République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche
ScientifiqueUniversité Saad Dahlab_Blida1

Faculté des sciences

Département

d'Informatique

icosnet

Mémoire de fin d'études pour l'obtention du diplôme de master en informatique

Option : Traitement Automatiques de la Langue

Thème :

Solution pour déterminer l'authenticité des documentsnumériques

Réalisé par :

Benadouda Asmaa

Merdji Roufaida

Sous la direction de :

Pr. Benblidia Nadjia Promotrice

Mr. Brahim Aimen Encadreur

2021/2022

Remerciements

Mes vifs remerciements sont d'abord à adressés à mon promotrice

Mme **BENBLIDIA Nadjia** et à mon encadreur Mr **BRAHIM Aimen** qui mon fait l'honneur de diriger ce travail de recherche.

Je tiens à leur exprimer ma gratitude et mon profond respect.

Je tiens aussi à exprimer ma gratitude à mes chers professeurs pour la qualité de l'enseignement qu'ils m'ont prodigué au cours de ses années.

Je remercie également tous ceux qui m'ont aidé à réaliser ce travail de façon directe ou indirecte.

Dédicaces

Je tiens à remercier mon DIEU pour toutes les bénédictions qu'il m'a données.

Je tiens à remercier mes chers parents qui m'ont aidé et soutenu pour tout, depuis ma naissance jusqu'à aujourd'hui.

Je tiens à remercier profondément mon promotrice Madame BENBLIDIA Nadjia et mon Encadreur M Brahim Aimen pour leurs suivi et ses efforts. Ensuite je remercie l'ensemble des membres de Jury qui mon fait l'honneur de bien vouloir étudier mon travail.

Je tiens à remercier Tous les membres de la famille MERDJI.

Je tiens à remercier Toute mes chères AMIES de la cité 5 pour le partage des bons et mauvais moments.

Merdji Rofaida

Dédicaces

Je dédie ce travail :

A mes parents qui m'ont soutenu et encouragé durant ces années d'études, qu'ils trouvent ici le témoignage de ma profonde reconnaissance.

A mes frères et ma petite sœur, mes grands parents et ceux qui ont partagé avec moi les moments d'émotions lors de la réalisation de ce mémoire.

A tous mes amis qui m'ont toujours encouragé tout au long de mon parcours

BenAdouda Asmaa

Résumé

La reconnaissance d'objets est une technique de Computer Vision utilisée pour l'identification d'objets présents dans des images et des vidéos. La reconnaissance d'images fait référence aux technologies qui identifient les lieux, les logos, les personnes, les objets, les bâtiments et plusieurs autres variables dans des images. L'objectif de ces domaines est d'apprendre aux machines à comprendre (reconnaitre) le contenu d'une image comme le font des humains.

Le but de notre travail est de concevoir et réaliser un système de classification d'images de documents (Cartes d'identités) de manière automatique. Ce système tend à répondre au problème rencontré par l'entreprise Icosnet dont l'objectif est de déterminer le faux du vrai des pièces d'identification, pour que l'entreprise puisse rendre ses services aux clients qui ont inséré leurs pièces d'identités.

Pour réaliser notre système, nous avons utilisé le Deep Learning via les réseaux de neurones convolutionnels (CNN) et le modèle de détection d'objets YOLOv5.

Après le partitionnement de la base de données et l'entrainement par les modèles cités, l'évaluation des résultats a engendré une précision de la classification de 73.4% ; le Rappel est de 95.5%, et le mAP@0.5 est de 92.8%.

Mots clés : Reconnaissance d'objets, Reconnaissance d'images, Classification d'image, Deep Learning, CNN, Yolov5, Carte d'identité.

Abstract

Object recognition is a Computer Vision technique used for object identification present in images and videos, and image recognition refers to technologies that identify places, logos, people, objects, buildings and several other variables in images. The objective of these domains is to teach machines to understand (recognize) the content of an image as humans do.

The purpose of our work is to realize and implement a system of classification of document images (Identity cards) in an automatic way, this system tends to answer the problem encountered by the company Icosnet whose objective is to determine the false true identification documents, so that the company can provide its services to customers who have inserted their identity documents.

To realize our system, we used Deep Learning via convolutional neural networks (CNN) and the object detection model YOLOv5

After the partitioning of the database and the training by the cited models, the evaluation of the results generated a precision of the classification of 73.4%; Recall is 95.5%, and mAP@0.5 is 92.8%.

Keywords: Objects recognition, Image recognition, Image classification, Deep Learning, CNN, Yolov5. National ID card.

ملخص

التعرف على الكائنات هو أحد تقنيات الرؤية الحاسوبية المستخدمة للتعرف على الأشياء الموجودة في الصور ومقاطع الفيديو ، ويشير التعرف على الصور إلى التقنيات التي تحدد الأماكن والشعارات والأشخاص والأشياء والمباني والعديد من المتغيرات الأخرى في الصور. الهدف من هذه المجالات هو تعليم الآلات لفهم (التعرف) على محتوى الصورة كما يفعل البشر.

الغرض من عملنا هو تحقيق وتنفيذ نظام تصنيف لصور المستندات (بطاقات الهوية) بطريقة آلية ، ويميل هذا النظام إلى الإجابة على المشكلة التي تواجهها شركة Icosnet التي تهدف إلى تحديد وثائق الهوية الحقيقية الكاذبة ، لذلك أن الشركة يمكن أن تقدم خدماتها للعملاء الذين أدخلوا وثائق هويتهم.

لتحقيق نظامنا ، استخدمنا التعلم العميق عبر الشبكات العصبية التلافيفية (CNN) ونموذج اكتشاف الكائن YOLOv5

بعد تقسيم قاعدة البيانات والتدريب من خلال النماذج المذكورة ، نتج عن تقييم النتائج دقة تصنيف تصل إلى 73.4٪ ؛ نسبة الاستدعاء 95.5٪ ، و mAP@0.5 هي 92.8٪.

الكلمات المفتاحية: التعرف على الأشياء ، التعرف على الصور ، تصنيف الصور ، التعلم العميق ، CNN، Yolov5. بطاقة الهوية الوطنية.

Table des matières

Remerciement

Résumé

Abstract

ملخص

Introduction générale……18

Recherche Bibliographique

Chapitre I : Généralités sur l'image numérique

Introduction……..……...21

1- Notion de définition et propriété d'une image …….……...21

2- Image……..…….21

2-1Image numérique……..……...…….…….21

2-2Caractéristiques d'une image numérique ……...……..……22

2-2-1Pixel …..…….…….22

2-2-2Dimension ……..…...23

2-2-3Résolution ……...….23

2-2-4Niveau d'intensité/ luminance 23

3- Différentes Types d'image …….……...23

3-1Images binaires (Bichromes) **……...…….23**

3-2Images à niveaux de gris (Monochromes)…..……...24

3-3Images en couleurs (Polychromes)……..……..25

4- Classification des images ……25

4-1Classification des images et machine learning ……..…..…..25

4-2Classification des images et les réseaux de neurones…….26

4-3Classification des images et le modèle de détection yolo ……..……..27

Conclusion……...…….……27

Chapitre II : Réseaux de neurones convolutifs et apprentissage profond

1-reseaux de neurones……...29

1-1Introduction …….……..29

1-2Système de réseau de neurones artificiels……..29

1-3Relation avec la biologie……30

Table des matières

1-4Réseau de neurones artificiels ……...30

1-5 Fonctions d'activations ……..…….....32

1-5-1Fonction "sigmoïde"……..……32
1-5-2Fonction "RELU"……..33
1-5-3La tangente hyperbolique……..…….……..34
1-5-4Fonction Softmax ……..…….…….34
1-6Architecture de réseau de neurones ……..…….……...35
1-7Réseau de neurones à réaction 'feed-forward'……..…….……...36
1-8Réseau de neurones récurent...……...36
1-9Différence entre 'reccurent neural network' et 'feed-forward neural network'…… 37

2-Apprentissage Automatique et Apprentissage profond…….....37

2-1 Introduction…...……....37

2-2 Apprentissage automatique 'Machine Learning'……..……...38

2-3 Apprentissage profond 'Deep Learning'……...38

2-4 Apprentissage entre 'Machine Learning' et 'Deep Learning'……...39

2-5Réseau de neurones à convolution 'Convolution neural network CNN'……...39

2-5-1 Qu'est-ce que le réseau neuronal convolutif (CNN ou ConvNet) ?.....39

2-5-2 Architecture et fonctionnement d'un CNN……..……..39

2-5-2-1partie de convolution.……40

2-5-2-2Partie Classification…..……..43

2-6Types d'apprentissages……...…….……...44

2-6-1Apprentissage supervisé……..……44

2-6-2Apprentissage non-supervisé…….....44

2-6-3Apprentissage par renforcement……44

2-7Mesures de performances…….……...……...45

2-7-1Matrice de confusion…….45

2-7-2Exactitude 'Accuracy'……...46

2-7-3Taux ……..46

2-7-4F-mesure 'moyenne harmonique'……46

2-7-5La courbe ROC (Received Operating Characteristic) …..……..46

Conclusion…….....……48

Chapitre III : Modèles de détection d'objets

Introduction ……..50

Table des matières

1- Détection d'objets……..……..50

1-1Définition……..……50

1-2 Méthodes……...……..……..50

1-3Modèles de détection par CNN……..…..……...……..…51

1-3-1 R-CNN……..……..52

1-3-2 Fast R-CNN…..……..52

1-3-3 Faster CNN……53

1-3-4 Mask R-CNN……..……54

1-3-5 SSD : Single Shot MultiBox Detector..……..54

1-3-6 YOLO……..……...55

1-3-6-1 Modèle YOLOv2…....……...59

1-3-6-2 Modèle YOLOv3…..…..……...60

1-3-6-3 Modèle YOLOv4…….……...61

1-3-6-4Modèle YOLOv5……..……...65

Conclusion…….……...…...66

Chapitre IV : Conception et implémentation

I Conception

1-Introduction ……...……68

2-Conception générale de notre système ……..……68

2-1 La Collecte de la base de donnés……...…...……68

2-1-1 Choix des classes du modèle1 ..……...69

2-1-2 Choix des classes du modèle2……..……69

2-2 Prétraitement de la base de donnés ……..……69

2-2-1 prétraitement sur la1ere base……..……69

2-2-2 prétraitement sur la1ere base ……..……...71

2-3 Apprentissage des modèle yolov5……..……..……..……...72

2-3-1 Attachement du yolov5sur l'environnement……..……...72

2-3-2 Préparation du modèle pour le training ……..……..……73

2-3-3 Détection du modèle ……...……..74

2-3-4 Etape pour détecter les coordonnées de chaque classe dans la carte (bouding box)…….……...…….74

3-3-5 Analyse une carte ……..……..….74

3-3-6 Récupération des autres informations…..……76

2-4 Tests et résultats ……...……77

Table des matières

2-4-1 Discussions …..……..……...78
2-4-2 Discussions sur yolo……..…….……..83
3-Architecture de la conception de notre système d'authentification ……..84
3-1 Architecture modèle1 yolov5 ….……...85
3-2 Architecture modèle2 yolov5 ……...…….86
3-3 Prétraitement de donnés (1) (2) ……..…..……...87
3-4 Apprentissage de yolov5 (modèle 1et2)….……89

II implémentation

4-Matériels et outils……..91

4-1 Matériel utilisé……...……91

4-2 Outils de développement ……...……...91
4-2-1 Google Colab ……...……...91
4-2-2 Python ...……..91
4-2-3 Opencv ……...…….92
4-2-4 Pytorch …...……92
4-2-5 PIL ..……93
4-2-6 Numpy..……...93
4-2-7 matplotlib…...……..93
4-2-8 Google Dtive ……...……94
4-2-9 makesense ……...……94
4-2-10 Fichier .Yaml……...……..94

Conclusion ……..…….95

Conclusion générale ……..……...96

Références bibliographiques
Annexe

Table des figures

Chapitre I : Généralités dur l'image numérique

Figure I.1 : Exemple d'une carte d'identité algérienne……...21

Figure I.2 : Représentation d'une image numérique ……...22

Figure I.3 : Image et pixel……...23

Figure I.4: Example de résolution de l'image…….23

Figure I.5 : Une image binaire, avec le tableau de valeurs correspondant. L'image est représentée en négatif (les valeurs 1 sont en noir valeurs 0 sont en blanc), afin de mieux visualiser les structures.....24

Figure I.6: Exemple d'image en niveaux de gris……24

Figure I.7 : Une image en niveaux de gris, agrandissement d'une zone de l'image, et affichage des valeurs constituant la matrice image ……..24

Chapitre II : Réseaux de neurones convolutifs et apprentissage profond

Figure II.1 : Architecture de réseau neuronal simple……..29

Figure II.2 : Structure d'un neurone biologique……..30

Figure II.3 : Réseau de neurone artificiel simple…….31

Figure II.4 : Réseau de neurones avec biais en entrée, x0 = +1, et poids wk0 = bk……....32

Figure II.5 : La représentation graphique de sigmoïde……33

Figure II.6 : La representtation graphique de Relu…….....33

Figure II.7 : La représentation graphique de tanh……...34

FigureII.8: La représentation graphique de la fonction softmax…….35

Figure II.9 : Architecture d'un ANN……...35

Figure II.10 : Réseaux de neurones à réaction…….36

Figure II.11: Réseau de neurones récurrent 'Recurrent Neural Network'…….....36

Figure II.12 : Explication de réseau de neurones récurrent 'Recurrent Neural Network'…...37

Figure II.13 : La relation entre l'IA, Machine Learning et Deep Learning…….38

Figure II.14 : Architecture de base d'un réseau neuronal convolutif……..39

Figure II.15 : Parcours de la fenêtre de filtre sur l'image……40

Figure II.16 : Le principe de convolution……41

Figure II.17 : Fonctionnement de la fonction RELU……..42

Figure II.18 : Le principe de Pooling avec un filtre 2*2 et un pas de 2……..42

Figure II.19 : La couche Fully-connected……...43

Figure II.20 : Illustration du pas de 2 pixels……44

Figure II.21 : Les différents types de l'apprentissage automatique…….45

Figure II.22 : Exemple d'illustration de la courbe ROC……47

Figure II.23 : Schéma des modèles de détection et classification par CNN……47

Chapitre III : Modèles de détection d'objets

Figure III.1 : Étapes importantes de la détection et de la reconnaissance des objets……..51

Figure III.2 : Architecture du modèle R-CNN……52

Figure III.3 : Architecture du modèle Fast R-CNN……53

Figure III.4 : Architecture du modèle Faster R-CNN……53

Figure III.5 : Architecture du modèle Mask R-CNN……54

Figure III.6 : Architecture du modèle SSD……...55

Figure III.7 : Architecture du modèle YOLO……56

Figure III.8 : image cellulaire ……..……56

Figure III.9 : Le vecteur prédit dans le cas d'une seule boite……...57

Figure III.10 : Union sur Intersection……...57

Figure III.11 : Exemples d'IoU……58

Figure III.12 : Le vecteur prédit dans le cas de plusieurs boites dans la cellule……...58

Figure III.13 : Un tenseur qui spécifie les emplacements de la boîte englobante et probabilités de classe…….....59

Figure III.14 : Architecture de darknet19……60

Figure III.15 : Architecture du modèle YOLOv3…….....61

Figure III.16 : Architecture du modèle YOLOv4……62

Figure III.17 : Architecture du darknet53……63

Figure III.18 : Une structure SPP……..63

Figure III.19 : Architectures de Réseau d'agrégation de chemins (PAN)…….64

Figure III.20 : Modification des SAM et PAN……64

Figure III.21 : architecture du YOLOv5……...65

Figure III.22 : Graphe de performance……66

Chapitre IV : Conception et implémentation

Figure IV.1 : exemple d'image valide de la base de donné ……68

Figure IV.2 : exemple d'image non valide de la base de donné ……69

Figure IV.3 : images valide de la base de données avec son annotation……70

Figure IV.4 : images non valide de la base de données avec son annotation……..70

Figure IV.5: images valide de la base de données avec sonannotation…….……..71

Table des figures

Figure IV.6 : les bases de données de train et de test……..72

Figure IV.7 : Montage sur Google Drive……72

Figure IV.8: Installing the YOLOv5 Environment……..72

Figure IV.9 : installation des exigences……...72

Figure IV.10 : paramètre du notebook **……73**

Figure IV.11 : Mettre en œuvre le processus de formation pour le premier base de donné ……..…….73

Figure IV.12 : Mettre en œuvre le processus de formation pour le deuxième base de donné **……..……73**

Figure IV .13: Emplacement de fichier Yaml dans yolov5…….74

Figure IV.14 : Exemple de détection de classe ……...…74

Figure IV.15 : Exemple d'extraction des informations (la détection)…….74

Figure IV.16 : load a model (custom trained model)……..75

Figure IV.17 : le seuil de la confiance ……75

Figure IV.18: Path and size ……...75

Figure IV.19 : GET labels and coordinate……...75

Figure IV.20 : importation des librairies…….…75

Figure IV.21 : Code python **……76**

Figure IV.22 : Extraction de face image ……76

Figure IV.23 : Code python pour la récupération des autres informations……..77

Figure IV.24 : Le Résultat Obtenu de Détection par Yolov5 sur le **model(1)……77**

Figure IV.25 : Le Résultat Obtenu de Détection par Yolov5 sur le **model(2)……78**

Figure IV.26 : Précision, rappel et mAP pour le model (1)……78

Figure IV.27 : résultat obtenu pour la valeur de précision, recall et le mAP model(1)…79

Figure IV.28 : résultat de la classification de test carte……...79

Figure IV.29 : Précision, rappel et mAP pour le model (2)……..…80

Figure IV.30 : résultat obtenu pour la valeur de précision, recall et le mAP model (2)……..80

Figure IV.31 : résultat sur les informations détecter **……..80**

Figure IV.32 : résultats obtenus de l'obtention des coordonnées avec l'index de labels …...81

Figure IV.33 : découpage de la photo ……82

Figure IV.34 : Résultat d'extraction de face-image…….82

Figure IV.35 : illustration du résultat ……83

Table des figures

Figure IV.36 : Architecture générale du système……84
Figure IV.37 : architecture du système d'authentification de documents……85
Figure IV.38 : architecture générale d'extraction des informations (2)……..86
Figure IV.39 : Exemple d'annotation nc=5 ……87
Figure IV.40 : Data directories structure (Datacard)……87
Figure IV.41 : Data directories structure (Datacardvalide)……...88
Figure IV.42 : fichier Yaml pour la premier base de donné……...88
Figure IV.43 : fichier Yaml pour la deuxième base de donné ……...89
Figure IV.44: Architecture de yolov5l6……...90
Figure IV.45 : Architecture de yolov5l6……..90
Figure IV.46 : logo de google colab ……...91
Figure IV.47 : Logo de Python……92
Figure IV.48 : logo OpenCv……92
Figure IV.49 : logo de PyTorch……...93
Figure IV.50 : logo de Pillow……..93
Figure IV.51 : logo NumPY……93
Figure IV.52 :logo matplotlib……..94
Figure IV.53: logo Google Drive……94
Figure IV.54 : Logo MakeSense ……94

Liste des tableaux

Tableau II.1 : Matrice de confusion ……..……45

TableauIV.1 : Les résultats de l'apprentissage model(1)……79

TableauIV.2 : Les résultats de l'apprentissage model(2)……83

Liste des abréviations

- **IA :** Intelligence Artificielle
- **DL :** Deep learning
- **ML** Machine learning
- **CNN:** Convolutional neural network
- **RNN :** reccurent neural network
- **ReLu :** RectifiedLinear Unit
- **MLP** Multi Layer perceptron
- **SSD:**Single Shot MultiBox Detector
- YOLO:You Only Look Once
- **SVM**: support vecteur machine
- **NN :** Neural Network
- **ANN:** Artificiel Neural Network
- **RNN** : Reccurent Neural Network
- **PPP** :point par pouce
- **DPI :** Dots per inch
- **IoU:** intersection sur union
- **SNM**: Suppression non maximale
- **SPP** :Pooling pyramidal spatial
- **PAN**: Path Aggregation Network

Introduction générale

Aujourd'hui nous avons tous remarqué l'énorme développement des capacités logicielles, informatiques, théorique et logistiques au cours de la dernière décennie. Un apport considérable a été fait dans le développement des algorithmes de l'intelligence artificielle dans plusieurs domaines tels que : la vision par ordinateur, la vidéosurveillance, les voiture autonomes, la robotique, l'aéronautique, les maisons intelligentes, des cités intelligentes… etc.

Avec l'essor des technologies nécessitant l'intelligence en général, et la reconnaissance et la détection d'objets en particulier, il est devenu nécessaire de développer des méthodes robustes assurant plus de précision et de rapidité.

Les besoins en matière d'analyse d'images et de reconnaissance d'objets grandissent au fur et à mesure que les ordinateurs contiennent de plus en plus d'images et de vidéos. Actuellement, beaucoup d'entreprises demandent des documents tels qu'une pièce d'identité, un permis de conduire, ou un passeport comme pièce d'identification. Ces différentes peuvent être insérées par l'utilisateur, comme données numériques, afin que le système puisse l'authentifier et lui offrir les services sollicités.

Icosnet SPA (**S**ociété **P**ar **A**ction), un opérateur algérien d'accès Internet, communication unifiée, de services d'hébergement et de cloud, fait face à ce problème. Elle souhaite disposer d'une solution informatique qui a pour but de déterminer le faux du vrai des pièces d'identification.

L'objectif de ce travail est de proposer un système intelligent pour la détection des pièces d'identifications. Ces dernières, acquises dans de mauvaises conditions, se caractérisent par différentes contraintes d'occlusion, d'éclairage et de faible résolution. Pour réaliser ce système, nous avons opté pour l'apprentissage profond en choisissant d'utiliser le modèle YOLO qui a prouvé ses performances dans la détection d'objets en temps réel.

Pour atteindre cet objectif, nous avons structuré ce mémoire en quatre chapitres

Le premier chapitre donne une présentation générale sur l'image et ses caractéristiques.

Le chapitre deux vise à définir les divers concepts de base de l'apprentissage profond (Deep Learning) ; les principaux modèles y seront décrits.

Le troisième chapitre : est consacré à l'étude des différentes architectures des méthodes de Deep learning pour la détection d'objets. Avec les différentes versions de notre modèle choisi YOLO.

Le quatrième chapitre : nous présentons la conception de notre système de détection avecles résultats expérimentaux obtenus par le modèle YOLOv5

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

Recherche Bibliographique

Chapitre I :
Généralités sur l’image numérique

Introduction

Dans nos jours l'image joue un rôle important plus que décoratif, elle est considérée comme l'un des moyens les plus efficaces pour communiquer avec les autres, aussi elle est capable de transmettre un message, donc elle facilite la compréhension.

De ce fait le traitement d'image est un ensemble des méthodes et techniques qui permettent de modifier et améliorer le résultat visuel d'une image, dans le but de récupérer des informations spécifiques et des jugées pertinentes. Dans ce chapitre nous rappelons quelques notions fondamentales sur le traitement d'image.

1- Notion de définition et propriété d'une image

Une carte d'identité parfois dénommée carte nationale, est un document officiel qu'une personne doit produire pour prouver de son identité (nom, prénom, date et lieu de naissance, nationalité, le sexe, le groupe sanguin), elle est utilisée pour passer un examen, voter, payer avec un chèque.

Figure I.1 : Exemple d'une carte d'identité algérienne

2- L'image

Est un ensemble structuré d'informations, qui donne une signification pour l'œil humaine, après affichage.

C'est aussi une représentation exacte ou analogique d'une scène réelle.

Elle est sous la forme d'une fonction I (x, y) de brillance analogique continue, définie dans un domaine borné, tel que x et y sont les coordonnées spatiales d'un point de l'image et I une fonction d'intensité lumineuse et de couleur. Sous cet aspect l'image est inexploitable par la machine, ce qui nécessite sa numérisation **[2].**

2-1 L'Image numérique

Dans le sens général image numérique désigne toute image qui a été acquise, traitée et sauvegardée sous une forme codée représentable par des nombres (valeurs numériques).

L'image numérique est une image dont la surface est divisée en éléments d'une taille fixe appelés cellules ou pixels, ayant chacune comme caractéristiques : le niveau de couleurs (où De gris) prélevé et l'emplacement correspondant dans l'image réelle.

La numérisation d'une image est la conversion de celle-ci de son état analogique en une image numérique représentée par une matrice bidimensionnelle de valeurs numériques f (x, y).
x, y : coordonnées cartésiennes d'un point de l'image.
f (x, y) : niveau d'intensité.

Pour des raisons de commodité de représentation pour l'affichage et l'adressage, les données images sont généralement rangées sous forme de tableau *I* de *n* lignes et *p* colonnes où chaque élément *I* (*x, y*) représente un pixel de l'image et sa valeur est associée à un niveau de gris codé sur *m* bits (2*m* niveaux de gris, 0 = *noir,* 2*m* − 1 = *blanc*).
La valeur en chaque point exprime la mesure d'intensité lumineuse perçue par le capteur.

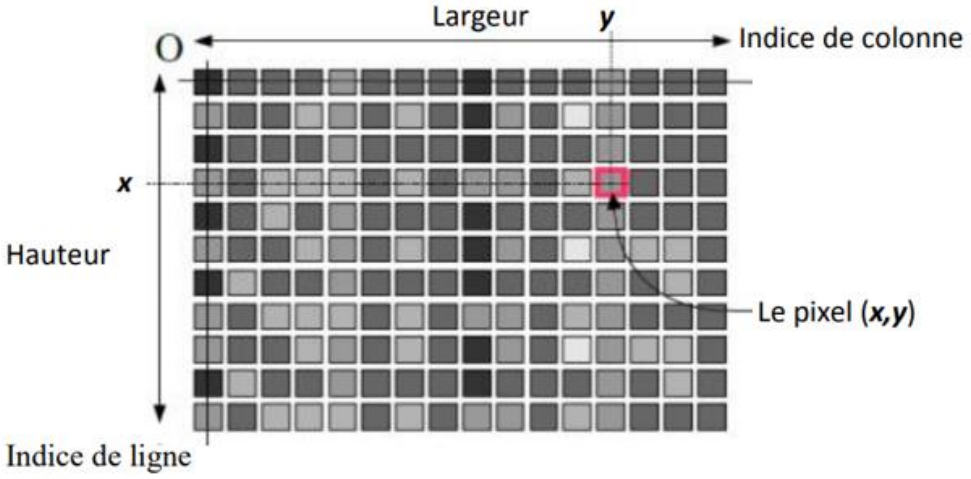


Figure I.2 : Représentation d'une image numérique **[2]**

2-2 Caractéristiques d'une image numérique

2-2-1 Pixel

Contraction de l'expression anglaise « Picture éléments » : éléments d'image, représente le plus petit élément d'une surface d'affichage, par exemple sur un écran d'ordinateur. Il est souvent présenté comme un petit carré de couleur.

Lorsqu'on zoome sur une image numérique matricielle (les images vectorielles ne sont pas basées sur les pixels), on aperçoit en effet que celle-ci se compose d'une multitude de petits carrés comme illustré dans la figure 2 suivante.

