



Université Mohamed Khider de Biskra  
Faculté des Sciences et de la Technologie  
Département de Génie Electrique

# MÉMOIRE DE MASTER

Sciences et Technologies  
Electronique  
Réseaux Télécommunication

Réf. : Entrez la référence du document

---

Présenté et soutenu par :  
**Trad Housseem Eddine**

Le : mercredi 30 septembre 2020

## La détection d'objet avec OpenCV et deep learning

---

### Jury :

M.me ZEHANI SORAYA	MCB	Université de Biskra	Président
M.me BARKAT AICHA	MAA	Université de Biskra	Examineur
M.me MEDOUAKH SAADIA	MCB	Université de Biskra	Rapporteur

الجمهورية الجزائرية الديمقراطية الشعبية  
République Algérienne Démocratique et Populaire  
وزارة التعليم العالي و البحث العلمي  
Ministère de l'enseignement Supérieur et de la recherche scientifique



Université Mohamed Khider Biskra  
Faculté des Sciences et de la Technologie  
Département de Génie Electrique  
Filière : Electronique  
Option : Réseaux et Télécommunication

Mémoire de Fin d'Etudes  
En vue de l'obtention du diplôme:

**MASTER**

# *Thème*

La détection d'objet avec OpenCV et deep  
learning

Présenté par :

**TRAD Housseem Eddine**

Avis favorable de l'encadreur :

**MEDOUAKH *Sâadia***

**Avis favorable du Président du Jury**

**ZEHANI SORAYA**

**Cachet et signature**

الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

وزارة التعليم العالي و البحث العلمي

Ministère de l'enseignement Supérieur et de la recherche scientifique



**Université Mohamed Khider Biskra**  
**Faculté des Sciences et de la Technologie**  
**Département de Génie Electrique**  
**Filière : Electronique**  
**Option : Réseaux et télécommunication**

# *Thème :*

La détection d'objet avec OpenCV et deep learning

**Proposé par : MEDOUAKH *Sâadia***

**Dirigé par : MEDOUAKH *Sâadia***

## **RESUMES (Français et Arab et Anglais)**

### **Résumé**

Les méthodes d'apprentissage en profondeur, en particulier les réseaux de neurones convolutifs (CNN), ont remporté un grand succès dans le domaine de la vision par ordinateur, et ce succès montre une grande supériorité sur les algorithmes de détection d'objets et lui donne la vitesse et la précision de détection, en particulier dans la surveillance en temps réel. Le but de ce travail est d'étudier et d'appliquer un algorithme de détection d'objets soumis à un apprentissage en profondeur. Nous nous sommes concentrés dans cette étude sur l'algorithme de détection d'objet SSD, qui était la méthode approuvée dans le côté pratique. Ainsi que, nous avons obtenu des résultats très satisfaisants, avec un aperçu qui exprime la grande supériorité de l'algorithme de détection d'objet SSD sur la méthode traditionnelle Haar.

**Mots clés :** Détection d'objets, Réseaux de neurones convolutifs (CNN), Apprentissage en profondeur (Deep Learning), SSD, Haar

# **Remerciements**

*Nous remercions tout d'abord **ALLAH** de nous avoir donné le courage d'accomplir ce travail.*

*Je tiens à exprimer mes remerciements à mon encadreur*

*Mme **MEDOUAKH** Sâadia*

*De m'avoir soutenue et fait confiance durant mon projet avec une Grande patience.*

*Avec son expérience dans la recherche et l'enseignement, avec ses conseils,*

*j'ai pu découvrir le monde de la recherche*

*scientifique dans le domaine du traitement d'image*

*Mes remerciements et ma profonde reconnaissance s'adressent à mesdames les membres de jury :*

*Présidente de jury, Mme **ZEHANI** Soraya, d'avoir acceptée de présider le jury de soutenance.*

*J'exprime également mes vifs remerciements, et ma profonde reconnaissance à Mme **BARKAT** Aicha d'avoir acceptée d'examiner ce mémoire*

*En second lieu, je remercie chaleureusement mes chers parents,*

*Mon père (Que Dieu lui fasse miséricorde et l'accueille dans son vaste paradis), ma mère, mes frères, et ma petite sœur pour leurs sacrifices, aides, soutiens et encouragements et à tous ceux qui de près ou de loin ont*

*contribués au bon déroulement de ce mémoire.*

*Je souhaite à présent adresser mes sincères remerciements*

*à toutes les personnes avec qui j'ai eu la chance*

*de travailler et à qui j'ai eu l'honneur de*

*côtoyer avant et pendant mon mémoire, et à tous les*

*enseignants, intervenants de l'Université Mohamed Khider BISKRA.*

# Dédicaces

*Je dédie ce travail,  
Fruit de nombreuses années d'étude à :  
à mes chers  
parents pour leur patience,  
**Ma chère mère.** Merci pour tes conseils, tes  
Sacrifices, ton soutien et tes encouragements  
**Papa** ma Gratitude ne suffit pas à exprimer ce  
qu'il mérite pour tout ses sacrifices depuis ma naissance,  
pendant mon  
enfance et même à l'âge  
adulte (Que Dieu lui fasse miséricorde et l'accueille dans son vaste  
paradis)  
À mes frères **ALLA El Dine, wassim,**  
À ma petite Sœur **Anfel.**  
À Tous mes enseignants.*

À tous mes amis et spécialement **Ziad,** qui m'ont soutenu  
dans l'accomplissement de cet  
Humble travail  
À tous mes professeurs et à tous ceux qui se sont  
engagés dans ces modestes travaux  
À tout ma famille mes oncles et mes tantes.  
Et À tous qui m'ont aide  
de près ou de loin pour la réalisation de ce travail.

*Houssem*

# Liste des tableaux

## *Chapitre 2 La détection d'objet par Deep Learning*

**Tableau 2.1 :** l'amélioration des performances après l'augmentation des données.....38

## *Chapitre 3 Implémentation et résultats*

**Tableau 3.1:** La BDD de Détection d'objets.....43

**Tableau 3.2 :** Résultats de détection d'objets en utilisant SSD sur quelques images.....51

# Liste des figures

## *Chapitre 1 Etat de l'Art sur la détection d'objets*

<b>Figure 1.1</b> : Principe de la détection d'objet.....	5
<b>Figure 1.2</b> : Exemples des domaines d'applications.....	5
<b>Figure 1.3</b> : Exemple des défient défis de détection.....	6
<b>Figure 1.4</b> : La chaine d'exécution de la méthode de «Viola&Jones ».....	8
<b>Figure 1.5</b> : Les descripteurs de Haar (Haar features).....	9
<b>Figure 1.6</b> : Modèle du cascade classifieurs.....	10
<b>Figure 1.7</b> : Exemple sur la détection d'objets par Viola et Jones.....	10
<b>Figure 1.8</b> : Erreur Top-5 sur IMAGENET.....	11
<b>Figure 1.9</b> : Principales catégories de détecteurs d'objet.....	11
<b>Figure 1.10</b> : Modèle Détecteurs à deux étages.....	12
<b>Figure 1.11</b> : Modèle Détecteurs à un étage.....	12
<b>Figure 1.12</b> : Architecture standard d'un réseau à convolutions.....	13
<b>Figure 1.13</b> : l'étape de convolution.....	14
<b>Figure 1.14</b> : Max pooling modèle (prend la plus grande valeur de chaque fenêtre).....	14
<b>Figure 1.15</b> : La partie du Couche entièrement connectée (fully connected).....	15

## *Chapitre 2 La détection d'objet par Deep Learning*

<b>Figure 2.1</b> : Le principe du mathématicien britannique 'Alan Turing'.....	18
<b>Figure 2.2</b> : Le père de l'intelligence artificielle « John McCarthy ».....	18
<b>Figure 2.3</b> : Le projet de L'expérience du chat par (Deep Learning).....	21
<b>Figure 2.4</b> : Un aperçu des performances récentes de détection d'objets.....	22
<b>Figure 2.5</b> : Modèles des réseaux neuronaux de Deep Learning.....	23
<b>Figure 2.6</b> : Les différents domaines d'application de Deep Learning.....	25
<b>Figure 2.7</b> : chronologique des méthodes de détection d'objets.....	25
<b>Figure 2.8</b> : Comparaison de la vitesse de test des algorithmes de détection d'objets.....	26
<b>Figure 2.9</b> : Illustration du structure de détection Faster- RCNN (Girshick et al 2015).....	27
<b>Figure 2.10</b> : L'architecture du réseau de proposition de région « RPN ».....	28
<b>Figure 1.11</b> : illustration du structure de la méthode YOLO (Redmon and Farhadi, 2015)..	29

<b>Figure 2.12</b> : Comment YOLO gère les boîtes englobantes.....	29
<b>Figure 2.13</b> : Précision (mAP) avec le temps d'inférence.....	30
<b>Figure 2.14</b> : Architecture de réseau SSD.....	31
<b>Figure 2.15</b> : Modèle VGG16.....	31
<b>Figure 2.16</b> : Les 4 prédictions d'objets.....	32
<b>Figure 2.17</b> : Présentation des performances des différents modèles d'architecture CNN dans la tâche de classification.....	33
<b>Figure 2.18</b> : Les perspectives multi-échelles des objets détectés.....	34
<b>Figure 2.19</b> : Boîtes par défaut de SSD aux cartes de caractéristiques 8x8 et 4x4.....	35

### ***Chapitre 3 Implémentation et résultats***

<b>Figure 3.1</b> : La convolution du MobileNet.....	40
<b>Figure 3.2</b> : Le logo du python.....	41
<b>Figure 3.3</b> : Le logo d'OpenCV.....	42
<b>Figure 3.4</b> : Installation de Python 3.6.1 .....	44
<b>Figure 3.5</b> : Environnement Pycharm.....	45
<b>Figure 3.6</b> : Création de l'environnement sous pycharm.....	45
<b>Figure 3.7</b> : Installation d'OpenCV par la commande pip.....	46
<b>Figure 3.8</b> : Test de fonctionnement d'OpenCV.....	46
<b>Figure 3.9</b> : Installation de Numpy par la commande pip.....	46
<b>Figure 3.10</b> : Test de fonctionnement de Numpy .....	47
Figure 3.11 : Test de fonctionnement d'argparse .....	47
<b>Figure 3.12</b> : Les bibliothèques d'importation.....	49
<b>Figure 3.13</b> : Arguments de ligne de commande.....	49
<b>Figure 3.14</b> : La liste des étiquettes de classes.....	49
<b>Figure 3.15</b> : Charge le modèle à utiliser.....	50
<b>Figure 3.16</b> : Création du blob et envoi dans le réseau afin d'avoir une prédiction.....	50
<b>Figure 3.17</b> : Comparaison de la méthode SSD avec la méthode Haar.....	55

## Liste des Abréviations

**DL:** Deep Learning

**ML:** Machine Learning

**SVM:** Support Vector Machines

**HOG :** Histogrammes de Gradient Orienté

**CNN :** Convolutional Neural Network

**RCNN :** Region Convolutional Neural Network

**BDD :** Base De Donnée

**AI :** Artificial Intelligence

**CV :** Computer Vision

**FPS :** Frame Per Second

**FC:** Fully Connected

**SSD:** Single Shot MultiBox Detector

**YOLO:** You Only Look Once

**MAP :** Mean Average Precision

**RPN :** Réseau de Proposition de Région

**HDD :** Hard Disk Drive

## Résumé

Les méthodes d'apprentissage en profondeur, en particulier les réseaux de neurones convolutifs (CNN), ont remporté un grand succès dans le domaine de la vision par ordinateur, et ce succès montre une grande supériorité sur les algorithmes de détection d'objets et lui donne la vitesse et la précision de détection, en particulier dans la surveillance en temps réel. Le but de ce travail est d'étudier et d'appliquer un algorithme de détection d'objets soumis à un apprentissage en profondeur. Nous nous sommes concentrés dans cette étude sur l'algorithme de détection d'objet SSD, qui était la méthode approuvée dans le côté pratique. Ainsi que, nous avons obtenu des résultats très satisfaisants, avec un aperçu qui exprime la grande supériorité de l'algorithme de détection d'objet SSD sur la méthode traditionnelle Haar.

**Mots clés :** Détection d'objets, Réseaux de neurones convolutifs (CNN), Apprentissage en profondeur (Deep Learning), SSD, Haar

## ملخص :

حققت اساليب التعلم العميقة، لا سيما الشبكات العصبية التلافيفية، نجاحا كبيرا في مجال رؤية الكمبيوتر، و يظهر هذا النجاح تفوقا شاسعا علي خوارزميات الكشف عن الاشياء و يمنح لها سرعة ودقة الكشف، خاصة في المراقبة في الوقت الحالي. الهدف من هذا العمل هو دراسة و تطبيق خوارزمية الكشف عن الأشياء الخاضعة للتعلم العميق. لقد قمنا بالتركيز في هذه الدراسة على خوارزمية اكتشاف الكائن SSD والتي كانت الطريقة المعتمدة في الجانب العملي. أيضاً، لقد حصلنا على نتائج مرضية للغاية، مع نظرة ثاقبة تعبر عن التفوق الكبير لخوارزمية اكتشاف الكائن SSD على طريقة Haar التقليدية.

**الكلمات الدلالية:** الكشف عن الأشياء، الشبكات العصبية التلافيفية، التعلم العميق، SSD، Haar.

## Abstract

Deep learning methods, especially convolutional neural networks (CNN), have been very successful in the field of computer vision, and this success shows great superiority over object detection algorithms and gives it speed and detection accuracy, especially in real-time monitoring. The aim of this work is to study and apply an algorithm for the detection of objects subjected to deep learning. We focused in this study on the SSD object detection algorithm, which was the approved method in the practical side. In addition, we have obtained very satisfactory results, with insight that expresses the great superiority of the object detection algorithm (SSD) over the traditional method (Haar).

**Keywords:** object detection, Convolutional neural networks (CNN), Deep learning, SSD, Haar.

# Tableau des matières

<b>REMERCIEMENT.....</b>	<b>I</b>
<b>DEDECACES.....</b>	<b>II</b>
<b>LISTE DES TABLEAUX.....</b>	<b>III</b>
<b>LISTE DES FIGURES.....</b>	<b>V</b>
<b>LISTE DES ABREVIATION.....</b>	<b>VI</b>
<b>RESUME.....</b>	<b>VII</b>
<b>TABLEAUX DES MATIERES.....</b>	<b>X</b>
<b>INTRODUCTION GENERALE.....</b>	<b>2</b>
<b>Chapitre 1 : Généralités sur la détection des objets.....</b>	<b>3</b>
1.1 Introduction .....	4
1.2 La détection d'objets .....	4
1.2.1 Le principe de la détection d'objets.....	4
1.2.2 Domaine d'application .....	5
1.2.3 Les defis de la détection d'objets.....	6
1.3 Etat de l'Art sur la détection d'objets.....	7
1.3.1 La méthode de détection d'objets par la méthode de Viola et Jones.....	7
1.3.1.1 Dèfinition.....	7
1.3.1.2 Les quatre principes clés de la méthode.....	8
1.3.1.3 Les Caractéristiques rectangulaires.....	8
1.3.1.4 L'image intégrale .....	9
1.3.1.5 L'algorithme AdaBoost .....	9
1.3.1.6 Le cascade classifieurs.....	9
1.4 Etat de l'Art sur la détection d'objets basés sur le Deep learning.....	10
1.4.1 Détecteurs à deux étages .....	11
1.4.2 Détecteurs à un étage.....	12
1.4.3 Convolution neural network (CNN).....	12
a. Couche de convolution .....	13
b. Couche de Pooling .....	14
c. La couche entièrement connectée (Fully-Connected).....	14
1.5 Conclusion.....	15
<b>Chapitre 2 : La détection d'objet par Deep Learning.....</b>	<b>16</b>
2.1 Introduction.....	17
2.2 Qu'est-ce que le machine learning ? .....	17

2.3 Qu'est-ce que l'apprentissage ?.....	18
2.3.1 Types d'apprentissage.....	19
2.3.1.1 L'apprentissage supervisé.....	19
2.3.1.2 L'apprentissage non-supervisé.....	19
2.4 Le Deep Learning .....	20
2.4.1 Concept .....	20
2.4.2 L'avantage du Deep Learning.....	21
2.4.3 Les différentes Architectures du Deep Learning .....	22
2.4.4. Application du Deep Learning.....	24
2.5 La détection d'objets basée sur Deep Learning.....	25
2.5.1 Faster R-CNNs .....	26
2.5.1.1 Réseau de proposition de région (RPN).....	27
2.5.2 You Only Look Once (YOLO) .....	28
2.5.3 Single Shot Detectors (SSDs) .....	29
2.5.3.1 Structure réseau du SSD.....	30
2.5.3.2. Modèle.....	33
a. Cartes de caractéristiques multi-échelles pour la détection.....	33
b. Prédicteurs convolutifs pour la détection.....	34
c. Boîtes par défaut et ratios d'aspect.....	34
2.5.3.3 Entraînement.....	35
a. Stratégie de correspondance.....	35
b. Objectif de l'Entraînement.....	36
c. Choix des échelles et ratios d'aspect pour les Boîtes par défaut.....	36
2.5.3.4 Extraction negative dure (Hard negative mining).....	37
2.5.3.5 Augmentation des données.....	37
2.6 Conclusion.....	38
<b>Chapitre 3 : Implémentation et résultats.....</b>	<b>39</b>
3.1 Introduction.....	40
3.2 Qu'est-ce qu'un MobileNet-SSD ?.....	40
3.3 Langage de programmation.....	41
3.3.1 Python.....	41
3.3.1.1 Pour quoi choisir python ?.....	41
3.3.1.2 Numpy.....	42
3.4 Bibliothèques OpenCV .....	42

3.5 Base de données (BDD).....	43
3.6 Environnement de travail.....	43
3.6.1 Environnement matériel ou le hardware .....	43
3.6.2 Environnement immatériel ou le software .....	44
3.6.2.1 Préparation de l'environnement .....	44
3.6.2.2 Installation de pycharm .....	44
3.6.2.3 Création de l'environnement sous pycharm.....	45
3.6.3 Installation des librairies.....	46
3.7 Détection d'objets par SSD.....	47
3.7.1 OpenCV DNN Module.....	47
3.7.2 Le caffe Model .....	48
3.7.3 Implémentation et résultats expérimentaux .....	48
3.8 Comparaison de la méthode SSD avec la méthode Haar.....	55
3.9 Conclusion .....	56
<b><i>Conclusion General</i></b> .....	<b>57</b>
<b><i>REFERENCE</i></b> .....	<b>64</b>

# Introduction général

## 1) Contexte

Depuis des années la détection d'objet été un sujet de recherche dans le domaine de la vision par ordinateur. Aujourd'hui, avec l'augmentation continue de l'utilisation de la détection d'objets dans des plusieurs applications, il est devenu nécessaire de développer des méthodes plus capable pour assurer toujours la précision et la rapidité.

La détection d'objets basée sur l'apprentissage en profondeur (Deep Learning) est bonne en termes de robustesse par rapport à la détection traditionnelle. Cependant, les modèles profonds sont difficiles à former. Les travaux en cours fournissent une perspective pour la recherche dans ce domaine, ce qui ouvre la concurrence devant ses méthodes. [1]

Le but de ce travail est la détection d'objets avec un réseau de base (MobileNet) qui est associé au réseau de neurones Single Shot Multibox Detector (SSD), cela nous permet d'obtenir une méthode rapide et efficace basée sur le Deep Learning.

## 2) Problématique

La détection d'objet et tous les systèmes de détection ont deux limitations principales :

Une limitation en termes de **précision**, et une limitation en termes du **temps**.

