

# Intelligent people detection system for IoT images and videos

Idrissou HEL HADJE SOUNOU ALIDOU  
(idrissou.h.h.s.alidou@aims-senegal.org)

African Institute for Mathematical Sciences (AIMS) Senegal

Supervised by:  
Pr Cherif DIALLO  
Gaston Berger University (UGB), Senegal

March 18, 2021

*Submitted in partial fulfilment of the requirements for the award of Master of Science in Mathematical Sciences at AIMS Senegal*



# DECLARATION

This work was carried out at AIMS Senegal in partial fulfilment of the requirements for a Master of Science Degree.

I hereby declare that except where due acknowledgement is made, this work has never been presented wholly or in part for the award of a degree at AIMS Senegal or any other University.



Idrissou Hel Hadje Sounon Alidou

# ACKNOWLEDGEMENTS

A'oudhou biLlahi mina shaytani rajim.

Au nom d'Allah, le Tout Miséricordieux, le Très Miséricordieux.

Toutes les louanges sont à Allah Subhānahu wa ta'āla, Seigneur de l'univers, le Savant dont la science est infinie et englobe toutes choses, le Créateur qui n'est pas créé et qui ne s'inspire d'aucun modèle dans Sa création, le Facilitateur de tout œuvre de bien. Que la paix et la bénédiction d'Allah soient sur Son Prophète et Messager, Muhammad, ainsi que sur sa famille et ses Compagnons.

Je rends grâce à Allah pour tous Ses bienfaits sur nous et pour m'avoir accordé la vie jusqu'à ce jour-ci. Je remercie mes parents et ainsi que toute ma famille pour leurs soutiens et assistances de tous les jours.

Je remercie le Prof. Neil Turok pour avoir fondé le réseau AIMS, me donnant ainsi l'opportunité de me faire former. Je remercie le président du centre AIMS-Sénégal Prof Mouhamed Moustapha Fall et le directeur académique Dr Franck Kalala Mutombo, pour nous avoir accompagnés tout au long de notre formation. Mes remerciements vont également à l'endroit de l'IT Mr Mohamed Lamine DIALLO pour son assistance technique, tous les autres membres du Staff de AIMS-Sénégal ainsi que tous les professeurs qui nous ont enseignés.

Je tiens à exprimer ma sincère gratitude à mon superviseur, Prof Chérif DIALLO, pour son initiative, son intérêt, son accompagnement et son soutien continu tout au long de ce travail.

Je remercie également tous les tuteurs, en particulier le Dr Cheikh b. Ndao et le Dr Damien Gbeve-wou HOUNDEDJI pour leurs assistances, les corrections et suggestions en vue de l'amélioration de ce travail.

Je termine en remerciant tous mes camarades, amis et toute personne qui s'est rendue utile durant ma formation.

# **DEDICATION**

This work is dedicated to  
anyone concerned about a more conscious world  
and using science and technology for the well-being  
of mankind and not to cause him harm.

# **Abstract**

Today, thanks to artificial neural networks, the field of computer vision is undergoing an unprecedented revolution. The rise of deep learning and the advent of big data have both facilitated the construction of neural networks, not only deep, but also increasingly efficient. This has resulted in the automation of IoT technologies and devices.

In this dissertation, we review the state of the art on IoTs and examine their potential in the production of Big Data. Next, we list some open source datasets used for object detection tasks. YOLO architectures are state-of-the-art detection algorithms. Thus, we are building four person detection models based on the YOLO v3 and YOLO v4 architectures. After evaluation, we obtained in terms of Average Accuracy (AP) a performance of about 88 % with the model based on YOLO v4.

# Résumé

Aujourd’hui, grâce aux réseaux neuronaux artificiels, le domaine de la vision par ordinateur connaît une révolution sans précédent. L’essor du deep learning et l’avènement du big data ont tous deux facilité la construction de réseaux de neurones, non seulement profonds, mais aussi de plus en plus performants. Ce qui a pour effet l’automatisation des technologies et appareils IoT.

Dans ce mémoire, nous passons en revue l’état de l’art sur les IoT et examinons leur potentiel dans la production de Big Data. Ensuite, nous listons quelques ensembles de données open source utilisés pour les tâches de détection d’objets. Les architectures YOLO sont des algorithmes de détection de pointe. Ainsi, nous construisons quatre modèles de détection de personne basés sur les architectures YOLO v3 et YOLO v4. Après évaluation, nous avons obtenu en termes de Précision Moyenne (AP) une performance d’environ 88 % avec le modèle basé sur YOLO v4.

# Contents

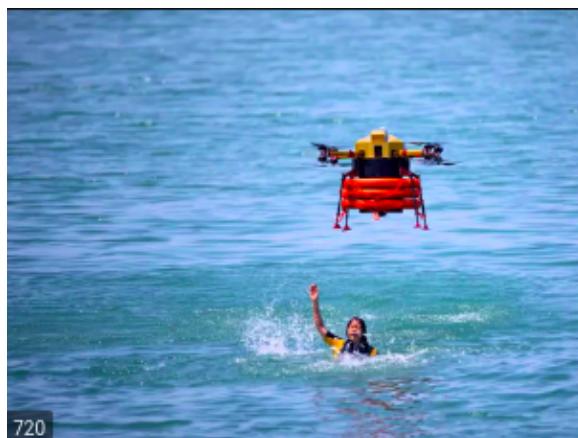
<b>Declaration</b>	i
<b>Acknowledgements</b>	ii
<b>Dedication</b>	iii
<b>Abstract</b>	iv
<b>1 Introduction</b>	1
<b>2 Les IoT: de véritables sources de production du Big Data</b>	3
2.1 Le concept de l'IoT . . . . .	3
2.2 État de l'art sur l'IoT . . . . .	3
2.3 Secteurs ou domaines d'application des IoT . . . . .	4
2.4 Le potentiel des IoT dans la production du Big Data . . . . .	6
2.5 Outils de gestion et de stockage du Big Data IoT . . . . .	7
<b>3 Le Deep Learning, une approche de pointe pour l'analyse des données IoT</b>	9
3.1 Rappel sur les réseaux de neurones artificiels . . . . .	9
3.2 Architectures des réseaux de neurones . . . . .	19
3.3 Deep Learning . . . . .	25
<b>4 Application : formation d'un modèle intelligent de détection de personnes</b>	34
4.1 Énoncé du problème . . . . .	34
4.2 Collecte, préparation et nettoyage des données . . . . .	34
<b>5 Conclusion et Perspectives</b>	40
5.1 Conclusion . . . . .	40
5.2 Contributions . . . . .	40
5.3 Perspectives . . . . .	40
<b>Références</b>	43

# 1. Introduction

Le secteur de la sécurité est l'un des domaines connaissant l'utilisation des technologies IoT. De nombreux objets connectés sont mis sur le marché afin de mieux répondre aux besoins sécuritaires de tous. C'est le cas de l'utilisation des systèmes de vidéo-surveillance pour une visibilité permanente des locaux. Des technologies IoT comme les capteurs, les caméras et les véhicules aériens sans pilote sont utilisés pour des tâches de surveillance et des missions de recherche et de sauvetage (voir figure 1.1b). Ces appareils sont également utilisés dans des zones à accès difficile et/ou dangereux pour les agents secouristes à l'exemple des zones sinistrées (voir figure 1.1a) souvent encombrées d'arbres abattus, de bâtiments effondrés, de routes brisées, etc.