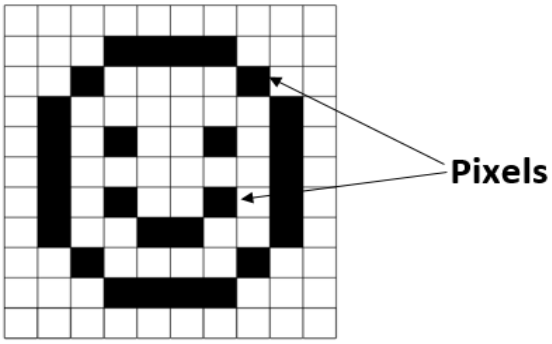


Figure I.3 :Image et pixel [3]

2-2-2 Dimension

C'est la taille de l'image elle se représente sous forme de matrice dont leur nombre de colonne est la hauteur de l'image et leur nombre de ligne est la largeur de l'image et ses éléments sont des valeurs numériques qui représentent des intensités lumineuses (en pixel).

2-2-3 Résolution

Désigne le nombre de pixels par unité de longueur de l'image, elle s'exprime en DPI (dots per inch) ou en ppp (points par pouce) et chaque pouce représente 1.54 cm, Plus le nombre de pixels est élevé par unité de longueur (plus la résolution est élevée) meilleure est la qualité de l'image, autrement dit la résolution d'une image correspond au niveau de détails qui vont être représentés sur une image, voir l'exemple ci-dessous.

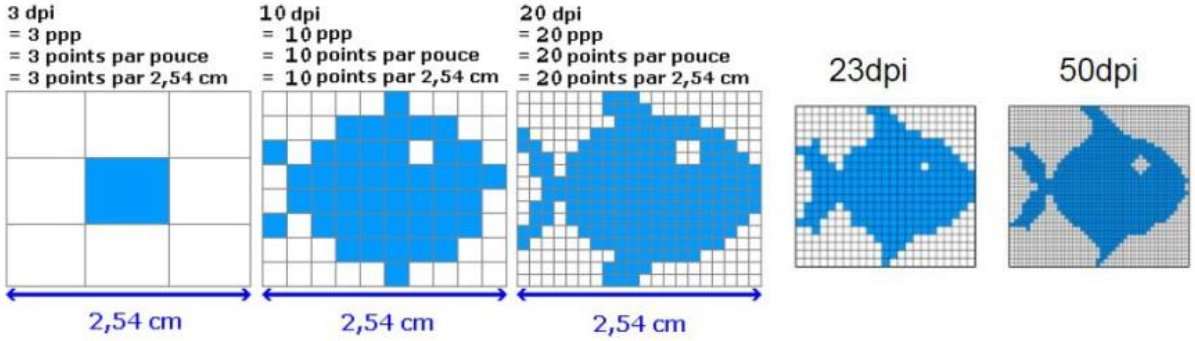


Figure I.4 : Exemple de résolution de l'image [3]

2-2-4 Niveau d'intensité/ luminance

C'est le degré de luminosité des points de l'image. Elle est définie aussi comme étant le quotient de l'intensité lumineuse d'une surface par l'aire apparente de cette surface, pour un observateur lointain, le mot luminance est substitué au mot brillance, qui correspond à l'éclat d'un objet.[4]

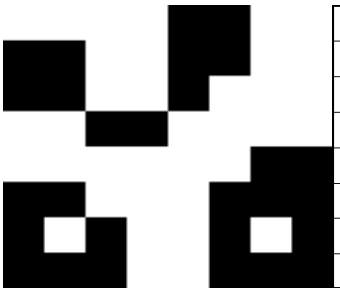
3- Différentes Types d'image

On distingue 3 types d'image

3-1 Images binaires (Bichromes)

C'est les images les plus simple qui n'ont que deux valeurs possibles pour stocker le pixel, dans les valeurs valent soit 0(noir) soit 1(blanc).

Généralement, les deux couleurs utilisées pour une image binaire sont le noir et le blanc.



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

Figure I.5: Une image binaire, avec le tableau de valeurs correspondant. L'image est représentée en négatif (les valeurs 1 sont en noir valeurs 0 sont en blanc), afin de mieux visualiser les structures**. [5]**

3-2 Images à niveaux de gris (Monochromes)

Le niveau de gris est la valeur de l'intensité lumineuse d'un pixel, le pixel est encodé sur un octet (8 bits) alors les valeurs sont dans l'intervalle [0,255] et a donc 256 valeurs possibles de gris. 0 pour le noir et 255 pour le blanc.

Figure I.6 : Exemple d'image en niveaux de gris

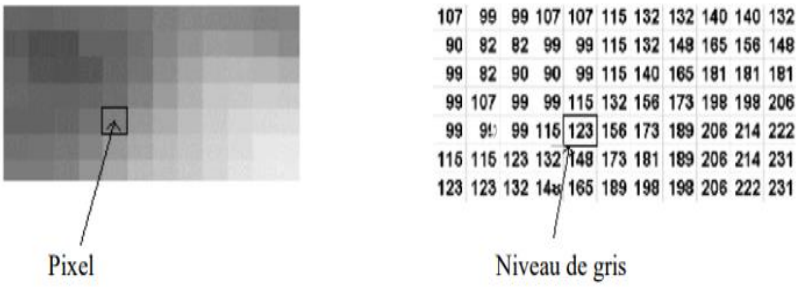


***Figure* I .7** *:* Une image en niveaux de gris, agrandissement d'une zone de l'image, et affichage des valeurs constituant la matrice image

3-3 Images en couleurs (Polychromes)

La représentation des couleurs s'effectue de la même manière que les images monochromes avec cependant quelques particularités. Pour cela il faut tout d'abord choisir un modèle de représentation, le modèle le plus utilisés de nos jours est le suivant :

- **Les systèmes émettant de la lumière (écrans d'ordinateurs,)** sont basés sur le principe de la synthèse additive : les couleurs sont composées d'un mélange de rouge, vert et bleu (modèle R.G.B pour Red, Green et Blue).
- **La représentation en couleurs réelles :** consiste à utiliser 24 bits pour chaque pointde l'image. 8 bits sont employés pour décrire la composante rouge (R), 8 bits pour le vert (G) et 8 bits pour le bleu (B). Il est ainsi possible de représenter environ 16,7 millions de couleurs différentes simultanément. Cela est cependant théorique, car aucun écran n'est capable d'afficher 16 millions de points. Dans la plus haute résolution (1600 x 1200), l'écran n'affiche que 1 920 000 points. Par ailleurs, l'œil humain n'est pas capable de distinguer autant de couleurs.

4- Classification des images :

La classification des images peut être définie comme étant la répartition systématique des images selon des classes connues au préalable. Donc classer une image c'est le faite de lui faire correspondre une classe, marquant ainsi sa parenté avec d'autres images.[5]

4-1 Classification des images et machine learning

La classification d'images est un problème fondamental en machine learning, qui a de nombreuses applications concrètes **[6]** Lorsque l'apprentissage automatique et la classification des images sont intégrés, les ordinateurs deviennent capables d'exécuter des tâches visuelles qui, jusqu'à récemment, ne pouvaient être effectuées que par des humains. L'ensemble, ces technologies offrent le potentiel 14 de percées dans l'automatisation, offrant de nouvelles opportunités numériques pour les entreprises dans divers domaines **[7]**

Pour une machine, une image est un tableau de nombres indiquant la luminosité (ou la couleur) de chaque pixel, et un signal sonore une suite de nombres indiquant la pression de l'air à chaque instant**.[6]**

Un système entraînable peut être vu comme une boite noire avec une entrée, par exemple une image, un son, ou un texte, et une sortie qui peut représenter la catégorie de l'objet dans l'image, le mot prononcé, ou le sujet dont parle le texte. On parle alors de systèmes de classification ou de reconnaissance des formes **[6]**

Dans sa forme la plus utilisée, l'apprentissage machine est supervisé: on montre en entrée de la machine une photo d'un objet, par exemple une voiture, et on lui donne la sortie désirée pour une voiture. Puis on lui montre la photo d'un chien avec la sortie désirée pour un chien. Après chaque exemple, la machine ajuste ses paramètres internes de manière à rapprocher sa sortie de la sortie désirée. Après avoir montré à la machine des milliers ou des millions d'exemples étiquetés avec leur catégorie, la machine devient capable de classifier correctement la plupart d'entre eux. Mais ce qui est plus intéressant, c'est qu'elle peut aussi classifier correctement des images de voiture ou de chien qu'elle n'a jamais vues durant la phase l'apprentissage. C'est ce qu'on appelle la capacité de généralisation**.[7]**

Jusqu'à récemment, les systèmes de reconnaissance des images classiques étaient composés de deux blocs: un extracteur de caractéristiques (feature extractor en anglais), suivi d'un classifieur entraînable simple. L'extracteur de caractéristiques est programmé «à la main», et transforme le tableau de nombres représentant l'image en une série de nombres, un vecteur de caractéristiques, dont chacun indique la présence ou l'absence d'un motif simple dans l'image. Ce vecteur est envoyé au classifieur, dont un type commun est le classifieur linéaire. Ce dernier calcule une somme pondérée des caractéristiques: chaque nombre est multiplié par un poids (positif ou négatif) avant d'être sommé. Si la somme est supérieure à un seuil, la classe est reconnue. Les poids forment une sorte de «prototype» pour la classe à laquelle le vecteur de caractéristiques est comparé. Les poids sont différents pour les classifieurs de chaque catégorie, et ce sont eux qui sont modifiés lors de l'apprentissage. Les premières méthodes de classification linéaire entraînable datent de la fin des années cinquante et sont toujours largement utilisées aujourd'hui. Elles prennent les doux noms de perceptron ou régression logistique. **[6]**

4-2 Classification des images et les réseaux de neurones

Le problème de l'approche classique de la reconnaissance des images est qu'un bon extracteur de caractéristiques est très difficile à construire, et qu'il doit être repensé pour chaque nouvelle application.

C'est là qu'intervient l'apprentissage profond ou deep learning en anglais. C'est une classe de méthodes dont les principes sont connus depuis la fin des années 1980, mais dont l'utilisation ne s'est vraiment généralisée que depuis 2012, environ.

L'idée est très simple: le système entraînable est constitué d'une série de modules, chacun représentant une étape de traitement. Chaque module est entraînable, comportant des paramètres ajustables similaires aux poids des classifieurs linéaires. Le système est entraîné de bout en bout: à chaque exemple, tous les paramètres de tous les modules sont ajustés de manière à rapprocher la sortie produite par le système de la sortie désirée. Le qualificatif profond vient de l'arrangement de ces modules en couches successives

Pour pouvoir entraîner le système de cette manière, il faut savoir dans quelle direction et de combien ajuster chaque paramètre de chaque module. Pour cela il faut calculer un gradient, c'est-à-dire pour chaque paramètre ajustable, la quantité par laquelle l'erreur en sortie augmentera ou diminuera lorsqu'on modifiera le paramètre d'une quantité donnée. Le calcul de ce gradient se fait par la méthode de rétropropagation, pratiquée depuis le milieu des années 1980.

Dans sa réalisation la plus commune, une architecture profonde peut être vue comme un réseau multicouche d'éléments simples, similaires aux classifieurs linéaires, interconnectés par des poids entraînables. C'est ce qu'on appelle un réseau neuronal multicouche.

Pourquoi neuronal? Un modèle extrêmement simplifié des neurones du cerveau les voit comme calculant une somme pondérée et activant leur sortie lorsque celle-ci dépasse un seuil. L'apprentissage modifie les efficacités des synapses, les poids des connexions entre neurones. Un réseau neuronal n'est pas un modèle précis des circuits du cerveau, mais est plutôt vu comme un modèle conceptuel ou fonctionnel. Le réseau neuronal est inspiré du cerveau un peu comme l'avion est inspiré de l'oiseau.

Ce qui fait l'avantage des architectures profondes, c'est leur capacité d'apprendre à représenter le monde de manière hiérarchique. Comme toutes les couches sont entraînables, nul besoin de construire un extracteur de caractéristiques à la main. L'entraînement s'en chargera. De plus, les premières couches extrairont des caractéristiques simples (présence de contours) que les couches suivantes combineront pour former des concepts de plus en plus complexes et abstraits: assemblages de contours en motifs, de motifs en parties d'objets, de parties d'objets en objets, etc **[5]**

4-3 Classification des images et le modèle de détection yolo

L'approche implique un réseau de neurones qui met bout-à-bout une multitude de neurones et qui prend une photo en entrée et prédit directement les cadres de délimitation, plus communément appelés « bounding box », et les étiquettes de classe pour chaque cadre de délimitation.

La classification d'images consiste à prédire la classe d'un objet dans une image.

- Entrée : une image avec un seul objet.
- Sortie : une étiquette de classe.

La localisation d'objets consiste à identifier l'emplacement d'un ou plusieurs objets dans une image et à dessiner un cadre de délimitation autour de leur étendue.

- Entrée : une image avec un ou plusieurs objets.
- Sortie : un ou plusieurs cadres de délimitation.

La détection d'objets combine ces deux tâches et dessine un cadre de délimitation autour de chaque objet dans l'image et leur attribue une classe.

- Entrée : une image avec un ou plusieurs objets.
- Sortie : un ou plusieurs cadres de délimitation et une étiquette de classe pour chaque cadre de délimitation.[8]

Conclusion

Nous avons abordé dans ce chapitre les notions générales de l'image et ses caractéristiques et en particulier l'image numérique, et Nous avons consacré ce chapitre à la présentation des notions de la classification ainsi que leurs intérêts dans le domaine d'imagerie

s

Chapitre II :

Réseaux de neurones convolutifs et apprentissage profond

1-Reseaux de neurones

1-1 Introduction

Un réseau de neurones artificiel est un système de calcul qui tente d'imiter ou du moins s'inspire de connections neuronales de notre système nerveux. Les réseaux de neurones artificiels sont également appelés réseaux de neurones**.[9]**

1-2 Système de réseau de neurones artificiel

Pour qu'un système soit considéré comme un NN (neural network), il doit contenir une structure de graphe orientée étiquetée où chaque nœud du graphe effectue un calcul simple. D'après la théorie des graphes, un graphe orienté se compose d'un ensemble de nœuds (c'est-à-dire de sommets) et d'un ensemble de connexions (c'est-à-dire d'arêtes) qui relient des paires de nœuds. Dans **(figure II.1).** Chaque nœud effectue un calcul simple. Chaque connexion transporte alors un signal (c'est-à-dire la sortie du calcul) d'un nœud à un autre, étiqueté par un poids indiquant la mesure dans laquelle le signal est amplifié ou diminué. Certaines connexions ont des poids positifs importants qui amplifient le signal, indiquant que le signal est très important lors de la classification. D'autres ont des poids négatifs, diminuant la force du signal, spécifiant ainsi que la sortie du nœud est moins importante dans la classification finale. Un système est un réseau neuronal artificiel s'il consiste en une structure de graphe **(comme dans la figure II.1**) avec des poids de connexion modifiables à l'aide d'un algorithme d'apprentissage**. [10]**

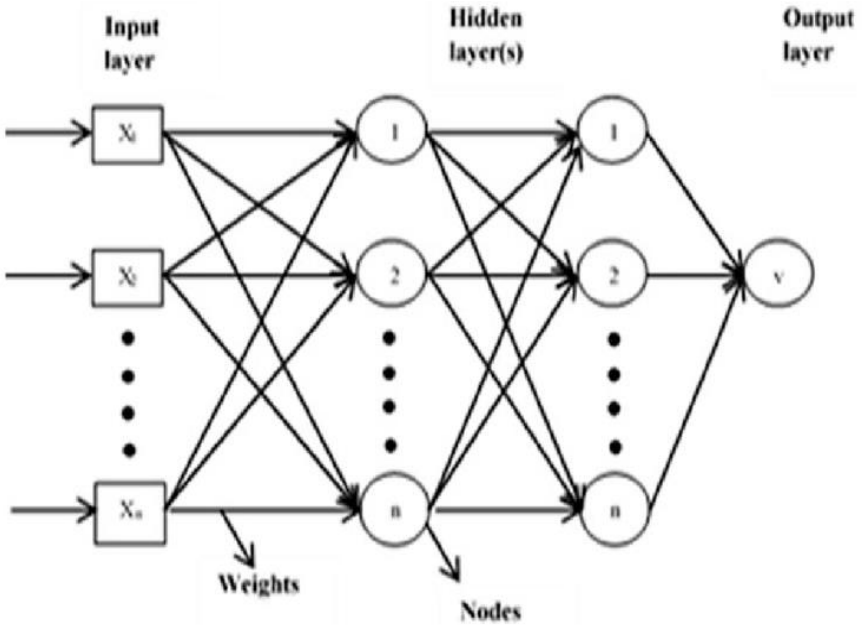


Figure II.1 : Architecture de réseau neuronal simple

1-3 Relation avec la biologie

Nos cerveaux sont composés d'environ 10 milliards de neurones, chacun étant connecté à environ 10 000 autres neurones. Le corps cellulaire du neurone est appelé le soma, où les entrées (dendrites) et les sorties (axones) connectent le soma à d'autres soma (**Figure II.2).** Chaque neurone reçoit des entrées électrochimiques d'autres neurones au niveau de leurs dendrites. Si ces entrées électriques sont suffisamment puissantes pour activer le neurone, alors le neurone activé transmet le signal le long de son axone, le transmettant aux dendrites d'autres neurones. Ces neurones attachés peuvent également se déclencher, poursuivant ainsi le processus de transmission du message. La clé à retenir ici est qu'un neurone ne se déclenchera que si le signal total reçu au niveau du soma dépasse un seuil donné. Cependant, gardez à l'esprit que les ANN sont simplement inspirés par ce que nous savons sur le cerveau et son fonctionnement **[10]**

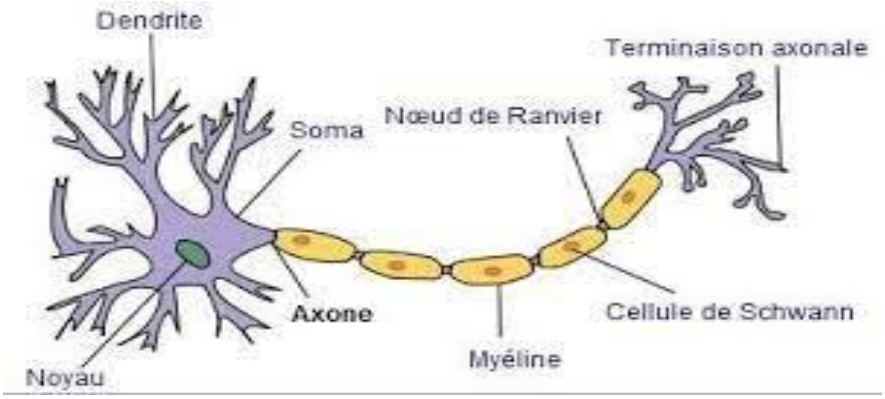


Figure II.2 : Structure d'un neurone biologique

1-4 Réseau de neurones artificiels

Les unités de calcul qui font partie d'un réseau de neurones sont appelées artificielles neurones ou pour faire court juste des neurones. Le schéma fonctionnel de **la Figure II.3** montre un modèle de neurone artificiel. Le modèle neuronal est composé des éléments suivants blocs de construction:

- Un ensemble de synapses ou liens de connexion, chacun caractérisé par un poids ou la force. Un signal xj à l'entrée de la synapse j connectée au neurone, k est multiplié par le poids synaptique wk.
- Un additionneur pour additionner les signaux d'entrée, pondérés par les forces synaptiques du neurone. Les opérations constituent ici un combinateur linéaire
- Une fonction d'activation, $\phi(\cdot)$, pour limiter l'amplitude de la sortie d'un neurone.

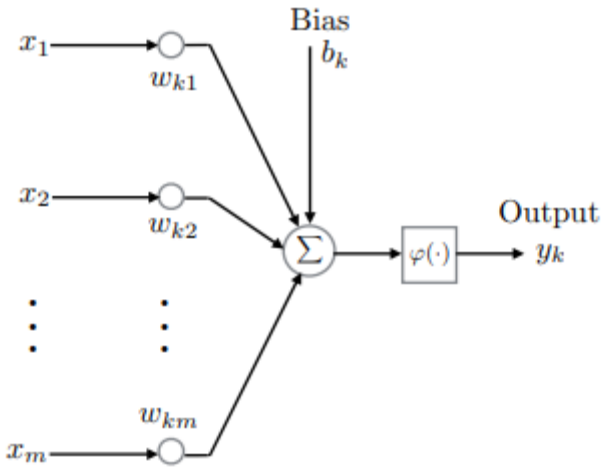


Figure II.3 : Réseau de neurone artificiel simple

Dans le modèle neuronal présenté à la **figure II.3**, nous pouvons voir un biais, b, appliqué au réseau. L'effet du biais est de diminuer ou d'augmenter l'apport net de la fonction d'activation selon qu'elle est négative ou positive.

Plus d'informations sur la fonction d'activation sont présentées dans la section suivante. Une représentation mathématique du réseau de neurones de la **figure II.3** est donnée par l'équation :

$$U_k=\sum_{j=1}^{m} wkj\ xj\ , \qquad \textbf{(II.1)}$$

$$y_k = \varphi(u_k + b_k),$$

Où x_1, x_2, . . . , xm sont les signaux d'entrée et w_1, w_2, . . . , w_m sont les poids synaptiques respectifs du neurone k. La sortie du neurone est y_k et u_k représente la sortie du combinateur linéaire due aux signaux d'entrée. Le biais est notée b_k et la fonction d'activation par ϕ. Le biais b_k est une affine transformation à la sortie u_k du combineur linéaire. Nous définissons maintenant le champ local induit ou potentiel d'activation comme :

$$v_k = u_k + b_k. \qquad \textbf{(II.2)}$$

Mettre tous les composants du réseau de neurones sous une forme plus compacte on aboutit aux équations suivantes :

$$v_k=\sum_{j=0}^{m} wkj\ xj\ , \qquad \textbf{(II.3)}$$

Et

$$y_k = \varphi(v_k) \qquad \textbf{(II.4)}$$

Où nous avons maintenant ajouté une nouvelle synapse avec entrée x0 = 1 et poids wk0 = bk tenant compte du biais. Dans **la figure II.4**, nous avons ajouté le biais comme un signal d'entrée fixe, x0 = 1, de poids w = b = 1 montrant comment **(II.3)** peut être interprété **[11]**

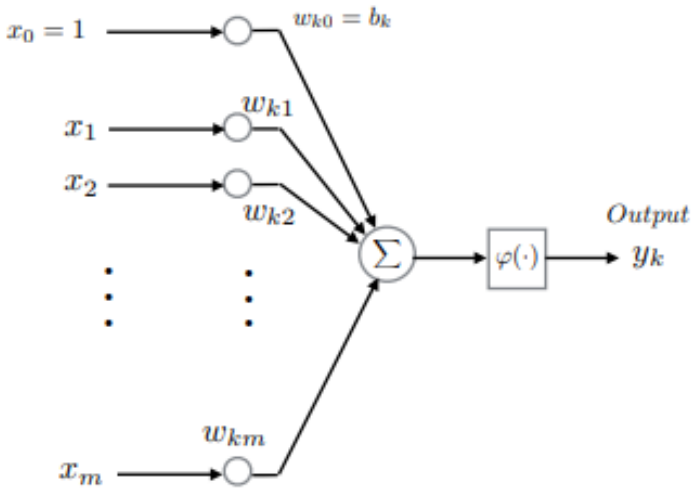


Figure II.4 : Réseau de neurones avec biais en entrée, x0 = +1, et poids wk0 = bk.

1-5 Fonctions d'activations :

les fonctions d'activations sont des équations mathématiques utilisées pour donner un sens à la classification faite par un modèle, ces fonctions peuvent être activées ou non en dépendances de l'importance de neurone qui se caractérise par sa valeur calculée par la fonction de pré-activation, dans les deux cas (fonction activé ou non) la fonction d'activation donne en sortie le résultat final de neurone. Ce résultat se diffère par rapport à la fonction utilisée et la fonction de cette dernière.