L'obtention des hautes performances pour un système de détection d'objet dans le monde réel est un problème ouvert pour les chercheurs depuis quelques années.

Malheureusement, les conditions non contrôlées telles que les Changement d'illumination, les conditions météorologiques, l'occultation partielles ou totales, les objets en mouvement rapide, mouvement rapide de caméra, l'ombrage. Alors que ces systèmes sont sensibles dans l'environnement réel non contrôlé.

### 3) Motivation

Dans le monde de la détection d'objet, **Google** ouvre sa technologie de détection d'objets et donne un coup de boost à la reconnaissance d'image sur smartphone.

Google met maintenant à la disposition du public une API délivrant la même technologie de détection d'objets que celle exploitée par *Street View* et Google Images. L'intelligence artificielle gagne toujours plus de terrain sur les smartphones. Google vient de livrer aux développeurs, via sa plateforme open source dédiée au Machine Learning, l'une de ses précieuses technologies : celle qui permet à son moteur de recherche d'images, à ses caméras *Nest* ou encore à son logiciel *Street View* d'identifier de multiples objets dans une seule image.

Google n'est pas le seul à vouloir développer ce domaine, d'autre géant de la technologie a également investi le terrain : Facebook avec ses Framework aussi.

### 4) Structure de mémoire

Ce mémoire se présente sous forme de trois chapitres :

**Le premier chapitre** donne une présentation générale sur la détection d'objet et ses Domaines d'application, en présentant les diverses méthodes de détection de Deep Learning.

**Ensuite Le chapitre 2** est consacré à l'étude des différentes architectures des méthodes de Deep learning à partir de type d'apprentissage et les différentes architectures du Deep Learning qui sont basés sur le CNN, en particulier la méthode SSD.

**Le chapitre 3** nous présentons les résultats expérimentaux obtenus par la méthode MobileNet SSD ainsi que des commentaires avec interprétations des résultats.

Nous terminerons ce mémoire par une conclusion générale et les perspectives.

# **Chapitre 1 : Généralités sur la détection des objets**

## **1.1 Introduction**

De nos jours, le monde devient plus intelligent et la technique de traitement d'image est fournie par le monde entier, comme la surveillance par caméra, le suivi d'objets, la détection et la classification des objets...etc. Mais, la détection d'objets est la partie la plus cruciale et particulière dans le traitement d'image. Les avantages de cette technique sont que nous pouvons l'utiliser à de nombreux objectifs pour faire nos travaux, afin qu'il devienne plus facile et plus rapide que l'homme [2]. Dans ce chapitre, nous fournissons les différents méthodes de détection et de classifications d'objet à partir de la première méthode qui a été proposée. Ensuite, nous allons concentrer sur les types des méthodes de l'apprentissage en profondeur (DL).

## **1.2 La détection d'objets**

La détection d'objets est l'un des domaines de recherche les plus actifs dans le domaine de la vision par ordinateur (CV). Où elle implique à la fois la classification de chaque objet dans l'image et sa localisation. Historiquement, la détection d'objets est apparue en 2001 lorsque Paul Viola et Michael Jones ont eu l'idée de Haar Cascades [3]. Aujourd'hui, avec l'augmentation continue de l'utilisation de la détection d'objets dans plusieurs applications telles que la vidéosurveillance, la robotique, l'auto conduite, etc... il est devenu nécessaire de développer des systèmes plus précis et plus rapide [4]. Les nouveaux algorithmes continuent de surpasser les anciens en termes de vitesse et de précision.

### **1.2.1 Le principe de la détection d'objets**

Le principe de la détection d'objets est le suivant : pour une image donnée, on recherche les régions de celle-ci qui pourraient contenir un objet puis pour chacune de ces régions découvertes, on l'extrait et on la classe à l'aide d'un modèle de classification d'image. Les régions de l'image d'origine ayant de bons résultats de classification sont conservés et les autres jetés. Ainsi, pour avoir une bonne méthode de détection d'objets, il est nécessaire d'avoir un algorithme solide de détection des régions et un bon algorithme de classification d'images.

[5].

La détection d'objets est une technologie informatique qui détermine la localisation et la taille de l'objet dans l'image numérique. Il détecte les caractéristiques des objets et ignore tout le reste. La sélection des objets est effectuée par classification à l'aide d'un modèle de base des données.



Figure 1.1 : Principe de la détection d'objet

1.2.2 Domaine d'application

Récemment plusieurs études ont démontré l'efficacité de la détection d'objets et son un rôle qui est très important dans divers domaines et large éventail, notamment :

- ❖ Robotique
- ❖ L'analyse d'images médicales
- ❖ La vidéosurveillance
- ❖ Militaire
- ❖ La voiture autonome
- ❖ Et la détection de la somnolence des conducteurs sur l'autoroute afin d'éviter les accidents peut être obtenue par la détection d'objets aussi.



Figure 1.2 : Exemples des domaines d'applications

### 1.2.3 Les défis de la détection d'objets

La détection d'objet est un problème difficile qui se pose dans un grand nombre d'applications de traitement d'images. Alor pour un détecteur idéal devrait avoir :

- ✓ Haute précision de localisation et de reconnaissance : le détecteur doit être capable de localiser et de reconnaître les objets dans les images avec précision.
- ✓ Haute efficacité en temps et en mémoire : la tâche de détection doit s'exécuter à une fréquence d'images suffisante (fps) avec une utilisation de mémoire et de stockage acceptable.

Pour plus de précision, nous avons plusieurs défis principaux :

Premièrement, les variations intra-classe, où chaque catégorie d'objet peut avoir de nombreuses instances d'objet. Ces instances varient dans plusieurs caractéristiques comme la couleur, la texture, la taille, la forme et différentes poses dans le cas de classes non rigides. Les variations sont causées par des changements dans un ensemble de facteurs tels que :

- Changement d'illumination
- Les conditions météorologiques
- Changement d'échelle
- Occultations partielles ou totales
- Déformation de l'objet
- Objet en mouvement rapide
- La corruption du bruit et une mauvaise résolution
- Mouvement de caméra
- Objet de petite taille
- Les arrière-plans
- L'ombrage

Pour l'efficacité, le deuxième défi est la nécessité de détecter des objets en temps réel. Cela nécessite souvent de grandes performances, ou on peut sacrifier la précision pour gagner la vitesse. D'autre part, nous devons construire un détecteur efficace qui fonctionne dans des appareils qui ont des capacités de calcul et un espace de stockage limités tels que les mobiles. [6]



Figure 1.3 : Exemple des différents défis de détection

### 1.3 Etat de l'Art sur la détection d'objets

Les méthodes de détection d'objets relèvent généralement d'approches basées sur l'apprentissage automatique (Machine Learning) ou d'approches basées sur l'apprentissage en profondeur (Deep Learning). Pour les approches d'apprentissage automatique, il devient nécessaire de définir d'abord les entités en utilisant l'une des méthodes ci-dessous, puis en utilisant une technique telle que la machine à vecteur de support (SVM) pour effectuer la classification. D'un autre côté, les techniques d'apprentissage en profondeur sont capables de détecter des objets de bout en bout sans définir spécifiquement de caractéristiques, et sont généralement basées sur des Réseaux de Neurones Convolutifs (CNN). Les réseaux de neurones convolutifs ont contribué à une augmentation significative de la précision de la détection d'objets et ont largement dépassé d'autres modèles classiques tels que le cadre Viola & Jones [3] et les histogrammes de gradient orienté (HoG). [7]

- Approches d'apprentissage automatique (*Machine Learning*) :
  - Cadre de détection d'objets Viola & Jones basé sur les fonctionnalités de Haar.
  - Transformation d'entité invariante à l'échelle (SIFT).
  - Histogramme de gradient orienté (HOG).
- Approches d'apprentissage en profondeur (*Deep Learning*) :
  - Propositions de régions (R-CNN, R-CNN rapide, R-CNN plus rapide)
  - Détecteur multiBox à un coup (SSD)
  - Vous ne regardez qu'une fois (YOLO)
  - Réseau de neurones de raffinement à un seul coup pour la détection d'objets (FineDet)
  - Retina-Net
  - Réseaux convolutifs déformables

#### 1.3.1 La méthode de la détection d'objets par la méthode «Viola et Jones »

##### 1.3.1.1 Définition

La méthode «**Viola & Jones**» a été proposée en 2001 au départ pour la détection des visages dans une image numérique ou séquence vidéo puis utilisée pour détecter d'autres objets comme les voitures. La bibliothèque **OpenCV** présente une implémentation de cette méthode sous le nom «**détecteur en cascades de Haar** ». Le point fort de cette méthode est la rapidité de détection ce qui la rend capable de s'exécuter en temps réel et de répondre aux exigences du traitement vidéo. Cependant, elle présente quelques limites telles que la difficulté de détection simultanée de plusieurs vues de même objet, la durée nécessaire à la phase d'apprentissage des cascades est relativement assez grande et le nombre d'échantillons d'apprentissage est important.

Cette méthode combine quatre principes clés qui sont les caractéristiques rectangulaires simples appelées des caractéristiques pseudo-Haar en raison de leur similitude avec les ondelettes de

Haar, l'approche d'image intégrale pour la détection rapide et efficace des caractéristiques, la méthode d'apprentissage adaptative AdaBoost [8].

### 1.3.1.2 Les quatre principes clés de la méthode :

1. Les caractéristiques **pseudo-Haar** permettent la détection des objets selon plusieurs échelles (détection multi-échelles).
2. **L'image intégrale** permet le calcul des caractéristiques en temps réel.
3. L'algorithme **AdaBoost** sélectionne les caractéristiques les plus discriminantes pour la classification et forme un classifieur de bonne performance.
4. **Le cascade** minimise le temps de calcul et affine les frontières de classification.

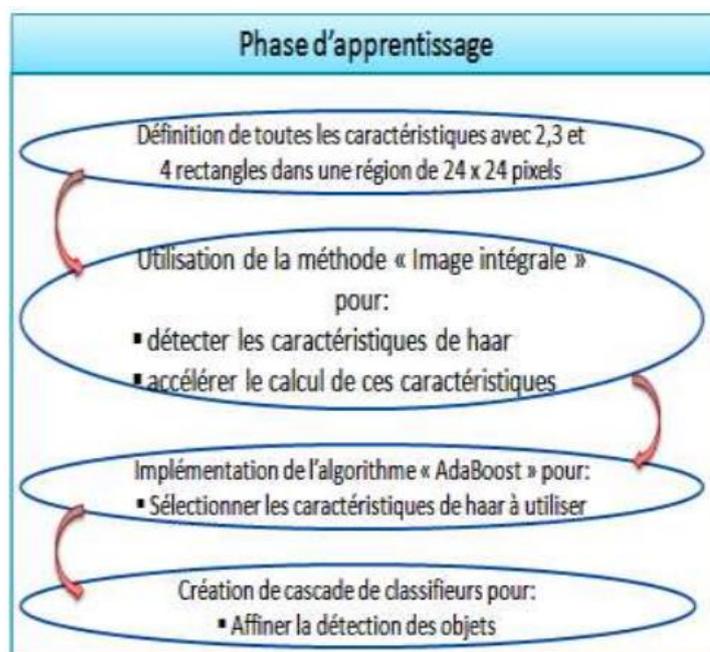


Figure 1.4 : La chaîne d'exécution de la méthode de «Viola&Jones » [9]

### 1.3.1.3 Les Caractéristiques rectangulaires

Viola et Jones proposant d'utiliser des caractéristiques, c'est-à-dire une représentation synthétique et informative, calculé à partir des valeurs des pixels. Viola et Jones définissent des caractéristiques très simple, les caractéristiques de pseudo-Haar, qui sont calculé par la déférence des sommes de Pixels de deux ou plusieurs zones rectangulaires adjacentes. [10]

Les caractéristiques sont calculées à toutes les positions et à toutes les échelles dans une fenêtre de détection de petite taille, typiquement de 24 x 24 pixels ou de 20 x 15 pixels .Un très grand nombre de caractéristiques par fenêtre est ainsi généré, Viola et Jones donnant l'exemple d'une fenêtre de taille 24 x 24 qui génère environ 160 000 caractéristiques. [11].

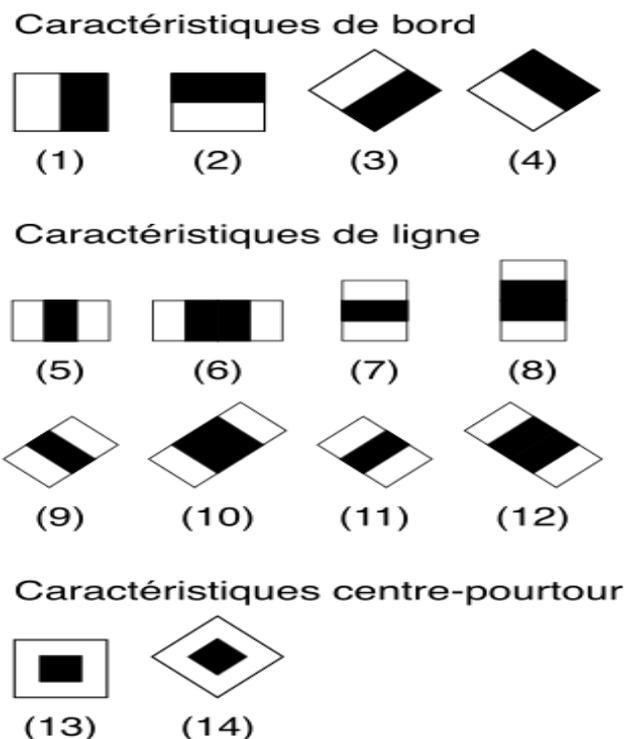


Figure 1.5 : Les descripteurs de Haar (Haar features) [12]

#### 1.3.1.4 L'image intégrale

Pour calculer rapidement et efficacement les caractéristiques pseudo-Haar sur une image, les autres proposent également une nouvelle méthode, qui s'appelle "image intégrale". C'est une représentation sous la forme d'une image, de même taille que l'image d'origine. [10]

#### 1.3.1.4 L'algorithme AdaBoost

Le 3<sup>ème</sup> élément clé de la méthode de Viola et Jones est l'utilisation d'une méthode de boosting afin de sélectionner les meilleures caractéristiques. Le **Adaboost** est un principe qui consiste à construire un classifieur « fort » à partir d'une combinaison pondérée de classifieurs « faible », c'est-à-dire donnant en moyenne une réponse meilleure qu'un tirage aléatoire, Viola et Jones adaptent ce principe en assimilant une caractéristique à un classifieur faible, en construisant un classifieur faible qui n'utilise qu'une seule caractéristique.

#### 1.3.1.5 Le cascade classifieurs

La méthode de Viola et Jones est basée sur une approche par recherche exhaustive sur l'ensemble de l'image, qui teste la présence de l'objet dans une fenêtre à toutes les positions et à plusieurs échelles. Cette approche est cependant extrêmement coûteuse en calcul [13]. La figure (1.6) illustre l'architecture de la cascade, les fenêtres sont traitées séquentiellement par les classifieurs, et rejetées immédiatement si la réponse est négative. [13]

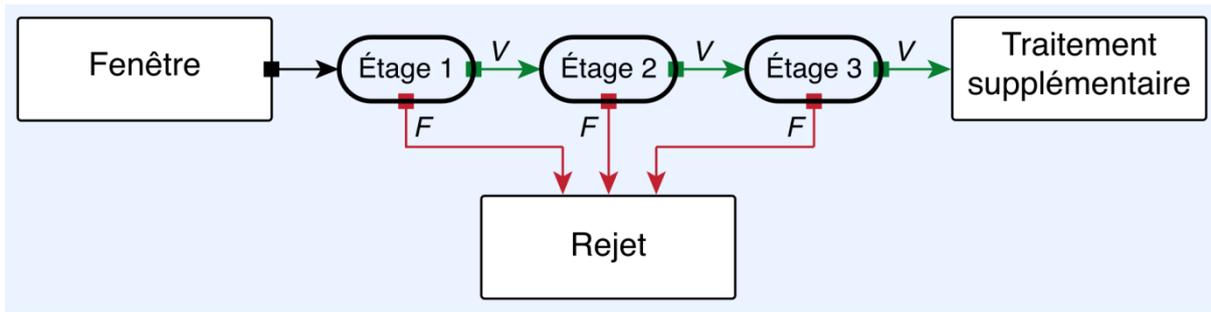
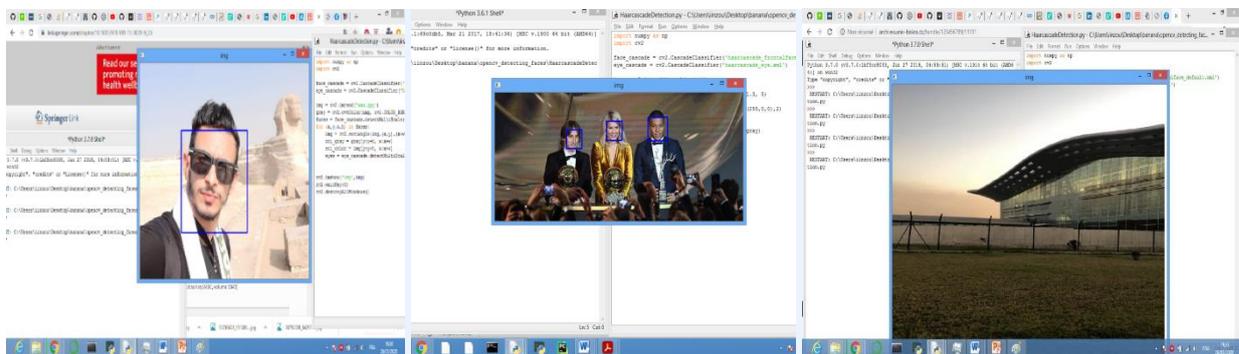


Figure 1.6 : Modèle du cascade classifieurs [13]

La figure (1.7) montre quelques exemples des résultats de la détection d'objets par la méthode Viola et Jones



(a) Détection d'un seul visage (b) Détection du plusieurs visages (c) Aucun objet détecté

Figure 1.7 : Exemple sur la détection d'objets par Viola et Jones.

### 1.4 Etat de l'Art sur la détection d'objets basés sur le Deep learning

L'apprentissage en profondeur, également connu sous le nom de Deep Learning ou de techniques basées sur l'apprentissage structuré profond, a récemment remporté un énorme succès dans le traitement d'images numériques pour la détection et la classification d'objets. En conséquence, ils gagnent rapidement en popularité et en attention auprès de la communauté de recherche en vision par ordinateur. Cette croissance des données d'image a conduit au besoin de détection et de classification automatiques à l'aide de classifieurs basés sur un réseau neuronal profond. Depuis 2012, nous constatons une diminution significative d'erreur comme la montre la figure ci-dessous. Les meilleurs classifieurs d'images se sont trompés de 27% de leurs suppositions en 2010, le Deep learning l'a réduit à 3,5% en 2015

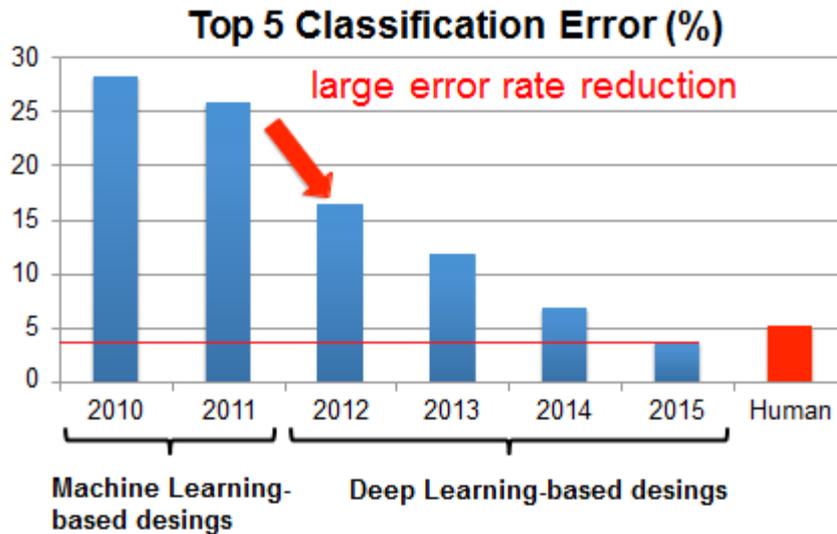


Figure 1.8: Erreur Top-5 sur IMAGENET [14]

Actuellement, les cadres de détection d'objets basés sur l'apprentissage profond peuvent être principalement divisés en deux familles (figure 1.9) : **Détecteurs à deux étages**, tels que le CNN régional (R-CNN) et ses variantes, et **Détecteurs à un étage**, tels que YOLO et ses variantes.