(a) Personne se trouvant dans une zone sinistrée



(b) Drone dans une mission de sauvetage

Figure 1.1

Cependant, ces technologies ne sont pas sélectives (toutes les images captées sont enregistrées ; qu'elles soient pertinentes ou non ; utiles ou non) et sont souvent télé-assistées. Ce qui ne facilite pas le sauvetage à temps des personnes se trouvant dans ces zones sinistrées. Ainsi, se présente donc la nécessité d'introduire un système intelligent de détection de personnes dans ces technologies. Afin d'y arriver, on s'est fixé deux objectifs. Dans un premier temps, contribuer à l'amélioration des systèmes de sécurité et de sauvetage. Et dans un second temps, rendre automatiques les dispositifs sécuritaires, de recherche et de sauvetages en construisant un modèle de détection automatique de personne.

Nous avons été motivés par le fait que les réseaux de neurones profonds ont prouvé ces dernières années leur capacité à simuler les actions humaines, et même les surpasser dans certains cas. Une belle illustration en est l'algorithme AlphaGo de Google DeepMind qui a eu les capacités de battre Ke Jie, le champion du monde du jeu Go, en 2017 [30]. En intégrant donc de tels algorithmes dans les technologies IoT, l'on peut espérer avoir les performances humaines au niveau de ces appareils. Ainsi, ces technologies pourront valablement remplacer l'homme dans les zones dangereuses et/ou inaccessibles. Aussi, ils pourront couvrir les zones de surveillance et ceci avec la plus grande efficacité. Le déploiement d'un tel modèle dans les caméras de surveillance, les capteurs et engins sans pilote pourrait aider les entreprises et particuliers à renforcer la sécurité

de leurs locaux et installations. Aussi, les gouvernements pourraient s'en servir afin de sécuriser les frontières et leurs populations, surveiller en temps réel les zones protégées, contrôler les plages et ports et ainsi renforcer efficacement la sécurité de leurs territoires.

Dans la section 1, nous avons rappelé l'importance de l'intégration d'un système de détection dans les IoT. Dans la section 2, nous avons fait l'état de l'art sur l'IoT et étudié leur potentiel dans la production du Big Data puis nous avons rappelé quelques domaines d'application des IoT. Ensuite, dans la section 3, nous avons étudié en quoi le Deep Learning est une approche de pointe dans l'analyse de le traitement des données IoT. Mais avant cela, nous avons rappelé quelques architectures des réseaux de neurones artificiels et leur mode de fonctionnement. La section 4 est consacrée aux différents processus de formation de notre modèle de détection et qui s'est achevée par l'évaluation dudit modèle. Enfin, dans la section 5, nous donnons une conclusion et des perspectives.

## 2. Les IoT: de véritables sources de production du Big Data

### 2.1 Le concept de l'IoT

Avec l'expansion de l'internet et des technologies de communication, les machines, les personnes et leur environnement sont devenus des objets qui peuvent interagir ensemble.

Le terme "Internet of Things" a été utilisé pour la première fois en 1999 par Kevin Ashton pour décrire un système de dispositifs interconnectés [7].

Dès lors, le concept IoT a été défini à plusieurs reprises par diverses organisations et chercheurs travaillant dans le domaine de l'IoT et de ses applications. L'UIT (Union Internationale des Télécommunications) définit l'internet des objets comme : "une infrastructure mondiale pour la société de l'information, permettant des services avancés en interconnectant des objets (physiques et virtuels) basés sur des technologies de l'information et de la communication interopérables existantes et en évolution" [11].

La vision de l'IoT ces dernières années va au-delà de la simple interconnexion d'objets ou d'appareils via Internet [3]. Les définitions du concept de l'IoT des différents groupes de recherche [14] en témoignent. L'IoT se veut aujourd'hui une planète plus intelligente dans laquelle les appareils et les humains interagissent non seulement entre eux, mais aussi avec leur environnement de n'importe quel lieu et à tout moment [24, 16]. Les appareils sont désormais dotés de systèmes de plus en plus intelligents avec des capacités d'auto-configuration et ceci avec l'essor de l'apprentissage automatique ; un sous-domaine de l'intelligence artificielle. Ainsi, l'homme se veut le contrôle et l'automatisation de son environnement.

### 2.2 État de l'art sur l'IoT

Ces dernières années, avec l'augmentation de la vitesse d'internet et les progrès des technologies de communication sans fil, l'IoT est devenu un domaine émergent attirant industries et chercheurs. De nombreux pays comme la Corée du Sud sont allés au-delà de la 4 G dans les services Internet [27]. Ce qui facilite l'interconnexion et l'échange des données entre différents dispositifs. Cette interconnexion dans le réseau IoT, est rendue possible grâce à plusieurs technologies. Parmi celles-ci, nous avons : les réseaux de capteurs sans fil et les protocoles de communication.

#### 2.2.1 Les réseaux de capteurs sans fil

Un réseau de capteurs sans fil ou **Wireless Sensor Networks (WSN)** est un réseau de micro-capteurs ou de capteurs capables de recueillir et de transmettre des données d'une manière autonome.

Un capteur est un dispositif physique capable de détecter les conditions environnementales et physiques telles que la lumière, la chaleur, le mouvement, la pression ou le son, etc.

Les capteurs jouent un rôle important dans la transmission des données à travers le réseau. Ils sont de plusieurs types et sont classés en fonction de leur fonctionnalité (voir figure 2.1). Nous pouvons citer : les capteurs sonores (par exemple, microphone), les capteurs de lumière et magnétiques, les capteurs optiques, les capteurs de température, les capteurs de distance, etc.

Les données générées par les capteurs et l'ensemble des appareils interconnectés sont transmises au réseau grâce aux protocoles de Communication.



(a) Capteur de lumière



(b) Capteur de pression

Figure 2.1: Quelques types de Capteurs

## 2.2.2 Protocole de communication

L'échange des données issues de ces appareils interconnectés et des capteurs à travers tout le réseau est assurée par diverses technologies appelées protocoles de communication. Parmi ceux-ci, nous avons : Le **RFID** (Radio Frequency Identification), le **Wi-Fi** (Wireless Fidelity), la technologie **M2M** (Machine to Machine), les technologies **Bluetooth**, le **ZigBee**, le **Z-Wave**, etc. [27, 16].

Au-delà de la transmission et l'échange des données à travers l'ensemble du réseau, toutes ces technologies offrent à l'homme la possibilité d'interagir avec son environnement. Une illustration en est donnée dans la figure 2.2

Les solutions IoT sont présentes dans presque tous les secteurs d'activités facilitant ainsi la vie de tous les jours.

## 2.3 Secteurs ou domaines d'application des IoT

Les applications potentielles de l'IoT sont nombreuses et diverses. Leur utilisation est variée et s'étend pratiquement à tous les domaines de la vie quotidienne des individus, à toutes les

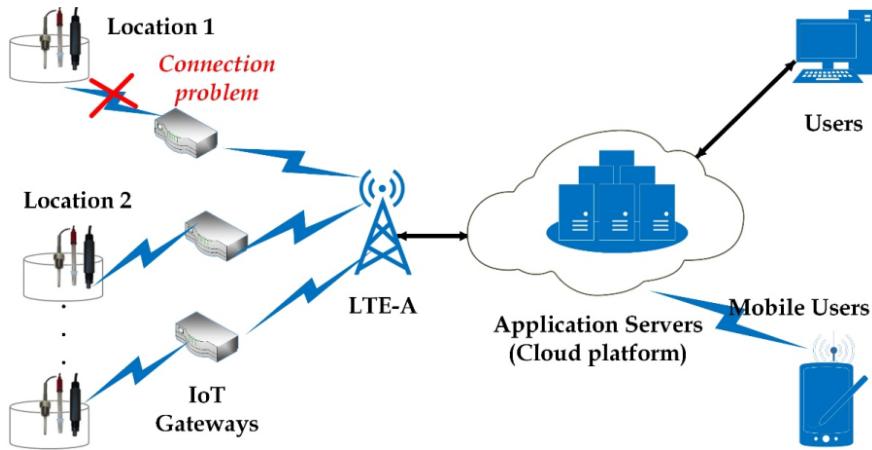


Figure 2.2: Transmission et échange de données dans un réseau IoT

entreprises et sociétés. Une illustration en est donnée dans la figure 2.3.