Les fonctions d'activations représentent un point clé dans le fonctionnement de neurone dans toutes ces phrases, le choix de bonne fonction d'activation assure un bon résultat de prédiction, une correction rapide des paramètres d'un modèle et d'un entrainement plus rapide. **[12]**

1-5-1 Fonction "sigmoïde" :

La fonction d'activation la plus courante utilisée dans l'histoire de la littérature NN est la fonction sigmoïde (Figure **II.5**), décrite par l'équation suivante :[10]

$$\varphi(z) = \frac{1}{1+e^{(-av)}} \quad [12] \qquad \textbf{(II.5)}$$

Avec z est le résultat de la pré-activation

Cette fonction représentée graphiquement par :

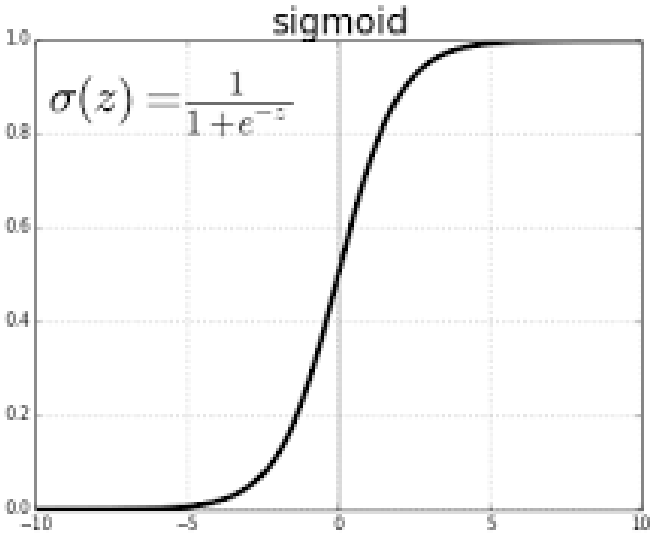


Figure II.5 : La représentation graphique de sigmoïde

Dans le cas de la rétro-propagation la dérivée de la fonction de sigmoïde sera utilisée cette dérivée est exprimées mathématiquement par :

$$F'(z) = F(z) \times (1 - F(z)) \text{ [13]} \qquad \text{(II.6)}$$

1-5-2 Fonction "RELU" :

Une unité linéaire rectifiée est plus intéressante modèle de vrais neurones, (Nair et Hinton, 2010). Il est construit par faire un grand nombre de copies à partir du sigmoïde. Ceci est fait sous l'hypothèse que toutes les copies ont les mêmes poids appris et Les biaises [3]

Cette fonction est représentée mathématiquement comme suit :

$$F(z) = \max(0, x) \qquad \text{(II.7)}$$

$$F'(z) = \begin{cases} 0 \; si \; x < 0 \\ 1 \; si \; x \geq 0 \end{cases} \text{ [13]} \qquad \text{(II.8)}$$

Et représentée graphiquement comme suit :

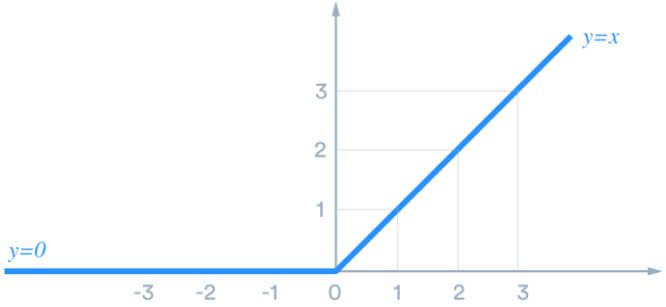


Figure II.6 : La representtation graphique de Relu

1-5-3 La tangente hyperbolique

La fonction hyperbolique ou tanh est souvent utilisée dans les réseaux de neurones comme fonction d'activation. Il fournit une sortie entre -1 et +1. Il s'agit d'une extension de la sigmoïde logistique; la différence est que la sortie s'étend ici entre -1 et +1. A également été largement utilisée comme fonction d'activation jusqu'à la fin des années 1990 **figure II.7**

L'équation de tanh est la suivante :

$$(z) = \tanh(z) = (e^{z} - e^{-z})/(e^{z} + e^{-z}) \qquad \textbf{(II.9)}$$

Et représentée graphiquement comme suit :

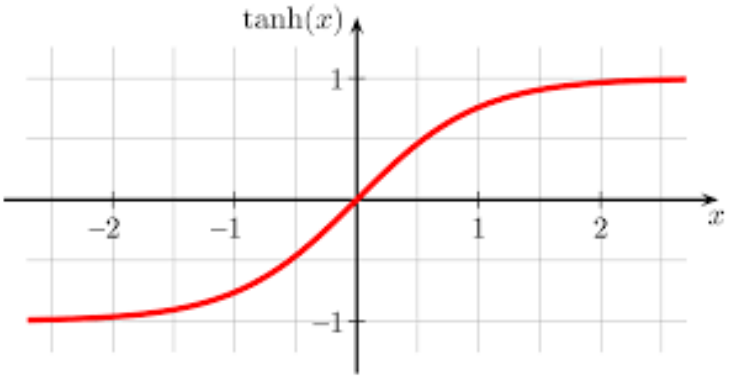


Figure II.7 : La représentation graphique de tanh

1-5-4 Fonction Softmax

La fonction Softmax permet-elle de transformer un vecteur réel **e**n vecteur **de** probabilité. On l'utilise souvent dans la couche finale d'un modèle de classification**,** notamment pour les problèmes multi classe. Dans la fonction Softmax, chaque vecteur est traité indépendamment**. (Figure II.8) [14]**

L'équation de softmax est la suivante :

$$F(z_i) = \frac{e^{zi}}{\sum_{=0}^{k} e^{zj}} \qquad [12] \qquad \textbf{(II.10)}$$

Et représentée graphiquement comme suit :

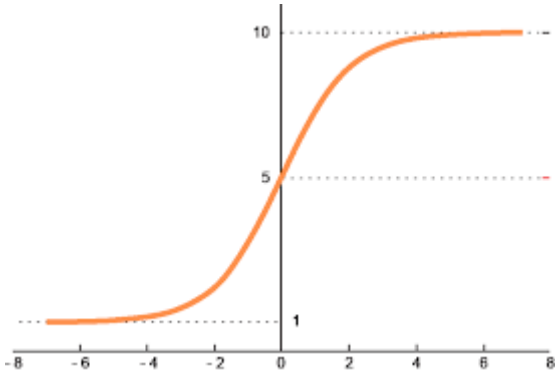


Figure II.8 : La représentation graphique de la fonction softmax

1-6 Architecture de réseau de neurones

Il existe de très nombreuses architectures NN (neural network) différentes, l'architecture la plus courante étant le réseau à réaction, comme le montre la **(figure II.9).** Dans ce type d'architecture, une connexion entre nœuds n'est autorisée que depuis les nœuds de la couche (layer) i vers les nœuds de la couche (layer) i + 1. Aucune connexion arrière ou intercouche (inter-layer) n'est autorisée. Lorsque les réseaux à action directe incluent des connexions de rétroaction (connexions de sortie qui alimentent les entrées), ils sont appelés réseaux neuronaux récurrents.**[10]**

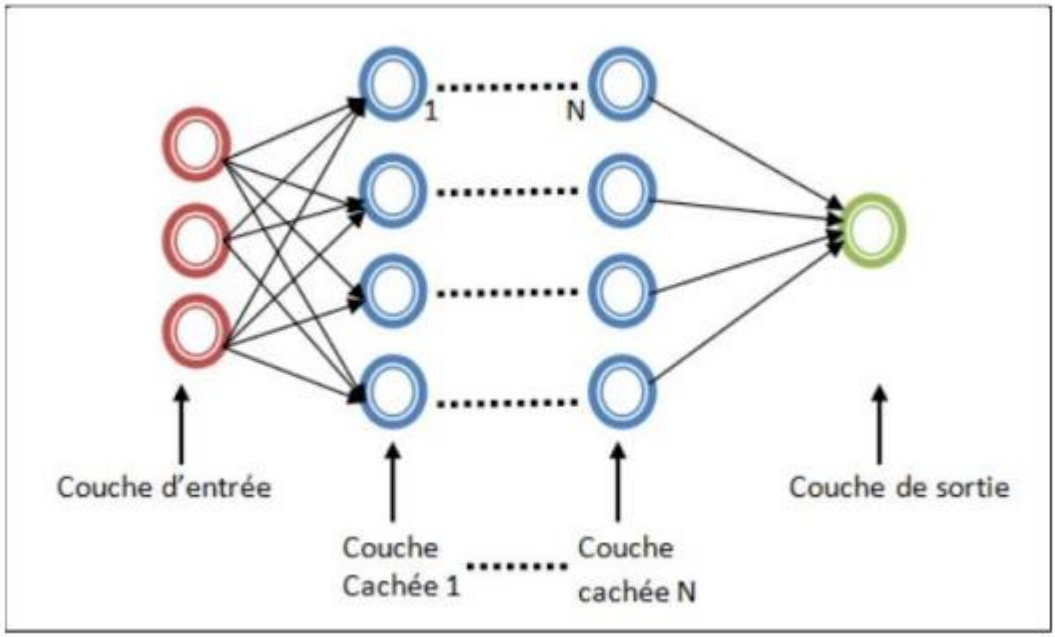


Figure II.9 : Architecture d'un ANN

1-7 Réseaux de neurones à réaction 'feed-forward'

Un réseau de neurones à rétroaction (feed-forward) permet aux informations de circuler uniquement dans le sens aller, des nœuds d'entrée, à travers les couches cachées et vers les nœuds de sortie. Il n'y a pas de cycles ou de boucles dans le réseau **(figure II.10).** Dans un réseau neuronal à réaction directe (feed-forward), les décisions sont basées sur l'entrée actuelle. Il ne mémorise pas les données passées et il n'ya pas de portée future. Les réseaux de neurones à réaction directe sont utilisés dans les problèmes généraux de régression et de classification **[15]**

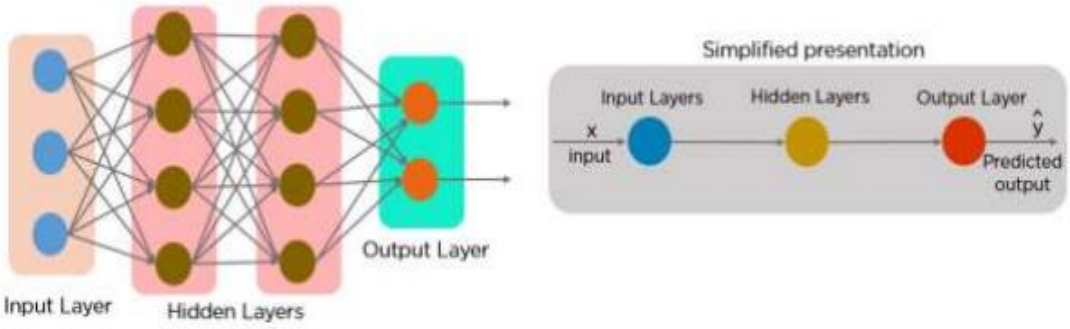


Figure II.10 : Réseaux de neurones à réaction

1-8 Réseaux de neurones récurent

Un réseau neuronal récurrent fonctionne sur le principe de sauvegarder la sortie d'une couche particulière et de la renvoyer à l'entrée afin de prédire la sortie de la couche. Les nœuds des différentes couches (layers) du réseau neuronal sont compressés pour former une seule couche de réseaux neuronaux récurrents. A, B et C sont les paramètres du réseau **(figure II.11)** Ici, « x » est la couche d'entrée, « h » est la couche cachée et « y » est la couche de sortie. A, B et C sont les paramètres réseau utilisés pour améliorer la sortie du modèle. A tout instant t donné, l'entrée courante est une combinaison d'entrée à x (t) et x (t-1). La sortie à un moment donné est récupérée sur le réseau pour améliorer la sortie. **(Figure II.12) [15]**

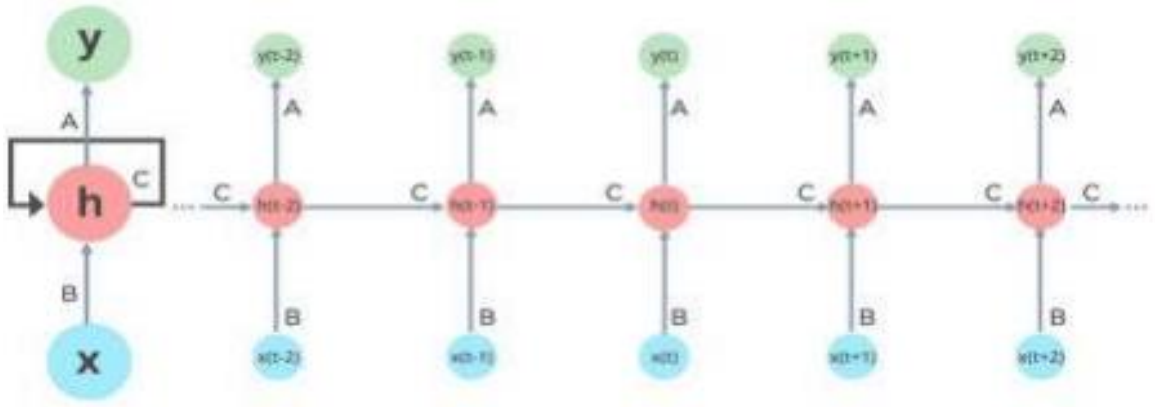


Figure II.11 : Réseau de neurones récurrent 'Recurrent Neural Network'

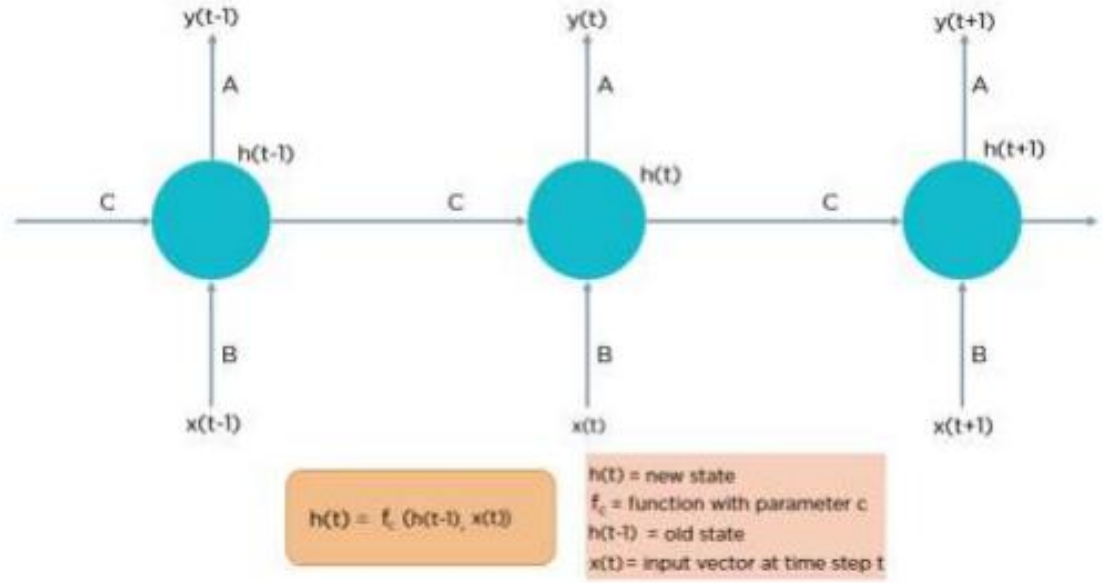


Figure II.12 : Explication de réseau de neurones récurrent 'Recurrent Neural Network'

1-9 Différence entre 'reccurent neural network' et 'feed-forward neural network'

Des réseaux de neurones récurrents 'reccurent neural network' ont été créés en raison de quelques problèmes dans le réseau de neurones à réaction (feed-forward neural network) :

- Impossible de gérer les données séquentielles
- Ne prend en compte que l'entrée actuelle
- Impossible de mémoriser les entrées précédentes

La solution à ces problèmes est le réseau neuronal récurrent (RNN). Un RNN peut gérer des données séquentielles, accepter les données d'entrée actuelles et les entrées reçues précédemment. Les RNN peuvent mémoriser les entrées précédentes grâce à leur mémoire interne **[15]**

2- Apprentissage automatique et Apprentissage profond

2-1 Introduction

L'apprentissage profond est un sous-domaine de l'apprentissage automatique, qui est à son tour un sous-domaine de l'intelligence artificielle 'IA' **(figure II.3)** L'objectif central de l'IA est de fournir un ensemble d'algorithmes et de techniques qui peuvent être utilisés pour résoudre des problèmes que les humains exécutent de manière intuitive et quasi automatique, mais qui sont par ailleurs très difficiles pour les ordinateurs. Un excellent exemple d'une telle classe de problèmes d'IA est l'interprétation et la compréhension du contenu d'une image - cette tâche est quelque chose qu'un humain peut faire avec peu ou pas d'effort, mais il s'est avéré extrêmement difficile à accomplir pour les machines. Alors que l'IA incarne un

ensemble important et diversifié de travaux liés au raisonnement automatique de la machine (inférence, planification, heuristique, etc.), le sous-champ d'apprentissage automatique a tendance à être spécifiquement intéressé par la reconnaissance des formes et l'apprentissage à partir des données.

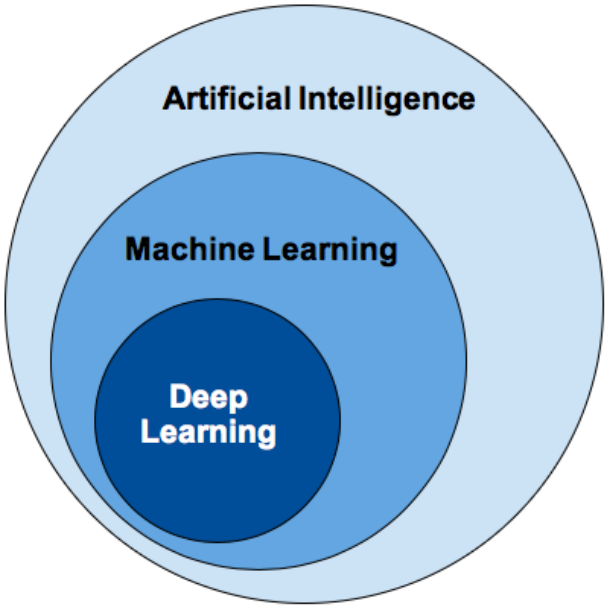


Figure II.13 : La relation entre l'IA, Machine Learning et Deep Learning

2-2 Apprentissage automatique 'Machine Learning'

Les algorithmes d'apprentissage automatique sont des algorithmes mathématiques qui permettent aux machines d'apprendre en imitant la façon dont les humains apprennent, bien que l'apprentissage automatique ne soit pas seulement des algorithmes, c'est aussi l'approche à partir de laquelle le problème est abordé. L'apprentissage automatique est essentiellement un moyen d'obtenir de l'intelligence artificielle **[9]**

2-3 Apprentissage profond 'Deep Learning'

Le Deep Learning ou apprentissage profond est un domaine de la machine Learning. Le Deep Learning se rapprocherait du cerveau humain, il va plus loin que la connexion entre les données et les algorithmes puisqu'il permet à la machine d'apprendre et de progresser grâce à son expérience. Le DL, qui fait appel à la fois aux connaissances en neurosciences, aux mathématiques et aux progrès technologiques, est aujourd'hui plébiscité comme une véritable révolution dans le domaine de l'intelligence artificielle, L'apprentissage profond est un apprentissage réalisé sur un réseau de neurones avec plusieurs couches cachées. Le principe du Deep Learning repose sur un apprentissage hiérarchique couche 26 par couche. Entre chaque couche interviennent des transformations non linéaires et chaque couche reçoit en entrée la sortie de la couche précédente Ces réseaux sont capables de catégoriser les informations des plus simples aux plus complexes. Pour un objet par exemple, la première

couche détecte des petits contours élémentaires, la second assemble ces contours en motifs puis les motifs en parties d'objets puis ces parties en objets.

2-4 Différence entre 'Machine Learning' et 'Deep Learning'

L'apprentissage automatique et l'apprentissage profond imitent la façon dont le cerveau humain apprend. Sa principale différence réside donc dans le type d'algorithmes utilisés dans chaque cas, bien que l'apprentissage profond soit plus similaire à l'apprentissage humain car il fonctionne avec des neurones. L'apprentissage automatique utilise généralement des arbres de décision : Relation entre 'AI, ML, DL'. 40 réseaux de neurones d'apprentissage en profondeur, qui sont plus évolués. De plus, les deux peuvent apprendre de manière supervisée ou non **[16]**

2-5 Réseau de neurones à convolution 'Convolution neural network CNN'

2-5-1 Qu'est-ce que le réseau neuronal convolutif (CNN ou ConvNet) ?

Un réseau neuronal convolutif (CNN/ConvNet) est une classe de réseaux neuronaux profonds, le plus souvent appliqués pour analyser l'imagerie visuelle. Maintenant, quand nous pensons à un réseau de neurones, nous pensons aux multiplications matricielles, mais ce n'est pas le cas avec ConvNet. Il utilise une technique spéciale appelée convolution. Or, en mathématiques, la convolution est une opération mathématique sur deux fonctions qui produit une troisième fonction qui exprime comment la forme de l'une est modifiée par l'autre**.[18]**

2-5-2 Architecture et fonctionnement d'un CNN

L'architecture de CNN est inspirée par l'organisation et la fonctionnalité du cortex visuel et conçue pour imiter le modèle de connectivité des neurones dans le cerveau humaine.

Les réseaux de neurones convolutifs sont à ce jour les modèles les plus performants pour classer des images. Ils comportent deux parties bien distinctes : la partie convolutive et la partie classification**.[19]**

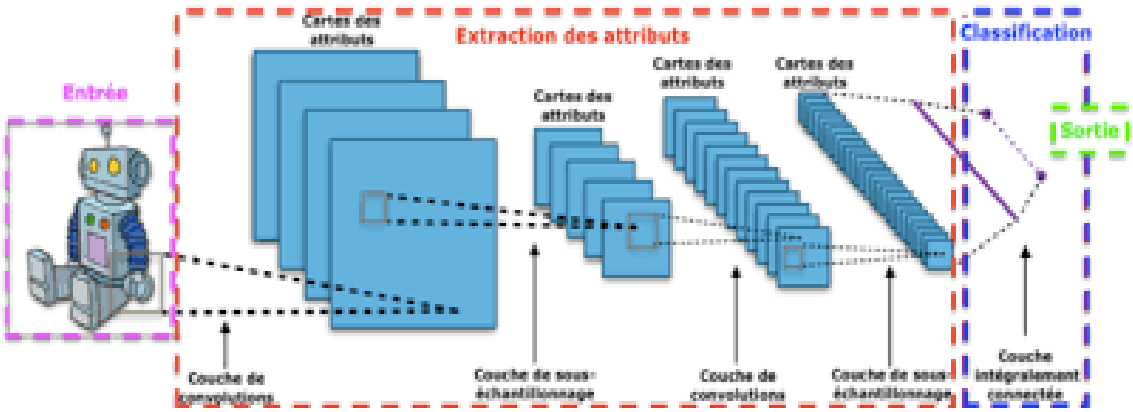


Figure II.14 : Architecture de base d'un réseau neuronal convolutif

2-5-2-1 Partie de convolution

Son rôle est d'extraire les caractéristiques des images. Une succession de filtres est appliquée à l'image d'entrée, pour produire de nouvelles images en sortie appelées cartes de convolutions. Certains filtres utilisent l'opération de maximum local (Pooling) pour redimensionner l'image en réduisant sa dimension, tout en préservant ses caractéristiques importantes. Finalement les cartes de convolutions sont chaînées dans un vecteur de caractéristiques appelé code CNN.

Dans la partie convolution on distingue 3 types de couches :

- Les couches de convolutions
- Les couches de correction ReLU
- Les couches de Pooling

1) Couche de convolution

La convolution est une opération mathématique simple généralement utilisée pour le traitement et la reconnaissance d'images. Sur une image, son effet s'assimile à un filtrage dont voici le fonctionnement **[20]**

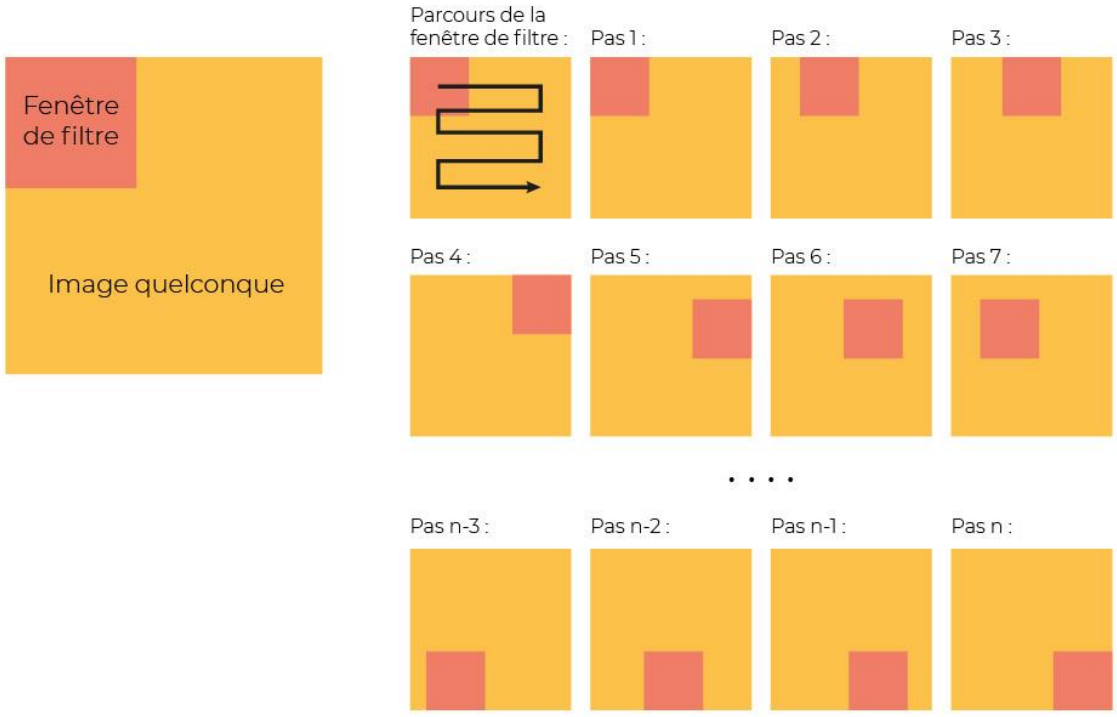


Figure II.15 : Parcours de la fenêtre de filtre sur l'image

1. Dans un premier temps, on définit la taille de la fenêtre de filtre située en haut à gauche.
2. La fenêtre de filtre, représentant la feature, se déplace progressivement de la gauche vers la droite d'un certain nombre de cases défini au préalable (le pas) jusqu'à arriver au bout de l'image.
3. À chaque portion d'image rencontrée, un calcul de convolution s'effectue permettant d'obtenir en sortie une carte d'activation ou featuremap qui indique où sont localisées les features (les caractéristiques) dans l'image : plus la featuremap est élevée, plus la portion de l'image balayée ressemble à la feature**.[21]**

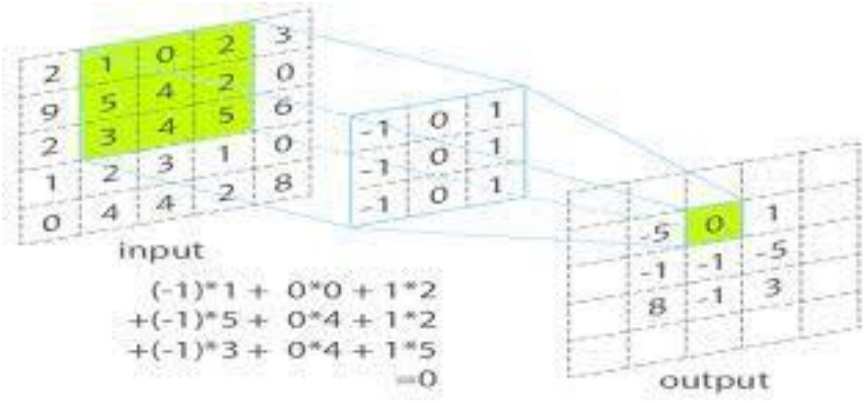


Figure II.16 : Le principe de convolution

La couche de convolution reçoit donc en entrée plusieurs images, et calcule la convolution de chacune d'entre elles avec chaque filtre. Les filtres correspondent exactement aux caractéristiques que l'on souhaite retrouver dans les images.

Pour chaque paire (image, filtre), une carte d'activation ou featuremap, lui est associée et qui sert à indiquer où se trouvent les features dans l'image : plus la valeur est élevée, plus l'endroit correspondant dans l'image ressemble à la feature.

2) Couche de correction (Relu)

Il est possible d'améliorer l'efficacité du traitement en insérant entre les couches de traitement une couche qui va appliquer une fonction mathématique (fonction d'activation) sur les signaux de sortie**. [22]**

On fait passer les cartes de convolutions à travers une couche d'activation non linéaire telle que RectifiedLinear Unit (ReLu), qui consiste à remplacer les nombres n.égatifs des images filtrées par des zéros**.[23]**

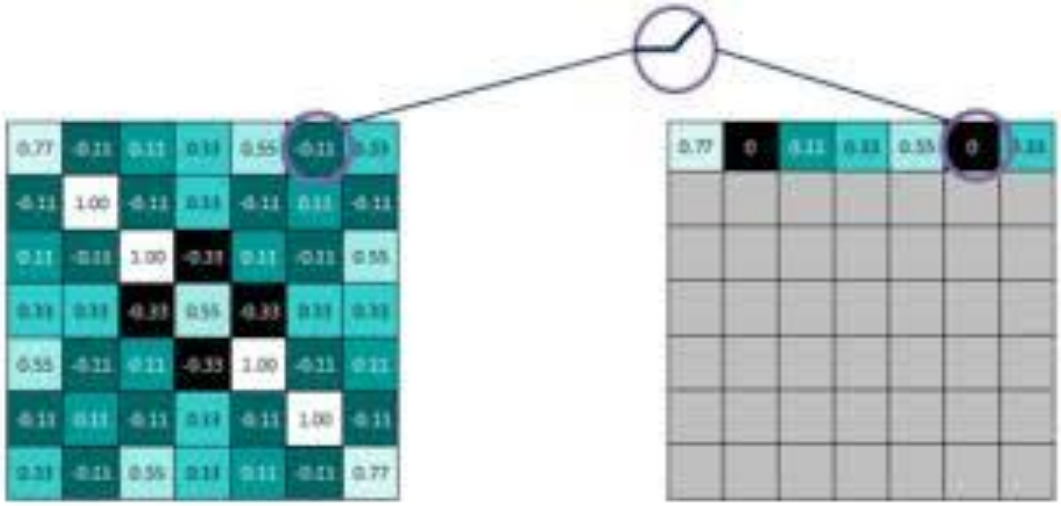


Figure II.17 : Fonctionnement de la fonction RELU

3) Couche Pooling (ou sous-échantillonnage)

Ce type de couche est souvent placé entre deux couches de convolution, elle consiste à un ré-échantillonnage des données; elle prend plusieurs featuremap en entrée et applique à chacune d'entre elle la fonction de Pooling, qui permet de réduire la taille des images en conservant leurs caractéristiques importantes.