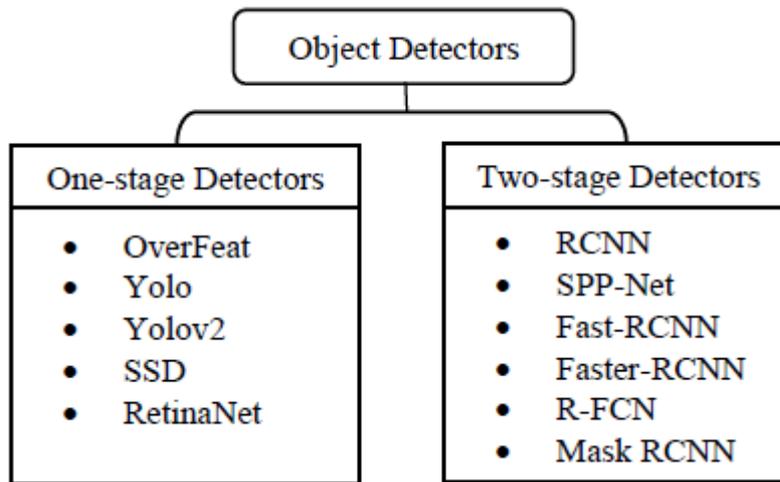


Figure 1.9 : Principales catégories de détecteurs d'objet [15]

### 1.4.1 Détecteurs à deux étages

Les détecteurs à deux étages ont donné une précision plus élevée avec meilleures performances et rapportent des résultats idéal mieux que les détecteurs à un étage dans la détection d'objets, mais ils sont généralement plus lents que les détecteurs à un étage car ils ont deux étapes (figure 1.10): La première étape utilise un réseau de proposition de région pour générer des régions d'intérêt qui ont une forte probabilité d'être soit un arrière-plan, soit un objet. La deuxième étape, un modèle basé sur l'apprentissage en profondeur est utilisé pour effectuer la classification finale et la régression par boîte englobante [15], ce qui les rend plus chronophages que les détecteurs à un étage qui utilisent directement un réseau pour la proposition de région et la classification final. [16]

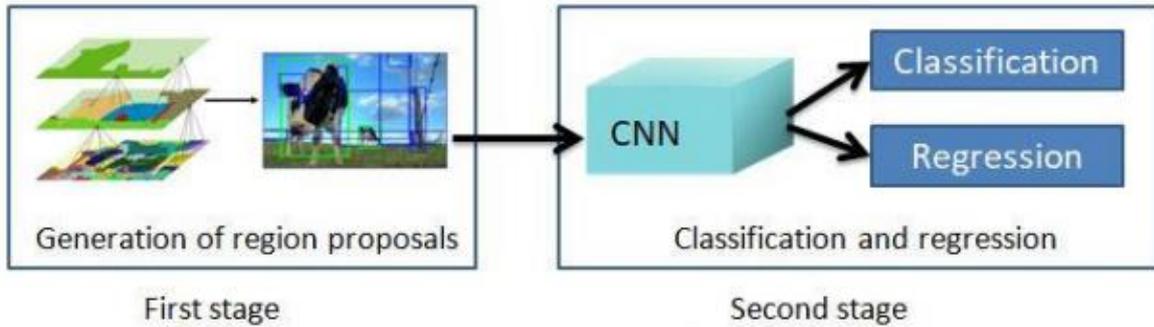


Figure 1.10 : Modèle Détecteurs à deux étages [17]

### 1.4.2 Détecteurs à un étage

Les détecteurs à un étage sont beaucoup plus rapides et plus recherchés pour les applications de détection d'objets en temps réel, mais ont des performances relativement médiocres par rapport aux détecteurs à deux étages [15]. Car un détecteur à un étage n'a pas d'étape séparée pour la génération de proposition (ou l'apprentissage d'une génération de proposition). Dans ce cas, les étapes de pré-détection de régions d'intérêt et de classification sont fusionnées en une seule étape de détection gérée par un seul réseau neuronal. [16]

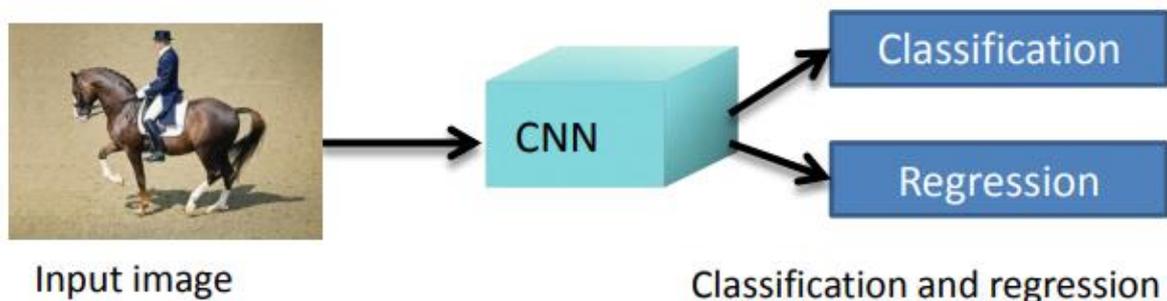


Figure 1.11: Modèle Détecteurs à un étage[17]

### 1.4.3 Convolution neural network (CNN)

Dans cette section, nous discuterons la méthode de détection la plus courante CNN. Convolution Neural Network (CNN ou ConvNet) est l'une des structures réseau les plus représentatives de la technologie d'apprentissage en profondeur, ont été introduits pour la première fois par Fukushima [18] et a connu un grand succès dans le domaine du traitement et de la reconnaissance d'images. CNN c'est un algorithme multicouche il a trois principaux types de couches partagé en deux partie. La première partie de ces couches consiste à extraire les caractéristiques des images et la deuxième partie de CNN est un réseau de neurones sa tâche est la classification des fonctionnalités (voir Figure 1.12 ) [18] [19].

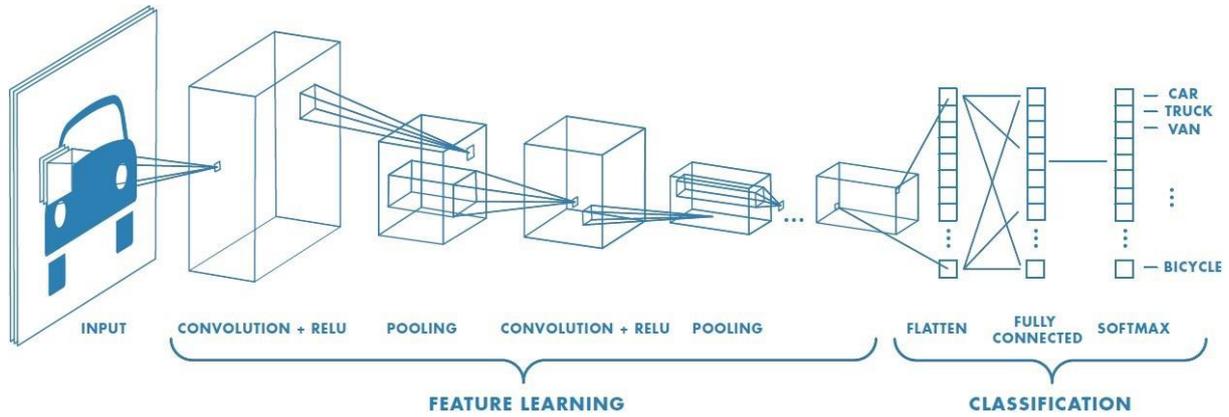


Figure 1.12 : Architecture standard d'un réseau à convolutions. [20]

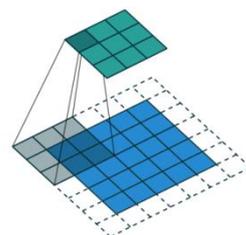
**a. Couche de convolution**

La couche de convolution est le bloc de construction de base d'un CNN. Le but principal de la convolution dans le cas d'un ConvNet est d'extraire des caractéristiques de l'image d'entrée. La convolution préserve la relation spatiale entre les pixels en apprenant les caractéristiques de l'image à l'aide de petits carrés de données d'entrée. Nous n'entrerons pas ici dans les détails mathématiques de la Convolution, mais nous essaierons de comprendre comment cela fonctionne sur les images.

Chaque image peut être considérée comme une matrice de valeurs de pixels. Prenons une image ( 5 x 5) dont les valeurs de pixel ne sont que de 0 et 1 (notez que pour une image en niveaux de gris, les valeurs de pixel vont de 0 à 255). La matrice verte ci-dessous (voir figure 1.13) est un cas particulier où les valeurs de pixel ne sont que de ( 0 et 1), et le carré jaune est appelé un «filtre» ou un «noyau». Les filtres agissent comme des détecteurs de caractéristiques de l'image d'entrée d'origine. [21] Maintenant, lorsque nous glissons la matrice de **filtre** sur la matrice (5 × 5) en partant du coin supérieur gauche, on obtient le résultat indiqué dans la figure ci-dessous.

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

1	0	1
0	1	0
1	0	1



Convolution

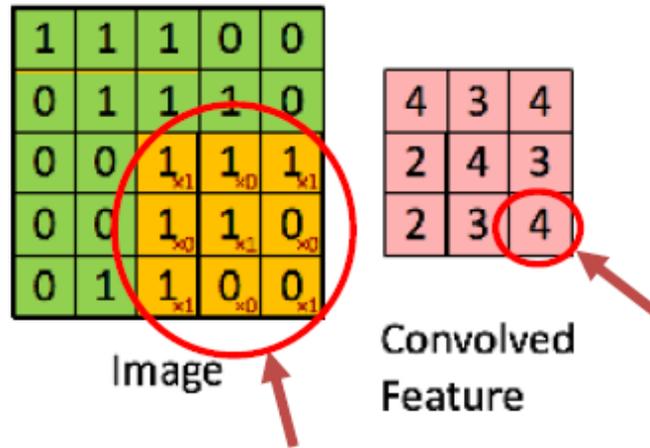


Figure 1.13 : l'étape de convolution. [55]

**b. Couche de Pooling**

La couche de pooling (POOL) est une opération de sous-échantillonnage typiquement appliquée après une couche convolution. Sa fonction est de réduire progressivement la taille de la carte des caractéristiques (matrice convolue) pour réduire la quantité des paramètres et de calculs dans le réseau, tout en conservant les informations importantes.

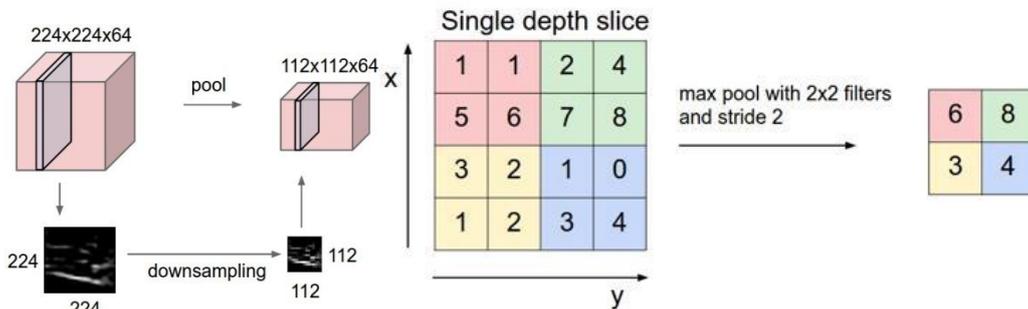


Figure 1.14 : Max pooling modèle (prend la plus grande valeur de chaque fenêtre) [22]

**c. La couche entièrement connectée (Fully-Connected )**

Enfin, après les couches de convolution et pooling, le raisonnement de haut niveau dans le réseau neuronal se fait via des couches totalement connectées. Dans les réseaux de neurones convolutifs, chaque couche agit comme un filtre de détection pour la présence de caractéristiques spécifiques ou des motifs présents dans les données d'origine. Les premières couches d'un réseau convolutif détectent des caractéristiques qui peuvent être reconnues et interprétées facilement. Les couches ultérieures détectent de plus en plus des caractéristiques plus abstraites. La dernière couche du réseau convolutif est capable de faire une classification ultra-spécifique en combinant toutes les caractéristiques spécifiques détectées par les couches précédentes. [23]

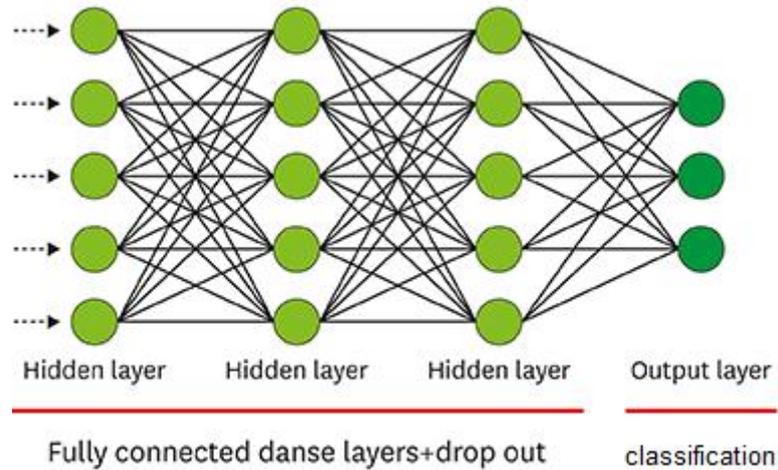


Figure 1.15 : La partie du Couche entièrement connectée (fully connected) [24]

## 1.4 Conclusion

Dans ce chapitre, nous avons présenté le principe de la détection d'objets et ces différents domaines d'applications. Ensuite, nous avons présenté un état de l'art sur la détection d'objets visant à montrer la diversité des approches proposées dans ce domaine. Ces approches sont divisées en deux catégories : approches basées sur l'apprentissage automatique (Machine Learning) et approches basées sur l'apprentissage en profondeur (Deep Learning). Enfin, nous avons discuté le principe et les différentes étapes des méthodes de détection les plus courantes dans les deux catégories : Viola & Jones et CNN.

## *Chapitre 2 : La détection d'objet par Deep Learning*

## 2.1. Introduction

Aujourd'hui, le Deep Learning, ce programme d'apprentissage approfondi comme nous disons en français, est utilisé par des dizaines d'applications et de technologies que nous utilisons au quotidien ( Google Map capable de repérer sur une image le numéro des rues, Facebook qui s'en sert pour la reconnaissance faciale sur les photos d'amis par exemple, Siri pour comprendre la voix et adapter ses réponses aux demandes de l'utilisateur, Skype ou Google Trad, qui traduisent des conversations orales en temps réel, et il y en a bien d'autres. [25] Si on entend beaucoup parler de Deep Learning aujourd'hui, ce n'est pas pour rien. Mais que se cache-t-il derrière ces deux mots ? À partir de ce chapitre, nous allons apprendre sur le concept de "Deep Learning" en commençant par la "machine Learning" qui est classé comme une branche majeure de l'intelligence artificielle, et nous pouvons le définir comme science ce qui permet à un ordinateur de se comporter et effectuer toutes ces tâches et d'autres efficacement sans être programmé explicitement. Ensuite, nous allons mettre en relief quelques notions de base liées à la détection d'objet par Deep Learning. Enfin en passeras par les 3 robustes méthodes de détection d'objet qui ont été proposées durant ces dernières années Faster-RCNN, YOLO et

## 2.2. Qu'est-ce que le machine Learning ?

Bien que l'apprentissage automatique ne soit pas nouveau, sa définition précise est encore confuse pour de nombreuses personnes. Le concept de Machine Learning (**ML**) remonte du milieu du 20ème siècle. Dans les années 1950, le mathématicien britannique **Alan Turing** imagine une machine capable d'apprendre. Au cours des décennies suivantes, différentes techniques de **ML** ont été développées pour créer des algorithmes capables d'apprendre et de s'améliorer de manière autonome. L'apprentissage automatique est une technologie d'intelligence artificielle qui permet aux ordinateurs d'apprendre sans être explicitement programmés pour le faire. Pour apprendre et grandir, les ordinateurs ont besoin de données à analyser et à former. En effet, les données volumineuses (Big Data) est le cœur du Machine Learning, la technologie qui libère tout le potentiel des données volumineuses (Big data). [26]



### 2.3.1. Types d'apprentissage

Dans le domaine du Machine Learning (apprentissage automatique en français), il existe deux principaux types d'apprentissages : supervisés et non supervisés. L'apprentissage supervisé et non supervisé sont tous deux une partie importante de l'apprentissage automatique. Nous expliquons la principale différence entre les deux types comme suit :

#### 2.3.1.1. L'apprentissage supervisé

L'apprentissage est dit supervisé lorsque les données qui entrent dans le processus sont déjà catégorisées (étiquetées) et que les algorithmes doivent s'en servir pour prédire un résultat en vue de pouvoir le faire plus tard lorsque les données ne seront plus catégorisées. Cette technique d'apprentissage est appelée « Supervised Learning » ou apprentissage supervisé. [28]

Afin de mieux comprendre ce concept, prenons un exemple : un utilisateur reçoit chaque jour un grand nombre d'e-mails, certains sont des e-mails d'entreprises importants et d'autres sont des e-mails indésirables non sollicités ou des spam. Un algorithme supervisé sera présenté avec un grand nombre d'e-mails qui ont déjà été étiquetés par l'utilisateur comme spam ou non spam. L'algorithme fonctionnera sur toutes les données étiquetées, faire des prédictions sur l'e-mail et voir si c'est un spam ou non. Cela signifie que l'algorithme examinera chaque exemple et fera une prédiction pour chacun pour savoir si l'e-mail est un spam ou pas. La première fois, l'algorithme fonctionne sur toutes les données non étiquetées, la plupart des e-mails seront mal étiquetés car il peut fonctionner assez mal au début. Cependant, après chaque exécution, l'algorithme compare sa prédiction au résultat souhaité (l'étiquette). Au fur et à mesure, l'algorithme apprendra à améliorer ses performances et sa précision.