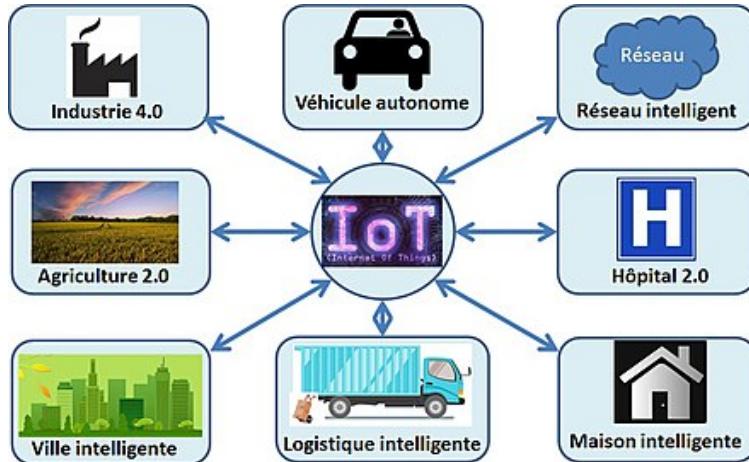


Figure 2.3: Domaines d'application des IoT

Le secteur de la domotique est un bon exemple d'application des IoT avec la création des smart homes et des smart cities. Ainsi, on assiste à l'automatisation des portes, l'optimisation de la consommation d'eau et d'électricité, la gestion des stationnements, optimisation des flux de circulation, etc. De nombreuses urgences environnementales sont désormais gérées de manière efficace et automatique avec les systèmes IoT.

Dans le secteur de l'énergie, les objets connectés sont au cœur de nombreuses réalisations telles que Smart Grids et les systèmes d'énergie renouvelable favorisant une meilleure réparation de l'énergie, une meilleure gestion des consommations énergétiques et la surveillance automatique des installations.

L'agriculture et l'élevage sont devenus intelligents avec l'installation des capteurs dans les zones agricoles et les fermes. Les capteurs sont utilisés afin de rendre l'irrigation et l'arrosage intelligent. Ce qui favorise une meilleure gestion des ressources en eau. L'utilisation des colliers et capteurs permettant de recueillir des informations sur la position et la santé du bétail. Ce qui permet le

suivi en temps réel du bétail et des cultures favorisant ainsi leur sécurisation et leur surveillance en temps réel.

Les réalisations dans le domaine de la médecine sont également considérables. Les capteurs IoT et les bracelets connectés dédiés aux soins de santé peuvent mesurer, surveiller et analyser de façon continue les différents états de santé des patients. Ainsi le rythme cardiaque, la pression sanguine, la saturation en oxygène sont envoyés de façon automatique dans des centres de traitement. Ce qui permet la surveillance à distance des patients et des interventions rapides pendant des moments critiques. Aussi, les résultats des analyses médicales sont fournis avec beaucoup plus de fiabilité et en des temps records.

Les domaines sécuritaires et militaires ont connu également d'énormes progrès avec la télésurveillance et l'utilisation des armes automatisées. Des drones et véhicules aériens sans pilote sont utilisés pour la surveillance des locaux, des villes, et même des territoires. Ces appareils sont également utilisés dans des champs de bataille, pour les sauvetages surtout en zone difficile.

L'optimisation du flux des passagers, le suivi des transports, la surveillance et sécurisation des infrastructures de transport et des marchandises sont quelques avancées des IoT dans le secteur des transports et de la logistique. La technologie des voitures autonomes représente un nouveau paradigme de l'IoT dans ces secteurs d'activités. Ces véhicules sont équipés d'un ensemble de dispositifs constitués de caméras et capteurs qui recueillent des informations sur le trafic routier et les panneaux de signalisation, etc. Ces informations sont ensuite traitées par des algorithmes d'apprentissage automatique. Ce qui leur permet de circuler tout seul sans aucune intervention humaine tout en respectant les règles de circulation et en évitant les obstacles.

Tous ces systèmes intégrants les IoT génèrent d'énormes quantités données. L'analyse et le traitement de ces données favorisent l'amélioration des services offerts et créent ainsi de la valeur.

## 2.4 Le potentiel des IoT dans la production du Big Data

Les appareils connectés sont reconnus pour leur production du big data. Le big data désigne un ensemble constitué de données de types variétés, produites de façon volumineuse et avec une grande vitesse. En effet, les technologies IoT produisent d'énormes quantités de données. Ces données produites avec une sont de formes et de types variés. Elles sont classées en deux grands groupes selon leur structure. Ainsi, nous avons les **données structurées** et les **données non-structurées**.

Les données structurées sont des données organisées, des données ayant des champs prédéfinis dans un format ciblé. Des exemples courants en sont les fichiers Excel ou les données issues des bases de données relationnelles ou tabulaires.

Les données non-structurées sont un autre type de données non organisées ou sans modèle prédéfini. Elles sont souvent de formes et de tailles différentes. Elles sont pour la plupart des données issues des applications web et mobiles, des capteurs, des signaux d'appareils, des examens d'imagerie médicale, etc.

La production de données par les systèmes IoT augmente de jour en jour. Ainsi, de nouveaux outils et technologies sont développés ces dernières années pour leur stockage et leur gestion.

## 2.5 Outils de gestion et de stockage du Big Data IoT

Les données sont le moteur qui alimente la plupart des entreprises. Même les industries ne disposant pas des technologies de pointe utilisent des quantités énormes de données pour obtenir un avantage concurrentiel, réduire les coûts et optimiser leur rendement. Parmi les outils et technologies utilisés pour le stockage et le traitement des données, nous pouvons citer :

- **Spark** : c'est un moteur d'analyse unifié pour le traitement de données à grande échelle.
- **Hadoop** : il est l'un des grands cadres technologiques de données les plus populaires utilisées pour le stockage et le traitement distribué de grands ensembles de données sur des clusters (serveurs placés en ligne).
- **MongoDB** : c'est un système de gestion base de données distribuée, basée sur des documents ; c'est-à-dire optimisé pour stocker les données au format "clé/valeur".

L'exploitation appropriée de ces données s'est révélée bénéfique pour dans les domaines d'activité, et même pour nos besoins quotidiens (voir figure 2.4). Les prévisions météorologiques, les ventes en ligne, le contrôle à distance des cultures, des transports et environnements de travail sont quelques résultats de cette exploitation de données.

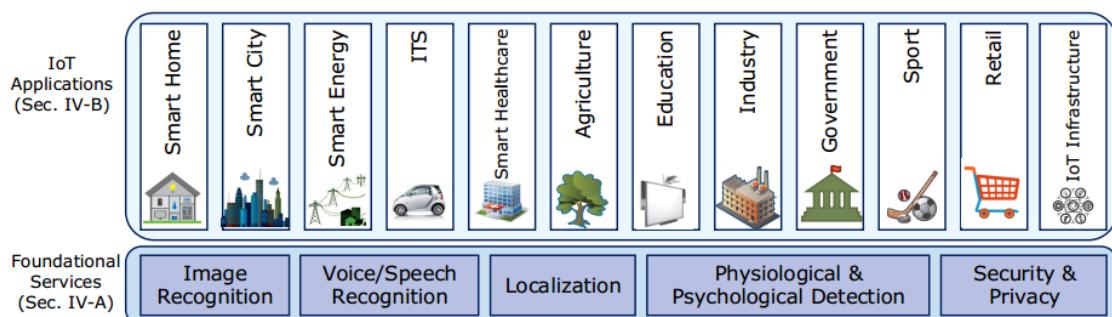


Figure 2.4: Domaines d'application et services IoT

Les approches traditionnelles d'analyse ne répondent plus aux types de données actuellement produites. Ces approches sont presque inutilisables pour les données IoT du fait de leur volume et de leur complexité. L'apprentissage automatique et en particulier l'apprentissage profond constituent les méthodologies prometteuses pour le traitement des données IoT. De nombreux secteurs d'activité et services sont confrontés à des défis liés à l'analyse du big data. Ils utilisent des applications comme celles de la prédition des fraudes, de prévision de marché ou de prévision météorologique, des applications de prédition des catastrophes naturelles, les véhicules autonomes, les systèmes de régularisation du flux de la circulation, etc. Toutes ces applications

se veulent plus performantes. Les données produites par ces dernières doivent donc être analysées d'une manière et instantanée afin de faciliter la prise de décisions à un moment précis.

