الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université 8 Mai 1945 –Guelma-

Faculté des Mathématique, d'Informatique et des Sciences de la Matière

Département d'Informatique

Mémoire de Fin d'Etudes Master

Filière : Informatique

Option : Sciences et Technologie de l'Information et de la Communication

Thème

Détection d'objets par Deep Neural Network à l'aide du modèle YOLO en temps réel.

Encadré par :

Dr. Hallaci Samir

Présenté par :

Mesbah Fethia

ANNÉE UNIVERSITAIRE: 2020/2021

Dédicaces

Je dédie ce travail, Fruit de nombreuses années d'étude à :

à mes chers parents pour leur patience,

Ma chère mère. Merci pour tes conseils, tes Sacrifices, ton soutien et tes encouragements

Papa ma Gratitude ne suffit pas à exprimer ce qu'il mérite pour

tout ses sacrifices depuis ma naissance,

pendant mon enfance et même à l'âge adulte

(Que Dieu lui fasse miséricorde et l'accueille dans son vaste paradis)

À mes frères Saber, Abdenoure, Abderahime

À Tous mes enseignants.

À tous mes amis et spécialement Ibtissem, Ahlem, Assia, Zaim, Abd el bacet, Manel, Hassiba qui m'ont soutenu dans l'accomplissement de cet Humble travail

À tous mes professeurs et à tous ceux qui se sont engagés

dans ces modestes travaux

À tout ma famille Mesbah mes tantes.

Et À tous qui m'ont aide de près ou de loin pour la réalisation de ce travail.

Fethia Mesbah

Remerciement

En premier lieu, nous remercions Dieu le très haut

qui nous a donné le courage et la volonté

de réaliser ce modeste travail.

Nous tenons à saisir cette occasion et adresser nos profonds remerciements

et nos profondes reconnaissances à toutes personnes

qui nous ont aidé de près ou de loin dans la réalisation de ce mémoire.

Nous remercions Mr. Hallaci Samir, en tant que Directeur de mémoire,

pour ses précieux conseils et son orientation tout au long de notre recherche.

Nos remerciements aux membres de jury qui ont accepté de juger notre travail.

Enfin nous exprimons notre profonde reconnaissance à tous responsables et enseignants de de départements d'informatique université de Guelma

qui ont contribué à notre formation.

Fethia Mesbah

Résumé

Aujourd'hui, en raison du développement continu des capacités informatiques et de la grande disponibilité des données, l'apprentissage en profondeur est utilisé dans de nombreux domaines, en particulier la vision par ordinateur. Grâce aux algorithmes d'apprentissage profond, le domaine de la détection d'objets, comme l'une des applications de vision par ordinateur, a connu une révolution complète. Ces applications sont basées sur les réseaux de neurones convolutifs CNN. Parce que les modèles de détection d'objets en temps réel ont connu une révolution très rapide, ils sont en termes de la vitesse de détection et de précision. Le but de ce travail est de rechercher et d'appliquer des algorithmes de détection d'objets d'apprentissage en profondeur pour aider les personnes aveugles en détectant les objets présents dans la maison.

Nous nous sommes concentrés dans cette étude sur l'algorithme de détection d'objet YOLO, qui est une méthode pratiquement reconnue et approuvée.

Par conséquent, nous avons essayé de ré-entraîner le model YOLO sur des sous classes de la base de données PASCAL VOC et sur la base de données ExDark qui est une base réalisée sous un environnement à faible luminosité, ou la majorité des images sont prise la nuit. Nous avons obtenu des résultats très satisfaisants avec un taux de précision 68%, avec un aperçu qui reflète la grande supériorité de l'algorithme de détection d'objet YOLO en temps réel sur beaucoup d'autres modèles.

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

Table des matières

Table des matières

Table des matières

Liste des Figures

Chapitre 01 : état de l'art

Chapitre 02 : les modèles de détection

Liste des Figures

Chapitre 03 : conception et implémentation

Liste des tableaux

Chapitre 01 : état de l'art

Liste des tableaux

Chapitre 02 : les modèles de détection

Chapitre 03 : conception et implémentation

Introduction générale

Aujourd'hui nous avons tous remarqué l'énorme développement des capacités logicielles, informatiques, théorique et logistiques au cours de la dernière décennie, ainsi le développement des algorithmes de l'intelligence artificielle dans plusieurs domaines de l'informatique tels que : la vision par ordinateur, la vidéosurveillance, les voiture autonomes, la robotique, l'aéronautique, des maisons intelligentes, des cités intelligentes; etc. étant donné ce développement est très rapidement et même spectaculaire, et nous ne faisons que commencer.

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

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

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

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

Donc la disponibilité des ressources matériels sur le cloud, ainsi la disponibilité des bases de données de qualité, et l'émergence des modes et des architectures de deep Learning basés CNN plus robustes, nous a poussé a créé un système intelligent qui aide les personnes aveugles à reconnaitre les objets existent dans la maison afin de les protéger, et pour réaliser ce

système nous avons choisi le model YOLO qui a prouver ses performances dans la détection d'objet en temps réel, donc on a ré-entraîner ce model sur des classes bien sélectionnées à partir des deux bases de données PASCAL VOC et ExDark. Cela nous a permis d'obtenir un système rapide et efficace.

Ce mémoire se présente sous forme de trois chapitres :

Le premier chapitre : donne une présentation générale sur l'intelligence artificiel en présentant les méthodes de l'apprentissage automatique et les principes fondamentale de deep learning avec les benchmarks base de données dans le domaine de la reconnaissance d'objets.

Ensuite Le chapitre 2 : 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 chapitre 3 : nous présentons la conception de notre système de détection avec les résultats expérimentaux obtenus par le modèle YOLOv4 ainsi que des commentaires avec interprétations des résultats.

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

Chapitre 01 : Introduction à L’IA et Deep Learning

Chapitre 01 : Introduction à L'IA et Deep Learning

1. Introduction :

L'intelligence artificielle est un grand domaine de recherche, il a connu une révolution très rapide dans les dernières années grâce au développement des matérielles de traitement de données. Parmi les domaines de recherche très connus dans IA on a le « deep learning » ou bien l'apprentissage au profond dans le domaine du traitement d'images, grâce à ces algorithmes, elle est possible d'analyser l'image, détecter et reconnaître les objets dans l'image.

Détecter les objets dans l'image est parmi les problèmes sur lequel les chercheurs travaillent pour créer des modèles qui donne un bon résultat de détection d'objet (des modèles plus précis et plus rapides « compromis précision/temps »).

Pour une meilleure utilisation des modèles de deep learning pour la détection d'objets, certains conditions techniques, théoriques et logistiques doivent être disponible, comme des machines qui Fournissent des GPUs et de l'espace de stockage parce que ces techniques ont besoin de faire des calculs énormes lors de l'apprentissage sur les grandes quantités de données (Big data) de façon minutieuse et profonde.

Dans ce chapitre on va voir d'une façon générale l'historique de l'intelligence artificiel, ainsi que nous allons ce concentrer sur l'apprentissage automatique et deep learning, d'abord on va voir :

- Pour quoi la détection et le suivi d'objets dans l'image ?
- L'histoire et l'évolution de l'intelligence artificielle, de l'apprentissage automatique et de l'apprentissage profond.
- Les concepts fondamentaux de Deep Learning (DL) et l'apprentissage automatique (Machine learning (ML))avec les modèles de détection et classification d'objets.
- Les déférentes base de données utilisés dans la phase d'apprentissage des modèles.

2. Pour quoi la détection d'objets :

La détection d'objet est un domaine très vaste et très important dans la recherche, parce que les recherches actuelles visent à crée des systèmes qui se rapproches des compétences de l'être humain dans la perception et le suivi et la reconnaissance d'objet. L'importance de la détection vienne du fait que le bon résultat dans cette phase donne un bon résultat de la reconnaissance et aussi cette phase n'est pas facile à traiter à cause de plusieurs problèmes tels que la taille d'objet

détecter, la lumière, la forme, la grande variété des objets, la vitesse de réponse en temps réel, la complexité des background… etc.

La détection d'objet elle est parmi les application pratique les plus intéressantes dans la vie courante, par exemple elle est utilisée dans la surveillance du trafic ou bien pour un véhicule avec une assistance de conduite automatique ou partiellement autonome, la détection des personnes qui ne porte pas des masques pendent la situation du « COVID -19 » , cette dernière est utilisé dans les entreprise pour détecter les produits bien fait et les mal fait (par exemple :entreprise qui produit des pièces donc le système va vérifier la qualité).

3. La détection et le suivi d'objets :

La détection d'objet dans une image consiste deux éléments principaux :

1. Tracer un cadre (Bounding boxes) au tour des différents objets pour positionner chaque objet dans l'image, cela veut dire « détection ».
2. Attacher des labels aux objets pour identifier les classes des objets détectés au fils du flux d'images « le suivi ».

Les méthodes les plus récentes peuvent être classées en deux catégories principales : les méthodes à une étape et les méthodes à deux étapes :

- ✓ Les méthodes à une étape donnent la priorité à la vitesse d'inférence. Parmi les exemples de modèles, citons YOLO, SSD et RetinaNet.
- ✓ Les méthodes à deux étapes donnent la priorité à la précision de la détection et les modèles d'exemple comprennent Faster R-CNN, Mask R-CNN et Cascade R-CNN.[1]

La détection est souvent attachée au suivi, d'où ce dernier consiste à prendre un ensemble initial de détections d'objets, à créer un identifiant unique pour chacune des détections initiales, puis à suivre chacun des objets lorsqu'ils se déplacent dans les images d'une vidéo en maintenant l'attribution de l'identifiant [2].

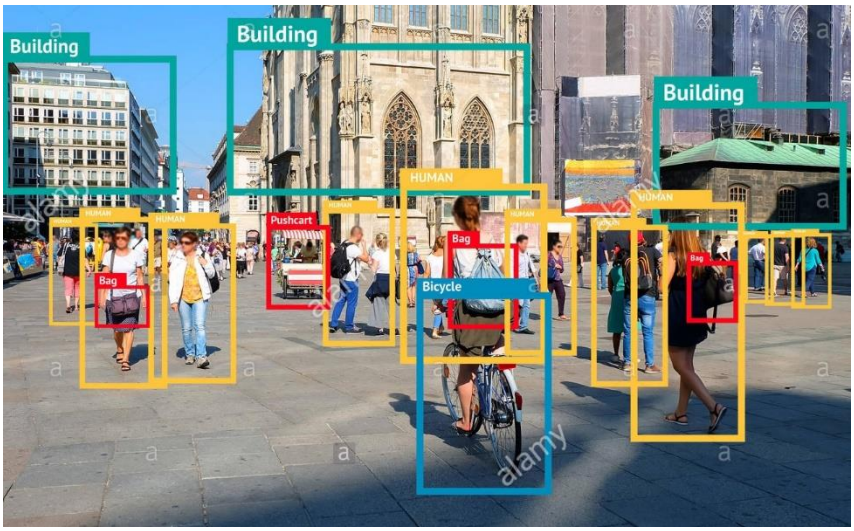


Figure 1.1 : Exemple de détection [27].

4. Les Notions fondamentales :

4.1. Intelligence Artificielle :

L'intelligence artificielle est l'étude de la manière dont les ordinateurs peuvent effectuer des tâches intelligentes qu'est dans le passé, ne pouvaient être réalisées que par des humains. [3]

Autre définition dit que L'intelligence artificielle est un ensemble de plusieurs technologies et théories informatiques et aussi un Domaines d'intérêt pour la pensée. la logique et l'intelligence visent à concevoir des programmes capables de résoudre des problèmes et de traiter le langage ainsi que d'exécuter autres tâches qu'était exclusive à l'être humain [4].

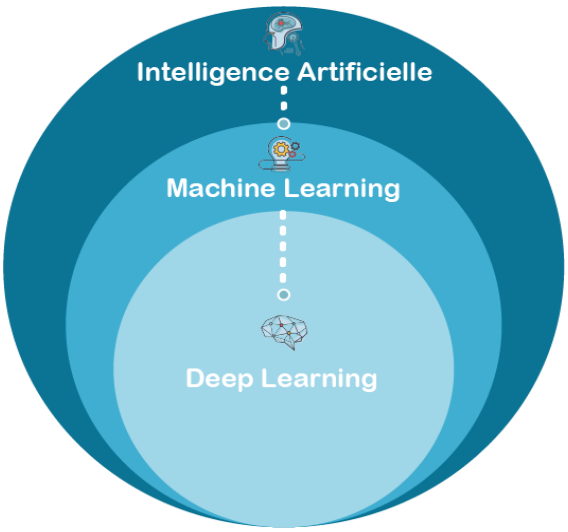


Figure 1.2 : La relation entre IA et ML et DL[28]

4.1.1. Historique de l'intelligence artificielle(IA) :

a. La 1ère période 1950-1970 :

Résolution de problèmes triviaux, aucun aspect pratique, GOFAI - La bonne vieille IA [10] :

Année	Événement
1942	Les 3 lois de la robotique par Isaac Asimov, D'autres séries de lois ont été proposées par des chercheurs. Depuis lors
1950	Le test de Turing propose par Alan Turing
1952	Le premier programme de jeu d'auto-apprentissage
1955	Le premier système d'IA, appelé Logic Theorist, a été conçu par Allen Newell et Herbert A. Simon, et mis en œuvre par J. Clifford Shaw.
1956	Conférence de Dartmouth, première utilisation des termes "Intelligence Artificielle/IA".
1957	- General Problem Solver (GPS) de Newell - Introduction de la première version des grammaires génératives.
1958	McCarthy développe le langage de programmation LISP
1959	- Le laboratoire d'IA du MIT (McCarthy et Minsky). - Le terme "apprentissage automatique" de Samuel
1961	- Premier robot industriel (Unimate) travaillant chez GM - SAINT, le premier système expert par Slagle (MIT).
1964	"STUDENT" le premier programme d'IA qui comprend le langage naturel.
1965	"ELIZA" le premier Chatbot et système expert basé sur l'IA
1966	- "Shakey" la première locomotive et le premier robot intelligent (SRI) - MAC HACK", programme de jeu d'échecs de Greenblatt, MIT.
1968	"SHRDLU", un premier programme informatique de compréhension du langage naturel.
1970	"WABOT-1", le premier robot anthropomorphe (Université de Waseda).
1972	"Prolog" langage de programmation logique
1973	- "Lighthill Report" le mauvais rapport d'avancement a provoqué le "First AI winter" qui est Réduit le financement de la recherche sur l'IA - le premier système qui comprenait un langage naturel dans le contexte d'un environnement similaire à un monde simplifié.
1974	- MYCIN", le premier système expert d'IA basé sur des règles pour les diagnostics médicaux. - le premier véhicule autonome, un "slider" mécanique (Stanford)

Tableau 1.1 : Première période de l'évolution de l'IA, ML et DL 1950-1970

b. La 2ème période 1980-2000:

Les chercheurs alimentent les machines en données étiquetées, Projets : ICOT - Japon '82, MCC - US '83, Alvey - UK '84. Et les algorithmes ont commencé à apparaître comme des éléments de systèmes plus vastes. Les solutions d'IA se sont avérées

utiles dans l'ensemble de l'industrie technologique, comme l'exploration de données, la robotique industrielle, etc. [10]

Année	Événement
1980	- Développement et commercialisation de machines basées sur LISP - "INTERNIST-1" Le premier système expert commercial.
1986	A driverless van by Mercedes-Benz, with cameras and sensors
1988	- Les "réseaux bayésiens", BN ou réseaux de croyance, inventés par Pearl. - Les chatbots, "Jabberwachy" et "Cleverbot", inventés par Carpenter.
1989	- Le premier véhicule autonome créé par la CMU à l'aide d'un réseau neuronal
1993	"Polly, le robot guide touristique, robotique basée sur le comportement (MIT)
1997	Deep Blue d'IBM bat Gary Kasparov aux échecs
1998	"Furby", le premier robot-animal de compagnie pour enfants
1999	- L'IA émotionnelle "Kismet" (MIT AI Lab) - AIBO est le premier robot domestique intelligent de Sony.
2000	Honda lance le robot humanoïde "ASIMO".
2002	i-Robot lance l'aspirateur robot autonome "Roomba".
2004	- Le premier défi pour les véhicules autonomes par la DARPA - Les rovers de la NASA "Spirit" et "Opportunity" explorent Mars
2005	Moteurs de recommandation basés sur l'IA
2006	"Lecture automatique" : compréhension autonome non supervisée de textes.
2007	- ImageNet", base de données visuelle pour la recherche de logiciels de reconnaissance d'objets. - CUDA, lancé par NVIDIA, est une plate-forme de calcul parallèle et une interface de programmation.
2009	- La voiture autonome construite par Google a réussi le test de conduite autonome du Nevada en 2014. - Des chercheurs en IA découvrent le GPU (Graphics Processing Unit) pour DL

Tableau 1.2 : Deuxième période de l'évolution de l'IA, de la ML et de la DL 1980-2000

c. La 3ème période 2010-2021:

L'ère de l'apprentissage automatique les ordinateurs acquièrent des connaissances a partir des données, et non plus à partir des humains. Les grandes entreprises technologiques investissent dans les applications commerciales de l'IA/ML.[10]

Année	Événement
2010	- Démocratiser l'accès aux données commence pour la reconnaissance d'images - L'IA de Narrative Science démontre sa capacité à rédiger des rapports
2011	- Apple a lancé "Siri" - Le "Watson" d'IBM remporte l'affrontement à Jeopardy
2013	- NEIL, un système d'analyse sémantique d'images de l'Université de Californie du Nord (CMU). "Vicarious" passe le premier test de Turing - CAPTCHA
2014	- Cortana de Microsoft "Alexa" d'Amazon
2015	- TensorFlow" de Google Brain, une bibliothèque ML (TPU) - "Open AI", une initiative à code source ouvert visant à développer l'IA au profit de l'humanité tout entière.
2016	- "Google Home" par Google - Alpha Go" : le Deepmind de Google a battu le champion n°1 du jeu de Go. - NVIDIA annonce un supercalculateur pour la DL et l'IA - Le robot humanoïde "Sophia" de Hanson Robotics, le premier robot citoyen - "PyTorch, une bibliothèque ML open source
2017	- Le laboratoire de recherche en IA de facebook a entraîné deux chatbots à communiquer entre eux afin d'apprendre à négocier ; les chatbots se sont écartés du langage humain et ont inventé leur propre langage pour communiquer entre eux. - "Caffe" Cadre DL open source.
2018	- BERT" de Google, la première représentation bidirectionnelle non supervisée du langage. - "Bixby" introduit par Samsung - Facebook détecte les visages et partage les photos avec les amis à qui ces photos appartiennent - L'IA de traitement du langage d'Alibaba a surpassé l'intelligence humaine lors d'un test de lecture et de compréhension de Standford
2020	- L'équipe DeepMind utilise les algorithmes DL "Agent 57" qui surpasse les humains aux jeux Atari avec l'apprentissage par renforcement profond. - Déploiements généralisés de réseaux "5G" dans le monde entier

Tableau 1.3 : Troisième période de l'évolution de l'IA, de la ML et de la DL 2010-2020

4.2. Apprentissage Automatique :

L'apprentissage automatique est un domaine d'étude interdisciplinaire qui rassemble des techniques issues de l'informatique, des statistiques, des mathématiques et des sciences cognitives, dont la biologie, la psychologie et la linguistique, pour n'en citer que quelques-unes.

Si l'idée d'apprendre à partir des données existe dans le milieu universitaire depuis plusieurs décennies, son entrée dans l'industrie technologique grand public a commencé au

début des années 2000. Cette croissance a coïncidé avec l'augmentation des données volumineuses résultant de l'explosion du Web, les gens ayant commencé à partager des données sur l'Internet.[5]

On distingue usuellement au moins trois types d'apprentissage machine : l'apprentissage par renforcement, l'apprentissage supervisé et l'apprentissage non supervisé. :

4.2.1.Apprentissage supervisé :

Supposant on donne des exemples étiquetés, comme des images de lettres manuscrites avec le nom de la lettre correspondante (étiquettes *a*, *b*, *Z*…). L'apprentissage consiste alors à construire une fonction capable de déterminer la lettre de l'alphabet à laquelle se rapporte chaque image. Cette forme d'apprentissage a fait des progrès considérables ces dernières années.[6]

4.2.2.Apprentissage non supervisé :

Lorsque les gens font référence à des systèmes capables d'apprendre par eux-mêmes, ils font référence à l'apprentissage non supervisé. Dans l'apprentissage non supervisé, l'algorithme d'apprentissage ne reçoit pas d'étiquettes pour les données, laissant l'algorithme trouver la structure à partir de l'entrée.

Comme les données ne sont pas étiquetées, il n'y a pas d'évaluation de la précision de la structure produite par l'algorithme.

Cela inclut le regroupement, la réduction de la dimensionnalité et l'apprentissage des règles d'association. L'algorithme peut ne jamais trouver la bonne sortie, mais modéliser la structure sous-jacente des données. [8]

4.2.3. L'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.[9]

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.[8]

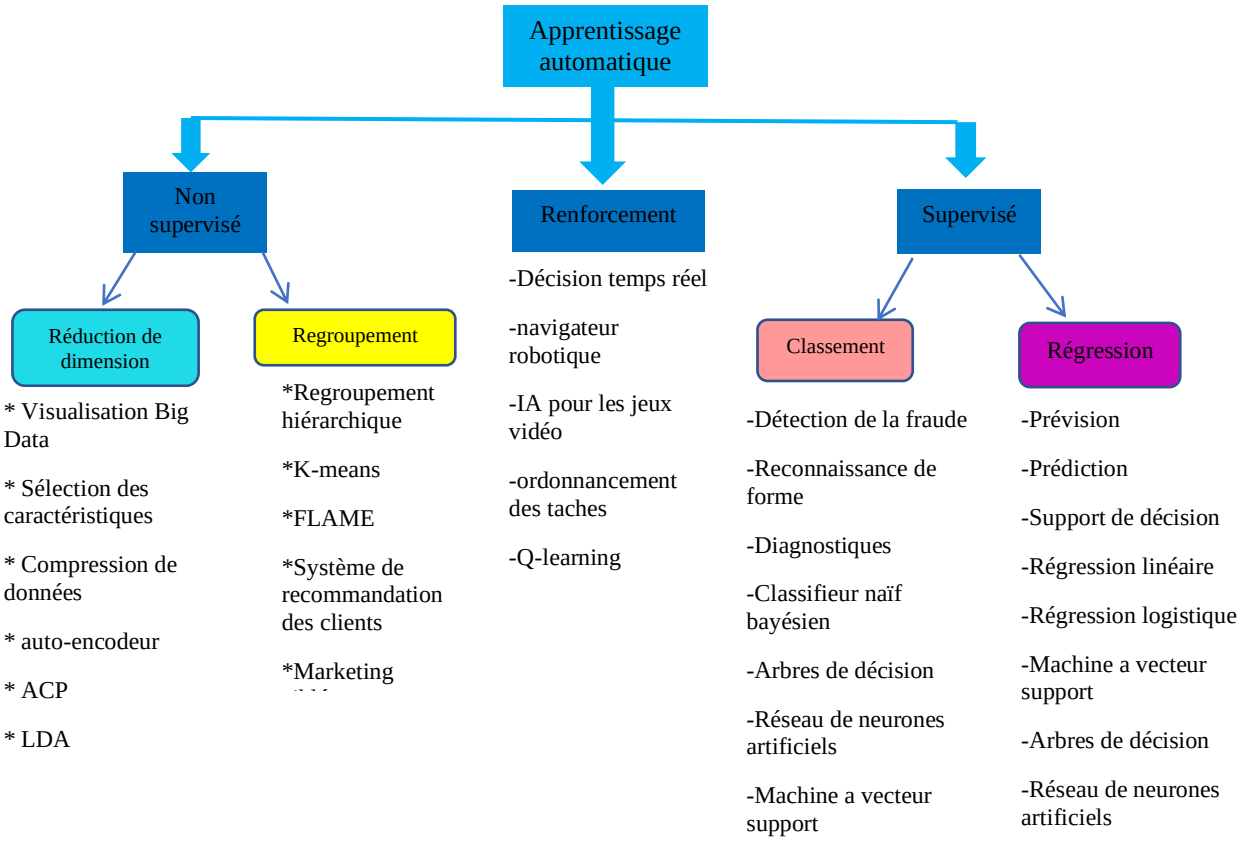


Figure 1.3 : les différents types de l'apprentissage automatique

4.3. Apprentissage profond :

Il y a plusieurs définitions de l'apprentissage profond en anglais « deep learning », certains le définissent comme un sous-domaine de l'apprentissage automatique qui s'intéresse aux algorithmes appelés réseaux de neurones artificiels, qui s'inspirent de la structure et du fonctionnement du cerveau humain. L'apprentissage à partir d'ensembles de données peut être supervisé, semi-supervisé ou non supervisé. [7]

4.3.1. Terminologie

- **Neurone biologique :**

Notre cerveau est composé de plus de 100 milliards de neurones, un neurone c'est une cible d'autres encoure qui va recevoir des signaux qui vont mélanger ces signaux pour retirer un signal qu'il va propager sur de nouveaux neurones.

