2 C:\\Detection Project\\yolov5\\data\\images\\bus.jpg: 640x480 4 persons, 1 bus, '
l = s[1:s.find('Palm Tree')].split()[-1]
```

Fig. IV.18 : Les informations extraites de la détection dans le code source

- Ensuite il faut afficher le nombre des palmiers détecté en temps réel dans l'image ou la vidéo qui a été passé dans le réseau, ainsi cela et réalisé avec une bibliothèque qui s'appelle Open-Cv, cette bibliothèque graphique est spécialisée dans le traitement d'images, que ce soit pour de la photo ou de la vidéo elle va nous aider à afficher le nombre des palmiers comptés dans un temps réel.

```
def plot_one_box(img, color= None, label=None):
    # Plots one bounding box on image img
    font_cv = cv2.FONT_HERSHEY_SIMPLEX
    cord = (0,80)
    form = cv2.LINE_AA
    if label:
        cv2.rectangle(img, (0,5), (700, 163),color= (0 , 0 , 0), thickness=-1)
        cv2.putText(img,label,cord, font_cv, 3,color,3,form)
    return img
```

FIG. IV.19 : Code source de la fonction qui utilise Open-cv pour afficher le nombre prédit dans l'image ou la vidéo en temps réel

- L'étape final consiste a enregistrer les résultats des prédictions de la détection et le nombre de palmier détecté dans l'image ou la vidéo que nous avons passé dans le modèle, donc nous avons réalisé cette tache avec le code source suivant :

```
# Save results (image with detections)
if save_img:
    if 'Palm Tree' in s:
        im0 = plot_one_box(im0,(255,255,255),str(l) + " Palm Tree" )

    if dataset.mode == 'image':
        cv2.imwrite(save_path, im0)
    else:  # 'video' or 'stream'
        if vid_path[i] != save_path:  # new video
            vid_path[i] = save_path
            if isinstance(vid_writer[i], cv2.VideoWriter):
                vid_writer[i].release()  # release previous video writer
            if vid_cap:  # video
                fps = vid_cap.get(cv2.CAP_PROP_FPS)
                w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
                h = int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
            else:  # stream
                fps, w, h = 30, im0.shape[1], im0.shape[0]
            save_path = str(Path(save_path).with_suffix('.mp4'))  # force *.mp4 suffix on results videos
            vid_writer[i] = cv2.VideoWriter(save_path, cv2.VideoWriter_fourcc(*'mp4v'), fps, (w, h))
        vid_writer[i].write(im0)
```

FIG. IV.20 : le code source pour afficher les résultats final dans l'image ou la vidéo

2.2.3 Ajuster les paramètres pour entraîner le modèle (Yolo v5)

Les paramètres et leurs valeurs sont présentés dans le tableau (IV.2) et sont expliqués ci-dessous :

Paramètre	Valeur par défaut	La nouvelle valeur	Description
Batches (Batch size)	1	64	Le Batch-size définit le nombre d'échantillons qui seront propagés sur le réseau
Epochs	100	200	Le nombre d'époques est un hyperparamètre qui définit le nombre de fois que l'algorithme d'apprentissage va travailler sur l'ensemble des données d'apprentissage
Cfg	Yolov5x	Yolov5s	Modèle d'architecture, 4 choix sont disponibles : yolo5s.yaml, yolov5m.yaml, yolov5l.yaml, yolov5x.yaml, on peut personnalisée ces fichiers
Workers	8 (CPU)	1 (GPU)	Workers est le nombre de cœurs de processeur utilisés. Si vous entraînez Multi-GPU, il s'agit du nombre de cœurs de processeur utilisés par GPU
Img	640	224	Le paramètre Img représente la taille d'entrer des images qui seront propagés sur le réseau
Hyperparameter	hyp.scratch.yaml	hyp.scratch.yaml	Les hyperparamètres sont des paramètres dont les valeurs contrôlent le processus d'apprentissage et déterminent les valeurs des paramètres du modèle pour augmenter ses performances
Nombre des classes	80	1	Comme son nom, ce paramètre représente le nombre des classes qui nous volons détecter

Tab. IV.2 : Modification des paramètres pour l'entraînement

3 Conclusion

Dans ce chapitre, nous avons présenté notre travail avec un schéma explicatif qui a consisté en un premier temps à la création de jeu de données (Dataset). Cela nécessité un grand effort afin d'annoter les échantillons de notre dataset. Nous avons aussi généré les labels pour notre modèle (txt pour YOLO v5). Après avoir augmenté la taille de notre jeu de données avec la technique de Data-augmentation, nous avons expliqué le modèle et ajusté ses paramètres, ensuite nous avons présenté un algorithme qui permet d'effectuer un comptage des objets palmiers détecter.

Dans le chapitre suivant nous allons exposer d'évaluer les résultats obtenus et de mesurer les performances de notre modèle prédictif.

Chapitre V

Résultats et Validation

1 Introduction

Dans le chapitre précédent, nous avions présenté la méthodologie utilisée pour atteindre nos objectifs. Dans celui-ci, nous présentons les aboutissements majeurs de nos travaux, c'est-à-dire les résultats du modèle entraîné pour la détection des arbres de palmiers avec des vidéo ou des images capturés avec drone.

2 Le matériel utilisé

Afin de réaliser l'apprentissage de ce projet, nous avons utilisé notre matériel (laptop) dont les principales caractéristiques sont les suivantes :

- Système d'exploitation : Windows 10 Famille.
- RAM 8 GO.
- Processeur (CPU) intel Core(TM) i7 7-7700HQ 2.80 GHz
- Carte graphique (GPU) NVIDIA GTX 1050 DDR5 4GB

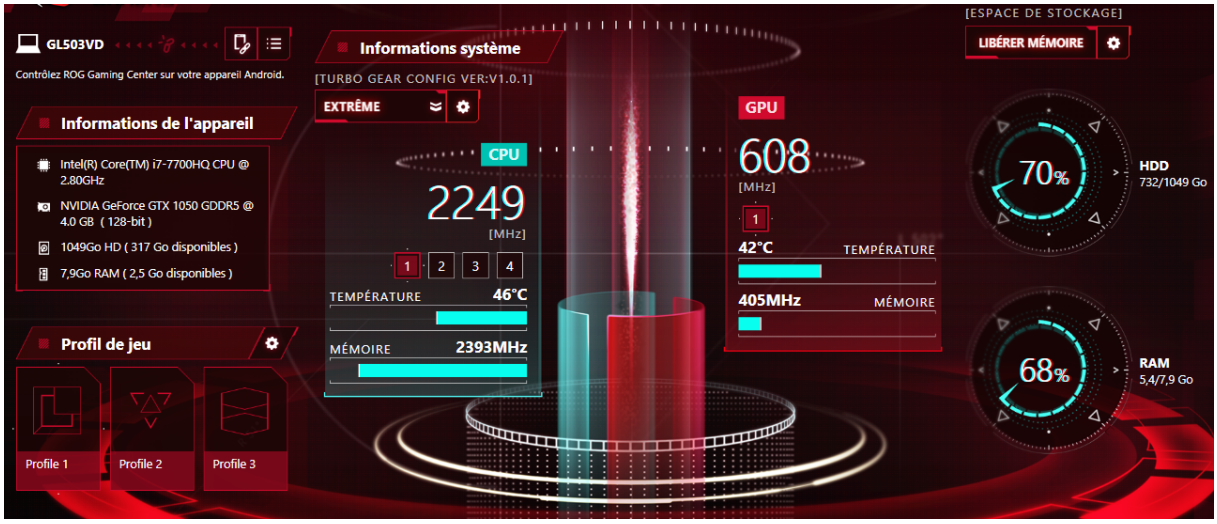


Fig. V.1 : Les informations du système utilisé

3 L'environnement de travail

3.1 Python

Python est un langage de programmation orienté objet de haut niveau, nous l'avons choisi en raison de son grand succès dans le domaine de l'intelligence artificielle, il est très rapide pour le développement des applications et cella grâce aux développeurs qui ont développé de nombreuses bibliothèques qui on facilité l'utilisation de ce langage. [78]

FIG. V.2 : Python logo [78]

3.2 Pytorch

PyTorch [79] est une bibliothèque logicielle open source d'apprentissage automatique d'apprentissage machine qui s'appuie sur Torch (en) développée par Facebook.