Dans l'exemple que nous avons utilisé, nous avons décrit un processus dans lequel un algorithme apprend à partir de données étiquetées (emails qui ont été catégorisés comme spam ou non-spam). [29]

#### 2.3.1.2. L'apprentissage non-supervisé

La deuxième classe d'algorithmes d'apprentissage automatique est appelée apprentissage non supervisé, dans ce cas, nous n'étiquetons pas les données au préalable, nous laissons plutôt l'algorithme arriver à sa conclusion. Ce type d'apprentissage est important car il est

beaucoup plus commun dans le cerveau humain que l'apprentissage supervisé. Les algorithmes d'apprentissage non supervisé sont particulièrement utilisés dans les problèmes de clustering, dans lesquels, étant donné une collection d'objets, nous voulons être en mesure de comprendre et de montrer leurs relations. Une approche standard consiste à définir une mesure de similarité entre deux objets, puis à rechercher tout groupe d'objets plus similaires les uns aux autres, par rapport aux objets des autres clusters. Par exemple, dans le cas précédent des e-mails spam/ non spam, l'algorithme peut être capable de trouver des éléments communs à tous les spam (par exemple, la présence de mots mal orthographiés). Bien que cela puisse fournir une classification meilleure qu'aléatoire, il n'est pas clair que les spam/non spam puissent être facilement séparés. [29]

### 2.4. Le Deep Learning

#### 2.4.1. Concept

Le Deep Learning (L'apprentissage en profondeur) est un type de l'apprentissage automatique, en utilisant les réseaux de neurones pour saisir modèles complexes. Ce type de technologie permet aux systèmes d'Intelligence Artificielle d'exécuter des tâches humaines, «telles que la reconnaissance visuelle d'objet de la vie réelle ou comprendre la parole. » Alors que le fonctionnement de ces réseaux inspirés du cerveau reste impénétrable, leurs algorithmes de couches interconnectées donnent aux machines la possibilité d'être entraînées et d'effectuer des tâches spécifiques. Les algorithmes qui pilotent ce processus représentent un plus grand nombre des variables plus abstraites. [30]

Historiquement, le Deep Learning remonte à 1943, lorsque Walter Pitts et Warren McCulloch ont créé un modèle informatique basé sur les réseaux neuronaux du cerveau humain. Ils ont utilisé une combinaison d'algorithmes et de mathématiques qu'ils ont appelé «logique de seuil» pour imiter le processus de pensée. Depuis lors, le Deep Learning a évolué régulièrement, avec seulement deux pauses significatives dans son développement. En 2011, la vitesse des GPU avait considérablement augmenté, ce qui permettait d'entraîner des réseaux de neurones convolutifs «sans» la pré-formation couche par couche. Avec l'augmentation de la vitesse de calcul, il est devenu évident que le Deep Learning présentait des avantages significatifs en termes d'efficacité et de vitesse. Un exemple est AlexNet, un réseau de neurones convolutifs dont l'architecture a remporté plusieurs concours internationaux en 2011 et 2012.

Toujours en 2012 avec la grande reprise, Google Brain (est un projet de recherche de Deep Learning conduit par Google) a publié les résultats d'un projet inhabituel appelé

L'expérience du chat (voir figure 2.3). Le projet à l'esprit libre a exploré les difficultés de «l'apprentissage non supervisé». Le Deep Learning utilise «l'apprentissage supervisé», ce qui signifie que le réseau neuronal convolutif est formé à l'aide de données étiquetées (pensez aux images d'ImageNet). Actuellement, le traitement du Big Data et l'évolution de l'intelligence artificielle dépendent tous deux du Deep Learning. Le Deep Learning est toujours en évolution et a besoin d'idées créatives. [31]

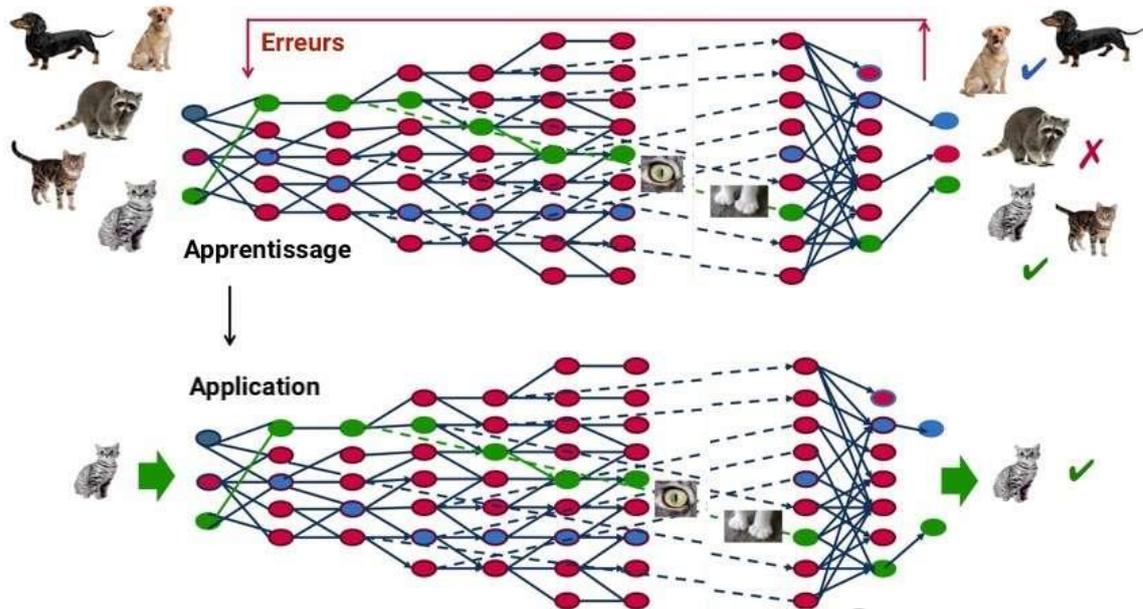


Figure 2.3 : Le projet de L'expérience du chat par (Deep Learning) [32]

#### 2.4.2. Avantage du Deep Learning

L'un des principaux avantages du Deep Learning est l'analyse et l'apprentissage des quantités massives de données, Ce qui lui a permis de pouvoir dans tous les domaines. Plus que ça on observe une amélioration significative des performances **MAP** (mesurées de précision moyenne) depuis l'arrivée du Deep Learning en 2012. **(a)** résultats de la compétition de détection d'objet gagnantes aux concours VOC2007-2012, et **(b)** résultats de la compétition de détection d'objet les plus élevé dans ILSVRC2013-2017 (les résultats dans les deux panels utilisent uniquement les données d'entraînement fournies) [16]

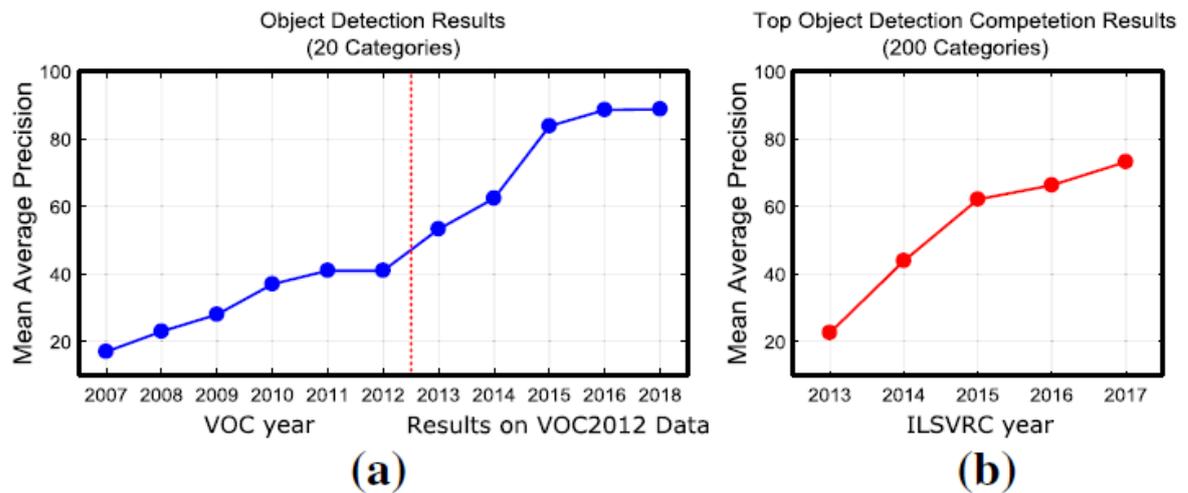


Figure 2.4 : Un aperçu des performances récentes de détection d'objets [16]

### 2.4.3. Les différentes Architectures du Deep Learning

Dans les réseaux neurones de Deep Learning les informations sont transmises à travers chaque couche, la sortie de la couche précédente fournissant une entrée pour la couche suivante. La première couche d'un réseau est appelée couche d'entrée, tandis que la dernière est appelée couche de sortie. Toutes les couches entre les deux sont appelées couches cachées (**Hidden layers**). Plus le nombre de neurones est élevé, plus le réseau est « profond ». Chaque couche est généralement un algorithme simple et uniforme contenant un type de fonction d'activation. [31] Comme le montre la figure ci-dessous.

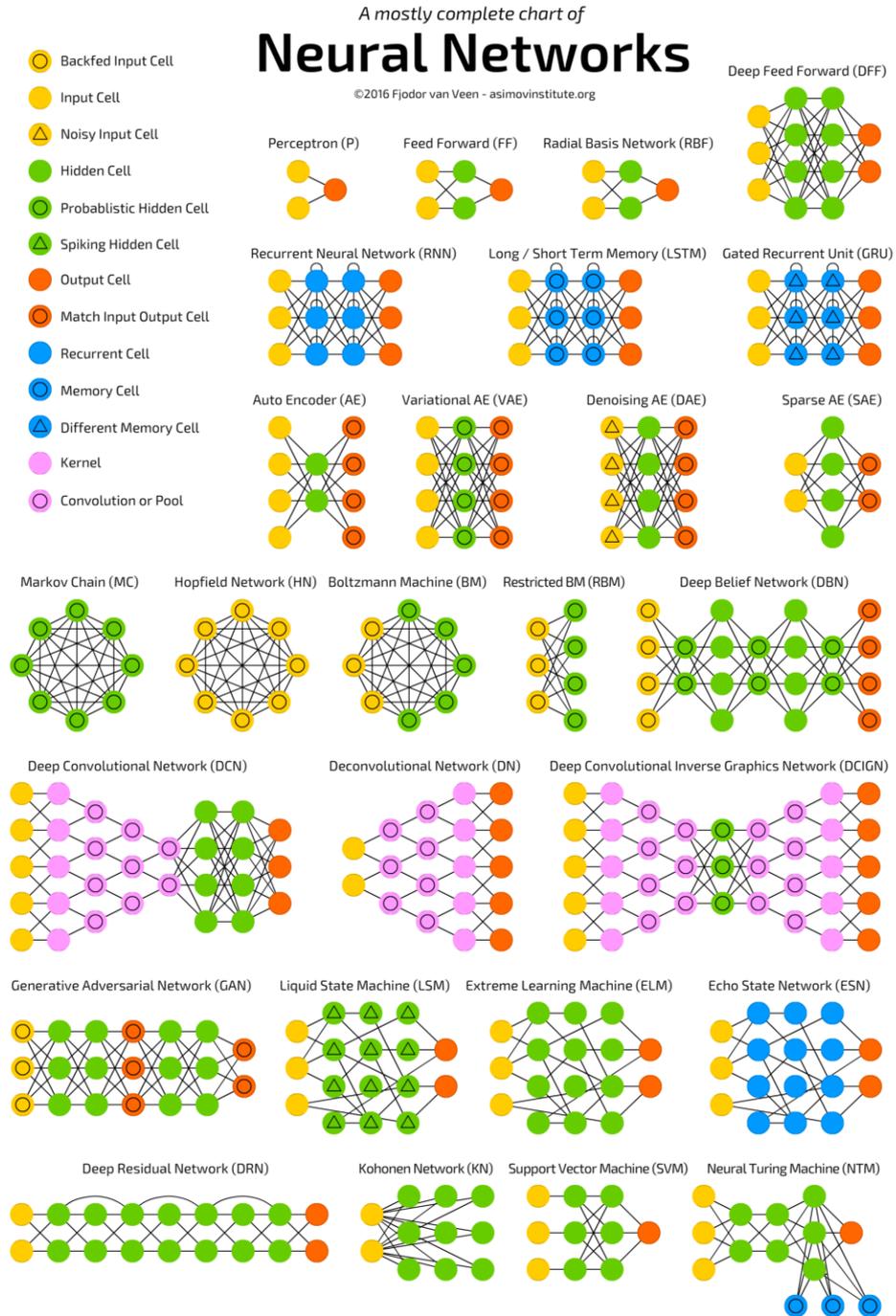


Figure 2.5 : Modèles des réseaux neuronaux de Deep Learning [34]

### 2.4.4. Applications du Deep Learning

Le Deep Learning est utilisé dans de nombreux domaines comme :

- Reconnaissance d'image,
- Traduction automatique,
- Voiture autonome,
- Diagnostic médical,
- Recommandations personnalisées,
- Modération automatique des réseaux sociaux,
- Prédiction financière et commerce automatisé,
- Identification de pièces défectueuses,
- Détection de malwares ou de fraudes,
- Chatbots (agents conversationnels),
- Exploration spatiale,
- Robots intelligents.

C'est aussi grâce au Deep Learning que l'intelligence artificielle de Google Alpha Go a réussi à battre les meilleurs champions de Go en 2016. Le moteur de recherche du géant américain est lui-même de plus en plus basé sur l'apprentissage par Deep Learning plutôt que sur des règles écrites.[32] Aussi, il est utilisé pour la reconnaissance faciale de Facebook pour identifier automatiquement vos amis sur les photos. C'est également cette technologie qui permet aussi à la reconnaissance faciale Face ID de l'iPhone X d'Apple de s'améliorer au fil du temps. [35] Aujourd'hui, le Deep Learning est même capable de « créer » tout seul des tableaux de Van Gogh ou de Rembrandt, d'inventer aussi un langage totalement nouveau pour communiquer entre deux machines. [32]

- Speech Recognition
  - Speech → Words
- Visual Object Recognition
  - ImageNet (car, dog)
- Object Detection
  - Face detection
  - pedestrian detection
- Drug Discovery
  - Predict drug activity
- Genomics
  - Deep Genomics company



Figure 2.6 : Les différents domaines d'application de Deep Learning [35]

### 2.5. La détection d'objets basée sur Deep Learning

Ces dernières années, la détection d'objets basée sur l'apprentissage en profondeur est bonne en termes de robustesse par rapport à la détection traditionnelle. Cependant, les modèles profonds sont difficiles à former. Les travaux en cours fournissent une perspective pour la recherche sur la détection d'objets, ce qui ouvre la concurrence devant les méthodes de détection d'objets.[1] Parmi eux, il y a : R-CNN , SPP-net , Fast R-CNN , OverFeat , MR-CNN&S-CNN , Faster R-CNN , HyperNet , RetinaNet , MSGR , StuffNet , SSD300 , SSD512 , OHEM , SDP+CRC , GCNN , SubCNN , GBD-Net , PVANET , YOLO900 , YOLOv2 , R-FCN , FPN , Mask R-CNN , DSSD and DSOD. [37] Certaines méthodes sont plus fiables et robustes que d'autres. Dans cette section, nous discuterons les méthodes de détection les plus courantes, Faster R-CNN, YOLO (You Only Look Once) et SSD (Single Shot MultiBox Detector). [21]

La figure suivante montre l'évolution des méthodes de détection d'objets au fil de temps.

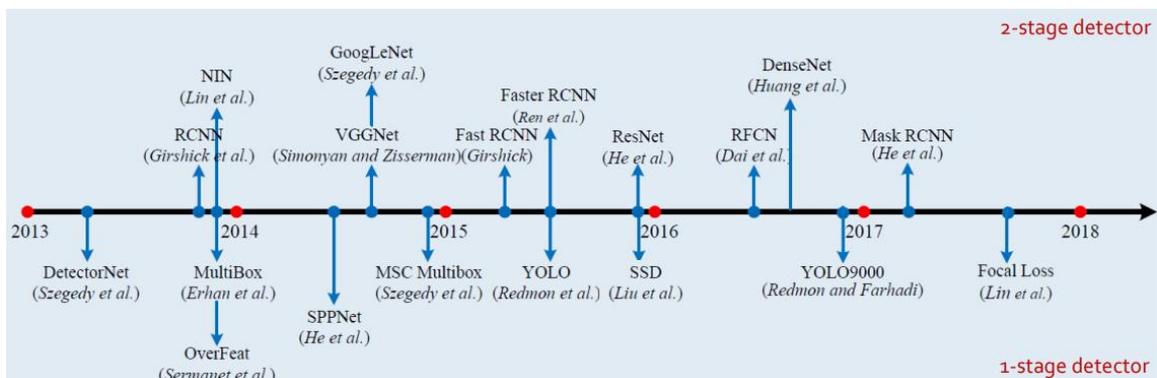
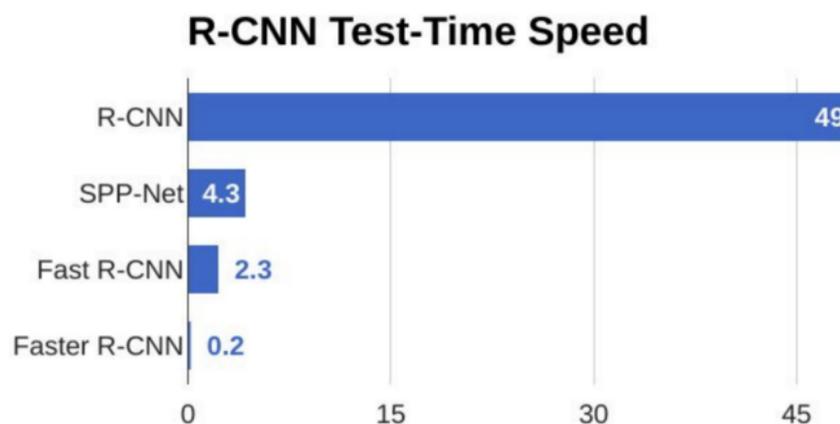


Figure 2.7 : chronologique des méthodes de détection d'objets. [38]

### 2.5.1. Faster R-CNNs

La principale chose qui ralentit R-CNN est le mécanisme de recherche sélective qui propose de nombreuses régions possibles et nécessite de les classer toutes. De plus, le processus de sélection de la région n'est pas «profond» et il n'y a pas d'apprentissage impliqué, ce qui limite sa précision. En 2015, Girshik et al. ont proposé un algorithme amélioré appelé Fast R-CNN [21], mais il s'appuyait toujours sur la recherche sélective, limitant ses performances. Shqing Ren et al. ont proposé un algorithme amélioré appelé Faster R-CNN, qui supprime complètement la recherche sélective et permet au réseau d'apprendre directement les propositions de région.

Faster R-CNN prend l'image source et la transmet à un CNN appelé un réseau de prédiction de région (RPN) (voir figure 2.10). Il prend en compte un grand nombre de régions possibles, encore plus que dans l'algorithme R-CNN d'origine, et utilise une méthode d'apprentissage en profondeur efficace pour prédire quelles régions sont les plus susceptibles d'être des objets d'intérêt. Les propositions de régions prédites sont ensuite remodelées à l'aide d'une couche de regroupement de régions d'intérêt (RoI). Cette couche elle-même est utilisée pour classer les images dans chaque région et prédire les valeurs de décalage pour les cadres de délimitation [39]. La figure ci-dessous montre les énormes gains de performances obtenus par Faster R-CNN par rapport aux R-CNN et Fast R-CNN originaux proposés par l'équipe de Girshik. À partir du graphique, vous pouvez voir que Faster R-CNN est beaucoup plus rapide que ses prédécesseurs. Par conséquent, il peut même être utilisé pour la détection d'objets en temps réel.