Les neurones biologiques ont trois composants principaux : les dendrites, les corps cellulaires et les axones (Figure 1.4) Les dendrites forment un réseau de récepteurs neuronaux qui transmettent des signaux électriques d'autres neurones au corps du neurone. Cela agit comme une sorte d'intégrateur en accumulant des charges. Lorsqu'un neurone devient suffi-

samment excité (lorsque la charge accumulée dépasse un certain seuil), il génère un potentiel électrique de propagation par un processus électrochimique Stimule éventuellement d'autres neurones grâce à ses axones.[11]

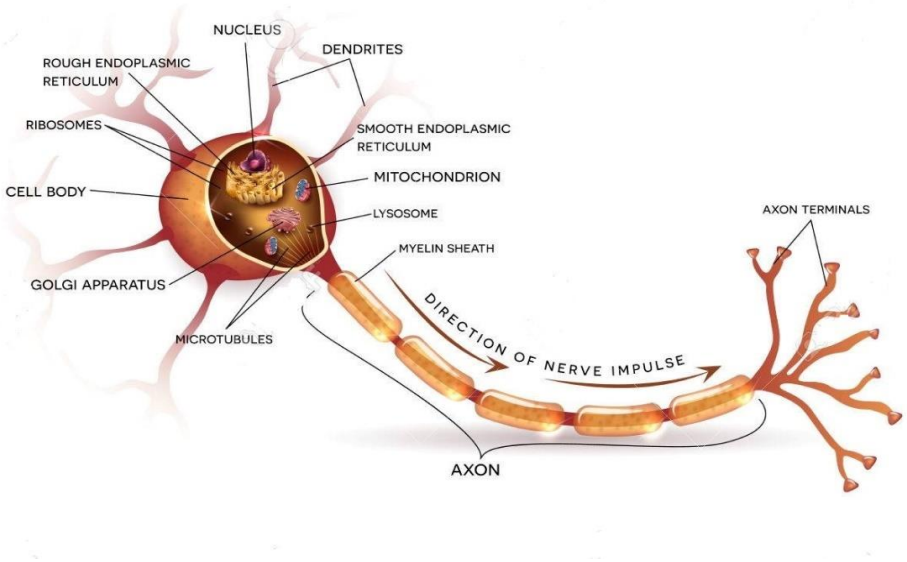


Figure 1.4 : Neurone Biologique [29]

- **Neurone artificiel**

A partir des neurones biologiques en a inspiré les neurones artificiels, en défini ce dernier comme une fonction qui apprend à manipuler des caractéristiques sur les quelles il va appliquer une fonction, et va ressortir une nouvelle valeur qui va se propager vers d'autre neurone.

Dans la figure suivant l'explication des composant d'un neurone artificiel et on expliquer leur fonctionnement :

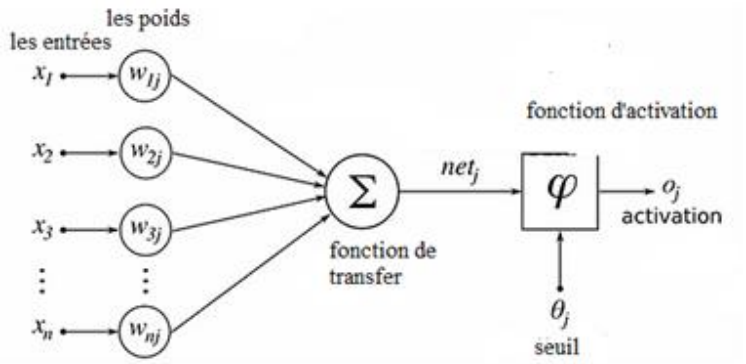


Figure 1.5 : Neurone Artificiel

On a les valeur entré (x_i) sont les caractéristiques, la fonction de combinaison calculer la somme pondérée de poids (w_i) et ces entré Le résultat n de la somme pondérée s'appelle le

niveau d'activation du neurone, Lorsque le niveau d'activation atteint ou dépasse le seuil, alors l'argument de devient positif (ou nul). Sinon, il est négatif [11]

- **Les poids (weight):**

Les poids sont les coefficients de l'équation que vous essayez de résoudre. Les poids négatifs réduisent la valeur d'une sortie. Lorsqu'un réseau neuronal est entrainé sur l'ensemble d'apprentissage, il est initialisé avec un ensemble de poids. Ces poids sont ensuite optimisés pendant la période d'apprentissage et les poids optimaux sont produits.

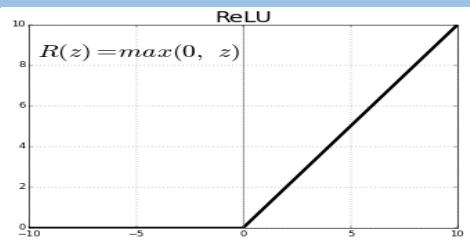
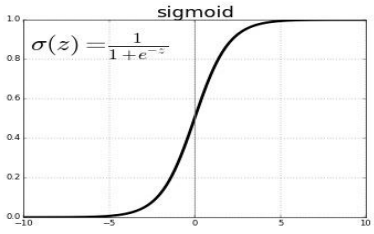
Le biais est simplement une valeur constante (ou un vecteur constant) qui est ajoutée au produit des entrées et des poids. Le biais est utilisé pour compenser le résultat.[19]

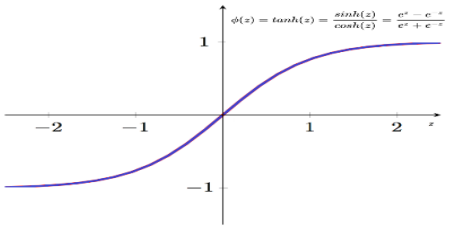
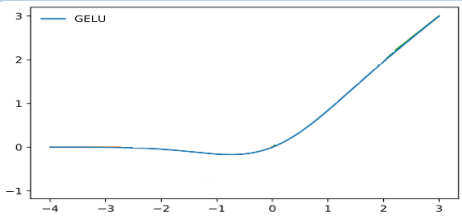
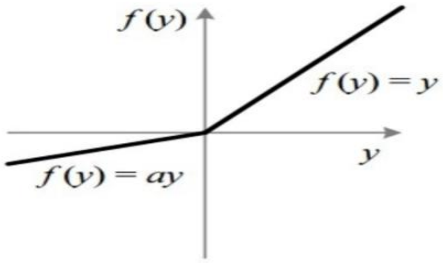
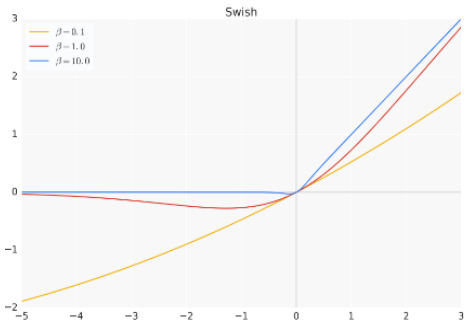
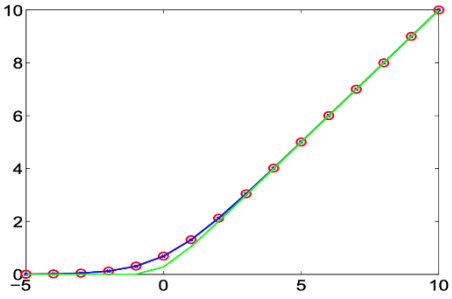
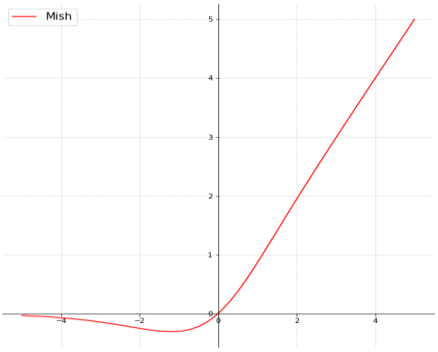
$$Y = \sum(poids * entrée) + bais$$

- **Fonction d'activation :**

Une fonction d'activation de perceptron non lisse signifie que le taux d'erreur du modèle est une fonction discontinue des paramètres de poids, ce qui rend difficile l'ajustement des poids optimaux en minimisant la fonction de perte. Pour résoudre ce problème, nous appliquons une fonction d'activation continue.[12]

Les fonctions d'activation sont des fonctions qui appliquent des transformations affinent combinant les poids et les caractéristiques d'entrée. Il s'agit généralement de fonctions non linéaires. L'unité linéaire rectifiée, où Relu, a été la plus populaire au cours de la dernière décennie, afin que le choix dépend par l'architecture et que de nombreuses alternatives aient émergé ces dernières années. [13]

La fonction	Courbe	Formule	L'année
ReLU[58]		f (x) = max(0, x)	2000
Sigmoid Activation [59]		$f(x) = \left(\frac{1}{1+e^{-x}}\right)$	2000

Tanh Activation [60]		$f(x) = \left(\frac{e^x - e^{-x}}{e^x + e^{-x}}\right)$	2000
GELU [61]		$f(x) = x.\frac{1}{2}\left[1 + \text{erf}\left(x/\sqrt{2}\right)\right]$	2016
Leaky ReLU [62]		f(x)= αx +x = $\begin{cases} x \; si \;\; x > 0 \\ ax \; si \; x \leq 0 \end{cases}$	2014
Swish [63]		$f(x) = x.sigmoid(x)$	2017
Softplus [64]		*f(x)=ln(1+ exp^x)*	2000
Mish [65]		*F(x) = Tanh Softplus(x)*	2019

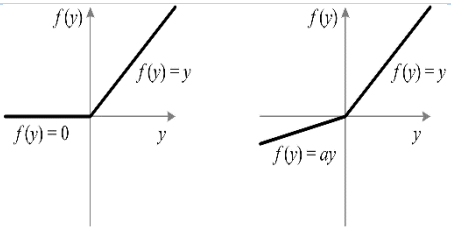
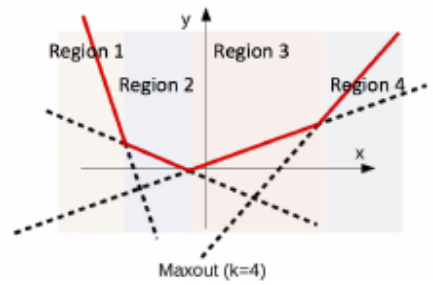
PReLU [66]		$f(x) = \begin{cases} x_i\,, si\ x_i > 0 \\ a_i x_i\,, si\ x_i \leq 0 \end{cases}$	2014
Maxout [67]		$f(x) = \max\,(w_1^T x + b_1, w_2^T x + b_2)$	2017

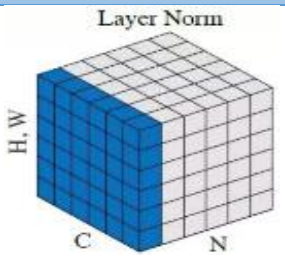
Tableau 1.4 : Top 10 des Fonctions d'Activation les Plus Utilisées.

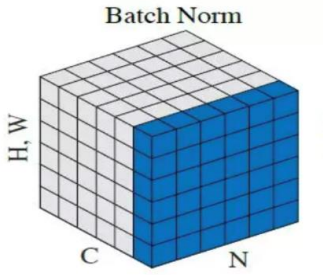
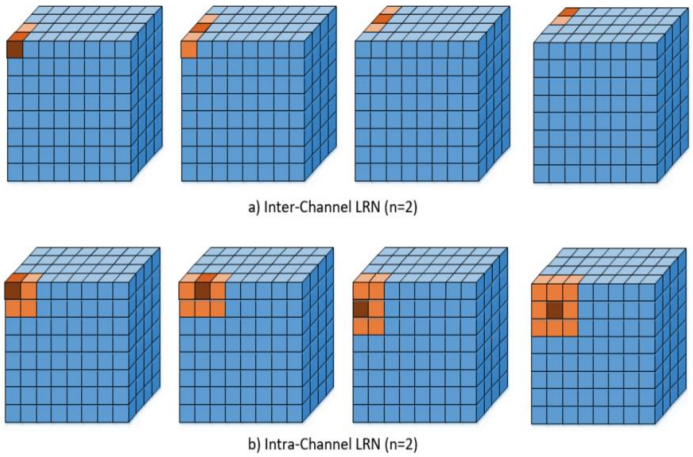
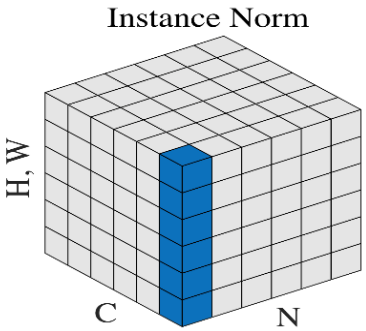
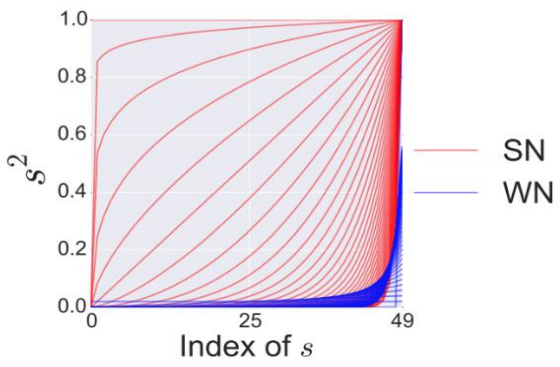
La fonction	Formule	L'année	Reference
TanhExp	$f(x) = xtanh(e^x)$	***2020***	**[68]**
m-arcsinh	$f(x) = arcsinh(x) * \frac{1}{12} * \sqrt{\lvert x \rvert}$	***2020***	**[69]**
Smooth Step	$\mathcal{S}(t) = \begin{cases} 0 & \text{if } t \leq -\gamma/2 \\ -\frac{2}{\gamma^3}t^3 + \frac{3}{2\gamma}t + \frac{1}{2} & \text{if } -\gamma/2 \leq t \leq \gamma/2 \\ 1 & \text{if } t \geq \gamma/2 \end{cases}$	***2020***	**[70]**
ASAF	$D_{\tilde{\pi},\pi_G}(\tau) = \frac{\prod_{t=0}^{T-1} \tilde{\pi}(a_t \lvert s_t)}{\prod_{t=0}^{T-1} \tilde{\pi}(a_l \lvert s_l) + \prod_{t=0}^{T-1} \pi_G(a_l \lvert s_l)}$	***2020***	**[71]**

Tableau 1.5 : Les Nouveaux Fonctions d'Activation.

Normalisation :

La normalisation des entrées est largement utilisée dans les modèles d'apprentissage automatique. Intuitivement, la normalisation d'une entrée supprime la différence de magnitude entre les différentes caractéristiques, ce qui favorise l'apprentissage, qui garantit que les données transformées possèdent certaines propriétés statistiques [15].

Méthode	figure	L'année
Layer Normalization [81]		2016

Batch Normalization [82]	 0	2015
Local Response Normalization [83]		2012
Instance Normalization [84]		2016
Adaptive Instance Normalization [85]	$$\mathrm{AdaIN}(x,y) = \sigma(y)\left(\frac{x-\mu(x)}{\sigma(x)}\right) + \mu(y)$$	2017
Spectral Normalization [86]		2018

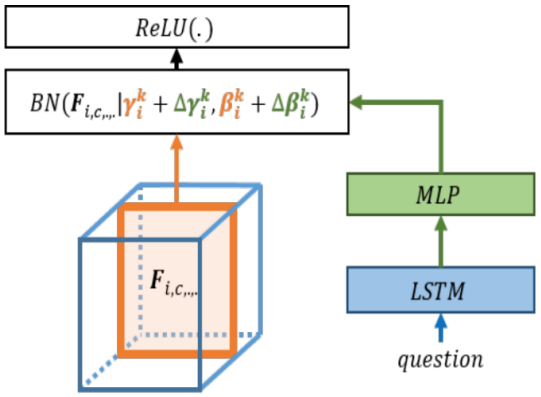
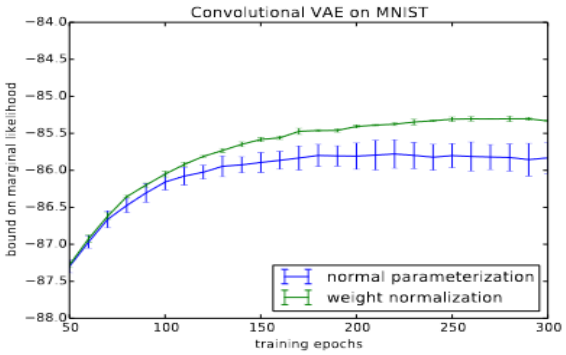
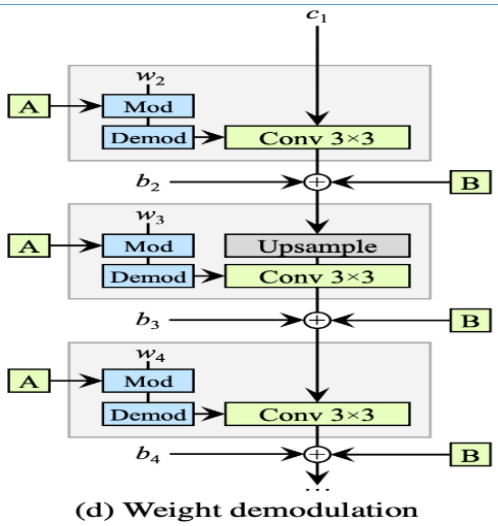
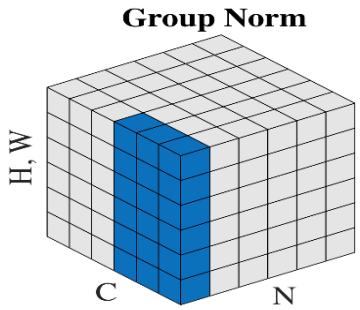
Conditional Batch Normalization [87]		2017
Weight Normalization [88]		2016
Weight Demodulation [89]	 (d) Weight demodulation	2019
Group Normalization [90]		2018

Tableau 1.6 : Top 10 des Méthodes de Normalisation les Plus Utilisées.

Optimisation :

En termes simples, les algorithmes d'optimisation sont chargés de réduire les pertes et de fournir des résultats aussi précis que possible. Le poids est initialisé en utilisant certaines stratégies d'initialisation et est mis à jour à chaque époque selon l'équation. Les meilleurs résultats sont obtenus en utilisant certaines stratégies d'optimisation ou algorithmes appelés Optimiseur [16].

Méthode	Propriétés	Avantages	Inconvénients
GD	Résoudre la valeur optimale le long de la direction de la descente du gradient. La méthode converge à un taux linéaire.	La solution est globalement optimale lorsque la fonction objective est convexe.	À chaque mise à jour des paramètres, les gradients des échantillons totaux doivent être calculés, ce qui entraîne un coût de calcul élevé.
SGD	Les paramètres de mise à jour sont calculés à l'aide d'un mini-batch échantillonné de façon aléatoire, et la méthode converge à un taux sous-linéaire.	Le temps de calcul pour chaque mise à jour ne dépend pas du nombre total d'échantillons d'entraînement, ce qui permet d'économiser beaucoup de frais de calcul.	Il est difficile de choisir un taux d'apprentissage approprié, et utiliser le même taux d'apprentissage pour tous les paramètres n'est pas approprié. Dans certains cas, la solution peut être bloquée au point de selle.
NAG	Accélérer la descente du gradient actuel en accumulant le gradient précédent en tant que momentum et effectuer le processus de mise à jour du gradient avec le momentum	Lorsque la direction du gradient change, le momentum peut ralentir la vitesse de mise à jour et réduire l'oscillation ; lorsque la direction du gradient reste, le momentum peut accélérer la mise à jour des paramètres. Le momentum aide à sortir de la solution localement optimale.	Il est difficile de choisir un taux d'apprentissage approprié.
AdaGrad	Le taux d'apprentissage est ajusté de manière adaptative en fonction de la somme des carrés de tous les gradients historiques.	Au début de la formation, le gradient cumulatif est plus petit, le taux d'apprentissage est plus grand et la vitesse d'apprentissage est plus rapide. La méthode est adaptée au traitement des problèmes de gradient clairsemé et le taux d'apprentissage de chaque paramètre s'ajuste de manière adaptative.	Au fur et à mesure que le temps d'apprentissage augmente, le gradient accumulé devient de plus en plus grand, ce qui fait que le taux d'apprentissage tend vers zéro et que les mises à jour des paramètres sont inefficaces. Un taux d'apprentissage manuel est toujours nécessaire. Il n'est pas adapté au traitement des problèmes non convexes.
AdaDelta/ ***RMSProp***	Changer la méthode d'accumulation du gradient total en une moyenne mobile exponentielle.	Améliorer le problème d'apprentissage inefficace dans la dernière phase d'AdaGrad. Il convient à l'optimisation de problèmes non stationnaires et non convexes	Dans la phase de formation tardive, le processus de mise à jour peut être répété autour du minimum local.
Adam	Combinez les méthodes adaptatives	Le processus de descente du gradient est	La méthode peut ne pas converger dans

	et la méthode du moment. Utilisez l'estimation du moment de premier ordre et l'estimation du moment de second ordre du gradient pour ajuster dynamiquement le taux d'apprentissage de chaque paramètre. Ajoutez la bi-correction	relativement stable. Il convient à la plupart des problèmes d'optimisation non convexes avec de grands ensembles de données et un espace de grande dimension.	certains cas.
SAG	L'ancien gradient de chaque échantillon et la somme des gradients de tous les échantillons sont conservés en mémoire. Pour chaque mise à jour, un échantillon est choisi au hasard et la somme des gradients est recalculée et utilisée comme direction mise à jour.	La méthode est un algorithme à convergence linéaire, qui est beaucoup plus rapide que la DGS.	La méthode n'est applicable qu'aux fonctions lisses et convexes et nécessite de stocker le gradient de chaque échantillon. Il est difficile de l'appliquer aux réseaux neuronaux non convexes.
SVRG	Au lieu de sauvegarder le gradient de chaque échantillon, le gradient moyen est sauvegardé à intervalles réguliers. La somme des gradients est mise à jour à chaque itération en calculant les gradients par rapport aux anciens paramètres et aux paramètres actuels pour les échantillons choisis au hasard.	La méthode ne nécessite pas de maintenir tous les gradients en mémoire, ce qui permet d'économiser les ressources mémoire. Il s'agit d'un algorithme de convergence linéaire	Pour l'appliquer à des réseaux neuronaux plus grands/plus profonds dont le coût d'apprentissage est un problème critique, des recherches supplémentaires sont encore nécessaires.
ADMM	La méthode résout les problèmes d'optimisation avec des contraintes linéaires en ajoutant un terme de pénalité à l'objectif et en séparant les variables en sous-problèmes qui peuvent être résolus de manière itérative	La méthode utilise les opérateurs séparables dans le problème d'optimisation convexe pour diviser un grand problème en plusieurs petits problèmes qui peuvent être résolus de manière distribuée. Ce cadre est pratique pour la plupart des problèmes d'optimisation à grande échelle.	Les résidus originaux et les résidus doubles sont tous deux liés au paramètre de pénalité dont la valeur est difficile à déterminer.
Frank-Wolfe	La méthode approxime la fonction objective avec une fonction linéaire, résout la programmation linéaire pour trouver la direction descendante réalisable, et effectue une recherche unidimensionnelle le long de la direction dans le domaine réalisable.	La méthode peut résoudre des problèmes d'optimisation avec des contraintes linéaires, dont la vitesse de convergence est rapide dans les premières itérations.	La méthode converge lentement dans les phases ultérieures. Lorsque le point itératif est proche de la solution optimale, la direction de recherche et le gradient de la fonction objectif ont tendance à être orthogonaux. Une telle direction n'est pas la meilleure direction descendante.

Tableau 1.7 : Résumé des méthodes d'Optimisation les plus utilisés [25]

Méthode	*Années*
Gravity [91]	2021
Distributed Shampoo [92]	2021
MADGRAD [93]	2021
ATMO [94]	2021
DAC [95]	2020
MAS [96]	2020
AMP [97]	2020
SLR [98]	2020
Adabelief [99]	2020
Apollo [100]	2020
MPSO [101]	2020
Grammatical evolution + Q-learning [102]	2020
AdaHessian [103]	2020

Tableau 1.8 : les Nouvelles Méthodes d'Optimisation (2020-2021).