PyTorch permet d'effectuer les calculs tensoriels nécessaires notamment pour l'apprentissage profond (deep learning). Ces calculs sont optimisés et effectués soit par le processeur (CPU) soit, lorsque c'est possible, par un processeur graphique (GPU) supportant CUDA comme notre cas.

FIG. V.3 : Pytorch logo [79]

3.3 Cuda et cuDNN

Si on lance un entraînement avec le CPU (central processing unit) la vitesse de calcul va prendre beaucoup de temps comparons à la vitesse du GPU (Graphics Processing Unit), il serait dommage de ne profiter de l'accélération de calculs offerts par notre carte graphique, heureusement NVIDIA offre gratuitement les deux outils suivants :

- **CUDA** : ils s'agit d'une technologie propriétaire de NVIDIA qui permettant d'effectuer du traitement parallèle via leur carte graphique, pour accéder a des meilleures performances.
- **cuDNN** : : il s'agit de la bibliothèque des primitives de NVIDIA concernant les réseaux de neurones. cuDNN va permettre d'accélérer la vitesse de nos traitements , qu'ils concernant les routines standard comme la backpropagation , les fonctions d'activations etc. Il fait partie du SDK de deep learning, et peut s'utiliser avec

différents backend (Caffe, Caffe2, Chainer, Keras, Matlab, MxNet, Tensorflow et Pytorch comme notre cas). [80]

FIG. V.4 : Cuda et cuDNN logo [80]

3.4 Open-Cv

Open CV est une bibliothèque graphique gratuite qui est développé à l'origine par Intel Corporation, spécialisée dans le traitement d'images en temps réel, cette bibliothèque support une grande variété de langages de programmation comme Python, Java, C++, etc elle peut traiter des images ou des vidéos identifier des objets.

Plus précisément la bibliothèque est conçu pour résoudre les problèmes de vision par ordinateur. [81]

FIG. V.5 : Open cv logo [81]

3.5 Roboflow

Roboflow est un outil de développement pour la vision par ordinateur, Roboflow fournit tout ce dont vous avez besoin pour transformer les images en informations. il contient

tous les outils nécessaires pour commencer à utiliser la vision par ordinateur, même si vous n'êtes pas un expert en apprentissage automatique. [73]

FIG. V.6 : Roboflow logo [73]

Plus précisément, vous pouvez utiliser Roboflow pour :

- Stockez et organisez vos données d'image.
- Annoter des images.
- Convertissez des dizaines de formats d'annotation différents.
- Comprenez et évaluez la qualité de vos ensembles de données.
- Traitez et augmentez vos images pour aider vos modèles à s'entraîner plus rapidement et à mieux généraliser.
- Contrôlez les versions des ensembles de données et partagez-les avec votre équipe.
- Déployez sur des appareils périphériques tels que NVIDIA Jetson.

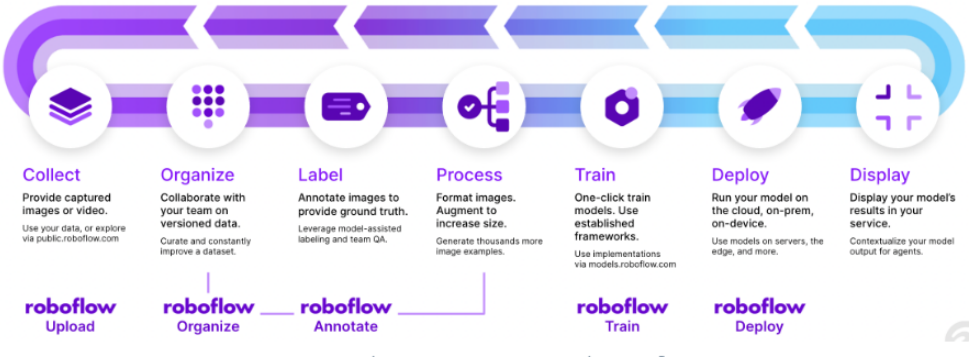


FIG. V.7 : Quesque roboflow peut faire [73]

4 Résultats

4.1 Le jeu de données (Dataset)

Le jeu de données utilisé dans l'étape d'apprentissage est composé d'environ 4063 images avec une taille de 224 x 224, ces images en été captures à partir des vidéos qui ont été prises avec des drones :

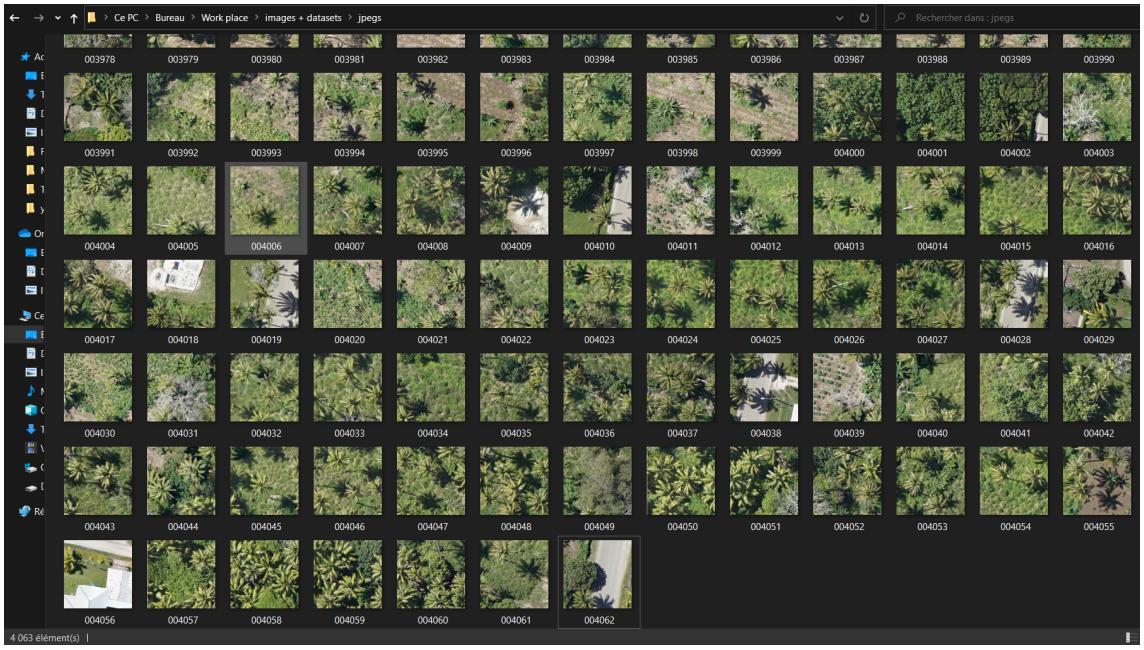


FIG. V.8 : Les images collecté pour notre jeu de données

Après une étape de filtrage des images collectées, dans laquelle nous avons sélectionné environ 1994 images utiles, puis nous avons étiqueté ces images à l'aide de Roboflow [73], ensuite nous avons augmenté le chiffre à 5633 images avec la technique de data augmentation, en effet ces images contiennent une moyenne de 3 palmiers étiquetés donc une moyenne de 16899 instances (palmiers étiquetés manuellement).

