



UNIVERSITÉ DE
CARTHAGE
Faculté des sciences de Bizerte
Département Informatique



Mémoire de fin d'études

Pour l'obtention du diplôme de Master professionnel en Data Science

DÉPLOIEMENT D'UN MODÈLE DE DÉTECTION ET RECONNAISSANCE DES LOGOS DANS UNE APPLICATION MOBILE

Au sein de

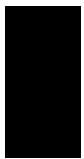
MarkomWeb (MW)



Réalisé par :
Adel Sidi M'Hamed

Encadré par :
Mme. OLFA dridi **FSB**
Mme. SAMAR Saidi **MW**

Année universitaire 2021-2022



DEDICATION

À ma très chère mère, quoi que je fasse ou dise, je ne pourrai pas te remercier comme je le devrais. 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, tu as toujours été à mes côtés pour me soutenir et m'encourager. Que ce travail exprime ma gratitude et mon affection.

À mes frères et sœurs pour leur soutien.

À tous mes proches, pour leur bienveillance et amour.

À vous tous,

Je dédie ce modeste travail.

Adel SIDI M'HAMED

REMERCIEMENT

Avant de présenter mon travail, je tiens à remercier toutes les personnes qui ont contribuées de près ou de loin à la réalisation de ce mémoire.

J'exprime aussi ma gratitude à Madame **Samar Saidi** mon encadrante organisme, pour le temps qu'elle a consacré pour me guider à travers ses paroles, ses écrits, ses conseils et ses critiques et me répondre à mes questions durant mon stage.

Je voudrais aussi exprimer ma profonde gratitude à Madame **Olfa Dridi**, mon superviseur, qui m'a donné des conseils tout au long de mon travail et n'a ménagé aucun effort pour contribuer à la réussite de ce projet.

Je remercie également mon professeur Monsieur **Anis Ben Aicha**, Coordinateur de Master MPDS, pour toutes les facilités accordées.

Enfin, j'adresse mes sincères remerciements également à tous les professeurs du département informatique de la FSB.

Abstract

The goal of this work is to research and apply deep learning object detection algorithms to help people, detecting counterfeit brands. In this study, we focused on the YOLO object detection algorithm, which is a practically recognized and approved method. Therefore, we tried to re-train the YOLO model on our manually collected database classes which is a bit small due to the lack of data. We obtained very satisfactory results, with an overview that reflects the great superiority of the YOLO real-time object detection algorithm over many other models.

Keywords :

Détection d'objets, Réseaux de neurones convolutifs (CNN), Apprentissage en profondeur (Deep Learning), YOLO.

Résumé

Le but de ce travail est de rechercher et d'appliquer des algorithmes de détection d'objets d'apprentissage en profondeur pour aider les personnes, en détectant les marques contrefaites. Nous nous sommes concentrés dans cette étude sur l'algorithme de détection d'objet YOLO, qui est une méthode pratiquement reconnue et approuvée. Par conséquent, nous avons essayé de ré-entraîner le model YOLO sur nos classes de la base de données collecté manuellement qui est une base un peu petite à cause de la manque des données. Nous avons obtenu des résultats très satisfaisants, avec un aperçu qui reflète la grande supériorité de l'algorithme de détection d'objet YOLO en temps réel sur beaucoup d'autres modèles.

TABLE DES MATIÈRES

LISTE DES FIGURES	ix
LISTE DES ABRÉVIATIONS	x
INTRODUCTION GÉNÉRALE	xii
1 CONTEXTE GÉNÉRAL DU PROJET	2
1.1 INTRODUCTION	3
1.2 PRÉSENTATION DE L'ENTREPRISE	3
1.2.1 MISSION D'ENTREPRISE	3
1.2.2 SERVICES D'ENTREPRISE	4
1.3 OBJECTIF DU PROJET	4
1.4 PROBLÉMATIQUE ET SOLUTION PROPOSÉE	5
1.4.1 PROBLÉMATIQUE	5
1.4.2 SOLUTION PROPOSÉE	6
1.5 ÉTUDE DE L'EXISTANT	6
1.6 ANALYSE DES BESOINS	7
1.6.1 BESOINS FONCTIONNELS	7
1.6.2 BESOINS NON FONCTIONNELS	8
1.6.2.1 SÉCURITÉ	8
1.6.2.2 PORTABILITÉ	8
1.6.2.3 PERFORMANCE	8
1.6.2.4 UTILISATION	8
1.6.2.5 DISPONIBILITÉ	8
1.7 RÉSULTATS ATTENDUS	9
1.8 CONCLUSION	9
2 DU ML AU DEEP LEARNING	10
2.1 INTRODUCTION	11

TABLE DES MATIÈRES

2.2	TYPES DE SYSTÈME D'APPRENTISSAGE AUTOMATIQUE	12
2.2.1	APPRENTISSAGE SUPERVISÉ	12
2.2.2	APPRENTISSAGE NON SUPERVISÉ	13
2.2.3	APPRENTISSAGE SEMI-SUPERVISÉ	14
2.2.4	APPRENTISSAGE AVEC RENFORCEMENT	15
2.3	DIFFÉRENCE ENTRE CLASSIFICATION ET RÉGRESSION	16
2.3.1	APPROXIMATION DE FONCTION	16
2.3.2	CLASSIFICATION	17
2.3.3	RÉGRESSION	17
2.4	RÉSEAUX DE NEURONES ARTIFICIELS	17
2.4.1	NEURONE BIOLOGIQUE	17
2.4.2	NEURONE FORMEL	18
2.4.2.1	FONCTION D'AGRÉGATION	19
2.4.2.2	FONCTION D'ACTIVATION	19
2.4.2.3	POIDS ET SEUILS	21
2.4.2.4	PERCEPTRON	21
2.4.2.5	RÉSEAU DE NEURONES A PROPAGATION AVANT (feed-forward)	23
2.4.3	APPRENTISSAGE DES RÉSEAUX DE NEURONES	23
2.4.3.1	APPRENTISSAGE DU PERCEPTRON	24
2.4.3.2	APPRENTISSAGE D'UN PERCEPTRON MULTICOUCHE	24
2.5	L'APPRENTISSAGE PROFOND (DEEP LEARNING)	26
2.5.1	INTÉRÊT DU DEEP LEARNING	26
2.5.2	ARCHITECTURES DES MODÈLES DE DEEP LEARNING	27
2.5.2.1	RÉSEAUX DE NEURONE A CONVOLUTION	27
2.5.3	RÉSEAUX DE NEURONES RÉCURRENTS	28
2.5.4	APPLICATION DU DEEP LEARNING	28
2.6	CONCLUSION	29
3	LES MODÈLES DE DÉTECTION	31
3.1	INTRODUCTION	32
3.2	RESEAUX DE NEURONES CONVOLUTION (CNN)	32
3.2.1	COUCHE DE CONVOLUTION	32
3.2.2	COUCHE D'ÉCHANTILLONNAGE	33
3.2.3	COUCHE COMPLÈTENT CONNECTÉE	34
3.3	LES MODÈLES DE DÉTECTION PAR CNN	34

TABLE DES MATIÈRES

3.3.1	R-CNN	36
3.3.2	FAST R-CNN	37
3.3.3	FASTER R-CNN	38
3.3.4	MASK R-CNN	38
3.3.5	SSD (SINGLE SHOT MULTIBOX DETECTOR)	39
3.3.6	YOLO (YOU ONLY LOOK ONCE)	40
3.3.6.1	LE MODÈLE YOLOV2	45
3.3.6.2	LE MODÈLE YOLOV3	46
3.3.6.3	LE MODÈLE YOLOV4	47
3.3.6.4	LE MODÈLE YOLOV5	48
3.4	MÉSUSER LA PERFORMANCE D'UN MODÈLE DE DÉTECTION	57
3.5	CONCLUSION	58
4	REALISATION	60
4.1	INTRODUCTION	61
4.2	ENVIRONNEMENT DE DÉVELOPPEMENT	61
4.2.1	ENVIRONNEMENT LOGICIEL	61
4.2.1.1	GOOGLE COLAB	61
4.2.1.2	ANDROID STUDIO	62
4.2.1.3	LABELIMG	63
4.2.2	OUTILS TECHNOLOGIQUES	63
4.2.2.1	LANGUE UTILISÉ	63
4.2.2.2	LES BIBLIOTHÈQUES ET LES FRAMEWORKS UTILISÉS	64
4.3	L'ENSEMBLE DES DONNÉES	65
4.3.1	COLLECTION DES DONNÉES	65
4.3.2	PRÉ-TRAITEMENT DES DONNÉES	66
4.3.2.1	NORMALISATION AVEC ÉGALISATION D'HISTOGRAMME	67
4.3.2.2	FILTRAGE AVEC LE MODE	67
4.3.2.3	LABELLISATION	68
4.3.2.4	DIVISER LES DONNÉES	70
4.4	APPRENTISSAGE DE MODEL YOLOV5	70
4.4.1	TÉLÉCHARGER LE MODÈLE YOLOV5	70
4.4.1.1	PRÉPARER LE MODÈLE POUR LE TRAIN	71
4.4.2	ENTRAÎNEMENT	73
4.4.2.1	EXTRACTION DE CARACTÉRISTIQUES	73
4.4.2.2	FINE TUNING	75

TABLE DES MATIÈRES

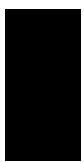
4.4.2.3	VALIDATION	76
4.4.2.4	INFÉRENCE	77
4.4.2.5	EXPORTATION VERS LE FORMAT TFLITE	78
4.5	APERÇU DE L'APPLICATION	79
4.6	CONCLUSION	80
CONCLUSION GÉNÉRALE		81
BIBLIOGRAPHIE		81

LISTE DES FIGURES

1.1	L'entreprise d'accueil	3
1.2	Contrefaçon des marques	5
1.3	Les différentes étapes	6
2.1	Un jeu d'entraînement étiqueté pour un apprentissage supervisé	13
2.2	Un jeu d'entraînement non étiqueté pour un apprentissage non supervisé	14
2.3	Apprentissage semi-supervisé	15
2.4	Apprentissage par renforcement	16
2.5	Neurone biologique	18
2.6	Neurone formel	19
2.7	Fonction d'activation	21
2.8	Perceptron à une unité de sortie	22
2.9	Perceptron à plusieurs unités de sortie	22
2.10	Perceptron multicouche à deux couches intermédiaires	23
3.1	Différents types de convolution	33
3.2	Exemple d'une opération de pooling de taille 2×2	33
3.3	Architecture du modèle R-CNN	36
3.4	Architecture du modèle Fast R-CNN	37
3.5	Architecture du modèle Faster R-CNN	38
3.6	Architecture du modèle Mask R-CNN	39
3.7	Architecture de model SSD	40
3.8	Architecture du modèle Yolo	40
3.9	Diviser l'image en (S^*S) grille	41
3.10	Le vecteur prédit dans le cas d'une seule boite	42
3.11	Union sur Intersection	42
3.12	Exemples d'IoU : courtoisie	43
3.13	Le vecteur prédit dans le cas de plusieurs boites dans la cellule	43
3.14	Un tenseur qui spécifie les emplacements des boîtes et les probabilités de classe.	44

LISTE DES FIGURES

3.15 Le résultat de suppression non maximale	45
3.16 Architecture de darknet19	46
3.17 Architecture de modèle yolov3	47
3.18 Architecture du modèle yolov4	48
3.19 Architecture du modèle yolov5	49
3.20 DenseNet	49
3.21 Architecture Darknet-53	53
3.22 Améliorée SPP	54
3.23 Architecture de FPN de Yolov3 [50]	54
3.24 Architecture du PAN	55
3.25 Les boîtes d'ancrages	56
 4.1 Logo Google Colab	62
4.2 Logo Andoid Studio	62
4.3 Logo LabelImg	63
4.4 Aperçu d'ensembles de données	66
4.5 Égalisation de l'histogramme	67
4.6 Filtrage avec le mode	68
4.7 LabelImg	69
4.8 Aperçu du dataset	69
4.9 Téléchargement de yolov5	70
4.10 Configuration GPU	71
4.11 Configuration	71
4.12 Fichier data_configuration	72
4.13 Les fichiers de configuration YAML	73
4.14 Resultats d'extraction de caractéristiques trainig	74
4.15 Résultats de 'fine tuning' training	76
4.16 Précision — Courbe de rappel de la répartition des données de test	77
4.17 Résultats de l'inférence	78
4.18 Interface de l'application.	79
4.19 Interface de prédiction de l'application.	80



LISTE DES ABRÉVIATIONS

ML Machine learning

DL Deep learning

IA intelligence artificiel

MLP Multiple layers perceptron

GPU Graphocal processing units

IBM International Business machines

CNN Convolutional neural network

ReLU Rectified linear unit

RNN Réseaux de neurones récurrents

Img image

Conv convolutionnelles

RoI Region of Interest Pooling

MAP mean Average Precision

FC Full Connected

CSP Cross Stage Partial

SPP Spatial Pyramid Pooling

FPN Feature Pyramid Networks

PAN Path Aggregation Network

GIoU Generalized Intersection over union

hyp hyperparameters

LISTE DES ABRÉVIATIONS

FPS Frames per seconds

TTA Test time Augmentation

BDD Base des donnée

YOLO You only look once

FSB Faculté des sciences de Bizerte

INTRODUCTION GÉNÉRALE

Aujourd’hui nous avons tous remarqué l’énorme développement des capacités logicielles, informatiques, théoriques et logistiques au cours de la dernière décennie, ainsi le développement des algorithmes de l’intelligence artificielle dans plusieurs domaines de l’informatique telle que : la vision par ordinateur, la vidéo surveillance, les voitures autonomes, la robotique, l’aéronautique, des maisons intelligentes, des cités intelligentes, etc. Étant donné ce développement est très rapidement et même spectaculaire, et nous ne faisons que commencer.

Aujourd’hui, avec cette flamber des techniques nécessitant l’intelligence en général, et la reconnaissance et la détection d’objets en particulier, il est devenu nécessaire de développer des méthodes plus capables pour assurer toujours la précision et la rapidité tout en étant indépendant.

L’apprentissage automatique est un domaine de l’intelligence artificielle, qui fait référence à la capacité des systèmes informatiques au sein des machines à trouver indépendamment des solutions aux problèmes en percevant différents modèles de données. Parmi les algorithmes qui permettent à la machine d’apprendre par elle-même grâce à l’apprentissage profond en anglais deep learning, c’est la simulation des neurones du corps humain.

La plupart des recherches sur l’apprentissage en profondeur se concentrent sur la recherche de méthodes permettant d’obtenir un degré élevé d’abstraction en analysant un grand ensemble de données à l’aide de variables linéaires et non linéaires. D’où vienne la nécessité d’avoir des bases de données très riches et divers et surtout bien étiquetées.

Le problème de la détection d’objet en temps réel attire toujours les chercheurs, vu qu’il est toujours un grand défi, surtout en termes de précision, et en termes du temps.

INTRODUCTION GÉNÉRALE

Donc la disponibilité des ressources matériels sur le cloud, ainsi la disponibilité des bases de données de qualité, et l'émergence des modes et des architectures de deep Learning basés CNN plus robustes, nous a poussé à créer un système intelligent qui aide à lutter contre la contrefaçon de marques et protège les clients contre l'arnaque, et pour réaliser ce système nous avons choisi le modèle YOLO qui a prouvé ses performances dans la détection d'objet en temps réel, donc on a entraîné ce modèle sur des classes bien sélectionnées à partir d'une base de données collectée manuellement. Cela nous a permis d'obtenir un système rapide et efficace.

Ce mémoire se présente sous forme de quatre chapitres :

Le premier chapitre : présente l'entreprise d'accueil, la problématique et la solution proposée, puis l'étude d'existence et les résultats attendus.

Ensuite Le chapitre 2 : est consacré une présentation générale sur l'intelligence artificiel en présentant les méthodes de l'apprentissage automatique et les principes fondamentaux de deep learning

Chapitre 3 : dans ce chapitre, nous présentons les méthodes de détection d'objets basées sur l'apprentissage profond, nous commençons par la structure générale d'un modèle de détection d'objets en passant par les méthodes de détection et nous terminons par le choix du modèle yolov5 en particulier.

Chapitre 4 : Dans ce chapitre nous présentons l'environnement de développement, puis le jeu de données et les différentes étapes du projet, nous terminons par l'interprétation des résultats et les interfaces de l'application.

CONTEXTE GÉNÉRAL DU PROJET

Sommaire

1.1	INTRODUCTION	3
1.2	PRÉSENTATION DE L'ENTREPRISE	3
1.2.1	MISSION D'ENTREPRISE	3
1.2.2	SERVICES D'ENTREPRISE	4
1.3	OBJECTIF DU PROJET	4
1.4	PROBLÉMATIQUE ET SOLUTION PROPOSÉE	5
1.4.1	PROBLÉMATIQUE	5
1.4.2	SOLUTION PROPOSÉE	6
1.5	ÉTUDE DE L'EXISTANT	6
1.6	ANALYSE DES BESOINS	7
1.6.1	BESOINS FONCTIONNELS	7
1.6.2	BESOINS NON FONCTIONNELS	8
1.7	RÉSULTATS ATTENDUS	9
1.8	CONCLUSION	9

1.1 INTRODUCTION

L'objectif de ce premier chapitre est de mettre notre projet dans son contexte. Tout d'abord, nous présentons l'entreprise d'accueil. Ensuite, la problématique et la solution proposée, puis nous décrivons l'étude d'existence et les résultats attendus, et enfin nous tirons des conclusions.

1.2 PRÉSENTATION DE L'ENTREPRISE

MarKomWeb est une agence de communication spécialisée en stratégies du digital, experte sur le conseil et la formation, la production de sites performants et le pilotage des campagnes d'acquisition et de fidélisation dans l'objectif de réaliser des performances au travers d'un suivi quotidien avec un rapport mensuel.



FIGURE 1.1 – L'entreprise d'accueil

1.2.1 MISSION D'ENTREPRISE

La mission de cette entreprise est de guider les utilisateurs vers une communication moderne, créative et optimisée pour aider à démarquer dans le monde du digital. MarKomWeb ajoute la touche de créativité dont votre marque a besoin pour compléter la croissance de votre entreprise.

Étant des spécialistes des canaux de communications digitales vous aident à :

- Concevoir des stratégies marketing en actionnant les bons leviers.

CONTEXTE GÉNÉRAL DU PROJET

- Rendre visible vos projets digitaux en faisant de la publicité ciblée.
- Gagner plus de visibilité pour votre site web grâce aux référencements.
- Créer un site web afin d'exposer votre travail en ligne avec un design qui correspond à vos objectifs.
- Concevoir et développer des applications mobiles sur mesure.
- Préparer une identité visuelle impactante et des créations publicitaires originales.
- Préparer des vidéos marketing pour les diffuser sur vos réseaux.

1.2.2 SERVICES D'ENTREPRISE

Les services de l'entreprise sont :

- Conseil en communication
- Communication digitale et community management
- Référencement
- Service Web/Mobile
- Graphisme et motion design

1.3 OBJECTIF DU PROJET

L'objectif final de notre projet est de détecter des logos afin de distinguer les produits contrefaits des produits originaux. Grâce à une application mobile, les clients peuvent vérifier si un produit est original.

En plus d'aider les utilisateurs à identifier le logo, cette application aide également les marques à lutter contre le piratage des logos. Ce projet peut aider les clients à vérifier l'authenticité d'un produit avant de l'acheter. Il leur évite d'être trompés.

En plus d'aider les consommateurs, ce projet est également utile aux marques dans leur lutte contre le piratage de logos et les produits contrefaits.

CONTEXTE GÉNÉRAL DU PROJET

Cette application a été conçue pour être extrêmement simple et conviviale et peut être utilisée par tout le monde.

1.4 PROBLÉMATIQUE ET SOLUTION PROPOSÉE

1.4.1 PROBLÉMATIQUE

Les logos, parfois aussi appelés marques, sont très importants dans le monde du marketing d'aujourd'hui. Les produits, les entreprises et les différentes ligues de jeu sont souvent reconnus par leurs logos respectifs. La reconnaissance des logos dans les images et les vidéos est la question clé dans un large éventail d'applications, telles que la détection des violations de droits d'auteur, le logo de véhicule pour les systèmes intelligents de contrôle du trafic, la réalité augmentée, le placement contextuel des publicités et autres.

Chaque année, les marques perdent une part importante de leur chiffre d'affaires à cause de marques contrefaites et de contrefaçons non autorisées. De plus, comme ces produits contrefaits sont généralement de qualité inférieure, ils finissent également par nuire à la crédibilité de la marque. Souvent, les consommateurs sont également floués de leur argent durement gagné, car ils finissent par payer une somme exorbitante pour une simple contrefaçon.

Il existe plusieurs fausses marques aujourd'hui et parfois nous ne pouvons pas distinguer à l'œil humain si ils sont vrais ou faux ; par exemple ces marques que l'on voit sur la photo ci-dessous la difficulté de distinguer le réel.



FIGURE 1.2 – Contrefaçon des marques

1.4.2 SOLUTION PROPOSÉE

La solution que nous proposons pour lutter contre la contrefaçon est un modèle d'apprentissage profond entraîné sur un jeu de données pour être capable de différencier les logos originaux et les faux logos afin de détecter le produit contrefait dans une image.

En tenant compte du fait que l'utilisateur simple ne peut pas utiliser ce modèle, nous avons décidé de construire une application mobile afin d'intégrer le modèle dans l'application pour faciliter l'utilisation à l'utilisateur.

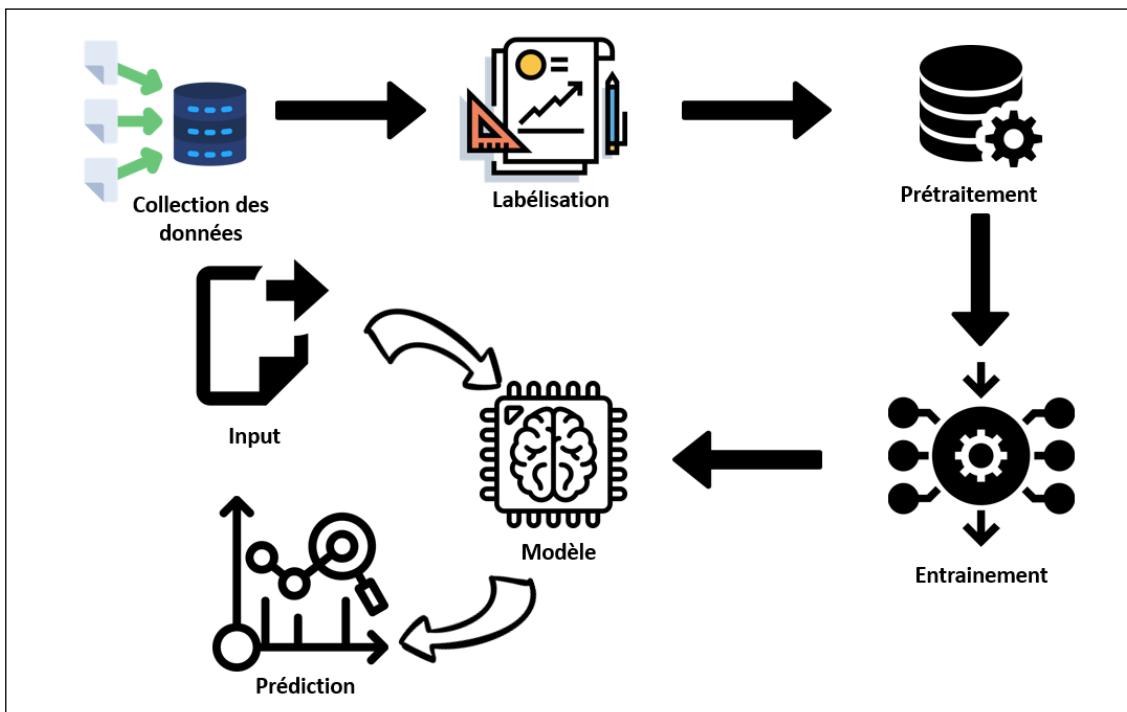


FIGURE 1.3 – Les différentes étapes

1.5 ÉTUDE DE L'EXISTANT

Les travaux précédents, intitulés "Shape matching and object recognition using shape contexts" et "ANSIG-An analytic signature for permutation-invariant two-dimensional shape representation", ont utilisé différents descripteurs globaux de l'image complète du logo, soit en tenant compte des contours du logo, soit en exploitant des descripteurs de forme tels que le contexte de forme. Un algorithme en deux étapes proposé dans "Logo detection based on spatial-spectral

CONTEXTE GÉNÉRAL DU PROJET

saliency and partial spatial context", qui tient compte des contextes locaux des points clés. Ils ont considéré la saillance spatiale-spectrale pour éviter l'impact d'un arrière-plan encombré et accélérer la détection et la localisation du logo. Une métrique appropriée est réalisée parmi les méthodes disponibles, en utilisant deux ensembles de données de fond d'œil publiquement disponibles. En outre, l'article propose une méthode de normalisation complète qui donne des résultats acceptables lorsqu'elle est appliquée à la normalisation des couleurs. L'inconvénient de cette méthode est qu'elle suppose qu'un logo est entièrement visible dans l'image, qu'il n'est pas corrompu par le bruit et qu'il n'est pas soumis à des transformations. De ce fait, elle ne peut pas être appliquée aux images du monde réel. La principale limite de cette approche est la résolution de l'image et leur solution s'est révélée très sensible aux occlusions [1][53].