- **Fonction de Perte**

Le réseau neuronal utilise des stratégies d'optimisation comme la descente de gradient stochastique pour minimiser l'erreur de l'algorithme. Cette erreur est calculée à l'aide d'une fonction de perte. Elle est utilisée pour quantifier les bonnes ou mauvaises performances du modèle.[14]

La fonction	*Formule*	*Année*
Focal Loss [104]	$FL(p_t) = -(1-p_t)^{\gamma} log(p_t)$	2017
Cycle Consistency Loss [105]	$l_{cyc}(G,F) = E_{x\sim Pdata(x)}\left[\left\|F(G(x)) - x\right\|_1\right] + E_{y\sim Pdata(y)}\left[\left\|G(F(y)) - y\right\|_1\right]$	2017
Triplet Loss [106]	$L_t(v_p, v_n) = -\frac{1}{MN}\sum_{i}^{M}\sum_{j}^{N} logprob(vp_i, vn_j)$	2018
GAN Least Squares Loss [107]	$\min\max V_{GAN}(D,G) = E_{x\sim Pdata(x)}[logD(x)]\left[\log(1 * -D(G(z)))\right]$	2016
InfoNCE [108]	$\mathcal{L}_N = -\mathbb{E}_X\left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j\in X} f_k(x_j, c_t)}\right]$	2018

GAN Hinge Loss [109]	$L_D = -\mathbb{E}_{(x,y)\sim p_{data}}[\min(0, -1 + D(x,y))] - \mathbb{E}_{z\sim p_z, y\sim p_{data}}[\min(0, -1 - D(G(z),y))]$ $L_G = -\mathbb{E}_{z\sim p_z, y\sim p_{data}} D(G(z),y)$	2017
NT-Xent [110]	$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}$	2016
WGAN-GP Loss [111]	$L = \mathbb{E}_{\tilde{\mathbf{x}}\sim\mathbb{P}_g}[D(\tilde{\mathbf{x}})] - \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r}[D(\mathbf{x})] + \lambda \mathbb{E}_{\hat{\mathbf{x}}\sim\mathbb{P}_{\hat{\mathbf{x}}}}\left[(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2\right]$	2017
ArcFace [112]	$L_3 = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{e^{s(\cos(\theta_{y_i}+m))}}{e^{s(\cos(\theta_{y_i}+m))} + \sum_{j=1, j\neq y_i}^{n} e^{s\cos\theta_j}}$	2018
Adaptive Robust Loss [113]	$f(x,\alpha,c) = \frac{\lvert\alpha-2\rvert}{\alpha}\left(\left(\frac{(x/c)^2}{\lvert\alpha-2\rvert}+1\right)^{\alpha/2} - 1\right)$	2017

Tableau 1.9 : Top 10 des Fonction de perte les plus utilisées.

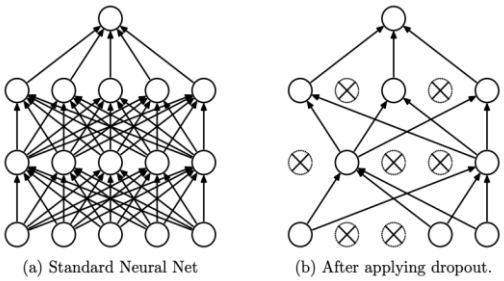
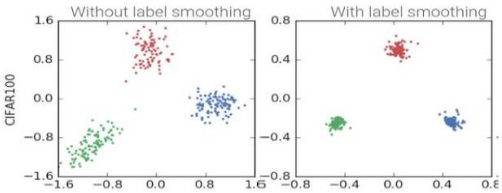
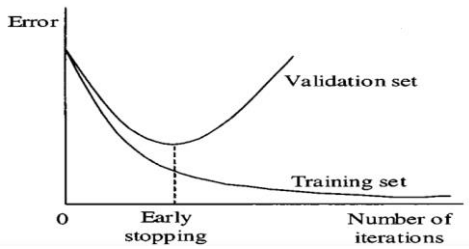
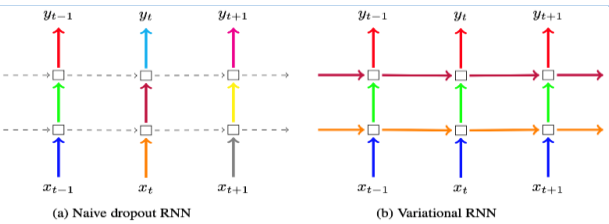
Fonction	*L'année*
Metrix [114]	2021
Supervised Contrastive Loss [115]	2020
Varifocal Loss [116]	2020
Dynamic SmoothL1 Loss [117]	2020
DSAM loss [118]	2020
Triplet Entropy Loss [119]	2020
Generalized Focal Loss [120]	2020

Tableau 1.10 : Les Nouveaux Fonctions de perte (2020-2021).

- **La régularisation**

Cette technique clé de machine learning vise à limiter le "sur-apprentissage " (overfitting) et à contrôler l'erreur de type variance pour aboutir à de meilleures performances.

Lors de l'apprentissage d'un modèle, la régularisation permet d'imposer une contrainte pour favoriser les modèles simples au détriment des modèles complexes. Autrement dit, cela permet de réduire l'erreur de type variance et d'améliorer la généralisation de la solution [17]

Méthode	Explication	Année
Dropout [121]	 (a) Standard Neural Net (b) After applying dropout.	2014
Weight Decay [122]	$L_{new}(w) = L_{original}(w) + \lambda w^T w$	1943
Attention Dropout [123]	$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$	2018
Label Smoothing [124]		1985
Entropy Regularization [125]	$H(X) = -\sum \pi(x) \log(\pi(x))$	2016
Early Stopping [126]		1995
R1 Regularization [127]	$R_1(\psi) = \frac{\gamma}{2} E_{p_D(x)}\left[\|\|\nabla D_\psi(x)\|\|^2\right]$	2018
Variational Dropout [128]	 (a) Naive dropout RNN (b) Variational RNN	2015
DropConnect [129]	La sortie est donnée comme : $r = a((M * W)v)$	2013

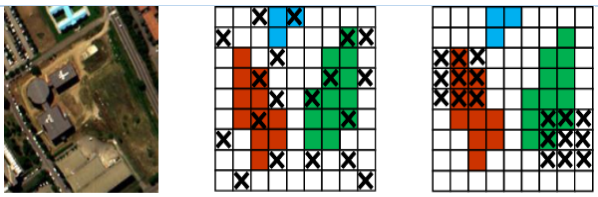
DropBlock [130]		2018

Tableau 1.11 : Top 10 des méthodes de régularisation les plus utilisées.

Méthode	Année
LayerScale [131]	2021
STTP [132]	2021
ALS [133]	2020

Tableau 1.12 : les nouvelles méthodes de régularisation (2020-2021).

4.3.2. Les architecture d'apprentissage profond

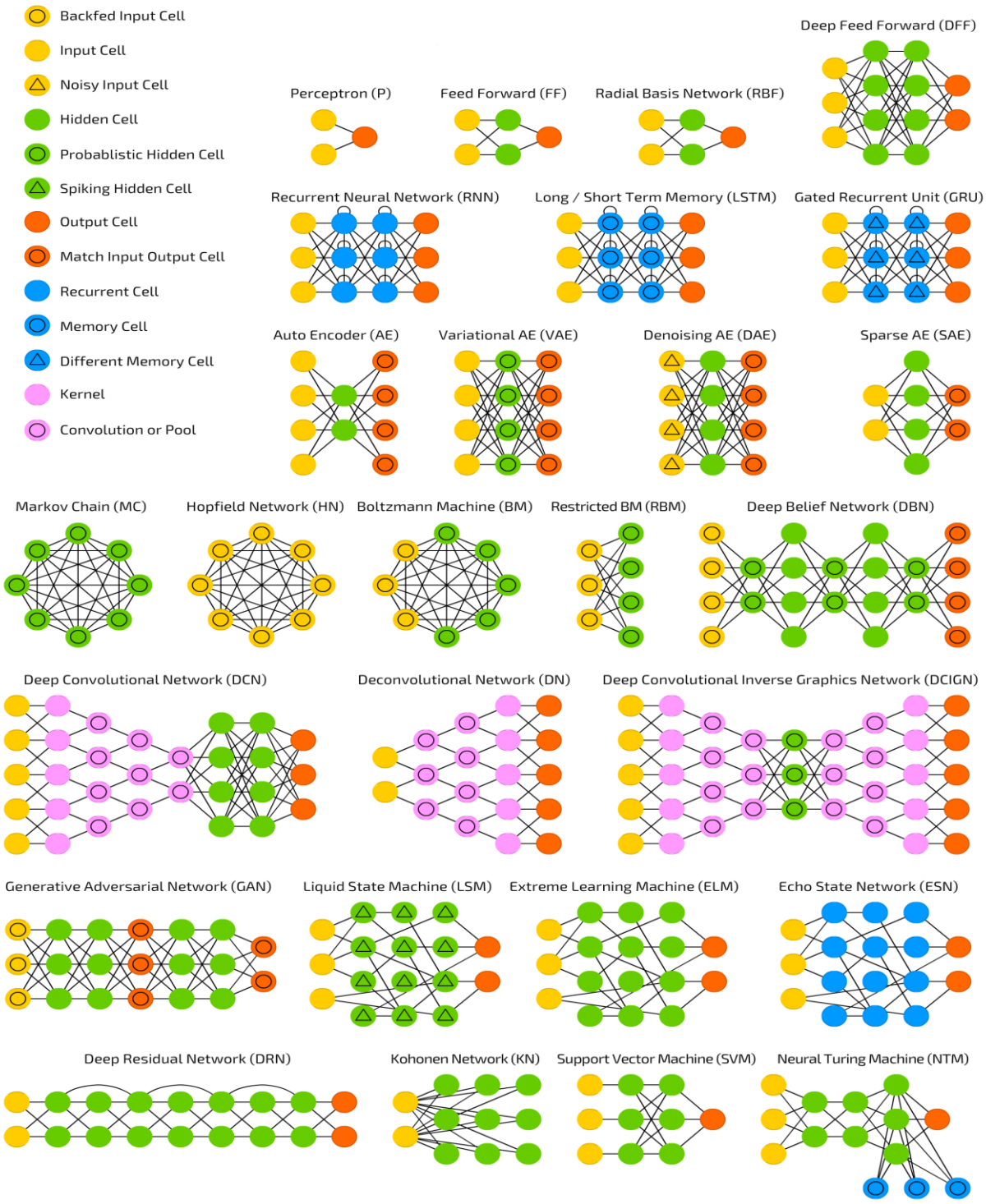


Figure 1.6: Les Différent Architecture d'Apprentissage Profond [30]

Les modèles d'apprentissage profond sont classés dans un arbre qui comporte trois branches : générative, discriminative et hybride. Dans chaque modèle, nous montrons quelques exemples de modèles d'apprentissage afin de voir la différence entre les trois modèles.

1) Réseaux Neuronaux Récurrents (RNN)

Le réseau neuronal récurrent (RNN) est un type de réseau où les connexions entre les unités forment un cycle dirigé. En réalité, des connexions plus complexes peuvent exister entre les unités cachées. Grâce à ces connexions récurrentes, les RNN peuvent traiter des données séquentielles, par exemple des vidéos et des phrases de discours, en présentant des comportements temporels dynamiques.[20]

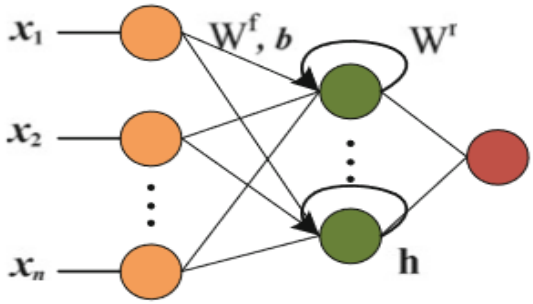


Figure 1.7 : Architecture de Réseaux Neuronaux Récurrents [20]

2) Réseaux Adversarial Génératifs (GAN)

Ce réseau qui combine entre les modèles génératifs et les modèles discriminatif en même temps, la distribution des donnés capturé par le modèle génératif après en distinguant les données d'entrée originales et les données provenant du modèle génératif.

Il s'agit d'un jeu à somme nulle entre les modèles génératif et discriminatif où le modèle génératif vise à contrefaire les données d'entrée originales, tandis que le modèle discriminatif vise à discriminer l'entrée originale et la sortie du modèle génératif.

Les points fort de ce réseau elles sont de conserver la cohérence après l'atteinte de l'équilibre, de ne pas nécessiter d'inférence approximative ou de chaînes de Markov, et de pouvoir être entraîné avec des données manquantes ou limitées. D'autre part, l'inconvénient de l'application du GAN est de trouver l'équilibre entre les modèles génératif et discriminatif.[18]

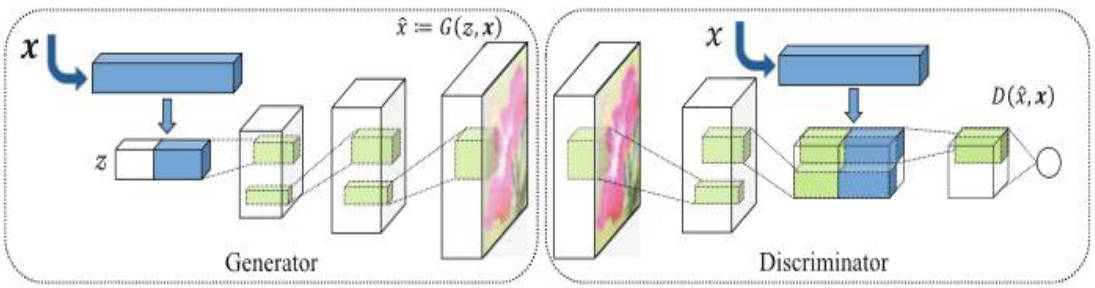


Figure 1.8: Architecture de Réseaux Adverbiaux Génératifs

3) Réseaux Neuronaux Convolutifs (CNN) :

Les réseaux neuronaux convolutifs (CNN), également appelés ConvNets, sont un type de réseaux neuronaux à anticipation bien adaptés aux tâches liées au domaine de la vision par ordinateur, notamment à la reconnaissance d'objets. [22]

Une architecture CNN typique comprend généralement des couches alternées de convolution et de mise en commun, suivies d'une ou plusieurs couches entièrement connectées à la fin. Dans certains cas, une couche entièrement connectée est remplacée par une couche de mise en commun de la moyenne globale. En plus des différentes fonctions de mappage, différentes unités de régulation telles que la normalisation et le dropout des lots sont également incorporées pour optimiser les performances du CNN. La disposition des composants du CNN joue un rôle fondamental dans la conception de nouvelles architectures et l'obtention de meilleures performances. Cette section aborde brièvement le rôle de ces composants dans une architecture CNN.[23]

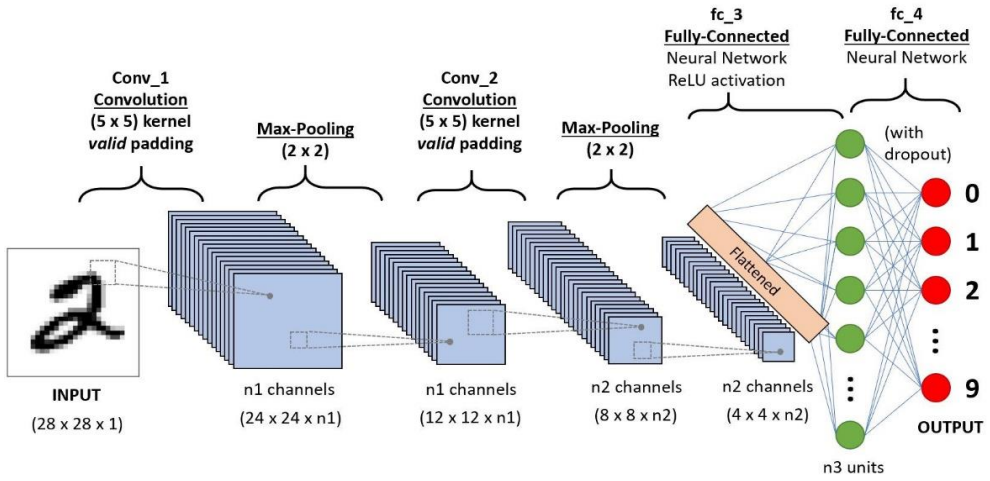


Figure 1.9 : architecture d'un réseau de neurone convolutif. [31]

- **La couche Convolution :** La couche de convolution est parfois appelée couche d'extraction de caractéristiques, car les caractéristiques de l'image sont extraites dans cette couche.

Tout d'abord, une partie de l'image est connectée à la couche Convolution pour effectuer une opération de convolution et calculer le produit scalaire entre le champ récepteur (c'est une région locale de l'image d'entrée ayant la même taille que celle du filtre) et le filtre Le résultat

de l'opération est un entier unique du volume de sortie. Ensuite, nous faisons glisser le filtre sur le champ récepteur suivant de la même image d'entrée par une foulée et refaisons la même opération. Cette opération est répétée par le même processus encore et encore jusqu'à ce que toute l'image soit parcourue.[21]

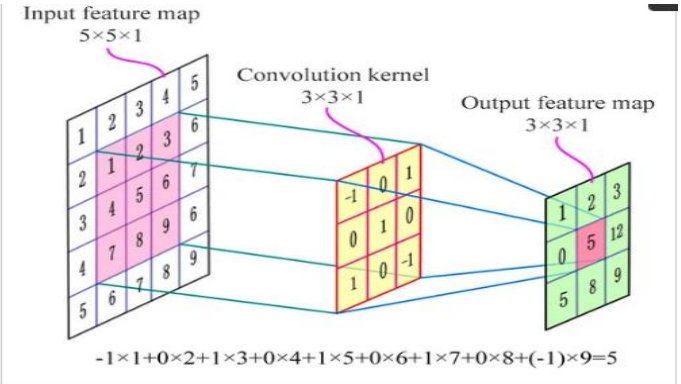


Figure 1.10 : La couche de convolution

- **Couche de mise en commun (Pooling) :** La couche de mise en commun (POOL) est une opération de sous-échantillonnage, généralement cette opération est appliquée entre deux couches de convolution.

Sa fonction est de réduire progressivement la taille de la carte de fonctionnalités (matrice de convolution) pour réduire les paramètres et les calculs réseau, tout en conservant les informations importantes.[24]

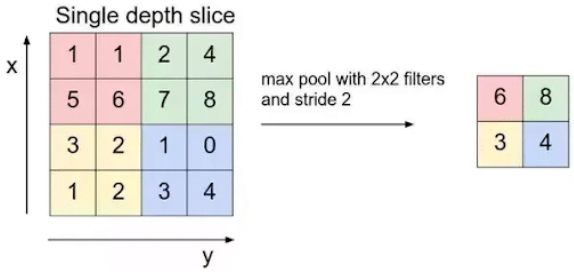


Figure 1.11 : la couche pooling

- **Couche entièrement connecté (Fully Connected) :** Les couches totalement connectées font les mêmes tâches que celles des ANN standard et tenteront de produire des notes de classe à partir des activations, pour les utiliser pour la classification. Il est également suggéré d'utiliser ReLu entre ces couches pour améliorer les performances.[26]

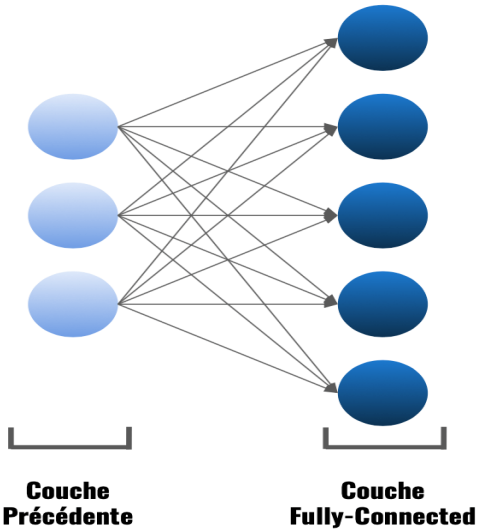


Figure 1.12 : Couche entièrement connecté (Fully Connected) [32]

- **Couche de sortie (output layer) :** La couche de sortie, c'est la dernière couche de réseaux qui contient les neurones qui identifient les classes de modèle, donc le nombre de neurones à cette couche dépende du nombre de classes.

Il y a plusieurs demains qui utiliser le CNN pour résoudre les problèmes et parmi ces demain 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 :

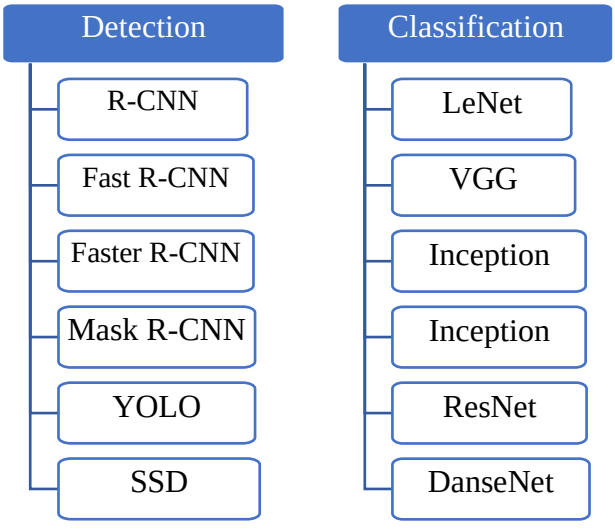


Figure 1.13 : schéma des modèles de détection et classification par CNN

5. Les bases de données :

Comme nous avons dit que les données jouent un rôle très important dans les algorithmes de machine Learning dans la partie d'apprentissage, elles sont l'instrument qui permet à AI de

comprendre comment les gens le pensent, aussi elles permettent d'accélérer la courbe d'apprentissage, plus il apprend et plus il devient précis.

Aujourd'hui, les bases de données sont accessibles à tout moment. Ceci permet aide à la révolution des modèles de machine Learning.

Dans cette partie nous allons voir les benchmarks base de données dans la détection avec leurs détails, aussi les modèles les plus utilisés dans chaque base :

5.1. La base de données MS COCO :

MS-COCO (Microsoft Common Objects in Context) est un jeu de données de reconnaissance visuelle largement utilisé, conçu pour stimuler la recherche sur la détection d'objets en mettant l'accent sur la compréhension complète de la scène. Cette base de données généralement utiliser dans : Détection d'objets, Détection des points clés, Segmentation panoptique, Sous-titrage, Segmentation, Pose dense. Segmentation de la substance

Nouveauté 2018, des substance et annotations panoptiques complètes pour toutes les images 2017 sont disponibles. Les challenge en 2019/2020 Toutes les données pour tous les défis restent inchangées. Mais en ajout la tache de Pose dense.[80]

Figure 1.14 : Exemple d'images de la base MScoco

```
annotation{
"id": int,
 "image_id": int,
 "category_id": int,
 "segmentation": RLE or [polygon],
 "area": float,
 "bbox": [x,y,width,height],
 "iscrowd": 0 or 1,
}
```

```
"annotations": [
    {
        "segmentation":
[[510.66,423.01,511.72,420.03,...,510.45,423.01]],
        "area": 702.10,
        "iscrowd": 0,
        "image_id":  397133,
        "bbox": [433.07,355.93,138.65,228.67],
        "category_id": 18,
        "id": 1768
    },
    {
        "segmentation":
       {
          "counts":[12,56,198,10]
          "size":[120, 240]
       }
        "area": 500.2,
        "iscrowd": 1,
        "image_id": 397122,
        "bbox": [473.07,395.93,38.65,28.67],
        "category_id": 18,
        "id": 1768
}]
```

Figure 1.15: Example d'annotation de la base mscoco

Dans le tableau suivant on verra tous les challenges de la base de données MSCOCO à partir le premier challenge 2014 jusqu'à 2018 :

Challenge (objet détection)	Classe	Image			Annotation (JSON)		Résumé (appren + val)		
		Appren	Validation	Test	Appren	Val	Image	Boxes	Box/Img
MS COCO 2014	80	83000	41000	41000	-	-	124,000	-	-
MS COCO 2015	80	82,783	40,504	81,434	604,907	291,875	123,287	896,782	7.3
MS COCO 2016	80	82,783	40,504	81,434	604,907	291,875	123,287	896,782	7.3
MS COCO 2017	80	118,287	5000	40,670	860,001	36,781	123,287	896,782	7.3
MS COCO 2018	80	118,287	5000	40,670	860,001	36,781	123,287	896,782	7.3

Tableau 1.13 les challenges de la base de données MS COCO à partir le premier challenge 2014 jusqu'à 2018

Les modèles les plus utiliser :

Tache	Model
Détection d'objets	Swin-L, RepPoints+ Self-adaptation
Détection d'objets en temps réel	YOLOv4-CSP-P7, Mask R-CNN
Instance segmentation	Swin-L
Segmentation panoptique	REFINE, MaX-DeepLab-L
Détection des points clés	EvoPose2D-L, HRNet
Sous-titrage des images	M2 Transformer

Tableau 1.14 Les modèles les plus utiliser dans la base de données mscoco

5.2. La base de données PASCAL VOC :

PASCAL Visual Object Classes (VOC), dans ce challenge il y a 20 catégories d'objets, dont les véhicules, les ménages, les animaux et autres. Chaque image de cette base de données comporte des annotations de segmentation au niveau du pixel, des annotations de boîte de délimitation, et des annotations de classes d'objets. Il y a deux tâches principales :

- La classification : Pour chacune des classes, prédire la présence/absence d'au moins un objet de cette classe dans une image test.