Les caractéristiques de notre jeu de donnes (Dataset) :

- **Cohérence des étiquettes :** Toutes les palmiers dans toutes les images sont étiquetées.
- **Précision de l'étiquette :**
  - Les étiquettes entoure étroitement chaque palmier.
  - Aucun espace existe entre un palmier et sa boîte englobant.
  - Aucun palmier na manquer d'étiquette.
- **Variété d'images** : avec l'aide de l'augmentation des donnes on a rassemblé des images de différentes heures de la journée, de différentes saisons, de conditions météorologiques différentes, d'éclairages différents, d'angles différents.

- **l'arrière-plan des images** : Les images d'arrière-plan sont des images sans objets qui sont ajoutées à un ensemble de données pour réduire les faux positifs (FP), notre jeu de données contient environ 3 % images d'arrière-plan, ainsi avec cette étape la précession augmente.

Fig. V.9 : Une partie des images qui ont été étiqueté manuellement

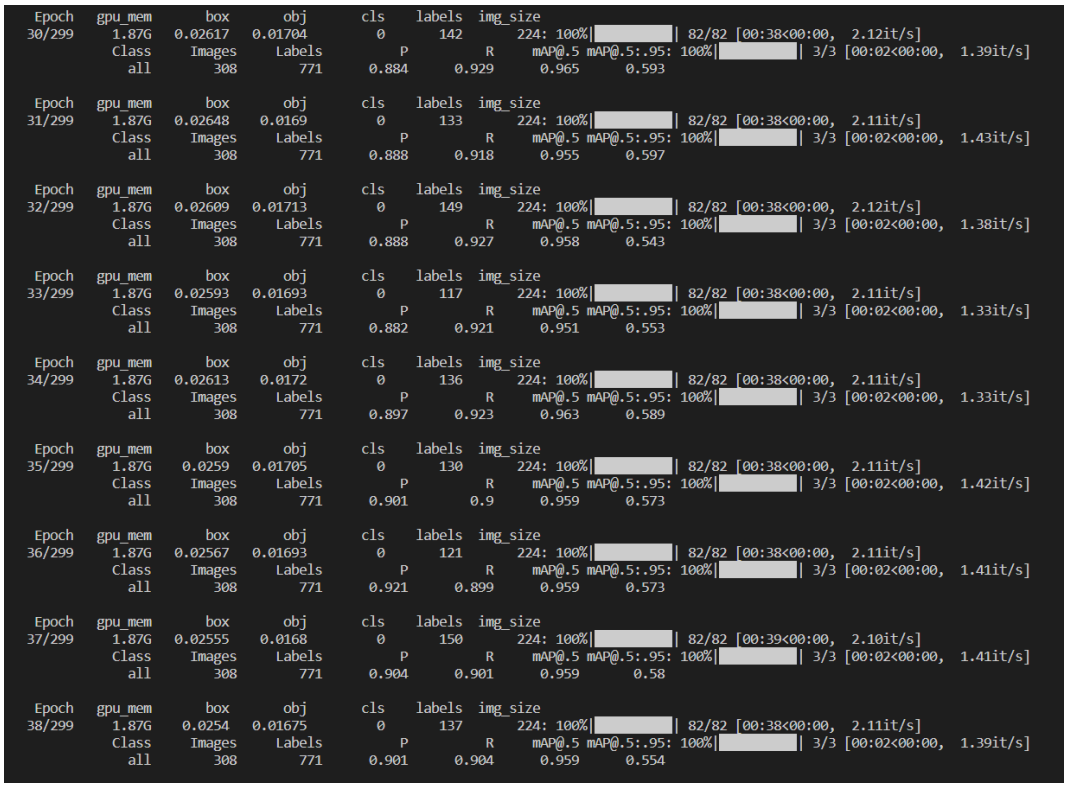
Maintenant que nous avons l'ensemble de données, nous avons converté les annotations dans un format accepté par YOLO avec Roboflow. [73]

4.2 L'étape de l'entraînement

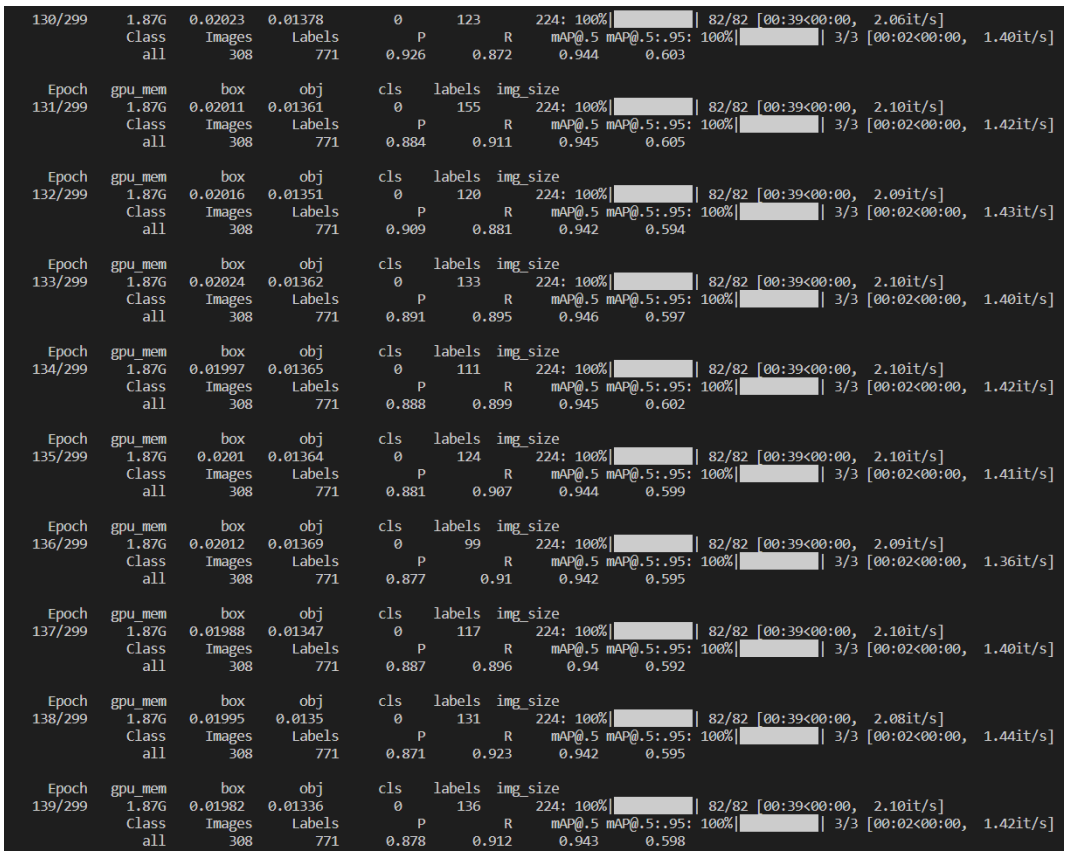
Dans cette étape en utilisant le jeu de données résultant à l'entraînement du modèle (YOLO v5). La figure (V.10) montre les paramètres d'entraînement, et la figure (V.11) ci dessous montre l'étape d'apprentissage de notre modèle :

```
(env) C:\Detection Project>python train.py --img 224 --cfg yolov5s.yaml --data data.yaml --epochs 300 --batch 64 --workers 1 --name train_result
```

Fig. V.10 : Les paramètres d'entraînement passé dans le modèle



(1)



(2)

Fig. V.11 : (1)(2) L'étape de l'apprentissage du réseau

La Figure (V.12) ci dessous montre l'architecture ainsi l'ensemble des paramètres du modèle (YOLO v5) :

```
                from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Conv                      [3, 32, 6, 2, 2]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     18816  models.common.C3                        [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  2    115712  models.common.C3                        [128, 128, 2]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  3    625152  models.common.C3                        [256, 256, 3]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  1   1182720  models.common.C3                        [512, 512, 1]
  9                -1  1    656896  models.common.SPPF                      [512, 512, 5]