Une autre solution existante pour la détection et la reconnaissance des logos qui est basée sur la définition des "similarités dépendantes du contexte". Dans ce processus, la correspondance est effectuée en divisant l'image du logo en lignes et en colonnes. Lorsque ce processus est réalisé, la correspondance est très précise. La solution s'est avérée très efficace et répond aux exigences de la détection et de la reconnaissance de logos dans des images du monde réel. La probabilité de réussite de la mise en correspondance et de la détection est élevée [54][55][56]

1.6 ANALYSE DES BESOINS

L'analyse des besoins vise à assurer la compréhension des besoins, des clients et de leurs besoins. L'analyse sera réalisée au niveau des applications.

1.6.1 BESOINS FONCTIONNELS

Être capable de faciliter la détection et la reconnaissance d'objets dans les photos. Pour ce faire, l'application doit permettre les fonctionnalités suivantes :

- Prendre des images en temps réel
- Détection des marques dans les images capturées

1.6.2 BESOINS NON FONCTIONNELS

Les exigences non fonctionnelles décrivent toutes les contraintes auxquelles l'application est soumise pour sa réalisation et son bon fonctionnement.

1.6.2.1 SÉCURITÉ

- L'application ne doit pas être connectée au réseau. L'entier du code source doit être disponible.

1.6.2.2 PORTABILITÉ

- L'application doit pouvoir être exécutée sur les systèmes Android et IOS.

1.6.2.3 PERFORMANCE

- Le délai de traitement doit être suffisamment court pour garantir un accès souhaitable.

1.6.2.4 UTILISATION

- L'application doit être intuitive et facile à utiliser. Les résultats affichés ne doivent pas être complexes ou compliqués à comprendre pour l'utilisateur final.

1.6.2.5 DISPONIBILITÉ

- L'application doit être disponible à tout moment sur le système d'exploitation où elle est installée.

1.7 RÉSULTATS ATTENDUS

Suite à notre étude complète de l'application et à notre travail sur la détection de logos, à la fin de ce projet, l'utilisateur devrait être capable de détecter les marques de la caméra du téléphone en temps réel pour détecter des logos dans une image.

Cette détection doit évidemment être la plus rapide possible pour respecter l'aspect temps réel.

1.8 CONCLUSION

Dans ce chapitre, nous avons essayé de présenter l'objectif général du projet, l'entreprise d'accueil et nous avons détaillé la problématique afin de donner une solution et les résultats attendus du projet.

DU ML AU DEEP LEARNING

Sommaire

2.1	INTRODUCTION	11
2.2	TYPES DE SYSTÈME D'APPRENTISSAGE AUTOMATIQUE	12
2.2.1	APPRENTISSAGE SUPERVISÉ	12
2.2.2	APPRENTISSAGE NON SUPERVISÉ	13
2.2.3	APPRENTISSAGE SEMI-SUPERVISÉ	14
2.2.4	APPRENTISSAGE AVEC RENFORCEMENT	15
2.3	DIFFÉRENCE ENTRE CLASSIFICATION ET RÉGRESSION	16
2.3.1	APPROXIMATION DE FONCTION	16
2.3.2	CLASSIFICATION	17
2.3.3	RÉGRESSION	17
2.4	RÉSEAUX DE NEURONES ARTIFICIELS	17
2.4.1	NEURONE BIOLOGIQUE	17
2.4.2	NEURONE FORMEL	18
2.4.3	APPRENTISSAGE DES RÉSEAUX DE NEURONES	23
2.5	L'APPRENTISSAGE PROFOND (DEEP LEARNING)	26
2.5.1	INTÉRÊT DU DEEP LEARNING	26
2.5.2	ARCHITECTURES DES MODÈLES DE DEEP LEARNING	27
2.5.3	RÉSEAUX DE NEURONES RÉCURRENTS	28
2.5.4	APPLICATION DU DEEP LEARNING	28
2.6	CONCLUSION	29

2.1 INTRODUCTION

Le machine learning, que l'on traduit en français par « apprentissage automatique », ou plus généralement l'intelligence artificielle, dont l'apprentissage automatique est un sous-domaine, évoque la science-fiction pour la plupart des gens. Cependant, l'apprentissage automatique n'est pas un rêve futuriste mais fait déjà partie de la vie quotidienne. En fait, il est déjà utilisé depuis des décennies comme dans les filtres anti-spam dans les années 90 ou dans les jeux vidéo depuis le début des années 2000. Aujourd'hui l'apprentissage automatique est partout, il existe que ce soit à travers la voiture autonome de Google, la reconnaissance vocale de Siri, la détection des visages de Facebook ou le magasin autonome d'Amazon. L'apprentissage automatique révolutionner le monde.

Définition 1.01 :

L'intelligence artificielle est définie par Marvin Minsky, qui fut un pionnier de l'IA, comme : « la construction de programmes informatiques qui effectuent des tâches qui sont, pour le moment, mieux accomplies de manière plus satisfaisante par des êtres humains parce qu'elles requièrent des processus mentaux de haut niveau tels que processus tels que : l'apprentissage perceptif, l'organisation de la mémoire et le raisonnement critique»[2].

Définition 1.02 :

Quant à l'apprentissage automatique, il a été défini en 1959 par Arthur Samuel comme suit : «L'apprentissage automatique est la discipline qui donne aux ordinateurs la capacité d'apprendre sans être explicitement programmés.»[3]

Dans ce chapitre, les types de systèmes d'apprentissage automatique seront approchés. Ensuite, nous aborderons la différence entre la régression et la classification. S'ensuivra des différents algorithmes utilisés dans le cadre de l'apprentissage automatique ainsi qu'une présentation des réseaux de neurones artificiels et enfin, la dernière partie se concentrera sur l'apprentissage profond et nous clôturerons ce chapitre.

2.2 TYPES DE SYSTÈME D'APPRENTISSAGE AUTOMATIQUE

Il existe plusieurs types de système d'apprentissage et cela varie en fonction du type de problème que l'on se pose. Il est alors utile de les classer en différentes catégories. Les systèmes de machine learning peuvent-être classés en fonction de l'importance et de la nature de la supervision qu'ils requièrent durant la phase d'entraînement. On distingue alors quatre grandes catégories : l'apprentissage supervisé, l'apprentissage non supervisé, l'apprentissage semi-supervisé et l'apprentissage avec renforcement.

2.2.1 APPRENTISSAGE SUPERVISÉ

L'apprentissage supervisé consiste en la conception d'un modèle reliant des données d'apprentissage à un ensemble de valeurs de sortie. C'est-à-dire que les données d'entraînement qu'on fournit à l'algorithme comportent les solutions désirées, appelées étiquettes (en anglais, labels). Cette méthode permet donc à l'algorithme d'apprendre en comparant sa sortie réelle avec les sorties enseignées, afin de trouver les erreurs et modifier le modèle en conséquent. L'apprentissage supervisé confère au modèle la possibilité de prédire des valeurs d'étiquette sur des données non étiquetées supplémentaires.

Soit D un ensemble de données, décrit par un ensemble de caractéristiques X , un algorithme d'apprentissage supervisé va trouver une fonction de mapping entre les variables prédictives en entrée X et la variable à prédire Y . La fonction de mapping décrivant la relation entre X et Y s'appelle un modèle de prédiction. Les caractéristiques X peuvent être des valeurs numériques, alphanumériques ou des images.

Un exemple d'utilisation de l'apprentissage supervisé est le filtre anti-spam, l'apprentissage s'effectue à l'aide de nombreux exemples d'e-mails qu'on a étiqueté spam ou normal. A partir de cela, le filtre doit alors être capable de classer de nouveaux e-mails.

Un autre exemple consiste à prédire le prix d'une voiture à partir des valeurs d'un certain nombre d'attributs ou variables qu'on appelle caractéristiques d'une observation ou features en anglais. Ces variables peuvent être le kilométrage, l'âge, la marque, etc. Ils sont également appelés variables explicatives ou prédictives. L'entraînement se fait alors à partir de ces variables et des étiquettes. La catégorie de la variable prédictive Y fait décliner l'apprentissage supervisé en deux sous catégories : la classification et la régression, ces deux concepts seront abordés plus tard[3].

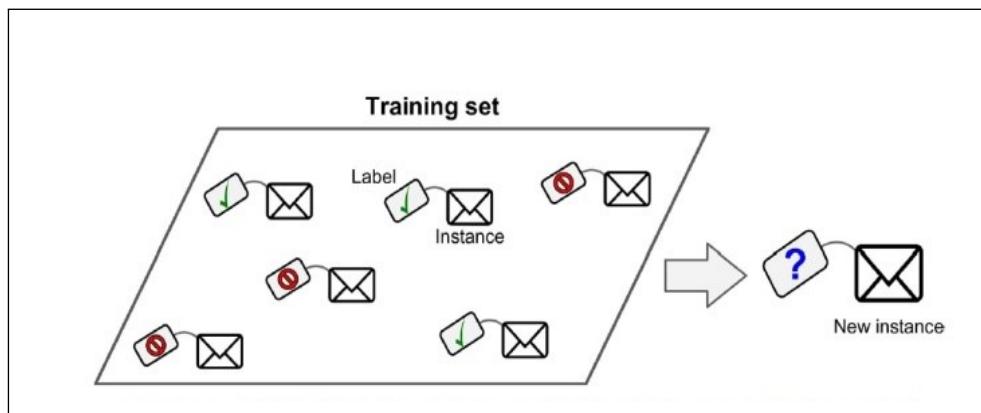


FIGURE 2.1 – Un jeu d'entraînement étiqueté pour un apprentissage supervisé

2.2.2 APPRENTISSAGE NON SUPERVISÉ

L'apprentissage non supervisé consiste en la conception d'un modèle structurant l'information, c'est-à-dire les données d'apprentissage ne sont pas étiquetées. Cette méthode permet donc à l'algorithme de trouver tout seul des points communs parmi les données d'entrée, le système apprend alors sans professeur. Comme l'étiquetage de données requiert beaucoup de temps, les méthodes d'apprentissage utilisant l'apprentissage non supervisé sont particulièrement utiles. L'apprentissage non supervisé peut être utilisé pour la réduction de dimension ou l'extraction de variable. Cette tâche consiste à simplifier les données sans perdre trop d'informations, il pourra ensuite être fourni à un autre algorithme d'apprentissage automatique (tel qu'un algorithme d'apprentissage supervisé). Le kilométrage d'une voiture, par exemple, peut être fortement corrélé à son âge, de sorte que l'algorithme de réduction de dimension les combinera en une seule variable représentant la vétusté de la voiture.

A première vue, on pourrait penser que l'apprentissage non supervisé a peu d'utilité dans les applications de la vraie vie, mais les applications de cette technique sont nombreuses. Les sites comme Amazon, Netflix ou encore Youtube utilisent les algorithmes de partitionnement ou Clustering en anglais pour faire des recommandations de produits ou de films. Il peut être également utilisé pour explorer de larges ensembles de données et de découvrir d'intéressantes relations entre les variables : pour un supermarché par exemple, exécuter une règle d'association sur les journaux de vente permettrait peut-être de découvrir que les personnes achetant de la sauce barbecue et des chips ont aussi tendance à acheter des grillades. Cela permettrait de réorganiser les rayons afin de présenter ces articles à proximité les uns des autres[3] [5] [6].

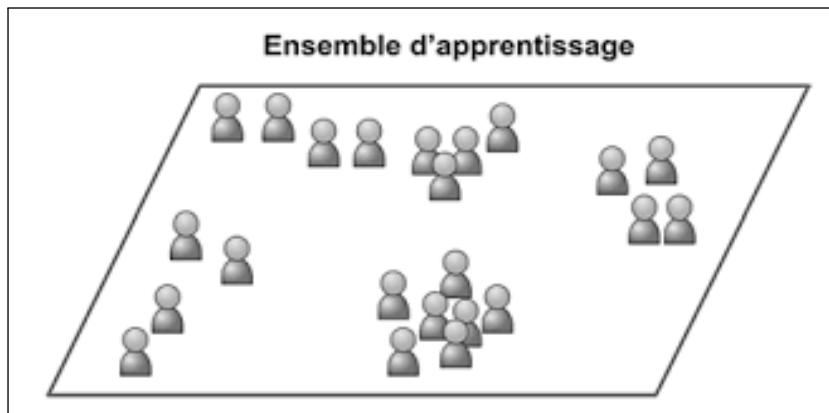


FIGURE 2.2 – Un jeu d'entraînement non étiqueté pour un apprentissage non supervisé

2.2.3 APPRENTISSAGE SEMI-SUPERVISÉ

L'apprentissage semi-supervisé vise à résoudre les problèmes avec relativement peu de données étiquetées et une grande quantité de données non étiquetées. L'apprentissage semi-supervisé réduit également le temps d'étiquetage de grande quantité de données par rapport à un apprentissage supervisé. Il a été démontré que l'utilisation de données non-étiquetées, en combinaison avec des données étiquetées, permet d'améliorer significativement la qualité de l'apprentissage. Ce type d'apprentissage a pour objectif de classer certaines des données non étiquetées à l'aide de l'ensemble d'informations étiquetées. Un exemple illustrant l'utilisation d'un apprentissage semi-supervisé est le service d'hébergement d'image : Google Photos. Une fois avoir téléchargé des photos de famille sur ce service, le système arrive à reconnaître qu'une personne A apparaît sur telle ou telle photos et qu'une personne B sur telle autres. Cela est

dû à la partie non supervisée de l'algorithme. Une fois que vous aviez identifié ces personnes, juste une étiquette par personne, le système sera capable de nommer les personnes figurant sur chaque photo, ce qui est utile pour des recherches ultérieures[3].

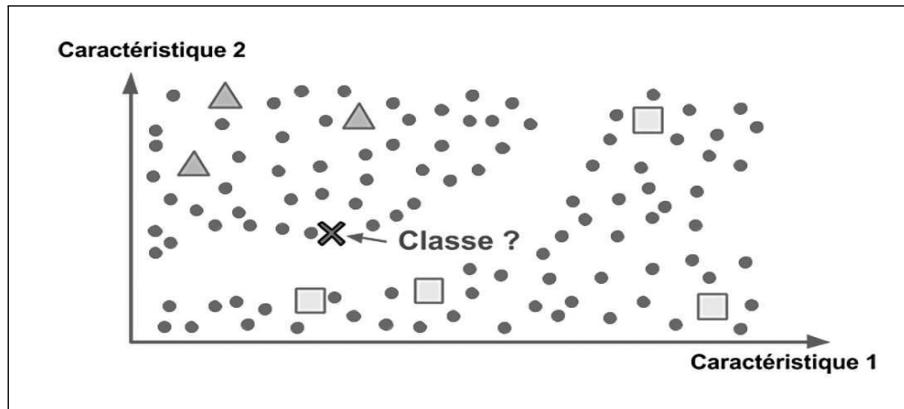


FIGURE 2.3 – Apprentissage semi-supervisé

2.2.4 APPRENTISSAGE AVEC RENFORCEMENT

L'apprentissage par renforcement est très différent des types d'apprentissage vus jusqu'ici. Il consiste à apprendre, à partir d'expériences successives, ce qu'il convient de faire de façon à trouver la meilleure solution. Le système d'apprentissage, que l'on appelle ici « agent », interagit avec l'environnement, en sélectionnant et accomplissant des actions afin de trouver la solution optimale et obtenir en retour des récompenses. L'agent essaie plusieurs solutions, on parle d'« exploration », observe la réaction de l'environnement et adapte son comportement, c'est-à-dire les variables pour trouver la meilleure stratégie. Pour ce type d'apprentissage, les données d'entraînement proviennent directement de l'environnement. L'apprentissage par renforcement peut être utilisé pour apprendre à un robot à marcher, ou à un programme à jouer, comme AlphaGo de DeepMind qui a vaincu l'un des meilleurs joueurs de go au niveau mondial. En 2018, une startup issue des travaux de l'université de Cambridge a formé une intelligence artificielle à la conduite automobile. Au bout de seulement vingt minutes, l'IA est parvenue à savoir comment maintenir la voiture sur sa voie de circulation. Durant cette expérience un chauffeur était présent à bord de la voiture pour corriger les écarts de trajectoire causés par le logiciel, en arrêtant la voiture. Le programme a rapidement progressé en suivant une logique

pénalité-récompense, en l'occurrence, respectivement, une intervention humaine et la distance maximale parcourue sans correction par le conducteur[3] [7]. [8]

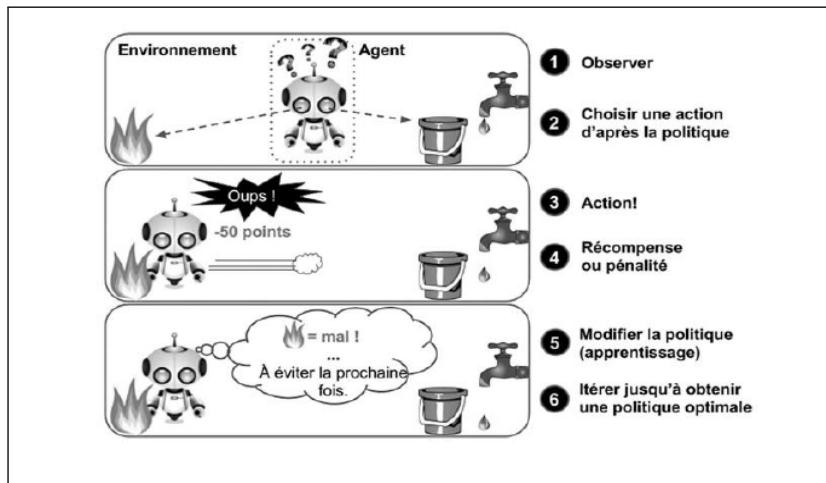


FIGURE 2.4 – Apprentissage par renforcement

2.3 DIFFÉRENCE ENTRE CLASSIFICATION ET RÉGRESSION

Il existe une grande différence entre les problèmes de classification et de régression. La classification consiste à prédire une étiquette et la régression à prédire une quantité. Avant de présenter ces deux concepts, il sera utile de comprendre la notion d'approximation de fonction.

2.3.1 APPROXIMATION DE FONCTION

Le terme de modélisation prédictive « en anglais : Predictive Modelling » fait référence à un ensemble de méthodes qui permettent d'analyser et d'interpréter des données définies afin de faire une prédiction sur des données futures. La modélisation prédictive peut être décrite comme un problème mathématique consistant à approximer une fonction de correspondance f entre les variables prédictives d'entrée X et la variable à prédire Y . C'est ce qu'on appelle un problème d'approximation de fonction. En général, toutes les tâches d'approximation de fonction peuvent être divisées en tâches de classification et tâches de régression[9].

2.3.2 CLASSIFICATION

Un algorithme de classification est utilisé lorsque la variable Y à prédire est discrète. Exemple : Classification des e-mails à l'aide du filtre anti-spam décrit ci-dessus. La classification peut avoir des variables d'entrée à valeur réelle ou à valeur discrète. Les modèles de classification prédisent généralement des valeurs continues en tant que probabilités d'appartenance à chaque classe de sortie. Les probabilités prédites peuvent être converties en valeurs de classe en choisissant l'étiquette de classe avec la probabilité la plus élevée[9].

2.3.3 RÉGRESSION

L'algorithme de régression est utilisé lorsque la variable Y à prédire est continue. Comme c'est le cas avec la prévision du prix d'une voiture. Les problèmes de régression nécessitent de prévoir des quantités. Il peut avoir des variables d'entrée à valeur réelle ou à valeur discrète. Notez que certains algorithmes ont le mot "régression" dans leur nom, comme la régression linéaire et la régression logistique, ce qui peut prêter à confusion car la régression linéaire est un algorithme de régression et la régression logistique est un algorithme de classification. Certains algorithmes peuvent être utilisés pour la classification et la régression avec des modifications mineures, tels que les arbres de décision et les réseaux de neurones artificiels[9].

2.4 RÉSEAUX DE NEURONES ARTIFICIELS

Les réseaux de neurones sont au cœur des progrès récents de l'apprentissage automatique. On ne peut pas parler d'apprentissage automatique sans parler des réseaux de neurones, un ensemble d'algorithmes dont le fonctionnement est inspiré des neurones biologiques.

2.4.1 NEURONE BIOLOGIQUE

Le cerveau est un organe fascinant, et on sait depuis longtemps que notre capacité à penser en découle. Les cellules les plus importantes du cortex cérébral sont les neurones, et leur nombre

chez l'homme atteint 100 milliards. Ils se composent de trois parties de base : le corps cellulaire, les dendrites et les axones. Le corps cellulaire contient le noyau du neurone et son rôle est d'effectuer les transformations biochimiques nécessaires aux enzymes de synthèse qui assurent la vie du neurone. Chaque neurone possède des dendrites, qui sont les principaux récepteurs du neurone pour recevoir les signaux qui l'atteignent. D'autre part, les axones agissent comme des transporteurs pour les signaux émis par les neurones. Les neurones sont connectés les uns aux autres selon une distribution spatiale complexe. La connexion physique entre deux neurones est établie par des synapses[10]. [11]

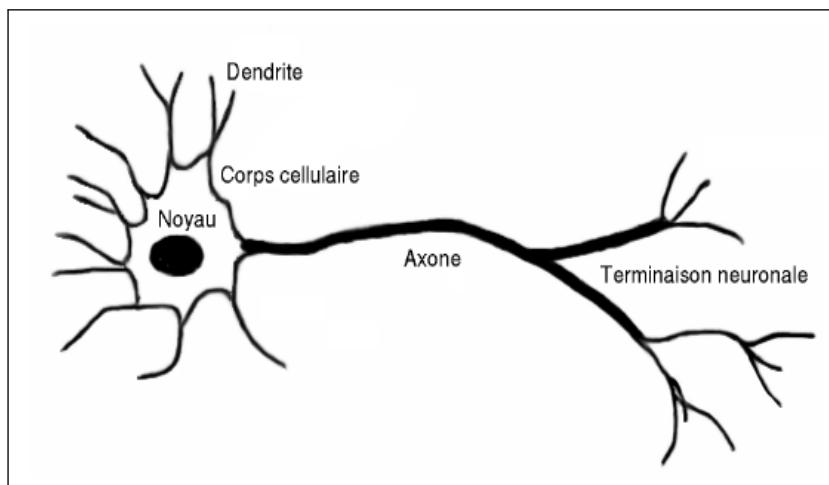


FIGURE 2.5 – Neurone biologique

2.4.2 NEURONE FORMEL

Les neurones formels ou artificiels sont conçus pour reprendre les fonctions des neurones biologiques. Les neurones reçoivent des entrées et fournissent des sorties grâce à des propriétés différentes. Dans les neurones formels, les poids permettent de modifier l'importance de certaines entrées par rapport à d'autres entrées. Les fonctions agrégées peuvent obtenir des valeurs uniques à partir de l'entrée et des poids correspondants. Les seuils permettent aux neurones de savoir quand agir ou non. Enfin, la fonction d'activation attribue à chaque valeur agrégée une valeur de sortie unique en fonction du seuil.