- Détection : Pour chacune des classes, prédire les boîtes de délimitation de chaque objet de cette classe dans une image de test

Les données ont été divisées en deux parties : 50 % pour la formation /validation et 50 % pour le test. Les distributions des images et des objets par classe sont approximativement égales dans les ensembles de apprentissage/validation et de test.[78]

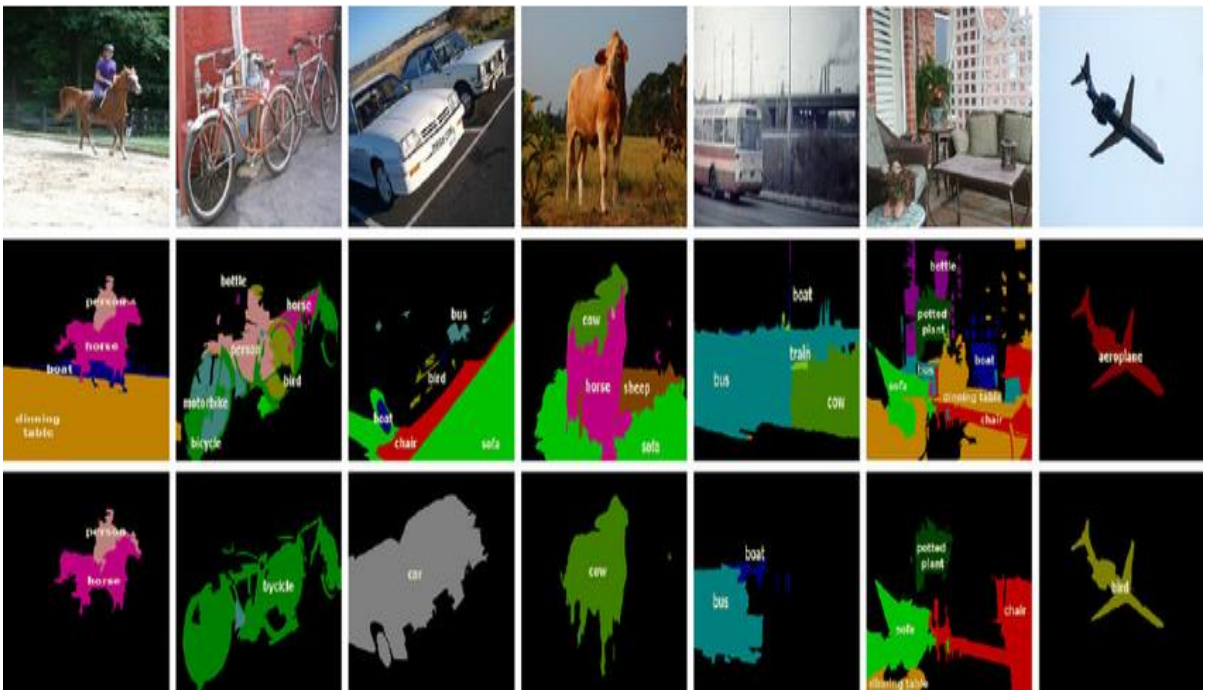


Figure 1.16 : Exemple d'images de la base Pascal Voc

```
<annotation>
        <folder>Kangaroo</folder>
        <filename>00001.jpg</filename>
        <path>./Kangaroo/stock-12.jpg</path>
        <source>
                <database>Kangaroo</database>
        </source>
        <size>
                <width>450</width>
                <height>319</height>
                <depth>3</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>kangaroo</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>233</xmin>
                        <ymin>89</ymin>
                        <xmax>386</xmax>
                        <ymax>262</ymax>
                </bndbox>
        </object>
</annotation>
```

Figure 1.17: Example d'annotation de la base pascal voc

Challenge (objet détection)	Classe	Image			Annotation (XML)		Résumé (appren + val)		
		Appren	Validation	Test	Appren	Val	Image	Boxes	Box/Img
PASCALVOC 2007	20	2501	2510	4952	6301	6307	5011	12608	2.4
PASCALVOC 2008	20	2113	2227	5717	5082	5281	4340	10363	2.4
PASCALVOC 2009	20	3473	3581	7689	8505	8713	7054	17218	2.4
PASCALVOC 2010	20	4998	5105	11635	11577	11797	10103	23374	2.4
PASCALVOC 2011	20	5721	5819	17412	13609	13841	11540	27450	2.4
PASCALVOC 2012	20	5717	5823	17412	13609	13841	11540	27450	2.4

Tableau 1.15 les challenge de détection d'objet dans pascalvoc 2007_2012

Les modèles les plus utiliser :

Tache	Model
Détection d'objets	Cascade Eff-B7, NAS-FPN ,SSD512
Détection d'objets en temps réel	YOLO
Sémantique segmentation	EfficientNet-L2+NAS-FPN GALDNet
Segmentation sémantique semi-supervisée	DMT, MaX-DeepLab-L
Détection d'objets faiblement supervisée	wetectron , CASD *

Tableau 1.16 les modèles les plus utilisé avec la base de données pascalvoc

5.3. La base de données Open Image

Open Images est à grande échelle en termes d'images (9 178 275), d'annotations (30 113 078 étiquettes au niveau de l'image, 15 440 132 boîtes de délimitation, 374 768 triplets de relations visuelles) et de nombre de concepts visuels (classes) (19794 pour les étiquettes au niveau de l'image et 600 pour les boîtes de délimitation). Cela en fait un outil idéal pour repousser les limites des méthodes gourmandes en données qui dominent l'état de l'art.

Pour la détection d'objets en particulier, l'échelle des annotations est sans précédent (15,4 millions de boîtes de délimitation pour 600 catégories sur 1,9 million d'images).

Open Images est un ensemble de données contenant ~9 millions d'URL d'images annotées (fichier *.CSV) avec des étiquettes couvrant plus de 6000 catégories., réparties en trois catégories : apprentissage, validation et test.[79]

	Apprentissage	Validation	Test
Image	9,011,219	41,620	125,436

Tableau 1.17 la division générale de la base de données OpenImage

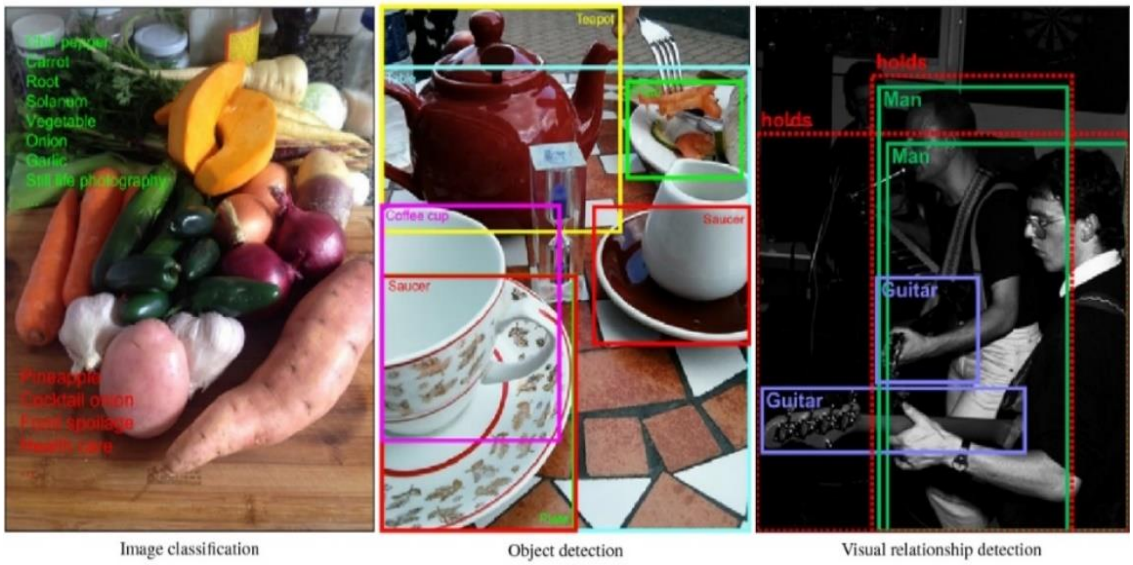


Figure 1.18 : Exemple d'images de la base open Image

```
ImageID,Source,LabelName,Confidence
000026e7ee790996,verification,/m/04hgtk,0
000026e7ee790996,verification,/m/07j7r,1
000026e7ee790996,crowdsource-verification,/m/01bqvp,1
000026e7ee790996,crowdsource-verification,/m/0csby,1
000026e7ee790996,verification,/m/01_m7,0
000026e7ee790996,verification,/m/01cbzq,1
000026e7ee790996,verification,/m/01czv3,0
000026e7ee790996,verification,/m/01v4jb,0
000026e7ee790996,verification,/m/03d1rd,0
...
```

Figure 1.19 : humain_csv annotation

Le challenge est basé sur la base de données Open Images. Les images de cette base de données sont très diverses et contiennent souvent des scènes complexes avec plusieurs objets. L'édition 2019 du défi comportait trois pistes :

- Détection d'objets : prédire une boîte de délimitation serrée autour de toutes les instances d'objets de 500 classes.
- Détection de relations visuelles : détection de paires d'objets dans des relations particulières.
- Segmentation d'instances : prédire les contours des instances d'objets de 300 classes.

Il y a 6 y version de la base de données open image

Version 01 : L'ensemble de données est divisé en un ensemble d'apprentissage (9011219 images) et un ensemble de validation (167057 images).

- Étiquettes au niveau de l'image

Les versions	Étiquettes au niveau de l'image	Apprentissage	Validation	Test	Classes	Classes-entrainer
Version 2 et 3	Étiquettes générées par la machine	78,977,695	512,093	1,545,835	7,870	4,966
	Étiquettes vérifiées par l'homme	20,868,755	551,390	1,667,399	19,693	5,000
Version 4	Étiquettes générées par la machine	78,977,695	512,093	1,545,835	7,870	4,764
	Étiquettes vérifiées par l'homme	27,894,289	551,390	1,667,399	19,794	7,186
Version 5	Étiquettes générées par la machine	164,819,642	681,179	2,061,177	15,387	8,386
	Étiquettes vérifiées par l'homme	34,069,33	595,339	1,799,883	19,943	8,658

Version 6	Étiquettes générées par la machine	164,819,642	681,179	2,061,177	15,387	9,034
	Étiquettes vérifiées par l'homme	57,524,352	595,339	1,799,883	19,957	9,605

Tableau 1.18 : Étiquettes au niveau de l'image

- Boîtes englobantes

Version	Images			Boxes (CSV)			
	Apprentissage	Validation	Test	Apprentissage	Validation	Test	Classe
Version 03	1,593,853	41,620	125,43	3,709,509	204,621	625,282	600
Version 4	1,743,042	41,620	125,436	14,610,229	204,621	625,282	600
Version 5 et 6	1,743,042	41,620	125,436	14,610,229	303,980	937,327	600

Tableau 1.19 : Boîtes englobantes

- Relations visuelles

Version	Triplés relationnels					
	Apprentissage	Validation	Test	Triolets de relations distinctes	Classes	Attributs
Version 04	374,768	3,983	12,248	329 obj-obj: 287 attr: 42	57	5
Version 05	374,768	3,991	12,314	329 obj-obj: 287 attr: 42	57	5
Version 06	3,174,291 non-attr: 348,560	27,243 non-attr: 4,951	82,746 non-attr: 14,403	1,466 non-attr: 1,384	288	

Tableau 1.20 : Relations visuelles

- Segmentations d'objets

Version	Images			Masques d'instance			
	Apprentissage	Validation	Test	Apprentissage	Validation	Test	Classes
Version 05 et 06	944,037	13,524	40,386	2,686,666	24,730	74,102	350

Tableau 1.21 : Segmentations d'objets

- Récits localisés

Images (v6)			Annotations narratives localisées		
Apprentissage	Validation	Test	Apprentissage	Validation	Test
504,413	41,620	125,436	507,444	41,691	126,020

Tableau 1.22 : Récits localisés

Les modèles les plus utiliser :

Tache	Modèle
Classification des images	Inception-ResnetV2
Détection d'objets	SSD, YOLO
Détection visuelle des relations	BAR-CNN avec ResNet50

Tableau 1.23 : les modèles les plus utilisé avec OpenImage

5.4. La base de données Image NET

Image Net est une base de données d'images organisée selon la hiérarchie de Word Net (actuellement uniquement les noms), dans laquelle chaque nœud de la hiérarchie est représenté par des centaines et des milliers d'images. Le projet a contribué à faire progresser la recherche sur la vision par ordinateur et l'apprentissage profond.[73]

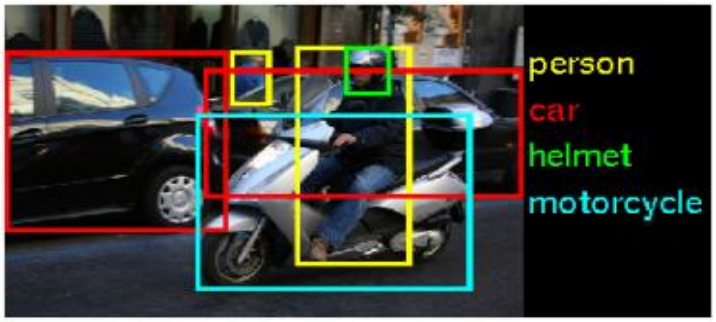


Figure 1.20 : Exemple d'images de la base ImageNet

L'Image Net Large Scale Visual Recognition Challenge (ILSVRC) évalue les algorithmes de détection d'objets et de classification d'images à grande échelle.

- ILSVRC 2010 : Image Classification
- ILSVRC 2011 : Classification avec localisation

❖ ILSVRC 2012 : novelle tasks Classification fine

Challenge	Classe	Image		
		Apprentissage	Validation	Test
ILSVRC 2010	1000	1,261,406	50,000	150,000
ILSVRC 2011	1000	1,229,413	50,000	100,000
ILSVRC 2012-2014	1000	1,281,167	50,000	100,000

Tableau 1.24 détails sur le dataset ImageNet pour la classification

❖ ILSVRC 2013 : novelle tasks détection d'objet

Challenge (Détection d'objet)	Classe	Image			Annotation (XML)		Box/img
		Apprentissage	Validation	Test	Apprentissage	Val	
ILSVRC 2013	200	395,909	20,121	40,152	345,854	55,502	1.1
ILSVRC 2014	200	456,567	20,121	40,152	478,807	55,502	1.1
ILSVRC 2015	200	456,567	20,121	51,294	478,807	55,502	1.1
ILSVRC 2016	200	456,567	20,121	60,000	478,807	55,502	1.1
ILSVRC 2017	200	456,567	20,121	65,500	478,807	55,502	1.1

Tableau 1.25 détails sur le dataset ImageNet pour la détection

❖ ILSVRC 2014 : L'ensemble d'apprentissage de données de détection sera considérablement élargi cette année par rapport à l'ILSVRC2013.

❖ ILSVRC 2015 : Ce challenge évalue les algorithmes de localisation/détection d'objets et de classification d'images/scènes à partir d'images et de vidéos à grande échelle. Plus précisément, les données du défi seront divisées en 8,1 millions d'images pour l'apprentissage, 20 000 images pour la validation et 381 000 images pour le test, provenant de 401 catégories de scènes.

❖ ILSVRC 2016 : novelle tasks Analyse syntaxique de scènes pour 150 catégories de trucs et d'objets discrets (en collaboration avec l'équipe Places du MIT). Plus précisément, les données du défi sont divisées en 20 000 images pour la formation, 2 000 images pour la validation et un autre lot d'images non retenues pour les tests.

❖ ILSVRC 2017 : Ce challenge évalue les algorithmes de localisation/détection d'objets à partir d'images/vidéos à grande échelle. Les équipes les plus performantes et innovantes seront invitées à présenter leurs travaux lors de l'atelier CVPR 2017.

- Localisation d'objets pour 1000 catégories.
- Détection d'objets pour 200 catégories entièrement étiquetées.
- Détection d'objets à partir de vidéos pour 30 catégories entièrement étiquetées. [74]

Les modèles les plus utiliser :

Tache	**Model**
Classification des images	Meta Pseudo Labels ResNet-50 + UDA +AutoDroupout
Classification semi-supervisée des images	SimCLRv2 self-distilled
Regroupement d'images	IDFD SPICF SCAN
Détection d'objets faiblement supervisée	PCL-OB-G-Ens + FRCNN
Reconnaissance d'images	LIO/ResNet-50

Tableau 1.26 : les modelés les plus utilisés avec ImageNet

Dans le tableau le résumer de benchmark base de données dans le Domain de reconnaissance des objets :

La base de données	***Total des images***	***Catégories***	***Images par catégorie***	***Objets par image***	***Taille de l'image***	***Année***	***Points forts***
PASCAL VOC (2012)	11,540	20	303–4087	2.4	470 × 380	2005	Couvre seulement 20 catégories courantes dans la vie quotidienne. Grand nombre d'images d'entraînement ; proche des applications du monde réel ; Variations intra-classe significativement plus importantes ; Objets dans le contexte de la scène ; Objets multiples dans une image ; Contient de nombreux échantillons difficiles.
ImageNet	14 millions+	21,841	-	1.5	500 × 400	2009	Grand nombre de catégories d'objets ; Plus d'instances et plus de Catégories d'objets par image ; Plus difficile que PASCAL VOC ; épine dorsale du défi ILSVRC Les images sont centrées sur l'objet
MS COCO	328,000+	91	-	7.3	640 × 480	2014	Encore plus proche des scénarios du monde réel ; Chaque image contient plus de

							d'instances d'objets et des informations d'annotation d'objets plus riches ; contient des données de notation de la segmentation des objets qui ne sont pas disponibles dans l'ensemble de données ImageNet.
Places	10 millions+	434	-	-	256 × 256	2014	Le plus grand ensemble de données étiquetées pour la reconnaissance de scènes ; quatre sous-ensembles Places365 Standard, Places365 Challenge, Places 205 et Places88 comme points de repère
Open Images	9 millions+	6000+	-	8.3	Varied	2017	Annoté avec des étiquettes au niveau de l'image, des boîtes de délimitation d'objets et des relations visuelles, Open Images V5 permet la détection d'objets à grande échelle. la segmentation des instances d'objets et la détection des relations visuelles.

Tableau 1.27 Bases de données populaires pour la reconnaissance d'objets [72]

6. Conclusion

Dans ce chapitre, nous avons présenté c'est quoi la détection et suivi d'objets. Ensuite, nous avons présenté un bref historique de l'intelligence artificielle, après nous avons vu les différents types l'apprentissage automatique (Machine Learning). De plus nous avons vu les différentes terminologies dans le domaine de l'apprentissage profond, après en présenter les 3 types principaux de deep learning. Enfin, nous avons présenté les différentes benchmark bases de données dans le domaine de la détection et reconnaissance d'objets.

Chapitre 02 : Les Modèles de Détection

1. Introduction :

La détection d'objets est une technique de vision par ordinateur qui a connu un changement révolutionnaire rapide, cette technique fait la combinaison de la classification et de la localisation d'objets. En 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.

2. Les modèles de détection par CNN :

Les architectures CNN sont capables d'apprendre des caractéristiques plus complexes. Il existe deux types de modèles de détection d'objets par les architectures CNN. Le premier type de détection en deux coups est basé sur la proposition de région et comprend des modèles tels que RCNN, SPP-NET, FRCNN, Faster RCNN et le second type la détection à un coup est basé sur la régression et comprend MultiBox, AttentionNet, G-CNN, YOLO, SSD...etc [36]

- ✓ ***Détection en deux coup :*** Comme son nom l'indique, cette méthode comporte deux étapes. La première est la proposition de régions, puis, dans la deuxième étape, la classification de ces régions et le raffinement de la prédiction de l'emplacement ont lieu.
- ✓ Au contraire, ***la détection à un coup*** saute l'étape de la proposition de région et produit la localisation finale et la prédiction du contenu en une seule fois [37].

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

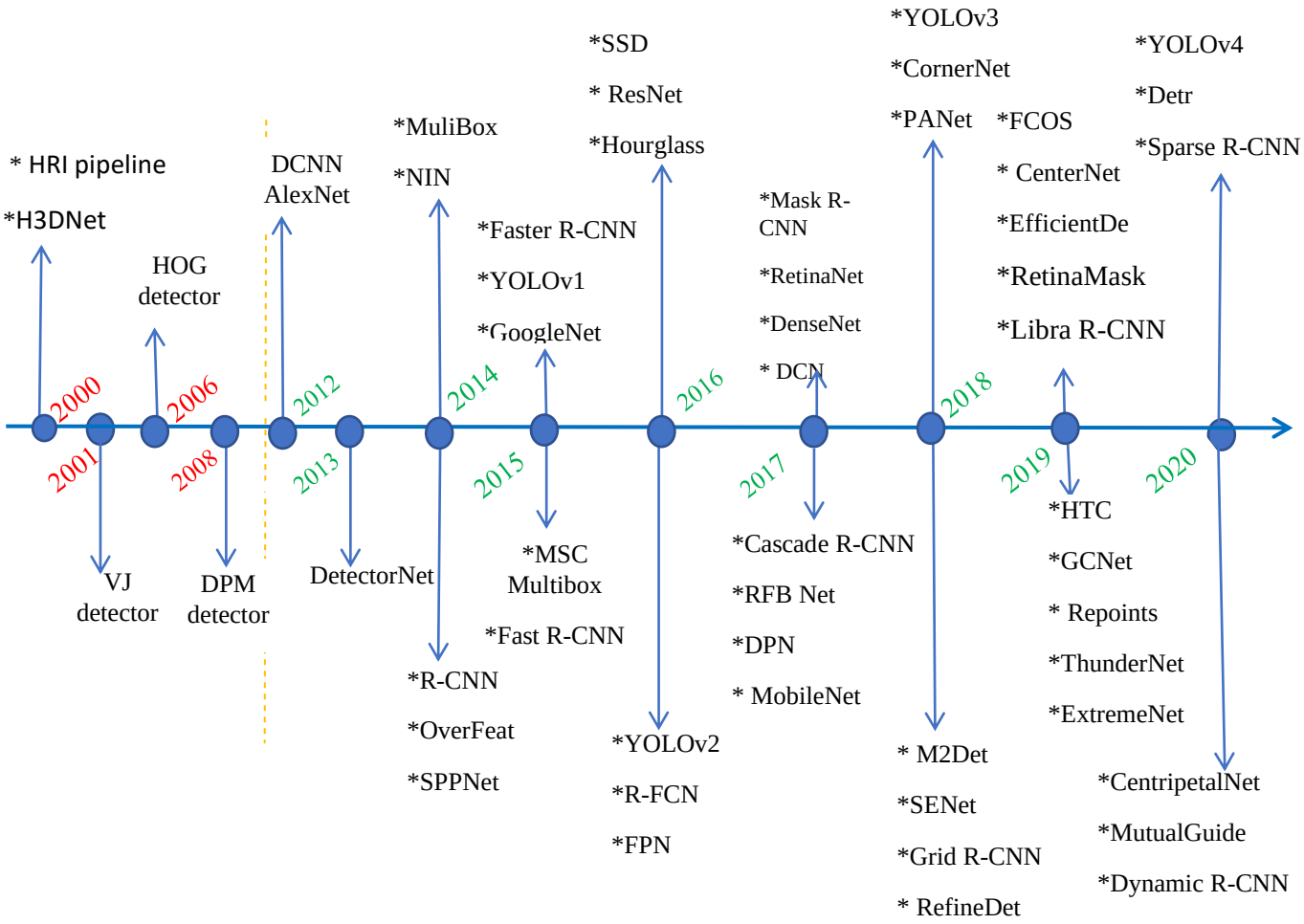


Figure 2.1 : Étapes importantes de la détection et de la reconnaissance des objets

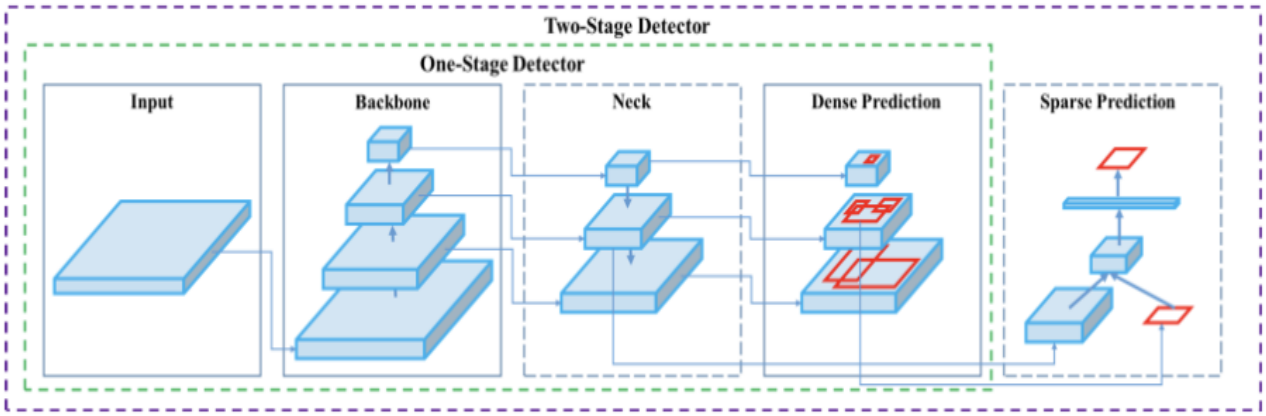


Figure 2.2 : Anatomie des différents types de détecteurs. [51]

- Entrée : Image, correctifs, Pyramide d'images
- Backbones: VGG16, ResNet-50, SpineNet, EfficientNet-B0/B7, CSPResNeXt50, CSPDarknet53

- Neck:
  - Blocs additionnels: SPP, ASPP, RFB, SAM
  - Blocs agrégation de chemins: FPN, PAN, NAS-FPN, Fully-connected FPN, BiFPN, ASFF, SFAM
- Heads:
  - Prédiction dense (une étape) : RPN, SSD, YOLO, RetinaNet, CornerNet, CenterNet, MatrixNet, FCOS.
  - Prédiction éparse (en deux étapes) : Faster R-CNN, R-FCN, Mask R-CNN, RepPoints. [51]

2.1. R-CNN :

Le model de détection d'objets R-CNN a été proposé par Ross Girshick en 2014 [33], ce modèle se compose de trois modules :

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

Le R-CNN atteint une précision moyenne (mAP) de 53,7% sur PASCAL VOC 2010. À titre de comparaison, et une précision moyenne de 35,1 % en utilisant les mêmes propositions de régions, mais avec une pyramide spatiale et une approche par sac de mots visuels. Sur l'ensemble de données de détection ILSVRC2013 de 200 classes, la mAP de R-CNN est 31,4 %, ce qui représente une grande amélioration par rapport à OverFeat, qui avait le meilleur résultat précédent avec 24,3 %.[33]

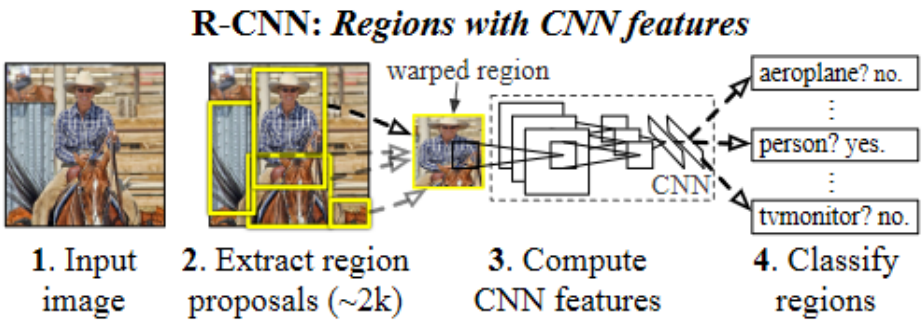


Figure 2.3 : architecture de model R-CNN [33]

2.2. Fast R-CNN :

Ross Girshick [35] à proposer un nouvel algorithme d'apprentissage qui corrige les inconvénients du R-CNN et du SPPNet, tout en améliorant leur vitesse et leur précision. Nous appelons cette méthode le R-CNN rapide (Fast R-CNN) car elle est comparativement rapide à entraîner et à teste.