Pour cela, l'image est découpée en plusieurs cellules régulières. Puis dans chaque cellule on garde la valeur maximale. Généralement, on utilise des cellules carrées, de petites tailles pour ne pas perdre beaucoup d'informations.

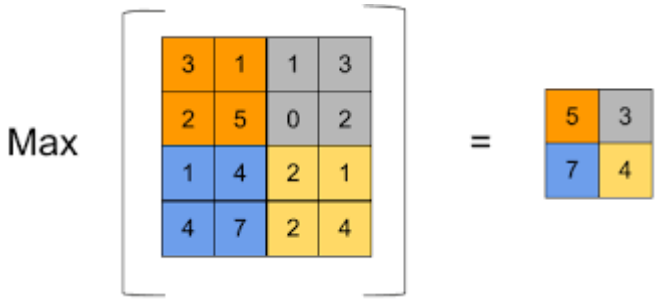


Figure II.18 : Le principe de Pooling avec un filtre 2*2 et un pas de 2

A la sortie de cette couche, on obtient en le même nombre de feature maps qu'en entrée, mais cette fois de taille réduite.

La couche de Pooling est utilisée pour minimiser le nombre de calculs et de paramètres dans le réseau. Afin de pouvoir contrôler l'overfitting (sur-apprentissage) et ainsi améliorer l'efficacité du réseau, une couche de Pooling est fréquemment insérée périodiquement entre deux couches de convolution successives.

A la fin du réseau, pour pouvoir indiquer la probabilité qu'une entité spécifique appartient à une certaine classe, un MLP (Multi Layer perceptron) souvent appelé Fully-Connected est ajouté.

2-5-2-2 Partie classification

Elle est constituée de couches entièrement connectées (perceptron multicouche).Elle prend la sortie de la partie convolutive qui est le code CNN comme entrée, afin de combiner ses caractéristiques pour classer l'image.

1) Couche Fully-connected (entièrement connecté)

Elle constitue toujours la dernière couche d'un réseau de neurones convolutif ou non. Elle reçoit un vecteur en entrée contenant les pixels aplatis de toutes les images filtrées, corrigées et réduites par le Pooling et produit un nouveau vecteur en sortie. Pour cela, elle applique une combinaison linéaire puis éventuellement une fonction d'activation aux valeurs reçues en entrée.

La dernière couche Fully-connected permet de classifier l'image en entrée du réseau : elle renvoie un vecteur de taille N, où N est le nombre de classes dans notre problème de classification d'images. Chaque élément du vecteur indique la probabilité pour l'image en entrée d'appartenir à une classe. **[23]**

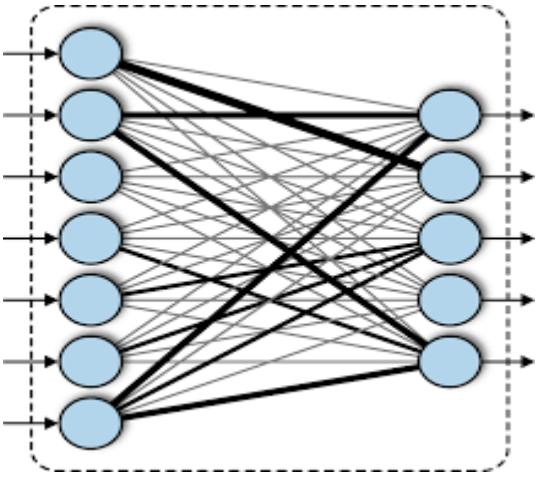


Figure II.19 : La couche Fully-connected

***Le pas 'Strides' :**

Le pas 'stride' est le nombre de pixels décalés sur la matrice d'entrée. Lorsque le pas est de 1, nous déplaçons les filtres sur 1 pixel à la fois. Lorsque le pas est de 2, nous déplaçons les filtres sur 2 pixels à la fois et ainsi de suite. La figure ci-dessous montre que la convolution fonctionnerait avec un pas de 2.

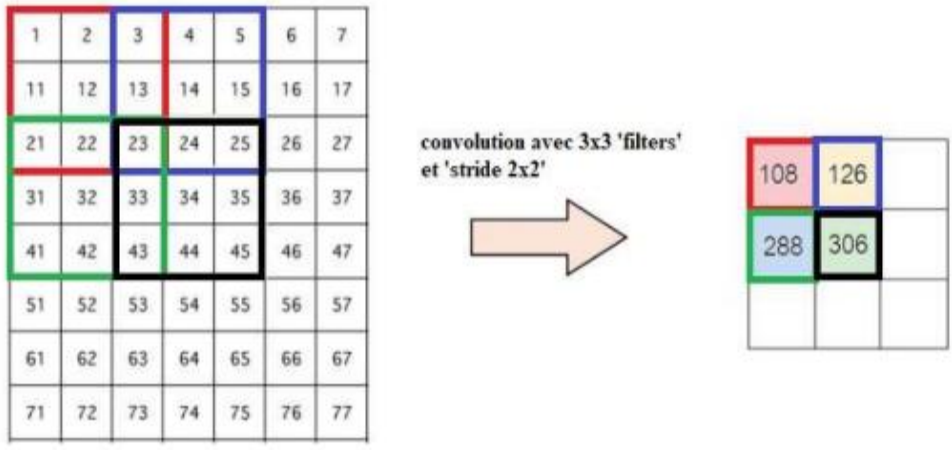


Figure II.20 : Illustration du pas de 2 pixels

2-6 Types d'apprentissage

Il existe quatre types d'apprentissage : l'apprentissage supervisé, l'apprentissage non supervisé, l'apprentissage semi-supervisé et l'apprentissage en transfert

2-6-1 Apprentissage supervisé

La classification d'images avec la méthode supervisée a comme objectif principal la définition d'un ensemble de règles, permettant de classer des objets dans des classes, à partir de certaines variables qui caractérisent ces objets. Pour cette méthode il faut posséder au sdépart un ensemble d'échantillons dit aussi d'apprentissage dont les classes sont connues. C'est-à-dire que chaque image possède une étiquette bien spécifique qui décrit sa classe d'appartenance. Cet échantillon est utilisé pour l'apprentissage des règles de classement comme première étape de classification, puis un deuxième échantillon indépendant, dit de validation ou de test est utilisé**.[24]**

2-6-2 Apprentissage non-supervisé

Ces méthodes procèdent d'une façon contraire que celles supervisées. C'est-à-dire ne nécessitent aucun apprentissage et aucune tâche préalable d'étiquetage manuel, autrement dit, les données sont non étiquetées. Alors, c'est au système d'extraire une règle d'appartenance de chaque image. Elles permettent de former à partir d'un nuage de points de n'importe quel espace un ensemble de groupes appelés Clusters. **.[24]**

2-6-3 Apprentissage par renforcement

Un algorithme d'apprentissage automatique par renforcement apprend de l'environnement s'il obtient de bons résultats, il reçoit une récompense, et l'objectif est de maximiser la récompense.[25]

L'algorithme reçoit un retour d'information concernant les récompenses et les punitions au fur et à mesure qu'il avance dans le problème. L'apprentissage par renforcement

permet de décider de la meilleure action suivante en fonction de son état actuel et en apprenant les comportements qui maximiseront la récompense [26]

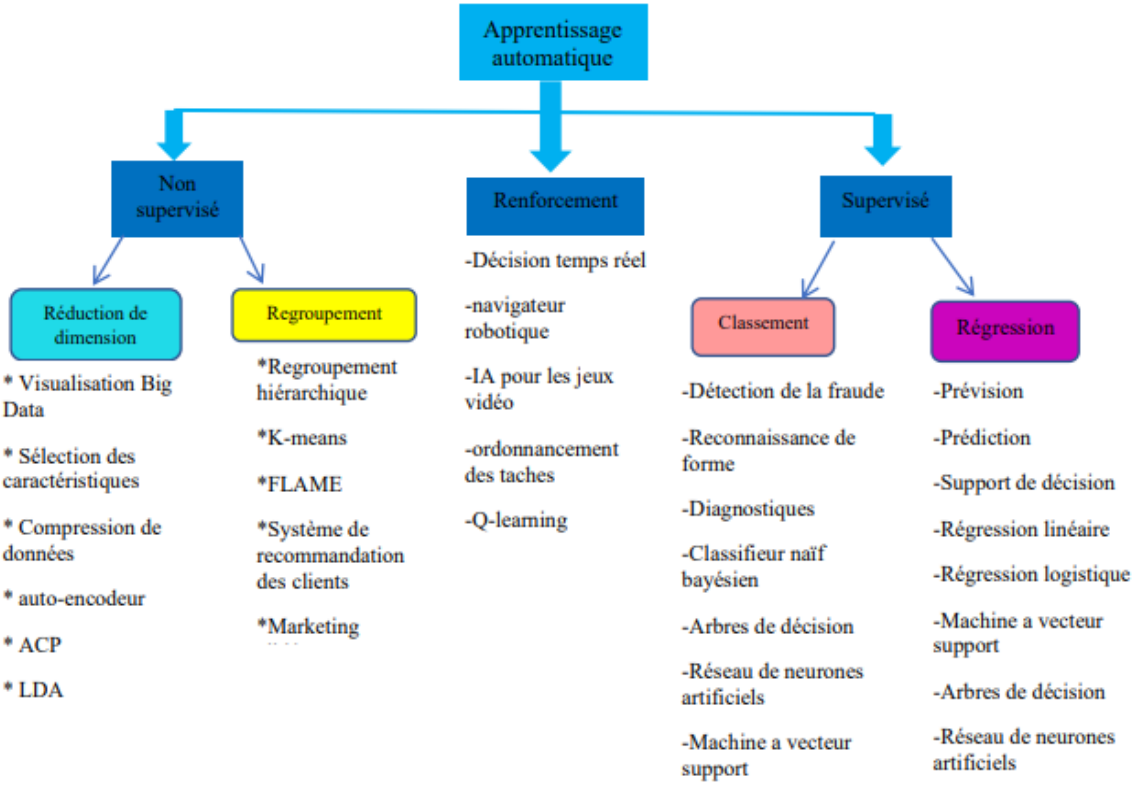


Figure II.21 : Les différents types de l'apprentissage automatique

2-7 Mesures de performances

Les évaluations de réseaux mise en œuvre nécessite le calcule d'un certain nombre de paramètres, on note :

a. Vrai positif 'True positive TP' : Le modèle prédit correctement la classe positive.
b. Vrai négatif 'True negative TN' : Le modèle prédit correctement la classe négative
c. Faux positif 'False positive FP' : Le modèle prédit incorrectement la classe positive.
d. Faux négatif 'False negative FN' : Le modèle prédit incorrectessment la classe négative.

2-7-1 Matrice de confusion

La matrice de confusion d'un système de classification binaire est :

	Classe réelle positive	**Classe réelle négative**
classe prédite positive	vrai positif (VP)	Faux positif (FP)
Classe prédite négative	Faux négatif (FN)	Vrai négatif (VN)

Tableau II.1 : matrice de confusion

2-7-2 Exactitude 'Accuracy'

Taux de bonnes prédictions.

$$Accuracy = \frac{vp+vn}{vp+vn+fp+fn} \quad [19] \quad \textbf{(II.11)}$$

2-7-3 Taux [27]

1. Taux vrai positif ou Sensitivité 'Recall'

Probabilité d'un cas positif si la prédiction est positive.

$$Taux\ vrai\ positif = \frac{vp}{vp+fp} = \frac{vp}{p} \quad \textbf{(II.12)}$$

2. Taux vrai négatif ou Spécificité

Probabilité d'une prédiction négative dans un cas négatif.

$$Taux\ vrai\ negatif = \frac{vn}{vn+fp} = \frac{vn}{n} \quad \textbf{(II.13)}$$

3. Taux faux positif

$$Taux\ faux\ positif = 1 - \frac{vn}{vn+fp} = 1 - \frac{vn}{n} = 1 - taux\ vrai\ negatif \quad \textbf{(II.14)}$$

4. Taux faux négatif

$$Taux\ faux\ negatif = 1 - \frac{vp}{vp+fp} = 1 - \frac{vp}{p} = 1 - taux\ vrai\ positif \quad \textbf{(II.15)}$$

2-7-4 F-mesure 'moyenne harmonique' [19]

C'est la moyenne harmonique entre le rappel et la précision.

$$FM = \frac{1}{\frac{1}{p}+\frac{1}{R}} = 2*\frac{p*R}{p+R} \quad \textbf{(II.16)}$$

Pour l'apprentissage de plusieurs classes, on généralise

***** rappel** = somme des rappels de chaque classe/nombre de classes

***** précision** = somme des précisions de chaque classe/nombre de classes

2-7-5 La courbe ROC (Received Operating Characteristic)

La courbe ROC consiste à un graphique qui représente l'ensemble de performances d'un modèle de classification pour tous les seuils de la classification. Cette courbe trace le taux de vrais positifs en fonction du taux de faux positifs.

Avec cet outil on peut avoir toutes les informations sur la performance du classifieur, et cela en utilisant l'aire sous la courbe. Plus elle se rapproche de 1, plus le classifieur est performant [16]

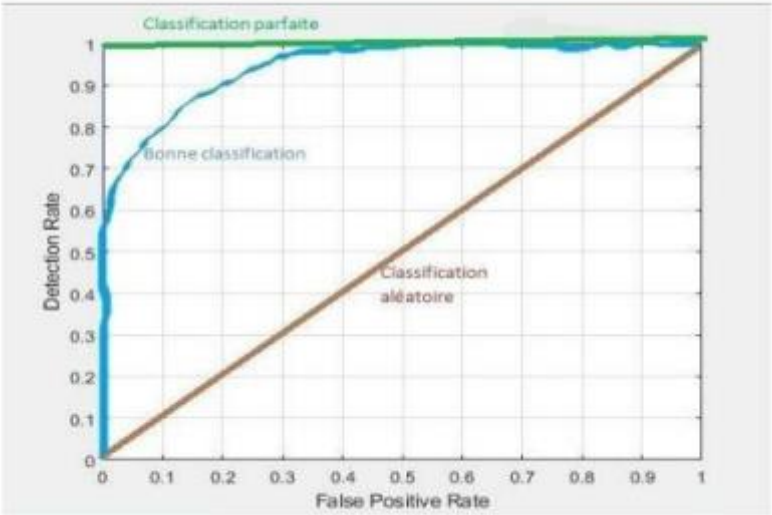


Figure II.22 : Exemple d'illustration de la courbe ROC

Il y a plusieurs domaine qui utiliser le CNN pour résoudre les problèmes et parmi ces domaine en a la détection d'objet et classification d'image, dans le schéma suivant nous allons voir les meilleurs model dans les deux demain précèdent :

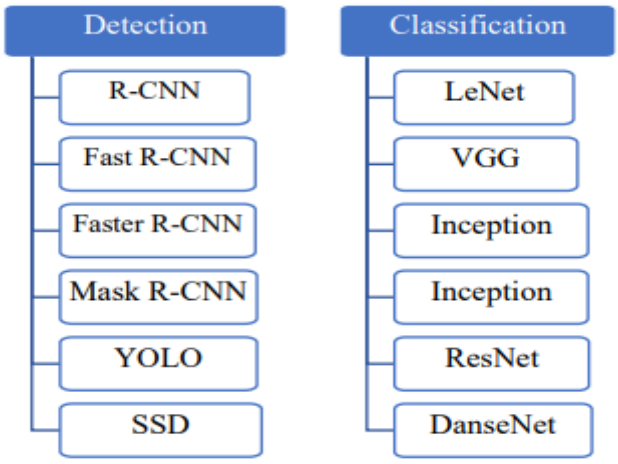


Figure II.23 : Schéma des modèles de détection et classification par CNN

Conclusion :

Nous avons consacré ce chapitre à la notion de réseau de neurone et sa relation avec la biologie, ainsi qu'à l'utilisation de la machine Learning et du Deep Learning dans ce domaine. Nous avons également abordé la notion des réseaux de neurones convolutifs, plus précisément, nous avons donné sa définition, son architecture et son principe de fonctionnement où nous avons cisté les différentes couches qui constituent ce réseau.

Chapitre III :
Modèles de détection

Introduction

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

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

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

1-Détection d'objet

1-1 Définition

La détection d'objet est une technique de vision par ordinateur dans laquelle un système logiciel peut détecter, localiser et tracer l'objet à partir d'une image ou d'une vidéo donnée. La technologie est capable de trouver des objets du monde réel comme la télévision, des fleurs, des vélos, des voitures, et peut également identifier des visages humains et des attributs instiller des vidéos et des images, et c'est l'un des attributs les plus particuliers de cette technologie. Il identifie la classe d'objets et leurs coordonnées spécifiques à l'emplacement dans une image ou une vidéo donnée**.[28]**

1-2 Méthodes

Les méthodes de détection d'objets relèvent généralement d'approches basées sur un réseau neuronal ou non neuronales. Pour les approches non neuronales, il devient nécessaire de définir d'abord les caractéristiques en utilisant l'une des méthodes ci-dessous, puis en utilisant une technique telle que la machine à vecteurs de support (SVM) pour effectuer la classification. D'autre part, les techniques neuronales 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) on mention quelques méthodes**:[29]**

- Approches non neuronales:
  - Cadre de détection d'objets Viola-Jones basé sur les fonctionnalités de Haar
  - Scale-invariant feature transform (SIFT)
  - Histogramme des caractéristiques des gradients orientés (HOG)

- Approches des réseaux de neurones
  - Propositions de région (R-CNN, Fast R-CNN, Faster R-CNN, cascade R-CNN)
  - Single Shot MultiBox Detector (SSD)
  - You Only Look Once (YOLO)
  - Single-Shot Refinement Neural Network for Object Detection (RefineDet)
  - Retina-Net
  - Reseaux convolutifs déformables

1-3 Modèles de détection par CNN

Les architectures CNN sont capables d'apprendre des caractéristiques plus complexes. Il existe deux types de modèles de détection d'objets par les architectures CNN. Le premier type de détection en deux coups est basé sur la proposition de région et comprend des modèles tels que RCNN, SPP-NET, FRCNN, Faster RCNN et le second type la détection à un coup est basé sur la régression et comprend MultiBox, AttentionNet, G-CNN, YOLO, SSD...etc. **[3]**

Il existe plusieurs model de détection d'objet, nous avons présenté dans schéma suivant, dans un ordre chronologique :

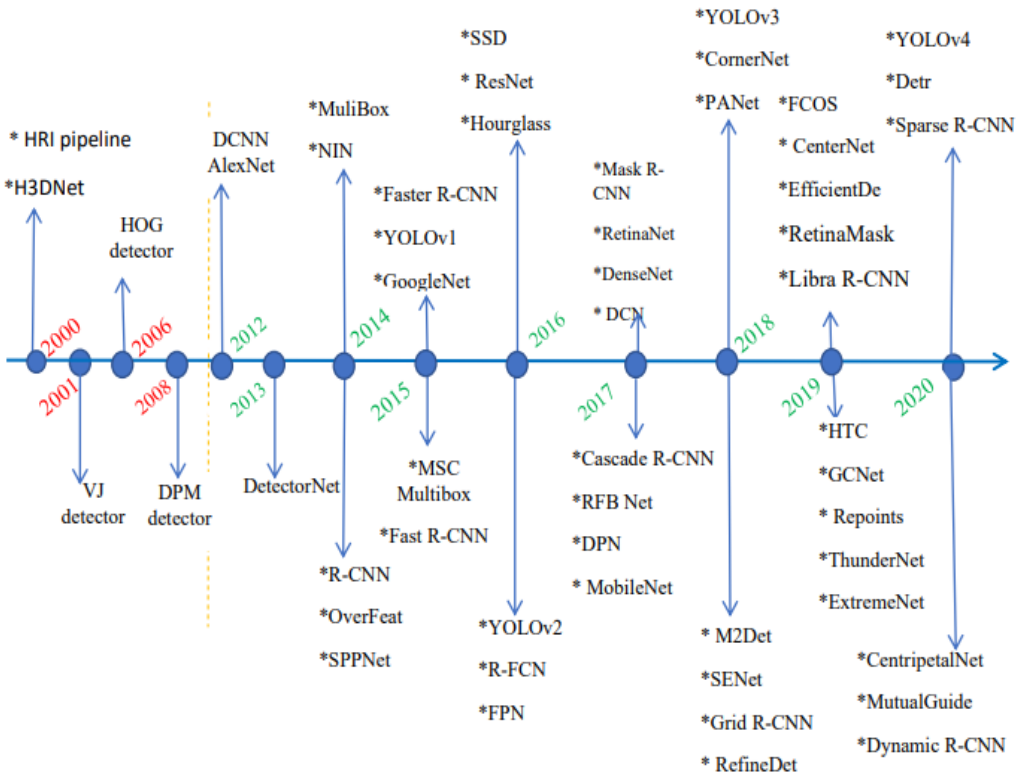


Figure III.1 : Étapes importantes de la détection et de la reconnaissance des objets [3]

1-3-1 R-CNN

Les modèles R-CNN sont une famille de réseaux de neurones convolutifs conçus pour la détection d'objets, développés par Ross Girshick, et al.

A partir d'une image donnée en entrée, le modèle va extraire de l'image les régions les plus susceptibles de contenir un objet. On parle de **zones d'intérêts.** Pour chacune de ces zones d'intérêts, un ensemble de boîtes englobantes (*bounding box*) va être généré. Ces boîtes sont classifiées et sélectionnées en fonction de leur probabilité à contenir l'objet. 2000 propositions de régions sont ainsi extraites. Ces régions sont ensuite reçues en entrée par le CNN. Le modèle peut alors détecter dans une image un (ou plusieurs) objet(s), pouvant appartenir à des classes différentes.

Les avantages sont donc de traiter l'image par morceau et non pas toute l'image comme pour un CNN simple et de pouvoir localiser plusieurs objets dans une image. C'est un traitement plus rapide et moins coûteux en puissance-machine**.[31]**

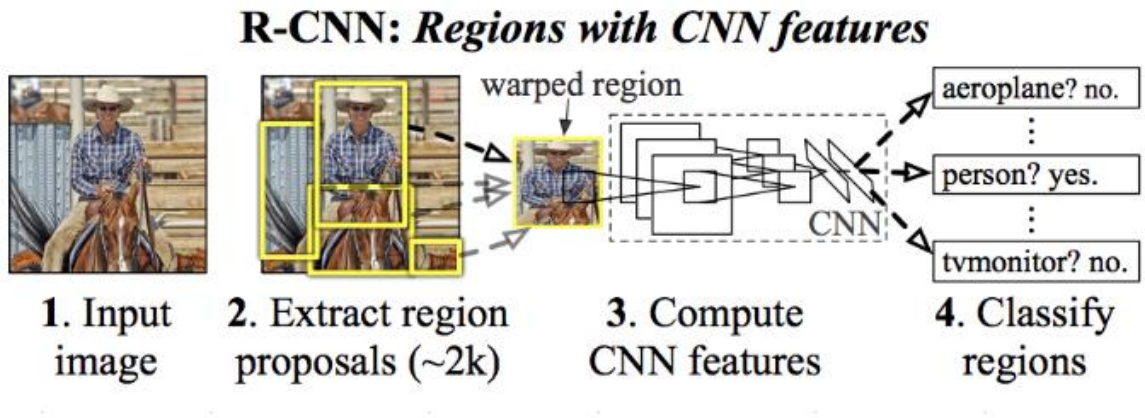


Figure III.2 : Architecture du modèle R-CNN [31]

1-3-2 Fast R-CNN

Avec un modèle R-CNN, chacune des zones d'intérêts proposées est reçue en entrée par le CNN, ainsi l'image de départ subit une convolution par zone proposée. En utilisant un modèle Fast R-CNN, c'est l'image de départ qui subit cette étape de convolution et c'est sur les *features map* générées que sont obtenues les propositions de régions susceptibles de contenir l'élément ciblé par le modèle. Cela constitue une amélioration en termes de temps d'exécution et de puissance requise**.[31]** elle est comparativement rapide à entraîner et à teste

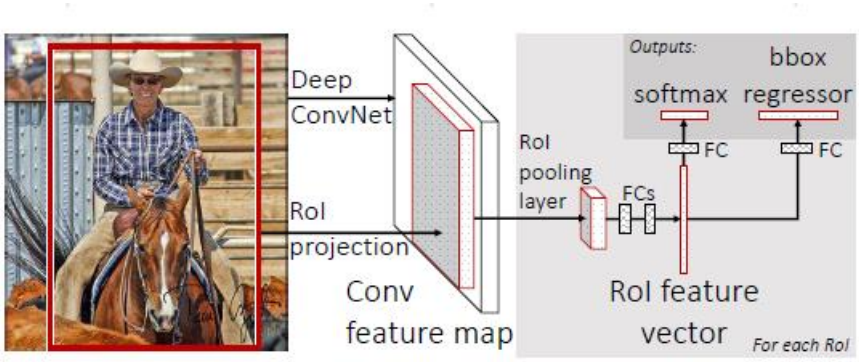


Figure III.3 : Architecture du modèle Fast R-CNN [31]

1-3-3 Faster R-CNN

Ce réseau proposer par Shaoqing Ren et al en 2016 ,il se décompose en deux modules principaux :

✓ Le premier module est un réseau convolutif profond qui créé la carte de caractéristiques convolutives qui utiliser par un module RPN (Réseau de Proposition de Région) qui prend cette carte (de n'importe quelle taille) et produit un ensemble de propositions d'objets rectangulaires, chacune avec un score de précision.

✓ Le second module est le détecteur Fast R-CNN qui utilise les régions proposées comme nous avons vu dans l'architecture fast R-CNN **[32]**

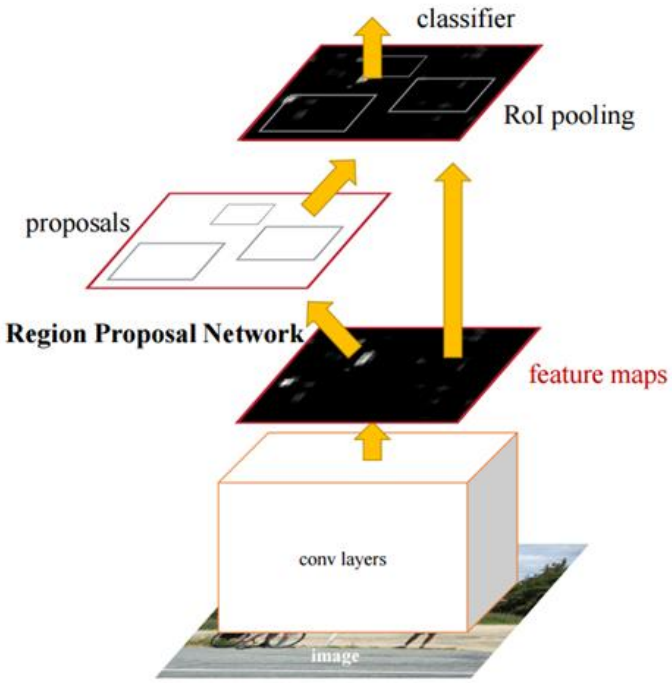


Figure III.4 : Architecture du modèle Faster R-CNN [32]

1-3-4 Mask R-CNN

Le modèle Mask R-CNN, comme les modèles précédents, permet de **détecter** des objets et de les **classifier**. Sa particularité est d'ajouter à cette tâche de détection la **segmentation d'instance**, c'est à dire que chaque pixel de l'image seront classés. Ainsi, cette double "compétence" représente un avantage par rapport à des modèles de détection, elle vient affiner le résultat proposé. De plus, contrairement à la segmentation sémantique, qui permet d'associer à chaque pixel un label, la segmentation d'instance associe un masque et un label à chaque objet, même si ces objets appartiennent à la même classe. Dans le cadre de notre étude où nous cherchons à extraire des toitures à partir d'images aériennes, chaque bâtiment est ainsi détecté indépendamment des autres et chacun a un masque qui lui est associé.