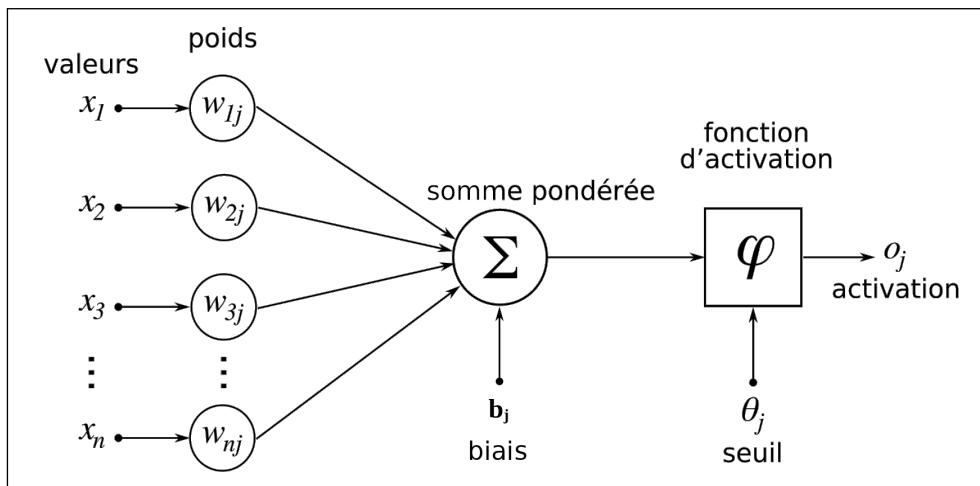


FIGURE 2.6 – Neurone formel

2.4.2.1 FONCTION D'AGRÉGATION

Il existe plusieurs types de fonctions d'agrégation, mais les plus courantes sont : les sommes pondérées et les calculs de distance. L'objectif est d'attribuer une valeur à toutes les entrées et à tous les poids.

La somme pondérée consiste à calculer la somme de toutes les entrées multipliées par leur poids c'est-à-dire :

$$\sum_{i=1}^n E_i * W_i \quad (2.1)$$

Le calcul de la distance consiste à comparer l'entrée au poids (qui est l'entrée attendue par le neurone) et à calculer la distance entre les deux. Dans ce cas, la distance est donnée par :

$$\sum_{i=1}^n (E_i * W_i)^2 \quad (2.2)$$

2.4.2.2 FONCTION D'ACTIVATION

Il existe différentes fonctions d'activation. Une fois que le neurone a obtenu la valeur donnée par la fonction d'agrégation, il la compare à la valeur seuil et utilise la fonction d'activation pour déterminer la sortie. Voici les fonctions d'activation les plus couramment utilisées :

Fonction de Heaviside : Il s'agit d'une fonction qui peut obtenir une sortie binaire, 1 si la valeur d'agrégation calculée est supérieure à un seuil, et 0 sinon.

$$H(x) = \begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{si } x \geq 0 \end{cases} \quad (2.3)$$

Elle est principalement utilisée pour résoudre des problèmes de classification en indiquant si un objet appartient à une classe particulière ou non.

La fonction sigmoïde est définie par la relation mathématique suivante :

$$f(x) = \frac{1}{1 + e^{-x}} \quad (2.4)$$

Elle est dérivable et donc contrairement à la fonction de Heaviside, permet de savoir vers quelle direction aller pour améliorer les résultats.

La fonction softmax : elle est utilisée pour de la classification multi-classe. Elle prend en entrée un vecteur $z = (z_1, \dots, z_k)$ de k nombre réels et en sort un vecteur $\sigma(z)$ de K nombres réels strictement positifs de somme 1. Elle est définie par la relation mathématique suivante : pour tout j

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad \text{for } i = 1, 2, \dots, K \quad (2.5)$$

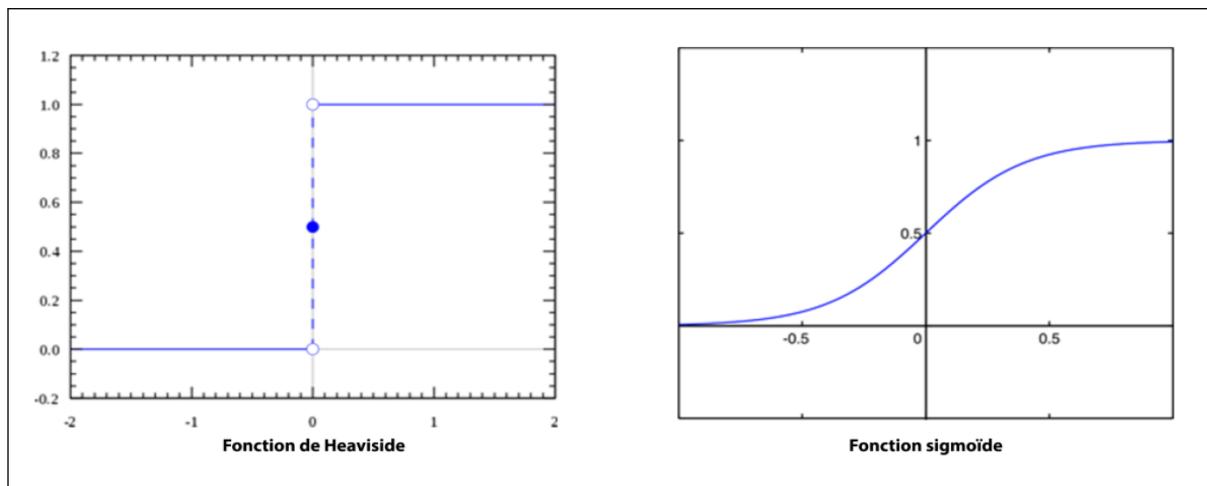


FIGURE 2.7 – Fonction d'activation

2.4.2.3 POIDS ET SEUILS

Les poids liant les neurones à leurs entrées et leurs seuils servent à distinguer les neurones. Un poids est attribué à chaque entrée, ce qui permet de modifier les entrées qui sont plus importantes que les autres. Le neurone doit fonctionner en fonction des seuils ou des biais. Le calcul des seuils et des poids pour des fonctions complexes est extrêmement difficile. Ensuite, l'apprentissage consiste à déterminer leurs valeurs idéales afin de produire le résultat escompté[12].

2.4.2.4 PERCEPTRON

Frank Rosenblatt a créé le perceptron en 1955. Il est constitué d'une première couche d'unités qui lisent les données. On peut inclure une unité de biais qui est toujours active. Une seule unité de sortie est connectée à ces unités. Les résultats d'une classification multi-classes peuvent cependant varier.

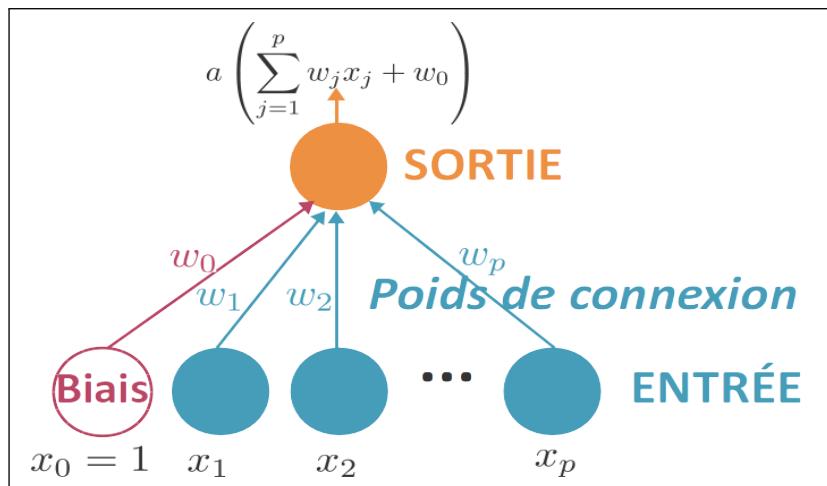


FIGURE 2.8 – Perceptron à une unité de sortie

Ici x_0 représente le biais, (x_1, x_2, \dots, x_p) et (w_1, w_2, \dots, w_p) représente respectivement les entrées et les poids correspondants à chaque entrée, $a(\sum_{j=1}^p w_j x_j + w_0)$ la fonction d'activation.

Le réseau aura autant de neurones de sortie qu'il y a de classes dans les problèmes de classification multi-classes. Chacune de ces unités sera donc liée à chaque unité d'entrée. Dans ce cas, c'est la sortie ayant la valeur la plus élevée qui est prise en compte.

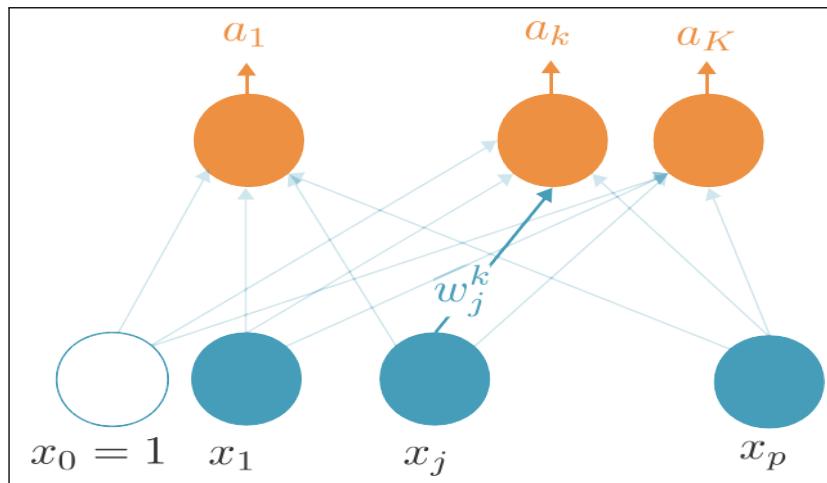


FIGURE 2.9 – Perceptron à plusieurs unités de sortie

Dans cet exemple K représente le nombre de classe.

Le perceptron est facile à mettre en œuvre, mais son principal défaut est qu'il ne peut résoudre que des problèmes linéairement séparables. La fonction d'activation (sigmoïde ou Heaviside) utilisée possède un seuil qui divise l'espace en deux régions.

2.4.2.5 RÉSEAU DE NEURONES A PROPAGATION AVANT (feed-forward)

Il est possible d'aller au-delà des capacités du perceptron en utilisant des réseaux de neurones à propagation directe. Ils aident à traiter les problèmes au-delà de ceux qui peuvent être résolus de manière linéaire. Entre les couches d'entrée et de sortie de ces réseaux, il y a 20 couches neuronales mises en cache. Selon cette architecture, chaque neurone de chaque couche est connecté à chaque neurone de la couche qui le précède. Dans une disposition feed-forward ou non-bouclée, les neurones ne sont connectés que dans une seule direction : les signaux entrants passent par les unités cachées sur leur chemin vers les unités de sortie potentielles.

L'exemple le plus connu de réseau neurones feed-forward est le réseau MPL ou Multi-layer perceptron[12]. [13]

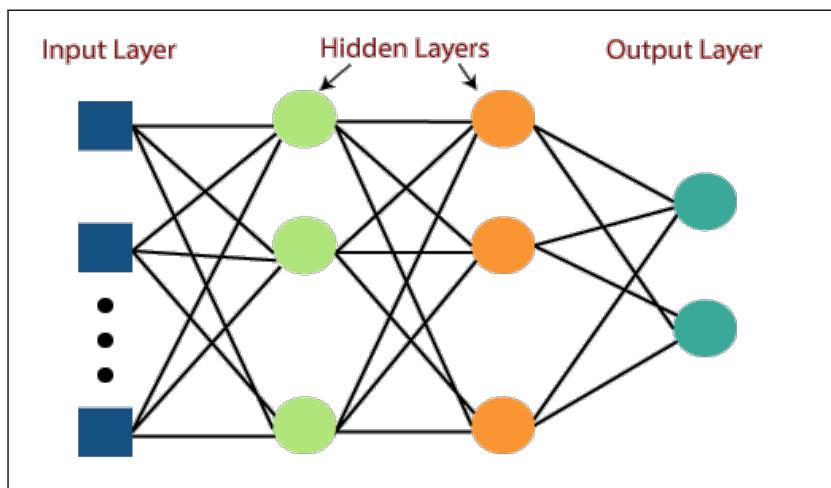


FIGURE 2.10 – Perceptron multicouche à deux couches intermédiaires

2.4.3 APPRENTISSAGE DES RÉSEAUX DE NEURONES

L'apprentissage est très important dans l'utilisation d'un réseau de neurone. Il consiste à calculer les valeurs des poids et des seuils du réseau.

2.4.3.1 APPRENTISSAGE DU PERCEPTRON

Un perceptron est entraîné en réduisant l'erreur de prédiction sur l'ensemble des données d'entrée. La méthode de cet apprentissage est itérative. Les poids de connexion doivent être modifiés après chaque observation afin de réduire l'erreur de prédiction actuelle du perceptron.

L'algorithme du gradient permet l'apprentissage des réseaux monocouches de type perceptron.

Soit les n Observations x^1, x^2, \dots, x^n qui sont observées séquentiellement, l'une après l'autre. Comme dans la régression linéaire, on commence par choisir aléatoirement des valeurs initiales $w_0^{(0)}, w_1^{(0)}, \dots, w_p^{(0)}$ pour les poids de connections. Après chaque observation $(x^{(i)}, y^{(i)})$ nous allons à chacun des poids la règle de mise à jour suivante :

$$w_j^{t+1} = w_j^t - \eta \frac{\partial \text{Erreur}(f(x^i), y^i)}{\partial w_j} \quad (2.6)$$

Ici x^i représente l'entrée de l'observation, y^i sa sortie attendue et η le taux d'apprentissage ou vitesse d'apprentissage (ou learning rate en anglais).

La méthode peut être exécutée sur l'ensemble de données d'apprentissage tout au long de l'itération, qui se termine lorsque le nombre prédéterminé d'itérations a été atteint ou que l'algorithme converge.

2.4.3.2 APPRENTISSAGE D'UN PERCEPTRON MULTICOUCHE

Le processus d'apprentissage d'un MLP (Multiple Layers Perceptron) utilise un algorithme de rétropropagation (ou Backpropagation en anglais), qui est basé sur la descente de gradient. On va chercher à minimiser la fonction d'erreur du réseau de neurones.

L'algorithme se présente comme suit :

Étape 1 : Initialisation de tous les poids à des valeurs aléatoires.

Étape 2 : Normalisation des données d'entraînement.

Étape 3 : Permuter aléatoirement les données. d'entraînement.

Étape 4 : Pour Chaque donnée d’entraînement n :

- Calculer les sorties observées en propageant les entrées vers l’avant ;
- Ajuster les poids en rétropropageant l’erreur observée :

$$w_{ji}(n) = w_{ji}(n - 1) + \Delta W_{ji}(n) = W_{ji}(n)\eta\delta_j(n)y_i(n) \quad (2.7)$$

Où $\delta_j(n)$ est le gradient local défini par :

$$\delta_j(n) = \begin{cases} \delta_j(n) = e_j(n)y_j(n)[1 - y_j(n)] & \text{si } j \in \text{couche de sortie} \\ \delta_j(n) = y_j(n)[1 - y_j(n)][\sum_{k=c} \delta_j(n)w_{kj}(n)] & \text{si } j \in \text{couche de caché} \end{cases} \quad (2.8)$$

η représente le taux d’apprentissage, $y_j(n)$ la sortie du neurone i sur la chouche précédente ou l’entrée i quand celle-ci n’existe pas.

Étape 5 : répéter les étapes 3 et 4 jusqu’à un nombre maximum d’itérations ou jusqu’à ce que la racine de l’erreur quadratique moyenne soit inférieure à un certain seuil.

Remarque :

Les ensembles de données utilisés dans l’apprentissage automatique ont généralement des ordres de grandeur distincts. La normalisation des données limite la gamme des valeurs numériques possibles. La normalisation peut être réalisée en utilisant la méthode de mise à l’échelle Min-Max. La formule suivante est utilisée pour modifier les données :

$$x_{normalize} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (2.9)$$

Avec

x_{min} et x_{max} : la plus petite et la plus grande valeur observée pour la caractéristique X.

x : la valeur de la caractéristique que nous voulons normaliser. La vitesse de convergence peut être augmentée en permutant aléatoirement les données ; sinon, si nous avons de nombreux cas de la même classe dans une rangée, la convergence sera assez lente[5][13][14].

2.5 L'APPRENTISSAGE PROFOND (DEEP LEARNING)

Le terme deep learning, qui se traduit en français par apprentissage profond est très populaire de nos jours. Cependant, les principes de cette méthode remontent à la fin des années 80. L'intérêt soudain pour l'apprentissage profond ces dernières années a été motivé par plusieurs critères. Tout d'abord, les GPU (Graphical Processing Units) capables de réaliser plus de mille milliards d'opérations par seconde à faible coût. Deuxièmement, des expériences menées simultanément chez Microsoft, Google et IBM (International Business Machines), ont montré que les réseaux profonds peuvent réduire de moitié le taux d'erreur des systèmes de reconnaissance vocale. Troisièmement, plusieurs records de reconnaissance d'images ont été battus par des réseaux convolutifs, dont un lors du concours de reconnaissance d'objets "ImageNet"[15].

Définition 1.03 :

Le deep learning repose sur l'idée des réseaux neuronaux artificiels et vise à traiter de grandes quantités de données en ajoutant des couches au réseau. Pour Yann LeCun, l'un des pionniers du deep learning et lauréat du prix Turing 2019, "la technologie du deep learning apprend comment représenter le monde. C'est-à-dire comment la machine va représenter la parole ou les images, par exemple"[16].

2.5.1 INTÉRÊT DU DEEP LEARNING

Les algorithmes traditionnels de machine learning fonctionnent très bien dans certaines situations. Ils semblent toutefois inefficaces lorsqu'il s'agit des principaux problèmes d'apprentissage profond, comme la reconnaissance de la parole et des objets.

L'étape d'extraction des caractères est la principale distinction entre les algorithmes de machine learning traditionnels et le deep learning . Les caractéristiques qui doivent être identifiées pour l'apprentissage automatique traditionnel dépendent des compétences humaines. Cette pratique est extrêmement difficile et nécessite un spécialiste du domaine ; parfois, les caractéristiques distinctives sont même impossibles à reconnaître pour les humains. Afin de résoudre ce problème, l'apprentissage profond utilise plusieurs couches de réseaux neuronaux.

Les premières couches extrairont des caractéristiques simples (la présence de contours), que les couches suivantes combineront pour créer des concepts de plus en plus abstraits et difficiles à comprendre tels que assemblages de contours en motifs, de motifs en parties d'objets, de parties d'objets en objets, etc.

Un autre avantage du deep learning est qu'il s'adapte bien, fournit de grandes quantités de données et produit de meilleures performances. Le "plateau de performance" est une limite supérieure à la quantité de données que peuvent recevoir les algorithmes ML traditionnels[15][17].

2.5.2 ARCHITECTURES DES MODÈLES DE DEEP LEARNING

Il existe différents types de réseaux neuronaux profonds. Les modèles les plus populaires sont les réseaux de neurones à convolution et les réseaux de neurones récurrents.

2.5.2.1 RÉSEAUX DE NEURONE A CONVOLUTION

Une architecture profonde particulièrement répandue est le réseau à convolution. Désignés par l'acronyme CNN, de l'anglais Convolutional Neural Network. Les premières versions ont été développées par Yann LeCun aux Bell Laboratories vers 1989. Les réseaux convolutifs sont une forme particulière de réseau neuronal multicouche dont l'architecture des connexions est inspirée du cortex visuel des animaux. Par exemple, chaque élément n'est connecté qu'à un petit nombre d'éléments voisins dans la couche précédente. Les CNN sont tout particulièrement utiles pour les applications basées sur la reconnaissance d'objets et la vision par ordinateur.

Les couches présentes dans un CNN sont de l'ordre de plusieurs dizaines, voire plusieurs centaines de couches. Toutes ces couches effectuent des opérations qui modifient les données dans l'objectif d'apprendre les caractéristiques spécifiques à ces données. On distingue trois principaux types de couches :

- la couche convolution qui fait passer les images d'entrée par un ensemble de filtres convolutifs, chacun de ces filtres activant certaines caractéristiques des images ;

- la couche ReLU (Rectified linear unit) permet d'accélérer et d'optimiser l'apprentissage en remplaçant toutes les valeurs négatives reçues en entrées par des zéros.
Elle joue le rôle de fonction d'activation ;
- la couche de pooling qui consiste à réduire la taille des images, tout en préservant leurs caractéristiques importantes.

Après l'apprentissage des caractéristiques, la dernière couche du CNN est un réseau complètement connectée (perceptron ou bien un MLP).

Les différentes couches d'un CNN seront expliquées plus en détails dans un autre chapitre.

2.5.3 RÉSEAUX DE NEURONES RÉCURRENTS

Une autre classe d'architecture autre que les CNN est également revenue au goût du jour, les réseaux de neurones récurrents ou recurrent neural network (RNN) en anglais. Ils ont également été créés dans les années 1980. Contrairement aux réseaux de neurones à propagation avant, dans un RNN, il existe non seulement des liens d'une couche vers les suivantes, mais aussi vers les couches précédentes. Ces architectures sont particulièrement adaptées au traitement de signaux séquentiels, tels que le texte. Une autre façon de penser les RNN est qu'ils ont une « mémoire » qui capture l'information sur ce qui a été calculé jusqu'ici. En théorie, les RNN peuvent utiliser des informations dans des séquences arbitrairement longues, mais dans la pratique, on les limite à regarder seulement quelques étapes en arrière.[17][18][19]

2.5.4 APPLICATION DU DEEP LEARNING

Les applications du deep learning sont très nombreuses et ne sont plus considérées comme de la science-fiction. Les géants du Net comme Google, Facebook et Amazon ne cessent d'investir fortement dans ce domaine.

En 2014, Google a acheté la société anglaise DeepMind, dont l'objectif est actuellement de « comprendre ce qu'est l'intelligence ». DeepMind est célèbre pour son programme AlphaGo, qui a battu le champion du monde de go. Un des grands projets de Google, la Google Car

renommé en 2016 par le projet Waymo, fonctionne à base de reconnaissance et de détection d'objet utilisant du deep learning pour la reconnaissance des panneaux pour savoir à quelle vitesse rouler, quelles voies ne pas prendre, quand s'arrêter, reconnaissance des lignes blanches etc. Aujourd'hui, ces voitures sont déjà utilisées en tant que robot taxi, c'est ainsi que WAL-MART (le leader mondial des supermarchés) et AVIS l'utilisent pour transporter des clients jusqu'à leur magasins.

Facebook, quant à lui, est un très grand acteur de l'intelligence artificielle, plus précisément le deep learning. Il choisit les messages qu'il affiche en utilisant un moteur de recommandation. Récemment, Facebook a mis en place un moteur d'intelligence artificielle pour détecter les tendances suicidaires. Facebook utilise également le deep learning pour l'analyse de scène dans les images pour masquer les images à caractère violent ou inapproprié [20].

Amazon utilise l'intelligence artificielle dans son moteur de recommandation, nommé Echo, et dans ses assistants basés sur son système de reconnaissance vocale Alexa. Mais une des grandes révolutions apportée par Amazon, est le fameux Amazon Go. Amazon Go est un magasin sans caisse, munie de centaine de caméras et des capteurs sur des étagères. Le magasin permet de se saisir les produits et de les ajouter au panier virtuel d'une l'application sur le smartphone du client qui, en retraversant les portiques, sera facturé et débité [21].

Le deep learning est aujourd'hui présent dans tous les domaines, et continuera sans doute à avoir une place importante dans les décennies à venir[22].

2.6 CONCLUSION

Le machine learning ne date pas d'aujourd'hui, les différents algorithmes utilisé dans le ML ont vu le jour durant le XXe siècle. Durant des années, les réseaux de neurones ont été mis aux oubliettes. L'émergence récent du deep learning, qui est une branche du machine learning, est due à l'évolution des GPU ainsi qu'à la disponibilité des données en grande quantité. On peut dire que cette technologie a été en avance sur son temps. Les applications du deep learning sont nombreuses et touche tous les domaines, que ce soit pour le traitement de la voix ou la

vision par ordinateur. Le chapitre suivant se focalisera sur la vision par ordinateur car elle est essentielle à ce projet.

LES MODÈLES DE DÉTECTION

Sommaire

3.1	INTRODUCTION	32
3.2	RESEAUX DE NEURONES CONVOLUTION (CNN)	32
3.2.1	COUCHE DE CONVOLUTION	32
3.2.2	COUCHE D'ÉCHANTILLONNAGE	33
3.2.3	COUCHE COMPLÈTENT CONNECTÉE	34
3.3	LES MODÈLES DE DÉTECTION PAR CNN	34
3.3.1	R-CNN	36
3.3.2	FAST R-CNN	37
3.3.3	FASTER R-CNN	38
3.3.4	MASK R-CNN	38
3.3.5	SSD (SINGLE SHOT MULTIBOX DETECTOR)	39
3.3.6	YOLO (YOU ONLY LOOK ONCE)	40
3.4	MÉSUSER LA PERFORMANCE D'UN MODÈLE DE DÉTECTION	57
3.5	CONCLUSION	58

3.1 INTRODUCTION

La détection d'objets est une technique de vision par ordinateur qui a connu un changement révolutionnaire rapide, cette technique fait la combinaison de la classification et de la localisation d'objets. En effet, un des thèmes les plus difficiles dans le domaine de la vision par ordinateur.