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

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

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

Le Fast R-CNN obtient un résultat de (66.1%) sur le VOC2010 et le meilleur résultat sur le VOC12 avec un a mAP de 65,7 % (et 68,4 % avec des données supplémentaires). [35]

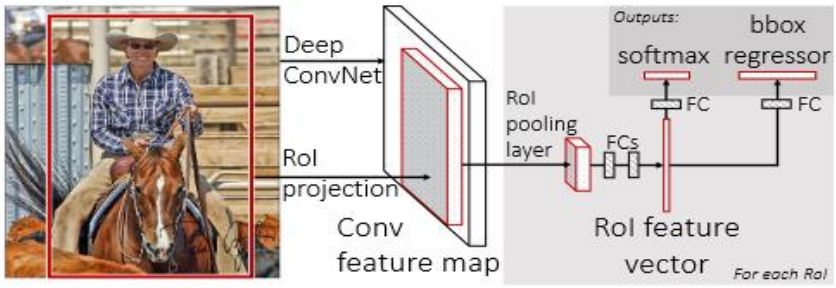


Figure 2.4: architecture de model Fast R-CNN [35]

2.3. Faster R-CNN :

Ce réseau proposer par Shaoqing Ren et al en 2016 [38], 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 [38] comme nous avons vu dans l'architecture fast R-CNN.

Le résultat obtenu par ce model avec la base de données mscoco (COCO val) avec mAP@[0.5] c'est 41.5% et on COCO test-dev avec mAP@[0.5] c'est 42.7% et avec mAP@[.5, .95] c'est 21.9.

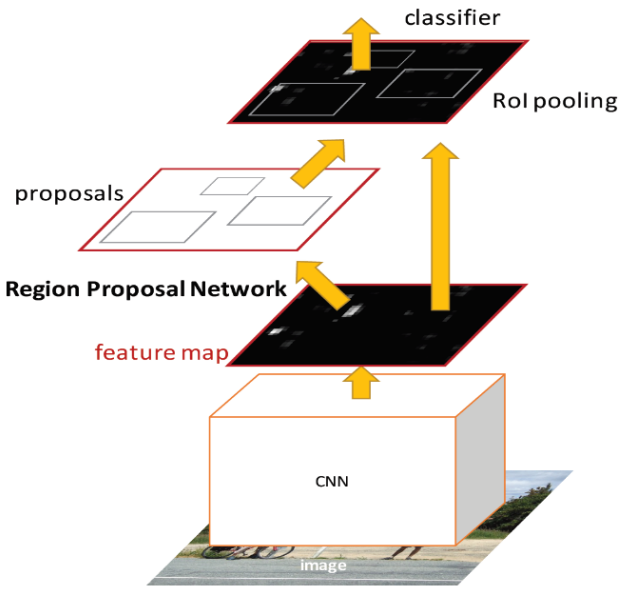


Figure 2.5: architecture de model Faster R-CNN [38]

2.4. Mask R-CNN :

Kaiming He et al [34] ont proposés une autre architecture appelé Mask R-CNN, étendu de Faster R-CNN en ajoutant une branche pour la prédiction d'un masque d'objet en parallèle avec la branche existante pour la reconnaissance de la boîte englobante.

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

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

Le résultat obtenu sur la base de données MS COCO + fine est 36.4 mAP sur la validation en utilisant ResNet-50-FPN.[34]

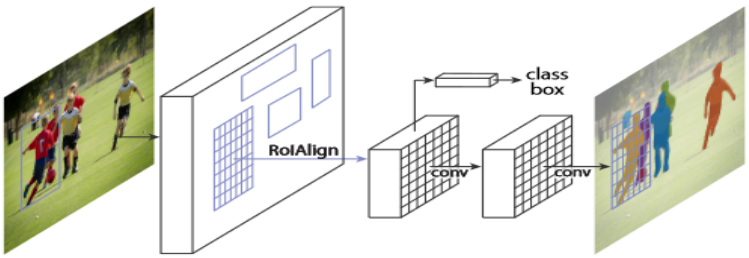


Figure 2.6: architecture de model Mask R-CNN [34]

2.5. SSD : Single Shot MultiBox Detector :

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

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

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

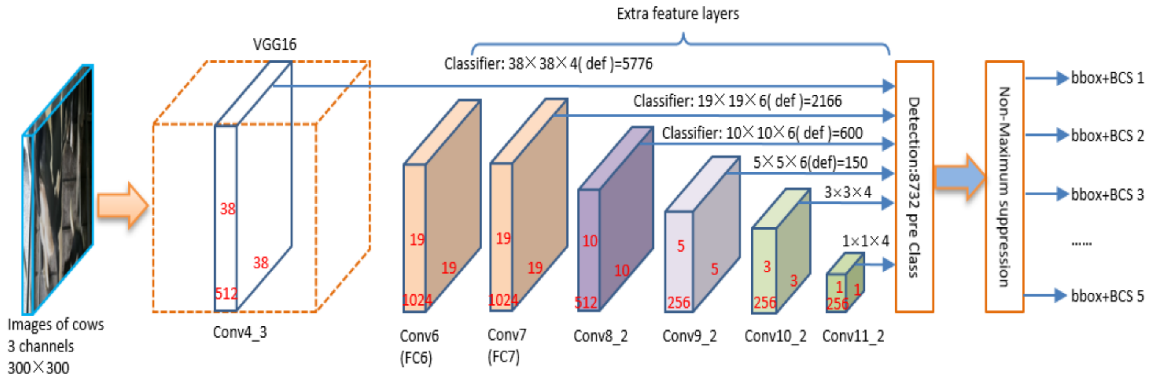


Figure 2.7 : architecture de model SSD [40]

Le résultat obtenu du modèle SSD512 entrainer sur COCO trainval35k puis affiner en pascal voc 2007+2012 c'est le meilleur résultat : 81,6% mAP, et 68.0% mAP avec le model SSD300 apprend sur voc 2007, et le résultat obtenu des modelés SSD300 et SSD512 entrainer sur la base de données COCO trainval35k est 41.2% mAP et 46.5% mAP.[39]

2.6. YOLO: You Only Look Once:

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

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.[41]

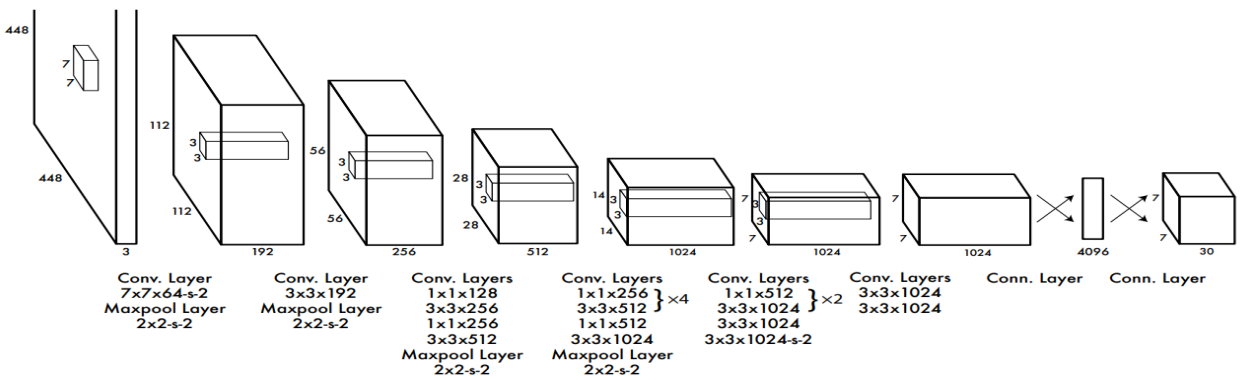


Figure 2.8 : Architecture de model YOLO [41]

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 [41].

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

- **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.

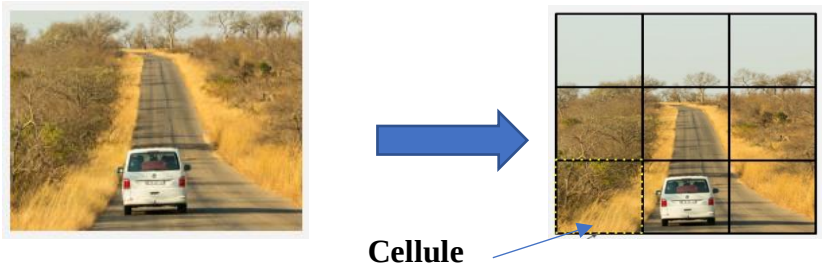


*Figure 2.9: Diviser l'image en (S*S) grille*

- **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 [41].

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 :

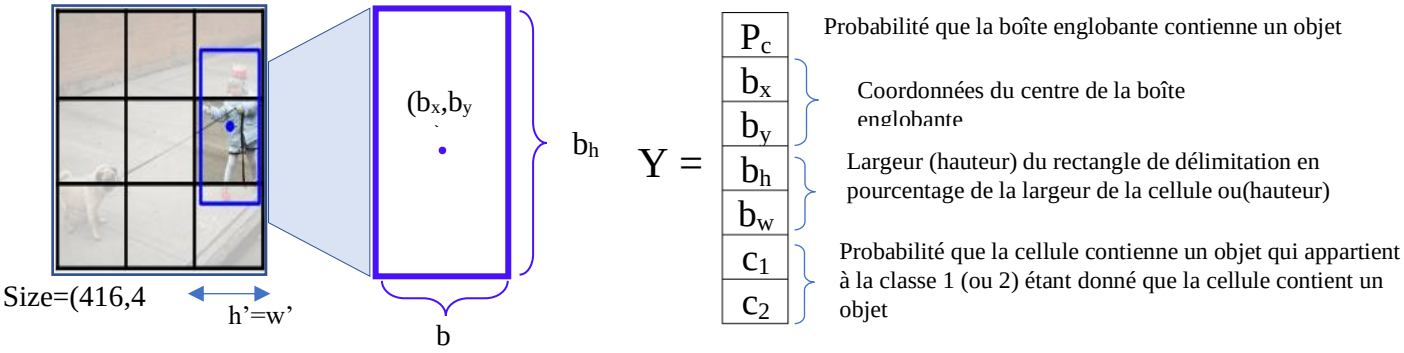


Figure 2.10: le vecteur prédit dans le cas d'une seule boite

Les valeurs de vecteur Y sont calculées au format YOLO :

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

$b_x = (x - h')/h'$, $b_y = (y - w')/w'$, $b_h = h/416$, $b_w = w/416$

a) Intersection sur Union (IoU) : L'intersection sur l'union (IoU) est la métrique d'évaluation de facto utilisée dans la détection d'objets. Elle est utilisée pour déterminer les vrais positifs et les faux positifs dans un ensemble de prédictions. Lorsqu'on utilise l'IoU comme mesure d'évaluation, il faut choisir un seuil de précision [42]**.** Il compare la boîte prédite avec la boîte détectable et peut calculer la surface comme suit :

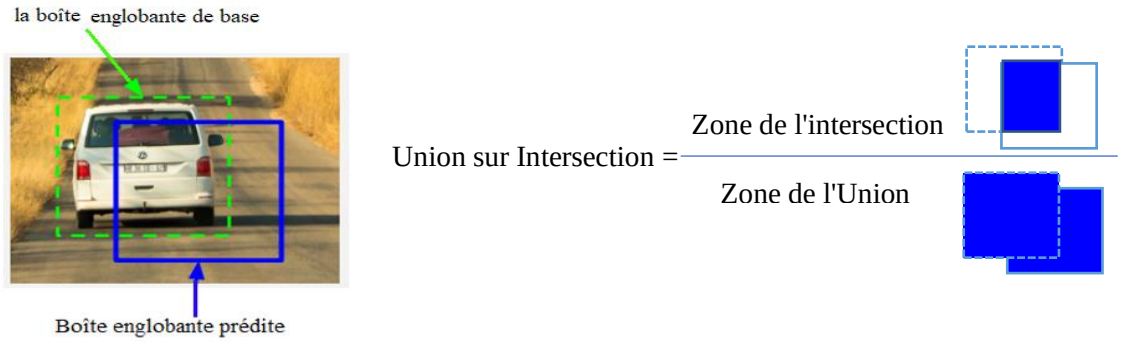


Figure 2.11 : Union sur Intersection

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

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

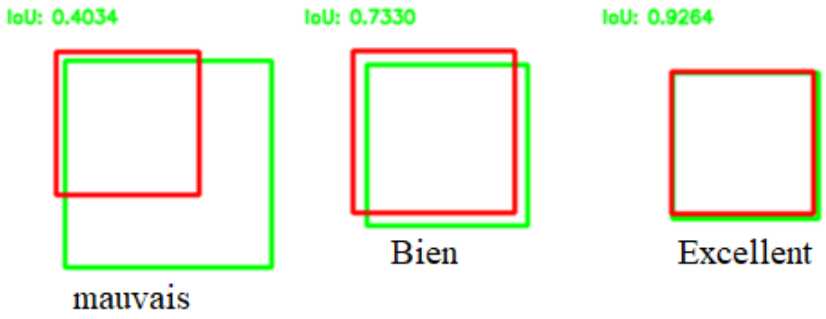


Figure 2.12 : Exemples d'IoU : courtoisie

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.[43]

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 :

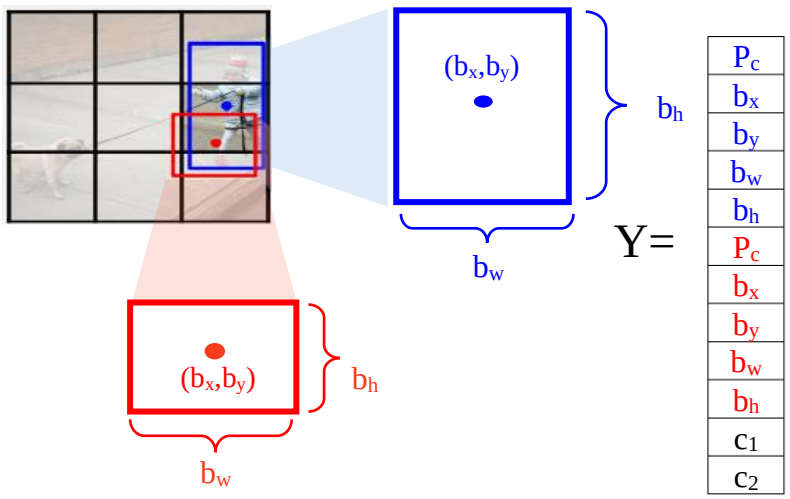


Figure2.13 : le vecteur prédit dans le cas de plusieurs boites dans la cellule

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

$S \times S \times (B * 5 + C)$. [41]

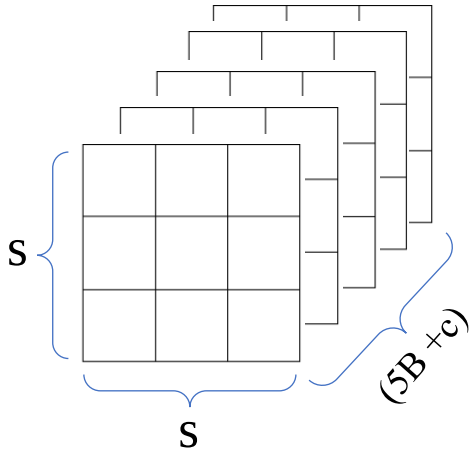


Figure 2.14 : Un tenseur qui spécifie les emplacements de la boîte englobante et probabilités de classe.

c) **Suppression non maximale :** cette étape c'est la dernière étape dans l'algorithme de détection, elle est utilisée c'est le même objet dans l'image détecter par plusieurs 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 [44] :

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

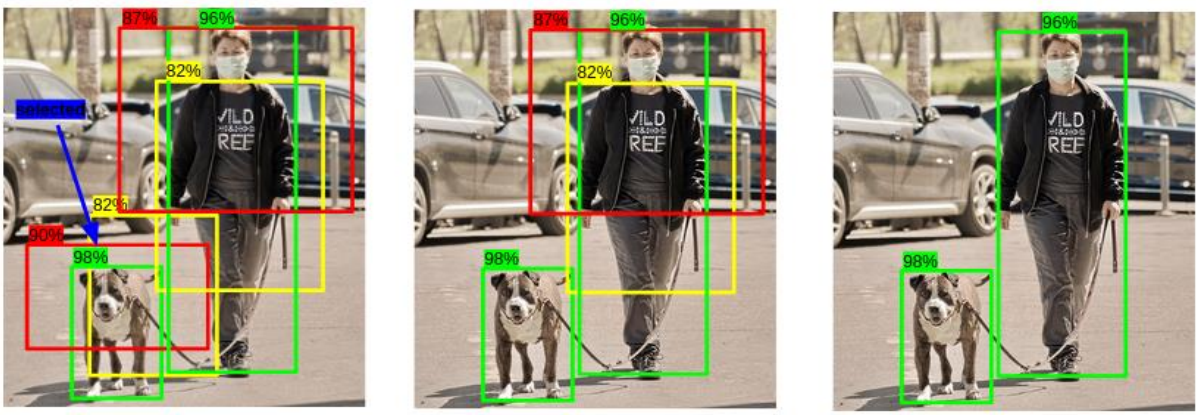


Figure 2.15 : Le résultat de Suppression non maximale

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

2.6.1. Le model yolov2 :

dans cette version, Joseph Redmon et Ali Farhadi [53] 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.[53]

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

Figure 2.16 : Architecture de darknet19 [53]

2.6.2.Le model yolov3 :

Joseph Redmon et Ali Farhadi en 2018 [46] on 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.

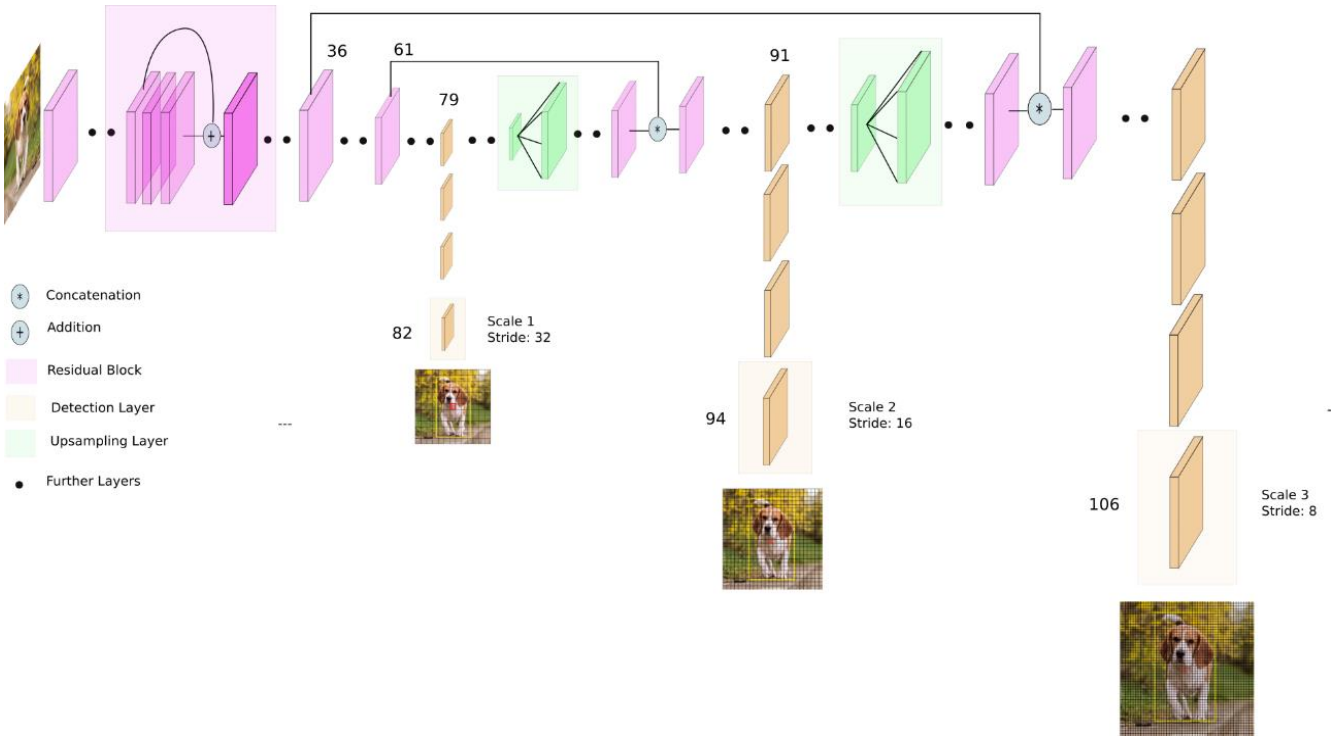


Figure 2.17: Architecture de model yolov3

2.6.3.Le model 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 :

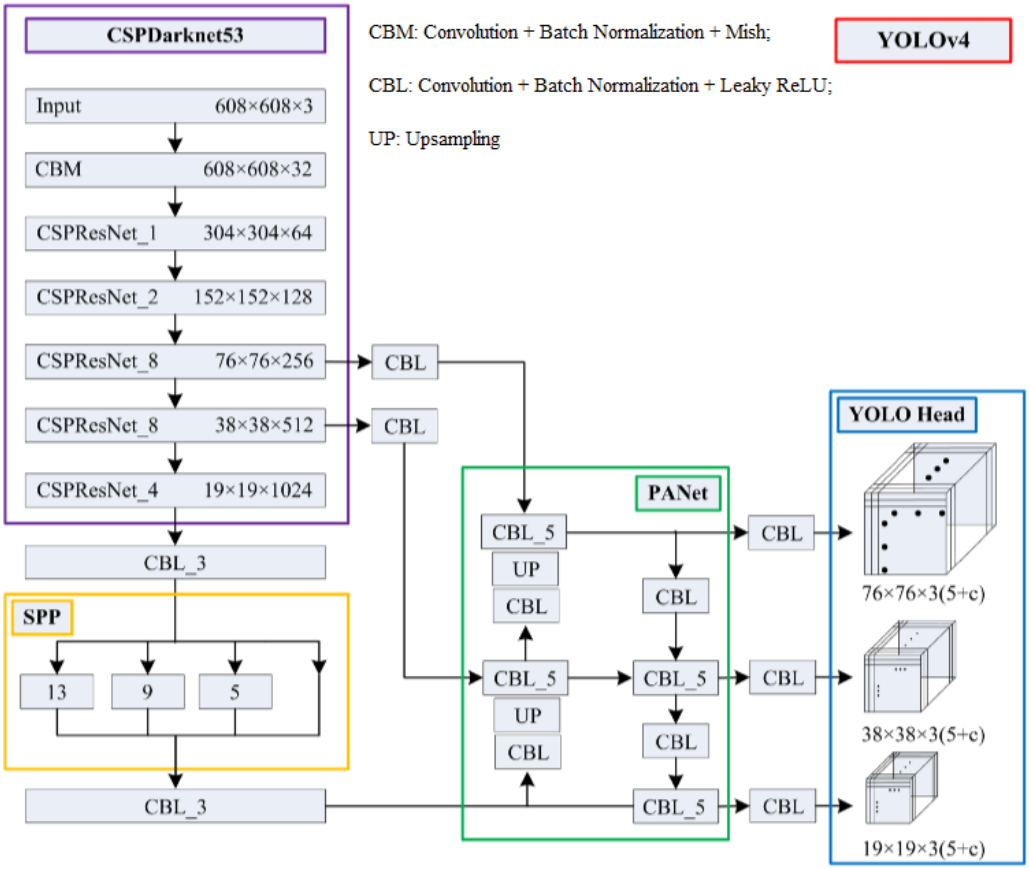


Figure 2.18 Architecture de model yolov4

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.[52]

- **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 [48], bloc SAM [50], bloc d'agrégation de chemins PAN [49], DIoU-NMS. [51]

a) CSPDarknet53 : est un réseau neuronal convolutif et une colonne vertébrale pour la détection d'objets qui utilise **DarkNet-53 [46].** Il utilise une stratégie CSPNet [45] pour partitionner la carte de caractéristiques de la couche de base en deux parties, puis les fusionne par le biais d'une hiérarchie transversale. L'utilisation d'une stratégie de partition et de fusion permet un flux de gradient plus important à travers le réseau.[47]

	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 2.19 Architecture de darknet53[46]

b) La pyramide spatiale Couche de mise en commun (SPP) : Les couches convolutives acceptent des entrées de taille arbitraire, mais elles produisent des sorties de taille variable. Les classificateurs (SVM/softmax) ou les couches entièrement connectées nécessitent des vecteurs de longueur fixe. De tels vecteurs peuvent être générés par l'approche du sac de mots (BoW) qui regroupe les caractéristiques. La mise en commun par pyramide spatiale, améliore l'approche BoW en ce sens qu'elle peut maintenir l'information spatiale par la mise en commun dans des bacs spatiaux locaux. Ces bacs spatiaux ont des tailles proportionnelles à la taille de l'image, de sorte que le nombre de bacs est fixe quelle que soit la taille de l'image. [48]

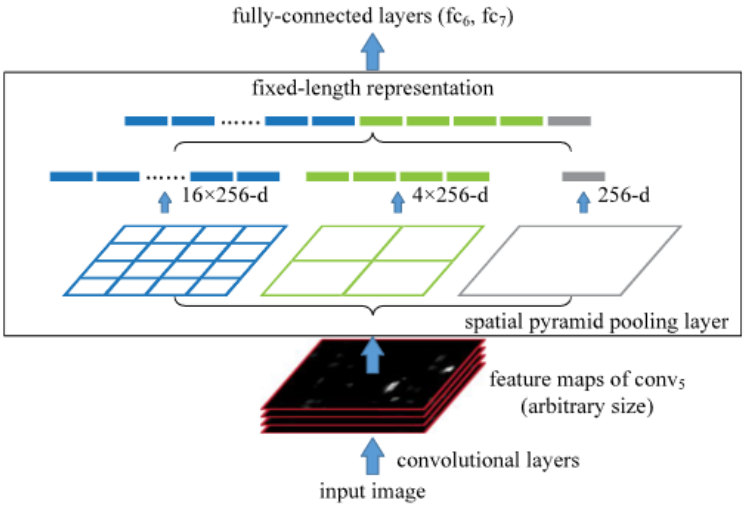


Figure 2.20 Une structure de réseau avec une couche de mise en commun pyramidale spatiale [48]

c) **Réseau d'agrégation de chemins (Path Aggregation Network PAN) :** ce réseau est proposé pour la segmentation d'instances. Ce réseau est suivie par quatre étapes : sur les deux premiers étapes (a et b dans la figure 2.21) on a créé une augmentation du chemin de bas en haut (b) pour raccourcir le chemin de l'information et améliorer la pyramide des caractéristiques avec des signaux de localisation précis existant dans les niveaux inférieurs dans 3 - ème étape faire un regroupement adaptatif des caractéristiques pour récupérer le chemin d'information brisé entre chaque proposition et tous les niveaux de caractéristiques.