Face à tous ces défis liés dans un premier temps à la nature et à la structure des données puis dans un second temps à la complexité du problème à résoudre, les techniques d'analyse des données ont montré leurs limites. Cependant, les techniques de l'apprentissage profond se sont révélées prometteuses pour faire face à de tels défis.

# **3. Le Deep Learning, une approche de pointe pour l'analyse des données IoT**

Avec le mouvement d'automatisation en cours, la question de la précision dans l'analyse des données devient très préoccupante, surtout quand on connaît la sensibilité des réactions automatiques que nécessitent certaines applications comme dans le cas des voitures autonomes. Les données générées par les applications et systèmes IoT nécessitent donc une analyse avancée et une meilleure vitesse de traitement.

L'apprentissage profond (de l'anglais Deep Learning) a récemment donné de très bons résultats dans divers domaines comme celui du traitement d'images, de la reconnaissance de parole, etc. Elle est une technique prometteuse pour l'analyse avancée des données aussi variées que complexes comme celles générées dans l'IoT. L'apprentissage profond est un sous-domaine de l'apprentissage automatique (ou Machine Learning) basé essentiellement sur les réseaux de neurones profonds.

Les réseaux de neurones sont d'excellents outils de modélisation. En raison de leur importance aussi bien dans l'apprentissage automatique que dans l'apprentissage profond, il est donc capital d'investiguer sur les différentes architectures ainsi que leur mode de fonctionnement.

## **3.1 Rappel sur les réseaux de neurones artificiels**

Introduits en 1943 par le neurophysiologiste Warren McCulloch et le mathématicien Walter Pitts, les réseaux de neurones artificiels se sont imposés sur d'autres modèles d'apprentissage automatique en raison de leur forte capacité à modéliser des problèmes et très complexes. Après avoir connu une longue période d'oubli, la réutilisation des réseaux de neurones artificiels a été favorisée par de nombreux facteurs. Entre autres, nous avons : l'augmentation considérable de la puissance de calcul avec l'introduction du GPU, l'augmentation de l'espace mémoire et de stockage chez les ordinateurs, le développement et l'amélioration des algorithmes d'entraînement. Voici autant de facteurs qui ont facilité le traitement des données multidimensionnelles et complexes et ceci dans un délai raisonnable. À tout cela, s'ajoute la disponibilité d'une énorme quantité de données avec l'essor du Big Data. Voici autant d'éléments qui ont favorisé le regain d'intérêt des réseaux de neurones artificiels.

Les premiers succès pratiques ont commencé en 2011 avec Dan Ciresan de l'IDSIA suivi du groupe de Hinton dans le challenge ImageNet en 2012.

À présent, passons en revue quelques définitions et concepts de base.

### 3.1.1 Définitions et concepts de base

#### 3.1.1.1 Définitions

Les **Réseaux de Neurones Artificiels (RNA)** sont un ensemble de neurones artificiels disposés en couches bien structurées.

Un **neurone artificiel** est une unité informatique, une unité de traitement de données dont le fonctionnement est inspiré de celui du neurone biologique. Placé en réseau, le neurone artificiel décide de l'amplitude de la valeur de l'information qui sera renvoyée aux neurones de la couche juxtaposée.

Le tout premier modèle de neurone artificiel fut proposé par le neurologue Warren McCulloch et le psychologue Walter Pitts. Le modèle proposé est une unité logique prenant une ou plusieurs entrées binaires (on/off) et une sortie également binaire. Ce modèle sera donc le point de départ d'une révolution phénoménale dans l'histoire de la modélisation. Avec cela, nous pouvons déjà construire un réseau de neurones artificiels qui calcule toute proposition logique souhaitée. Des modèles qui peuvent apprendre de problèmes de plus en plus complexes ont commencé par émerger. En 1957, un nouveau modèle, le **Threshold Logic Unit (TLU)**, ou parfois **Linear Threshold Unit (LTU)**, est né avec l'invention du **Perceptron** par Frank Rosenblatt. Les entrées et sorties sont désormais des nombres (au lieu de valeurs binaires) et chaque connexion d'entrée est pondérée.

La TLU (voir figure 3.1) calcule une somme pondérée de ses entrées [12],

$$(z = w_1x_1 + w_2x_2 + \dots + w_nx_n = \mathbf{x}^T \mathbf{w}),$$

puis applique une fonction  $f$  (la fonction Heaviside, par exemple) à cette somme et génère le résultat  $y(x)$  défini par :

$$y(x) = f(z)$$

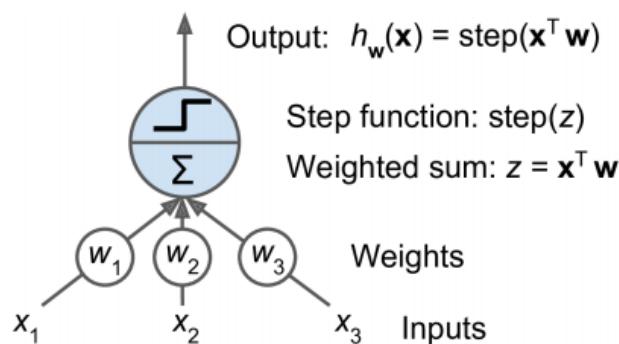


Figure 3.1: Threshold Logic Unit recevant 3 entrées,  $x_1, x_2, x_3$ .

Les réseaux de neurones, tels qu'ils existent actuellement, sont capables d'effectuer des tâches très complexes suite à la modification apportée au niveau de l'unité logique de seuil (TLU). Les fonctions d'étapes de la TLU sont remplacées par d'autres fonctions plus souples.

En termes mathématiques, un neurone artificiel (voir figure 3.2) est une fonction transformant les données d'entrée  $\mathbf{X}$  en une sortie  $\mathbf{y}$  à travers une fonction paramétrée et bornée appelée **fonction d'activation**. Le traitement de l'information au niveau du neurone se fait en deux étapes : dans un premier temps, il calcule une somme pondérée :

$$\mathbf{x}^\top \mathbf{w} + b = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$$

de ses entrées  $\{x_i\}$  à laquelle est ajoutée une certaine valeur de biais puis dans un second temps génère, grâce à la fonction d'activation, la sortie :

$$y(x) = f(\mathbf{x}^\top \mathbf{w}).$$

$\{w_j\}$  est le poids (ou **paramètre**) de connexion associé à l'entrée  $\{x_j\}$  du neurone ; la valeur  $y(x)$  est la **sortie** du neurone et  $b$  le **biais**.