Un système de détection d'objet peut détecter, localiser et tracer l'objet (déterminer où se trouvent les objets dans une image donnée) et identifie la catégorie de cette dernière (personne, table, chaise, etc.). L'emplacement est indiqué en dessinant une boîte de délimitation (boite englobante) autour de l'objet, La capacité à localiser l'objet dans une image définit la performance de l'algorithme utilisé pour la détection.

Il existe différents types d'algorithmes de détection d'objets, certains sont des techniques traditionnelles et d'autres des techniques modernes développées récemment. Ces dernières diffèrent les unes des autres en fonction de leur précision, de leur vitesse, des ressources matérielles requises, et même le nombre de classes prise en charge.

3.2 RESEAUX DE NEURONES CONVOLUTION (CNN)

3.2.1 COUCHE DE CONVOLUTION

Un réseau de neurones convolutif est un réseau de neurones qui utilise une opération mathématique qui s'appelle convolution ou produit de convolution. Il s'agit d'une opération linéaire. Chaque réseau de neurones convolutif contient au moins une couche de convolution. Une couche de convolution est caractérisée par :

- Les dimensions des noyaux de convolution, généralement une convolution à une dimension égale à 2 avec des noyaux carrés
- Le nombre des filtres de convolution C , c'est le nombre de cartes d'activations, ou cartes de caractéristiques, en sortie de la couche. Ces cartes sont représentées sous

la forme de tenseurs de dimension 3 (H, W, C) avec H la hauteur des cartes, W la largeur et C le nombre de canaux.

- Le pas de convolution –ou stride-s. C'est le pas de décalage du noyau de convolution à chaque calcul.
- Le padding p . C'est le paramètre permettant de dépasser la taille de l'image pour appliquer la convolution en ajoutant des pixels autour de l'image.

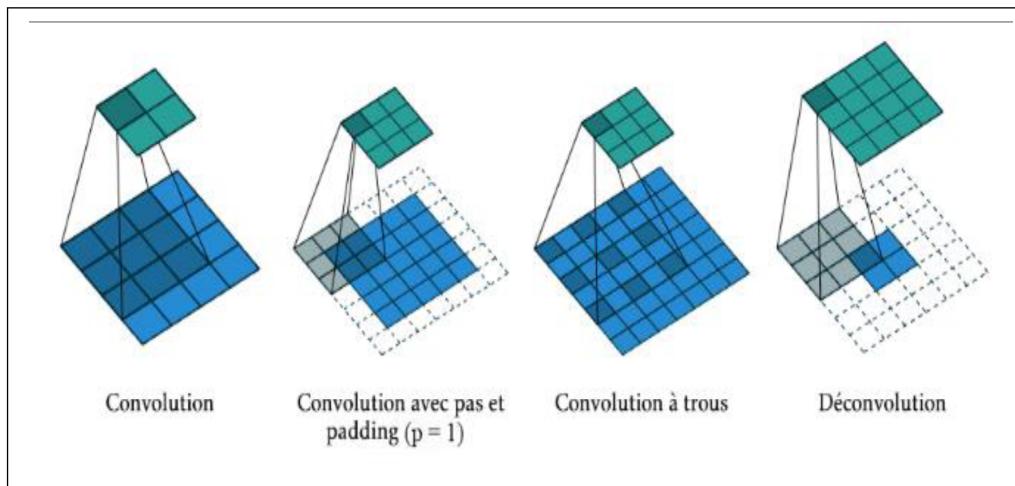


FIGURE 3.1 – Différents types de convolution

3.2.2 COUCHE D'ÉCHANTILLONNAGE

Semblable à la couche de convolution, la couche d'échantillonnage est chargée de réduire la taille spatiale des cartes de caractéristiques, mais elle conserve les informations les plus importantes. Il existe différents types d'échantillonnage dont l'échantillonnage maximum ou Max Pooling, l'échantillonnage moyen ou Average Pooling.

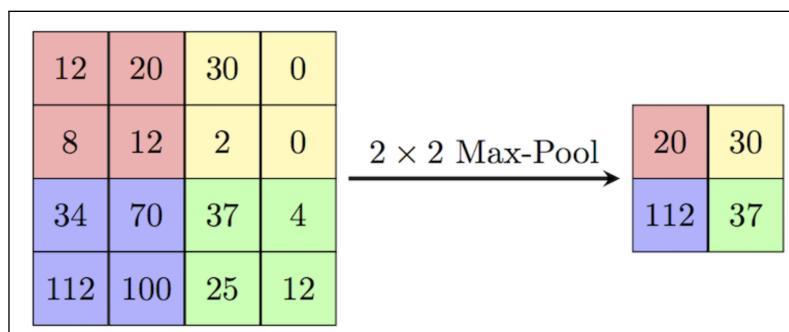


FIGURE 3.2 – Exemple d'une opération de pooling de taille 2x2

3.2.3 COUCHE COMPLÈTENT CONNECTÉE

La couche complètement connectée est un Perceptron multi-couches traditionnel qui utilise une fonction d'activation (par exemple softmax) sur le vecteur de sortie afin d'ajouter la non-linéarité. Le terme « complètement connecté » implique que chaque neurone de la couche précédente est connecté à chaque neurone de la couche suivante. Leurs activations peuvent donc être calculées avec une multiplication matricielle suivie d'un offset de biais. L'équation de softmax écrit sous la forme :

$$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \quad (3.1)$$

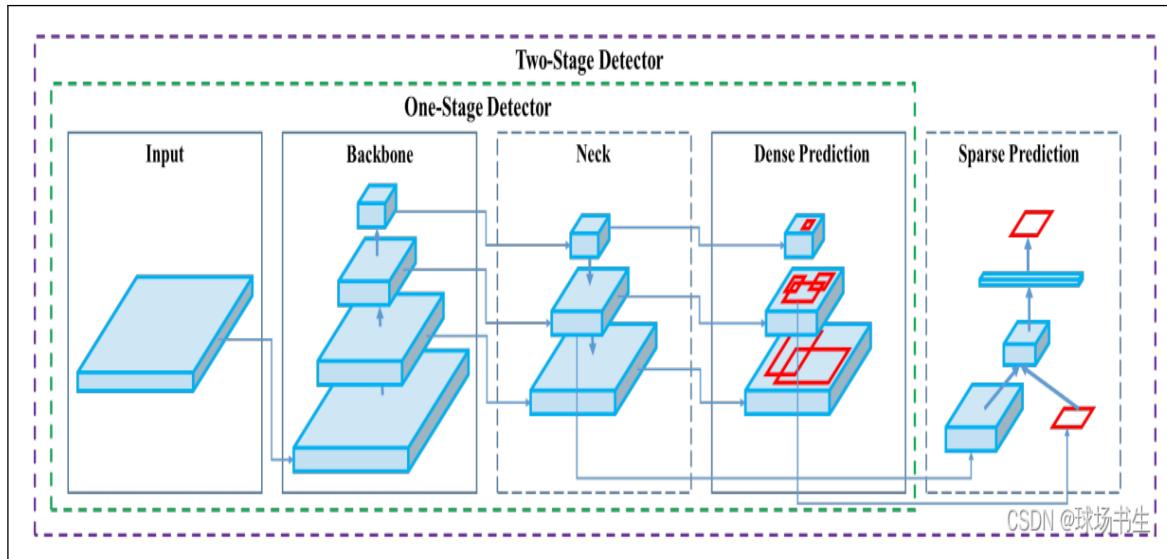
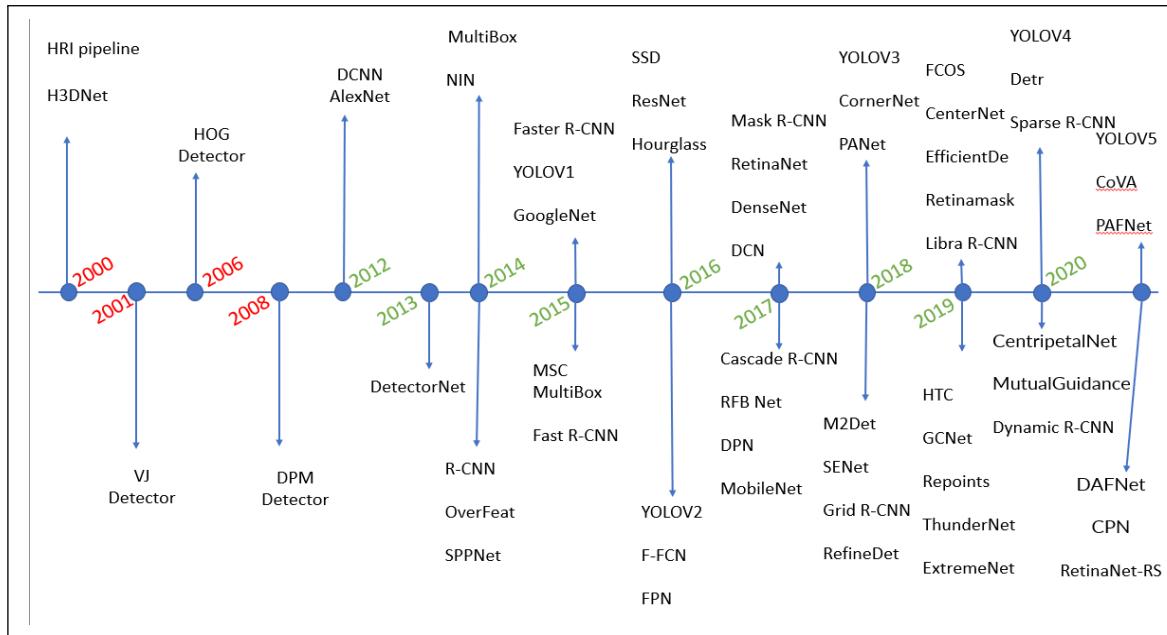
3.3 LES MODÈLES DE DÉTECTION PAR CNN

Les architectures CNN sont capables d'apprendre des caractéristiques plus complexes. Il existe deux types de modèles de détection d'objets par les architectures CNN. Le premier type de détection en deux coups est basé sur la proposition de région et comprend des modèles tels que RCNN, SPP-NET, FRCNN, Faster RCNN et le second type la détection à un coup est basé sur la régression et comprend MultiBox, AttentionNet, G-CNN, YOLO, SSD...etc [34]

- **Détection en deux coup :** Comme son nom l'indique, cette méthode comporte deux étapes. La première est la proposition de régions, puis, dans la deuxième étape, la classification de ces régions et le raffinement de la prédiction de l'emplacement ont lieu.
- Au contraire, **la détection à un coup** saute l'étape de la proposition de région et produit la localisation finale et la prédiction du contenu en une seule fois.[35]

Il existe plusieurs modèles de détection d'objet, nous les présentons dans le schéma suivant, dans un ordre chronologique :

LES MODÈLES DE DÉTECTION



- **Entrée :** Image, correctifs, Pyramide d'images.
- **Backbones :** VGG16, ResNet-50, SpineNet, EfficientNet-B0/B7, CSPResNeXt50, CSPDarknet53.
- **Neck :**
 - **Blocs additionnels :** SPP, ASPP, RFB, SAM.
 - **Blocs agrégation de chemins :** FPN, PAN, NAS-FPN, Fully-connected FPN, BiFPN, ASFF, SFAM
- **Heads :**

- **Prédiction dense (une étape) :** RPN, SSD, YOLO, RetinaNet, CornerNet, CenterNet, MatrixNet, FCOS.
- **Prédiction éparse (en deux étapes) :** Faster R-CNN, R-FCN, Mask R-CNN, RepPoints[36].

3.3.1 R-CNN

Le modèle de détection d'objets R-CNN a été proposé par Ross Girshick en 2014 [37], ce modèle comporte trois modules :

- **Génération de propositions régionales** : indépendantes de la catégorie, qui définissent l'ensemble des détections candidates disponibles pour notre détecteur.
- **Extraction de caractéristiques** : le deuxième module est un grand réseau neuronal convolutionnel qui extrait un vecteur caractéristique de longueur fixe de chaque région.
- **Classification et localisation** : Le troisième module est un ensemble de SVM linéaires spécifiques à chaque classe.

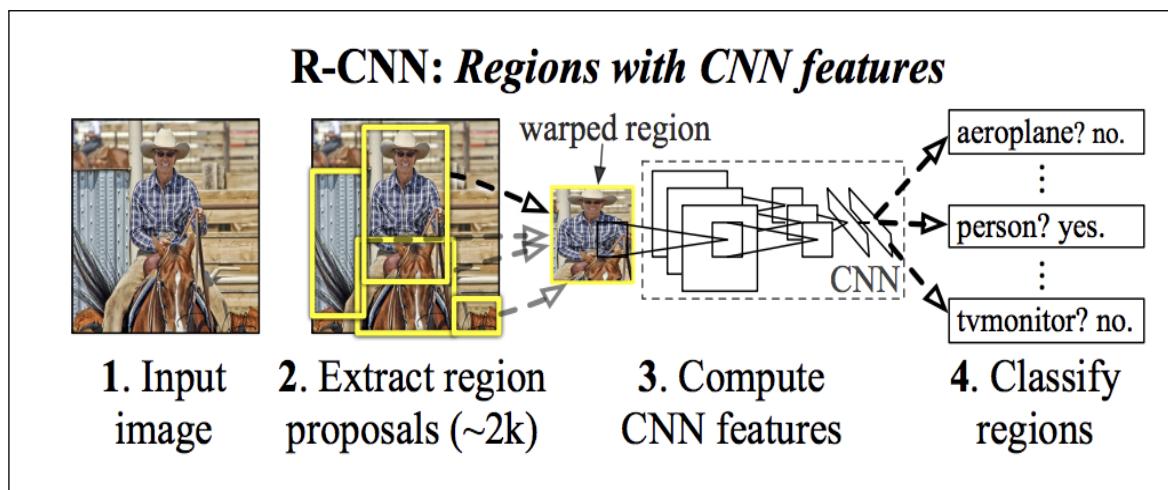


FIGURE 3.3 – Architecture du modèle R-CNN

3.3.2 FAST R-CNN

Ross Girshick [38] a proposé un nouvel algorithme d'apprentissage qui corrige les inconvénients du R-CNN et du SPPNet, tout en améliorant leur vitesse et leur précision. cette méthode est appelée le R-CNN rapide (Fast R-CNN) car elle est comparativement rapide à entraîner et à tester.

Ce réseau prend en entrée une image entière et un ensemble de propositions d'objets. Le réseau traite d'abord l'image entière avec plusieurs couches convolutionnelles (conv) et de mise en commun maximale pour produire une carte de caractéristiques.

Ensuite, pour chaque proposition d'objet, une couche de mise en commun des régions d'intérêt (RoI) extrait un vecteur de caractéristiques de longueur fixe de la carte de caractéristiques. Chaque vecteur de caractéristiques est introduit dans une séquence de couches entièrement connectées (FC) qui se ramifient finalement en deux couches de sortie soeurs :

- Une couche qui produit des estimations de probabilité softmax sur K classes d'objets plus une classe de "fond".
- Une couche qui produit quatre nombres à valeur réelle pour chacune des K classes d'objets. Chaque ensemble de 4 valeurs code les positions raffinées de la boîte de liaison pour l'une des K classes.

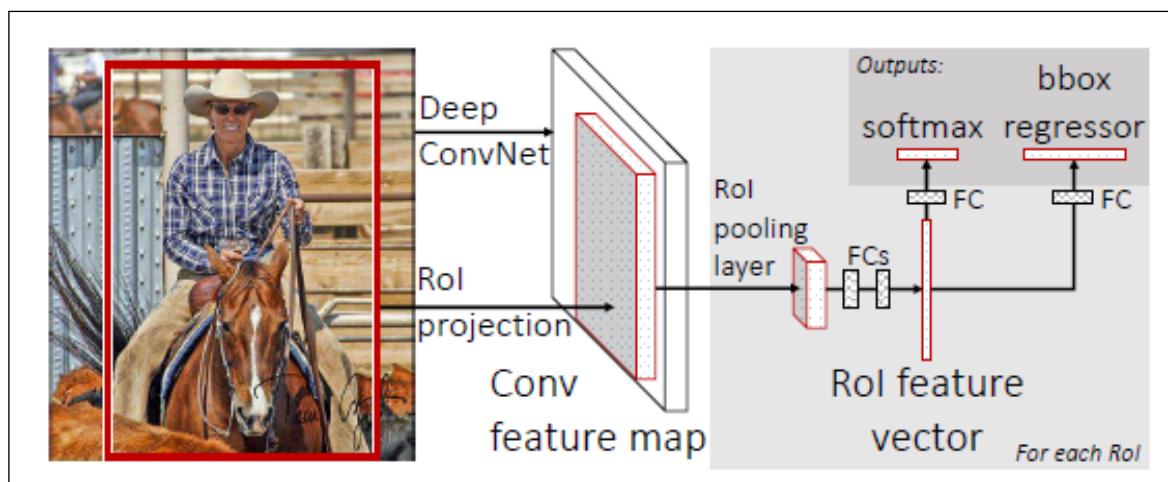


FIGURE 3.4 – Architecture du modèle Fast R-CNN

3.3.3 FASTER R-CNN

Ce réseau est proposé par Shaoqing Ren et al en 2016 [39]. Il se décompose en deux modules principaux :

- Le premier module est un réseau convolutif profond qui crée la carte de caractéristiques convolutives qui utiliser par un module RPN (Réseau de Proposition de Région) qui prend cette carte (de n'importe quelle taille) et produit un ensemble de propositions d'objets rectangulaires, chacune avec un score de précision.
- Le second module est le détecteur Fast R-CNN qui utilise les régions proposées [39] comme nous avons vu dans l'architecture fast R-CNN.

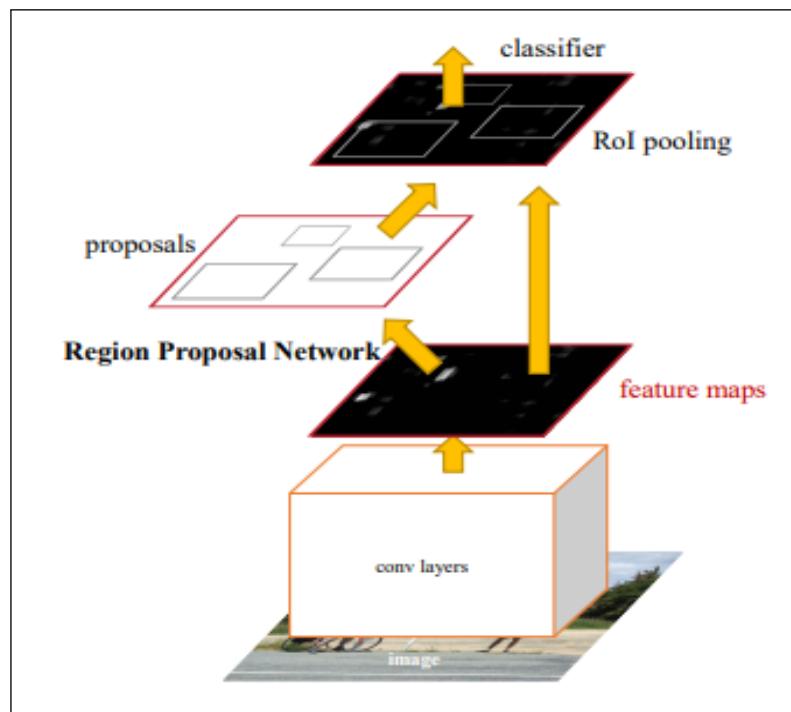


FIGURE 3.5 – Architecture du modèle Faster R-CNN

3.3.4 MASK R-CNN

Kaiming He et al [40] ont proposé une autre architecture appelée Mask R-CNN, étendue de Faster R-CNN en ajoutant une branche pour la prédiction d'un masque d'objet en parallèle avec la branche existante pour la reconnaissance de la boîte englobante.

Le Mask R-CNN masqué est conceptuellement simple : Le Faster R-CNN a deux sorties pour chaque objet candidat, une étiquette de classe et un décalage de la boîte englobante ; à cela, nous ajoutons une troisième branche qui produit le masque de l'objet.

La branche 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.

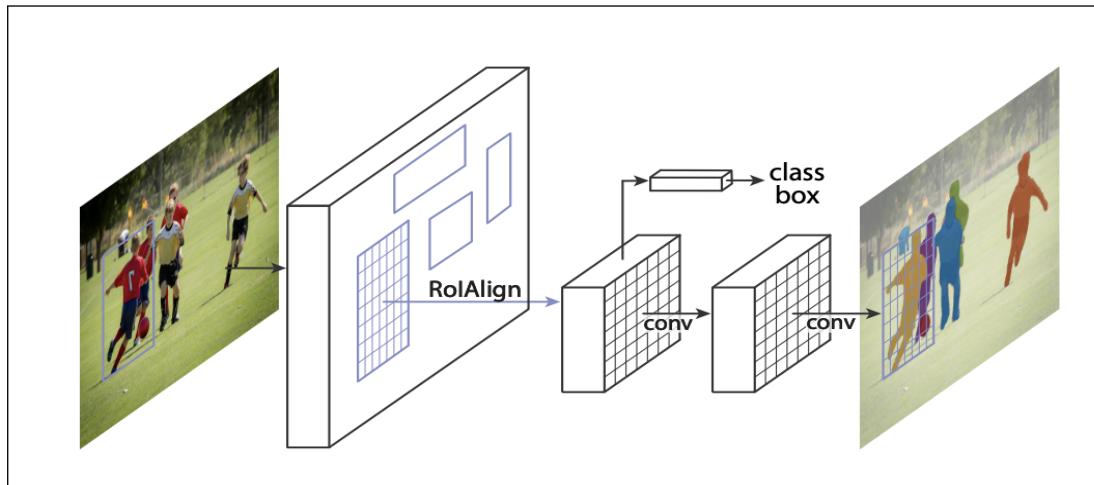


FIGURE 3.6 – Architecture du modèle Mask R-CNN

3.3.5 SSD (SINGLE SHOT MULTIBOX DETECTOR)

L'approche de la SSD est basée sur un réseau convolutif feed-forward qui produit une collection de boîtes englobantes de taille fixe et des scores pour la présence d'instances de classes d'objets dans ces boîtes, suivi d'une étape de suppression non maximale pour produire les detections finales. Les premières couches du réseau sont basées sur une architecture standard utilisée pour la classification d'images de haute qualité (VGG-16) (tronquée avant toute couche de classification), qui s'appelle réseau de base.

Ensuite une structure auxiliaire au réseau pour produire des détections avec les caractéristiques clés suivantes :

- Cartes de caractéristiques multi-échelles pour la détection.
- Prédicteurs convolutifs pour la détection.

- Boîtes et aspect par défaut.[41]

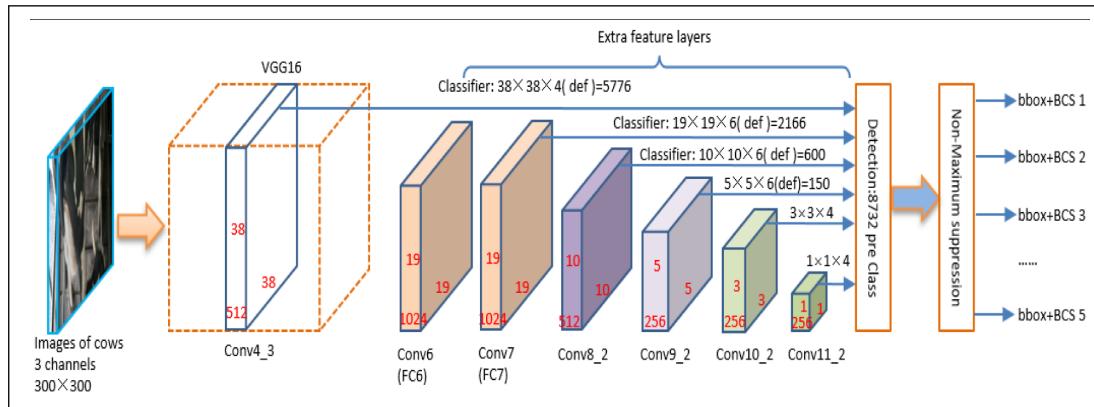


FIGURE 3.7 – Architecture de model SSD

3.3.6 YOLO (YOU ONLY LOOK ONCE)

You Only Look Once ou YOLO est l'un des algorithmes populaires de détection d'objets utilisé par les chercheurs du monde entier. Il a été décrit pour la première fois dans en 2015 dans l'article de Joseph Redmon et al [42].