**Figure 2.8 : Comparaison de la vitesse de test des algorithmes de détection d'objets**

[40]

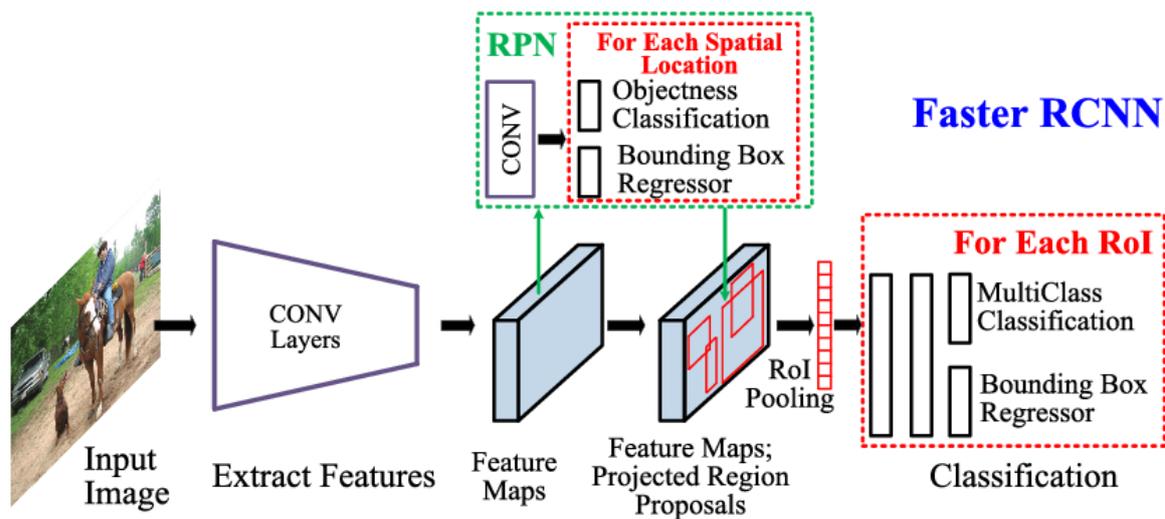


Figure 2.9 : Illustration du structure de détection Faster- RCNN [33]

### 2.5.1.1. Réseau de proposition de région (RPN)

L'algorithme alimente les régions possibles, générées par les ancres définies à l'étape précédente, dans le RPN, un CNN spécial utilisé pour prédire les régions avec des objets d'intérêt. Le RPN prédit la possibilité qu'une ancre soit en arrière-plan ou au premier plan et affine l'ancre ou la boîte englobante.

Les données d'entraînement du RPN sont les ancres et un ensemble de boîtes de vérité. Les ancres qui ont un chevauchement plus élevé avec les boîtes de vérité doivent être étiquetées comme premier plan, tandis que les autres doivent être étiquetées comme arrière-plan. Le RPN convertit l'image en entités et considère chaque caractéristique à l'aide des 9 ancres, avec deux étiquettes possibles pour chacune (arrière-plan ou premier plan).

Enfin, la sortie est introduite dans une fonction d'activation de régression Softmax ou logistique, pour prédire les étiquettes de chaque ancre. Un processus similaire est utilisé pour affiner les ancrages et définir les cadres de délimitation des entités sélectionnées. Les ancres trouvées au premier plan sont transmises à l'étape suivante de l'algorithme R-CNN. [41]

La figure ci-dessous illustre l'architecture du réseau de proposition de région RPN.

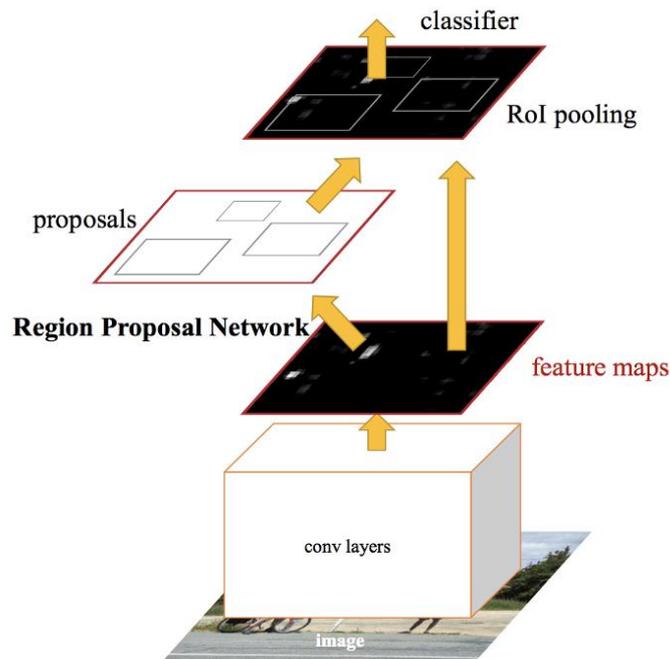


Figure 2.10 : L'architecture du réseau de proposition de région « RPN » [42]

### 2.5.2. YOLO (You Only Look Once)

YOLO a été publié en 2015 par Joseph Redmon. La 2<sup>ème</sup> version de YOLO, appelée YOLO9000, également connue sous le nom de YOLOv2 et la troisième version de YOLO, appelée YOLOv3, sortie en mai 2018. L'algorithme «YOLO» est un meilleur algorithme qui s'attaque au problème de la prédiction des boîtes englobantes précises tout en utilisant la technique de la fenêtre glissante à convolution. La structure de la méthode YOLO est illustrée dans la figure (2.12). Mais comme le montre la figure (2.13), le fonctionnement de YOLO est que nous prenons une image et la divisons en une grille SxS, dans chacune des grilles, nous prenons N boîtes englobantes. Pour chacune des boîtes englobantes, le réseau génère une probabilité de classe et des valeurs de décalage pour la boîte englobante. Les cadres de délimitation ayant la probabilité de classe supérieure à une valeur de seuil sont sélectionnés et utilisés pour localiser l'objet dans l'image.

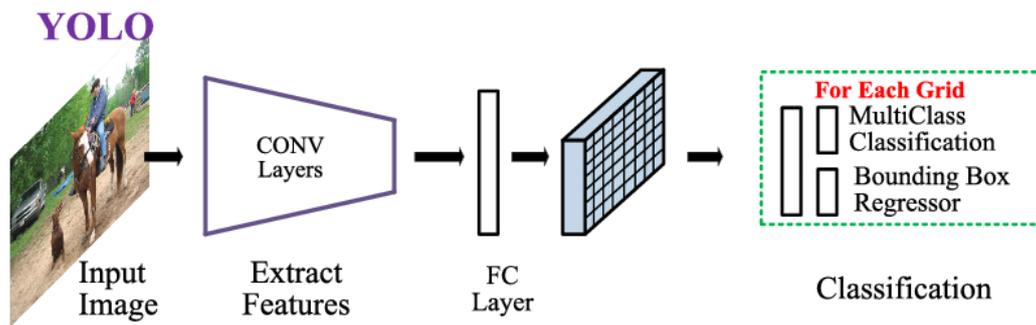


Figure 1.11 : illustration du structure de la méthode YOLO (Redmon and Farhadi, 2015) [33]

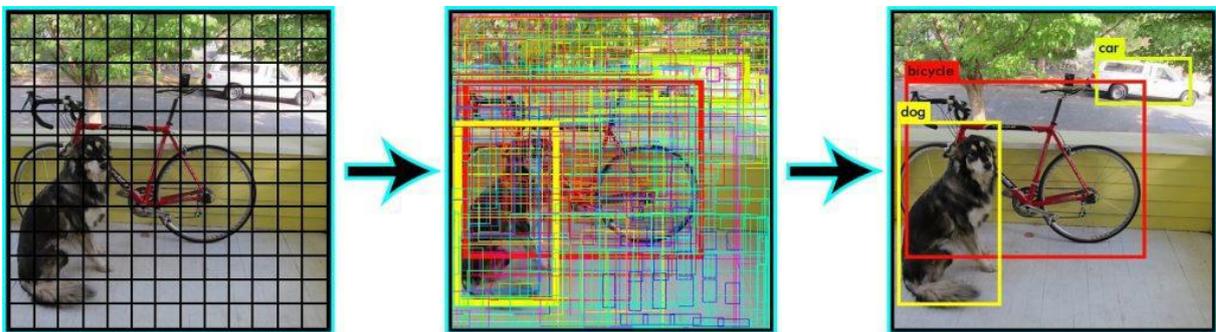


Figure 2.12 : Comment YOLO gère les boîtes englobantes. [43]

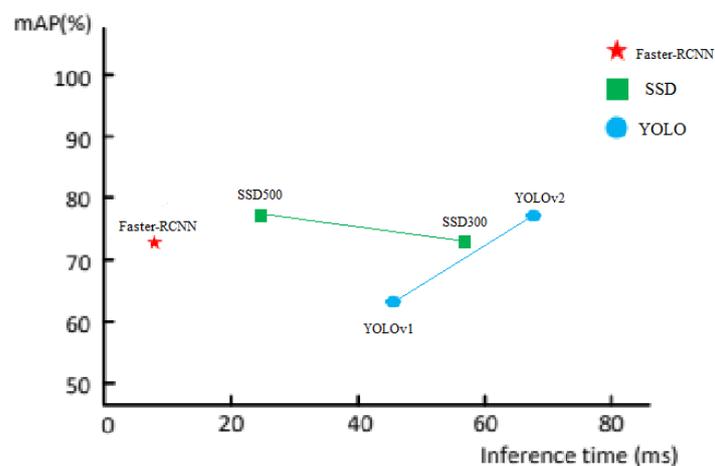
YOLO est des ordres de grandeur plus rapide (fonctionne à 45 FPS sur Titan X) que les autres algorithmes de détection d'objets. La limitation de l'algorithme YOLO est qu'il se débat avec de petits objets dans l'image, par exemple il pourrait avoir des difficultés à détecter un troupeau d'oiseaux. Cela est dû aux contraintes de l'algorithme. L'avantage de l'algorithme YOLO est qu'il est très rapide car au moment de l'exécution, l'image n'a été exécutée qu'une seule fois sur CNN, ce qui rend YOLO beaucoup plus rapide que Faster R-CNN et peut être s'exécuter en temps réel. [40]

### 2.5.3. SSD (Single Shot MultiBox Detector)

Single Shot MultiBox Detector (SSD) a été publié en 2015 par Liu et al. Single shot indique que l'algorithme SSD appartient à la méthode en une étape. MultiBox indique que le SSD est une prédiction multi-objets. SSD développé à l'origine par Google, est une combinaison de vitesse et de précision. SSD exécute un CNN sur l'image d'entrée une seule fois et génère une carte des caractéristiques. Ensuite, il exécute un petit noyau convolutif de taille  $3 \times 3$  sur la carte des caractéristiques pour prédire les boîtes englobantes et la probabilité de classification. Le SSD a également des boîtes d'ancrage similaires à Faster

R-CNN, mais SSD apprend le décalage et la probabilité de classe plutôt que la boîte. Et pour gérer l'échelle d'objet de différentes tailles, le SSD prédit des boîtes de délimitation après plusieurs couches convolutives, car chaque couche convolutive agit à une échelle différente. [21]

La figure ci-dessous montre la précision avec le temps d'inférence pour les méthodes Faster R-CNN et SSD, YOLO. On observe sur cette figure **SSD300** atteint **74,3% mAP** à **59 FPS** tandis que **SSD500** atteint **76,9% mAP** à **22 FPS**, ce qui surpasse le Faster R-CNN (73,2% mAP à 7 FPS) et **YOLOv1** (63,4% mAP à 45 FPS) et **YOLOv2** (76,8% mAP à 67 FPS) dans sa précision avec un Fps très satisfaisant.



**Figure 2.13 : Précision (mAP) avec le temps d'inférence [44]**

SSD utilise la technique de MultiBox. Il prend l'image en entier et la fait passer à travers de multiples couches de convolution de différente taille. Il se compose de 2 parties :

1. Extraction des cartes des caractéristiques.
2. Application des filtres de convolution sur chaque case pour détecter les objets.

### 2.5.3.1. Structure réseau du SSD

Le SSD utilise VGG16 comme modèle de base, puis ajoute une couche de convolution sur la base de VGG16. La structure réseau du SSD est illustrée à la figure 2.15. On peut clairement voir que le SSD utilise des cartes de caractéristiques multi-échelles pour la détection. La taille d'image d'entrée du modèle est 300×300 (Peut également être 512×512, Ce qui n'est pas différent de l'ancienne structure du réseau, sauf qu'une couche de convolution est ajoutée à la fin, ce qui n'est pas traité dans ce paragraphe).

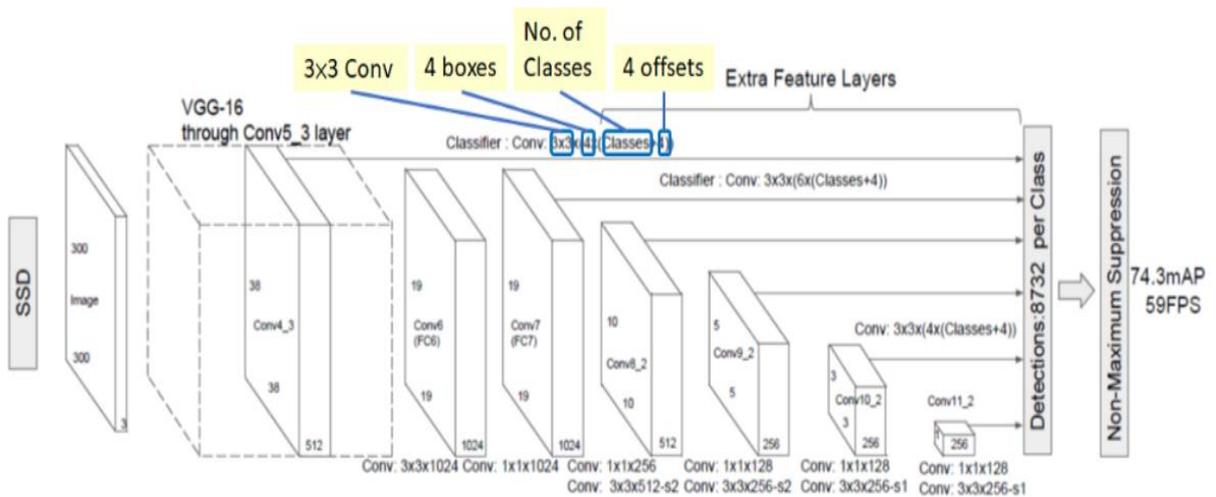


Figure 2.14 : Architecture de réseau SSD [42]

SSD utilise le modèle de base VGG16 pour extraire les cartes de caractéristiques (voir figure 2.16). Ensuite, il détecte les objets à l'aide du calque Conv4\_3. À titre d'illustration, nous dessinons le Conv4\_3 à  $8 \times 8$  spatialement (il devrait être  $38 \times 38$ ). Pour chaque cellule (également appelée emplacement), il effectue 4 prédictions d'objets, comme l'illustrer la figure (2.17).

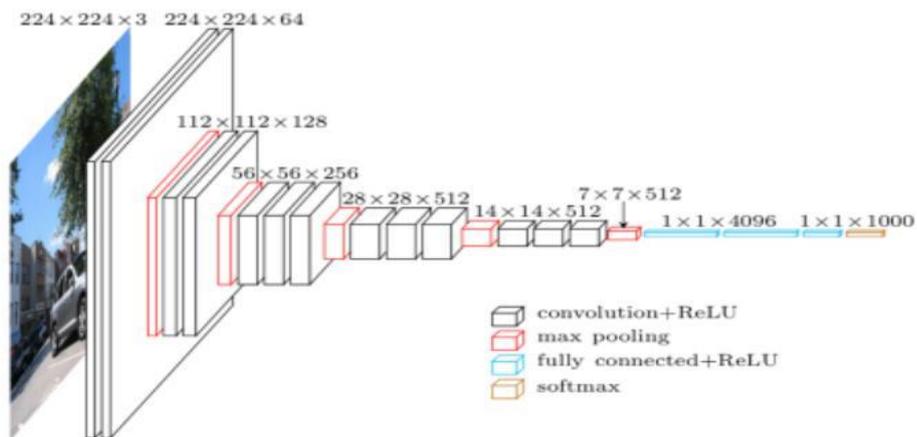
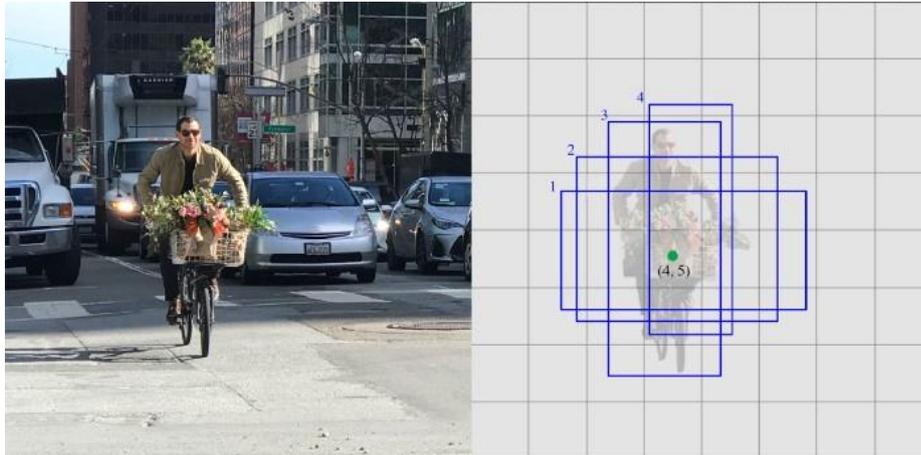


Figure 2.15 : Modèle VGG16

Chaque prédiction se compose d'une zone de délimitation et de 21 scores pour chaque classe (une classe supplémentaire pour aucun objet), et nous choisissons le score le plus élevé comme classe pour l'objet délimité. Conv4\_3 fait un total de  $38 \times 38 \times 4$  prédictions : quatre prédictions par cellule quelle que soit la profondeur des cartes d'entités. Comme prévu, de nombreuses prédictions ne contiennent aucun objet. SSD réserve une classe «0» pour indiquer qu'il n'a aucun objet.

Le SSD ajoute 6 couches de convolution auxiliaires supplémentaires après le VGG16. Cinq d'entre eux seront ajoutés pour la détection d'objets. Dans trois de ces couches, nous faisons 6 prédictions au lieu de 4. Au total, SSD fait 8732 prédictions en utilisant 6 couches, comme le montre la figure 2.15. [45]



**Figure 2.16 : Les 4 prédictions d'objets.**

➤ Pour plus de détail sur la structure de SSD en a :

Conv4\_3, il est de taille  $38 \times 38 \times 512$ . Et il y a 4 boîtes englobantes et chaque boîte englobante aura (classes + 4) sorties. Alor, à Conv4\_3, la sortie est de  $38 \times 38 \times 4 \times (c + 4)$ . Supposons qu'il y ait 20 classes d'objets plus une classe d'arrière-plan, la sortie est :  $38 \times 38 \times 4 \times (21 + 4) = 144\ 400$ . En termes de nombre de boîtes englobantes, il y a  $38 \times 38 \times 4 = 5776$  boîtes englobantes.