Mask R-CNN est une extension du modèle Faster R-CNN. Aux deux types de sorties générées par ce dernier, qui sont la classe de l'objet présent sur l'image et la boîte englobante associée, s'ajoute une troisième branche dont la sortie est le masque de l'objet.

Cet algorithme créé par Facebook est une combinaison de modèles déjà existants : *le réseau convolutif ResNet101* ; un réseau de proposition de régions (*regions proposal network, RPN*) ; un classificateur binaire de masque.[31]

Il est **rapide**, relativement **simple à implémenter** et **flexible** quant aux tâches pouvant lui être incombées (il peut notamment permettre de qualifier la "posture" d'une personne par détection de *keypoints*). Il a aussi été utilisé dans beaucoup de challenges (comme le challenge CrowdAI Mapping Challenge) et de projets relatifs à la détection d'objets à partir d'images aériennes/satellites, ce qui a motivé notre choix de ce modèle.

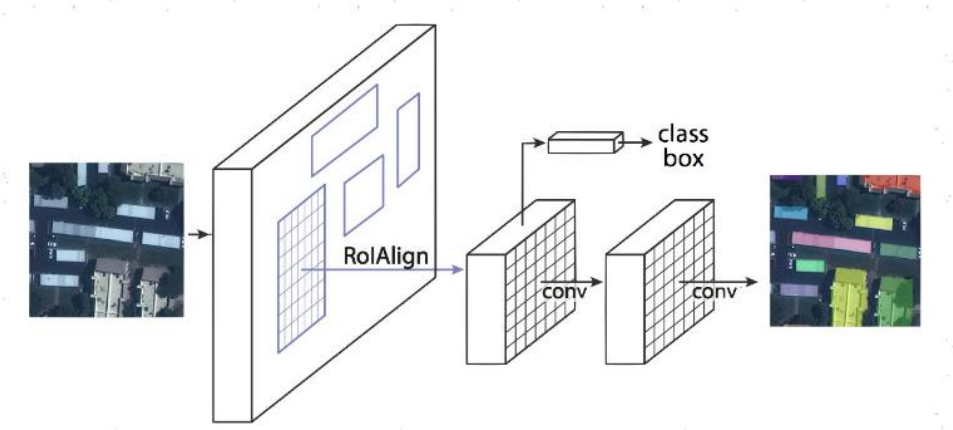


Figure III.5 : Architecture du modèle Mask R-CNN [31]

1-3-5 SSD: Single Shot MultiBox Detector

L'approche SSD est basée sur un réseau convolutif à anticipation qui produit une collection de taille fixe de boîtes englobantes et des 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). Nous ajoutons ensuite une structure auxiliaire au réseau pour produire des détections avec les caractéristiques clés suivantes : **[33]**

- Cartes de caractéristiques multi-échelles pour la détection
- Prédicteurs convolutifs pour la détection
- Boîtes et aspect par défaut

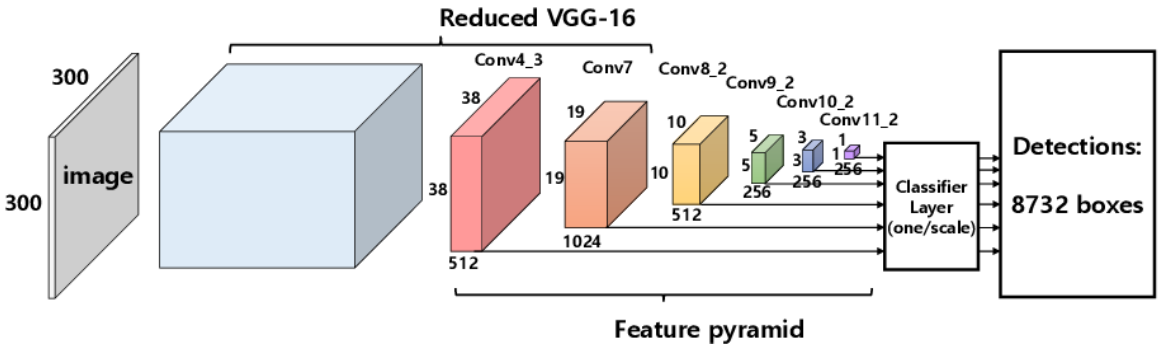


Figure III.6 : Architecture du modèle SSD [33]

1-3-6 YOLO

YOLO, acronyme de 'You only look once', est un algorithme de détection d'objet qui divise les images en un système de grille. Chaque cellule de la grille est responsable de la détection des objets en elle-même.

YOLO est l'un des algorithmes de détection d'objets les plus connus en raison de sa rapidité et de sa précision**.[34]**

Le réseau utilise les caractéristiques de l'image entière pour prédire chaque boîte englobante. Il prédit également toutes les boîtes englobantes de toutes les classes d'une image simultanément. Cela signifie que ce réseau raisonne globalement sur l'ensemble de l'image et sur tous les objets qu'elle contient. La conception YOLO permet un apprentissage de bout en bout et des vitesses en temps réel tout en maintenant une précision moyenne élevée**.[35]**

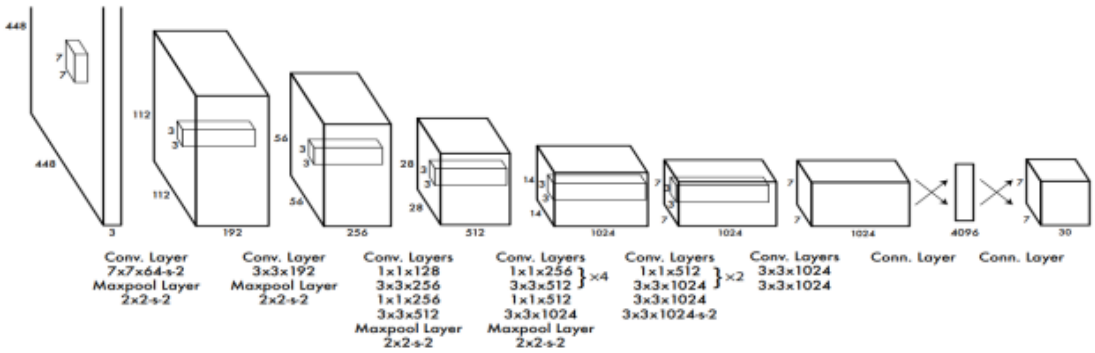


Figure III.7 : Architecture du modèle YOLO [43]

L'architecture de ce réseau est inspirée du modèle GoogLeNet pour la classification d'images. Ce réseau comporte 24 couches convolutives suivies de 2 couches entièrement connectées. Au lieu des modules d'initialisation utilisés par GoogLeNet, elle a utilisé simplement des couches de réduction 1 × 1 suivies de 3 × 3 couches convolutives [35]

L'algorithme de modèle YOLO est diviser on 3 étapes :

1) Diviser l'image en cellules avec une taille S x S

On divise l'image en une grille de taille S x S (en exemple 3 x3), ce qui donne N cellules au total. Cette cellule de la grille est responsable de la détection de cet objet.

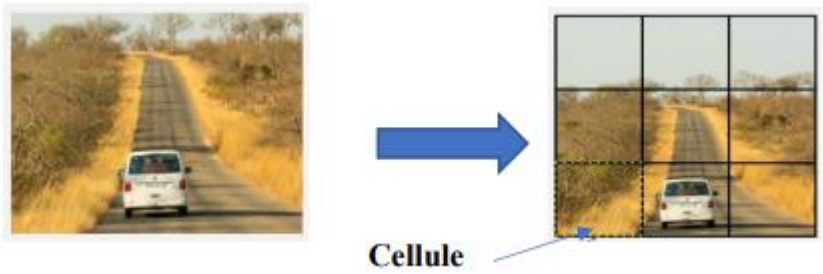


Figure III.8 : image cellulaire

2) Chaque cellule prédit B boîtes englobante

Après la divisons de l'image à N cellules, Chaque cellule de la grille prédit des boîtes englobante B et des scores de confiance pour ces boîtes, Chaque boîte englobante est constituée de 5 prédictions : x, y, w, h, et confiance. Les coordonnées (x, y) représentent le centre de la boîte par rapport aux limites de la cellule de la grille de la boîte par rapport aux limites de la cellule de la grille. La largeur et la hauteur sont prédites par rapport à l'image entière. Enfin, la prédiction de confiance représente le IoU entre la boîte la boîte prédite et toute boîte de vérité terrain [35]

Exemple : où il y a 3x3 cellules (S=3), chaque cellule prédit 1 boîte limitante (B=1), et les objets sont soit chien = 1, soit humain = 2, (C=2). Pour chaque cellule, le CNN prédit un vecteur Y :

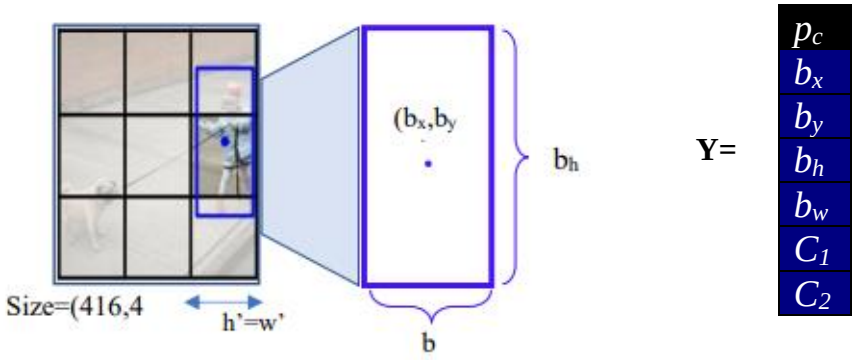


Figure III.9 : Le vecteur prédit dans le cas d'une seule boite

****p_c :** Probabilité que la boîte englobante contienne un objet
****b_x, b_y :** Coordonnées du centre de la boîte englobante
****b_h, b_w ;** Largeur (hauteur) du rectangle de délimitation en pourcentage de la largeur de la cellule ou(hauteur)
****c_1, c_2:** Probabilité que la cellule contienne un objet qui appartient à la classe 1 (ou 2) étant donné que la cellule contient un objet

Les valeurs de vecteur Y sont calculées au format YOLO :
Pc= la prédiction de confiance représente le IoU entre la boîte prédite et la boîte de vérité terrain.

$$bx = (x - h') /h', \quad by = (y - w') /w', \quad bh = h/416, \quad bw = w/416$$

a) Intersection sur Union (IoU)

L'intersection sur Union est une métrique d'évaluation utilisée pour mesurer la précision d'un détecteur d'objet sur un ensemble de données particulier. Nous voyons souvent cette métrique d'évaluation utilisée dans les défis de détection d'objets tels que le populaire défi PASCAL VOC.

Il compare la boîte prédite avec la boîte détectable et peut calculer la surface comme suit :

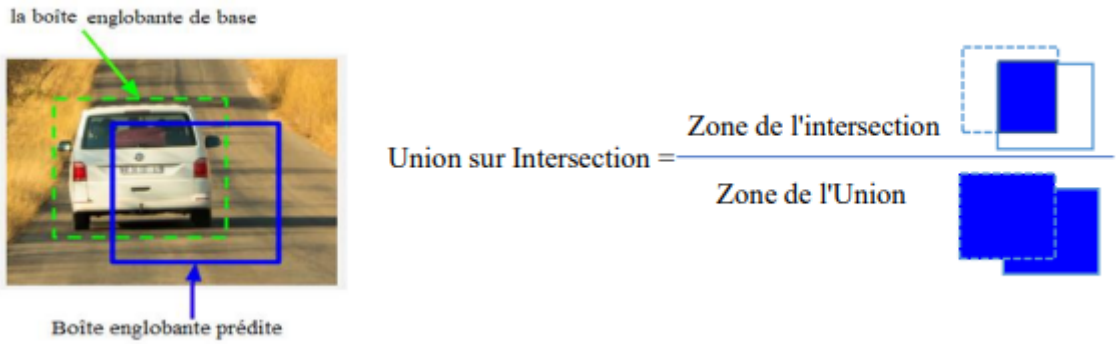


Figure III.10 : Union sur Intersection

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

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

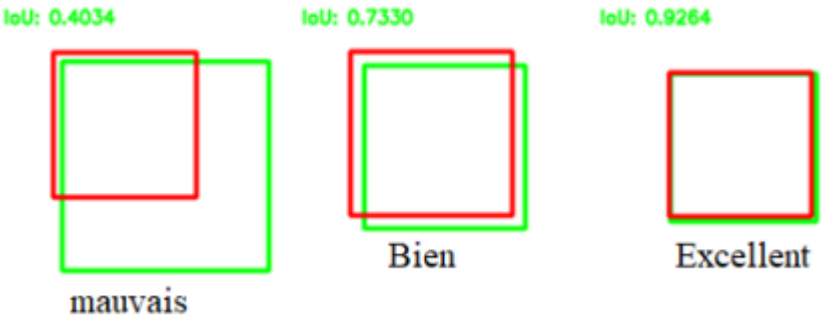


Figure III.11 : Exemples d'IoU

b) Boîte d'ancrage (Anchor Box) :

Dans l'exemple précédant poser qu'en a prédite une seule boite englobante mais si on a plus qu'une boite dans la même cellule, donc Les boîtes d'ancrage sont l'algorithme de YOLO qui sépare les objets si prédire plusieurs boîtes englobantes se trouvent dans la même cellule de grille**.[36]**

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

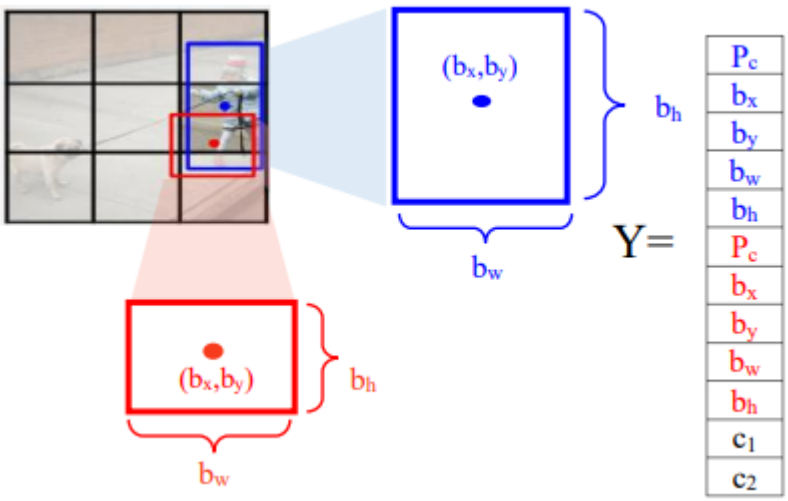


Figure III.12 : Le vecteur prédit dans le cas de plusieurs boites dans la cellule

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

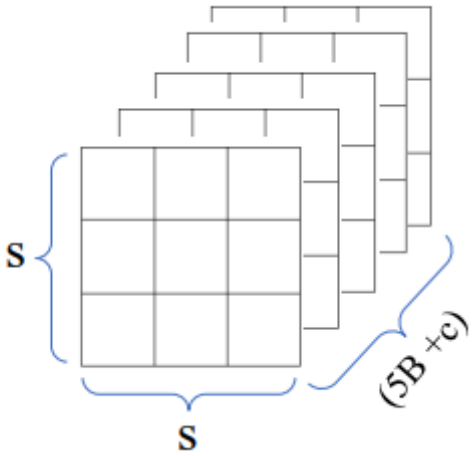


Figure III.13 : Un tenseur qui spécifie les emplacements de la boîte englobante et probabilités de classe.

3) Suppression non maximale

Cette étape c'est la dernière étape dans l'algorithme de détection, elle est utilisée c'est le même objet dans l'image détecter par plusieurs boit englobant, Cette technique est utilisée pour "supprimer" les boîtes englobantes les moins probables et ne garder que la meilleure. Alors le processus de cette technique passe à 5 étapes **[37]**

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

***Il y a eu 6 versions du modèle jusqu'à présent, chaque nouvelle version améliorant la précédente en termes de vitesse et de précision.**

1-3-6-1 Modèle YOLOv2 (Better, Faster, Stronger)

Dans cette version, Joseph Redmon et Ali Farhadi on essayer de créer un model meilleur, plus rapide, plus fort et pour cela ils ont fait une amélioration dans la 1 ère architecture de model YOLO, où ils ont utilisé Darknet-19 comme backbone, et la structure complète est passée à 30 couches, contre 26 couches pour YOLO v1 et inclusion de couches de normalisation par lots après chaque couche de convolution, aussi augmente la résolution à

448 pour la détection et grille avec stride=32 avec prédiction de 5 boîtes limitantes à chaque cellule, Les boîtes d'ancrage ont été introduites**.[29]**

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

Figure III.14 : Architecture du darknet19

1-3-6-2 Modèle YOLOv3 (une amélioration progressive)

Joseph Redmon et Ali Farhadi en 2018 ont aussi fait une amélioration progressive dans la version précèdent, où ils ont utilisé comme backbone Darknet53 pour extraction de caractéristique, et pour calculer un score d'objectalité de chaque boîte de délimitation en utilisant une régression logistique. Pour les prédictions de classe ils ont utilisés la perte d'entropie croisée binaire. Dans la version de yolov2 il y a un problème pour la détection des petits objets mais dans cette version la représentation des boîtes à 3 échelles différentes**.[29]**

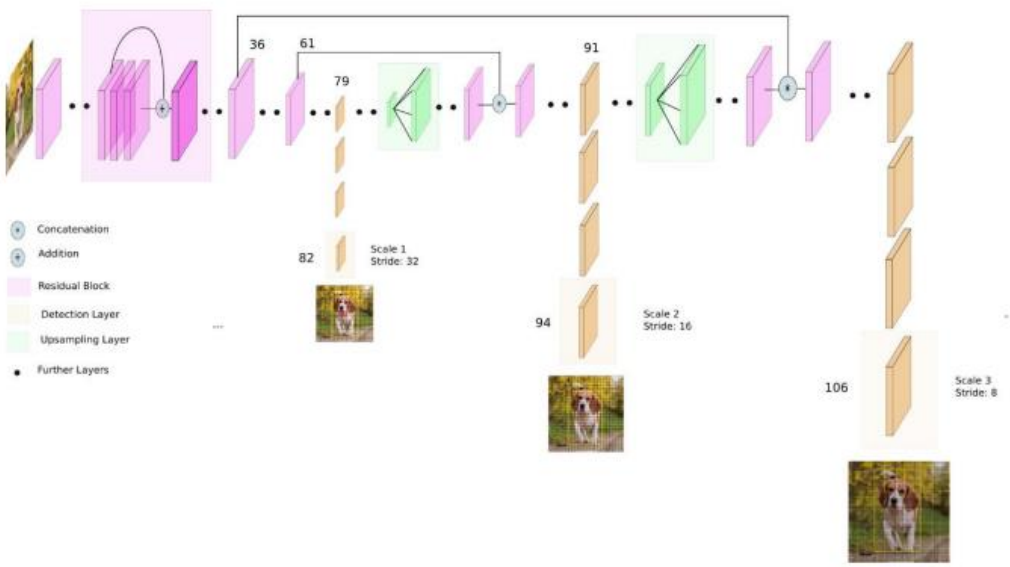


Figure III.15 : Architecture du modèle YOLOv3

1-3-6-3 Modèle YOLOv4

Comme tous les modèles de détection d'objet yolov4 composé de trois parties :

Backbone : CSPDarknet53, **Neck :** SPP, PANet, **Head :** Même que YOLOv3

Dans la figure suivant nous allons illustrer l'architecture de yolov4 on détaille :

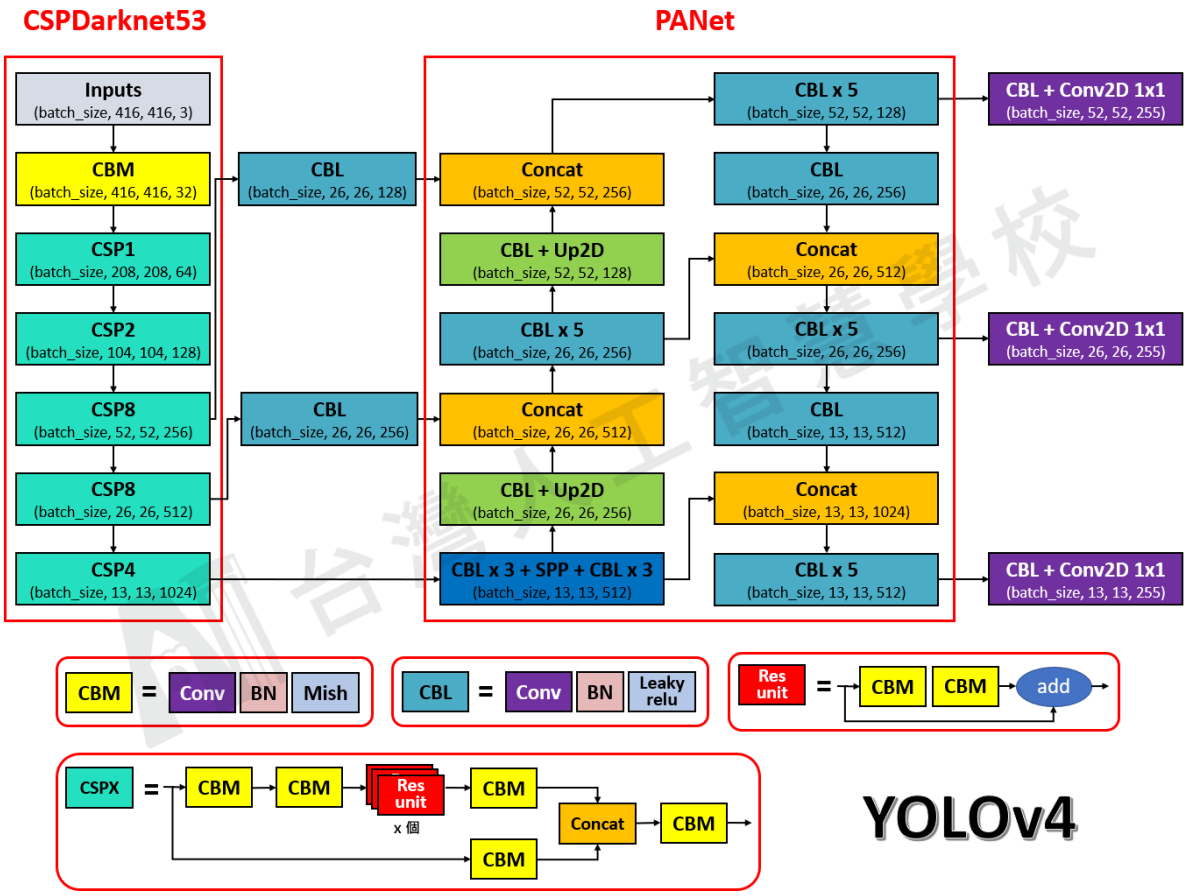


Figure III.16 : Architecture du modèle YOLOv4

YOLO v4 utilise Bag of freebies (BOF) et Bag of specials (BOS). Le BoF fait, référence aux méthodes qui affectent la stratégie de formation, BoS sont des modules et des méthodes de post-traitement qui augmentent le coût de l'inférence mais améliorent également la précision de la détection des objets.

- ✓ Sac de gratuité (BoF) pour backbone : CutMix et Mosaic, régularisation DropBlock, Lissage des étiquettes de classe
- ✓ Sac de spécialités (BoS) pour le backbone : Activation de Mish, connexions partielles inter-étapes (CSP), connexions résiduelles pondérées à entrées multiples (MiWRC).
- ✓ Sac de gratuité (BoF) pour le détecteur : Perte de CIoU, CmBN, régularisation DropBlock, augmentation des données en mosaïque, formation auto-adversaire, élimination de la sensibilité de la grille, utilisation de plusieurs ancrages pour les détecteurs. Sensibilité de la grille, utilisation de plusieurs ancrages pour une seule vérité de base, planificateur de recuit en cosinus, hyperparamètres optimaux, formes d'entraînement aléatoires
- ✓ Sac de spécialités (BoS) pour le détecteur : Activation de Mish, Bloc SPP, bloc SAM, bloc d'agrégation de chemins PAN, DIoU-NMS

1) CSPDarknet53 :

CSPDarknet53 est un réseau neuronal convolutif et une colonne vertébrale pour la détection d'objets qui utilise DarkNet-53. Il utilise une stratégie CSPNet pour diviser la carte des caractéristiques de la couche de base en deux parties, puis les fusionne via une hiérarchie à plusieurs étapes. L'utilisation d'une stratégie de division et de fusion permet un flux plus dégradé à travers le réseau**.[38]**

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

Figure III.17 : Architecture du darknet53

2) La pyramide spatiale Couche de mise en commun (SPP)

Le Pooling pyramidal spatial (SPP) est une couche de pooling qui supprime la contrainte de taille fixe du réseau, c'est-à-dire qu'un CNN ne nécessite pas d'image d'entrée de taille fixe. La couche SPP regroupe les caractéristiques et génère des sorties de longueur fixe, qui sont ensuite introduites dans les couches entièrement connectées (ou d'autres classificateurs).