Enfin, pour capturer différentes vues de chaque proposition, ils augmentent la prédiction du masque avec de minuscules couches entièrement connectées (FC), en fusionnant les prédictions de ces deux points de vue, la diversité de l'information augmente et des masques de meilleure qualité sont produits. [49]

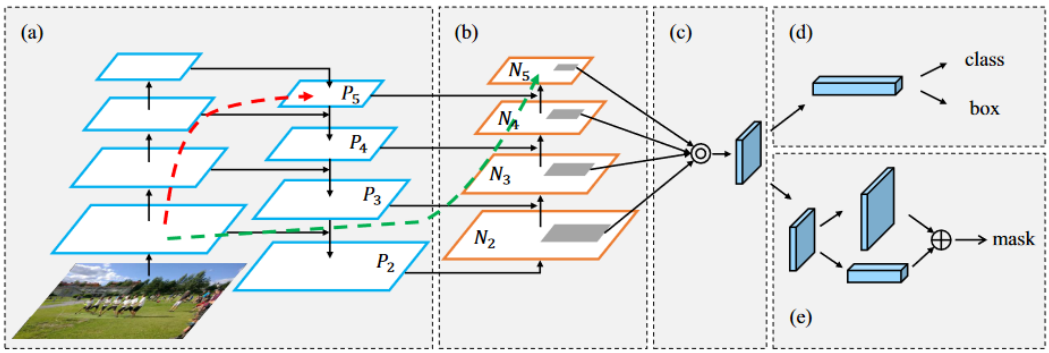


Figure 2.21 architectures de* Réseau d'agrégation de chemins *(PAN) [49]

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 2.22 [51].

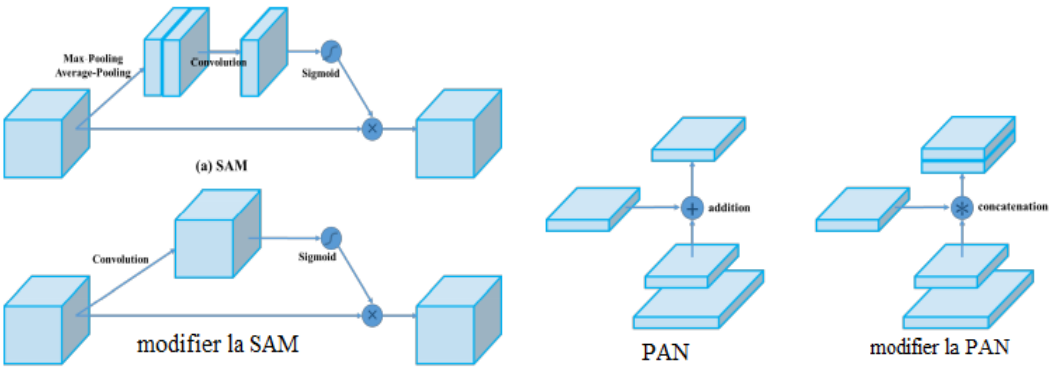


Figure 2.22 Modification des SAM et PAN

2.6.4. Le model yolov5 :

Ce model est controversé parce qu'aucun document n'a encore été publié (jusqu'au moment où nous écrivons ces lignes) par son auteur Glenn Jocher pour que la communauté scientifique puisse évaluer son benchmark. Il ne semble pas non plus avoir mis en œuvre de nouvelles techniques pour se revendiquer comme la prochaine version de YOLO. Il est plutôt considéré comme l'extension PyTorch de YOLOv3.

- ✓ **YOLOv5 Backbone :** Il utilise CSPDarknet pour l'extraction de caractéristiques à partir d'images composées de réseaux partiels à plusieurs niveaux.
- ✓ **YOLOv5 Neck :** Il utilise PANet pour générer un réseau de pyramides de caractéristiques pour effectuer l'agrégation des caractéristiques et le transmettre à Head pour la prédiction.
- ✓ **YOLOv5 Head :** Couches qui génère des prédictions à partir des boîtes d'ancrage pour la détection d'objets.
- ✓ **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.[55]

2.6.5.Le model yolovX :

Ce model basé sur l'architecture de yolov3, où il utilise darknet53 comme backbone et une couche SPP, deux expériences analytiques indiquent que le couplage tête de détection couplée peut nuire aux performances :

- ✓ Remplacement de tête de YOLO par une tête découplée améliore considérablement la vitesse de convergence.
- ✓ La tête découplée tête découplée est essentielle à la version de bout en bout de YOLO.

Faire une modification légèrement certaines stratégies d'entraînement par rapport à l'implémentation originale, en ajoutant la mise à jour des poids *EMA*, le programme cosinus *LR*, la perte IoU et la branche IoU-aware. Utilisé la perte *BCE* pour l'entraînement des branches *cls* et *obj*, et la perte IoU pour l'entraînement de la branche *reg*. [54]

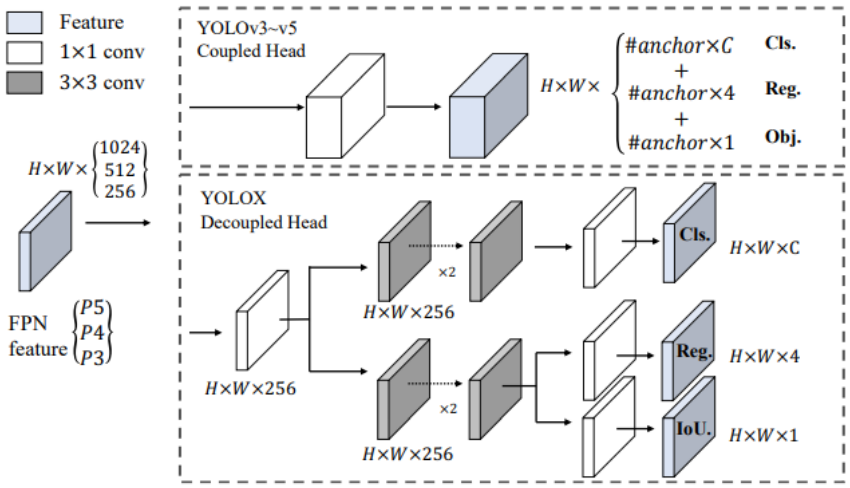


Figure 2.23 Illustration de la différence entre la tête YOLOv3 et la tête découplée proposée [54].

Dans le ***Tableau 2.1, on présente quelques*** résultats sur diffèrent version de yolo :

Model yolo	Dataset	Map	Size	FPS
Yolov1	Pascalvoc 07+12	63.4%	448	45
Yolov2	Pascalvoc 07+12	78.6%	544	40
Yolov3	MS COCO	33.0%	608	20
Yolov4	MS COCO	43.5%	608	23
Yolov5-x	MS COCO	50.4%	640	62.5
Yolovx-x	MS COCO	51.2%	640	57.8

Tableau 2.1 les résultats de défèrent version yolo

3. Mesurer la performance d'un modèle de détection :

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

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

Exemple :

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

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

$$\boldsymbol{précision} = \frac{\boldsymbol{vrai\ positif}}{\boldsymbol{vrai\ positif} + \boldsymbol{faux\ positif}}$$

b) Rappel : Le rappel est un indicateur qui mesure la capacité du modèle à prédire l'ensemble des résultats attendus**.** Calculer par [56]:

$$\boldsymbol{Rappel} = \frac{\boldsymbol{vrai\ positif}}{\boldsymbol{vrai\ positif} + \boldsymbol{faux\ négatif}}$$

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

$$\text{accuracy} = \frac{\boldsymbol{vrai\ positif + vrai\ négatif}}{\boldsymbol{total}}$$

4. 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 model 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 6 version de YOLO et la différence entre chaque version et basé beaucoup plus sur la version YOLOv4 qui il utilise dans notre conception , on conclure avec la mesure de performance d'un model de détection.

Chapitre 03 : Conception et Implémentation

1. Introduction :

Nous avons présenté dans le premier chapitre les principes fondamentaux de l'apprentissage automatique et l'apprentissage profond avec les benchmarks base de données de détection d'objet et dans le deuxième chapitre nous avons vu les différentes techniques ou bien les différentes architectures des modèles de détection d'objet en temps réel.

D'après les modelés de détection nous avons vu le modèle YOLO avec ses différents algorithmes, alors nous avons choisir ce modelé pour notre conception, pour résoudre notre problème qui se résume en la création d'un système de détection d'objet pour aider les personne aveugle, nous avons choisir le modèle yolov4, que nous avons reconfigurer selon nos besoins, puis ré-entraîner sur certains classes sélectionnés pour notre contexte à partir de la base de données pascal voc2012 et la base de données ExDark.

Utilisation du langage de programmation Python dans l'environnement de programmation libre de Google Colab. Ce chapitre se concentre sur la conception de notre approche ainsi que les détails de notre model choisir yolov4 avec l'expérimentation et les résultats.

2. Conception01 :

2.1. Schéma de conception

Notre système se compose de 3 modules principales:

- Prétraitement de données
- Apprentissage de model yolov4 sur nos données choisis
- Test et résultat

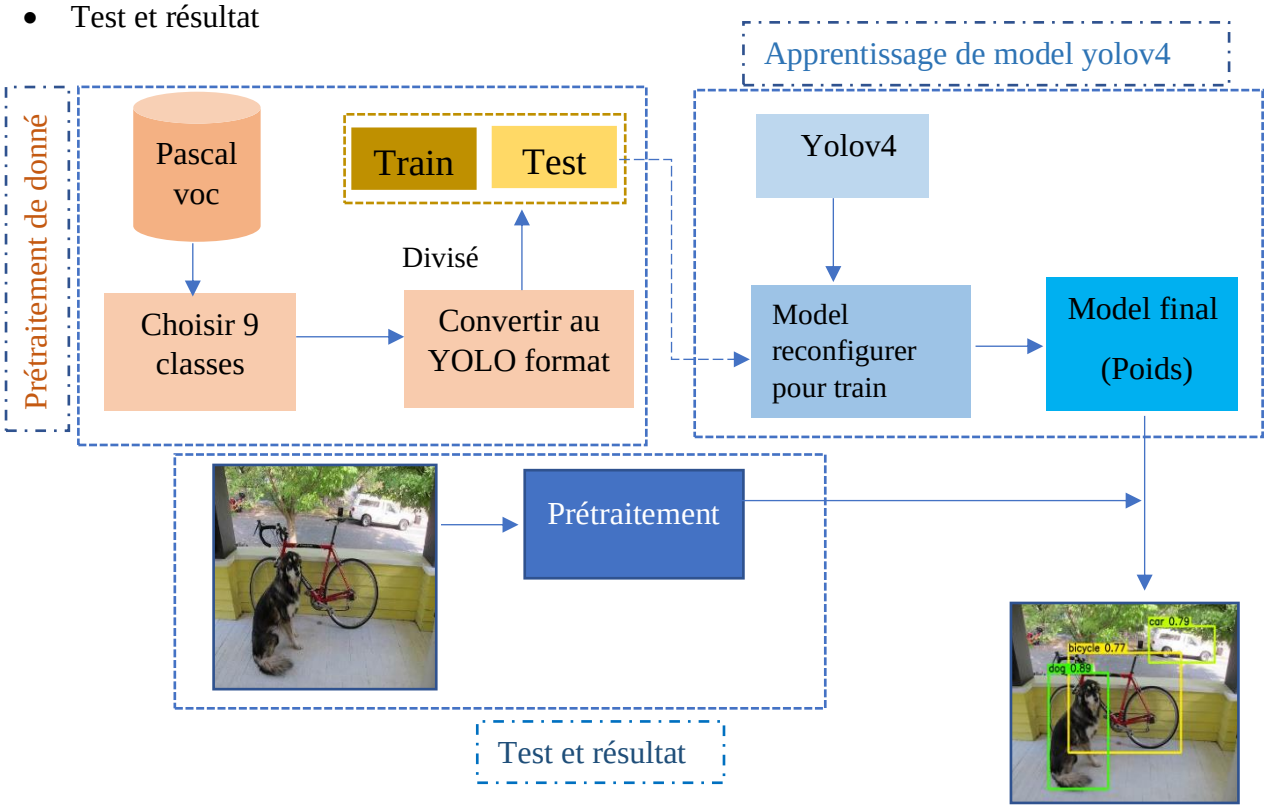


Figure3.1 architecture générale de notre conception

2.1.1. Le choix de la base de données :

Le choix de la base de données n'était pas facile, car nous avons d'abord essayé de constituer une base de données à travers les benchmarks base dans la détection d'objet, et cette idée sera pour nous une base de données contenant un grand nombre d'image, aussi pour augmenter les possibilités des positions et la forme des objets choisi, dans notre travail c'est les objets qui sont à l'intérieur de la maison, pour lui dicter les noms des objets pour lui faciliter la vie.

Mais nous n'avons pas pu, car dans la première idée nous avons essayé de combiner entre la base e données PASCAL VOC et MSCOCO mais nous avons remarqué qu'il y a une classe manquante dans le fichier d'annotation de la base dans la base Mscoco, et aussi il existait quelques annotations qui n'étaient pas correctes (la figure 3.2 les exemples d'annotation incorrecte), et pour éviter les erreurs au cours de l'apprentissage on a annulé cette idée.

Nous avons aussi essayé de combiner la base PASCAL VOC avec OpenImage ou bien Image Net, mais à cause de la grande taille des deux bases, nous ne pouvions pas télécharger et donc nous ne pouvions pas crées des nouvelles sous-classes pour augmenter le nombre des classes existant.

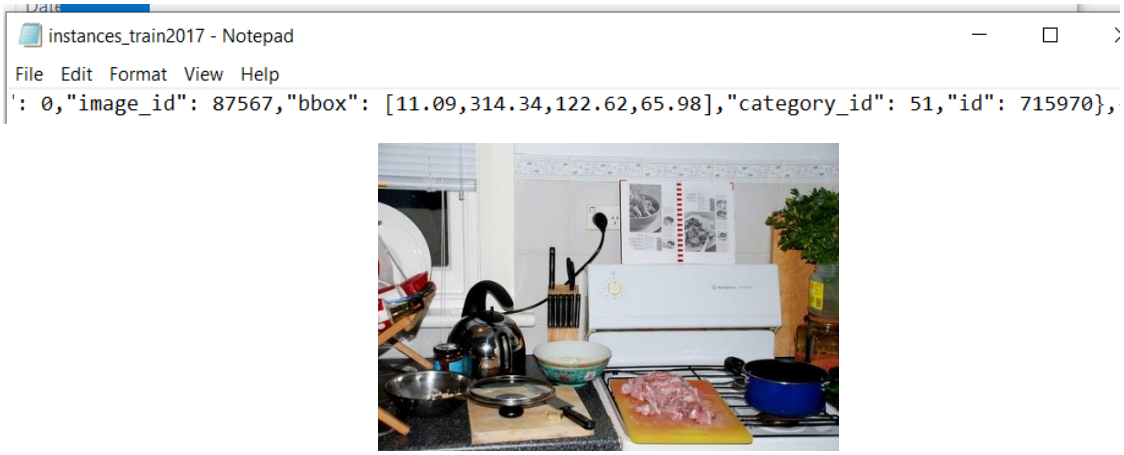


Figure3.2 images de la base de données mscoco avec mauvaise annotation

Dans la première photo la classe indiquée dans le fichier d'annotation c'est la classe 51 (normalement broccoli) alors que le broccoli n'existe pas dans cette photo.
La classe indiquée dans le fichier d'annotation c'est la classe 67 (normalement keyboard) alors que le Keyboard n'existe pas dans cette photo.

2.1.2. Prétraitement de donnée :

1) Choisir 09 classe

Cette étape consiste à appliquer quelque traitement sur la base de données PASCAL VOC, tel que le choix des 9 classe (qu'on trouve a l'intérieur de la maison), après convertir l'annotation de chaque image au format d'annotation YOLO.

La création des 9 sous-classe de pascal voc (bottle, cat, chair, diningtable, dog, sofa, potted-plant, tv-monitor, Person), consiste à lire chaque fichier d'annotation (.xml) ; après tester si l'objet n'existe pas dans la liste de nos classes choisi, alors elle sera retiré du fichier annotation (13112 images).

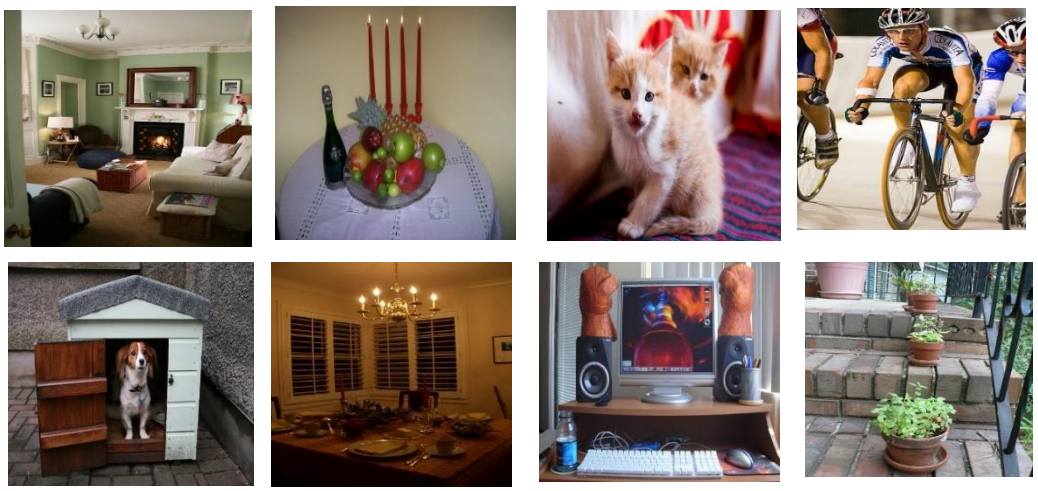


Figure3.3 Exemple d'images de la base de données

2) Convertir au format yolo :

La deuxième étape consiste à créer l'annotation des sous-classes choisis sous le format yolo, Les valeurs de la boîte englobante de model yolo sont normalisées entre 0 et 1. Chaque fichier d'annotation (par exemple 2007_000027.txt) contient des lignes où chaque ligne représente un objet et chaque objet représenté par numéro de la classe et boîte de englobante dans l'image. Une ligne a le format suivant :

<Numéro de classe> <centre_x> <centre_y> <largeur> <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

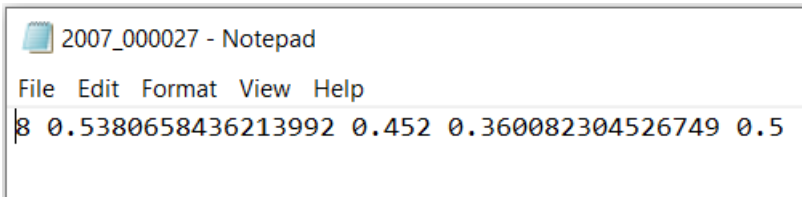


Figure 3.4 Exemple d'annotation de la base de données

3) Diviser la base (train et test) :

La dernière étape dans ce module c'est la division des données en deux parties, un pour l'apprentissage (train.txt), et autre pour le test (test.txt), ces deux fichiers contiennent les liens (paths) des images de l'apprentissage et de test, dans notre cas nous avons choisi 70% pour l'apprentissage et 30% pour test.

2.1.3. Apprentissage de model yolov4 :

Ce module contient deux étapes principales sont :

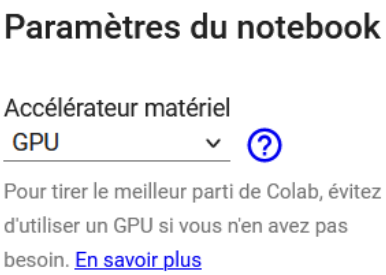
1) Télécharger le modèle yolov4 :

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

```
[ ]  !git clone https://github.com/AlexeyAB/darknet
```

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

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



- Accéder au fichier darknet après, faites des changements dans le makefile pour activer OPENCV et le GPU et CUDA

```
[4]  %cd darknet/
     !sed -i 's/OPENCV=0/OPENCV=1/' Makefile
     !sed -i 's/GPU=0/GPU=1/' Makefile
     !sed -i 's/CUDNN=0/CUDNN=1/' Makefile
     !sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
     !sed -i 's/LIBSO=0/LIBSO=1/' Makefile
```

Après la configuration de *Colab*, nous avons télécharger la base de données dans un dossier appeler **« data »**, après crées deux fichier **« dataset.name »** qui contient les nomes de nos classes, et **« dataset.data »** qui contient les chemins (paths) au fichier « **train.txt »** et « **test.txt »** et **« dataset.name » et** l'emplacement du **répertoire « backup »** pour enregistrer les poids **(weights)** de model au cours de l'entrainement.

Télécharger les poids pré-entraînés de YOLOv4, au lieu d'entraîner un modèle à partir de zéro, nous avons utilisé les poids « **yolov4.conv.137** » qui ont été entraînés jusqu'à 137 couches convolutives.

2) Préparer le model pour le train :

Cette étape consiste à changer les paramètres dans le fichier de configuration « **yolov4-custom.cfg** » de modèle pour adapter notre base de données choisi, les paramètres qui sont changés étaient calculer comme suite :

- Le nombre de batch et subdivision (batch = 64, subdivision = 16) ou bien une valeur multiple à 32
- Défini la taille de model (width=416, height=416)
- Changer la ligne max_batches en (classes*2000 donc 9*2000 =18000)
- Changer les étapes de la ligne à 80% et 90% de max_batches (steps=14400,16200)
- Remplacez la ligne classes=80 par votre nombre d'objets dans chacune des 3 couches [yolo] : classes = 9
- Changez [filters=255] en filters = (classes + 5) x 3 dans les 3 [convolutions] avant chaque couche [yolo], gardez à l'esprit qu'il doit seulement s'agir de la dernière [convolution] avant chacune des couches [yolo]. (filtres = (9+5) *3=42)).