De même pour les autres couches de conv :

Conv7:  $19 \times 19 \times 6 = 2166$  boîtes (6 boîtes pour chaque emplacement)

Conv8\_2:  $10 \times 10 \times 6 = 600$  boîtes (6 boîtes pour chaque emplacement)

Conv9\_2:  $5 \times 5 \times 6 = 150$  boîtes (6 boîtes pour chaque emplacement)

Conv10\_2:  $3 \times 3 \times 4 = 36$  boîtes (4 boîtes pour chaque emplacement)

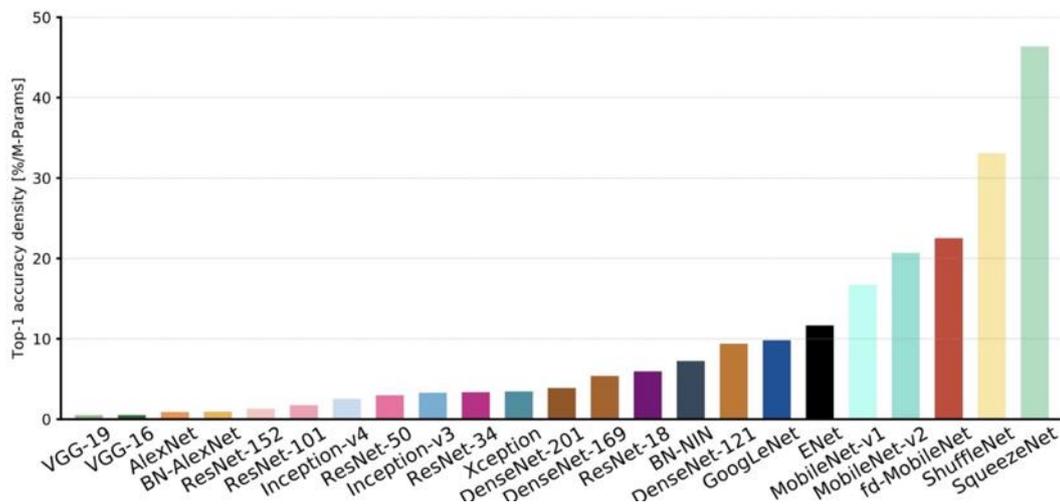
Conv11\_2:  $1 \times 1 \times 4 = 4$  boîtes (4 boîtes pour chaque emplacement)

Si nous les résumons, nous obtenons  $5776 + 2166 + 600 + 150 + 36 + 4 = 8732$  boîtes au total. Pour le YOLO, il y a  $7 \times 7$  emplacements à la fin avec 2 boîtes de délimitation pour

chaque emplacement. YOLO n'a obtenu que  $7 \times 7 \times 2 = 98$  boîtes. Par conséquent, le SSD a 8732 boîtes de délimitation, ce qui est plus que celui de YOLO. Ce qui le rend beaucoup plus rapide que les approches basées sur RPN. [46]

**2.5.3.2. Modèle**

L'approche SSD est basée sur un réseau convolutif à action directe qui produit une collection de taille fixe de boîtes englobantes et de scores pour la présence d'instances de classe d'objets dans ces boîtes, suivie d'une étape de suppression non maximale pour produire les détections finales. Les premières couches de réseau sont basées sur une architecture standard utilisée pour la classification d'images de haute qualité (tronquée avant toute couche de classification (Réseaux d'extraction de caractéristiques, tels que : VGG, googlenet, alexnet)), qui s'appelle le réseau de base.



**Figure 2.17: Présentation des performances des différents modèles d'architecture CNN dans la tâche de classification [47]**

Ensuite, il ajoute une structure auxiliaire au réseau, résultant en des détections avec les principales caractéristiques suivantes [48] :

- **Cartes de caractéristiques multi-échelles pour la détection :**

Des couches d'entités convolutives sont ajoutées à la fin du réseau de base tronqué. Ces couches diminuent progressivement de taille et permettent des prédictions de détections à plusieurs échelles. Le modèle convolutif de prédiction des détections est différent pour chaque couche d'entités. [48]



Figure 2.18 : Les perspectives multi-échelles des objets détectés

- **Prédicteurs convolutifs pour la détection :**

Chaque couche d'entités ajoutée (ou éventuellement une couche d'entités existante du réseau de base) peut produire un ensemble fixe de prédictions de détection à l'aide d'un ensemble de filtres convolutifs. Celles-ci sont indiquées au-dessus de l'architecture du réseau SSD sur la Figure 2.15. Pour une couche de caractéristiques de taille  $(m \times n)$  avec  $p$  canaux, l'élément de base pour prédire les paramètres d'une détection de potentiel est un petit noyau  $(3 \times 3 \times p)$  qui produit soit un score pour une catégorie ou un décalage de forme par rapport aux coordonnées de la boîte par défaut. À chacun des  $(m \times n)$  emplacements où le noyau est appliqué, il produit une valeur de sortie. Les valeurs de sortie du décalage de la boîte englobante sont mesurées par rapport à une position de boîte par défaut par rapport à chaque emplacement de la carte de caractéristiques. [48]

- **Boîtes par défaut et ratios d'aspect :**

Un ensemble de boîtes de délimitation par défaut sont associées à chaque cellule de carte d'entités, pour plusieurs cartes d'entités en haut du réseau. Les boîtes par défaut placent la carte des caractéristiques de manière convolutionnelle, de sorte que la position de chaque boîte par rapport à sa cellule correspondante soit fixe. À chaque cellule de la carte d'entités, nous prédisons les décalages par rapport aux formes de boîte par défaut dans la cellule, ainsi que les scores par classe qui indiquent la présence d'une instance de classe dans chacune de ces boîtes. Plus précisément, pour chaque boîte sur  $k$  à un emplacement donné, nous calculons les scores de classe  $c$  et les 4 décalages par rapport à la forme de boîte par défaut d'origine. Il en résulte un total de  $(c + 4) k$  filtres qui sont appliqués autour de

chaque emplacement dans la carte d'entités, donnant des sorties de  $(c + 4) k \times m \times n$  pour une  $m \times n$  carte d'entités. Pour une illustration des boîtes par défaut (voir figure 2.18). Nos boîtes par défaut sont similaires aux boîtes d'ancrage utilisées dans Faster RCNN [2], cependant nous les appliquons à plusieurs cartes de caractéristiques de différentes résolutions. Autoriser différentes formes de boîte par défaut dans plusieurs cartes d'entités nous permet de discrétiser efficacement l'espace des formes de boîte de sortie possibles. [48]

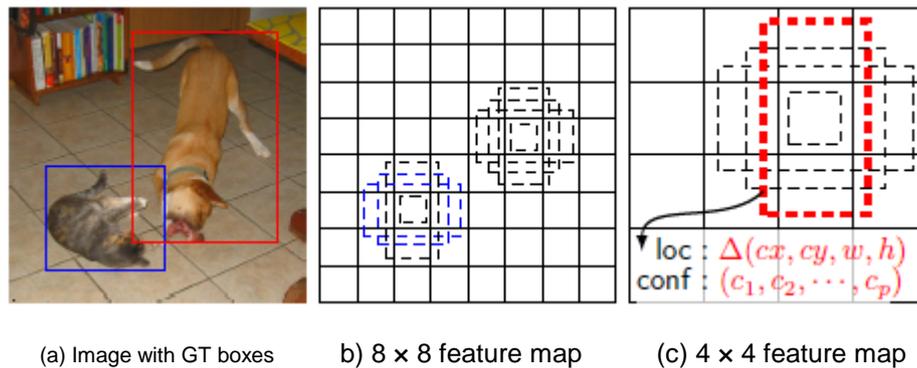


Figure 2.19 : Boîtes par défaut de SSD aux cartes de caractéristiques 8x8 et 4x4. [48]

### 2.5.3.3. Entraînement

La principale différence entre l'apprentissage du SSD et l'apprentissage d'un détecteur typique qui utilise des propositions de région, est que les informations de vérité doivent être affectées à des sorties spécifiques dans l'ensemble fixe de sorties de détecteur. Une version de ceci est également requise pour la formation en YOLO et pour l'étape de proposition de région de Faster R-CNN et MultiBox. Une fois cette affectation déterminée, la fonction de perte et la rétro-propagation sont appliquées de bout en bout. La formation implique également de choisir l'ensemble de boîtes et d'échelles par défaut pour la détection, ainsi que les stratégies d'extraction et d'augmentation des données négatives. [48]

- **Stratégie de correspondance :**

Pendant la formation, nous devons déterminer quelles boîtes par défaut correspondent à une détection de vérité et former le réseau en conséquence. Pour chaque boîte de vérité, nous sélectionnons parmi les boîtes par défaut qui varient selon l'emplacement, le rapport

hauteur / largeur et l'échelle. Nous commençons par faire correspondre chaque boîte de vérité à la boîte par défaut avec le meilleur chevauchement jaccard (comme dans MultiBox. Contrairement à MultiBox, nous associons ensuite les boîtes par défaut à toute vérité avec un chevauchement jaccard supérieur à un seuil (0,5). Cela simplifie le problème d'apprentissage, ce qui permet au réseau de prédire des scores élevés pour plusieurs boîtes par défaut qui se grounds plutôt que de lui demander de choisir uniquement celle avec un chevauchement maximal. [48]

Cette stratégie de correspondance encourage chaque prédiction à prédire des formes plus proches de la boîte par défaut correspondant. Par conséquent, nos prédictions sont plus diversifiées et plus stables dans la formation.

- **Objectif de l'Entraînement :**

L'objectif de formation SSD est dérivé de l'objectif Multibox mais étendu pour gérer plusieurs catégories d'objets. Ils utilisent la perte de localisation, qui mesure la différence entre les boîtes de délimitation prévues par le réseau et celles de la vérité, entre la boîte prédite et les paramètres de la boîte de vérité, et une perte softmax pour la perte de confiance, qui mesure la dépendance du réseau à avoir un objet à l'intérieur d'une boîte englobante, de sorte que la fonction de perte objectif globale soit une somme pondérée de la perte de localisation et de la perte de confiance . [48]

- **Choix des échelles et ratios d'aspect pour les Boîtes par défaut :**

Pour gérer différentes échelles d'objets, certaines méthodes suggèrent de traiter l'image à différentes tailles et de combiner les résultats par la suite. SSD utilise à la fois les cartes de caractéristiques inférieures et supérieures pour la détection. La figure 2.18 montre deux exemples de cartes de caractéristiques ( $8 \times 8$  et  $4 \times 4$ ) qui sont utilisées dans SSD. Dans le SSD, les boîtes par défaut n'ont pas nécessairement besoin de correspondre aux champs récepteurs réels de chaque couche. Par exemple, sur la figure 2.18, le chien correspond à une boîte par défaut dans la carte des caractéristiques  $4 \times 4$ , mais pas à des boîtes par défaut dans l'entité  $8 \times 8$  carte. En effet, ces boîtes ont des échelles différentes et ne correspondent pas à la boîte pour chien, et sont donc considérées comme des négatifs

pendant la formation. L'échelle des boîtes par défaut pour chaque carte d'entités est calculée comme suit : [48]

$$s_k = s_{\min} + \frac{s_{\max} - s_{\min}}{m - 1}(k - 1), \quad k \in [1, m]$$

Où  $m$  fait référence au nombre de cartes d'entités, et  $s_{\min}$  est 0,2 et  $s_{\max}$  est 0,9, ce qui signifie que la couche la plus basse a une échelle de 0,2 et la couche la plus élevée a une échelle de 0,9, et toutes les couches intermédiaires sont régulièrement espacées.

#### 2.5.3.4 Extraction négative dure (Hard negative mining)

Après l'étape de mise en correspondance, la plupart des cases par défaut sont négatives, en particulier lorsque le nombre de cases par défaut possibles est important. Cela introduit un déséquilibre significatif entre les exemples de formation positifs et négatifs. Au lieu d'utiliser tous les exemples négatifs, nous les trions en utilisant la perte de confiance la plus élevée pour chaque boîte par défaut et choisissons les meilleurs afin que le rapport entre les négatifs et les positifs soit au plus de 3: 1. Nous avons constaté que cela conduit à une optimisation plus rapide et une formation plus stable. [48]

#### 2.5.3.5 Augmentation des données

L'utilisation de l'augmentation des données (Data Augmentation) peut améliorer les performances du SSD. Pour rendre le modèle plus robuste à différentes tailles et formes d'objets d'entrée, chaque image d'apprentissage est échantillonnée de manière aléatoire par l'une des options suivantes :

- Utilisez toute l'image d'entrée d'origine.
- Échantillonnez un patch de sorte que le chevauchement minimum de jaccard avec les objets soit de 0,1 - 0,3 - 0,5 - 0,7 ou 0,9.
- Échantillonnez un patch au hasard.

La taille de chaque patch échantillonné est  $[0,1, 1]$  de la taille de l'image d'origine et le rapport hauteur / largeur est compris entre 2 et 2. Nous conservons la partie superposée de la boîte de vérité terrain si le centre de celle-ci est dans le patch échantillonné. Après l'étape d'échantillonnage susmentionnée, chaque patch échantillonné est redimensionné à

une taille fixe et retourné horizontalement avec une probabilité de 0,5 en plus d'appliquer des distorsions photométriques similaires à celles. [48]

data augmentation	SSD300		
horizontal flip	✓	✓	✓
random crop & color distortion		✓	✓
random expansion			✓
VOC2007 test mAP	65.5	74.3	77.2

Tableau 2.1 : l'amélioration des performances après l'augmentation des données

2.6. Conclusion

Dans cet article, nous avons présenté un aperçu des méthodes de détection d'objets basées sur l'apprentissage en profondeur. Nous avons commencé par une brève histoire de Deep Learning et passé en revue les méthodes de détection d'objets les plus connus et les plus utilisées avec ses architectures. Après nous avons basés sur la méthode SSD. Après nous avons basés sur la méthode SSD qui utilise un réseau CNN pour la détection, mais utilise des cartes de caractéristiques à plusieurs échelles. Le Single Shot Detector prédit les "bounding boxes" ainsi que la classe en une seule étape contrairement à deux comme le fait R-CNN. Il est bien meilleur que Yolo en termes de précision et de vitesse.

# **Chapitre 3 : Implémentation et résultats**

### 3.1. Introduction

Dans ce chapitre, nous verrons comment appliquer la détection d'objets à l'aide Deep Learning. Plus précisément, nous allons utiliser l'algorithme MobileNet SSD avec le module dnn d'OpenCV pour construire notre détecteur d'objets. A la fin, nous examinerons les résultats de l'application du détecteur MobileNet SSD à des exemples d'images d'entrées.

### 3.2. Qu'est-ce qu'un MobileNet-SSD ?

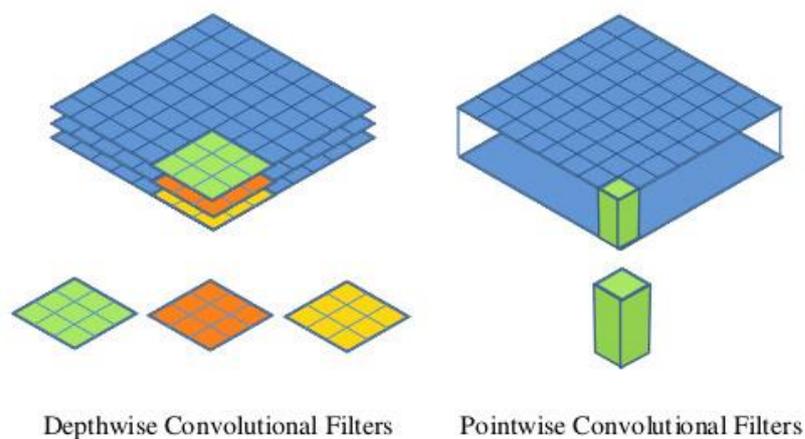
MobileNet est un réseau de neurones convolutifs, développé par **Google**, permettant de classifier 21 types d'objets et présentant des avantages tels que légèreté, rapidité et précision. Ce réseau de neurones est associé au réseau de neurones Single Shot Multibox Detector (SSD) permettant d'identifier toutes les zones dans l'image présentant un élément à classifier.

La combinaison de MobileNet et de SSD permet d'obtenir une méthode rapide et efficace de détection d'objets basée sur le Deep Learning.

Le modèle que nous utilisons est une implémentation sous Caffe entraînée.

La question se pose de savoir pourquoi nous utilisons MobileNet, pourquoi nous ne pouvons pas utiliser ResNet, VGG ou AlexNet ?

La réponse est simple. ResNet ou VGG ou AlexNet a une grande taille de réseau et augmente le taux de calcul, alors que dans MobileNet il existe une architecture simple consistant en une convolution en profondeur  $3 \times 3$  suivie d'une convolution point par point  $1 \times 1$ , comme illustré dans la figure 3.1.



**Figure 3.1 : La convolution du MobileNet.**

### 3.3. Langage de programmation

#### 3.3.1. Python

Le langage de programmation Python a été créé en 1989 par Guido van Rossum, aux Pays-Bas. Le nom Python vient d'un hommage à la série télévisée Monty Python's Flying Circus. Python est un langage de script de haut niveau, structuré en open source, la première version publique de ce langage a été publiée en 1991. La dernière version de Python est la version 3.8.5 qui a été introduit le 20 juillet 2020. La version 2 de Python est désormais obsolète et cessera d'être maintenue après le 1er janvier 2020. Dans la mesure du possible évitez de l'utiliser. Python est le langage de programmation le plus utilisé dans le domaine du Machine Learning, du Big Data et de la Data Science.

##### 3.3.1.1. Pourquoi choisir python ?

Ce langage de programmation présente des nombreuses caractéristiques intéressantes tel que:

- Il est multiplateforme (portable). C'est-à-dire qu'il fonctionne sur de nombreux systèmes d'exploitation : Windows, Mac OS X, Linux, Android, iOS, depuis les mini-ordinateurs Raspberry Pi jusqu'aux supercalculateurs.
- Il est gratuit. Vous pouvez l'installer sur autant d'ordinateurs que vous voulez (même sur votre téléphone !).
- C'est un langage de haut niveau. Il demande relativement peu de connaissance sur le fonctionnement d'un ordinateur pour être utilisé.
- C'est un langage interprété. Un script Python n'a pas besoin d'être compilé pour être exécuté, contrairement à des langages comme le C ou le C++. Il convient bien à des scripts d'une dizaine de lignes qu'à des projets complexes de plusieurs dizaines de milliers de lignes. Il est relativement simple à prendre en main.
- Toutes ces caractéristiques font que Python est désormais enseigné dans de nombreuses formations, depuis l'enseignement secondaire jusqu'à l'enseignement supérieur.



Figure 3.2 : Le logo du python

### 3.3.1.2. Numpy

NumPy est une extension du langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux. Plus précisément, cette bibliothèque logicielle libre et open source fournit de multiples fonctions permettant notamment de créer directement un tableau depuis un fichier ou au contraire de sauvegarder un tableau dans un fichier, et manipuler des vecteurs, matrices et polynômes. On peut aussi l'intégrer avec le code C / C ++ et Fortran. [49]

### 3.4. Bibliothèque OpenCV

OpenCV (Open Source Computer Vision) est une bibliothèque libre de vision par ordinateur. Cette bibliothèque est écrite en C et C++ et peut être utilisée sous Linux, Windows et Mac OS X. Des interfaces ont été développées pour Python, Ruby, Matlab et autre langage. Open CV est orienté vers des applications en temps réel. Un des buts d'OpenCV est d'aider les gens à construire rapidement des applications sophistiquées de vision à l'aide d'infrastructure simple de vision par ordinateur. La bibliothèque d'OpenCV contient plus de 500 fonctions [50]. Et plus de 2500 algorithmes optimisés, qui ont une excellente précision en termes de performances et de vitesse dans la vision par ordinateur et d'apprentissage automatique classiques et de pointe. Ces algorithmes peuvent être utilisés pour détecter et reconnaître des visages, identifier des objets, classer des actions humaines dans des vidéos, suivre les mouvements de caméra, suivre des objets en mouvement, extraire des modèles 3D d'objets, La bibliothèque est largement constituée de sociétés professionnelles, de groupes de recherche et d'autres groupes. De nombreuses entreprises bien établies telles que Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda et Toyota qui emploient la bibliothèque. [51]

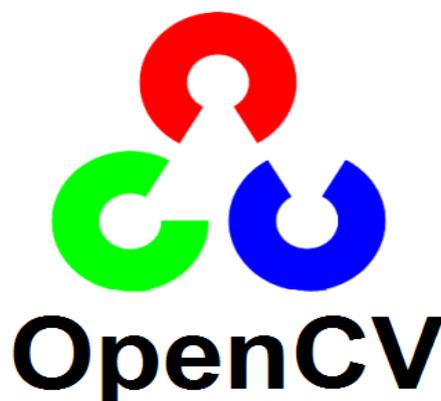


Figure 3.3 : Le logo d'OpenCV.