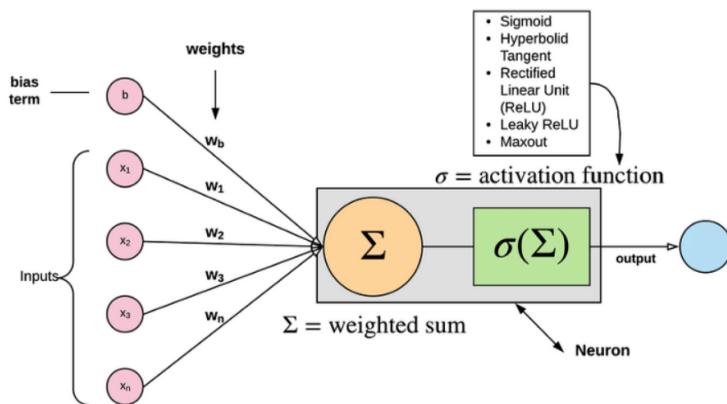


Figure 3.2: Neurone à n entrées,  $x_1, x_2, \dots, x_n$

### Remarques :

- Les valeurs numériques des poids de connexion sont automatiquement ajustées au cours d'une phase appelée **apprentissage** ou **entraînement** du réseau.
- Le biais joue un rôle important dans la formation de la sortie d'un neurone. Ce rôle fut mis en évidence dans la figure 3.3a. Dans cette figure, pour une même fonction sigmoïde, cinq (05) différentes valeurs de biais ont été expérimentées en utilisant les mêmes poids de connexion. Nous pouvons constater que ces courbes sigmoïdes se décalent l'une de l'autre. Cependant, dans la figure 3.3b où le biais est maintenu nul tout en faisant varier le poids, l'on s'aperçoit que toutes les courbes prennent la même valeur lorsque  $x$  est égal à 0. Par conséquent, deux neurones différents (ayant la même fonction d'activation) avec des poids de connexion différents peuvent avoir la même sortie. Ce qui n'est pas bon pour le traitement des données. Ainsi, quel que soit le neurone, le biais ne doit donc pas être initialisé à zéro au début de l'apprentissage.

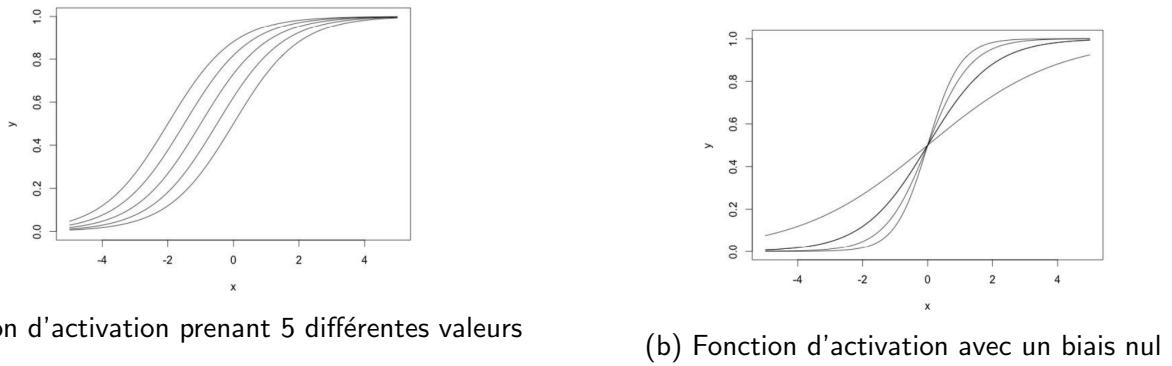


Figure 3.3: Mise en évidence du rôle du biais dans une fonction sigmoïde [13]

### 3.1.1.2 Quelques concepts clés

Avant d'approfondir notre étude, il est important de se familiariser avec certains termes ou expressions.

Les architectures des réseaux neuronaux sont constituées de neurones organisés en **couches** (voir la figure 3.4). Deux neurones sont dits **connectés** lorsque la sortie de l'un d'eux est utilisée comme entrée pour le second. La couche recevant les données est appelée **couche d'entrée** et la **couche de sortie** est celle renvoyant la sortie du réseau. Toute couche située entre les deux précédentes est appelée **couche cachée**.

Les couches cachées jouent un rôle prépondérant dans la sortie du réseau. Elles permettent l'extraction de caractéristiques essentielles lors du traitement des données. Ainsi, la complexité du problème peut nécessiter plus d'une couche cachée.

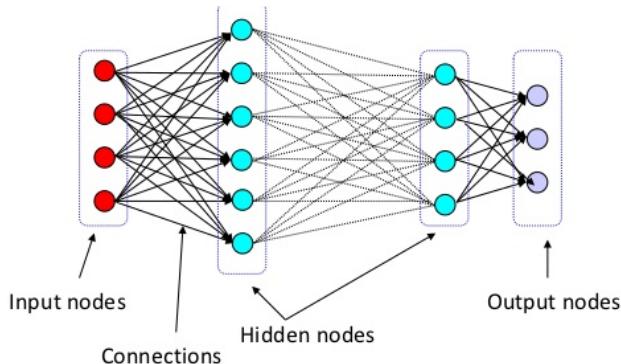


Figure 3.4: Architecture d'un réseau de neurones artificiels

L'efficacité d'un réseau neuronal, si elle dépend de la qualité des données utilisées, dépend également du nombre de couches qu'il contient et du nombre de neurones par couche. Les réseaux neuronaux qui ont plus de deux couches cachées sont appelés **réseaux de neurones profonds**. Aujourd'hui, le nombre typique de couches utilisées dans un réseau de neurones profonds est de l'ordre des centaines.

Les neurones sont les éléments essentiels du processus de traitement des données. Leur insuffisance dans une couche peut être la cause du manque de performance du réseau. Il est donc conseillé d'utiliser une technique permettant le choix optimal du nombre de neurones par couche. Du nombre de ces techniques, nous pouvons citer la validation croisée.

### 3.1.2 Fonction d'activation

Le processus d'implémentation d'un réseau de neurones nécessite la construction de plusieurs fonctions d'activation chacune effectuant une tâche bien spécifique dans le processus de traitement des informations qui arrivent aux neurones. Il s'agit d'une fonction mathématique, souvent non-linéaire, permettant de mettre à l'échelle les valeurs passant par un neurone du réseau entraînant ainsi une réponse de ce dernier.

Il y existe plusieurs types de fonctions d'activation ; chacune d'elles étant utilisée en fonction de ses propriétés. Les plus couramment utilisées dans la pratique sont les suivantes :

#### 1. Fonction sigmoïde

La fonction sigmoïde, aussi connue sous le nom de fonction logistique, appartient à une famille de fonctions en forme de S. Elle prend en entrée une valeur réelle et renvoie une valeur réelle de sortie comprise entre 0 et 1. Elle est définie par :

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

où  $z = \sum_{k=1}^n w_k x_k + b$ .

La figure 3.5 nous montre la courbe de la fonction sigmoïde.

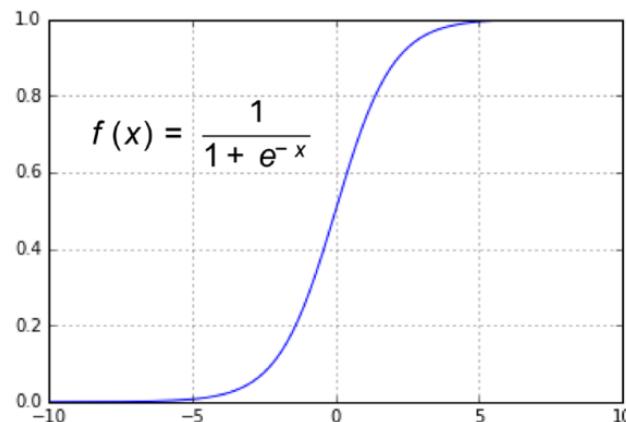


Figure 3.5: Fonction sigmoïde

La fonction sigmoïde convient aux réseaux de neurones dans lesquels seule une sortie positive est nécessaire. Elle est souvent utilisée dans la couche de sortie des modèles de classification binaire.