 10                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12          [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  1    361984  models.common.C3                        [512, 256, 1, False]
 14                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 16          [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  1     90880  models.common.C3                        [256, 128, 1, False]
 18                -1  1    147712  models.common.Conv                      [128, 128, 3, 2]
 19          [-1, 14]  1         0  models.common.Concat                    [1]
 20                -1  1    296448  models.common.C3                        [256, 256, 1, False]
 21                -1  1    590336  models.common.Conv                      [256, 256, 3, 2]
 22          [-1, 10]  1         0  models.common.Concat                    [1]
 23                -1  1   1182720  models.common.C3                        [512, 512, 1, False]
 24      [17, 20, 23]  1     16182  models.yolo.Detect                      [1, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90, 156, 198, 373, 326]], [
128, 256, 512]]
Model Summary: 270 layers, 7022326 parameters, 7022326 gradients, 15.8 GFLOPs
```

Fig. V.12 : Architecture du modèle et ses paramétrées

4.3 Évaluation des performances

Après une étape d'apprentissage de notre modèle sur le jeu de données, la figure (V.13) présente les résultats de l'apprentissage obtenu :

```
162 epochs completed in 1.925 hours.
Optimizer stripped from runs\train\big_data\weights\last.pt, 14.3MB
Optimizer stripped from runs\train\big_data\weights\best.pt, 14.3MB

Validating runs\train\big_data\weights\best.pt...
Fusing layers...
Model Summary: 213 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
             Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100%|          | 3/3 [00:03<00:00,  1.24s/it]
               all        308        771      0.892      0.923      0.965      0.613
Results saved to runs\train\big_data
```

Fig. V.13 : Les résultats de l'apprentissage obtenu

L'entraînement a duré environ 2 heures 35 minutes.

- **La précision** : La précision est la capacité d'un modèle à identifier uniquement les objets pertinents, la figure V.14 présente la courbe d'évolution de la valeur de précision :

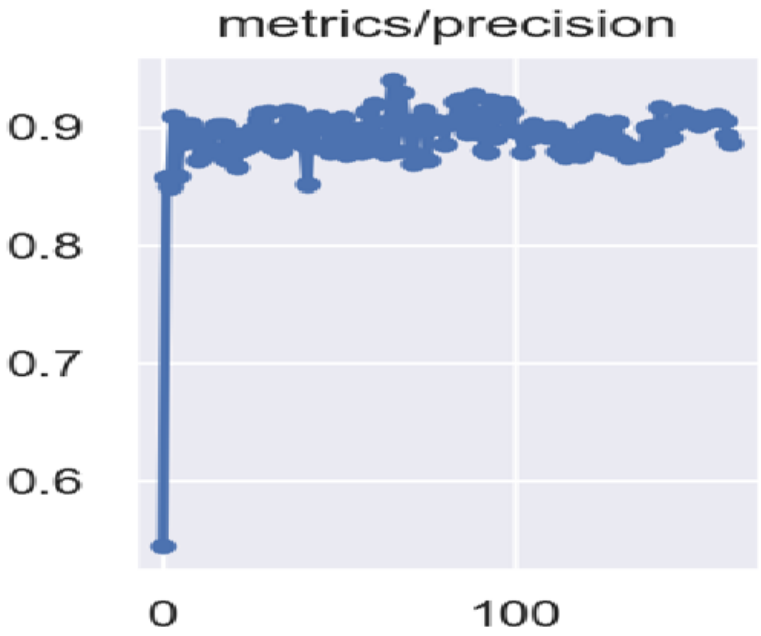


Fig. V.14 : Courbe de précision

Cela signifie que notre modèle est performant à 89,2% pour identifier uniquement les objets pertinents.