### 3.5. Base de données (BDD)

La méthode MobileNet SSD a d'abord été formée sur la BDD de COCO, puis a été affinée sur PASCAL VOC atteignant 72,7% mAP (précision Average moyenne). Par exemple - Dans l'ensemble de données Pascal VOC 2007, SSD300 a 79,6% de mAP et SSD512 a 81,6% de mAP, ce qui est plus rapide que le R-CNN de 78,8% de mAP. Jetons un coup d'œil sur les BDD les plus utilisées et les plus connus dans le domaine de la détection.

Dataset	Total Images	Categories	Image Size	Started Year
MNIST[8]	60,000	10	28x28	1998
ImageNet[9]	>14 Millions	21841	500x400	2009
Caltech-101[10]	9,145	101	300x200	2004
Caltech-256 [11]	30,607	256	300x200	2007
MS COCO[12]	>328,000	91	640x480	2014
PASCAL VOC(2012)[13]	11,540	20	470x380	2005
CIFAR-10[14]	60,000	10	32x32	2009
Scene-15[15]	4,485	15	256x256	2006
Tiny images [16]	>79 Millions	53,464	32x32	2006
SUN[17]	131,072	908	500x300	2010
Open Images [18]	>9 Millions	>6000	varied	2017

Tableau 3.1: La BDD de Détection d'objets [15]

### 3.6. Environnement de travail

Dans cette section, nous présenterons les environnements matériel et logiciel de notre travail.

#### 3.6.1. Environnement matériel ou le hardware

L'ensemble de matériel pour réaliser ce travail, PC HP i3 séries 630 dont les caractéristiques:

- Procésseur: Intel® Core (TM) i3 M370.
- Ram: 4 go.
- Carte graphique : il n'y a pas.
- Disque dur : 500 go HDD.
- Système d'exploitation : Windows 8 professionnel.
- Type du système : système d'exploitation 64 bits.

### 3.6.2. Environnement immatériel ou le software

#### 3.6.2.1. Préparation de l'environnement

Pour pouvoir aller au bout de nos objectifs, on va avoir besoin de télécharger un certain nombre de dépendances (Python 3) (que vous devriez avoir déjà installé), ainsi que le logiciel Pycharm et certain librairies.



**Figure 3 .4 : Installation de Python 3.6.1**

#### 3.6.2.2. Installation de Pycharm

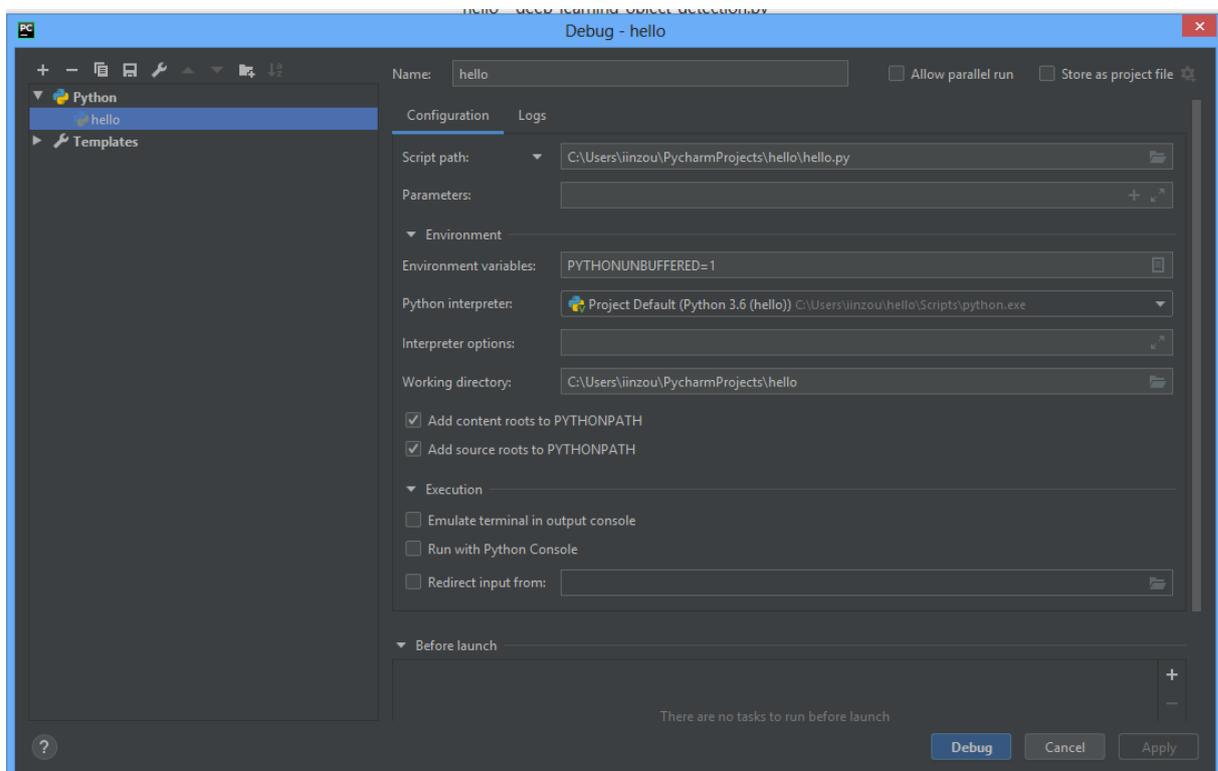
PyCharm est un environnement de développement intégré utilisé pour programmer en Python. Il permet l'analyse de code et contient un débogueur graphique. Il permet également la gestion des tests unitaires, l'intégration de logiciel de gestion de versions, et supporte le développement web avec Django. Développé par l'entreprise tchèque JetBrains, c'est un logiciel multi-plateforme qui fonctionne sous Windows, Mac OS X et Linux. Il est décliné en édition professionnelle, diffusé sous licence propriétaire, et en édition communautaire diffusé sous licence Apache. [52]



**Figure 3.5 : Environnement Pycharm**

### 3.6.2.3 Création de l'environnement sous pycharm :

Après l'installation du python en vas créer notre environnement sur pycharm.



**Figure 3.6 : Création de l'environnement sous pycharm**

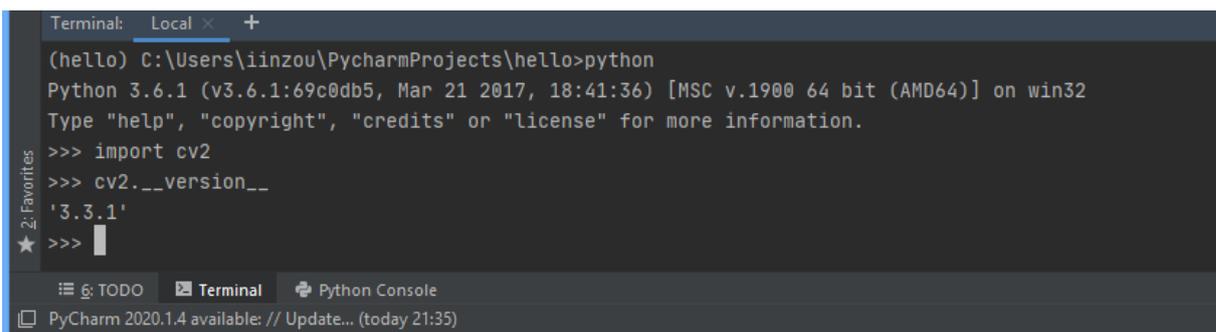
### 3.6.3 Installation des librairies:

On va en utiliser plusieurs, qui seront téléchargées par le biais de pip (installé en même temps avec Python). Saisissez les commandes suivantes dans la console Windows (Cmd) pour tout récupérer :

```
E:\python\opencv>pip install numpy-1.13.1+mk1-cp36-cp36m-win_amd64.whl
Processing e:\python\opencv\numpy-1.13.1+mk1-cp36-cp36m-win_amd64.whl
Installing collected packages: numpy
Successfully installed numpy-1.13.1+mk1

E:\python\opencv>pip install opencv_python-3.3.1-cp36-cp36m-win_amd64.whl
Processing e:\python\opencv\opencv_python-3.3.1-cp36-cp36m-win_amd64.whl
Installing collected packages: opencv-python
Successfully installed opencv-python-3.3.1
```

Figure 3.7 : Installation d'OpenCV par la commande pip

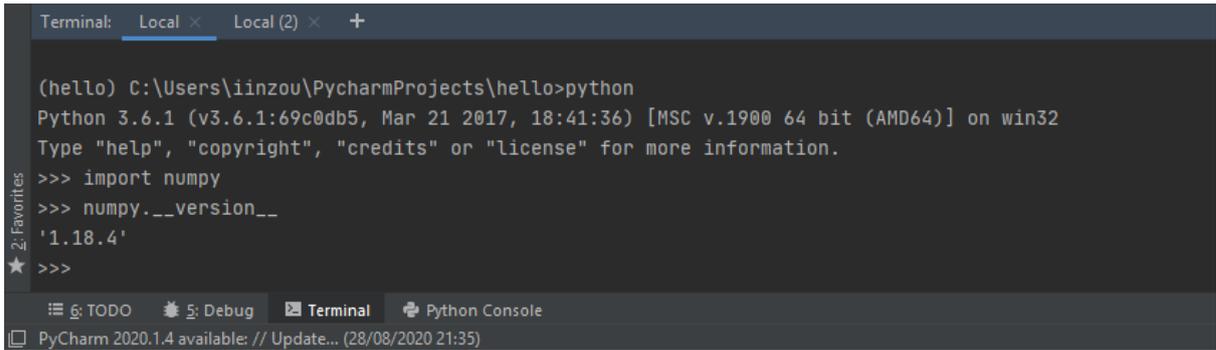


```
Terminal: Local x +
(hello) C:\Users\iinzou\PycharmProjects\hello>python
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 18:41:36) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'3.3.1'
>>>
```

Figure 3.8 : Test de fonctionnement d'OpenCV

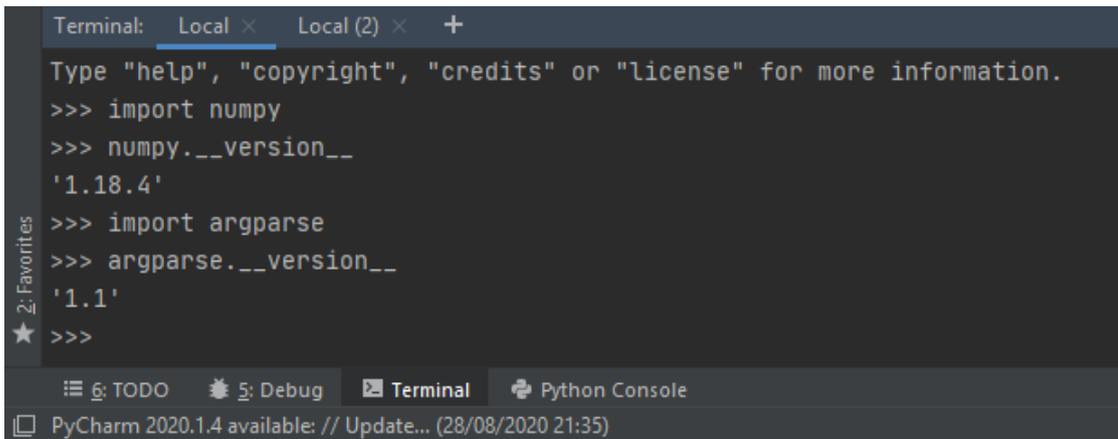
```
C:\Users\Administrator>pip install numpy
Collecting numpy
  Downloading numpy-1.18.4-cp36-none-win_amd64.whl (7.7MB)
    100% |#####| 7.7MB 95kB/s
Installing collected packages: numpy
Successfully installed numpy-1.18.4
You are using pip version 8.1.1, however version 9.0.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
C:\Users\Administrator>
```

Figure 3.9 : Installation de Numpy par la commande pip



```
Terminal: Local x Local (2) x +
(hello) C:\Users\linzou\PycharmProjects\hello>python
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 18:41:36) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> numpy.__version__
'1.18.4'
>>>
```

Figure 3.10 : Test de fonctionnement de Numpy



```
Terminal: Local x Local (2) x +
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> numpy.__version__
'1.18.4'
>>> import argparse
>>> argparse.__version__
'1.1'
>>>
```

Figure 3.11 : Test de fonctionnement de argparse

## 3.7. Détection d'objets par SSD

### 3.7.1. OpenCV DNN Module

C'est une librairie qui permet l'utilisation (inférence) de modèles de Deep Learning pré-entraînés. Il ne permet pas l'entraînement, mais uniquement la prédiction. Il supporte les framework de Deep Learning suivants, sont listées seulement ceux qui font la détection d'objets (avec les modèles de détection d'objets pré-entraînés supportés par OpenCV) [53]:

- **Darknet** : YOLOv2, YOLOv3, YOLOv3-tiny
- **Caffe** : SSD-VGG, SSD-MobileNet, Faster-RCNN
- **Tensorflow** : SSD-Inception, SSD-MobileNet, Faster-RCNN

Nous avons utilisé le module DNN de OpenCV pour faire la détection d'objet par soucis de simplicité de mise en place. Les modèles étant déjà existants, il suffit de télécharger leurs poids et leur topologie afin de les utiliser directement dans OpenCV.

Dans ce travail, nous avons utilisé Caffe (SSD), car il contient tout ce dont nous avons besoin.

### 3.7.2 Le caffe Model

Le modèle que nous utilisons est une implémentation sous Caffe. Caffe est un cadre d'apprentissage en profondeur (framework) qui nous permet de former et de tester les CNN en plus, il nous permet d'utiliser python ou matlab pour accéder à l'architecture du réseau, aux poids, aux biais et aux résultats. Il est développé par Berkeley et AI Research (BAIR) et par des contributeurs de la communauté, il est utilisé autour du monde pour la recherche académique et les projets commerciaux. Les modèles sont définis par configuration sans codage en dur et la formation peut basculer entre le CPU (Central Processing Unit) et le GPU (Graphical processing Unit) en définissant un seul indicateur. [54]

Le module DNN d'OpenCV fonctionne un peu mieux avec les modèles Caffe en ce moment. Mais, dans les prochaines versions d'OpenCV, le chargement du modèle TensorFlow deviendra plus robuste, mais pour le moment, OpenCV prend un peu mieux en charge le chargement des modèles Caffe.

### 3.7.3. Implémentation et résultats expérimentaux

Le code Python de détection et reconnaissance d'objets, se base sur un modèle d'entraînement déjà défini. Pour ce cas, MobileNet SSD a été entraîné pour reconnaître une liste de 21 objets tels qu'une bouteille, un chien, un chat, une personne... ect. Bien sûr, la liste d'objet n'est pas exhaustive. Vous pouvez créer votre propre modèle, pour reconnaître un type d'objet précis.

Nous commençons par créer un répertoire, nommée sur “ deep\_learning\_object\_detection ” afin de stocker les fichiers nécessaires sont :

- Le programme Python : deep\_learning\_object\_detection.py
- Le fichier entraîné aux 21 types d'objets : MobileNetSSD\_deploy.caffemodel
- Le fichier de configuration : MobileNetSSD\_deploy.prototxt

Tout d'abord, nous importons les packages nécessaires pour que notre modèle puisse fonctionner- le module DNN est inclus dans cv2.

```
# import the necessary packages
import numpy as np
import argparse
import cv2
```

Figure 3.12 : Les bibliothèques d'importation.

Ensuite, nous analysons nos arguments de ligne de commande :

- `--image` : Le chemin vers l'image d'entrée.
- `--prototxt` : Le chemin vers le fichier prototxt Caffe.
- `--model` : Le chemin vers le modèle pré-entraîné.
- `--confidence` : Le seuil de probabilité minimum pour filtrer les détections faibles. La valeur par défaut est 50%.

```
# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
                help="path to input image")
ap.add_argument("-p", "--prototxt", required=True,
                help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
                help="path to Caffe pre-trained model")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
                help="minimum probability to filter weak detections")
args = vars(ap.parse_args())
```

Figure 3.13 : Arguments de ligne de commande.

Puis, initialisons la liste des étiquettes de classes que MobileNet SSD a été formé pour détecter.

```
CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
           "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
           "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
           "sofa", "train", "tvmonitor"]
COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))
```

Figure 3.14 : La liste des étiquettes de classes

Maintenant, nous devons charger notre modèle. Pour ce faire, on utilise la fonction : OpenCV readNetFrom...

```
# load our serialized model from disk
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])
```

**Figure 3.15 : Charge le modèle à utiliser**

Ensuite, nous avons chargé notre image de requête et préparé notre blob, que nous transmettrons via le réseau neuronal et obtiendrons les détections et les prédictions. Le "blob" est une étape de pré-processing, il faut donner au réseau une image dans le bon format afin qu'il l'accepte.

```
# load the input image and construct an input blob for the image
# by resizing to a fixed 300x300 pixels and then normalizing it
image = cv2.imread(args["image"])
(h, w) = image.shape[:2]
blob = cv2.dnn.blobFromImage(cv2.resize(image, (300, 300)), 1.0,
                             (300, 300), (104.0, 177.0, 123.0))
# pass the blob through the network and obtain the detections and predictions
print("[INFO] computing object detections...")
net.setInput(blob)
detections = net.forward()
```

**Figure 3.16 : Création du blob et envoi dans le réseau afin d'avoir une prédiction**

Enfin, nous commençons par boucler nos détections, en gardant à l'esprit que plusieurs objets peuvent être détectés dans une seule image. Nous appliquons également une vérification à la confiance (c'est-à-dire la probabilité) associée à chaque détection. Si la confiance est suffisamment élevée (c'est-à-dire au-dessus du seuil), nous afficherons la prédiction dans le terminal et dessinerons la prédiction sur l'image avec du texte et un cadre de sélection coloré.

Nous avons appliqué notre code sur des différents exemples pour détecter les objets possibles. Le tableau suivant montre les résultats de certaines des images testées.

Tableau 3.2 : Résultats de détection d'objets en utilisant SSD sur quelques images.