Cependant, il existe quelques limitations dans l'utilisation de cette fonction. Le gradient de la fonction sigmoïde étant donné par :  $f'(u) = f(u)[1 - f(u)]$ , il s'annule donc dans des zones où la fonction sigmoïde est 0 ou 1. C'est le **problème de disparition du gradient (vanishing gradient)**. Il intervient chez un neurone (en général, au niveau des couches cachées) lorsque le gradient de la fonction coût est extrêmement petit, empêchant ainsi la mise à jour effective de la valeur des poids de connexion (voir sous-section 3.1.4.1). Ce qui a pour conséquence la perte de l'information.

## 2. La fonction tangente hyperbolique

Une autre fonction d'activation qui marche presque toujours mieux que la fonction logistique est **fonction tangente hyperbolique (tanh)**. Elle est donnée par :

$$\tanh(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}$$

Elle transforme toute entrée réelle en une valeur comprise entre -1 et 1 (voir figure 3.6). Elle a une propriété intéressante de centrage à zéro de la sortie ; propriété facilitant l'apprentissage du réseau. Toutefois, elle est également confrontée au problème du vanishing gradient car sature à ses extrémités et donc son gradient devient extrêmement petit. Mais elle reste néanmoins très utilisée.

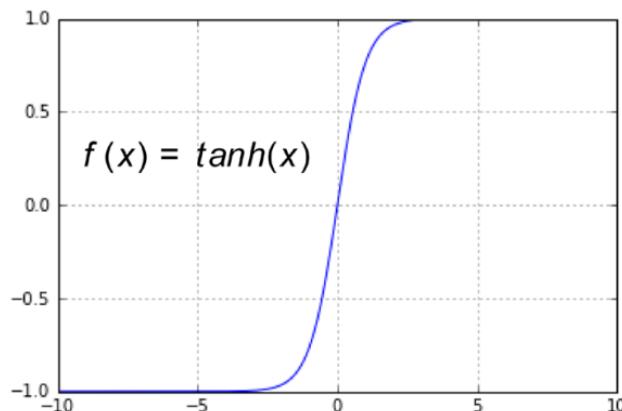


Figure 3.6: Fonction tangente hyperbolique

## 3. La fonction ReLU (Rectified Linear Unit)

La fonction ReLU est la fonction d'activation la plus utilisée de nos jours [21] surtout pour les unités cachées [15]. Il s'agit d'une fonction linéaire par morceaux donnée par :

$$\text{ReLU}(z) = \max\{0, z\}$$

Elle permet un entraînement rapide du neurone et conduit à des gradients plus importants et cohérents.

Toutefois, la fonction ReLU présente une limitation lorsque la somme pondérée est négative : c'est le **problème du ReLU mourant** ou (**Dying ReLU**) dû à la nullité du gradient.

L'unité ReLU peut se figer et ainsi ne contribue plus à la sortie du réseau. Ce problème a été corrigé avec l'introduction des variantes du ReLU. Et du nombre de ceux-ci, nous avons : le **Leaky ReLU (LReLU)**, le **Parametric ReLU (PReLU)** et l' **Exponential Linear Unit (ELU)**.

Les fonctions ReLU ont l'avantage d'accélérer l'apprentissage du modèle. Le problème de la disparition du gradient est moins fréquente au niveau des fonctions ReLU. Cependant, elles sont souvent confrontées au problème d'**explosion du gradient** (le gradient de la fonction ReLU devient très grand pour certaines valeurs causant ainsi l'instabilité du réseau). Il est avantageux d'alimenter le réseau des données en des lots de petite taille lors de son entraînement afin de contourner ce problème.

#### 4. La fonction Softmax

Aussi appelée fonction exponentielle normalisée, la fonction softmax transforme la somme pondérée de ses entrées en une sortie dans la plage de (0, 1) de sorte que la somme cumulée de toutes les sorties soit égale 1. Elle est généralement utilisée comme une fonction d'activation dans les neurones de la dernière couche pour des problèmes de classification multi-classes. Chaque sortie du neurone pourra représenter la probabilité d'appartenance à l'une des classes.

Dans l'exemple ci-après, la fonction softmax donnée par :

$$\sigma(z)_k = \frac{\exp(z_k)}{\sum_{i=1}^3 \exp(z_i)}, i = 1, 2, 3$$

peut être utilisée pour une classification à 3 classes ; la sortie  $\sigma(z)_k$  du neurone  $k$  étant la probabilité pour que l'instance  $z = (z_1, z_2, z_3)$  appartienne à la classe  $k$ . La figure 3.7 nous donne une illustration de la courbe de cette fonction.

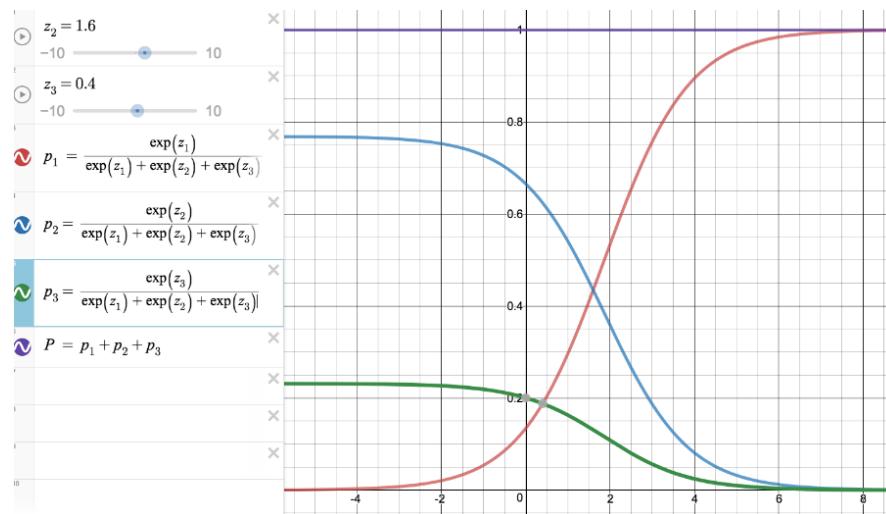


Figure 3.7: Fonction Softmax à trois sorties

### Remarques :

- La propriété de non-linéarité des fonctions d'activation est très importante à partir du moment où un empilement profond de couches constituées de neurones à fonction d'activation linéaire est équivalent à une seule couche. En veillant sur cette propriété, on pourra donc former des réseaux très performants puisque la résolution de problèmes très complexes est souvent liée à la profondeur du réseau de neurones.
- Il est aussi important par exemple que la fonction d'activation ne soit pas un polynôme afin d'éviter le problème d'explosion du gradient.
- Les propriétés de la fonction d'activation choisie influent en effet sur le fonctionnement du neurone artificiel. Il est donc important de choisir une fonction d'activation en tenant compte de des propriétés afin d'obtenir un bon modèle.

### 3.1.3 Réseau entièrement connecté, réseau partiellement connecté

Un **réseau de neurones entièrement connecté** consiste en une série de couches entièrement connectées. Une **couche entièrement connectée** est celle pour laquelle tous ces neurones sont connectés à tous les neurones de la couche suivante. Un réseau qui n'est pas entièrement connecté est dit **partiellement connecté**.

Les couches entièrement connectées sont couramment utilisées dans les réseaux de neurones récurrents (voir section 3.2.5) et les réseaux de neurones convolutifs (voir section 3.2.6). L'implémentation d'un réseau entièrement connecté nécessite une quantité importante de ressource de stockage et en calculs. Mais, pour traiter les problèmes de stockage et de calcul sans nuire à la précision du réseau, on remplace certaines couches par celles qui sont peu connectées. Dans la pratique, on supprime certaines connexions entre les neurones en fixant simplement les poids à zéro.