- **Le rappel (Évaluation de la localisation)** : Le rappel est la capacité d'un modèle à trouver toutes les boîtes englobantes correctes,

  La figure V.15 présente la courbe d'évolution de la valeur de rappel :

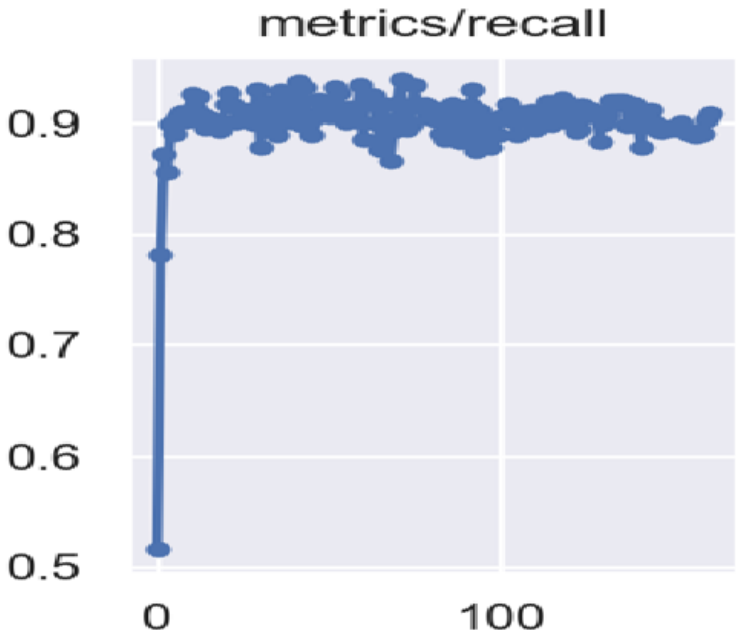


Fig. V.15 : Courbe de rappel

Cela signifie que notre modèle est performant à 92,3% pour proposer des régions pertinentes.

- **mAP (Mean Average Precision)** : La précision moyenne (mAP) est une méthode utilisée pour évaluer les modèles de détection d'objets tels que le R-CNN rapide, YOLO, le R-CNN à masque, etc. Elle calcule la précision en tenant compte de la précision de la classification et aussi de la localisation.

  La moyenne des valeurs de précision moyenne (mAP) est calculée sur les valeurs de rappel de 0 à 1, la figure V16 présente la courbe d'évolution de la valeur de mAP@.50IOU :

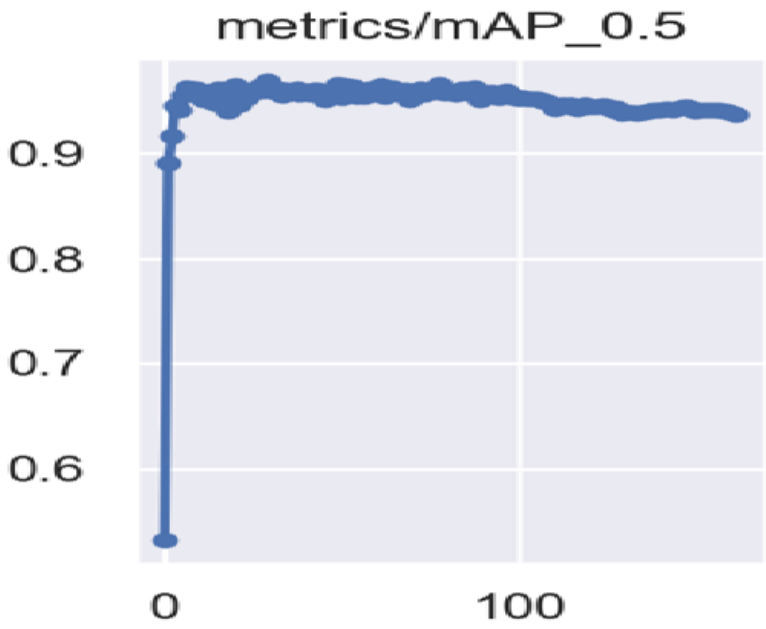


Fig. V.16 : Courbe d'évolution du mAP@.50IOU

Cela signifie que notre modèle est performant à 96,5%, lorsque le seuil IOU est de 50.

4.4 Les résultats de la détection

Après l'exécution de l'apprentissage, nous avons obtenu les poids d'apprentissage, ainsi nous allons passé à l'étape final de détection.

Dans l'étape de détection, l'entrée pour l'inférence peut être une image, une vidéo, un répertoire, une webcam, un flux ou même un lien youtube. Dans la commande de détection dans la figure(V.17) suivante :

```
(env) C:\Detection Project\yolov5>python detect.py --source C:\Users\oussa\OneDrive\Bureau\TEST\5.mp4 --weights runs\train\OneClass\weights\best.pt
detect: weights=['runs\\train\\OneClass\\weights\\best.pt'], source=C:\Users\oussa\OneDrive\Bureau\TEST\5.mp4, data=data\coco128.yaml, imgsz=[640, 640], conf_thres=0.25, iou_thres=0.45,
max_det=1000, device=, view_img=False, save_txt=False, save_conf=False, save_crop=False, nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, update=False, pro
ject=runs\detect, name=exp, exist_ok=False, line_thickness=3, hide_labels=False, hide_conf=False, half=False, dnn=False
YOLOv5  2022-5-28 torch 1.7.1+cu101 CUDA:0 (GeForce GTX 1050, 4096MiB)
```

Fig. V.17 : Détection commande

Les figures ci-dessous représente les différents résultats sur des vidéos et des images prise avec des drones

FIG. V.18 : Résultat de notre modèle sur une image de test.

La figure ci-dessous montre une prédiction effectuée par notre modèle sur une séquence vidéo.

FIG. V.19 : Résultat de notre modèle sur une séquence vidéo.

(a)

(b)

FIG. V.20 : Résultat de notre modèle sur des images de jeu de données Stanford Drone Dataset

La Fig. V.21 montre que notre modèle capable de détecter et de compter des palmiers dans des zones forestières dense.

Fig. V.21 : Détection et comptage des palmiers dans des zones forestières dense

Fig. V.22 : Résultat obtenu par notre modèle

la Fig. V.23 et Fig. V.24 montrent l'efficacité de notre modèle pour effectuer une détection et comptage dans les différents endroits réguliers ou dense.

FIG. V.23 : Résultat de détection sur une image dont la distribution des palmier est régulière

FIG. V.24 : Résultat de détection sur une image dont la distribution des palmier est irrégulière et dense

5 Comparaison

Pour démontrer l'efficacité de notre approche, nous allons comparer le modèle avec la couche optimisé est l'ancien modèle, en effectuant une évaluation quantitative en comparant la précision de chaque modèle.

Nous avons utilisé trois différentes métriques d'évaluation, la première est le rappel, la seconde est la précision, et la dernière est le Fscore. [82]

- Precision = TP / (TP + FP)
- Recall = TP / (TP + FN)
- F1-score = 2 (Precision x Recall) / (Precision + Recall)

5.1 Comparaison quantitative

Le tableau montre une comparaison entre le modèle optimisé et l'ancien modèle sur notre jeu de données (dataset) utilisés comme objet de référence.

Modèle	Precision	Recall	F-score	Temps d'entraînement
Yolov5 (BottleneckCSP)	81,6 %	92,1 %	86,53 %	4 h 36 m
Yolov5 (C3 couche optimisé)	89,2 %	92,3 %	90,72 %	2 h 32 m 50 s

Tab. V.2 : Comparaison quantitative

Nous constatons que notre méthode donne de meilleurs résultats pour la détection des palmiers que l'ancien modèle, il a été déterminé que le modèle optimisé est plus performant que l'ancien modèle, qui donne un score F de 90,72 % ce qui prouve son efficacité pour la détection des palmiers, et le temps d'entraînement dans le modèle optimisé est beaucoup plus moins que le temps d'entraînement dans l'ancien modèle.