Le réseau utilise les caractéristiques de l'image entière pour prédire chaque boîte englobante. Il prédit également toutes les boîtes englobantes de toutes les classes d'une image simultanément. Cela signifie que ce réseau raisonne globalement sur l'ensemble de l'image et sur tous les objets qu'elle contient. La conception YOLO permet un apprentissage de bout en bout et des vitesses en temps réel tout en maintenant une précision moyenne élevée [42].

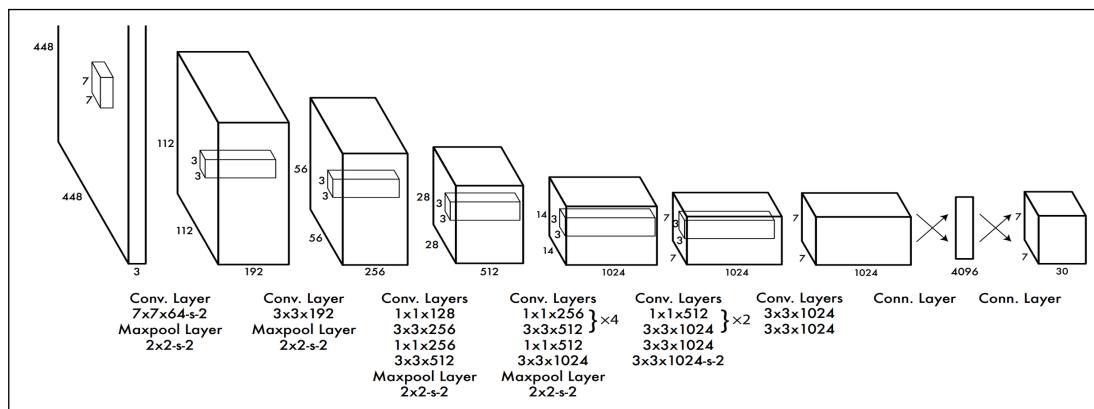


FIGURE 3.8 – Architecture du modèle Yolo

L'architecture de ce réseau est inspirée du modèle GoogLeNet pour la classification d'images. Ce réseau comporte 24 couches convolutives suivies de 2 couches entièrement connectées. Au lieu des modules d'initialisation utilisés par GoogLeNet, elle a utilisé simplement des couches de réduction 1×1 suivies de 3×3 couches convolutives [42].

L'algorithme de modèle YOLO est divisé en 3 étapes :

- **Diviser l'image en cellules avec une taille $S \times S$:** On divise l'image en une grille de taille $S \times S$ (en exemple 3×3), ce qui donne N cellules au total. Cette cellule de la grille est responsable de la détection de cet objet.

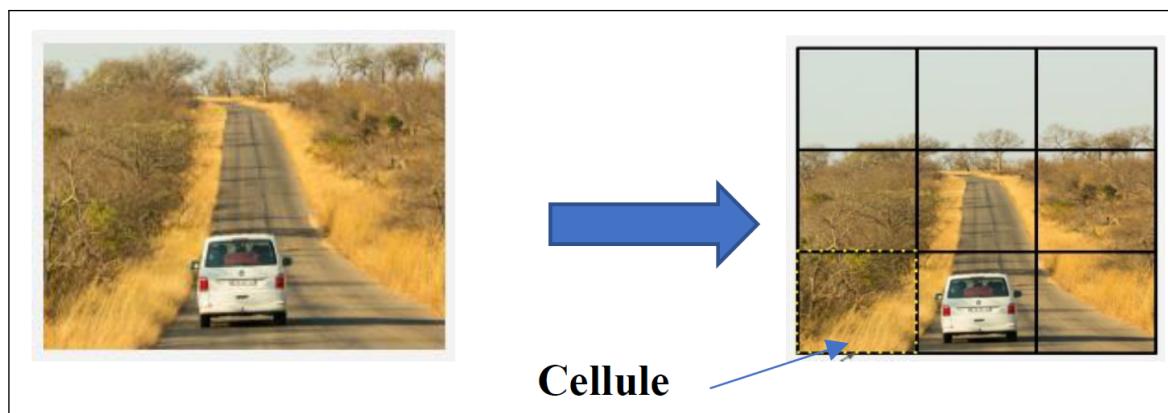


FIGURE 3.9 – Diviser l'image en ($S \times S$) grille

- **Chaque cellule prédit B boîtes englobante :**

Après la division de l'image à N cellules, Chaque cellule de la grille prédit des boîtes englobante B et des scores de confiance pour ces boîtes, Chaque boîte englobante est constituée de 5 prédictions : x , y , w , h , et confiance. Les coordonnées (x , y) représentent le centre de la boîte par rapport aux limites de la cellule de la grille de la boîte par rapport aux limites de la cellule de la grille. La largeur et la hauteur sont prédites en fonction de l'image complète. Enfin, la prédition de confiance représente le IoU entre la boîte la boîte prédite et toute boîte de vérité terrain [42].

Exemple : où il y a 3×3 cellules ($S=3$), chaque cellule prédit 1 boîte limitante ($B=1$), et les objets sont soit chien = 1, soit humain = 2, ($C=2$). Pour chaque cellule, le CNN prédit un vecteur Y :

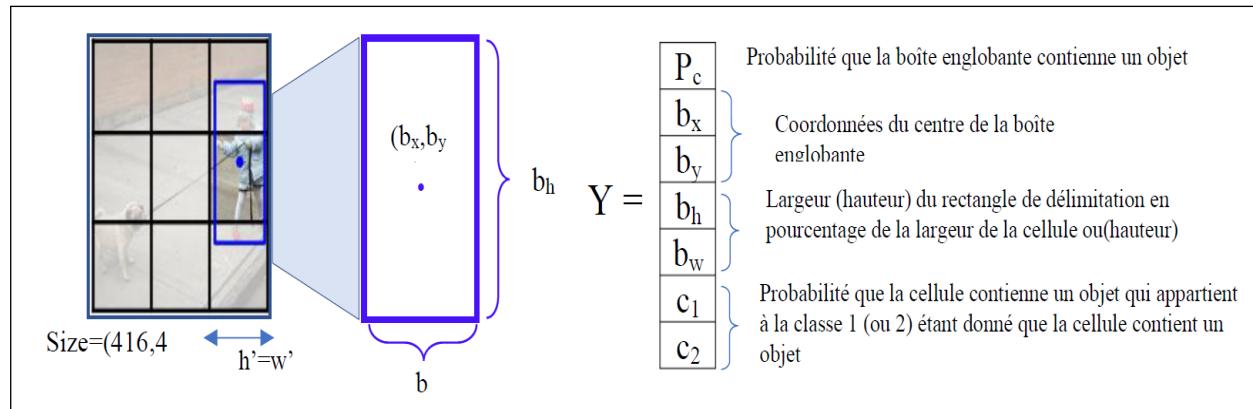


FIGURE 3.10 – Le vecteur prédit dans le cas d'une seule boîte

Les valeurs de vecteur Y sont calculées au format YOLO :

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

$$b_x = \frac{(x-h')}{h'}, b_y = \frac{(y-w')}{w'}, b_h = \frac{h}{416}, b_w = \frac{w}{416}.$$

a) Intersection sur Union (IoU) : L'intersection sur l'union (IoU) est la métrique d'évaluation de facto utilisée dans la détection d'objets. Elle est utilisée pour déterminer les vrais positifs et les faux positifs dans un ensemble de prédictions. Lorsqu'on utilise l'IoU comme mesure d'évaluation, il faut choisir un seuil de précision [43]. Il compare la boîte prédite avec la boîte détectable et peut calculer la surface comme suit :

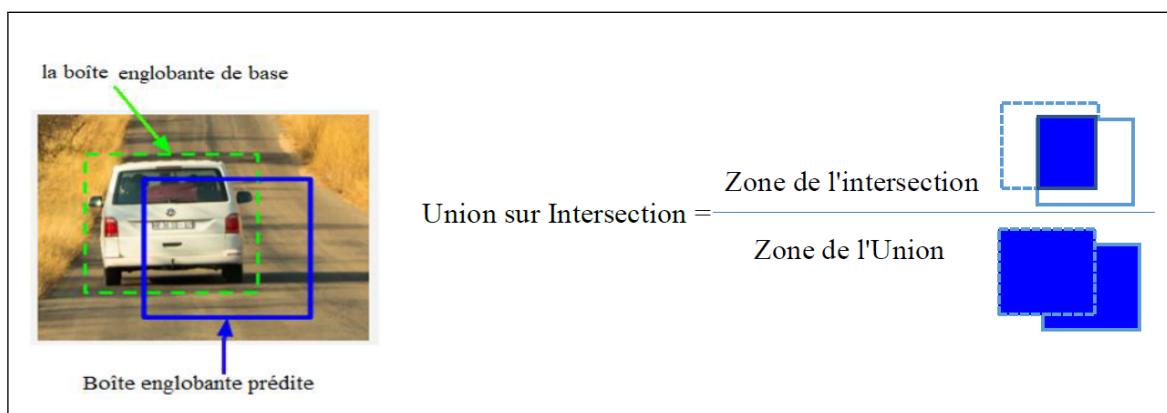


FIGURE 3.11 – Union sur Intersection

Pendant l'apprentissage, l'indice de confiance IoU elle est calculer entre la boîte prédite et la boîte de base. Dans la figure ci-dessus, il y a des exemples de bons et de mauvais scores d'intersection sur Union.

Comme vous pouvez le voir, les boîtes englobantes prédites qui se chevauchent fortement avec les boîtes englobantes de base ont des scores plus élevés que celles qui se chevauchent moins.

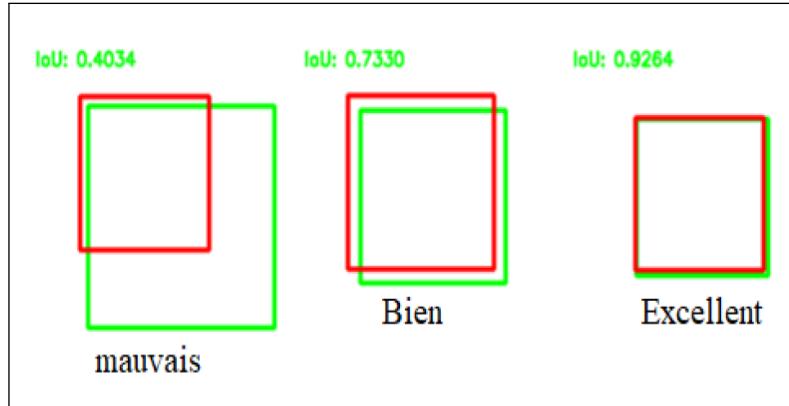


FIGURE 3.12 – Exemples d’IoU : courtoisie

b) Boîte d’ancrage (Anchor Box) : Dans l’exemple précédent, nous n’avons prédit qu’une seule boîte d’ancrage, mais parfois nous avons plus d’une boîte dans la même cellule, donc YOLO sépare les objets si nous avons plus d’une boîte d’ancrage prédite dans la même cellule de grille.[42].

Comme nous avons dit dans la première partie que chaque cellule représenter par un vecteur, dans le cas il y a plusieurs boîtes dans la même cellule en augmenter le vecteur comme suite :

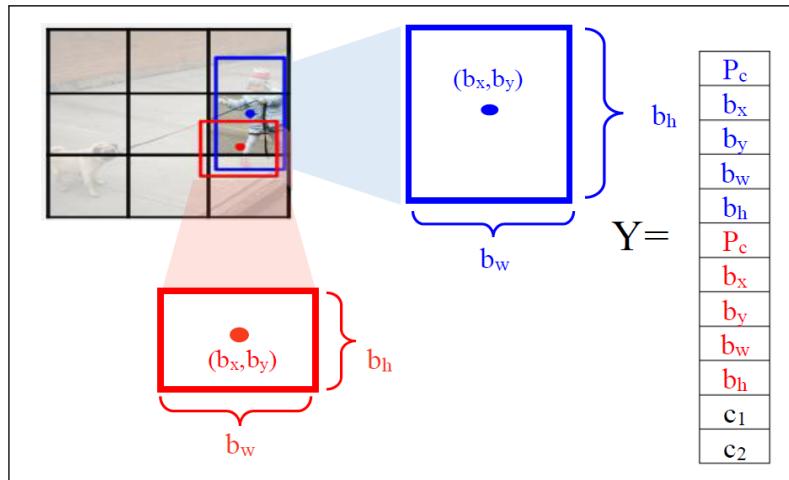


FIGURE 3.13 – Le vecteur prédit dans le cas de plusieurs boîtes dans la cellule

Donc en général la formule si on divise l'image en une grille $S \times S$ et, pour chaque cellule de la grille, il prédit B boîtes de délimitation, la confiance pour ces boîtes et les probabilités de classe C . Ces prédictions sont encodées sous la forme d'un tenseur $S * S * (B * 5 + C)$ [43].

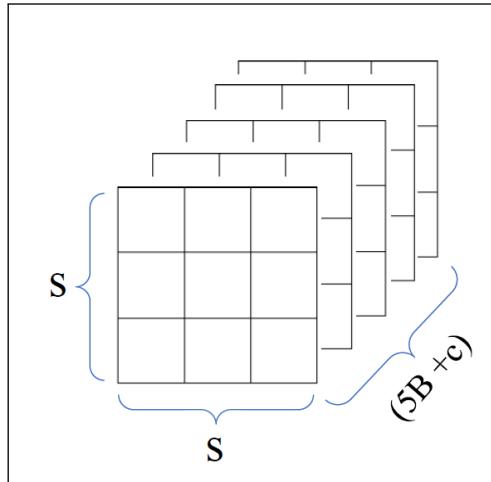


FIGURE 3.14 – Un tenseur qui spécifie les emplacements des boîtes et les probabilités de classe.

c) **Suppression non maximale** : cette étape c'est la dernière étape dans l'algorithme de détection, elle est utilisée c'est le même objet dans l'image détecter par plusieurs boîtes englobant, cette technique est utilisée pour "supprimer" les boîtes englobantes les moins probables et ne garder que la meilleure. Alors le processus de cette technique passe à 5 étapes [43] :

- **Étape 1** : Sélectionner la boîte avec le score d'objectivité le plus élevé.
- **Étape 2** : Ensuite, on compare le chevauchement (intersection sur union) de cette boîte avec d'autres boîtes.
- **Étape 3** : Supprimez les boîtes englobantes dont le chevauchement (intersection sur union) est $> 50\%$.
- **Étape 4** : Passez ensuite au score d'objectivité le plus élevé suivant.
- **Étape 5** : Enfin, répétez les étapes 2 à 4 jusqu'à fini tous les objets dans l'image.

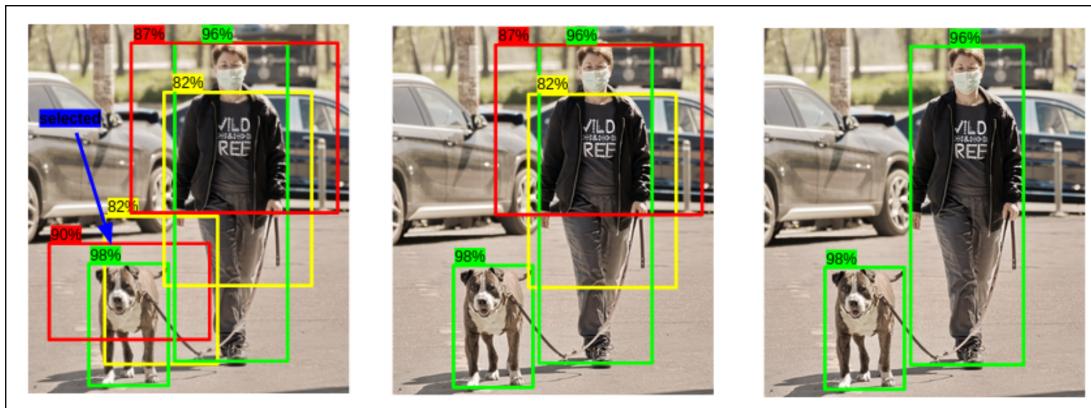


FIGURE 3.15 – Le résultat de suppression non maximale

Il y a eu 7 versions du modèle jusqu'à présent, chaque nouvelle version améliorant la précédente en termes de vitesse et de précision.

3.3.6.1 LE MODÈLE YOLOV2

Dans cette version, Joseph Redmon et Ali Farhadi [45] ont essayé de créer un modèle meilleur, plus rapide, plus fort et pour cela ils ont fait une amélioration dans la 1ère architecture de modèle yolo, où ils ont utilisé Darknet-19 comme backbone, et la structure complète est passée à 30 couches, contre 26 couches pour YOLO v1 et inclusion de couches de normalisation par lots après chaque couche de convolution, aussi augmente la résolution à 448 pour la détection et grille avec stride=32 avec prédition de 5 boîtes limitantes à chaque cellule, Les boîtes d'ancrage ont été introduites[43].

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	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		$2 \times 2/2$	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			

FIGURE 3.16 – Architecture de darknet19

3.3.6.2 LE MODÈLE YOLOV3

Joseph Redmon et Ali Farhadi en 2018 [43] ont aussi fait une amélioration progressive dans la version précédent, où ils ont utilisé comme backbone Darknet53 pour extraction de caractéristique, et pour calculer un score d'objectalité de chaque boîte de délimitation en utilisant une régression logistique. Pour les prédictions de classe ils ont utilisés la perte d'entropie croisée binaire . Dans la version de yolov2 il y a un problème pour la détection des petits objets mais dans cette version la représentation des boîtes à 3 échelles différentes.

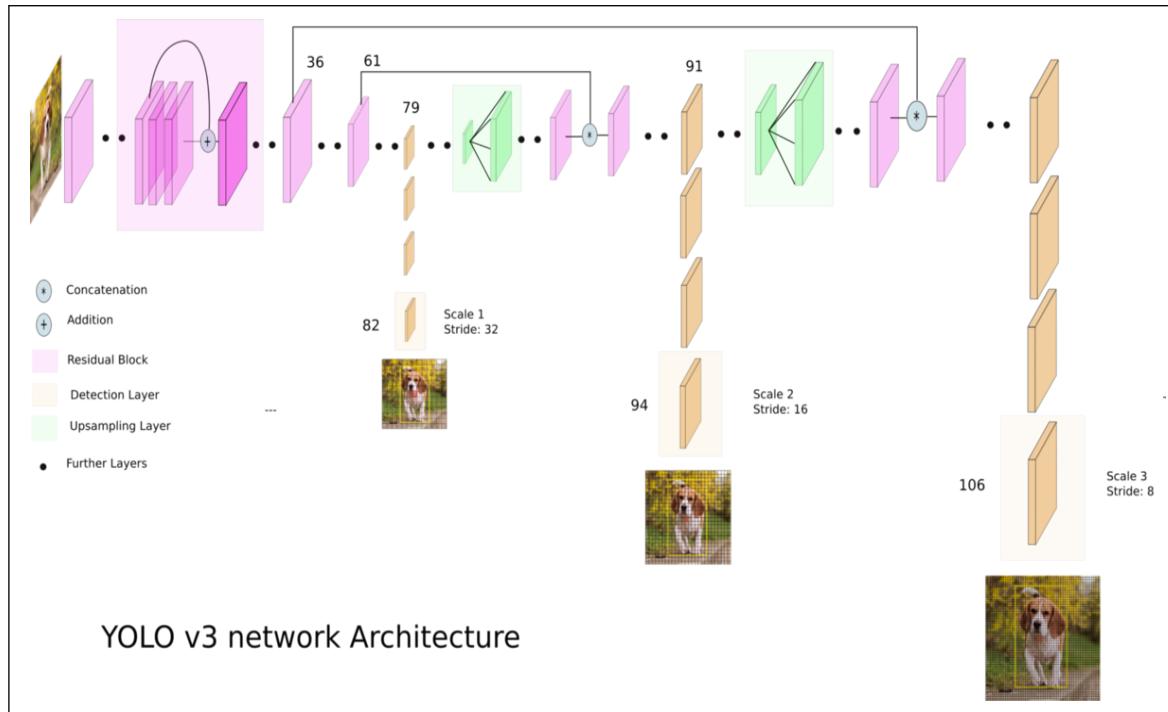


FIGURE 3.17 – Architecture de modèle yolov3

3.3.6.3 LE MODÈLE YOLOV4

Il existe un grand nombre de fonctionnalités censées améliorer la précision du réseau neuronal convolutif (CNN). Des tests pratiques de combinaisons de ces caractéristiques sur de grands ensembles de données et une justification théorique du résultat sont nécessaires. Certaines fonctionnalités fonctionnent sur certains modèles exclusivement et pour certains problèmes exclusivement, ou uniquement pour des ensembles de données à petite échelle ; tandis que certaines fonctionnalités, telles que la normalisation par lots et les connexions résiduelles, sont applicables à la majorité des modèles, des tâches et des ensembles de données. Nous supposons que ces fonctionnalités universelles incluent les connexions résiduelles pondérées (WRC), les connexions partielles inter-étapes (CSP), la normalisation croisée des mini-lots (CmBN), la formation auto-adversaire (SAT) et l’activation Mish. Nous utilisons de nouvelles fonctionnalités : WRC, CSP, CmBN, SAT, Mish activation, Mosaic data augmentation, CmBN, DropBlock regularization, et CIoU loss, et combinons certains d’entre eux pour obtenir des résultats de pointe : 43,5% AP (65,7% AP50) pour l’ensemble de données MS COCO à une vitesse en temps réel d’environ 65 FPS sur Tesla V100.

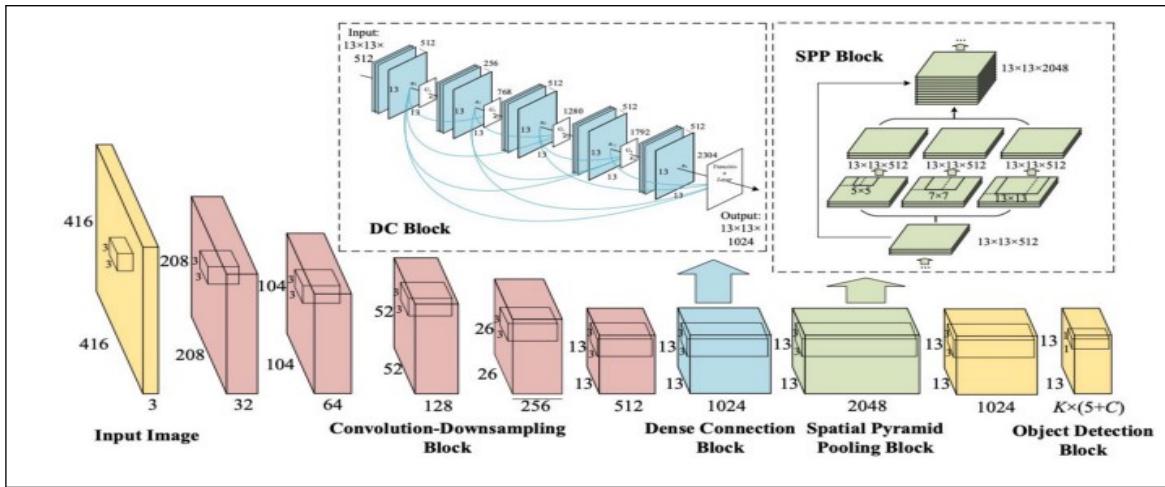


FIGURE 3.18 – Architecture du modèle yolov4

3.3.6.4 LE MODÈLE YOLOV5

Comme tous les modèles de détection d'objet yolov5 composé de trois parties :

- **Backbone (Colonne vertébrale)** : Un réseau de neurones convolutifs qui agrège et forme des caractéristiques d'image à différentes granularités.
- **Neck (Cou)** : Une série de couches pour mélanger et combiner les caractéristiques de l'image pour les transmettre à la prédiction.
- **Head (Tête)** : Consomme les caractéristiques du cou et effectue des étapes de prédiction de boîte et de classe.