Image d'entrée	Image de sortie (Image après détection d'objets)
<p>A)</p> 	 <pre> xt.txt --model MobileNetSSD_deploy.caffemodel [INFO] loading model... [INFO] computing object detections... [INFO] boat: 99.99% </pre>
<p>B)</p> 	 <pre> otxt.txt --model MobileNetSSD_deploy.caffemodel [INFO] loading model... [INFO] computing object detections... [INFO] dog: 91.14% </pre>

C)



Output

car: 99.48%  
 horse: 99.85%  
 dog: 57.64%  
 person: 85.77%

```
Terminal: Local Local (2) +
[INFO] loading model...
[INFO] computing object detections...
[INFO] car: 99.48%
[INFO] cat: 57.13%
[INFO] dog: 57.64%
[INFO] horse: 99.85%
[INFO] person: 85.77%
```

D)



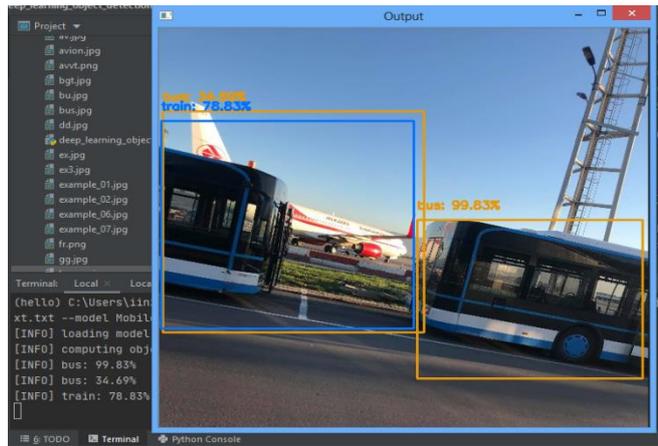
Output

car: 99.88%  
 person: 46.50%  
 person: 36.84%  
 person: 35.93%  
 person: 32.89%  
 person: 29.24%  
 person: 28.97%  
 person: 25.32%  
 person: 25.27%

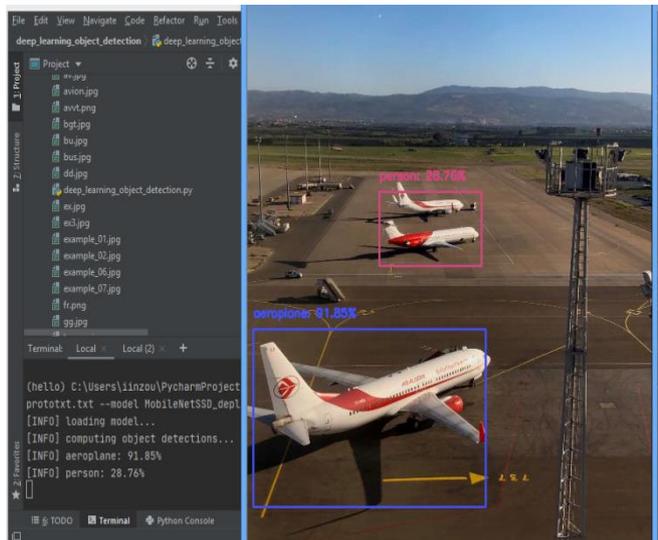
```
Terminal: Local Local
[INFO] person: 46.50%
[INFO] person: 36.84%
[INFO] person: 35.93%
[INFO] person: 32.89%
[INFO] person: 29.24%
[INFO] person: 28.97%
[INFO] person: 25.32%
[INFO] person: 25.27%
```

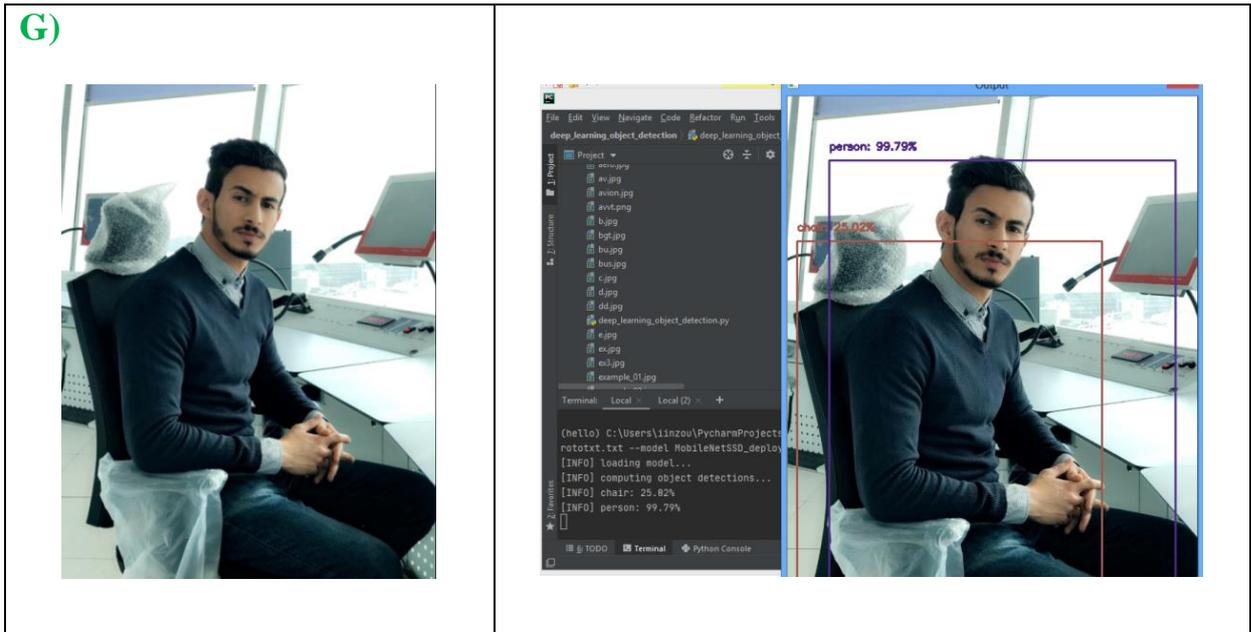
PyCharm 2020.1.4 available: // Update... (20/09/2020 16:21)

E)



F)





Les résultats obtenus sont encourageant avec une perte minimal de certaines objets en raison de leur similitude avec d'autres objets ou bien notre model n'est pas entérinée sur l'objet perdu.

Sur l'image **A** : On observe une détection parfaite.

Sur l'image **B** : On constate un échec dans la détection d'objet, au lieu de détecté un lion il a détecté un chien car notre model n'est pas entérinée sur cet objet, et le choix du chien n'est pas au pif parce que il a une ressemblance entre les 2 animaux.

Sur l'image **C** : On observe une détection parfaite avec une perte d'un seul objet (la personne qui porte une veste rouge là-bas), c'est parce qu'il est loin (défi d'échelle).

Sur l'image **D** : On observe une détection acceptable avec une perte de détection de quelques objets à cause d'Occultations partielle et les petits objets du loin (défi d'échelle).

Sur l'image **E** : On observe une détection parfaite des 2 bus mais un échec de détection d'avion à cause de défi (d'Occultations partielle).

Sur l'image **F** : On observe une détection parfaite d'avion AIR ALGERIE, mais une perte de détection des autres avions car ils sont très loin (défi d'échelle).

Sur l'image **G** : On observe une détection parfaite sur ma photo avec la chaise, mais une perte détection d'écran de supervision, car notre model n'est pas entérinée sur cet objet.

### 3.8 Comparaison de la méthode SSD avec la méthode Haar

Dans cette section, nous présentons une étude comparative entre la méthode SSD et Haar, pour démontrer l'efficacité et pour évaluer les performances de la méthode de détection d'objets par Deep Learning. Après les résultats obtenus on remarque que, pour les applications en temps réel, il n'est pas vraiment efficace d'utiliser les classificateurs Haar en raison du temps d'exécution élevé qu'ils nécessitent, tandis que la méthode SSD de Deep Learning a la possibilité d'entraînée et de détecté des objets au même temps avec des performances excellente. Ainsi que, Les réseaux de neurones convolutifs surpassent les classificateurs en cascade dans un certain nombre de cas. Il est plus que raisonnable de choisir CNN pour une raison de robustesse ou rapidité.

#### Haar méthode

#### SSD méthode



Figure 3.17 : Comparaison de la méthode SSD avec la méthode Haar

**3.9 Conclusion**

Nous avons présenté dans ce chapitre l'implémentation de l'approche de détection d'objet basé sur le Deep Learning, pour cela nous avons utilisé le réseau de neurones convolutifs (MobileNet) qui a été entraîné déjà sur 21 objets avec notre méthode (SSD) pour que nous assurons une meilleur précision avec le moins du temps. Pour une détection de plus d'objets soit vous utilisez un algorithme pré-entraîné sur une BDD des objets que tu veux ou bien vous prouvez entraîné l'algorithme sur votre propre BDD.

# *Conclusion Général*

### 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 dataset (BDD) plus volumineux et de modèles plus grands et une conséquence de nouvelles idées, d'algorithmes et d'architectures réseau améliorées. « Grâce aux Deep Learning, l'avenir de l'intelligence artificielle est prometteur ».

Bien que, le Deep Learning (DL) ait fait progresser le monde plus rapidement que jamais, il reste encore du chemin à parcourir. Nous sommes encore loin de comprendre pleinement comment fonctionne l'apprentissage profond, comment nous pouvons rendre les machines plus intelligentes, plus proches ou plus intelligentes que les humains, ou apprendre exactement comme les humains. Le DL a résolu de nombreux problèmes tout en prenant les technologies dans une autre dimension. Cependant, l'humanité doit faire face à de nombreux problèmes difficiles. Nous espérons que le deep learning et l'IA seront beaucoup plus consacrés à l'amélioration de l'humanité, à la réalisation des recherches scientifiques les plus dures, et enfin mais pas au moins, pour rendre le monde meilleur pour chaque être humain.

La détection d'objets est l'épine dorsale de nombreuses applications pratiques de la vision par ordinateur. Mais reste qu'on a des nombreux défis dans l'affichage des variables actuelles et toutes les difficultés auxquelles la cible est exposée. Bien que, les capacités des activités réalisées dans le domaine de la détection d'objets soient nombreuses, aucun moyen ne peut être considéré comme 100% fiable, mais sans ambages et selon la mesure, de nouveaux travaux tentent d'améliorer les scores pour obtenir de meilleurs résultats.

Après avoir achevé notre conception nous avons donné les outils nécessaires pour la réalisation de notre travail. Nous avons présenté aussi l'environnement de développement. Ensuite, nous avons présenté l'essentiel de notre implémentation et application en donnant quelques captures d'écran qui expliquent le déroulement et le fonctionnement de notre travail. Ainsi que, les résultats obtenus par la méthode SSD de détection d'objet qui donne meilleur précision avec le moins du temps. Enfin, avant de passer aux perspectives, ce travail nous a permis d'appliquer nos connaissances sur les réseaux de neurones et d'acquérir de nouvelles connaissances dans ce domaine, et le temps que nous avons passé à lire les articles a été une bonne initiation à la recherche.

Comme perspectives de futures recherches, nous souhaitons pour les prochains projets de fin d'étude, à :

- Élargir notre travail de détection d'objet mais selon différentes méthodes.
- Il est aussi intéressant de faire une étude comparative entre les différentes méthodes de détection des objets en temps réel pour voir leur performance dans le domaine de la vision par ordinateur.
- De travailler avec le modèle déjà entraîné tels que le VGG16 et faire une comparaison avec MobileNet.
- D'utiliser un algorithme pré-entraîné sur une BDD personnel.

### REFERENCES

- [1] Tang, C., Feng, Y., Yang, X., Zheng, C. et Zhou, Y. (2017). La détection d'objets basée sur le Deep Learning. 2017 4e Conférence internationale sur les sciences de l'information et l'ingénierie de contrôle (ICISCE).]
- [2] Saaod M .Rasheed1, Ahmet cinar, International Journal of Innovative Research in Science, Engineering and Technology May 2019
- [3] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, 2001, pp. I-I
- [4] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part-Based Models," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, pp. 1627-1645, 2010
- [5] <https://www.actuia.com/contribution/jeancharlesrisch/segmentation-et-detection-dobjets-en-temps-reel-avec-tensorflow>
- [6] International Conference on Artificial Intelligence and Information Technology MARCH 4–6, 2019, ALGERIA
- [7] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, 2005, pp. 886-893
- [8] thèse de doctora « SAAD Narimen ». Y. Freund et R. Schapire. "Experiments with a new boosting algorithm", International Conference on Machine Learning, pages 148–156
- [9] Mémoire de fin d'étude en vue de l'obtention du diplôme Master en informatique, BERKANE Chahrazed BERKANI Afef, 2014-2015
- [10] sakhri Ahmed Rami, Irgens P., Bader C., Le T., Saxena D., Ababei C. 'An efficient and cost effective FPGA based implementation of the Viola-Jones face detection algorithm ',(2017) HardwareX, 1 ,pp.68-75
- [11] M.FODDA. 'DETECTION ET RECONNAISSANCE DE VISAGE',19 Octobre 2010

## Bibliographie

---

- [12] Rainer Lienhart et Jochen Maydt, « An Extended Set of Haar-like Features for Rapid Object Detection », IEEE ICIP, 2002
- [13] Paul Viola et Michael Jones, « Robust Real-time Object Detection », IJCV, 2001
- [14] Samira Khan, ADVANCED COMPUTER ARCHITECTURE ML Accelerators, University of Virginia Feb 6, 2019
- [15] <https://www.researchgate.net/publication/332781107>
- [16] Li Liu<sup>1,2</sup> · Wanli Ouyang<sup>3</sup> · Xiaogang Wang<sup>4</sup> · Paul Fieguth<sup>5</sup> · Jie Chen<sup>2</sup> · Xinwang Liu<sup>1</sup> · Matti Pietikäinen<sup>2</sup> International Journal of Computer Vision Deep Learning for Generic Object Detection Accepted: 26 September 2019
- [17] Farhana Sultana<sup>1</sup> , Abu Sufian<sup>1</sup> , Paramartha Dutta<sup>2</sup>, Review of Object Detection Algorithms using CNN, ICCDC 2019, HIT, Haldia
- [18] FUKUSHIMA Kunihiko, MAYAKE Sei. Neocognitron: a self-organizing neural network model for a mechanism of visual pattern recognition. 1982
- [18] O'SHEA Keiron, NASH Ryan. An introduction to convolutional neural networks. 2015
- [19] WANG Peng, XU Jiaming, XU Bo, LIU Chenglin, ZHANG Heng, WANG Fangyuan HAO Hongwei. Semantic clustering and convolutional neural networks for short text categorization. 2015
- [20] <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
- [21] Jussi Jokela, PERSON COUNTER USING REAL-TIME OBJECT DETECTION AND A SMALL NEURAL NETWORK, URKU UNIVERSITY OF APPLIED SCIENCES Information and Communications Technology, 2018
- [22] <http://cs231n.github.io/convolutional-networks>
- [23] <https://medium.com/@junehaoching/beginners-guide-to-understanding-convolutional-neural-networks-5209e5d9f717>
- [24] [https://jpis.org/ViewImage.phpType=F&aid=612629&id=F1&afn=1150\\_JPIS\\_48\\_2\\_114&fn=jpis-48-114-g001\\_1150JPIS](https://jpis.org/ViewImage.phpType=F&aid=612629&id=F1&afn=1150_JPIS_48_2_114&fn=jpis-48-114-g001_1150JPIS)

## Bibliographie

---

- [25] <https://www.intelligence-artificielle-school.com/>
- [26] <https://www.lebigdata.fr/machine-Learning-et-big-data>
- [27] international University,, <https://www.supinfo.com/articles/single/6041-machine-Learning-introduction>
- [28] <https://medium.com/@bobkrc/machine-learning-apprentissage-supervis%C3%A9-ou-non-supervis%C3%A9-bced5be4fd7f>
- [29] Mèmoir de Master 2, OULMI Mehdi - KALOUNE Salim, Classification d'objets avec le Deep Learning, Université Akli Mohand Oulhadj de Bouira 2017/2018
- [30] Par Michelle Knight le 30 juillet 2020, <https://www.dataversity.net/what-is-Deep-Learning/>
- [31] Par Keith D. Foote ,7 février 2017 <https://www.dataversity.net/brief-history-Deep-Learning/#>
- [32] <https://www.futura-sciences.com/tech/definitions/intelligence-artificielle-Deep-Learning-17262/>
- [33] Li Liu<sup>1,2</sup> · Wanli Ouyang<sup>3</sup> · Xiaogang Wang<sup>4</sup> · Paul Fieguth<sup>5</sup> · Jie Chen<sup>2</sup> · Xinwang Liu<sup>1</sup> · Matti Pietikäinen<sup>2</sup>, International Journal of Computer Vision, Deep Learning for Generic Object Detection: A Survey, 26 September 2019
- [34] <https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464>
- [35] Yann LeCun<sup>1,2</sup>, Yoshua Bengio<sup>3</sup> & Geoffrey Hinton Université de Montréal Article in Nature · May 2015
- [36] <https://www.futura-sciences.com/tech/definitions/intelligence-artificielle-Deep-Learning-17262/>
- [37] Zhong-Qiu Zhao, Member, IEEE, Peng Zheng, Shou-tao Xu, and Xindong Wu, Fellow, IEEE ; Object Detection with Deep Learning, 16 Apr 2019
- [38] Wei Liu, et al., “SSD: Single Shot MultiBox Detector”, ECCV 2016

## Bibliographie

---

- [39] <https://missinglink.ai/guides/convolutional-neural-networks/faster-r-cnn-detecting-objects-without-wait/>
- [40] <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e>
- [41] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, 6 Jan 2016
- [42] Song, X., Jiang, P., & Zhu, H. (2019). Research on Unmanned Vessel Surface Object Detection Based on Fusion of SSD and Faster-RCNN. 2019 Chinese Automation Congress (CAC)
- [43] B.Tech, Conseiller: Sk.M.Gouse, OBJECT DETECTION AND IDENTIFICATION A Project Report, November 2019
- [44] Zhao, Y., Zhao, J., Zhao, C., Xiong, W., Li, Q., et Yang, J. (2019). Détection d'objets en temps réel robuste basée sur l'apprentissage en profondeur pour des images de télédétection à très haute résolution. IGARSS 2019-2019 Symposium international de l'IEEE sur les géosciences et la télédétection
- [45] [https://medium.com/@jonathan\\_hui/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06](https://medium.com/@jonathan_hui/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06)
- [46] <https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11>
- [47] Canziani A., Molnar T., Burzawa L., Sheik D., Chaurasia A., Culurciello E., (September 8, 2018). Analysis of deep neural networks. Retrieved from <https://medium.com/@culurciello/analysis-of-deep-neural-networks-dcf398e71aae>
- [48] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., et Berg, .SSD: Détecteur MultiBox Single Shot. Notes de cours en informatique, 29 Dec 2016
- [49] <https://fr.wikipedia.org/wiki/NumPy>
- [50] G.BRADSKI & A.KAELHER, "Learning OpenCV Computer Vision with the OpenCV Library" O'Reilly, Sebastopol, 2008

## Bibliographie

---

[51] Sagi, B., Nemat-Nasser, S. C., Kerr, R., Hayek, R., Downing, C., & Hecht-Nielsen, R. (2001). A biologically motivated solution to the cocktail party problem. *Neural Computation*, 13(7), 1575-1602.

[52] <https://fr.wikipedia.org/wiki/PyCharm>

[53] Patrick Audriaz, Travail de Bachelor, Filière télécommunications, orientation internet et communication, Haute école d'ingénierie et d'architecture Fribourg, 9 juillet 2019

[54] Jia, Yangqing and Shelhamer, Evan and Donahue, Jeff and Karayev, Sergey and Long, Jonathan and Girshick, Ross and Guadarrama, Sergio and Darrell, Trevor. *Caffe: Convolutional architecture for Fast Feature Embedding*, 2014.

[55] <https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>