6 Test et résultat sur d'autre jeu de données

Dans cette partie nous allons tester/valider les performances du modèle Yolov5 optimisé est réent-rainé sur le jeu de données COCO [83] qui contient 80 classes, c'est-à-dire 80 objets.

Ce test est établi sur des données choisies aléatoirement sur le web, et on a obtenu de bons résultats même dans des scènes complexes. et nous avons obtenues les résultats suivants :

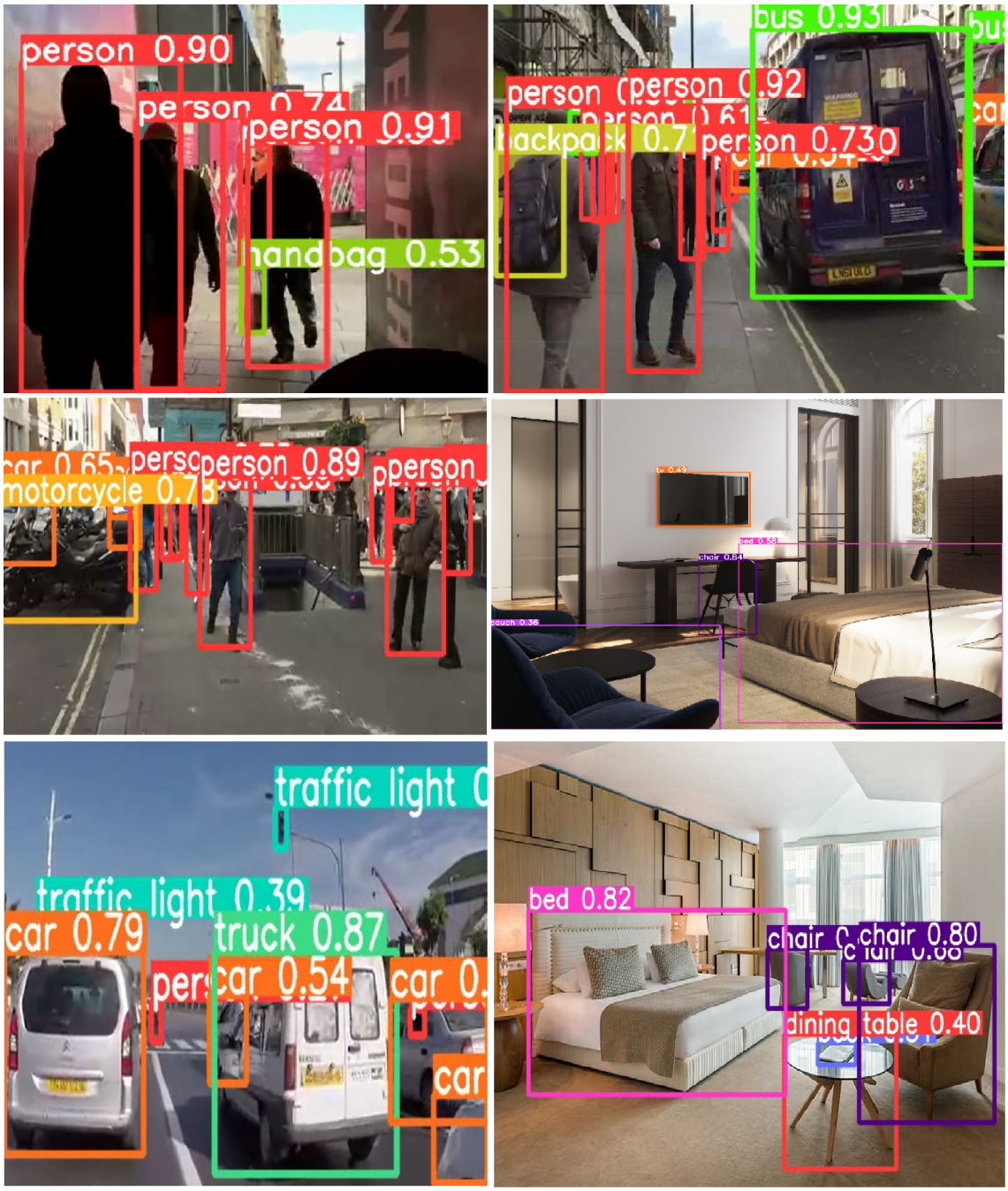


Fig. V.25 : Résultats obtenu par notre modèle sur le jeu de données COCO

Nous constatons que la plupart des boîtes de délimitation cadrent bien les objets avec une précision élevée. L'acuraccy du modèle optimisé est très intéressante par apport aux autres modèles de détection en temps réel.

7 Conclusion

La détection d'objets est une tâche de vision par ordinateur qui consiste à identifier la présence, l'emplacement, et le type d'un ou plusieurs objets dans une image. En s'appuyant sur des méthodes de reconnaissance d'objets, de localisation d'objets et la classification des objets. Les techniques basées sur les réseaux de neurones en particulier l'architecture convolutif du modèle YOLO, permettent d'obtenir des résultats satisfaisants capable d'effectuer la détection d'objets en temps réel.

Dans ce chapitre nous avons présenté les résultats obtenus et nous avons procédé à l'évaluation du modèle, ainsi cette étude propose une nouvelle approche pour détecter les palmiers à partir d'images ou des vidéos prise par des drones dans des région choisies aléatoirement, on a utilisé un système basé sur YOLOv5 en effet nous avons modifié l'architecture du Backbone afin d'optimiser l'étape de détection et de reconnaissance des palmiers. Notre modèle produit de bons résultats en terme de précision.

Conclusion et travaux futurs

Au cours des dernières années, la détection d'objets d'apprentissage en profondeur a parcouru un long chemin, Aujourd'hui, de nombreuses applications utilisent les réseaux de détection d'objets comme l'un de leurs composants principaux. C'est dans votre téléphone, ordinateur, voiture, appareil photo, drone, et plus encore. Il sera intéressant (et peut-être effrayant) de voir ce qui peut être réalisé avec des réseaux de neurones de plus en plus avancés.

Dans ce mémoire, nous avons utilisé une méthode basée sur les réseaux de neurones convolutifs, comme nous avons pu déjà l'indiquer, les modèles de détection moderne sont en mesure d'apporter des solutions pour faciliter la détection d'objets à partir d'images ou vidéo de drones, mais ça reste un problème complexe pour classer et trouver un nombre inconnu de palmiers individuels dans des zones différents et denses avec un nombre infini d'objets, afin de remédier à ce problème, nous avons proposé une nouvelle méthode pour détecter les palmiers à huile à partir d'images de drones, en effets cette méthode consiste à une technique de pré-traitement des images dans laquelle une Dataset a été formée avec les annotations correspondantes et le jeu de données a été augmentée et uniformisée avec une technique de Data augmentation, ainsi nous avons choisi le modèle YOLO v5 qui a satisfait nos objectifs, YOLO v5 a prouver ses performances avec des résultats intéressantes qui ont obtenu dans des zones choisis aléatoirement.