La mise en commun pyramidale spatiale (SPP) ajoute une nouvelle couche entre les couches convolutionnelles et les couches entièrement connectées. Son travail consiste à mapper n'importe quelle entrée de taille vers une sortie de taille fixe. **[39]**

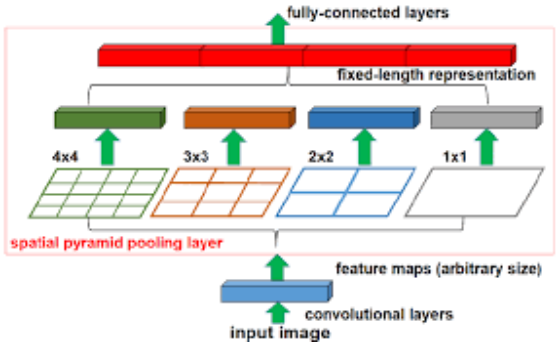


Figure III.18 : Une structure SPP

3) Réseau d'agrégation de chemins (Path Aggregation Network PAN)

La façon dont l'information se propage dans les réseaux de neurones est d'une grande importance. Dans cet article, nous proposons un réseau d'agrégation de chemins (PANet) visant à stimuler le flux d'informations dans un cadre de segmentation d'instances basé sur des propositions. Plus précisément, nous améliorons l'ensemble de la hiérarchie des fonctionnalités avec des signaux de localisation précis dans les couches inférieures par une augmentation de chemin ascendante, ce qui raccourcit le chemin d'information entre les couches inférieures et la fonctionnalité la plus élevée. Nous présentons le regroupement de fonctionnalités adaptatif, qui relie la grille de fonctionnalités et tous les niveaux de fonctionnalités pour que les informations utiles de chaque niveau de fonctionnalité se propagent directement aux sous-réseaux de proposition suivants. Une branche complémentaire capturant différentes vues pour chaque proposition est créée pour améliorer encore la prédiction du masque. Ces améliorations sont simples à mettre en œuvre, avec une surcharge de calcul supplémentaire subtile. Notre PANet atteint la 1ère place dans la tâche COCO 2017 Challenge Instance Segmentation et la 2ème place dans la tâche Object Detection sans formation en gros lots. Il est également à la pointe de la technologie sur MVD et Cityscapes**.[40]**

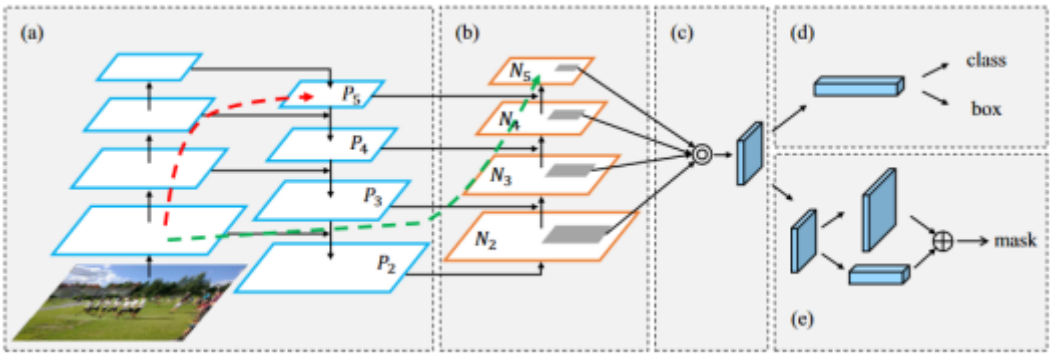


Figure III.19 : Architectures de Réseau d'agrégation de chemins (PAN)

Dans l'architecture de yolov4 ils ont modifiés la SAM pour passer d'une attention spatiale à une attention ponctuelle, et nous remplaçons le raccourci de connexion du PAN par une concaténation, comme le montrent respectivement dans la **figure III.20**

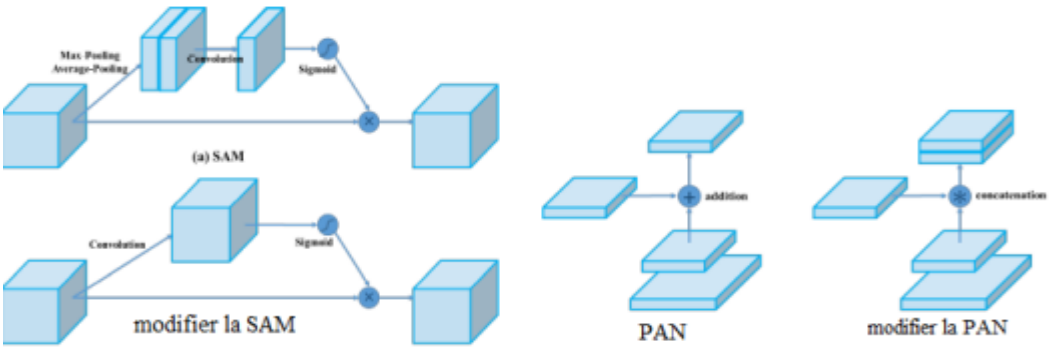


Figure III.20 : Modification des SAM et PAN

1-3-6-4 Modèle YOLOv5 :

Yolov5 ressemble presque à Yolov4 avec certaines des différences suivantes :

- Yolov4 est publié dans le framework Darknet, qui est **écrit en C**. Yolov5 est basé sur le framework PyTorch**.**
- Yolov4 utilise **.cfg** pour la configuration tandis que Yolov5 utilise le fichier **.yaml** pour la configuration. **[41]**

1-3-6-4-1 Architecture du YOLOv5

L'architecture réseau de Yolov5. Il se compose de trois parties :
(1) Backbone : CSPDarknet : Un réseau neuronal convolutif qui agrège et forme des caractéristiques d'image à différentes granularités.
(2) Neck : PANet Une série de couches pour mélanger et combiner les caractéristiques de l'image pour les transmettre à la prédiction.
(3) Head : Yolo Layer Consomme les fonctionnalités du cou et prend des étapes de prédiction de boîte et de classe.

Les données sont d'abord entrées dans CSPDarknet pour l'extraction de caractéristiques, puis transmises à PANet pour la fusion de caractéristiques. Enfin, Yolo Layer génère des résultats de détection (classe, score, emplacement, taille**).[42]**

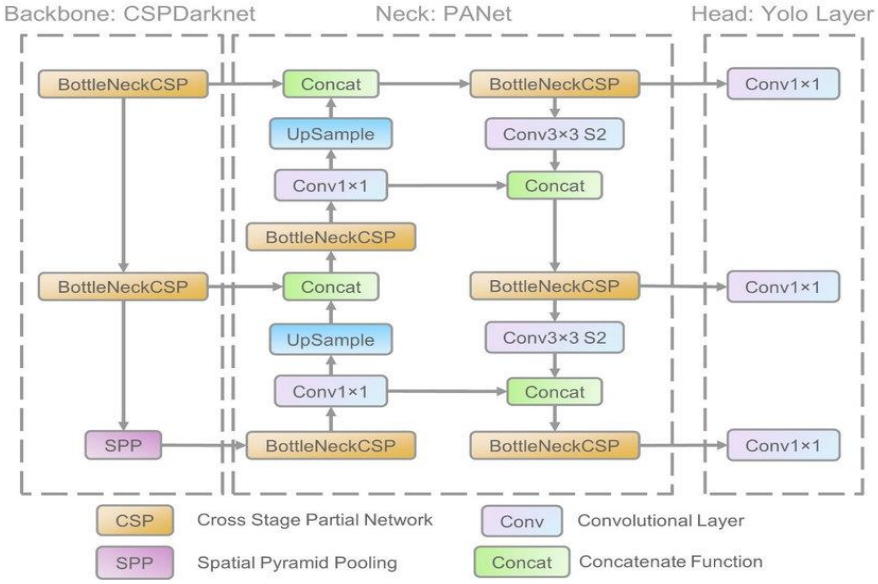


Figure II.21 : Architecture du YOLOv5 [29]

- ✓ **Activation et optimisation :** YOLOv5 utilise l'activation sigmoïde et ReLU fuyante, ainsi que SGD et ADAM comme options d'optimisation.
- ✓ **Fonction de perte :** Il utilise l'entropie croisée binaire avec une perte logits

**** Dans le graphique, l'objectif est de produire un modèle de détecteur d'objet très performant (axe Y) par rapport à son temps d'inférence (axe X). Les résultats préliminaires montrent que YOLOv5 réussit extrêmement bien à cette fin par rapport à d'autres techniques de pointe.

Pour entraîner le YOLOv5 Glenn a proposé 4 versions.

1- YOLOv5-s qui est une petite version
2- YOLOv5-m qui est une version moyenne
3- YOLOv5-l qui est une grande version
4- YOLOv5-x qui est une version extra-large

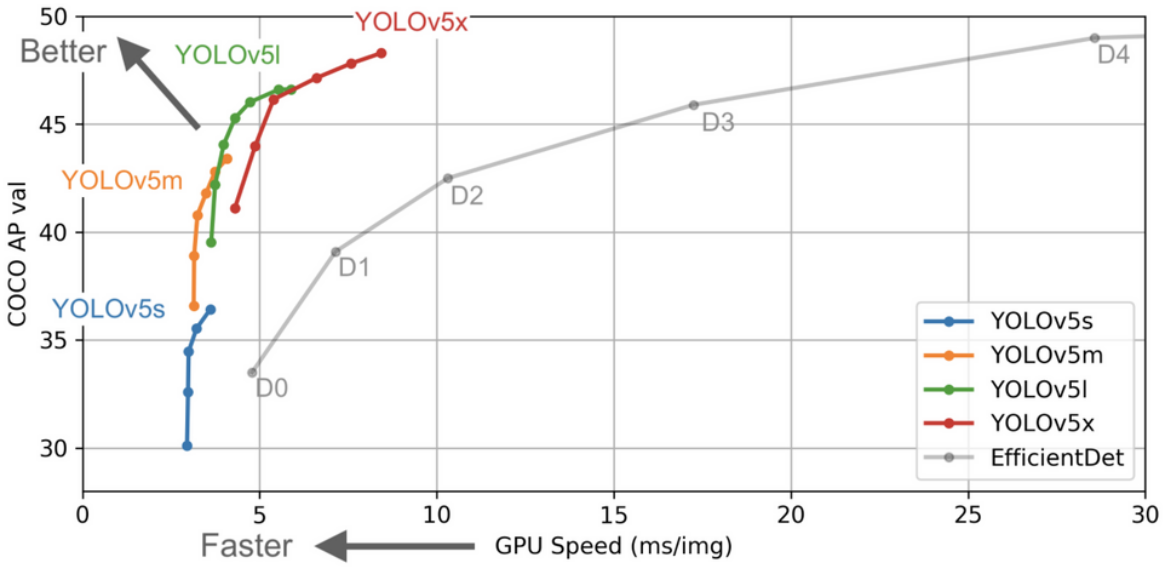


Figure II.22 : Graphe de performance

Conclusion

Dans ce chapitre, nous avons présenté un aperçu des méthodes de détection d'objets basées sur l'apprentissage en profondeur, Nous avons commencé par structure générale d'un modèle de détection base sur 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 YOLO. Nous avons vu leur algorithme détaillé avec les 5 version de YOLO et la différence entre chaque version et basé beaucoup plus sur la version YOLOv5 qui il utilise dans notre conception, on conclure avec la mesure de performance d'un modèle de détection

Chapitre IV :
Conception et implémentation

I Conception

1-Introduction

Ce dernier chapitre est consacré à la conception et la mise en place de notre projet qui permettra d'Identifier la carte nationale d'identité et d'identifier des extractions texte dans des images en utilisant modèle de détection d'objets qui basé sur méthode prédire de la boite englobante. Dans ce chapitre nous allons décrire également les différentes parties de notre système, les détails relatifs à chaque phase ainsi leurs interactions qui sont présentées dans les sous-sections suivantes**.**

2-Conception générale de notre système

Notre système se composé de quatre étapes principales

- La collecte de la base de donnés
- Prétraitement de base de donnés
- Apprentissage de model yolov5
- Test et résultat

2-1 La collecte de la base de donnés

La collecte de données n'était pas facile, car les données utilisées sont très sensibles, C'est pourquoi nous avons eu du mal à collecter des cartes nationales biométriques, Nous sommes allés à la commune de Fouka afin de nous fournir des données pour les cartes d'identité nationales sur lesquelles travailler, mais notre demande a été rejeté en raison de la confidentialité des informations personnelles , Nous avons donc décidé de demander cela à nos parents, amis, voisins, et membre de la famille, certains nous les ont donné simplement, d'autres difficilement.

Nous avons collecté environ 191 photos d'identités valides et 153 photos aléatoires pour le train, comme c'est montré respectivement dans la **figure IV.1** et la **figure IV.2**

Figure IV.1 : exemple d'image valide de la base de donné

Figure IV.2 : exemple d'image non valide de la base de donné

2-1-1 Choix des classes du modèle (1)

D'après le concept de notre premier modèle classifier les pièces d'identification (carte nationale), nous voyons que nous n'avons besoin que de deux classes, nous avons donc deux classes traités. Une carte valide et une carte non valide

2-1-2 Choix des classes du modèle (2)

Ici, le choix des classes est selon nos besoins d'informations d'identifier la carte el les informations de son propriétaire, don nous avons choisir id de la carte, le nom, le prénom, la date de naissance, et surtout la photo de la personne.

2-2 Prétraitement de la base de données

2-2-1 Prétraitement de la 1er base

Lorsque nous avons recherché le fonctionnement de YOLOv5, nous avons constaté qu'il fonctionne avec labels. Alors on a utilisé le site **makesense.ia** pour faire une classification entre les cartes nationales valides et non valides, comme le montre la figure ci-dessous.

Nous avons annoté les cartes dans les images collectées, et publié le projet au format de zip.

Figure IV.3 : image valide de la base de données avec son annotation

Figure IV.4 : image non valide de la base de données avec son annotation

- La classe indiquée dans le fichier d'annotation est la classe 0 (une carte valide). **Figure IV.3**
- La classe indiquée dans le fichier d'annotation est la classe 1 (une carte non valide). **Figure IV.4**

Une ligne a le format suivant :

<Numéro de classe><centre-x><centre-y><largeur><la hauteur>

Numéro de classe : L'index de la classe dans la liste des classes

Centre_x : La valeur x du centre normalisé de la boîte englobante

Centre_y : Valeur normalisée du centre y de la boîte englobante

Largeur : La valeur de la largeur normalisée de la boîte englobante.

Hauteur : La valeur de la hauteur normalisée de la boîte englobante.

2-2-2 Prétraitement de 2éme base

Nous avons pris les cartes d'identité correctes, et avons fait une localisation ou bien Bounding box sur les éléments requis, que le système doit connaître et lire dans une bibliothèque OCR, donc nous avons utilisé le même site **makesense.ia** pour la labélisation.

Nous avons choisi les informations suivantes :

Id_card, familly_name, name, birthday, face_image

Figure IV.5: image labélisé de la base de données avec son annotation

- La classe indiquée dans le fichier d'annotation est la classe 0 (id_card)
- La classe indiquée dans le fichier d'annotation est la classe 1 (familly_name).
- La classe indiquée dans le fichier d'annotation est la classe 2 (name).
- La classe indiquée dans le fichier d'annotation est la classe 3 (birthday).
- La classe indiquée dans le fichier d'annotation est la classe 4 (face_image).

Avant d'entrer à l'apprentissage du modèle de YOLOv5 nous avons importé notre base de données à google drive

Nom ↓	Propriétaire	Dernière modification
test_card_information	moi	8 sept. 2022 moi
test_card_detection	moi	8 sept. 2022 moi
New_cards_data_2	moi	8 sept. 2022 moi
data2	moi	8 sept. 2022 moi

Figure IV.6 : les bases de données de train et de test

2- 3 Apprentissage du modèle YOLOv5

Ce module contient deux étapes principales qui sont :

2-3-1 Attachement du YOLOv5 sur l'environnement de travail 'Google Colab »

Étant donné que Colab est un service Google, il permet de se lier à un compte Google Drive personnel pour obtenir des données de ce dernier (**figure IV.6)** qui sont à utiliser dans l'entraînement du modèle.

```
#upload google drive in our colab

from google.colab import drive
drive.mount('/content/drive')
```

Figure IV.7 : Montage sur Google Drive.

Ensuite télécharger le fichier **ultralytics** qui contient le modèle YOLOv5 avec la commande :

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

Figure IV.8: Installing the YOLOv5 Environment

Lorsque le référentiel est cloné, installez les exigences :

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

Figure IV.9 : installation des exigences

Ensuite, nous avons préparé l'enivrement Google Colab par la configuration de certains paramètres.

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

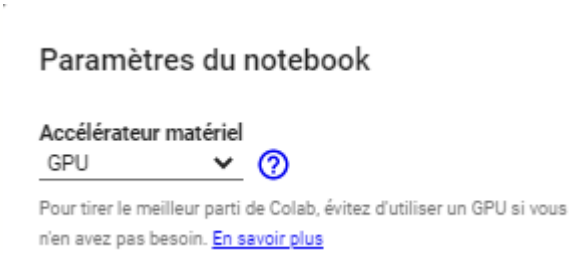


Figure IV.10 : paramètre du notebook

2-3-2 Préparation du modèle pour le training

Cette étape consiste à entrainer le modèle **(Figure IV.11)** après sa configuration

```
!python train.py --img 1080 --batch 4 --epochs 120 --data cards.yaml --weights yolov5l6.pt --cache
```

Figure IV.11 : Mettre en œuvre le processus de formation pour la première base de données

```
!python train.py --img 720 --batch 2 --epochs 100 --data cart_info.yaml --weights yolov5s.pt --cache
```

Figure IV.12 : Mettre en œuvre le processus de formation pour la deuxième base de données

Notez que ces arguments spécifient la formation :

Où :

- **img :** pour définir la taille de l'image d'entrée. « vous pouvez choisir une autre taille. »
- **batch :** pour définir la taille du lot et doit toujours être le plus grand possible
- **epochs :** pour définir le nombre d'époques d'entraînement.
- **data :** le chemin d'accès au fichier data.yaml contenant le résumé de l'ensemble de données
- **wheights** : pour spécifier un chemin vers les poids.
- **cache:** images de cache pour une formation plus rapide

*****Définition de fichier.Yaml**

Un fichier YAML décrie les chemins et les noms. Le dossier contenant les fichiers du jeu des données doit être situé à côté du dossier YOLOv5. En d'autres termes, dans le dossier du projet et non dans le dossier cloné. Le fichier YAML doit se trouver dans le dossier YOLOv5/data.

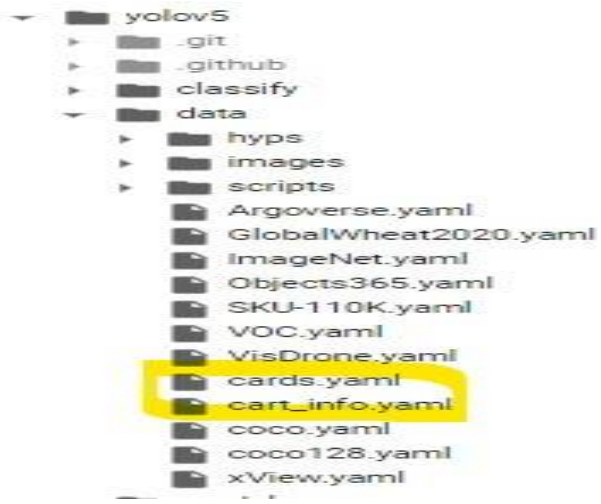


Figure IV .13: Emplacement du fichier Yaml dans YOLOv5

2-3-3 Détection du modèle

2-3-3-1 Etape pour faire une détection sur la carte

Nous pouvons choisir les meilleurs ou les derniers poids, où les poids entraînés peuvent être utilisés pour la détection de la classe d'image.

```
!python detect.py --source '/content/drive/MyDrive/test_card_detection' --weights '/content/best_card_detection.pt' --conf 0.6 --iou 0.45 --augment --project 'asmaa' --name 'detect_test_22'
```

Figure IV.14 : Exemple de détection de classe

2-3-3-2 Etape pour faire l'extraction des informations depuis une carte valide

Nous pouvons choisir les meilleurs ou les derniers poids, où les poids entraînés peuvent être utilisés pour l'extraction des informations

```
!python detect.py --source '/content/drive/MyDrive/test_card_information' --weights '/content/best_card_information.pt'
--conf 0.6 --iou 0.45 --augment --project 'asmaa_information' --name 'detect_test_1'
```

Figure IV.15 : Exemple d'extraction des informations (la détection)

Où :

Source: Chemin d'entrée

Weights: chemin des poids

Conf : seuil de confiance

Iou : Seuil IoU pour NMS (Non Max Suppression)

Augment : inférence augmentée

2-3-4 Etape pour détecter les coordonnées de chaque classe dans la carte (bouding box)

Nous avons chargé un modèle YOLOv5s personnalisé formé sur mesure "best_card_information.pt" avec PyTorch Hub.

```
[ ]  model= torch.hub.load('ultralytics/yolov5','custom','/content/best_card_information.pt')
```

Figure IV.16 : load a model (custom trained model)

Les modèles YOLOv5 contiennent divers attributs d'inférence tels que le seuil de confiance, donc on a défini la confiance à 0.6.

```
# Set Confidence
model.conf = 0.6
```

Figure IV.17 : le seuil de la confiance

Nous avons mis un lot d'images et le size et transmet une image pour l'inférence

```
#predection
results = model("/content/drive/MyDrive/test_card_information/303611828_1554967798239998_6689758452834536604_n.jpg", size=720)
```

Figure IV.18: Path and size

```
#Get labels and cords
labels, cord_thres = results.xyxyn[0][:, -1].numpy(), results.xyxyn[0][:,:-1].numpy()
```

Figure IV.19 : GET labels and coordinate

2-3-5 Analyse d'une carte d'identité

Nous allons retrouver les mêmes difficultés.Nous utiliserons comme language Python et les librairies OpenCV et Tesseract (pour l'OCR). Les premières lignes ne doivent pas générer d'erreur sans quoi vous devrez importer les librairies via pip par exemple.

```
[ ]  try:
         from PIL import Image
     except ImportError:
         import Image
     import pytesseract
     from pytesseract import Output
     import cv2
     import numpy as np
     from matplotlib import pyplot as plt
     import matplotlib.pyplot as plt
     import matplotlib.image as img
     import sys
     from google.colab.patches import cv2_imshow
```

Figure IV.20 : importation des librairies

2-3-5-1 Découpage de la photo

Nous allons tout d'abord reconnaître et découper la photo d'identité en utilisant la reconnaissance faciale d'OpenCV.

```
[ ]
    #Découpage de la photo
    imagePath = r'/content/drive/MyDrive/test_card_information/303611828_1554967798239998_6689758452834536604_n.jpg'
    dirCascadeFiles = r'../opencv/haarcascades_cuda/'
    image = cv2.imread(imagePath)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    cascadefile = cv2.data.haarcascades + "haarcascade_frontalface_alt.xml"
    classCascade = cv2.CascadeClassifier(cascadefile)
    faces = classCascade.detectMultiScale(
        gray,
        scaleFactor=1.1,
        minNeighbors=5,
        minSize=(30, 30),
        flags = cv2.CASCADE_SCALE_IMAGE
    )
    print("Il y a {0} visage(s).".format(len(faces)))
    # Coordonnées des rectangles des visages détectés (x, y, w, h)
    for (x, y, w, h) in faces:
        cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)

    cv2_imshow(image)
```

Figure IV.21 : Code python

Nous avons utilisé ce modèle la : haarcascade_frontalface_default.xml pour ce type de détection.(reconnaissance faciale).

Nous pouvons donc l'extraire simplement en faisant une opération de slicing avec numpy :

```
#extraction de face image
f = faces[0]
cv2_imshow(image[f[1]:f[1]+f[3], f[0]:f[0]+f[2]])
```

Figure IV.22 : Extraction de face image

2-3-6 Récupération des autres informations

L'objectif est maintenant de récupérer les informations labélisées de la carte d'identité sous forme de bouding box avec ses noms et cordonnés comme pourcentage.

```python
#Récupération des autres informations
# Coordonnées des rectangle des autres informations
cpt=0
for (x,y,w,h,z) in cord_thres:
  num1=int(labels[cpt])
  list_class=['id_card', 'family_name', 'name', 'birthday', 'face_image' ]
  name1=list_class[num1]

  x=int(x*1128)
  y=int(y*2000)
  h=int(h*1128)
  w=int(w*2000)
  z=int(z*100)
  print(x,y,h,w,z)
  print(name1)
  name1_image = image [x:x+h ,y:y+w]
  cv2_imshow(name1_image)
  cv2.rectangle(image,(x,y),(x+w,y+h),(0,255,0),2)
  print(name1)
  cpt=cpt+1
```

Figure IV.23 : Code python pour la récupération des autres informations

2-4 Tests et résultats

Dans cette partie nous allons tester/valider les performances du modèle YOLOv5 ré-entrainé sur les deux classes.

Ce test est établi sur des données choisies, et on a obtenu de bons résultats même dans des scènes complexes.

On peut voir certains résultats obtenus dans la figure suivante **figure IV.26**

Figure IV.24 : Le Résultat Obtenu de Détection par Yolov5 sur le **modèle(1)**

Nous avons fait des tests sur **le modèle (2)** et obtenu les résultats suivants :

Figure IV.25 : Le Résultat Obtenu de Détection par YOLOv5 sur le **modèle(2)**

2-4-1 Discussions

Afin de montrer les résultats obtenus par notre modèle, on illustre dans ce qui suit les résultats en termes de précision et rappel ainsi que la précision moyenne (mAP) (the mean Average Precision).

A. Résultats obtenus du 1 er modèle

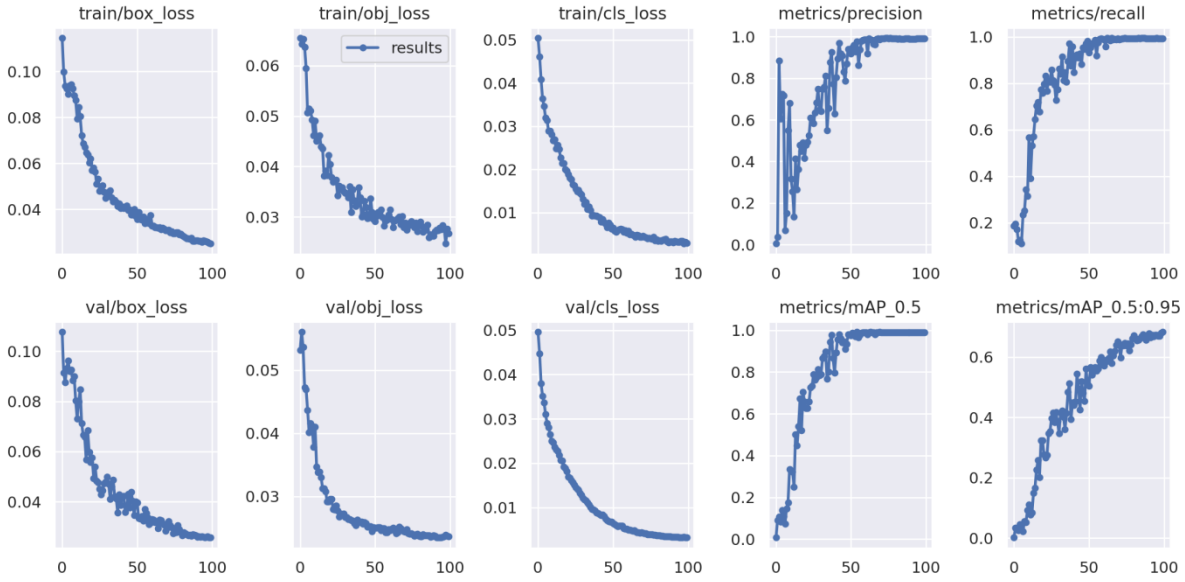


Figure IV.26 : Graph Précision, rappel et mAP pour le modèle (1)

```
Validating runs/train/exp3/weights/best.pt...
Fusing layers...
Model summary: 476 layers, 76126356 parameters, 0 gradients, 109.9 GFLOPs
                 Class     Images  Instances          P          R      mAP@.5 mAP@.5:.95: 100% 9/9 [00:02<00:00,  3
                   all         69         45      0.734      0.955      0.928      0.896
                  card         69         15      0.695          1      0.934      0.934
              not_card         69         30      0.773       0.91      0.923      0.859
Results saved to runs/train/exp3
```

Figure IV.27 : résultat obtenu pour la valeur de précision, recall et le mAP modèle(1)

```
Fusing layers...
Model summary: 476 layers, 76126356 parameters, 0 gradients, 109.9 GFLOPs
image 1/14 /content/drive/MyDrive/test_card_detection/301962551_489221616363467_8068486136354125521_n.jpg: 640x384 1 not_card, 121.9ms
image 2/14 /content/drive/MyDrive/test_card_detection/302056697_380139710962927_2607933388562526810_n.jpg: 640x384 1 card, 1 not_card, 123.9ms
image 3/14 /content/drive/MyDrive/test_card_detection/302065428_641787473837501_4704009851924172983_n.jpg: 640x384 1 card, 1 not_card, 123.8ms
image 4/14 /content/drive/MyDrive/test_card_detection/302085627_471794054580005_5753484012769093379_n.jpg: 384x640 1 not_card, 116.4ms
image 5/14 /content/drive/MyDrive/test_card_detection/302390141_970322941029345_398117403071478143_n.jpg: 384x640 1 not_card, 103.2ms
image 6/14 /content/drive/MyDrive/test_card_detection/303611828_1554967798239998_6689758452834536604_n.jpg: 640x384 1 card, 123.8ms
image 7/14 /content/drive/MyDrive/test_card_detection/304797267_3211687889047646_9004079519462832131_n.jpg: 640x384 1 card, 120.5ms
image 8/14 /content/drive/MyDrive/test_card_detection/304986036_617079056733762_2962477571772969189_n.jpg: 384x640 1 card, 121.2ms
image 9/14 /content/drive/MyDrive/test_card_detection/304987842_425760952872967_673250862962048147_n.jpg: 384x640 1 card, 1 not_card, 85.4ms
image 10/14 /content/drive/MyDrive/test_card_detection/305247836_5355607924536869_8812062738134957001_n.jpg: 640x384 1 card, 83.2ms
image 11/14 /content/drive/MyDrive/test_card_detection/305249068_617199950012863_6735310732157378548_n.jpg: 384x640 1 not_card, 121.3ms
image 12/14 /content/drive/MyDrive/test_card_detection/305533375_772700337325904_1918379669506898676_n.jpg: 384x640 1 card, 1 not_card, 85.5ms
image 13/14 /content/drive/MyDrive/test_card_detection/305572717_1405632259921538_6369252228904104 74_n.jpg: 384x640 1 card, 121.3ms
image 14/14 /content/drive/MyDrive/test_card_detection/305645289_775120237071202_4826978820675221158_n.jpg: 384x640 1 not_card, 119.9ms
Speed: 0.6ms pre-process, 112.2ms inference, 1.0ms NMS per image at shape (1, 3, 640, 640)
Results saved to asmaa/detect_test_223
```

Figure IV.28 : résultat de la classification de test carte

Après l'analyse des résultats obtenus,

On constate les remarques suivantes : D'après la **Figure IV.28** la précision et le rappel de l'apprentissage augmentent avec le nombre d'époques, ceci reflète qu'à chaque époque le modèle apprenne plus d'informations. Si la précision est faible on aura besoin de plus d'information pour faire apprendre notre modèle et par conséquent on doit augmenter le nombre d'époques et vice versa.