Dans la figure 3.8, nous avons une illustration d'une couche entièrement connectée et de celle partiellement connectée.

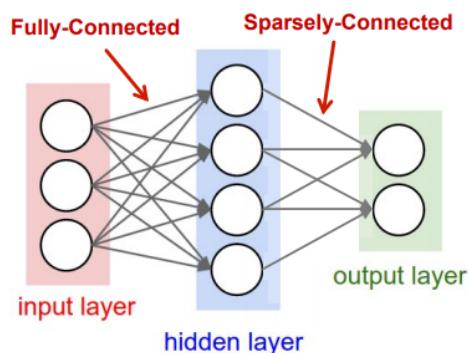


Figure 3.8: Couche entièrement connectée vs couche partiellement connectée

### 3.1.4 Apprentissage ou entraînement d'un réseau neuronal

Entraîner un modèle, en particulier un réseau neuronal, signifie minimiser l'erreur de prédiction sur les données d'apprentissage. Ce qui revient à rechercher les paramètres du modèle qui minimisent la fonction de coût sur l'ensemble des données d'apprentissage. Nombreux sont les algorithmes et techniques utilisés à cet effet. Passons en revue certains d'entre eux.

#### 3.1.4.1 Algorithme de la descente du gradient

La descente du gradient est un algorithme d'optimisation très important dans la recherche de solutions optimales à un large éventail de problèmes. L'idée générale de cet algorithme est de minimiser de manière itérative la fonction de coût en utilisant son gradient.

#### 3.1.4.2 Algorithme de rétro-propagation

Introduit en 1986 par David Rumelhart, Geoffrey Hinton et Ronald Williams, l'algorithme de rétro-propagation, autrefois abandonné en raison de ses exigences en calcul, bénéficie aujourd'hui des récents progrès des processeurs graphiques, ou GPU (de l'anglais Graphics Processing Unit) ; ce qui est d'une grande utilité surtout pour l'entraînement de réseaux neuronaux multicouches.

La rétro-propagation est un algorithme itératif basé sur l'algorithme du gradient qui permet d'ajuster les poids de connexion du réseau neuronal. Elle consiste en une alternance de deux phases (**une phase montante (ou forward)** et **une phase descendante (ou backward)**) à travers le réseau [8, 12]. La mise à jour des poids se fait au moment de la phase descendante après que le gradient soit calculé.

Le poids et le biais sont respectivement mis à jour suivant les équations (3.1.1) et (3.1.2) [28, 31].

$$w_{updated}^l = w_{old}^l - \eta \left( \frac{\partial E}{\partial w_n^l} \right) \quad (3.1.1)$$

$$b_{updated}^l = b_{old}^l - \eta \left( \frac{\partial E}{\partial b_n^l} \right) \quad (3.1.2)$$

- $w_{updated}^l$  et  $b_{updated}^l$  sont respectivement le poids mis à jour et le biais mis à jour au niveau de la couche  $l$ .
- $w_{old}^l$  et  $b_{old}^l$  sont respectivement le poids et le biais de la couche  $l$  avant la mise à jour.
- $\eta$  est le taux d'apprentissage.
- $E$  est l'erreur.

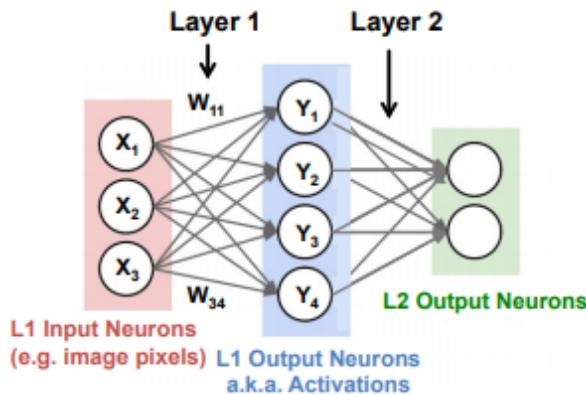
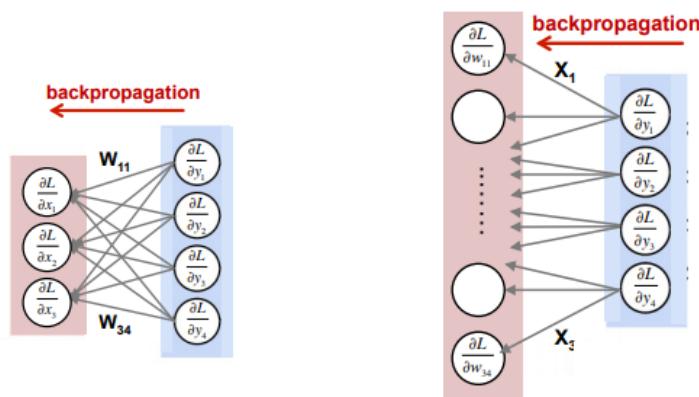


Figure 3.9: Un exemple de forward propagation à travers un réseau neuronal



(a) Compute the gradient of the loss relative to the filter inputs      (b) Compute the gradient of the loss relative to the weights

Figure 3.10: Un exemple de backward pass à travers un réseau neuronal

**NB :** Il est très important d'initialiser le biais et des poids de connexion avant de commencer l'entraînement du réseau de neurones. Aussi, cette initialisation doit se faire de façon aléatoire en évitant de mettre à 0 tous les poids.

Le réseau neuronal au cours de son entraînement est alimenté de données. Cette alimentation en données peut se faire grâce à plusieurs techniques. Et du nombre de celles-ci, nous avons l'apprentissage par lots (ou batch learning) et l'apprentissage en ligne (ou online learning).

### 3.1.4.3 Batch and Online Learning

#### 1. Batch Learning :

L'apprentissage par lots est un algorithme permettant au système d'entraîner le modèle d'apprentissage en utilisant toutes les données disponibles [12]. Le réseau est alimenté à chaque fois par des données en lots, mais la mise à jour des poids de connexion se fait

sur l'ensemble des échantillons d'apprentissage fournis. Ce type d'apprentissage nécessite souvent beaucoup de temps de traitement et des ressources informatiques.

## 2. Online learning :

L'apprentissage incrémental aussi appelé apprentissage en ligne est un algorithme permettant d'entraîner le réseau de manière incrémentielle. Les instances de données sont alimentées de manière séquentielle, soit individuellement, soit par petits lots appelés mini-batchs. Contrairement au batch learning, la mise à jour des poids de connexion se fait à chaque séquence d'alimentation du réseau. Cette technique d'alimentation des données est idéal pour l'entraînement des modèles embarqués dans la plupart des appareils connectés (comme les smartphones, les voitures autonomes, etc.) [12]. Mieux encore, c'est une bonne option pour les appareils aux ressources informatiques limitées (CPU, faible espace mémoire et espace disque, etc.).

**NB :** Il faut noter que l'entraînement des modèles se fait hors des appareils sus-mentionnés. Seulement, ils reçoivent la mise à jour d'un modèle donné après chaque séquence d'entraînement. Ce qui permet leur permet de s'améliorer ou de s'adapter aux nouveaux flux de données.

# 3.2 Architectures des réseaux de neurones

Les réseaux de neurones se présentent sous différentes architectures ; chacune avec ses particularités.

## 3.2.1 Machine de Boltzmann restreinte

Les machines de Boltzmann restreinte ont été conçues à l'origine à des fins d'apprentissage non supervisé.

Une machine Boltzmann restreinte est un réseau neuronal artificiel dont l'architecture est un ensemble de deux couches : une couche **visible** constituée d'unités dites **visibles** et une autre couche dite **cachée** constituée d'unités dites **cachées**. Les neurones d'une même couche sont indépendants entre eux. Cependant, l'état d'une unité visible peut affecter celui d'une unité cachée, et vice-versa.