L'étape suivante c'est exécutez la command « **!make** » pour construire les fichier dans darknet, ensuite la command «**!chmod +x./darknet** » pour autoriser à utiliser le fichier darknet.exe qui permet de lancer l'apprentissage.

L'apprentissage de model yolo lancer par une commande qui a besoins de paramètres comme arguments d'entrer qui sont le nom de fichier « **dataset.data** » et le fichier de

configuration (*. cfg) , les poids pré-entraînés (pour la première fois on utilise **yolov4.conv.137 et après on utilise toujours les best**.weights), ce changement de nom du fichier des poids est dû aux coupures et aux interruptions causés par internet et par la politique de colab qui arrête tous les ressources chaque fois a mi-nuit.

```
!./darknet detector train data/dataset.data cfg/yolov4-custom.cfg backup/yolov4-custom_best.weights -dont_show  -map
```

L'utilisation du flag '**dont_show**' car il n'y a pas d'écran d'affichage dans *google Colab* pour tracer le graph de mAP dans l'apprentissage. La valeur de perte affiche chaque itération de l'apprentissage et la valeur de mAP elle est calculer dans chaque 4 epoch.

```
 (next mAP calculation at 18477 iterations)
 Last accuracy mAP@0.5 = 67.46 %, best = 67.46 %
 17999: 2.384780, 1.768936 avg loss, 0.000010 rate, 24.262146 seconds, 1151936 images, 0.875089 hours left
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.606549), count: 15, class_loss = 5.827608,
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.740774), count: 31, class_loss = 5.658989,
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.857157), count: 28, class_loss = 4.065880,
 total_bbox = 510204, rewritten_bbox = 1.158752 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.624856), count: 7, class_loss = 1.780611, :
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.833651), count: 27, class_loss = 2.955473,
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.861049), count: 39, class_loss = 3.057359,
 total_bbox = 510277, rewritten_bbox = 1.158586 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.806870), count: 1, class_loss = 0.095987, :
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.864714), count: 19, class_loss = 1.966528,
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.876799), count: 35, class_loss = 1.962772,
 total_bbox = 510332, rewritten_bbox = 1.158462 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.780949), count: 15, class_loss = 1.871289,
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.848270), count: 38, class_loss = 4.036150,
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.879800), count: 35, class_loss = 4.023078,
 total_bbox = 510420, rewritten_bbox = 1.158262 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.723230), count: 11, class_loss = 2.924998,
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.756549), count: 51, class_loss = 9.384042,
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.821695), count: 65, class_loss = 6.198361,
 total_bbox = 510547, rewritten_bbox = 1.158365 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.796097), count: 7, class_loss = 0.624660, :
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.842253), count: 26, class_loss = 1.222370,
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.866419), count: 39, class_loss = 1.775840,
 total_bbox = 510619, rewritten_bbox = 1.158202 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.000000), count: 1, class_loss = 0.000028, :
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.831938), count: 5, class_loss = 0.393955, :
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.882312), count: 13, class_loss = 0.175718,
 total_bbox = 510637, rewritten_bbox = 1.158161 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.815577), count: 2, class_loss = 0.013951, :
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.893796), count: 19, class_loss = 0.156280,
```

Figure 3.5 l'affichage du processus de l'apprentissage.

Le résultat de l'apprentissage obtenu avec mAP@0.5 c'est 67.52%, dans le tableau suivant on a affiché tous les résultats obtenus :

Précision	***Rappel***	***F1-score***	***average IoU***
0.75	0.73	0.74	63.07 %

Tableau 3.1. Les résultats de l'apprentissage

Dans le tableau suivant nous avons les résultats obtenu pour chaque class :

Class	***Image par classe***	***précision***	***Vrai positive***	***Faux positive***
bottle	812	52.93%	292	116
cat	1128	90.05%	385	37

chair	1366	55.60%	544	350
dining table	691	55.67%	148	90
dog	1341	87.08%	443	84
sofa	742	60.46%	160	118
potted plant	613	54.44%	221	127
TV monitor	645	71.08%	176	53
Person	9583	80.34%	4699	1361

Tableau 3.2. Les résultats de l'apprentissage de chaque class

2.2. L'implémentation :

Pour ré-entrainer le model yolov4 sur les sous-classes la base de données PASCAL VOC nous avons utilisé l'environnement libre de *google Colab* aussi des libraires opencv en langage python.

2.2.1. L'Environnement google Colab :

Google Colab est un produit de Google, comme son nom l'indique. Il s'agit essentiellement d'un environnement de bloc-notes gratuit qui fonctionne entièrement dans le nuage. Il dispose de fonctionnalités qui vous aident à modifier des documents de la même manière que vous travaillez avec Google Docs. Colab prend en charge de nombreuses bibliothèques d'apprentissage automatique populaires et de haut niveau qui peuvent être facilement chargées dans votre notebook.

Google Colab nous offre trois types de runtime pour nos ordinateurs portables : CPUs, GPUs, et TPUs, Colab nous offre un total de 12 heures d'exécution continue. Après cela, toute la machine virtuelle est effacée et nous devons repartir de zéro. à cause de limite du l'utilisation des ressource *google Colab*.

Nous pouvons exécuter plusieurs instances CPU, GPU et TPU simultanément dans Google collab, mais les ressources sont partagées entre ces instances [76].

Pour notre apprentissage nous avons utilisé les ressources de Colab avec RAM 12.69 GB et Disque de taille 107.72 GB, la version de GPU c'est Tesla k80, CUDA-version est 11000, cuDNN: 7.6.5.

2.2.2. Python :

Python est un langage de programmation de haut niveau, polyvalent et très populaire. Le langage de programmation Python (le dernier Python 3) est utilisé dans le

développement Web, les applications d'apprentissage automatique, ainsi que dans toutes les technologies de pointe de l'industrie logicielle. Le langage de programmation Python convient très bien aux débutants, ainsi qu'aux programmeurs expérimentés dans d'autres langages de programmation comme C++ et Java [75].

Figure 3.6 logo python

2.2.3. Opencv :

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

La bibliothèque compte plus de 2500 algorithmes optimisés, ce qui inclut un ensemble complet d'algorithmes de vision par ordinateur et d'apprentissage automatique classiques et de pointe. Ces algorithmes peuvent être utilisés pour détecter et reconnaître des visages, identifier des objets, classer des actions humaines dans des vidéos, suivre les mouvements de la caméra, suivre des objets en mouvement, extraire des modèles 3D d'objets, etc. **[77]**

Dans le processus de l'apprentissage nous avons utilisé la version 3.2.0.

Figure 3.7 logo OpenCV

2.2.4. Chargement de model :

Dans cette étape nous utiliserons la fonction prédéfinie de opencv « cv2.dnn.readNet » pour charger le réseau en mémoire. Elle détecte automatiquement la configuration et le cadre en fonction du nom du fichier spécifié. Dans notre cas, il s'agit d'un fichier «. weight »

Nous avons créé une liste qui contient les labels des objets spécifiques que nous avons choisi (9 sous classes). Après nous avons récupéré les couches de notre modèle pour obtenir les couches de sortie. Par la fonction « getUnconnectedOutLayers())

2.2.5. Prétraitement de l'image de test :

Nous utilisons la fonction « cv2.dnn.blobFromImage » qui renvoie un blob qui est notre image d'entrée après soustraction de la moyenne, normalisation et échange de canaux.

- Pour la soustraction de la moyenne nous utilisons « np.mean » puis on soustrait chaque pixel de l'image par cette moyenne,
- Après avoir effectué la soustraction de la moyenne, nous pouvons mettre à l'échelle nos images par un certain facteur. Cette valeur est par défaut de `1.0` (c'est-à-dire, pas de mise à l'échelle) mais nous pouvons également fournir une autre valeur (1/255).
- Resize de l'image, Nous fournissons ici la taille spatiale attendue par le réseau neuronal convolutif, dans notre cas nous utilisons (416*416).
- OpenCV suppose que les images sont dans l'ordre des canaux BGR ; cependant, le model suppose que nous utilisons l'ordre RGB. Pour résoudre cette divergence, nous pouvons intervertir les canaux R et B dans l'image en mettant la valeur de swapRB à `True`.

2.2.6. La Détection :

Après les prétraitement de l'image nous passons à la phase de détection, cette étape consiste à:

- Passez image blob de résultat de prétraitement dans l'algorithme de model yolo par « net.setInput(blob) »
- Utilisez « net.forward() » pour transmettre le blob à la couche de sortie que nous avons générer dans la première phase de chargement de modèle et générer le résultat.

Le résultat de la détection c'est tous les boites englobante détecter à la dernière couche de modèle (sur les 3 scale) qui passent le degré de confiance >0.5 avec le nom de la classe et le degré de confiance. Nous utilisons la fonction NMS dans opencv « cv2.dnn.NMSBoxes » pour effectuer la Suppression Non-Maximale. On utilise un seuil de score et un seuil de NMS comme arguments.

2.3. Test et résultat :

Dans cette partie nous allons tester/valider les performances du model yolov4 ré-entrainé sur les neufs sous classes sélectionnés (pour un contexte bien précis qui est des objets qu'on trouve à la maison) à partir de PASCAL VOC'12.

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

On peut voir certains résultats obtenus dans la figure suivante (3.8):

(a) (b)

(c) (d)

(e)

Figure 3.8 Le Résultat Obtenu de Détection par Yolov4

Nous avons faite des tests sur la base de données ExDark et nous avons obtenues les résultats suivants :

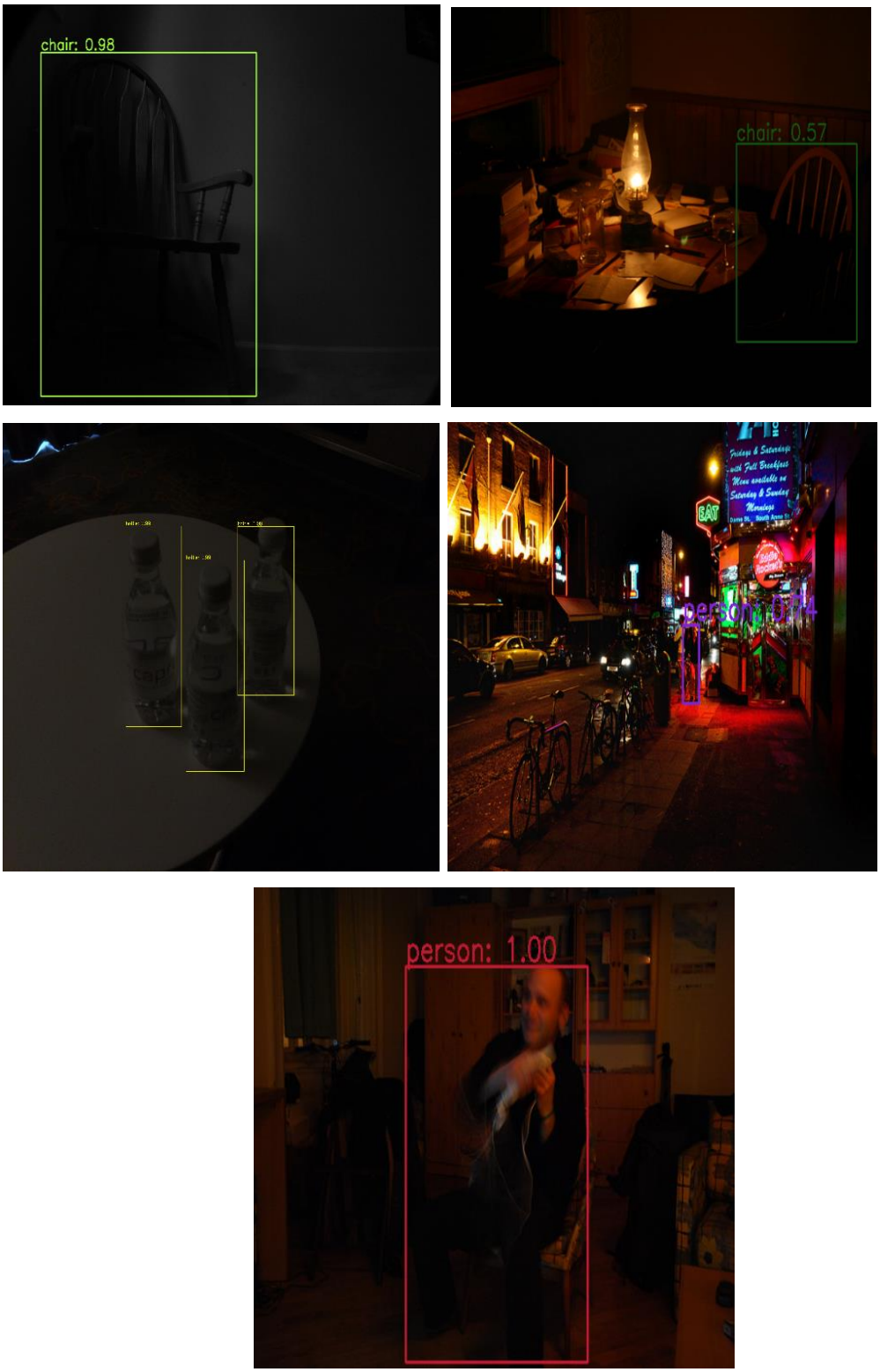


Figure 3.9 le résultat obtenu de détection par yolov4 sur ExDark

En webcam les résultats de prediction sont donnés et calculés dans 0.6 second en moyenne (avec CPU).

2.4. Discussion :

D'après les résultats obtenus (*tableau 3.2* et *figure 3.7 et figure3.8*) nous avons vu que la performance de model yolo on détection de diffèrent objet dans l'image est plus précis surtout dans les classes avec un mAP plus que 80% telq que les personnes et les chats. Aussi nous avons vu que portant la précision de l'objet 'chaise' dans le résultat de train c'est 55% mais dans le test elle donne un bon résultat de détection quel que soit la position da la chaise (de prés ou de loin, complet ou bien une partie)

Points forts :

- Le Système est puissant dans une image de complexité elevée (plusieurs class comme la figure 3.8 (d), mauvaise qualité (e), mauvaise éclairage (figure3.9), occlusion d'objet et la taille(b).
- Le Système capable à travaille sur des images ou bien vidéo/cam avec une résolution (416) (en gardant un bon rapport temps/fps).
- La Plupart des boites englobantes cadrent bien les objets avec une précision (IoU) élevé (figure 3.8, figure 3.9)
- l'Acuraccy de model est général très intéressante par apport aux modèles de détection en temps réel.

Les problèmes :

- Problème de limite de l'utilisation de gpu.
- Chaque fois il y'a initialisation des fichiers des poids sur darknet à cause de la politique de colab (historique n'est pas sauvegarder).
- Si on lance le train deuxième fois il ne termine pas l'apprentissage sur le dernier arrêt.
- Le Problème des bases de données, soit trop gros, soit problème d'étiquetage.
- L'Apprentissage prend des temps incroyables (plusieurs itérations = des semaines)

3. Conception 02 :

Dans cette partie nous avons entrainer le model yolov4 sur la base de données ExDark, le schéma suivant c'est la conception :

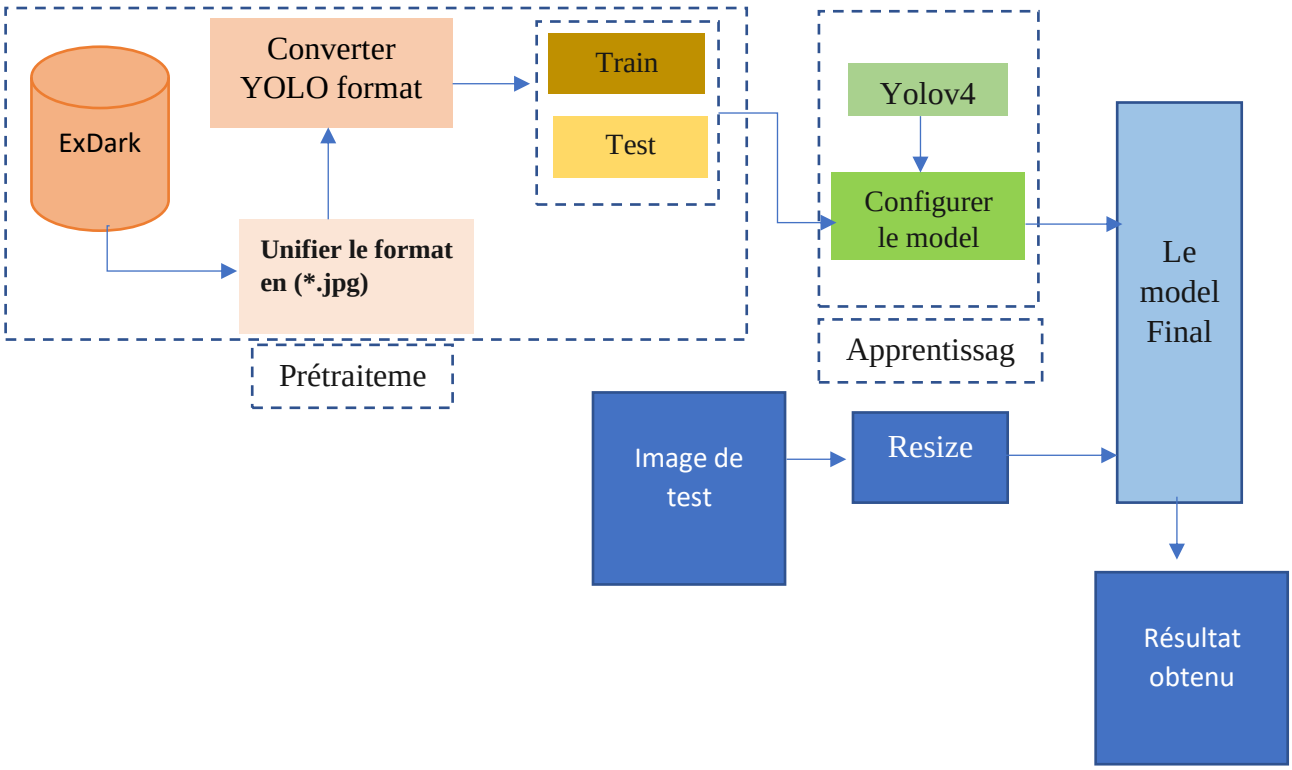


Figure 3.10 conception d'apprentissage yolo sur ExDark

3.1. Prétraitement :

3.1.1. La base de données ExDark :

La base de données (ExDARK) est une collection de 7 363 images de faible luminosité, allant d'environnements très peu éclairés jusqu'au crépuscule (soit 10 conditions différentes), avec 12 classes d'objets (similaires à PASCAL VOC) annotées à la fois au niveau de la classe d'image et des boîtes de délimitation locales des objets [57].

Les objets dans cette base diviser comme suite :

L'objet	***Nombre d'image***
Bicyclette	652
Bateau	679
Bouteille	547
Bus	527
Voiture	638
Chat	735
Chaise	648
Tasse	519
Chien	801
Moto	503

Personnes	609
Table	505

Tableau 3.3 le nombre d'image de la base de données ExDark par class

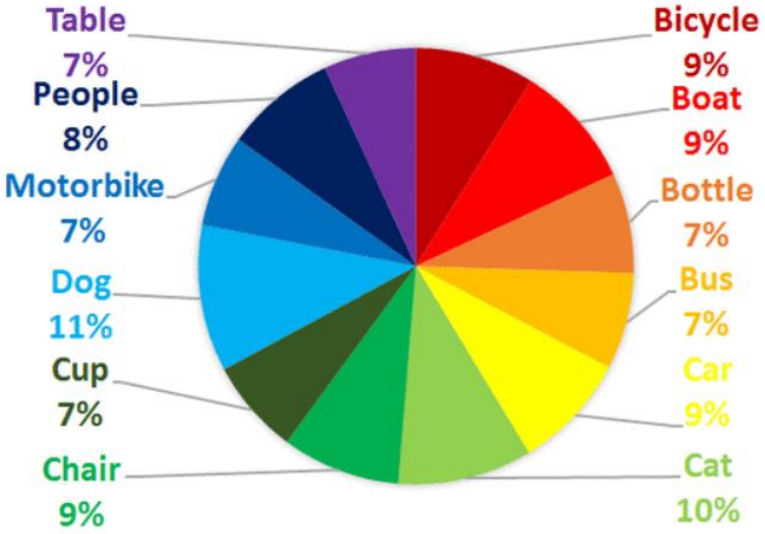


Figure 3.11 le pourcentage de nombre d'image par classe

1) **Unifier le format :**

Le format des images dans la base de données ExDark est *.jpg et *.png, alors pour faciliter la division de la base de données (apprentissage/test) on a unifier toutes les images en la même format(*.jpg)

2) **La conversion au format yolo :**

Comme nous avons fait dans la première partie, on a aussi converti le format d'annotation ExDark au format yolo, alors que l'annotation des image ExDark étaient comme la figure suivante :

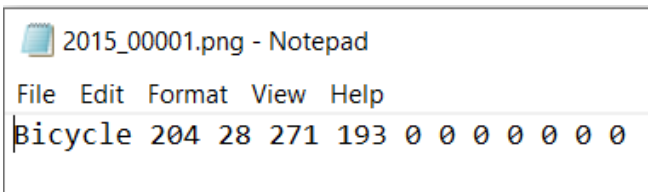


Figure 3.12 format d'annotations de fichier ExDark

Les annotations sont générées à l'aide de la Computer Vision Matlab Toolbox (PMT) de Piotr.

Le format de chaque '.txt' est le suivant :

(a) 1ère colonne : Nom de la classe d'objet

(b) 2ème à 5ème colonne : Coordonnées de la boîte englobante [l t w h].

La conversion avec la même méthode utiliser dans la partie 01.

3) **Diviser la base de données :** la même méthode utiliser dans la première partie avec le même pourcentage

4) **Configurer le model pour l'apprentissage :**

La configuration de l'environnement c'est même que la partie 01, la configuration de model c'est la même que la premier mais avec les données calculer par rapport à la base de données ExDark, alors les modifications du fichier de configuration sont :

- Le nombre de batch et subdivision (batch = 64, subdivision = 16)
- La taille de model (width=416, height=416)
- Changer la ligne max_batches en (classes*2000 donc 12*2000 =24000)
- Changer les étapes de la ligne à 80% et 90% de max_batches (steps= 19200,21600)
- Remplacez la ligne classes=80 par votre nombre d'objets dans chacune des 3 couches [yolo] : classes = 12
- Changez [filters=255] en filters = (classes + 5) x 3 dans les 3 [convolutions] avant chaque couche [yolo], gardez à l'esprit qu'il doit seulement s'agir de la dernière [convolution] avant chacune des couches [yolo]. (filtres = (12+5) *3=51)).