De même, Le mAP augmente avec le nombre d'époques. Le mAP compare la boîte englobante de la vérité au sol à la boîte détectée et renvoie un score. Plus le score est élevé, plus le modèle est précis dans ses détections.

Le résultat de l'apprentissage obtenu avec mAP@0.5 c'est 92.8%Dans le tableau suivant, on a affiché tous les résultats obtenus :

Précision	Rappel	mAP@0.5	mAP@0.5: .95
73.4%	95.5%	92.8%	89.6 %

TableauIV.1 : Les résultats de l'apprentissage modèle(1)

Ce résultat confirme l'efficacité de notre approche pour prédire correctement les signes exécutés dans divers environnements.

B. Résultats obtenus du modèle 2

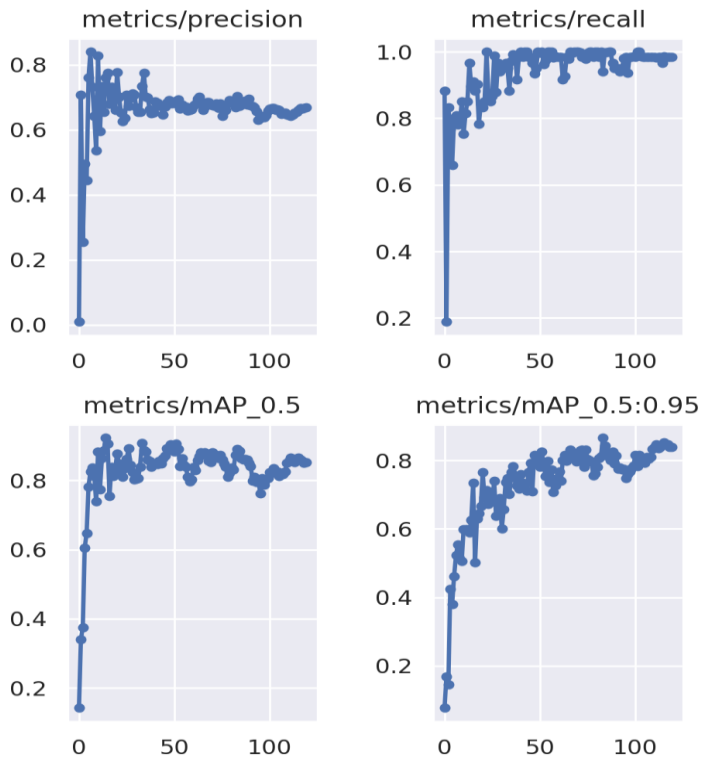


Figure IV.29 : Précision, rappel et mAP pour le modèle (2)

```
Validating runs/train/exp2/weights/best.pt...
Fusing layers...
Model summary: 213 layers, 7023610 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP@.5 mAP@.5:.95: 100% 11/11 [00:01<00:00,  9.46it/s]
                   all         42        210      0.992      0.993       0.99      0.681
             id_card           42         42          1          1      0.995      0.649
         family_name           42         43          1      0.966      0.987      0.619
                name           42         41       0.97          1       0.98      0.634
          birth_day             42         42      0.995          1      0.995      0.701
          face_image            42         42      0.994          1      0.995      0.801
Results saved to runs/train/exp2
```

Figure IV.30 : résultat obtenu pour la valeur de précision, recall et le mAP modèle (2)

```
Fusing layers...
Model summary: 213 layers, 7023610 parameters, 0 gradients, 15.8 GFLOPs
image 1/5 /content/drive/MyDrive/test_card_information/302065428_641787473837501_4704009851924172983_n.jpg: 640x384 1 id_card, 1 family_name, 1 name, 1 birth_day, 1 face_image, 620.2ms
image 2/5 /content/drive/MyDrive/test_card_information/303611828_1554967798239998_6689758452834536604_n.jpg: 640x384 1 id_card, 1 family_name, 1 name, 1 birth_day, 2 face_images, 586.5ms
image 3/5 /content/drive/MyDrive/test_card_information/304797267_3211687889047646_9004079519462832131_n.jpg: 640x384 1 id_card, 1 family_name, 1 name, 1 birth_day, 1 face_image, 592.1ms
image 4/5 /content/drive/MyDrive/test_card_information/304986036_617079056733762_2962477571772969189_n.jpg: 384x640 1 id_card, 1 family_name, 1 name, 1 birth_day, 1 face_image, 583.2ms
image 5/5 /content/drive/MyDrive/test_card_information/305247836_5355607924536869_8812062738134957001_n.jpg: 640x384 1 id_card, 1 family_name, 1 name, 1 birth_day, 1 face_image, 577.1ms
Speed: 1.3ms pre-process, 591.8ms inference, 0.9ms NMS per image at shape (1, 3, 640, 640)
Results saved to asmaa_information/detect_test_1
```

Figure IV.31 : résultat sur les informations détectées

Après l'analyse des résultats obtenus,

On constate les remarques suivantes : D'après la **Figure IV.31** la précision et le rappel de l'apprentissage augmentent avec le nombre d'époques, ceci reflète qu'à chaque époque le

modèle apprenne plus d'informations. Si la précision est faible alors on aura besoin de plus d'information pour faire apprendre notre modèle et par conséquent on doit augmenter le nombre d'époques et vice versa.

De même, Le mAP augmente avec le nombre d'époques. Le mAP compare la boîte englobante de la vérité au sol à la boîte détectée et renvoie un score. Plus le score est élevé, plus le modèle est précis dans ses détections.

Le résultat de l'apprentissage obtenu avec mAP@0.5 est 99% dans le tableau suivant, on a affiché tous les résultats obtenus :

Classe	Précision	Rappel	mAP@0.5	mAP@0.5:0.95
All	99.2%	99.3%	99%	68.1%
Id_card	1%	1%	99.5%	64.9%
Family_name	1%	99.6%	98.7%	61.9%
Name	97%	1%	98%	63.4%
Birth_day	99.5%	1%	99.5%	70.1%
Face_image	99.4%	1%	99.5%	80.1%

TableauIV.2 : Les résultats de l'apprentissage modèle2)

Ce résultat confirme l'efficacité de notre approche pour prédire correctement les signes exécutés dans divers environnements.

C. **Résultats obtenus des coordonnées de la boîte englobante de chaque classe**

Après l'exécution nous avons obtenus ces résultats qui affichent le numéro de la classe et ses coordonnées avec le pourcentage de prédiction de la boite englobante :

```
#Get labels and cords
labels,cord_thres = results.xyxyn[0][:, -1].numpy(), results.xyxyn[0][:,:-1].numpy()
print(labels)
print (cord_thres)
```

```
[        4         0         3         1         2]
[[  0.043889   0.36725   0.30604   0.57096   0.93771]
 [   0.30074   0.33887   0.48877   0.36827   0.93059]
 [   0.65987   0.54819   0.81278   0.57449   0.88095]
 [   0.81857   0.48077   0.88525   0.50438   0.82951]
 [   0.80962   0.51476   0.88291   0.54162   0.81736]]
```

Figure IV.32 : résultats de l'obtention des coordonnées avec l'index de labels

D. **Résultats obtenus du découpage de la photo**

Quand nous appliquons le **modèle haarcascade_frontalface_alt.xml**.,nous obtenons un résultat comme celui dans la **Figure IV.35** :

Figure IV.33: découpage de la photo

E. Résultats obtenus de l'extraction de « face image »

Après l'exécution de code nous avons obtenir ce résultat

Figure IV.34 : Résultat d'extraction de « face-image »

F. Résultats obtenus de la récupération des informations

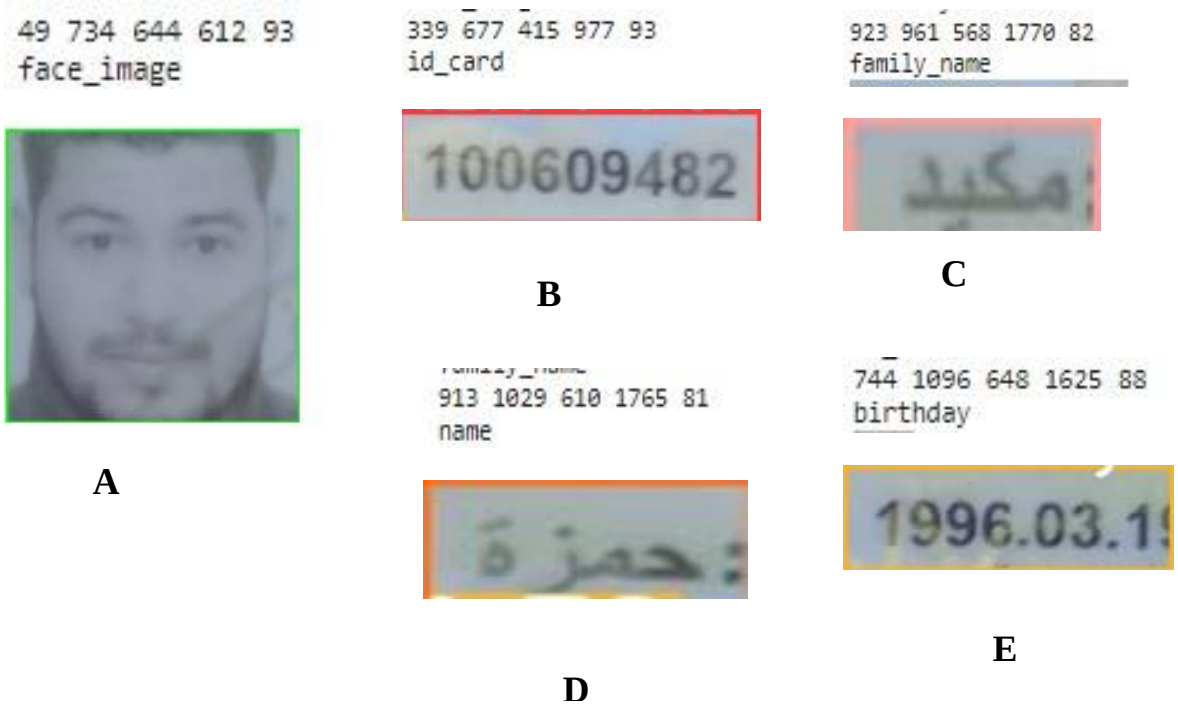


Figure IV.35 : illustration du résultat

2-4-2 Discussions sur YOLO

YOLO :

D'après les résultats obtenus, nous avons vu que la performance du modèle YOLO dans la détection des différents objets dans l'image est plus précis surtout dans les classes avec un mAP plus que 90%. Aussi nous avons vu que malgré la précision de l'objet 'card' dans le résultat de train c'est 73%, le test donne un bon résultat de détection quel que soit la position de la carte (de près ou de loin).

Points forts :

- La Plupart des boites englobantes cadrent bien les objets avec une précision (IoU) élevée (figure IV.26, figure IV.27).
- Le Système est puissant dans une image de complexité élevée, mauvaise qualité, mauvaise éclairage (figure IV.26), occlusion d'objet et la taille.
- L'entraînement de YOLO est simple, il peut être réalisé très facilement, Il suffit de spécifier les datasets la configuration et les poids
- YOLO est rapide à tester sur pas mal de photos et le niveau de confiance est vraiment correcte

Les problèmes :

- Problème de limite de l'utilisation de gpu.

- Chaque fois il y'a initialisation des fichiers des poids sur YOLOv5 à cause de la politique de Colab (historique n'est pas sauvegardé).
- Si on lance le train pour une deuxième fois ,il ne termine pas l'apprentissage sur le dernier arrêt.
- Le Problème des bases de données est qu'elles soient trop grandes ou elles présentent un problème d'étiquetage.
- L'Apprentissage occupe beaucoup de temps (plusieurs itérations = des semaines)

3- Architecture de la conception du Système d'identification

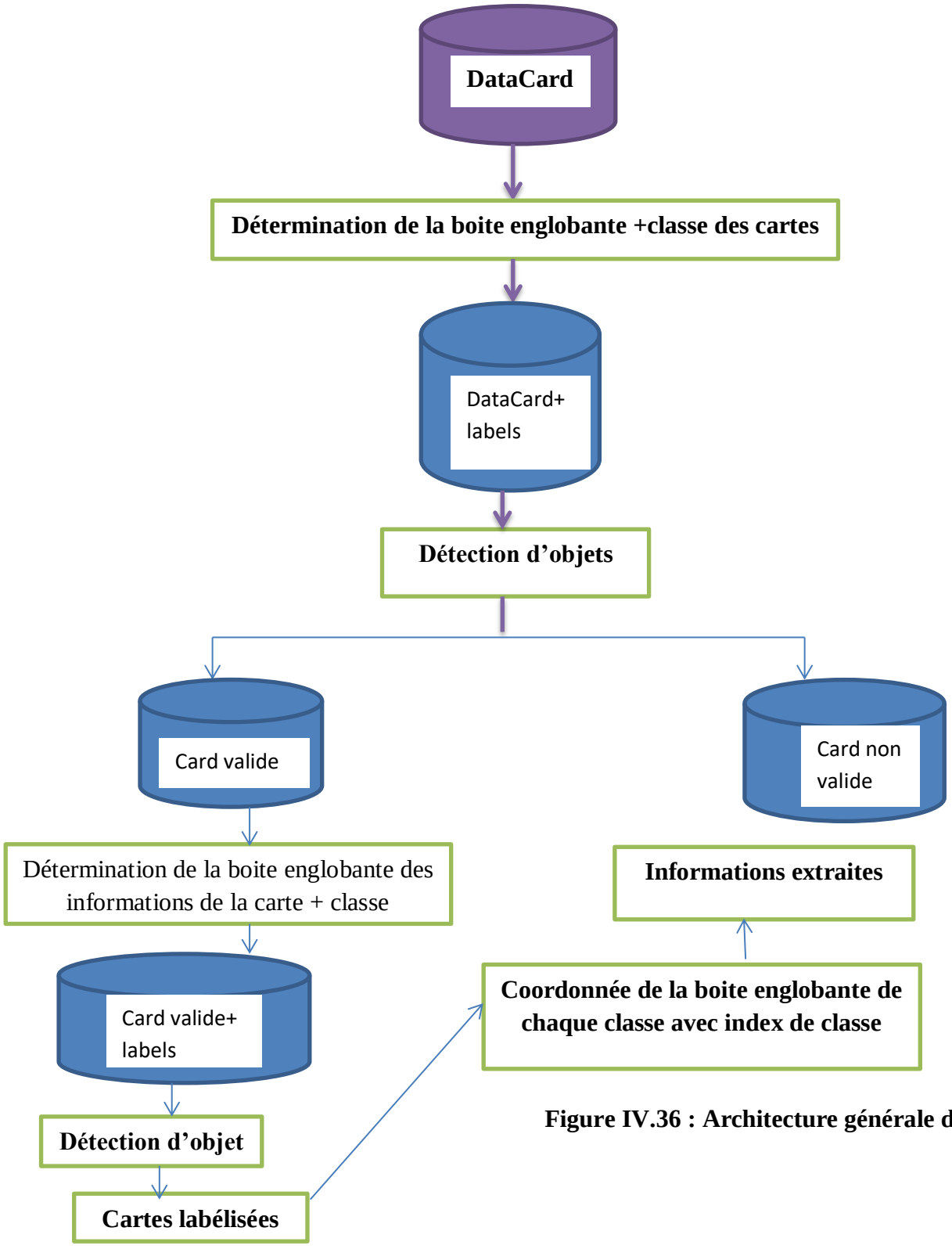


Figure IV.36 : Architecture générale du système

3-1 L'architecture de la conception générale du modèle (1):

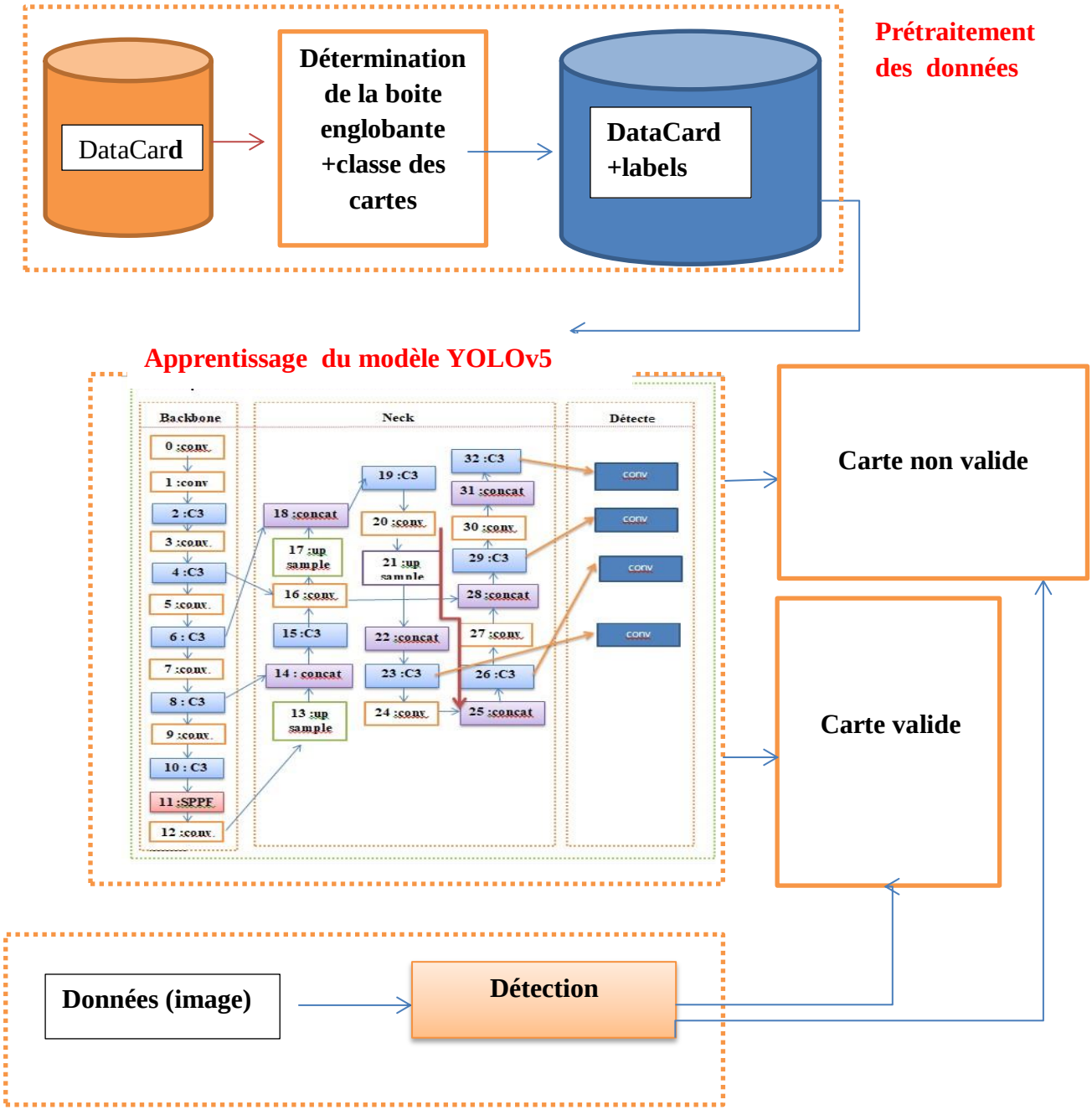


Figure IV.37 : Architecture du système d'authentification des documents

3-2 L'architecture de conception générale du modèle (2) :

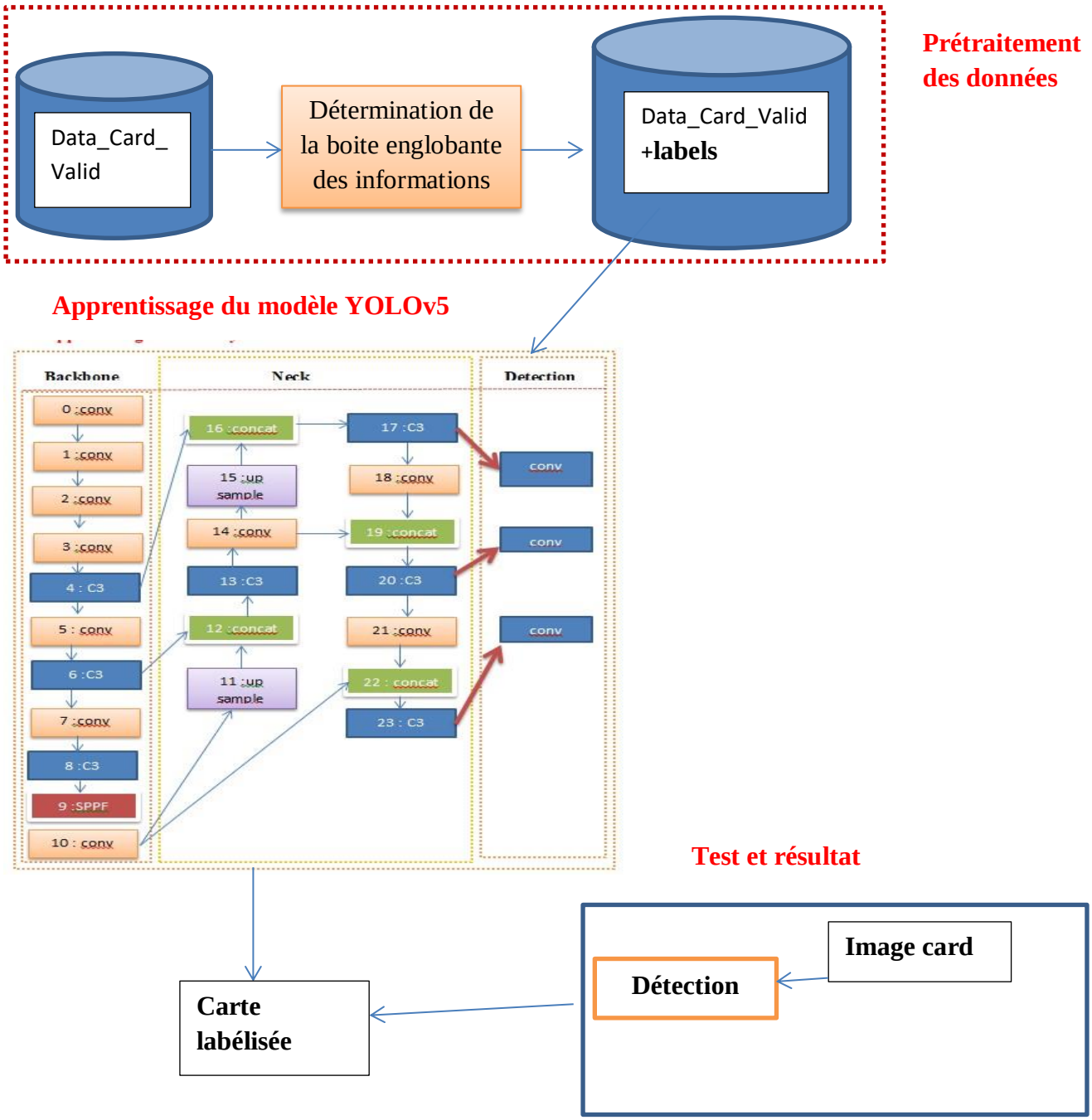


Figure IV.38 : Architecture générale de l'extraction des informations (2)

3-3 Prétraitement de donnés (1) (2)

La plupart des plates-formes d'annotation (ex : makesense.ia) prennent en charge l'exportation au format d'étiquetage YOLO, fournissant un fichier texte d'annotations par image. Chaque fichier texte contient une annotation de boîte englobante (BBox) pour chacun des objets de l'image. Les annotations sont normalisées à la taille de l'image et se situent dans la plage de 0 à 1. Elles sont représentées au format suivant :

< object-class-ID> <X center> <Y center> <Box width> <Box height>

S'il y a des objets dans l'image, le contenu du fichier texte des annotations YOLO pourrait ressembler à ceci :

```
0 0.384309 0.435106 0.170745 0.053191
1 0.787234 0.694681 0.054255 0.048936
2 0.780851 0.774468 0.070213 0.055319
3 0.701862 0.853191 0.142021 0.063830
4 0.189628 0.705319 0.247340 0.431915
```

Figure IV.39 : Exemple d'annotation nc=5

1. **Structure des répertoires de données**

Pour se conformer à la structure des répertoires d'Ultralytics, les données sont fournies selon la structure suivante :