L'architecture de modèle YOLOv5 se compose comme le montre la figure 3.17 :

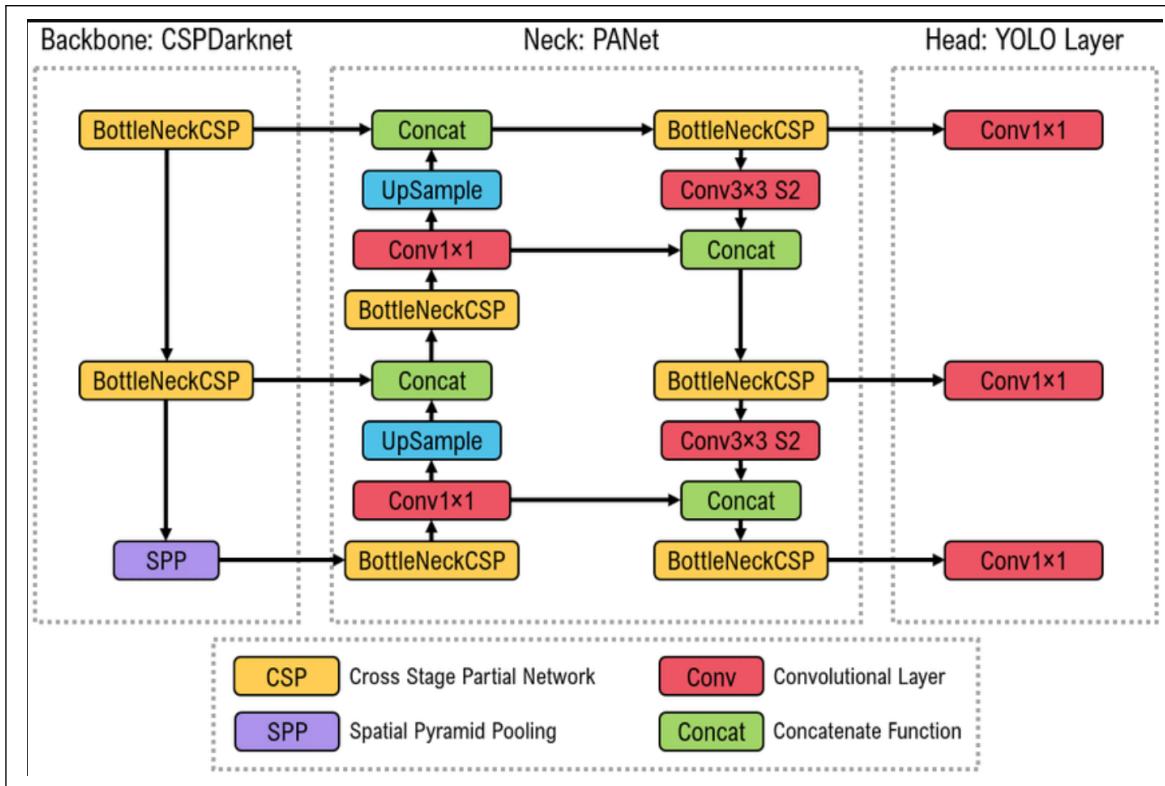


FIGURE 3.19 – Architecture du modèle yolov5

- **Backbone (Colonne vertébrale)**

a) Cross Stage Partial Network (CSP) : Les modèles CSP (Cross Stage Partial Darkent) sont dérivés de l'architecture DenseNet qui utilise l'entrée précédente et la concatène avec l'entrée actuelle avant de passer à la couche dense [47]. DenseNet a été conçu pour connecter des couches dans un réseau de neurones très profond dans le but d'atténuer les problèmes de gradient de disparition [48].

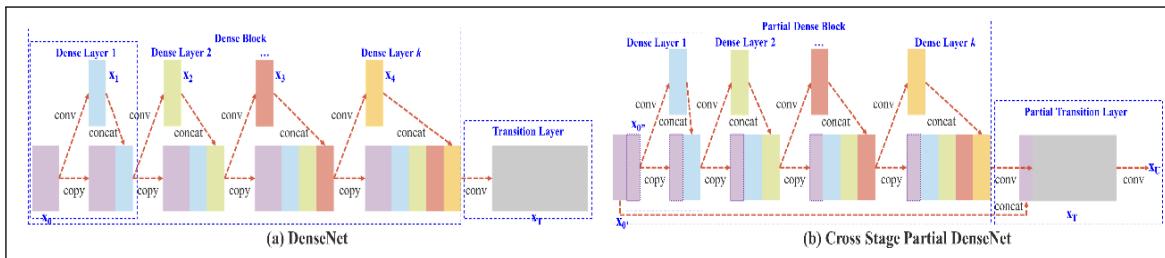


FIGURE 3.20 – DenseNet

Dans la figure ci-dessus, nous avons Illustrations de (a) DenseNet et (b) de Cross Stage Partial DenseNet (CSPDenseNet). CSPNet sépare la carte des caractéristiques de la couche de

base en deux parties, une partie passera par un bloc dense et une couche de transition ; l'autre partie est ensuite combinée avec la carte des caractéristiques transmise à l'étape suivante.

DenseNet : Chaque étape d'un DenseNet contient un bloc dense et une couche de transition, et chaque bloc dense est composé de k couches denses. La sortie de la i^{th} couche dense sera concaténée avec l'entrée de la couche dense i^{th} , et le résultat concaténé deviendra l'entrée de la $(i + 1)^{th}$ couche dense. Les équations montrant le mécanisme mentionné ci-dessus peuvent être exprimées comme suit [48] :

$$x_1 = w_1 * x_0 \quad (3.2)$$

$$x_2 = w_2 * [x_0, x_1] \quad (3.3)$$

$$x_k = w_k * [x_0, x_1, \dots, x_{k-1}] \quad (3.4)$$

Où $*$ représente l'opérateur de convolution, et $[x_0, x_1, \dots]$ signifie concaténer x_0, x_1, \dots et w_i et x_i sont respectivement les poids et la sortie de la couche dense. Si l'on utilise un algorithme de propagation vers l'avant pour mettre à jour les poids, les équations de mise à jour des poids peuvent s'écrire comme [48] :

$$w_1 = f(w_1, g_0) \quad (3.5)$$

$$w_2 = f(w_2, g_0, g_1) \quad (3.6)$$

$$w_k = f(w_k, g_0, g_1, \dots, g_{k-1}) \quad (3.7)$$

Où f est la fonction de mise à jour du poids, et g_i représente le gradient propagé à la i^{th} couche dense. Nous pouvons constater qu'une grande quantité d'informations de gradient est

réutilisée pour mettre à jour les poids de différentes couches denses. Il en résultera que les différentes couches denses apprendront à plusieurs reprises les informations de gradient copiées.

Cross Stage Partial Dense Net : L'architecture à une étape du CSPDenseNet proposé est illustrée à la figure 3.18 (b). Une étape de CSPDenseNet est composée d'un bloc dense partiel et d'une couche de transition partielle. Dans un bloc dense partiel, les cartes de caractéristiques de la couche de base dans une étape sont divisées en deux parties via le canal $x_0 = [x'_0, x''_0]$. Entre x'_0 et x''_0 premier est directement lié à la fin de l'étage, et le second va traverser un bloc dense. Toutes les étapes impliquées dans une couche de transition partielle sont les suivantes : Premièrement, la sortie des couches denses, $[x''_0, x_1, \dots, x_k]$ subira une couche de transition. Deuxièmement, la sortie de cette couche de transition, x_T subira, sera concaténée avec x''_0 subira et subira une autre couche de transition, puis générera la sortie x_U subira. Les équations de la réussite et de la mise à jour du poids de CSPDenseNet sont présentées dans les équations suivantes , respectivement [48].

$$x_k = w_k * [x''_0, x_1, \dots, x_{k-1}] \quad (3.8)$$

$$x_T = w_T * [x'_0, x_1, \dots, x_K] \quad (3.9)$$

$$x_U = w_U * [x''_0, x_T] \quad (3.10)$$

$$w_k = f(w_k, g_0, g_1, \dots, g_{k-1}) \quad (3.11)$$

$$w_T = f(w_T, g_0, g_1, \dots, g_k) \quad (3.12)$$

$$w_U = f(w_U, g_0, g_T) \quad (3.13)$$

Le CSP maintient les caractéristiques par propagation, encourage le réseau à réutiliser les caractéristiques et réduit le nombre de paramètres réseau, aide à préserver les caractéristiques à grains fin pour les transmettre plus efficacement aux couches plus profondes. Considérant qu'une augmentation excessive des couches convolutives densément connectées peut conduire à une diminution de la vitesse de détection, seul le dernier bloc convolutif qui peut extraire les caractéristiques sémantiques les plus riches du réseau dorsal Darknet-53 est amélioré pour devenir un bloc dense [48].

b) CSPDarknet

Il s'agit d'un réseau de neurones convolutifs et d'une dorsale pour la détection d'objets qui utilise DarkNet-53. Il utilise une stratégie CSPNet pour partitionner la carte des caractéristiques de la couche de base en deux parties, puis les fusionne via une hiérarchie à plusieurs niveaux. L'utilisation d'une stratégie de division et de fusion permet un plus grand flux de gradient à travers le réseau. Les modèles utilisent les connexions CSP avec le Darknet-53 (Figure 3.19) ci-dessous comme épine dorsale dans l'extraction de fonctionnalités.

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

FIGURE 3.21 – Architecture Darknet-53

c) **Spatial Pyramid Pooling** Un bloc de mise en commun de pyramide spatiale « SPP » avec trois couches de mise en commun maximum illustré sur la figure 3.8 est introduit entre le bloc DC et la couche de détection d’objet dans le réseau. La convolution 1×1 est utilisée pour réduire le nombre de cartes de caractéristiques en entrée de 1024 à 512. Après cela, les cartes de caractéristiques sont regroupées à différentes échelles ; $size_{pool} \times size_{pool}$ représente la taille des fenêtres coulissantes, $size_{fmap} \times size_{fmap}$ représente la taille des cartes de caractéristiques, puis $size_{pool} = \left[\frac{size_{fmap}}{n_i} \right]$ [49].

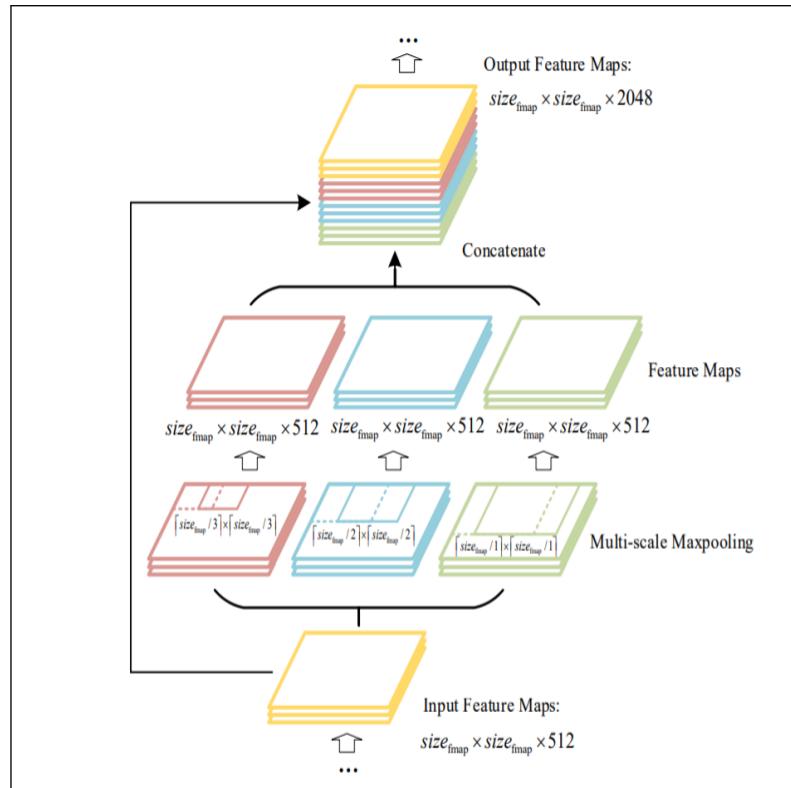


FIGURE 3.22 – Améliorée SPP

- Neck (Cou)

Yolov5 utilise PANet comme Cou(Neck) pour agréger les fonctionnalités et il est basé sur frameworks FPN, tout en améliorant la diffusion de l'information. L'architecture FPN a mis en oeuvre un chemin descendant pour transférer les caractéristiques sémantiques (de la couche de haut niveau), puis les concaténer en caractéristiques à grain fin (de la couche de bas niveau dans la dorsale) pour prédire les petits objets dans le détecteur à grande échelle figure 3.21.

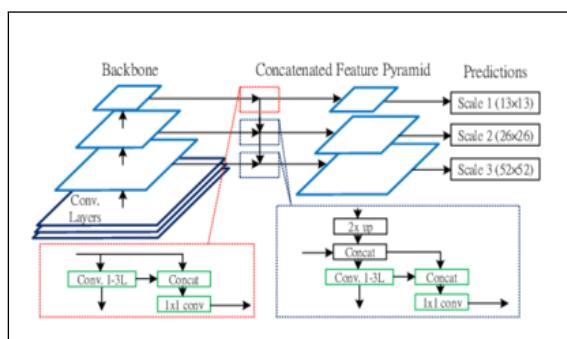


FIGURE 3.23 – Architecture de FPN de Yolov3 [50]

Le réseau d'agrégation de chemins « PAN » est une version avancée de FPN et il fonctionne de la même manière que les FPN, mais ils ont ajouté un chemin d'augmentation ascendant comme le montre la figure 3.10(b) (ci-dessous) 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 [51].

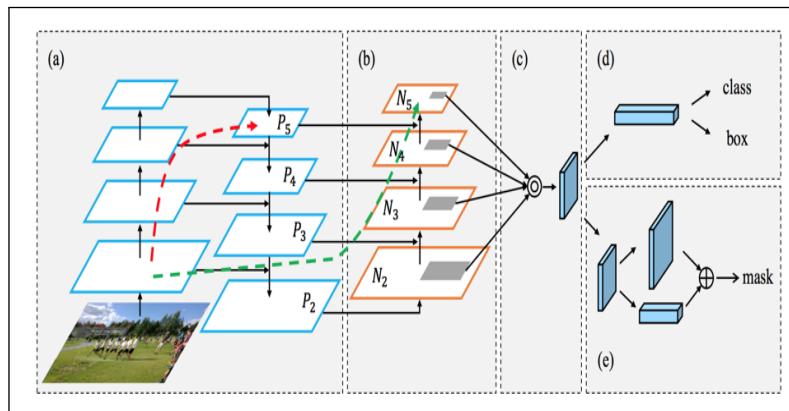


FIGURE 3.24 – Architecture du PAN

a) Colonne vertébrale FPN. (b) Augmentation de la trajectoire ascendante. (c) Mise en commun des fonctionnalités adaptatives. (d) Branche-boîte. (e) Fusion entièrement connectée.

- **Head (Tête)**

La tête est principalement utilisée dans la partie de détection finale. Il applique une boîte d'ancre sur la carte des caractéristiques et génère le vecteur de sortie finale avec la probabilité de classe, le score de l'objet et la boîte englobant.

- **La boîte d'ancre (Anchor box)** La boîte d'ancre est une liste de boîtes prédéfinies qui correspondent le mieux aux objets souhaités. Les boîtes englobantes n'ont pas seulement été prédites en fonction des boîtes de vérité terrain, mais également des boîtes d'ancre prédéfinies, en Yolov5 la boîte d'ancre est automatiquement apprise en fonction des données d'entraînement.

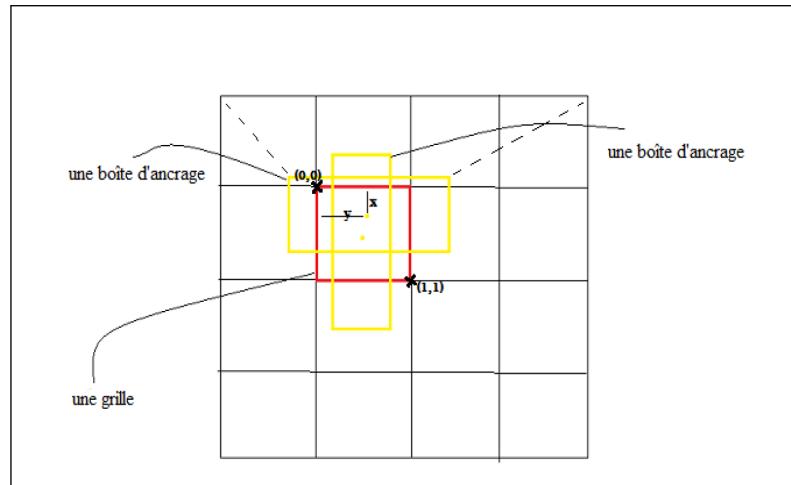


FIGURE 3.25 – Les boîtes d’ancrages

L'idée de la boîte d'ancrage est de combiner les dimensions de sortie de plusieurs objets dans une seule cellule de grille.

- **Generalized Intersection over Union**

L'idée de Generalized Intersection over Union « GioU » est de rapprocher la boîte prédite de la vérité terrain malgré l'absence de chevauchement qui est fonction de perte où se formule comme suit :

$$GIoU = \frac{A \cap B}{A \cup B} - \frac{|c/(A \cup B) - |}{|C|} = IoU - \frac{|c/(A \cup B) - |}{|C|} \quad (3.14)$$

Où :

- A et B sont les boîtes englobantes de prédiction et de vérité terrain.
- C est la plus petite enveloppe convexe englobe à la fois A et B .

3.4 MÉSUSER LA PERFORMANCE D'UN MODÈLE DE DÉTECTION

Les indicateurs de performance utilisés pour prendre une vue générale sur l'efficacité de notre modèle, ces indicateurs calculés par la matrice de confidence qui contient des nombres sont :

- **Vrais positifs** : le nombre de fois où le modèle prédit correctement la classe positive.
- **Vrais négatifs** : le nombre de fois où le modèle prédit correctement la classe négative.
- **Faux positifs** : le nombre de fois où le modèle prédit la classe positive alors que le résultat attendu était la classe négative.
- **Faux négatifs** : le nombre de fois où le modèle prédit la classe négative alors que le résultat attendu était la classe positive.

Exemple :

Vrai positif	Faux positif
Prédiction : bâtiment présent	Prédiction : bâtiment présent
Résultat attendu : bâtiment présent	Résultat attendu : pas de bâtiment
Faux négatif	Vrai négatif
Prédiction : pas de bâtiment	Prédiction : pas de bâtiment
Résultat attendu: bâtiment présent	Résultat attendu: pas de bâtiment

a) **Précision** : La précision est le pourcentage de détections correctes. Il met en évidence l'exactitude des prédictions. Il se calcule par ce ratio :

$$Prsicion = \frac{vraipositive}{vraipositif + fauxpositif} \quad (3.15)$$

b) Rappel : Le rappel est un indicateur qui mesure la capacité du modèle à prédire l'ensemble des résultats attendus. Calculer par :

$$Rappel = \frac{vraipositif}{vraipositif + fauxnégatif} \quad (3.16)$$

c) L'accuracy : il indique le pourcentage de bonnes prédictions :

$$L'accuracy = \frac{vraipositif + vraingatif}{total} \quad (3.17)$$

En outre, sur l' ensemble de données MS COCO , une métrique de référence importante est le temps d'inférence (ms/Frame, inférieur est meilleur) ou Images par seconde (FPS, plus élevé est meilleur). Les progrès rapides de la technologie de vision par ordinateur sont très visibles lorsque l'on examine les comparaisons de temps d'inférence. Sur la base des temps d'inférence actuels (le plus bas est le meilleur), YOLO atteint 3,5 ms par image. Notez comment l'introduction de YOLO (détecteur à une étape) a conduit à des temps d'inférence considérablement plus rapides par rapport à toutes les méthodes précédemment établies, telles que la méthode à deux étapes Mask R-CNN (333 ms). Sur le plan technique, il est assez complexe de comparer différentes architectures et versions de modèles de manière significative. Et Edge AI devient une partie intégrante des solutions d'IA évolutives, les nouveaux algorithmes sont livrés avec une version plus légère optimisée pour les bords [51].

3.5 CONCLUSION

Dans ce chapitre, nous avons présenté un aperçu des méthodes de détection d'objets basées sur l'apprentissage profond. Nous avons commencé par la structure générale d'un modèle de détection basé sur le Deep Learning et avons passé en revue les méthodes de détection d'objets les plus connues et les plus utilisées avec leurs architectures. Nous nous sommes ensuite concentrés sur la méthode YOLO. Nous avons vu leur algorithme détaillé avec les 5 versions de YOLO et la différence entre chaque version et en se basant beaucoup plus sur la version YOLOv5 que

LES MODÈLES DE DÉTECTION

nous avons utilisée dans notre conception, nous concluons avec la mesure de performance d'un modèle de détection.

REALISATION

Sommaire

4.1	INTRODUCTION	61
4.2	ENVIRONNEMENT DE DÉVELOPPEMENT	61
4.2.1	ENVIRONNEMENT LOGICIEL	61
4.2.2	OUTILS TECHNOLOGIQUES	63
4.3	L'ENSEMBLE DES DONNÉES	65
4.3.1	COLLECTION DES DONNÉES	65
4.3.2	PRÉ-TRAITEMENT DES DONNÉES	66
4.4	APPRENTISSAGE DE MODEL YOLOV5	70
4.4.1	TÉLÉCHARGER LE MODÈLE YOLOV5	70
4.4.2	ENTRAÎNEMENT	73
4.5	APERÇU DE L'APPLICATION	79
4.6	CONCLUSION	80

4.1 INTRODUCTION

Dans ce chapitre, nous présentons l'environnement de développement. Tout d'abord, nous commençons par les jeux de données et les différentes étapes du projet. Ensuite, nous décrivons les différentes architectures du modèle. Enfin, nous présentons les interfaces de l'application.

4.2 ENVIRONNEMENT DE DÉVELOPPEMENT

Cette partie est consacrée pour la présentation des différents outils logiciels nécessaires et les technologies qui vont nous permettre de faire une analyse prédictive et descriptive. D'autre part, nous allons expliquer comment nous avons préparé notre environnement de travail avec l'approche Data Science.

4.2.1 ENVIRONNEMENT LOGICIEL

4.2.1.1 GOOGLE COLAB

Google Colab est un service cloud, offert par Google. Colab qui nous permet d'écrire et d'exécuter du code Python arbitraire via le navigateur. C'est un environnement particulièrement adapté au machine Learning, à l'analyse de données et à l'éducation. Les notebooks Colab sont des notebooks Jupyter qui s'exécutent dans le cloud et sont hautement intégrés à Google Drive. Le plus grand avantage est que Colab prend en charge les bibliothèques d'apprentissage automatique les plus populaires qui peuvent être facilement chargées dans votre Notebook. En termes plus techniques, Colab est un service hébergé de notebooks Jupyter qui ne nécessite aucune configuration et permet d'accéder sans frais à des ressources informatiques, dont des GPU.[24]



FIGURE 4.1 – Logo Google Colab

4.2.1.2 ANDROID STUDIO

Android Studio est un environnement de développement pour développer des applications mobiles Android. Il est basé sur IntelliJ IDEA et utilise le moteur de production Gradle. Android Studio permet principalement d'éditer les fichiers Java/Kotlin et les fichiers de configuration XML d'une application Android.

Il propose entre autres des outils pour gérer le développement d'applications multilingues et permet de visualiser rapidement la mise en page des écrans sur des écrans de résolutions variées simultanément. Il intègre par ailleurs un émulateur permettant de faire tourner un système Android virtuel sur un ordinateur.

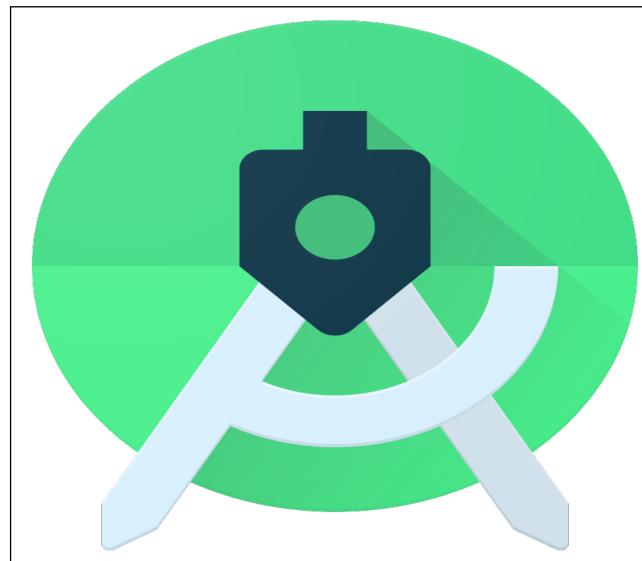


FIGURE 4.2 – Logo Android Studio

4.2.1.3 LABELIMG

LabelImg est un outil gratuit et open source pour l'étiquetage graphique des images. Il est écrit en Python et utilise QT pour son interface graphique. C'est un moyen facile et gratuit d'étiqueter quelques centaines d'images pour essayer votre prochain projet de détection d'objets. Les étiquettes sont utilisées pour aider à identifier les composants de vos données que vous souhaitez entraîner votre modèle à identifier dans des ensembles de données qui ne sont pas étiquetés.[25]



FIGURE 4.3 – Logo LabelImg

4.2.2 OUTILS TECHNOLOGIQUES

Cette section sera consacrée à présenter nos choix technologiques et à montrer leurs caractéristiques.