Le processus de l'apprentissage fait avec les même poids ré-entériner « **yolov4.conv.137** »

Le résultat de l'apprentissage obtenu avec mAP@0.5 c'est 67.52%, dans le tableau suivant on a affiché tous les résultats obtenus :

Précision	***Rappel***	***F1-score***	***average IoU***
0.76	0.70	0.73	60.11 %

Tableau 3.4 Les résultats de l'apprentissage

Dans le tableau suivant nous avons vu le résultat obtenu par chaque class :

Class	***Nombe img***	***précision***	***Vrai positive***	***Faux positive***
Bicyclette	652	83.43%	288,	72
Bateau	679	70.67%	301	82
Bouteille	547	61.42%	312	122)
Bus	527	87.84%	201	33
Voiture	638	77.46%	756	186
Chat	735	74.02%	222	65

Chaise	648	66.62%	458	214
Tasse	519	68.87%	357	125
Chien	801	79.46%	275	65
Moto	503	80.59%	240	71
Personnes	609	76.72%	1769	497
Table	505	55.66%	260	140

Tableau 3.5 Les résultats de l'apprentissage de chaque class

3.2. Test et résultat : Nous avons fait des tests sur des données choisies aléatoirement sur le web, et on a obtenu le résultat suivant :

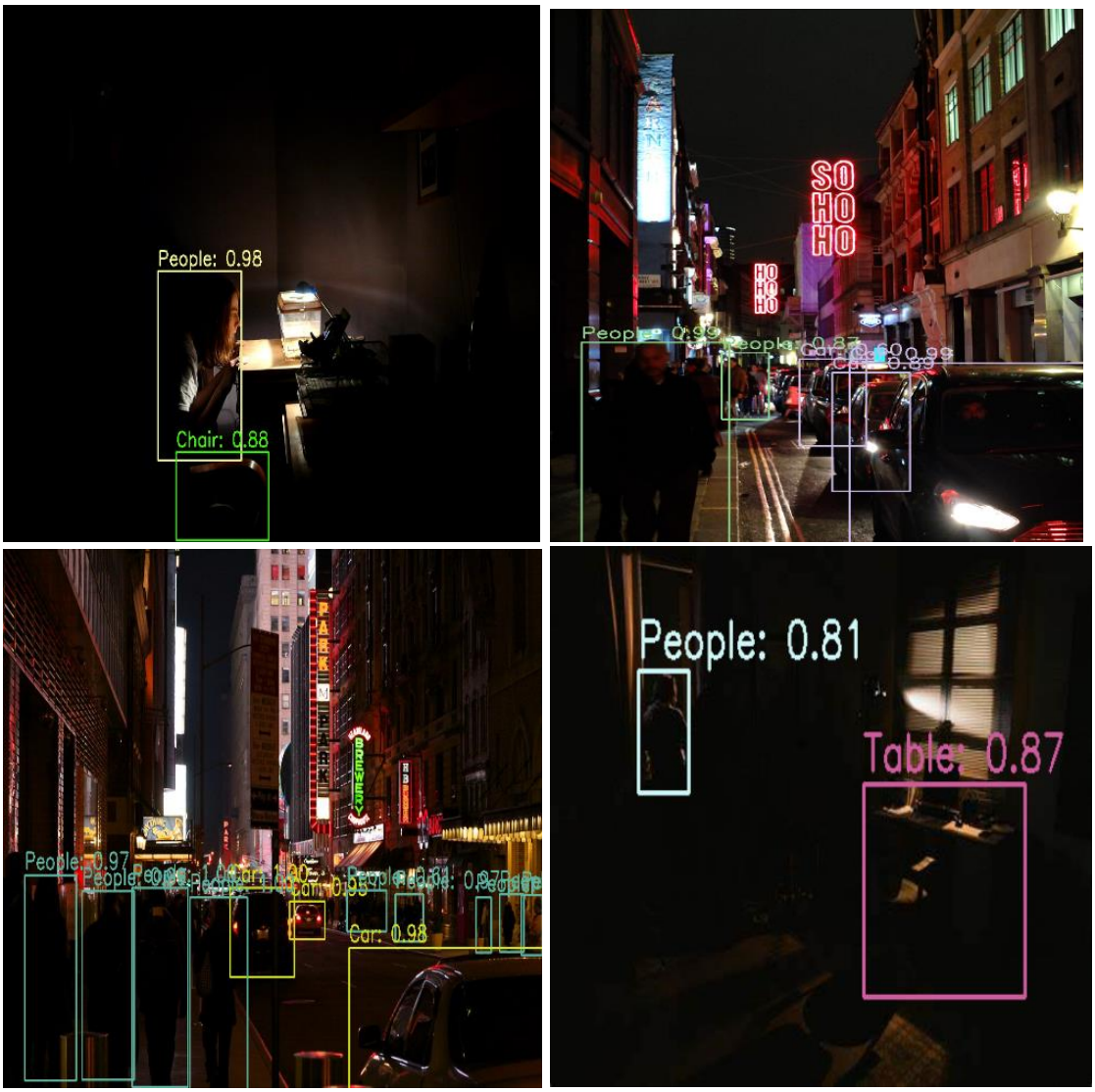


Figure 3.13 le résultat obtenu de détection par yolov4 ré-entraîner sur ExDark

4. Conclusion

Nous avons présenté dans ce chapitre l'implémentation de l'approche de détection d'objet basé sur le Deep Learning, pour cela nous avons utilisé le réseau de neurones convolutifs (YOLOv4) qui a été pré-entrainer jusqu'à 137 couches de convolution et on a réentériner sur 9 classes de la base de données pascal voc et réentériner sur la base de données ExDark, 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 tu veux ou bien vous prouvez entraîner l'algorithme sur votre propre BDD.

Conclusion générale :

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

Une des principales applications de la vision par ordinateur qui basé sue technique de l'apprentissage automatique est la détection d'objet. Depuis de nombreuses années, les scientifiques ont cherché des moyens efficaces pour obtenir de bons résultats dans ce domaine. Les méthodes comme celle de Viola et Jones ou les HOG ont été la référence pendant plusieurs années. Elles ont permis un tournant majeur dans le problème de détection d'objet. Mais en 2012, le mariage du deep learning avec cette discipline, les détrône et met en avant les CNN.

Cela nous a dirigés, dans ce mémoire, vers le développement du système de détection d'objet comme moyen pour aide les personne aveugle, dans le but de protéger et aide à connaitre les choses dans leur environnement dans la maison. Pour réaliser ce système nous sommes basés sur les réseaux de neurones convolutif (CNN), Ces derniers sont utiles pour la simulation de n'importe quel problème difficile à décrire avec des modèles physiques et mathématiques en raison de la capacité des réseaux de neurones d'apprendre par des exemples. Nous avons utilisé le model CNN yolov4 parce qu'il a prouvée sa performance dans la détection d'objet en temps réel.

Pour l'entrainement de model nous avons utilisé la base de données du benchmark PASCAL VOC comme source de données, et ExDark, et nous avons fait la détection sur 9 classes dans première base et 12 objets dans la deuxième base, et nous avons obtenu des résultats satisfaisants de détection dans diffèrent cas (compromis temps/précision gagner).

Pour conclure, la majorité des objectifs tracés dans de ce travail ont été atteints, mais il reste toujours des perspectives et des améliorations possibles qui peuvent encore être réalisé dans le future, telles que :

- Utiliser un autre model de détection d'objet en temps réel tel que le model SSD et faire une comparaison avec d'autre model de détection en temps réel.
- Changer le backbone de yolov4 utiliser MobileNet et comparé le résultat obtenu avec le DarkNet.
- Essayer d'entrer le model YOLO sur une autre base de données qui contient plusieurs classes pour donner plus de performance à notre système.
- Utiliser les algorithmes d'amélioration de la qualité des images pour améliorer la détection.

Référence

[1] https://paperswithcode.com/task/object-detection, dernière consultation 24/05/2021

[2] https://paperswithcode.com/task/object-tracking, dernière consultation 24/05/2021

[3] Caiming.Z et Yang.L “Study on artificial intelligence: The state of the art and future prospects” ScienceDirect, Journal of Industrial Information Integration page 01 (23) 8 Mai 2021

[4] Marco. A « Comme Exigence Partielle de la Maitrise en Philosophie » Mémoire présenté à l’université du Québec à Trois-Rivières, page 04, Mars 1992.

[5] Ekaba Bisong «Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners» page 169 OTTAWA, ON, Canada 2019

[6] Encyclopædia Universalis « Apprentissage Profond ou Deep Learning » https://www.universalis.fr/encyclopedie/apprentissage-profond-deep-learning/1-differents-types-d-apprentissage-machine, dernière consultation 24/05/2021

[7] Donald J. Norris « Machine Learning with the Raspberry Pi » page 213 Barrington, NH, USA 2020

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

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

[10] Randa Aouassa et Hallaci Samir « CNN Based Deep Face Recognition » page8_10 Mémoire Master informatique, Université 8 mai 1945 Guelma, Octobre 2020.

[11] Khodja Fouad « Conception d'un système intelligent à base de réseaux de neurones artificiels pour l'étude de la dynamique des streamers à la surface des polymères » page 17, Mémoire de Magister en Electrotechnique, Université des Sciences et de la Technologie d’Oran, Octobre 2011

[12] Farnoush Farhadi « learning activation functions in deep neural networks»page 56, Mémoire de Maîtrise ès Sciences Appliquées, Université De Montréal, Décembre 2017

[13] https://paperswithcode.com/methods/category/activation-functions consulter le 18/04/2021

[14] Loss Functions in Deep Learning: An Overview

Référence

https://analyticsindiamag.com/loss-functions-in-deep-learning-an-overview/ consulter le 18/04/2021

[15] Lei .H et autre « Normalisation Techniques in Training DNNS : Methodology,Analysis and Application » page 02-03, 27 septembre 2020

[16] Aarthi Kasirajan« Optimization Techniques popularly used in Deep Learning» https://medium.com/@minions.k/optimization-techniques-popularly-used-in-deep-learning-3c219ec8e0cc consulter le 19/04/2021

[17] data analytics post « régularisation » https://dataanalyticspost.com/Lexique/regularisation/ consulter le 19/04/2021

[18] K. Kim et al. « Network Intrusion Detection using Deep Learning» page 32,2018

[19] https://medium.com/fintechexplained/neural-networks-bias-and-weights-10b53e6285da consulter le 29/06/2021

[20] X. Jiang , A.Hadid ,Y. PangEric Granger ,X. Feng «Deep Learning in Object Detection and Recognition» Springer Nature Singapore Pte Ltd. 2019, page 08.

[21] Louam Abdelhak Bilal « Deep Learning basé sur les méthodes de réduction pour la reconnaissance de visage » Mémoire De Master Sciences et Technologies Télécommunication Réseaux et Télécommunication, Université Mohamed Khaider de Biskra ,2019, page 14.

[22] hiba hakim et Ali Fadhil« survey: Convolution Neural networks in Object Detection » Article dans Journal of Physics Conference Series · février, 2021,page 02.

[23] Asifullah Khan et autre « A Survey of the Recent Architectures of Deep Convolutional Neural Networks » Publié dans Artificial Intelligence Review DOI, 21 avril 2020.

[24] Trad Houssem Eddine « La détection d'objet avec OpenCV et deep learning » Mémoire De Master Sciences et Technologies Electronique Réseaux Télécommunication, Université Mohamed Khider de Biskra, 30 septembre 2020, page 14

[25] Shiliang Sun et autre « A Survey of Optimization Methods froma Machine Learning Perspective » School of Computer Science and Technology, East China NormalUniversit,23 October 2019, page 10

[26] Oulmi Mehdi et Kaloune Salim « Classification d'objets avec le Deep Learning » Mémoire de Master En Informatique, Université Akli Mohand Oulhadj de Bouira, 2018, page 30.

[27] BITMOVIN https://bitmovin.com/object-detection. consulter le 29/06/2021

[28] https://www.bial-r.com/2019/05/22/comprendre-le-machine-learning-et-le-deep-learning. consulter le 29/06/2021

Référence

[29] https://xsj.699pic.com/tupian/0nzwtm.html consulter le 29/06/2021

[30] https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464. consulter le 29/06/2021

[31] https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network-architecture. consulter le 29/06/2021

[32] https://inside-machinelearning.com/cnn-couche-de-convolution. consulter le 29/06/2021

[33] R. Girshick et autre « Rich feature hierarchies for accurate object detection and semantic segmentation » IEEE Conference on Computer Vision and Pattern Recognition,2014

[34] K. He, G. Gkioxari, P. Doll´ar, and R. Girshick « Mask R-CNN» dans les Acts Conférence internationale de l'IEEE sur la vision par ordinateur ICCV,24 Janvier 2018

[35] R. Girshick « Fast R-CNN », Conférence internationale de l'IEEE sur la vision par ordinateur, 2015

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

[37] Object Detection Using OpenCV YOLO | Great Learning https://www.mygreatlearning.com/blog/yolo-object-detection-using-opencv/, consultation 04/08/2021

[38] 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

[39] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Fu, C. et Berg, A. C, «SSD: Single Shot MultiBox Detector » Preprint sur https://arxiv.org/abs/1512.02325, 2016

[40] An Improved Single Shot Multibox Detector Method Applied in Body Condition Score for Dairy Cows, https://www.mdpi.com/2076-2615/9/7/470/html, dernier consultation 04/08/2021.

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

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

[43] Guide to Object Detection using YOLO | by Jantakarn | Medium https://medium.com/@aumjantakarn/guide-to-object-detection-using-yolo-33d74d7091d9 denier consultation 13/08/2021

Référence

[44] Selecting the Right Bounding Box Using Non-Max Suppression (with implementation) https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression -with-implementation dernier consultation 13/08/2021.

[45] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. « CSPNet: A new backbone that can enhance learning capability of CNN». La conférence de l'IEEE sur la vision par ordinateur et la reconnaissance des formes (CVPR Workshop), 2020.

[46] Joseph Redmon and Ali Farhadi. « YOLOv3: An incremental improvement ». Preprint sur arXiv :1804.02767, 2018

[47] https://paperswithcode.com/method/cspdarknet53 dernier consultation 25/08/2021

[48] He, K., Zhang, X., Ren, S., et al.: « Spatial pyramid pooling in deep convolutional networks for visual recognition ». Ieee Transactions On Pattern Analysis Et Machine Intelligence, vol. 37, page 1906, septembre 2015.

[49] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. «Path aggregation network for instance segmentation». Dans les actes de la conférence de l'IEEE sur la vision informatique et la reconnaissance des formes. (CVPR), page 8759- 8760, 2018.

[50] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. « CBAM : Convolutional block attention module ». Dans les actes de la conférence européenne sur la vision par ordinateur. (ECCV), page 6,2018

[51] Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: « YOLOv4: Optimal Speed and Accuracy of Object Detection ». arXiv 2020. arXiv preprint. arXiv:2004.10934. pp. 2–7 (2020).

[52] YOLO-v4 Object Detector | reckoning.dev https://reckoning.dev/blog/yolo-v4/ dernier consultation 26/08/2021

[53] Joseph Redmon and Ali Farhadi. « YOLO9000 : better, faster, stronger ». Dans la conférence de l'IEEE sur la vision par ordinateur et la reconnaissance des formes., CVPR 2017, Honolulu, HI, USA, pages 2-6, Juillet 21-26, 2017.

[54] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, et Jian Sun. «Yolox: Exceeding yolo series in 2021 ». arXiv preprint arXiv:2107.08430, 2021

[55] Introduction to YOLOv5 Object Detection with Tutorial | MLK - Machine Learning Knowledge https://machinelearningknowledge.ai/introduction-to-yolov5-object-detection-with-tutorial dernier consultation 05/09/2021

Référence

[56] MAKINA-CORPUS Extraction d'objets pour la cartographie par deep-learning : évaluation du modèle : https://makina-corpus.com/blog/metier/2020/extraction-dobjets-pour-la-cartographie-par-deep-learning-evaluation-du-modele. dernier consultation 06/09/2021

[57] https://github.com/cs-chan/Exclusively-Dark-Image-Dataset dernier consultation 06/09/2021

[58] https://qiita.com/mine820/items/03e2bbd9b603383486e0 dernier consultation 06/09/2021

[59] Comprendre les réseaux de neurones. Du neurone au RNN, CNN et Deep Learning (ichi.pro) https://ichi.pro/fr/comprendre-les-reseaux-de-neurones-du-neurone-au-rnn-cnn-et-deep-learning-225987909257872 dernier consultation 06/09/2021

[60] https://nickmccullum.com/python-deep-learning/deep-learning-activation-functions dernier consultation 06/09/2021

[61] GELU Explained | Papers With Code https://paperswithcode.com/method/gelu dernier consultation 06/09/2021

[62] Leaky ReLU Explained | Papers With Code https://paperswithcode.com/method/leaky-relu dernier consultation 06/09/2021

[63] Swish Explained | Papers With Code https://paperswithcode.com/method/swish dernier consultation 06/09/2021

[64] Softplus Explained | Papers With Code https://paperswithcode.com/method/softplus dernier consultation 06/09/2021

[65] Mish activation function and Pytorch implementation - Programmer Sought https://www.programmersought.com/article/82006174117 dernier consultation 06/09/2021

[66] PReLU Explained | Papers With Code https://paperswithcode.com/method/prelu dernier consultation 06/09/2021

[67] Zhibin Liao et Gustavo Carneiro «On the Importance of Normalisation Layers in Deep Learning with Piecewise Linear Activation Units» arXiv:1508.00330v1 [cs.CV] ,3 aout 2015

[68] TanhExp Explained | Papers With Code https://paperswithcode.com/method/tanhexp

[69] Parisi, L. «M-arcsinh: An efficient and reliable function for SVM and MLP inscikit-learn». arXiv preprint arXiv:2009.07530.,16 September 2020.

[70] H. Hazimeh, N. Ponomareva, P. Mol, Z. Tan, and R. Mazumder, «The tree ensemble layer: Differentiability meets conditional computation, » arXiv preprint arXiv:2002.07772, 2020.

[71] Paul Barde et autre « Adversarial Soft Advantage Fitting: Imitation Learning without Policy Optimization »arXiv:2006.13258v6 [cs.LG] 16 avril 2021

Référence

[72] Li Liu, Wanli Ouyang, Xiaogang Wang, · Paul Fieguth, Jie Chen, Xinwang Liu, Matti Pietikäinen «Deep Learning for Generic Object Detection: A Survey »International Journal of Computer Vision https://doi.org/10.1007/s11263-019-01247-4, 26 septembre 2019

[73] ImageNet (image-net.org) https://image-net.org/about dernier consultation 06/09/2021

[74] ImageNet (image-net.org) https://image-net.org/challenges/LSVRC/index.php dernier consultation 06/09/2021

[75] Python Programming Language - GeeksforGeeks https://www.geeksforgeeks.org/python-programming-language dernier consultation 07/09/2021

[76] Google Colab tutorial - Great Learning (mygreatlearning.com) https://www.mygreatlearning.com/blog/google-colab-tutorial/ dernier consultation 07/09/2021

[77] About - OpenCV https://opencv.org/about dernier consultation 07/09/2021

[78] The PASCAL Visual Object Classes Challenge 2012 (VOC2012) (ox.ac.uk) http://host.robots.ox.ac.uk/pascal/VOC/voc2012/

[79] Open Images V6 - Description (storage.googleapis.com) https://storage.googleapis.com/openimages/web/factsfigures.html

[80] https://cocodataset.org/#home consultation 06/09/2021

[81] Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton.. « Layer normalization». CoRR abs/1607.06450. 2016

[82] S. Ioffe and C. Szegedy « Batch normalization: Accelerating deep network training by reducing internal covariate shift». In Proceedings of ICML, 2015.

[83] A. Krizhevsky, I. Sutskever, and G. E. Hinton. « Imagenet classification with deep convolutional neural networks ». In Advances in neural information processing systems, 2012

[84] Ulyanov, A. Vedaldi, and V. Lempitsky. « Instance normalization: The missing ingredient for fast stylization». arXiv preprint arXiv:1607.08022, 2016

[85] X. Huang and S. J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. CoRR, abs/1703.06868, 2017

[86] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. CoRR, abs/1802.05957, 2018

[87] H. De Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville. Modulating early visual processing by language. In Advances in Neural Information Processing Systems (NeurIPS), 2017

[88] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and Improving the Image Quality of StyleGAN,” in Proc. IEEE/CVF Conference on Computer Vision and Patter Recognition, 2020.

[89] Tim Salimans and Diederik P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. arXiv preprint arXiv:1602.07868, 2016

[90] Yuxin Wu and Kaiming He. Group normalization. In Proc. Eur. Conf. Comp. Vis, 2018.

Référence

[91] D.Bahrami et S.Pouriyan Zadeh « Gravity Optimizer: a Kinematic Approach on Optimization in Deep Learning »ArXiv abs/2101.09192,2021

[92] R. Anil, V. Gupta, T. Koren, K. Regan, Y. Singer « Towards Practical Second Order Optimization for Deep Learning » ICLR Conference Blind Submission,2021

[93] A. Defazio, S. Jelassi « Adaptivity without Compromise: A Momentumized, Adaptive, Dual Averaged Gradient Method for Stochastic Optimization», arXiv:2101.11075v3,2121

[94] L. Nicola, I. Gallo, et R. La Grassa. «Combining Optimization Methods Using an Adaptive Meta Optimizer»*Algorithms* 14, no. 6: 186. https://doi.org/10.3390/a14060186,2021

[95] A. Biedenkapp et H. Furkan Bozkurt et T. Eimer et F. Hutter et M. Lindauer « Dynamic Algorithm Configuration: Foundation of a New Meta-Algorithmic Framework», European Conference on Artificial Intelligence – ECAI,2020

[96] N. Landro,et autre «Mixing adam and sgd: a combined optimization method with pythorch, » https://gitlab.com/nicolalandro/multi optimizer, 2020.

[97] Y. Zheng, R. Zhang, and Y. Mao, «Regularizing Neural Networks via Adversarial Model Perturbation, »Available: http://arXiv.org/abs/,2020

[98] D. Gurevin, S. Zhou, L. Pepin, B. Li, M. Bragin, C. Ding, and F. Miao, "Enabling retrain-free deep neural network pruning using surrogate lagrangian relaxation," arXiv preprint arXiv:2012.10079, 2020.

[99] J. Zhuang, T. Tang, Y. Ding, S. Tatikonda, N. Dvornek, X. Papademetris, et J. Duncan « Adabelief optimizer: Adapting stepsizes by the belief in observed gradients». NeurIPS, 2020.

[100] Xuezhe Ma. Apollo: An adaptive parameter-wise diagonal quasi-newton method for nonconvex stochastic optimization. arXiv preprint arXiv:2009.13586, 2020.

[101] M. D. Phung, Q. P. Ha, Motion-encoded particle swarm optimization for moving target search using UAVs, Applied Soft Computing 106705doi:10.1016/j.asoc.2020.106705.2020

[102] leonardo L. Custode and Giovanni Iacca.. Evolutionary learning of interpretable decision trees. arXiv preprint arXiv:2012.07723 (2020).

[103] Z. Yao, A. Gholami, S. Shen, K. Keutzer, and M. W. Mahoney, "Adahessian: An adaptive second order optimizer for machine learning," arXiv preprint arXiv:2006.00719, 2020.

[104] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. ´ Focal loss for dense object detection. arXiv preprint arXiv:1708.02002, 2017

[105] .-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired imageto-image translation using cycle-consistent adversarial networks. In IEEE International Conference on Computer Vision (ICCV), 2017

[106] Dong, Xingping et Shen, Jianbing "Triplet Loss in Siamese Network for Object Tracking" Proceedings of the European Conference on Computer Vision (ECCV) September 2018

[107] X.Mao, Q. Li, H. Xie, R. Y. Lau, and Z. Wang. Multiclass generative adversarial networks with the l2 loss function. arXiv preprint arXiv:1611.04076, 2016

[108] aron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. arXiv:1807.03748, 2018.

Référence

[109] J. H. Lim and J. C. Ye. Geometric gan. arXiv preprint arXiv:1705.02894, 2017.

[110] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In NIPS, 2016

[111] Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of Wasserstein GANs. NIPS, 2017.

[112] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019

[113] J. T. Barron. A more general robust loss function. CoRR, abs/1701.03077, 2017.

[114] Shashanka Venkataramanan, Bill Psomas, Yannis Avrithis, Ewa Kijak, Laurent Amsaleg, and Konstantinos Karantzalos. It takes two to tango: Mixup for deep metric learning. arXiv preprint arXiv:2106.04990, 2021

[115] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. arXiv preprint arXiv:2004.11362, 2020.

[116] Haoyang Zhang, Ying Wang, Feras Dayoub, and Niko Sunderhauf. Varifocalnet: An iou-aware dense object de- ¨ tector. arXiv preprint arXiv:2008.13367, 2020.

[117] Hongkai Zhang, Hong Chang, Bingpeng Ma, Naiyan Wang, and Xilin Chen. Dynamic R-CNN: Towards high quality object detection via dynamic training. In Proceedings of the European Conference on Computer Vision (ECCV), pages 260–275. Springer, 2020

[118] Jiangtao Kong and Yu Cheng and K. Li and Junliang Xing "DSAM: A Distance Shrinking with Angular Marginalizing Loss for High Performance Vehicle Re-identificatio,ArXiv abs/2011.06228,2020

[119] R. van der Merwe, "Triplet entropy loss: improving the generalisation of short speech language identification systems," 2020, https://arxiv.org/abs/2012.03775.

[120] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Tang, and J. Yang. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. In NeurIPS, 2020.

[121] Nitish Srivastava and Geoffrey Hinton and Alex Krizhevsky and Ilya Sutskever and Ruslan Salakhutdinov "Dropout: A Simple Way to Prevent Neural Networks from Overfitting" Journal of Machine Learning Research},2014

[122] Weight Decay Explained | Papers With Code
https://paperswithcode.com/method/weight-decay dernier consultation 28/08/2021

[123] Attention Dropout Explained | Papers With Code
https://paperswithcode.com/method/attention-dropout dernier consultation 28/08/2021

[124] Label Smoothing Explained | Papers With Code
https://paperswithcode.com/method/label-smoothing dernier consultation 28/08/2021

[125] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. arXiv preprint arXiv:1602.01783, 2016.

Référence

[126] Early Stopping Explained | Papers With Code
https://paperswithcode.com/method/early-stopping dernier consultation 28/08/2021

[127] L. Mescheder, A. Geiger, and S. Nowozin. Which training methods for GANs do actually converge? CoRR, abs/1801.04406, 2018

[128] Yarin Gal. A theoretically grounded application of dropout in recurrent neural networks. arXiv preprint arXiv:1512.05287, 2015.

[129] Regularization of Neural Networks Using DropConnect – NYU Center for Data Science https://cds.nyu.edu/projects/regularization-neural-networks-using-dropconnect dernier consultation 28/08/2020

[130] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. DropBlock: A regularization method for convolutional networks. In Advances in Neural Information Processing Systems (NIPS), pages 10727–10737, 2018

[131] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. arXiv:2103.17239, 2021.

[132] Anton Obukhov, Maxim Rakhuba, Alexander Liniger, Zhiwu Huang, Stamatios Georgoulis, Dengxin Dai, and Luc Van Gool. Spectral tensor train parameterization of deep learning layers. In International Conference on Artificial Intelligence and Statistics, pages 3547–3555. PMLR, 2021.

[133] Kim, Taehyeon and Kim, Jonghyup and Yun, Seyoung « Efficient Model for Image Classification With Regularization Tricks" Proceedings of Machine Learning Research, PMLR,2020