Notons néanmoins que plusieurs améliorations potentielles ont été observées à l'usage et pourraient faire l'objet de travaux futurs, cette étude propose une nouvelle approche pour détecter les palmiers à huile à partir d'images ou vidéo de drones, notre méthode produit de bons résultats en termes de précision et de qualité visuel, et cela en prenant en compte tous les paramètres du modèle.

Bibliographie

[1] Dominique Lambert. *La robotique et l'intelligence artificielle*. Fidélité, 2019.

[2] AN Ramesh et al. "Artificial intelligence in medicine." In : *Annals of the Royal College of Surgeons of England* 86.5 (2004), p. 334.

[3] Yann Le Cun. *Quand la machine apprend : la révolution des neurones artificiels et de l'apprentissage profond*. Odile Jacob, 2019.

[4] Nicolas Stroppa. "Définitions et caractérisations de modèles à base d'analogies pour l'apprentissage automatique des langues naturelles". Thèse de doct. Télécom ParisTech, 2005.

[5] Andreas Maier et al. "A gentle introduction to deep learning in medical image processing". In : *Zeitschrift für Medizinische Physik* 29.2 (2019), p. 86-101.

[6] Philippe Esling et Ninon Devis. "Creativity in the era of artificial intelligence". In : *arXiv preprint arXiv :2008.05959* (2020).

[7] Jacques Haiech. "Parcourir l'histoire de l'intelligence artificielle, pour mieux la définir et la comprendre". In : *médecine/sciences* 36.10 (2020), p. 919-923.

[8] CPC Munaiseche, DR Kaparang et Parabelem Tinno Dolf Rompas. "An Expert system for diagnosing eye diseases using forward chaining method". In : *IOP Conference Series : Materials Science and Engineering*. T. 306. 1. IOP Publishing. 2018, p. 012023.

[9] Laurence Negrello. *Systèmes experts et intelligence artificielle*. Schneider Electric España SA, 1991.

[10] Shahid Hussain Wani, Abreen Rashid et Mohammad Ilyas Malik. "DISTRIBUTED DATA MINING USING MULTI AGENT DATA". In : ().

[11] John Tranier. "Vers une vision intégrale des systèmes multi-agents : Contribution à l'intégration des concepts d'agent, d'environnement, d'organisation et d'institution." Thèse de doct. Université Montpellier II-Sciences et Techniques du Languedoc, 2007.

[12] Batta Mahesh. "Machine learning algorithms-a review". In : *International Journal of Science and Research (IJSR).[Internet]* 9 (2020), p. 381-386.

[13] Dominique Cardon, Jean-Philippe Cointet et Antoine Mazières. "La revanche des neurones". In : *Réseaux* 5 (2018), p. 173-220.

[14] Zuolin Dong et al. "Face detection in security monitoring based on artificial intelligence video retrieval technology". In : *IEEE Access* 8 (2020), p. 63421-63433.

[15] Rolando J Cárdenas, Cesar A Beltrán et Juan C Gutiérrez. "Small face detection using deep learning on surveillance videos". In : *environment* 2.5 (2019), p. 14.

[16] CURÉ Noé et al. "Véhicule autonome et connecté : communication V2X". In : ().

[17] Alexandre Mathieu-Fritz. "L'intelligence artificielle en médecine : des promesses aux usages… en passant par la conception". In : *Sciences sociales et santé* 39.2 (2021), p. 71-78.

[18] Yvan Foster. "Watson d'IBM, le petit futé de l'informatique cognitive". In : *Gestion* 40.1 (2015), p. 86-89.

[19] *L'Université de Berne et l'Inselspital créent le Center for Artificial Intelligence in Medicine*. https://www.ggba-switzerland.ch/luniversite-de-berne-linselspital-creent-center-for-artificial-intelligence-in-medicine/.

[20] Kevin Renier et al. "Vulgarisation des chatbots et analyse business." In : (2017).

[21] Chloé-Agathe Azencott. *Introduction au machine learning*. Dunod, 2019.

[22] Meherwar Fatima, Maruf Pasha et al. "Survey of machine learning algorithms for disease diagnostic". In : *Journal of Intelligent Learning Systems and Applications* 9.01 (2017), p. 1.

[23] Taiwo Oladipupo Ayodele. "Types of machine learning algorithms". In : *New advances in machine learning* 3 (2010), p. 19-48.

[24] Jesper E Van Engelen et Holger H Hoos. "A survey on semi-supervised learning". In : *Machine Learning* 109.2 (2020), p. 373-440.

[25] Xing Lin et al. "All-optical machine learning using diffractive deep neural networks". In : *Science* 361.6406 (2018), p. 1004-1008.

[26] http://www.psychomedia.qc.ca/lexique/definition/apprentissage-profond.

[27] Christian Janiesch, Patrick Zschech et Kai Heinrich. "Machine learning and deep learning". In : *Electronic Markets* 31.3 (2021), p. 685-695.

[28] HABBA Abdelaziz, ISHAK Omar, Abdelwhab OUAHAB et al. "La classification des images satellitaires par l'apprentissage profonde (deep learning)". Thèse de doct. Université Ahmed Draïa-Adrar, 2019.

[29] Clément Douarre David Rousseau. *Introduction à l'apprentissage profond (deep learning) de l'intelligence artificielle*. https://culturesciencesphysique.ens-lyon.fr/ressource/IA-apprentissage-Rousseau.xml. octobre 2021, CultureSciences Physique - ISSN 2554-876X.

[30] Rikiya Yamashita et al. "Convolutional neural networks : an overview and application in radiology". In : *Insights into imaging* 9.4 (2018), p. 611-629.

[31] Pierre Buyssens et Abderrahim Elmoataz. "Réseaux de neurones convolutionnels multi-échelle pour la classification cellulaire". In : *RFIA 2016*. 2016.

[32] Ye Zhang et Byron Wallace. "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification". In : *arXiv preprint arXiv :1510.03820* (2015).

[33] Zhengxia Zou et al. “Object detection in 20 years : A survey”. In : *arXiv preprint arXiv :1905.05055* (2019).

[34] Ajeet Ram Pathak, Manjusha Pandey et Siddharth Rautaray. “Application of deep learning for object detection”. In : *Procedia computer science* 132 (2018), p. 1706-1717.

[35] Zhong-Qiu Zhao et al. “Object detection with deep learning : A review”. In : *IEEE transactions on neural networks and learning systems* 30.11 (2019), p. 3212-3232.

[36] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In : *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, p. 580-587.

[37] Ross Girshick. “Fast r-cnn”. In : *Proceedings of the IEEE international conference on computer vision*. 2015, p. 1440-1448.

[38] Shaoqing Ren et al. “Faster r-cnn : Towards real-time object detection with region proposal networks”. In : *Advances in neural information processing systems* 28 (2015).

[39] Kaiming He et al. “Mask r-cnn”. In : *Proceedings of the IEEE international conference on computer vision*. 2017, p. 2961-2969.