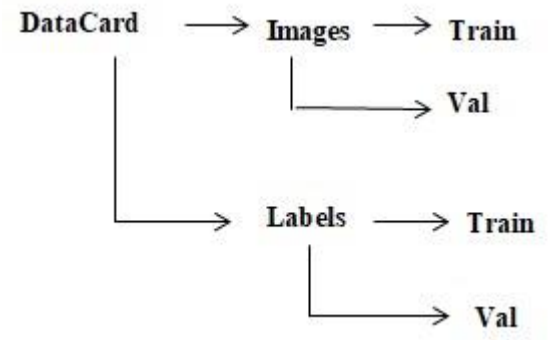


Figure IV.40 : Data directories structure (Datacard)

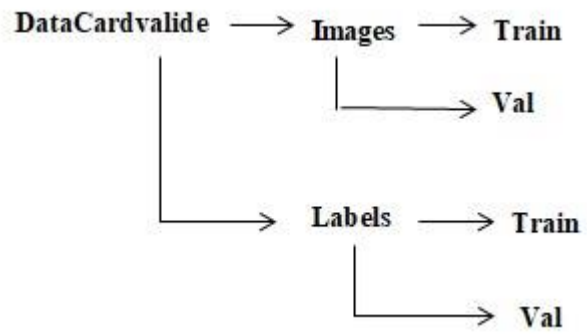


Figure IV.41 : Data directories structure (Datacardvalide)

2. Le fichier de configuration des données

Décrit les paramètres du jeu de données. Étant donné que nous nous entraînons sur notre ensemble de données personnalisé sur les images, nous allons modifier ce fichier et fournir : les chemins vers les ensembles de données d'entraînement, de validation et de test (facultatif) ; le nombre de classes (nc) ; et les noms des classes dans le même ordre que leur index.

cards.yaml

```yaml
path: /content/drive/MyDrive/New_cards_data_2 # dataset root dir
train: images/train  # train images (relative to 'path') 128 images
val: images/valid  # val images (relative to 'path') 128 images
test:  # test images (optional)

# Classes
names:
  0: card
  1: not_card
```

Figure IV.42 : Fichier Yaml pour la premier base de donné

cart_info.yaml

```yaml
path: /content/drive/MyDrive/data2  # dataset root dir
train: images/train  # train images (relative to 'path') 128 images
val: images/valid  # val images (relative to 'path') 128 images
test:  # test images (optional)

# Classes
names:
  0: id_card
  1: family_name
  2: name
  3: birth_day
  4: face_image
```

Figure IV.43 : Fichier Yaml pour la deuxième base de donné

3-4 Apprentissage de yolov5 (modèle 1et2)

YOLOv5 est basé sur l'architecture de détection YOLO et utilise l'excellent stratégie d'optimisation d'algorithmes dans le domaine des réseaux de neurones convolutifs ces dernières années, tels que les ancres de boîte englobante d'apprentissage automatique, l'augmentation des données de mosaïque, le cross-stage réseau partiel, et ainsi de suite; ils sont responsables de différentes fonctions dans différents endroits de l'architecture YOLOv5. Dans l'architecture, YOLOv5 se compose de quatre parties principales : entrée,colonne vertébrale, cou et sortie. Le terminal d'entrée contient principalement le prétraitement de les données, y compris l'augmentation des données en mosaïque et le remplissage adaptatif de l'image. En ordre pour s'adapter à différents ensembles de données, YOLOv5 intègre le calcul de cadre d'ancrage adaptatif sur l'entrée, afin qu'il puisse automatiquement définir la taille initiale du cadre d'ancrage lorsque l'ensemble de données changements. Le réseau dorsal utilise principalement un réseau partiel inter-étages (CSP) et mise en commun pyramidale spatiale (SPP) pour extraire des cartes d'entités de différentes tailles à partir de l'entréeimage par convolution multiple et mise en commun. BottleneckCSP est utilisé pour réduire la quantité de calcul et augmenter la vitesse d'inférence, tandis que la structure SPP réalise la fonctionnalité extraction à partir d'échelles différentes pour la même carte d'entités, et peut générer trois échelles cartes d'entités, ce qui contribue à améliorer la précision de la détection. Dans le réseau du cou, la caractéristique des structures pyramidales de FPN et PAN sont utilisées. La structure FPN transmet de fortes caractéristiques sémantiques des cartes de caractéristiques supérieures dans les cartes de caractéristiques inférieures. Au même temps, la structure PAN transmet de fortes caractéristiques de localisation à partir de cartes de caractéristiques inférieures dans des cartes de caractéristiques supérieures. Ces deux structures renforcent conjointement la caractéristique extraite de différentes couches réseau dans Backbone fusion, ce qui améliore encore la détection aptitude. En tant qu'étape de détection finale, la sortie de la tête est principalement utilisée pour prédire les cibles de tailles différentes sur les cartes d'entités. **[48]**

Le YOLOv5 se compose de quatre architectures, nommées YOLOv5s, YOLOv5m, YOLOv5l et YOLOv5x. La principale différence entre eux réside dans le nombre de modules

d'extraction de caractéristiques et de noyaux de convolution à des emplacements spécifiques sur le réseau.

Dans le modèle 1 on a utilisé l'architecture yolov5l6 on a obtenu cette structure

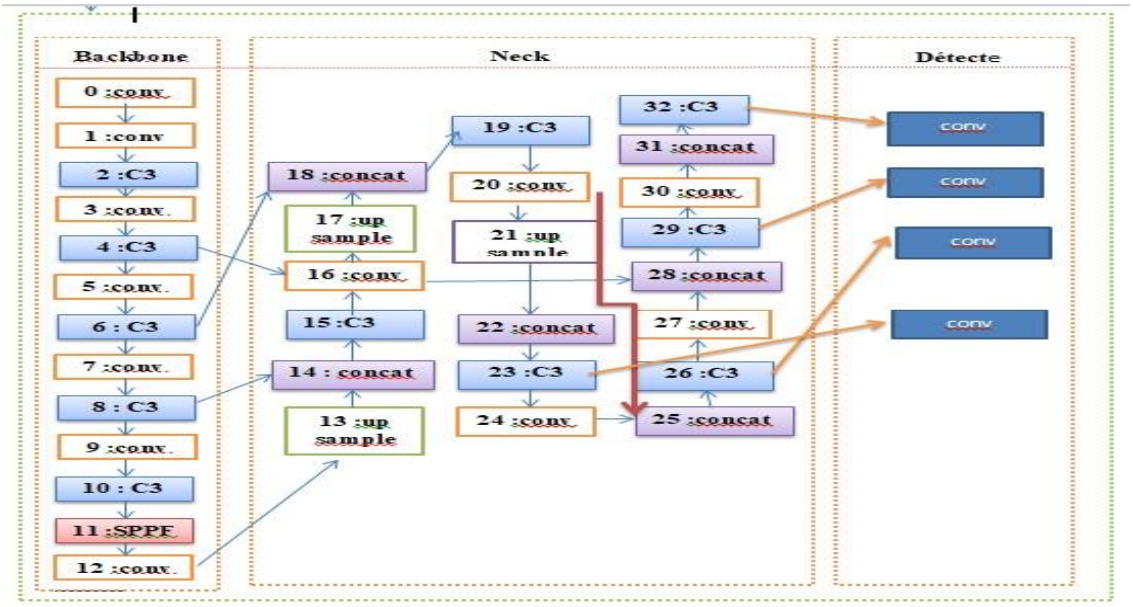


Figure IV.44: Architecture du YOLOv5l6

Dans le modèle 2 on a utilisé l'architecture YOLOv5s et on a obtenu cette structure

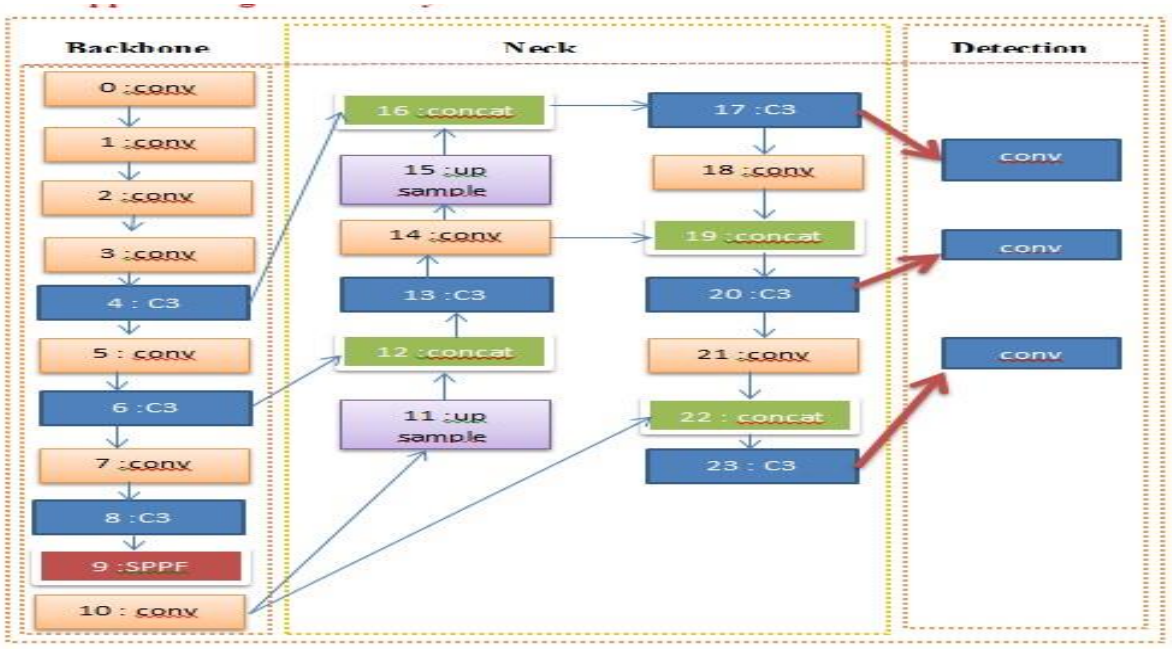


Figure IV.45 : Architecture du YOLOv5l6

II Implémentation

4-Matériels et outils

Pour développer notre système nous avons utilisé le matériel et le logiciel suivants :

4-1 Matériel utilisé

Nous avons travaillé sur un ordinateur personnel, les informations suivantes représentes la configuration du matériel utilisé.

Processeur: Intel(R) Core (TM) i5-6200U CPU @ 2.30GHz 2.40 GHz

RAM: 8,00 Go

Type de système : Système d'exploitation 64 bits, processeur x64

4-2 Outils de développement

Pour la mise en œuvre des différentes étapes discutées, nous avons opté pour les outils répandus, des technologies reconnues et des versions stables. Dans l'ensemble, nous nous sommes basés sur l'utilisation des technologies, nous permettant de trainer, d'implémenter et de tester nous modèle « yolov5 ».

4-2-1 Google Cloab :

Google Cloab est un service cloud, offert par **Google** (gratuit), basé sur Jupyter Notebook. Cette plateforme permet d'entraîner des modèles de Machine Learning directement dans le cloud et le tout est gratuit. Certains des avantages de **Cloab** incluent une installation facile et le partage des Notebooks entre utilisateurs en temps réel (comme les autres documents de la suite G-cloud ex. G-sheet). Par contre, charger un fichier csv nécessite d'écrire du code.[44]

Figure IV.46 : logo de Google Colab [44]

4-2-2 Python

Python est un langage de programmation de haut niveau Crée par Guido van Rossum et sorti en1991, on l'a choisi parce que il a obtenu un grand succès dans le domaine de Machine Learning grâce aux développeurs qui ont développé de nombreuses bibliothèques, ce qui rend l'utilisation de ce langage facile.

Figure IV.47 : Logo du Python

4-2-3 OpenCv

OpenCV (Open Source Computer Vision Library) est une bibliothèque de logiciels open source de vision par ordinateur et d'apprentissage automatique. OpenCV a été conçu pour fournir une infrastructure commune pour les applications de vision par ordinateur et pour accélérer l'utilisation de la perception par machine dans les produits commerciaux. En tant que produit sous licence BSD, OpenCV permet aux entreprises d'utiliser et de modifier facilement le code**.**

La bibliothèque contient plus de 2500 algorithmes optimisés, qui comprennent un ensemble complet d'algorithmes de vision par ordinateur et d'apprentissage automatique classiques et à la pointe de la technologie. Ces algorithmes peuvent être utilisés pour détecter et reconnaître des visages, identifier des objets, classer des actions humaines dans des vidéos, suivre des mouvements de caméra, suivre des objets en mouvement, extraire des modèles 3D d'objets, produire des nuages de points 3D à partir de caméras stéréo, assembler des images pour produire une haute résolution image d'une scène entière, trouver des images similaires dans une base de données d'images, supprimer les yeux rouges des images prises au flash, suivre les mouvements des yeux, reconnaître le paysage et établir des marqueurs pour le superposer avec la réalité augmentée, etc. OpenCV compte plus de 47 000 utilisateurs communauté et nombre estimé de téléchargements dépassant 18 millions. La bibliothèque est largement utilisée dans les entreprises, les groupes de recherche et les organismes gouvernementaux. **[45]**

Figure IV.48 : logo OpenCv [45]

4-2-4 Pytorch

PyTorch est une bibliothèque logicielle Python open source d'apprentissage machine qui s'appuie sur Torch (en) développée par Meta. PyTorch permet d'effectuer les calculs tensoriels nécessaires notamment pour l'apprentissage profond « Deep Learning ». Ces calculs sont optimisés et effectués soit par le processeur « CPU », soit lorsque c'est possible par un

processeur graphique « GPU » supportant CUDA. Il est issu des équipes de recherche de Facebook, et avant cela de Ronan Collobert dans l'équipe de Samy Bengio à l'IDIAP.

Figure IV.49 : logo de PyTorch

4-2-5 PIL

Pillow est une bibliothèque d'imagerie Python « PIL », qui prend en charge l'ouverture, la manipulation et l'enregistrement d'images. La version actuelle identifie et lit un grand nombre de formats. La prise en charge de l'écriture est volontairement limitée aux formats d'échange et de présentation les plus couramment utilisés.

Figure IV.50 : logo de Pillow

4-2-6 Numpy

Numpy est un projet open source visant à permettre le calcul numérique avec python. Il s'agit d'une bibliothèque python qui fournit un Object de tableau multidimensionnel, divers objet dérivés (tels que des tableaux et des matrices masqués). [46]

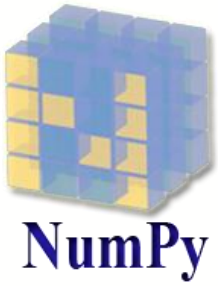


Figure IV.51 : logo NumPY[46]

4-2-7 matplotlib

Matplotlib est une bibliothèque Python open source, initialement développée par le neurobiologiste John Hunter en 2002. L'objectif était de visualiser les signaux électriques du cerveau de personnes épileptiques. Pour y parvenir, il souhaitait répliquer les fonctionnalités de création graphique de MATLAB avec Python

Figure IV.52 :logo matplotlib

4-2-8 Google Drive

Google Drive ou Google Disque au Canada est un service de stockage et de partage de fichiers dans le cloud lancé par la société Google. Google Drive, qui regroupe Google Docs, Sheets, Slides et Drawings, est une suite bureautique permettant de modifier des documents, des feuilles de calcul, des présentations, des dessins, des formulaires, etc. Les utilisateurs peuvent rechercher les fichiers partagés publiquement sur Google Drive par l'entremise de moteurs de recherche Web.[47]

Figure IV.53: Logo du Google Drive [47]

4-2-9 makeSense

Est un outil en ligne gratuit pour étiqueter les photos. Grâce à l'utilisation d'un navigateur, il ne nécessite aucune installation compliquée- il suffit de visiter le site Web et vous êtes prêt à partir. Peu importe également le système d'exploitation sur lequel vous travaillez Il est parfait pour les petits projets d'apprentissage en profondeur de la vision par ordinateur, ce qui rend le processus de préparation d'un ensemble de données beaucoup plus facile et plus rapide. Les étiquettes préparées peuvent être téléchargées dans l'un des nombreux formats pris en charge. L'application a été écrite en TypeScript et elle est basée sur le duo React/Redux. [48]

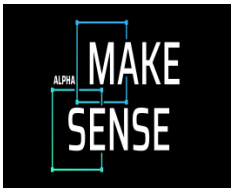


Figure IV.54 : Logo du MakeSense [48]

4-2-10 Fichier .Yaml

YAML est un langage de sérialisation des données qui est souvent utilisé pour coder des fichiers de configuration. Pour certains, YAML est l'acronyme de *Yet Another Markup Language*, pour d'autres, c'est l'acronyme récursif de *YAML Ain't Markup Language* (YAML n'est pas un langage de balisage), ce qui souligne que le langage YAML s'utilise pour représenter des données plutôt que des documents.

Conclusion

Nous avons présenté dans ce chapitre l'implémentation de l'approche de détection d'objet quelle est basée sur le Deep Learning, pour cela nous avons utilisé le réseau de neurones convolutifs (YOLOv5) qui a été pré-entrainer jusqu'à 607 couches de convolution pour le premier modèle et 270 pour le deuxième modèle pour que nous assurons une meilleure 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 vous voulez ou bien vous prouvez entraîner l'algorithme sur votre propre BDD.

Conclusion générale

Notre travail a pour but de concevoir et de réaliser un système d'authentification et de classification des cartes pour le compte de l'entreprise icosnet. Ce système permet aussi l'extraction des informations labélisées dans la carte nationale en utilisant le modèle de détection d'objets par les architectures CNN. Le modèle yolov5 a généré des résultats jugés positifs dans des environnements complexes. La précision est supérieure à 80% pour les deux modèles.

Dans le futur nous pensons améliorer notre système ; nous pouvons citer quelques pistes de recherche dont :

- Augmenter la base de données utilisée
- Ajouter plus de fonctionnalités (mobile et site web).
- Appliquer notre modèle sur autres bases de données contenant d'autres documents numériques.
- Ajouter des méthodes de reconnaissance faciale.

Références bibliographiques

Références bibliographiques

[1] réalisé par ASRI ROFAIDA et BOUZIANE NESRINE, « conception et développement d'une solution d'AI pour le contrôle et la gestion des services de la VAZI Box » , mémoire de master en informatique, , 09/09/2020, p111-112

[2]EL-Hachemi Guerrout, représentation d'une image numérique, « https://www.researchgate.net/figure/Representation-dune-image numerique23_fig1_344266838 » 2022/08

[3]Raphaël Isdant. « Traitement numérique de l'image ». In : (2009) (cf. p. 7, 8).

[4] M r Mokri Mohammed Zakaria, « Classification des images avec les réseaux de neurones convolutionnels », Mémoire de fin d'études Pour l'obtention du diplôme de Master en informatique, p3,et p10.

[5] David Legland ,image numérique, 27 Juin 2019 « https://www6.inrae.fr/pfl-cepia/Axe-images/Tutoriel/L-image-numerique » 2022/08/09

[6] Yann LeCun, Informatique et sciences numérique Microsoft Word - Intelligence Artificielle – Y. LeCun.docx (college-de-france.fr) 2022/08

[7] Feuilletage_520.pdf (dunod.com) 2022/08

[8] touraya alhassani, « You Only Look Once – un réseau de neurones pour la détection d'objets » 26,08,2021 **https://blog.octo.com/you-only-look-once-un-reseau-de-neurones-pour-la-detection dobjets/**

[9] Mr TOUAHRI Islam, Mémoire de Master Option : Système Informatiques, Détection et Classification des Véhicules par Réseaux de Neurones à Convolution CNN, , 2019/2020

[10] Rosebrock, Adrian. Deep Learning for Computer Vision with Python: Starter Bundle. PyImageSearch, (2017)

[11] Gilberto Batres-Estrada, Deep Learning for Multivariate Financial Time Series, June 4, 2015

[12] mémoire fin d'étude pour l'obtention de diplôme de master en informatique, thème « classification des images satellitaires pour l'aide à la gestion des catastrophes naturelles en

utilisant l'apprentissage profond », option « ingénierie des logiciels », Mr BRAHIM AIMEN et Mr NEBIH AKRAM

[13] Danqing Liu, A Practical Guide to ReLU, medium Nov 30, 2017

[14] Tom Keldenich, inside machine learning, https://inside-machinelearning.com, 22/08/02

[15] Avijet Biswal, simplilearn,11/08/22,https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn

[16]bismart,https://blog.bismart.com/en/difference-between-machine-learning-deep-learning 2022/09/02

[17] medhusher basavarajaiah, 90/03/2019, 6 basic things to know about Convolution

https://medium.com/@bdhuma/6-basic-things-to-know-about-convolution-daef5e1bc411

[18] **ManavMandal**, analytics vidhya, 23/07/2021, https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks cnn/#h2_3

[19] pierre, 13/01/2020, Classification de pages Web via Deep Learning – Réseau de Neurones Convolutif – Anakeyn

[20] LES RÉSEAUX DE NEURONES CONVOLUTIFS,17/04/2018,**Les réseaux de neurones convolutifs. (natural-solutions.eu)** 09/2022

[21] Gael, datakeen, 25/02/2018, **3 Algorithmes de Deep Learning expliqués en Langage Humain - Datakeen** 09/2022

[22] « Classification des images avec les réseaux de neurones convolutionnels ». Mémoire de fin d'études pour l'obtention du diplôme de Master en informatique. 03/07/2017

[23] benlahmar, datasciencetoday, 12/10/2018,DataScienceToday - Les réseaux de neurones convolutifs 09/2022

[24] « La Classification d'images d'insectes ravageurs en utilisant le Deep Learning » Mémoire de Fin d'Etudes En vue de l'obtention du diplôme de master en informatique 2019/2020

[25] Tanay Agrawal « Hyperparameter Optimization in Machine Learning » page 03 Bangalore, Karnataka, India 2021

[26] Arjun Panesar « Machine Learning and AI for Healthcare» page 74-75-76 Coventry, UK 2021

[27] projeduc, introduction à l'apprentissage automatique https://projeduc.github.io/intro_apprentissage_automatique/introduction.html 2022/09

[28] Sayantani Sanyal , « What is object detection?", https://www.analyticsinsight.net/object-detection-by-tensorflow-an-emerging transformative-trend/

[29] Utilisation de méthodes de Deep Learning pour l'extraction de texte dans les images, Mémoire de fin d'études pour l'obtention du diplôme de Master en informatique. 2020-2021

[30] MLK« 6 Different Types of Object Detection Algorithms in Nutshell», https://machinelearningknowledge.ai/different-types-of-object-detection-algorithms, dernier consultation 04/08/2021

[31]daphné lercier, Makine Blog, 02/06/2020 https://makina-corpus.com/sig-webmapping/extraction-dobjets-pour-la-cartographie-par-deep-learning-choix-du-modele#:~:text=R%2DCNN%20(Regional%20Convolutional%20Neural,susceptibles%20de%20contenir%20un%20objet. 09/2022

[32] S. Ren, K. He, R. Girshick, and J. Sun. « Faster R-CNN: Towards real-time object detection with region proposal networks ». dans Conférence NIPS, 2015

[33] Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu , Alexander C. Berg et Wei lue, springer link, https://link.springer.com/chapter/10.1007/978-3-319-46448-0_2

[34] yolov5 documentation, https://docs.ultralytics.com/ 09/2022

[35] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. «You only look once: Unified, realtime object detection ». Preprint sur arXiv:1506.02640, 2015

[36] jantakarn, 5/05/2020,Guide to Object Detection using YOLO https://medium.com/@aumjantakarn/guide-to-object-detection-using-yolo-33d74d7091d9

[37] AishwaryaSingh, analyticsvidhya,Selecting the Right Bounding Box Using Non-Max Suppression (with implementation) 03/08/2020 **https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/**

[38] Alexy Bochkovskiy, CSPDarknet53, 23/04/2020 **https://paperswithcode.com/method/cspdarknet53**

[39]similar answer, https://similaranswer.fr/quest-ce-que-la-couche-spp/ 09/2022

[40] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, Jiaya Jia, Path Aggregation Network for Instance Segmentation, 18/09/2018, https://arxiv.org/abs/1803.01534

[41]surya gulta, Object Detection Algorithm — YOLO v5 Architecture, 2/08/2021 https://medium.com/analytics-vidhya/object-detection-algorithm-yolo-v5architecture

[42] Renjie Xu,Haifeng Lin, Kangjie Lu, Lin Cao and Yunfei Liu , 13/02/2021 **"A Forest Fire Detection System Based on Ensemble Learning"** https://www.researchgate.net/figure/The-network-architecture-of-Yolov5-It consists-of-three-parts-1-Backbone-CSPDarknet_fig1_349299852

[43] jiatu wu, « complexity and accuracy analysis of common artificiel neural networks on pedestrian detection »,2018 https://www.researchgate.net/figure/YOLO-architecture-YOLO-architecture-is-inspired-by-GooLeNet-model-for-image_fig2_329038564

[44]Simon prudhomme, « 2 manières de télécharger un fichier csv dans Google Colab »,15-12-2019, https://medium.com/@simonprdhm/2-mani%C3%A8res-simple-de-charger-un-fichier csv-dans-google-colab-3b86616d248a

[45] Opencv, https://opencv.org/about/ 20/09/2022

[46] java T point, Python NumPy Tutorial https://www.javatpoint.com/numpy-tutorial 20/09/2022

[47] wikipedia https://fr.wikipedia.org/wiki/Google_Drive 20/09/2022

[48]_Zhuang Li, Xincheng Tian , Xin Liu , Yan Liu and Xiaorui Shi , article "A Two-Stage Industrial Defect Detection Framework Based on Improved-YOLOv5 and Optimized-Inception-ResnetV2 Models" p3-4

Annexe

1-présentation de l'organisme d'accueil

Nous commençons notre projet en parlant et en présentant l'organisme d'accueil de la société d'ICOSNET, et ses déférents domaines et activités et son organisme interne

1-1 Présentation d'ICOSNET

ICOSNET SPA (Société Par Action) est une société privée fondée en 1999, se positionne comme un opérateur d'accès internet et de solutions de télécommunications. Elle s'impose aujourd'hui sur le marché de la convergence voix et données pour les PME/PMI et les grands comptes multinationaux installés en Algérie, son siège social est au Centre des Affaires El Qods, 6éme niveau de la tour Centrale, Cheraga, Algérie.

Figure A.1 : Logo d'ICOSNET SPA (ICOSNET) **[1]**

Sur le marché algérien, ICOSNET est un opérateur à part entière. Ce positionnement permet de s'adresser à une clientèle large, de convaincre des clients de taille significative et de pouvoir proposer des solutions de connexion et de communication économiquement plus avantageuses et plus abouties, avec ses solutions dans les différents domaines de la technologie :

- Hébergement.
- Internet des objets.
- Data center.
- Impression numérique.
- Internet haut débit et accès spécialisé.
- Control d'accès.

Le réseau d'ICOSNET aujourd'hui couvre plusieurs entreprises Algériennes et grands groupes internationaux et elle a des ambitions pour une implantation sur le territoire national, comme nous montrons dans la **figure A.2**

Figure A.2 : Réseau de service ICOSNET (ICOSNET)**[1]**

1-2 Organisme interne d'ICOSNET

La culture de l'innovation est mise en évidence par l'organisation Interne de l'entreprise qui se compte de 3 directions d'ingénierie des services par métier et une division de développement qui travaillent de concert pour la conception et la mise au point de nouvelles offres. Elle se compose de trois organes principaux : **.[1]**

- La structure commerciale.
- La structure administrative.
- La structure technique.