4.2.2.1 LANGUE UTILISÉ

Dans la partie construction du modèle, nous avons utilisé plusieurs bibliothèques sous le langage python. Python est un langage de programmation de haut niveau, interprété (il n'y a pas d'étape de compilation) et orienté objet avec une sémantique dynamique. Il est largement utilisé par une grande communauté de développeurs et de programmeurs. Python est un langage simple, facile à apprendre et permet une bonne réduction des coûts de maintenance du code. Les bibliothèques Python (packages) favorisent la modularité et la réutilisabilité des codes. Python

et ses bibliothèques sont disponibles (en source ou en binaires) gratuitement pour la plupart des plateformes et peuvent être redistribués gratuitement.[26]

4.2.2.2 LES BIBLIOTHÈQUES ET LES FRAMEWORKS UTILISÉS

Comme nous le savons, Python est utilisé pour diverses applications et il existe différentes bibliothèques à des fins différentes. Dans notre projets, nous avons utilisés les bibliothèques et les Frameworks suivantes :

Pandas : est une bibliothèque écrite pour le langage de programmation Python permettant la manipulation et l'analyse des données. Elle propose en particulier des structures de données et des opérations de manipulation de tableaux numériques et de séries temporelles. Pandas est un logiciel libre sous licence BSD.[27]

NumPy : est une bibliothèque pour langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.[28]

Matplotlib : est une bibliothèque du langage de programmation Python destinée à tracer et visualiser des données sous forme de graphiques. Elle peut être combinée avec les bibliothèques python de calcul scientifique NumPy et SciPy.[28]

Keras : permet d'interagir avec les algorithmes de réseaux de neurones profonds et d'apprentissage automatique. Conçue pour permettre une expérimentation rapide avec les réseaux de neurones profonds, elle se concentre sur son ergonomie, sa modularité et ses capacités d'extension.[29]

TensorFlow : Est une plate-forme Open Source de bout en bout dédiée au machine learning. Elle propose un écosystème complet et flexible d'outils, de bibliothèques et de ressources communautaires permettant aux chercheurs d'avancer dans le domaine du machine learning, et aux développeurs de créer et de déployer facilement des applications qui exploitent cette technologie.[30]

Pillow : est une bibliothèque de traitement d'images pour le langage de programmation Python. Elle permet d'ouvrir, de manipuler, et de sauvegarder différents formats de fichiers

graphiques. La bibliothèque est disponible librement selon les termes de la Python Imaging Library license.[31]

PyTorch : est une bibliothèque logicielle Python open source d'apprentissage machine qui s'appuie sur Torch développée par Meta. PyTorch est gouverné par la PyTorch Foundation. PyTorch permet d'effectuer les calculs tensoriels nécessaires notamment pour l'apprentissage profond.[32]

Flutter : est un kit de développement logiciel d'interface utilisateur open-source créé par Google. Il est utilisé pour développer des applications pour Android, iOS, Linux, Mac, Windows, Google Fuchsia et le web à partir d'une seule base de code.[33]

4.3 L'ENSEMBLE DES DONNÉES

Afin de construire un modèle d'apprentissage profond, nous avons besoin d'un ensemble de données, dans ce cas une collection de données qui contient généralement des données manquantes, des attributs catégoriques, des données différentes, etc... C'est pour cela qu'il est nécessaire d'effectuer un ensemble d'étapes afin de préparer la collecte de données pour entraîner et tester un modèle d'apprentissage profond.

4.3.1 COLLECTION DES DONNÉES

La collecte de données est primordiale pour un projet d'apprentissage profond. C'est une étape importante qui consomme beaucoup de temps et de ressources. Le jeu de données utilisé dans notre expérience est un ensemble d'images de la marque Starbucks. Malheureusement, il n'y a pas de jeux de données ouverts disponibles pour le public de Starbucks. J'ai donc commencé par la méthode classique consistant à rechercher des images sur le Web et à les sélectionner manuellement pour les télécharger. Comme il faut un grand volume de données, cette méthode n'est pas efficace. J'ai donc trouvé un programme pour parcourir le Web et télécharger les images que je voulais. L'un de ces programmes est Télécharger toutes les images, une ex-

tension Google Chrome (téléchargeur d'images - Imageye) qui vous permet de télécharger un tas d'images en une seule fois.

J'ai collecté ces images à partir de différents sites (shutterstock.com, istockphoto.com, Pinterest, pngitem ... etc) qui contient 432 images, Voici un aperçu de notre ensemble de données.



FIGURE 4.4 – Aperçu d'ensembles de données

4.3.2 PRÉ-TRAITEMENT DES DONNÉES

L'acte de traitement d'une image peut impliquer un large éventail de procédures, et le produit final peut être une autre image, une modification, ou même simplement un paramètre de l'original. Ce résultat peut ensuite être utilisé pour d'autres recherches ou prises de décision. C'est le fondement de la vision par ordinateur, qui est essentielle à de nombreuses applications dans le monde réel, notamment la robotique, les voitures sans conducteur et la reconnaissance des objets. Les images peuvent être transformées, manipulées et utilisées pour obtenir des informations précieuses grâce au traitement d'images.

4.3.2.1 NORMALISATION AVEC ÉGALISATION D'HISTOGRAMME

Le contraste global d'une image peut être ajusté par l'égalisation de l'histogramme, une technique de base du traitement d'image, en mettant à jour la distribution de l'intensité des pixels de l'image. Cela permet aux zones à faible contraste d'avoir un contraste plus élevé dans l'image de sortie.

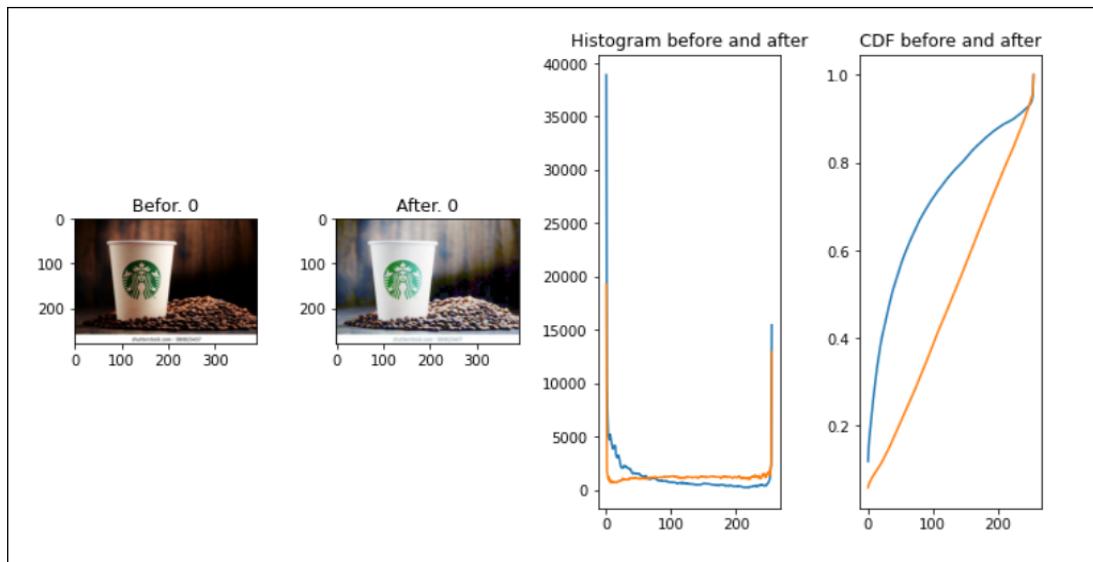


FIGURE 4.5 – Égalisation de l'histogramme

4.3.2.2 FILTRAGE AVEC LE MODE

Le filtrage d'images modifie l'apparence d'une image en changeant les couleurs des pixels. L'application de filtres sur des photographies permet d'augmenter le contraste et d'obtenir un certain nombre d'effets intéressants. Il existe plusieurs types de filtrage (Min, Max, moyenne, mode, médiane...). Après une expérience, On a constaté que le meilleur type qui peut nous aider est le mode.



FIGURE 4.6 – Filtrage avec le mode

4.3.2.3 LABELLISATION

Maintenant que nous avons les photos, il va falloir repérer l'emplacement du logo qu'il faudra ensuite retrouver. C'est ce que l'on appelle la phase d'étiquetage. Malheureusement, cette étape n'est pas la plus agréable car vous vous rendrez vite compte qu'elle est longue et désagréable ! Le but est d'identifier et d'annoter les cadres qui comportent un logo (ou plusieurs logos !) pour chacune des photos/images (432 ici).

Il n'y a pas d'autre option, le travail doit être fait manuellement sur chaque image ! Même s'il n'est pas toujours approprié dans un cadre professionnel, LabelImg est vraiment simple à utiliser et nous aidera à créer ce qui suit :

- un fichier texte avec les coordonnées de chaque image contenant le ou les objets à détecter, un fichier par image.
- un fichier contenant les classes ou les bibliothèques. Dans notre cas, nous ne créerons qu'une seule classe :Starbucks.

REALISATION

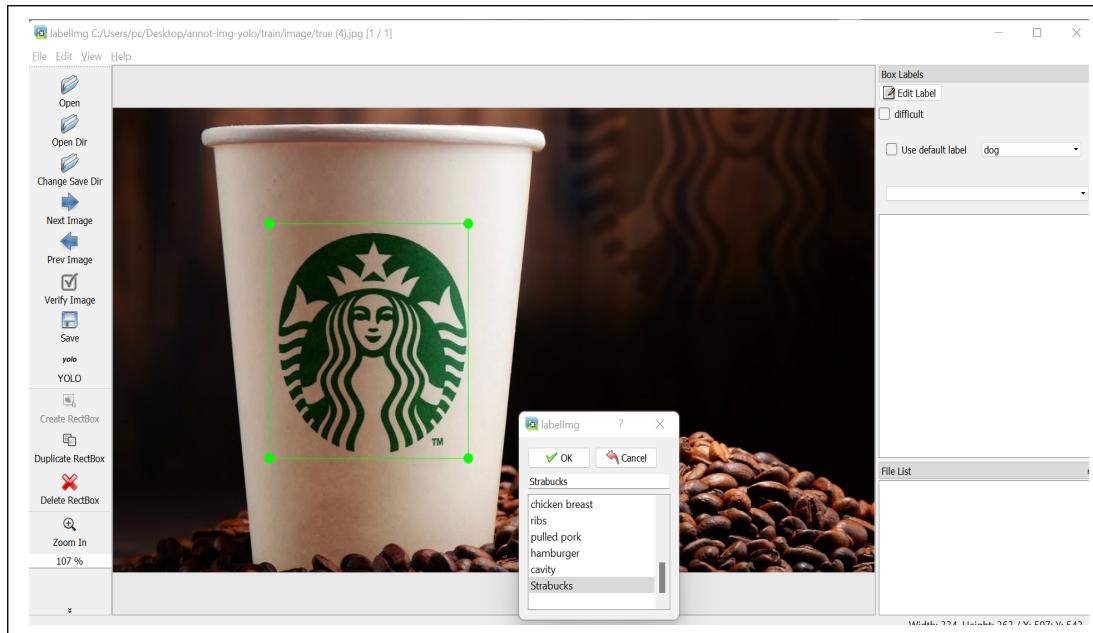


FIGURE 4.7 – LabelImg

Une fois terminé la labellisation de nos images nous trouvons dans le même répertoire, un fichier txt qui porte le même nom que l'image et qui contient les coordonnées du ou des rectangles contenant les objets. Voici un exemple pour une image contenant 4 logos :

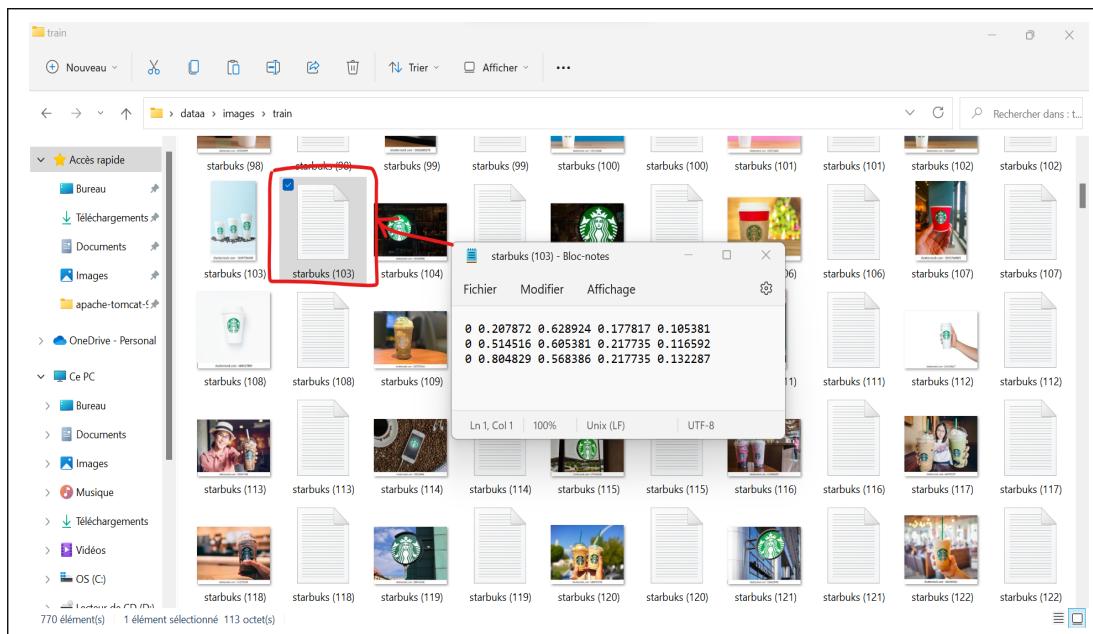


FIGURE 4.8 – Aperçu du dataset

4.3.2.4 DIVISER LES DONNÉES

La dernière étape dans ce module c'est la division des données en deux parties, un pour l'apprentissage (train), et autre pour le test (test), dans notre cas nous avons choisi 80% pour l'apprentissage et 20% pour test.

4.4 APPRENTISSAGE DE MODEL YOLOV5

Cette section contient deux étapes principales :

4.4.1 TÉLÉCHARGER LE MODÈLE YOLOV5

Cette étape consiste à connecter google Colab avec notre google drive, ensuite télécharger le fichier qui contient le model yolov4 avec le command :

```
!git clone https://github.com/ultralytics/yolov5
```

FIGURE 4.9 – Téléchargement de yolov5

Ensuite, nous avons préparé l'enivrement google Colab par la configuration de certain paramètre sont :

- Changer les paramètres du notebook, et choisir le matériel GPU pour le l'apprentissage du modèle.

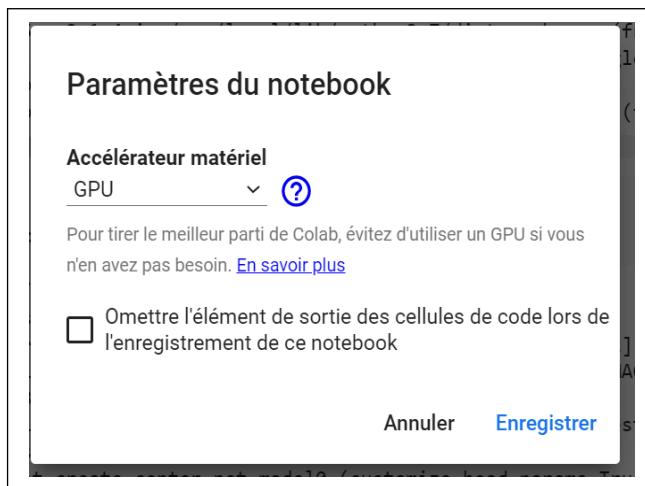


FIGURE 4.10 – Configuration GPU

- Accéder au fichier yolo après, installer les dépendances.

```
%cd yolov5  
%pip install -qr requirements.txt # install
```

FIGURE 4.11 – Configuration

Après la configuration de Colab, nous avons télécharger la base de données dans un dossier appeler « data », ce répertoire contient deux dossiers (images ,labels).

4.4.1.1 PRÉPARER LE MODÈLE POUR LE TRAIN

Dans cette étape, nous allons nous concentrer sur les configurations et nous allons les diviser en trois fichiers YAML, qui sont fournis avec le référentiel lui-même. Nous allons personnaliser ces fichiers pour la tâche à accomplir, afin de répondre à nos besoins souhaités.

le fichier data-configurations décrit les paramètres de l'ensemble de données. Puisque nous nous entraînons sur notre jeu de données pingouin personnalisé, nous allons modifier ce fichier et fournir : les chemins vers les jeux de données d'entraînement, de validation et de test (facultatif) ; le nombre de classes (nc) ; et les noms des classes dans le même ordre que leur index. Dans ce projet, nous n'avons qu'une seule classe, nommée 'Pingouin'. Nous avons nommé notre fichier de configuration de données personnalisées 'custom_data.yaml' et l'avons placé dans le répertoire 'data'. Le contenu de ce fichier YAML est le suivant :

```
1 train: /content/gdrive/MyDrive/dataaa/images/train # train images (relative to 'path')
2 val: /content/gdrive/MyDrive/dataaa/images/val # val images (relative to 'path')
3
4
5 # Classes
6 nc: 1 # number of classes
7 names: ['Starbucks'] # class names
8
9
10
```

FIGURE 4.12 – Fichier data_configuration

Le fichier de configurations de modèle dicte l’architecture du modèle. Ultralytics supporte plusieurs architectures YOLOv5, nommées modèles P5, qui varient principalement par la taille de leurs paramètres : YOLOv5n (nano), YOLOv5s (small), YOLOv5m (medium), YOLOv5l (large), YOLOv5x (extra large). Ces architectures sont adaptées à la formation avec une taille d’image de 640*640 pixels. Série supplémentaire, optimisée pour l’entraînement avec une taille d’image plus grande de 1280*1280, appelée P6 (YOLOv5n6, YOLOv5s6, YOLOv5m6, YOLOv5l6, YOLOv5x6). Les modèles P6 incluent une couche de sortie supplémentaire pour la détection d’objets plus grands. Ils bénéficient le plus d’un entraînement à plus haute résolution et produisent de meilleurs résultats.[23]

Lorsque le training est initialisée à partir de poids pré-formés comme dans ce projet, il n’est pas nécessaire de modifier le fichier de configuration du modèle puisque le modèle sera extrait avec les poids pré-formés.

Le fichier hyperparameters-configurations définit les hyperparamètres pour le training, y compris le taux d’apprentissage, momentum, les pertes, les augmentations, etc. Ultralytics fournit un fichier d’hyperparamètres par défaut sous le répertoire ’data/hyp/hyp.scratch.yaml’. Il est principalement recommandé de commencer l’entraînement avec des hyperparamètres par défaut pour établir une base de performances, comme nous le ferons dans ce projet.

Les fichiers de configuration YAML sont imbriqués dans les répertoires suivants :

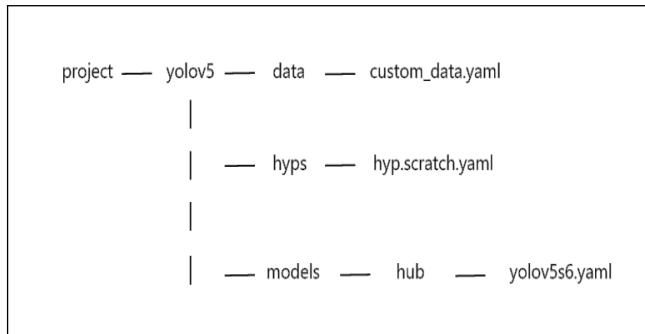


FIGURE 4.13 – Les fichiers de configuration YAML

4.4.2 ENTRAÎNEMENT

Étant donné que mon ensemble de données sur la marque (Starbucks) est relativement petit (432 images), l'apprentissage par transfert devrait produire de meilleurs résultats que la formation à partir de zéro. Le modèle par défaut d'Ultralytic a été pré-formé sur l'ensemble de données COCO, bien que d'autres modèles pré-formés soient également pris en charge (VOC, Argoverse, VisDrone, GlobalWheat, xView, Objects365, SKU-110K). COCO est un ensemble de données de détection d'objets avec des images de scènes quotidiennes. Il contient 80 classes, y compris la classe "oiseau" associée, mais pas la classe "Starbucks". Notre modèle sera initialisé avec les poids d'un modèle COCO pré-entraîné, en passant le nom du modèle à l'argument 'poids'. Le modèle pré-formé sera automatiquement téléchargé.

4.4.2.1 EXTRACTION DE CARACTÉRISTIQUES

Les modèles sont composés de deux parties principales : les couches dorsales qui servent d'extracteur de caractéristiques, et les couches de tête qui calculent les prédictions de sortie. Pour compenser la petite taille de l'ensemble de données, nous utiliserons le même backbone que le modèle COCO pré-entraîné, et n'entraînerons que la tête du modèle. Le backbone de YOLOv5s6 consiste en 12 couches, qui seront fixées par l'argument 'freeze'.

```
!python train.py --batch 32 --epochs 150 --data 'custom_data.yaml' --weights 'yolov5s6.pt' --project 'runs_starbucks' --name 'feature_extraction' --cache --freeze 12
```

- **batch** : définir taille de lot.
- **epochs** : définit le nombre d'époques d'entraînement.

- **data** : chemin d'accès au fichier de configuration des données.
- **weights** : spécifie un chemin vers les poids pour commencer l'apprentissage par transfert. Ici nous choisissons le point de contrôle générique COCO pretrained.
- **project** : nom du projet
- **name** : nom de l'emplacement de l'apprentissage.
- **cache** : images en cache pour un training plus rapide.
- **freeze** : nombre de layers freeze

Après l'entraînement nous pouvons afficher les métriques et les pertes enregistrées dans le fichier 'results.png' :

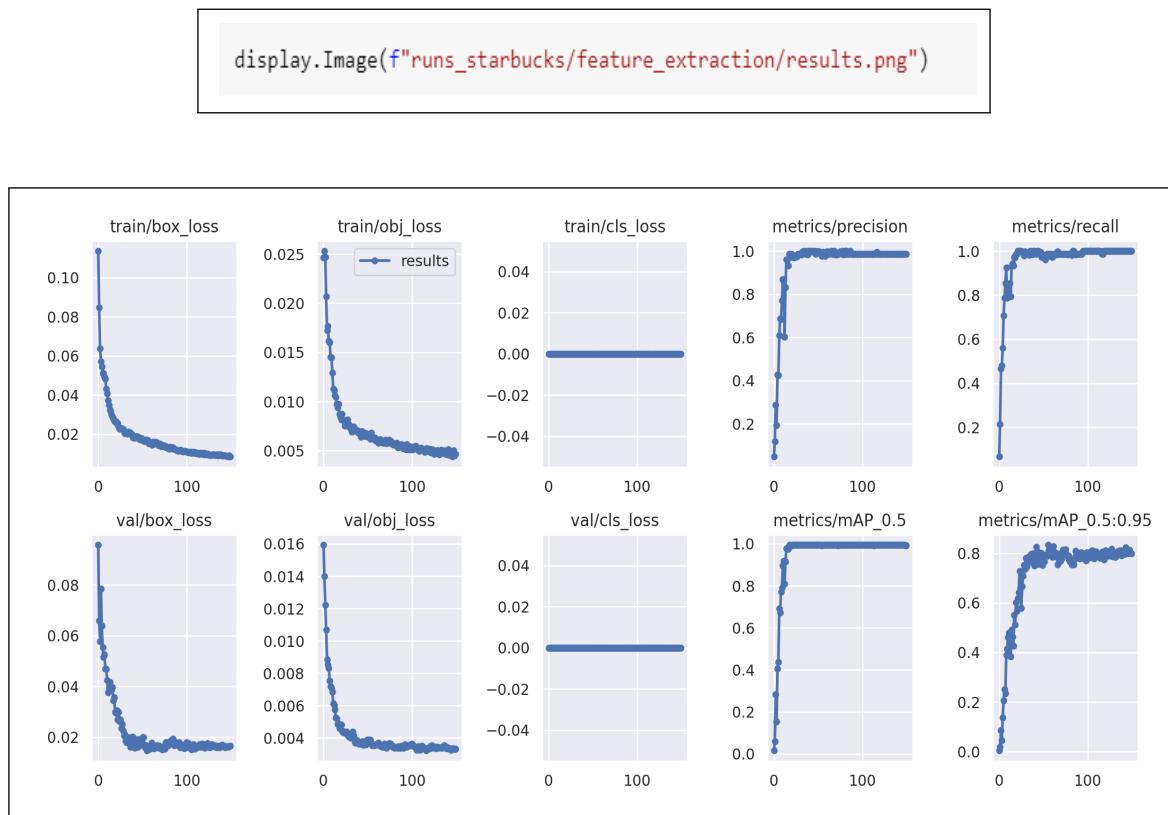


FIGURE 4.14 – Resultats d'extraction de caractéristiques trainig

Pour mieux comprendre les résultats, résumons les pertes et les mesures YOLOv5. La fonction de perte YOLO est composée de trois parties :

- **box_loss** : perte de régression de la boîte englobante (erreur quadratique moyenne).
- **obj_loss** : la confiance de la présence de l'objet est la perte d'objectivité (Binary Cross Entropy).