[40] Wei Liu et al. “Ssd : Single shot multibox detector”. In : *European conference on computer vision*. Springer. 2016, p. 21-37.

[41] Joseph Redmon et al. “You only look once : Unified, real-time object detection”. In : *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, p. 779-788.

[42] Jiahui Yu et al. “Unitbox : An advanced object detection network”. In : *Proceedings of the 24th ACM international conference on Multimedia*. 2016, p. 516-520.

[43] Aleksa Ćorović et al. “The real-time detection of traffic participants using YOLO algorithm”. In : *2018 26th Telecommunications Forum (TELFOR)*. IEEE. 2018, p. 1-4.

[44] Καλλιόπη Βίλταρη. “Classification of traffic signs based on object recognition”. In : (2021).

[45] Joseph Redmon et Ali Farhadi. “YOLO9000: better, faster, stronger”. In : *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, p. 7263-7271.

[46] Joseph Redmon et Ali Farhadi. “Yolov3: An incremental improvement”. In : *arXiv preprint arXiv :1804.02767* (2018).

[47] MG Dorrer et AE Tolmacheva. “Comparison of the YOLOv3 and Mask R-CNN architectures’ efficiency in the smart refrigerator’s computer vision”. In : *Journal of Physics : Conference Series*. T. 1679. 4. IOP Publishing. 2020, p. 042022.

[48] Alexey Bochkovskiy, Chien-Yao Wang et Hong-Yuan Mark Liao. “Yolov4: Optimal speed and accuracy of object detection”. In : *arXiv preprint arXiv :2004.10934* (2020).

[49] Chien-Yao Wang et al. “CSPNet : A new backbone that can enhance learning capability of CNN”. In : *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, p. 390-391.

[50] Kaiming He et al. “Spatial pyramid pooling in deep convolutional networks for visual recognition”. In : *IEEE transactions on pattern analysis and machine intelligence* 37.9 (2015), p. 1904-1916.

[51] Shu Liu et al. “Path aggregation network for instance segmentation”. In : *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, p. 8759-8768.

[52] Karen Simonyan et Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In : *arXiv preprint arXiv :1409.1556* (2014).

[53] Kaiming He et al. “Deep residual learning for image recognition”. In : *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, p. 770-778.

[54] Xianzhi Du et al. “Spinenet : Learning scale-permuted backbone for recognition and localization”. In : *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, p. 11592-11601.

[55] Mingxing Tan et Quoc Le. “Efficientnet : Rethinking model scaling for convolutional neural networks”. In : *International conference on machine learning*. PMLR. 2019, p. 6105-6114.

[56] Rahul Vishwakarma et Ravigopal Vennelakanti. “Cnn model & tuning for global road damage detection”. In : *2020 IEEE International Conference on Big Data (Big Data)*. IEEE. 2020, p. 5609-5615.

[57] Tsung-Yi Lin et al. “Feature pyramid networks for object detection”. In : *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, p. 2117-2125.

[58] Golnaz Ghiasi, Tsung-Yi Lin et Quoc V Le. “Nas-fpn : Learning scalable feature pyramid architecture for object detection”. In : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, p. 7036-7045.

[59] Mingxing Tan, Ruoming Pang et Quoc V Le. “Efficientdet : Scalable and efficient object detection”. In : *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, p. 10781-10790.

[60] Songtao Liu, Di Huang et Yunhong Wang. “Learning spatial fusion for single-shot object detection”. In : *arXiv preprint arXiv :1911.09516* (2019).

[61] Qijie Zhao et al. “M2det : A single-shot object detector based on multi-level feature pyramid network”. In : *Proceedings of the AAAI conference on artificial intelligence*. T. 33. 01. 2019, p. 9259-9266.

[62] Tsung-Yi Lin et al. “Focal loss for dense object detection”. In : *Proceedings of the IEEE international conference on computer vision*. 2017, p. 2980-2988.

[63] Hei Law et Jia Deng. “Cornernet : Detecting objects as paired keypoints”. In : *Proceedings of the European conference on computer vision (ECCV)*. 2018, p. 734-750.

[64] Kaiwen Duan et al. “Centernet : Keypoint triplets for object detection”. In : *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, p. 6569-6578.

[65] Abdullah Rashwan, Agastya Kalra et Pascal Poupart. “Matrix Nets : A new deep architecture for object detection”. In : *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019, p. 0–0.

[66] Zhi Tian et al. “Fcos : Fully convolutional one-stage object detection”. In : *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, p. 9627-9636.

[67] Jifeng Dai et al. “R-fcn : Object detection via region-based fully convolutional networks”. In : *Advances in neural information processing systems* 29 (2016).

[68] Ze Yang et al. “Reppoints : Point set representation for object detection”. In : *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, p. 9657-9666.

[69] Xinni Liu et al. “Automatic detection of oil palm tree from UAV images based on the deep learning method”. In : *Applied Artificial Intelligence* 35.1 (2021), p. 13-24.

[70] Weijia Li et al. “Deep learning based oil palm tree detection and counting for high-resolution remote sensing images”. In : *Remote sensing* 9.1 (2016), p. 22.

[71] Adel Ammar, Anis Koubaa et Bilel Benjdira. “Deep-Learning-based Automated Palm Tree Counting and Geolocation in Large Farms from Aerial Geotagged Images”. In : *Agronomy* 11.8 (2021), p. 1458.

[72] Chee Cheong Lee et al. “No. 2 Oil Palm Tree Detection from High Resolution Drone Image Using Convolutional Neural Network”. In : *Journal of Engineering Technology and Applied Physics* 1.2 (2019), p. 6-9.

[73] https://roboflow.com/.

[74] Van Dyk, David A., and Xiao-Li Meng”. “The art of data augmentation. Journal of Computational and Graphical Statistics”. In : (2001).

[75] Luke Taylor et Geoff Nitschke. “Improving deep learning with generic data augmentation”. In : *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. 2018, p. 1542-1547.

[76] Do Thuan. “Evolution of yolo algorithm and yolov5: the state-of-the-art object detection algorithm”. In : (2021).

[77] Stefan Elfwing, Eiji Uchibe et Kenji Doya. “Sigmoid-weighted linear units for neural network function approximation in reinforcement learning”. In : *Neural Networks* 107 (2018), p. 3-11.

[78] *Python 3.7.0 Python Software Foundation. Python Language Reference,Reference Available at.* https://www.python.org/.

[79] *Pytorch , version 1.7.1+cu101, Reference.* https://pytorch.org/.

[80] *Cuda and cuDNN , Cuda : version 10.2, cuDNN : version v8.3.3, Reference.* https://developer.nvidia.com/deep-learning-software/.

[81] *OpenCv, Reference.* https://opencv.org/.

[82] Muhammed Enes ATİK, Zaide DURAN et Roni ÖZGÜNLÜK. “Comparison of YOLO Versions for Object Detection from Aerial Images”. In : *International Journal of Environment and Geoinformatics* 9.2 (), p. 87-93.

[83] *COCO Dataset,Reference Available at.* https://cocodataset.org/.