- **cls_loss** : la perte de classification (Cross Entropy).

Étant donné que nos données n'ont qu'une seule classe, il n'y a pas d'erreur d'identification de classe et l'erreur de classification est constamment nulle.

Precision mesure la quantité de prédictions de bbox correctes (Vrais positifs / (Vrais positifs + Faux positifs)) et le Rappel mesure la proportion de vraies bbox qui ont été correctement prédites (Vrais positifs / (Vrais positifs + Faux négatifs)). 'mAP_0.5' est la précision moyenne moyenne (mAP) au seuil IoU (Intersection over Union) de 0,5. 'mAP_0.5 :0.95' est la mAP moyenne sur différents seuils IoU, allant de 0,5 à 0,95.

4.4.2.2 FINE TUNING

La dernière étape facultative de l'entraînement est le réglage fin(Fine tuning), qui consiste à dégeler l'ensemble du modèle que nous avons obtenu ci-dessus, et à le réentraîner sur nos données avec un taux d'apprentissage très faible. Cela peut potentiellement apporter des améliorations significatives, en adaptant progressivement les fonctionnalités pré-entraînées aux nouvelles données. Le paramètre de taux d'apprentissage peut être ajusté dans le fichier hyperparameters-configurations. Pour le faire, nous adopterons les hyperparamètres définis dans le fichier intégré "hyp.yaml", qui a un taux d'apprentissage beaucoup plus faible que celui par défaut. Les poids seront initialisés avec les poids enregistrés à l'étape précédente.

```
|python train.py --hyp 'runs_starbucks/feature_extraction/hyp.yaml' --batch 16 --epochs 100 --data 'data/custom_data.yaml' --weights 'runs_starbucks/feature_extraction/weights/best.pt' --project 'runs_starbucks' --name 'fine-tuning' --cache
```

- **hyp** : chemin d'accès au fichier de configuration des hyperparamètres

Comme nous pouvons le voir ci-dessous, pendant la phase de réglage fin, les mesures et les pertes continuent de s'améliorer.

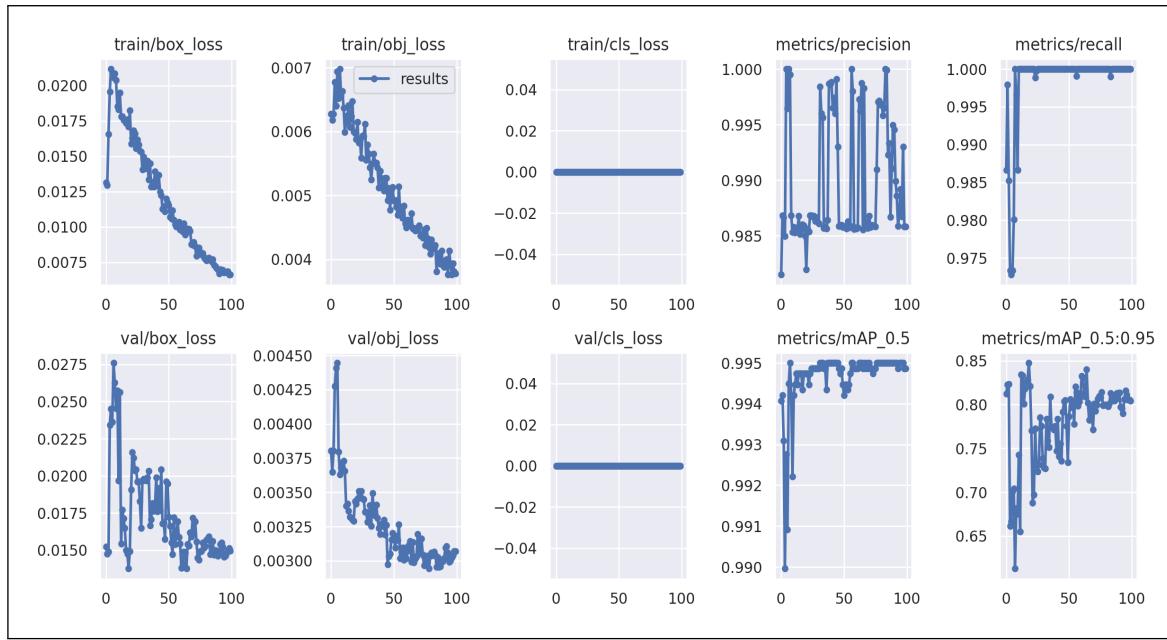


FIGURE 4.15 – Résultats de 'fine tuning' training

4.4.2.3 VALIDATION

Pour évaluer notre modèle, nous utiliserons le script de validation. Les performances peuvent être évaluées sur les fractionnements d'ensembles de données d'entraînement, de validation ou de test, contrôlés par l'argument 'task'. Ici, la division de l'ensemble de données de test est en cours d'évaluation :

```
!python val.py --weights 'runs_starbucks/fine-tuning/weights/best.pt' --batch 64 --data 'data/custom_data.yaml' --task test --project 'runs_starbucks' --name 'validation_on_test_data' --augment
```

On peut également obtenir la courbe Précision-Rappel, qui est automatiquement sauvegardée à chaque validation.

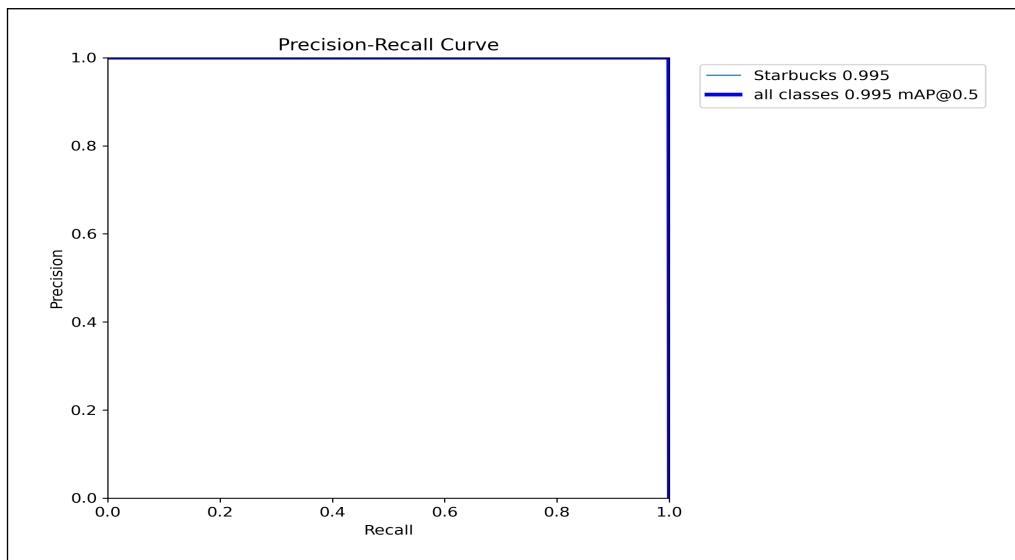


FIGURE 4.16 – Précision — Courbe de rappel de la répartition des données de test

4.4.2.4 INFÉRENCE

Une fois que nous avons obtenu des performances d’entraînement satisfaisantes, notre modèle est prêt pour l’inférence. Lors de l’inférence, nous pouvons encore augmenter la précision des prédictions en appliquant des augmentations de temps de test (TTA) : chaque image est augmentée (retournement horizontal et 3 résolutions différentes), et la prédiction finale est un ensemble de toutes ces augmentations. Si nous sommes serrés sur le taux d’images par seconde (FPS), nous devrons abandonner le TTA car l’inférence avec lui est 2 à 3 fois plus longue.

L’entrée pour l’inférence peut être une image, une vidéo, un répertoire, une webcam, un flux ou même un lien youtube. Dans notre cas, nous avons utilisé le jeu de données de test

```
!python detect.py --weights 'runs_starbucks/fine-tuning/weights/best.pt' --conf 0.6 --source '/content/gdrive/MyDrive/dataaa/images/test' --project 'runs_starbucks' --name 'detect_test' --augment --line=3
```

- **source** : chemin d’entrée.
- **conf** : seuil de confiance.
- **augment** : inférence augmentée (TTA).

Les résultats des inférences sont automatiquement enregistrés dans le dossier défini. Examinons un échantillon des prédictions du test :



FIGURE 4.17 – Résultats de l’inférence

4.4.2.5 EXPORTATION VERS LE FORMAT TFLITE

À présent, notre modèle est terminé et enregistré selon la convention PyTorch commune avec l’extension de fichier '.pt'. Ensuite nous pouvons exporter le modèle vers le format tflite (TensorFlow Lite). TensorFlow Lite est la solution légère de TensorFlow pour les appareils mobiles et intégrés. Il vous permet d’exécuter des modèles d’apprentissage automatique sur des appareils périphériques avec une faible latence, éliminant ainsi le besoin d’un serveur.

Le script 'export.py' est utilisé pour convertir les modèles PyTorch en Tflite ou autres formats, en ajoutant le format du type à l’argument 'include'. La commande suivante est utilisée pour exporter notre modèle de Starbucks vers tflite. Ces nouveaux formats de fichiers sont enregistrés dans le même dossier 'weights' que le modèle PyTorch.

```
!python export.py --weights 'runs_starbucks/fine-tuning/weights/best.pt' --include tflite --data 'custom_data.yaml' --imgsz 640 640
```

Après avoir exporté notre modèle au format 'tflite', nous sommes prêts à le mettre en œuvre dans l’application mobile.

4.5 APERÇU DE L'APPLICATION

Après l'implémentation de notre modèle, nous l'avons sauvegardé afin de l'utiliser sur de nouvelles données, lors de la phase de déploiement. Nous avons ensuite développé une application mobile qui détecte le logo dans l'image et affiche un pourcentage de similarité. Voici un aperçu de l'application.

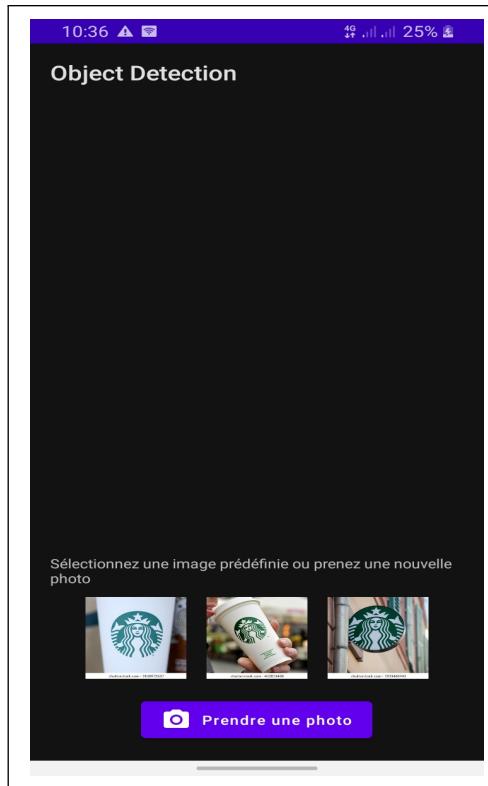


FIGURE 4.18 – Interface de l'application.

Après avoir pris une photo en temps réel, l'application affiche le pourcentage de similarité entre cette photo et notre jeu de données.

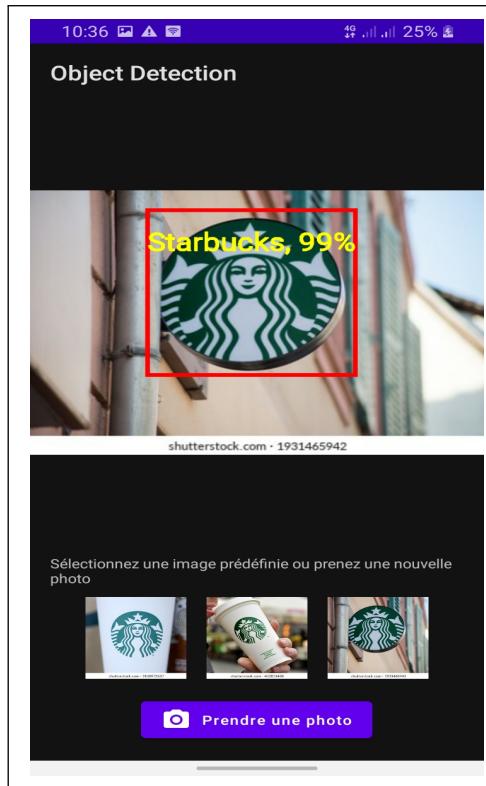


FIGURE 4.19 – Interface de prédiction de l'application.

4.6 CONCLUSION

Nous avons présenté tout au long de ce chapitre, la mise en œuvre de notre approche, la préparation de notre base de données, l'environnement de développement, et terminer avec les interfaces de l'application mobile.

CONCLUSION GÉNÉRALE

Au cours des dernières années, principalement en raison des progrès de l'apprentissage en profondeur, la qualité de la description d'images et de la détection d'objets a progressé à un rythme spectaculaire. La plupart de ces progrès sont le résultat d'un matériel plus puissant, des bases de données (BDD) plus volumineux et de modèles plus grands et une conséquence de nouvelles idées, d'algorithmes et d'architectures réseaux améliorés. « Grâce aux Deep Learning, l'avenir de l'intelligence artificielle est prometteur »

L'une des principales applications de la vision par ordinateur basée sur la technique de l'apprentissage automatique est la détection d'objets. Pendant de nombreuses années, les scientifiques ont cherché des moyens efficaces pour obtenir de bons résultats dans ce domaine. Des méthodes comme Viola et Jones ou HOG ont été la référence pendant de nombreuses années. Elles ont permis un tournant majeur dans la problématique de la détection d'objets. Mais en 2012, le mariage de l'apprentissage profond avec cette discipline, les a détrônés et a mis en avant le CNN.

Cela nous a dirigés, dans ce mémoire, vers le développement du système de détection d'objet comme moyen pour aider les personnes, dans le but de protéger et aider à connaître marques contrefaites. Pour réaliser ce système nous sommes basés sur les réseaux de neurones convolutifs (CNN). Ces derniers sont utiles pour la simulation de n'importe quel problème difficile à décrire avec des modèles physiques et mathématiques en raison de la capacité des réseaux de neurones d'apprendre par des exemples. Nous avons utilisé le modèle CNN yolov5 parce qu'il a prouvé sa performance dans la détection d'objet en temps réel.

Nous allons également poursuivre ce projet pour construire un modèle capable d'identifier toutes les marques et le déployer dans une application mobile payante.

BIBLIOGRAPHIE

- [1] **B.Suganya and A.C.Santha Sheela.** Finding Fake Logo Using CDS Logo Detection And Recognition Algorithm :
<https://www.ijser.org/researchpaper/Finding-Fake-Logo-Using-CDS-Logo-Detection-And-Recognition-Algorithm.pdf>
- [2] **A. Mercier.** L'information face à l'intelligence artificielle : promesses et dangers :
<https://larevuedesmedias.ina.fr/linformation-face-lintelligence-artificielle-promesses-et-dangers>
- [3] **A. Géron** « Machine Learning avec Scikit-Learn », Dunod : Paris, 2017.
- [4] **M.Taffar**, « Initiation à l'apprentissage automatique », Master Course, MENT Informatique - Faculté des Sciences Exactes et Informatique..
- [5] **Y. Benzaki**, « Machine Learning made easy »,
<https://mrmint.fr/>, 2019.
- [6] **J. B. Metomo**, « Machine Learning : Introduction à l'apprentissage automatique »
<https://www.supinfo.com/articles/single/6041-machine-learning-introduction-apprentissageautomatique>, 10 octobre 2017.
- [7] **I. Bellin, N. Vayatis, J. Audiffren, S. Peignier A. Nicolai** « Apprentissage par renforcement »,
<https://dataanalyticspost.com/Lexique/apprentissage-par-renforcement>, Fevrier 2019.
- [8] **M. Zaffagni**, « Cette IA a appris à conduire une voiture autonome en 20 minutes », <https://www.futura-sciences.com/tech/actualites/intelligence-artificielle-cette-ia-appris-conduire-voiture-autonome-20-minutes-71942/>, 09 Juillet 2018.

- [9] **J. Brownlee**, « Difference Between Classification and Regression in Machine Learning », <https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/>, décembre 2017.
- [10] « **Les réseaux de Neurones Artificiels** », Cours, Université de M'SILA, Faculté des Sciences et Sciences de l'Ingénieur, Département d'électrotechnique, 2006.
- [11] **V. Mathivet**, « L'Intelligence Artificielle pour les développeurs », Eni : France, Décembre 2014.
- [12] **V. Mathivet**, « L'Intelligence Artificielle pour les développeurs », Eni : France, Décembre 2014.
- [13] **A. Azencott**, « <https://openclassrooms.com/fr/courses/4470406-utilisez-des-modeles-supervises-non-lineaires/> », 31 Mai 2019.
- [14] **M. Parizeau**, « Le perceptron multicouche et son algorithme de rétropropagation des erreurs », Université Laval, Département de génie électrique et de génie informatique, Septembre 2004.
- [15] **Y. LeCun**, « Les Enjeux de la Recherche en Intelligence Artificielle », Chaire Informatique et Sciences Numériques Collège de France, AU : 2015-2016.
- [16] **M. Tual**, « Comment le deep learning révolutionne l'intelligence artificielle », <https://www.lemonde.fr/pixels/article/2015/07/24/comment-le-deep-learning-revolutionne-l-intelligence-artificielle-4695929-4408996.html>, 24 juillet 2015.
- [17] **M. D. Youcef**, « Deep Learning pour la classification des images », Mémoire de Master, Université Abou Bakr Belkaïd Tlemcen, Faculté des sciences, Département informatique, 2017.
- [18] « Neurones à Convolution 3 choses à savoir », <https://fr.mathworks.com/solutions/deep-learning/convolutional-neural-network.html>, 2019.
- [19] [24] P. Monasse, K. Nadjahi, « Classez et segmentez des données visuelles», <https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles>, Mai 2019.

BIBLIOGRAPHIE

- [20] **S. Soudoplatoff**, « L'intelligence artificielle : l'expertise par tout accessible à tous », fondaopol : Paris, Février 2018.
- [21] « Google cars self-drive to Walmart supermarket in trial », <https://www.bbc.com/news/technology-44957251>, 25 Juillet 2018.
- [22] **F. Fauconnier**, « Le magasin sans caisse Amazon Go s'ouvre au public », <https://www.lsa-conso.fr/le-magasin-sans-caisse-amazon-go-s-ouvre-au-public,278202>, 22 Janvier 2018.
- [23] <https://zenodo.org/record/#.Y0rKaXbMK5d>
- [24] <https://research.google.com/colaboratory/faq.html?hl=fr>
- [25] <https://blog.roboflow.com/labelimg/>
- [26] <https://docs.python.org/fr/3/library/index.html>
- [27] <https://fr.wikipedia.org/wiki/Pandas>
- [28] <https://fr.wikipedia.org/wiki/NumPy>
- [29] <https://keras.io/>
- [30] <https://www.tensorflow.org/>
- [31] <https://fr.wikipedia.org/wiki/Pillow>
- [32] <https://fr.wikipedia.org/wiki/pytorch>
- [33] <https://fr.wikipedia.org/wiki/flutter>
- [34] MLK« 6 Different Types of Object Detection Algorithms in Nutshell», <https://machinelearningknowledge.ai/different-types-of-object-detection-algorithms>
- [35] **Object Detection Using OpenCV YOLO | Great Learning** <https://www.mygreatlearning.com/blog/yolo-object-detection-using-opencv>.
- [36] Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M. : « YOLOv4 : Optimal Speed and Accuracy of Object Detection ». arXiv 2020. arXiv preprint. arXiv :2004.10934. pp. 2–7 (2020).
- [37] **R. Girshick et autre** « Rich feature hierarchies for accurate object detection and semantic segmentation » IEEE Conference on Computer Vision and Pattern Recognition, 2014

- [38] **R. Girshick** « Fast R-CNN », Conférence internationale de l'IEEE sur la vision par ordinateur, 2015.
- [39] **S. Ren, K. He, R. Girshick, and J. Sun.** « Faster R-CNN : Towards real-time object detection with region proposal networks ».
- [40] **K. He, G. Gkioxari, P. Doll ar, and R. Girshick** « Mask R-CNN» dans les Actes Conférence internationale de l'IEEE sur la vision par ordinateur ICCV ;
- [41] **Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Fu, C. et Berg, A. C.**, «SSD : Single Shot MultiBox Detector » Preprint sur <https://arxiv.org/abs/1512.02325>,
- [42] **J. Redmon, S. Divvala, R. Girshick, and A. Farhadi.** « You only look once : Unified, real-time object detection ». Preprint sur arXiv :1506.02640.
- [43] **H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese.** Generalized intersection over union : A metric and a loss for bounding box regression. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 659
- [44] **Selecting the Right Bounding Box Using Non-Max Suppression (with implementation)** <https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression -with-implementation>
- [45] **Joseph Redmon and Ali Farhadi.** « YOLO9000 : better, faster, stronger ». Dans la conférence de l'IEEE sur la vision par ordinateur et la reconnaissance des formes., CVPR 2017, Honolulu, HI, USA, pages 2-6.
- [46] **Joseph Redmon and Ali Farhadi.** « YOLOv3 : An incremental improvement ». Preprint sur arXiv :1804.02767, 2018
- [47] **Huang, G., et al.** (2017). Densely connected convolutional networks. Proceedings of the IEEE conference on computer vision and pattern recognition.
- [48] **Wang, C.-Y., et al.** (2020). CSPNet : A new backbone that can enhance learning capability of CNN. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops.
- [49] **Huang, Z., et al.** (2020). "DC-SPP-YOLO : Dense connection and spatial pyramid pooling based YOLO for object detection." Information Sciences 522 : 241-258.

BIBLIOGRAPHIE

- [50] Chen, P.-Y., et al. (2019). Smaller object detection for real-time embedded traffic flow estimation using fish-eye cameras. 2019 IEEE International Conference on Image Processing (ICIP), IEEE.
- [51] .Liu, S., et al. (2018). Path aggregation network for instance segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition.
- [52] <https://viso.ai/deep-learning/object-detection/>
- [53] **R. Datta, D. Joshi, J. Li, and J. Z. Wang**, "Image retrieval : Ideas, influences, and trends of the new age," ACM Comput. Surv., vol. 40, No. 2, pp. 1–60, 2008.
- [54] **L. Ballan, M. Bertini, A. Del Bimbo, L. Seidenari, and G. Serra**, "Event detection and recognition for semantic annotation of video," Multimedia Tools Appl., vol. 51, no. 1, pp. 279–302, 2011.
- [55] **Y. Jing and S. Baluja**, "Pagerank for product image search," in Proc.WWW, Beijing, China, 2008, pp. 307–316.
- [56] **L. Ballan, M. Bertini, and A. Jain**, "A system for automatic detection and recognition of advertising trademarks in sports videos," in Proc. ACM Multimedia, Vancouver, BC, Canada, 2008, pp. 991–992.