Une structure typique d'un RBM est présentée dans la figure 3.11.

Les modèles basés sur la RBM sont généralement utilisés pour initialiser les poids d'un réseau de neurones dans des applications plus récentes.

## 3.2.2 Réseaux neuronaux à action directe

L'architecture des réseaux neuronaux à action directe (en anglais, **Feed-Forward Neural Networks**) (voir figure 3.12) est la première et la plus simple des architectures des réseaux neuronaux. L'expression Feed-forwarded fait tout simplement référence à la procédure du traitement

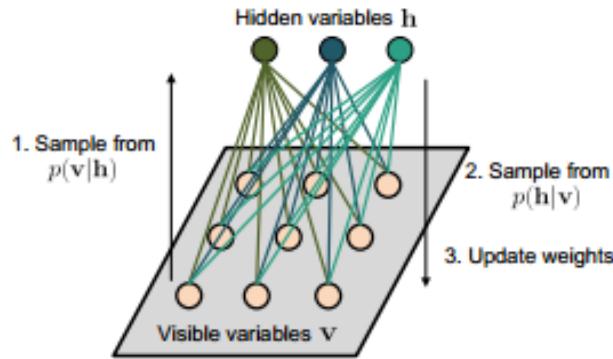


Figure 3.11: Architecture d'une Machine de Boltzmann restreinte.

des données par le réseau neuronal. Dans ce type de réseau, l'information ne se propage que dans une seule direction : des neurones d'entrée aux neurones de sortie, en passant par les couches cachées [28]. Aussi, les connexions entre les neurones sont établies avec une restriction de la formation de boucles ou de cycles et il n'y a pas de connexions entre les neurones d'une même couche.

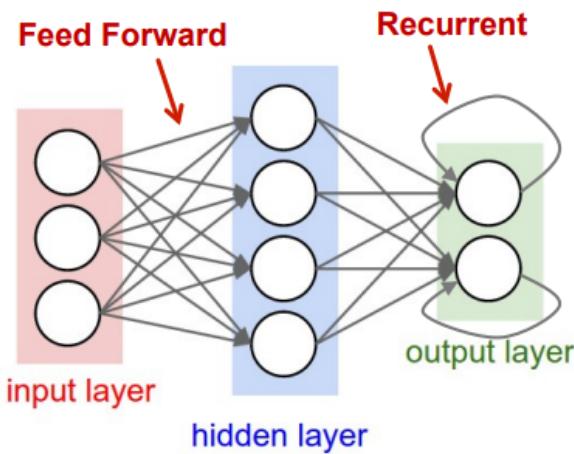


Figure 3.12: Architecture d'un réseaux neuronal à action directe

### 3.2.3 Perceptron multicouche

Le Perceptron multicouche appartient à la catégorie des réseaux de neurones à action directe. Son architecture est constituée de trois couches : une couche d'entrée, une couche cachée et une couche de sortie.

Les couches cachées peuvent être constituées d'autant de neurones que possible. Par ailleurs, ces neurones ont la particularité de n'utiliser que des fonctions sigmoïdes comme fonctions d'activation.

La figure 3.13 est celle d'un Perceptron multicouche à quatre couches.

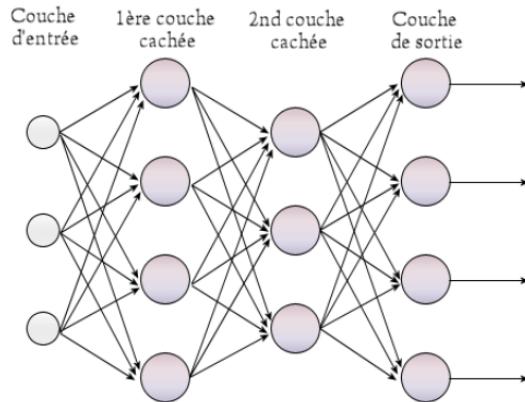


Figure 3.13: Perceptron multicouche à quatre couches

### 3.2.4 Auto-encodeur

Un Auto-encodeur (AE) est une variante du Perceptron Multicouche [23]. Il tente de reconstruire ses données d'entrée dans la couche de sortie par un processus de codage et de décodage.

Son architecture consiste en un réseau à trois couches (voir figure 3.14) dans lequel les couches d'entrée et de sortie ont le même nombre de neurones. Sa couche intermédiaire compte moins de neurones que les autres couches.

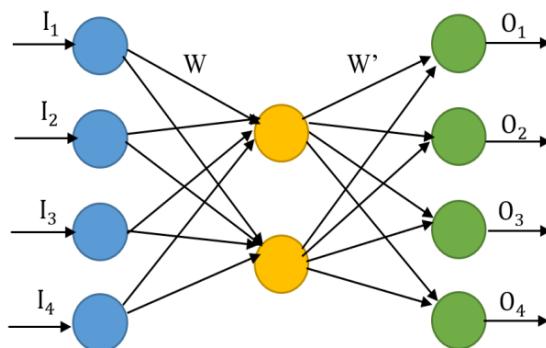


Figure 3.14: Autoencoder

### 3.2.5 Réseau de Neurones récurrents

Introduits dans les années 1980, les réseaux neuronaux récurrents (RNR) sont adaptés et très efficaces pour les problèmes dans lesquels les données d'entrée se présentent sous forme de séquence (série d'entités où l'ordre est important) [15]. Contrairement aux réseaux de neurones à action directe, l'architecture d'un réseau de neurones récurrent n'a aucune contrainte dans le sens des connexions des neurones. Ainsi, ces connexions peuvent être des boucles ou des cycles. Les RNRs ont des connexions récurrentes qui propagent l'information entre les neurones d'une même couche.

La principale caractéristique de ces réseaux est que l'information y est renvoyée en boucle. Certaines opérations intermédiaires génèrent des valeurs qui sont stockées en interne dans le réseau et utilisées comme entrées pour d'autres opérations en conjonction avec le traitement d'une entrée ultérieure [28]. Cette propriété donne aux réseaux neuronaux récurrents une sorte de mémoire qu'ils peuvent utiliser pour mieux comprendre les données séquentielles (séries temporelles par exemple). Ce qui constitue une force dans le traitement des informations contextuelles.

Les réseaux de neurones récurrents se composent d'une ou plusieurs couches. Le modèle d'une seule couche le plus connu est le réseau temporel de Hopfield.

Étant donné une séquence d'entrées  $x = x_1, x_2, \dots, x_t$ , un réseau neuronal récurrent standard effectue les opérations suivantes :

$$s_t = \sigma_s(W_s x_t + W_s x_{t-1} + b_s)$$

$$h_t = \sigma_h(W_h x_t + b_h)$$

où :

- $s_t$  représente l'état du réseau au temps t ;
- $h_t$  est la sortie (cachée) du réseau au temps t ;
- $W_x$  et  $W_h$  sont des poids vectoriels ;
- $b_s$  et  $b_h$  sont des biais.

L'architecture d'une couche récurrente peut être observée dans la figure 3.15

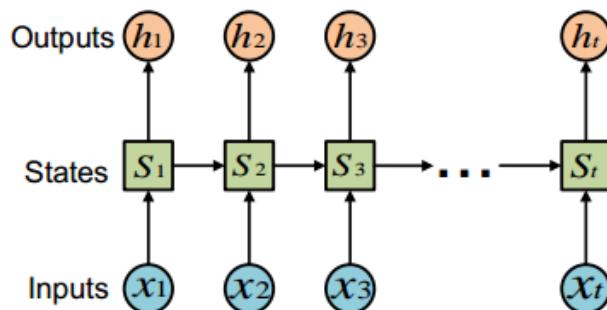


Figure 3.15: Architecture d'une couche récurrente

Parmi les variantes des RNN, nous avons : le réseau LSTM (Long Short-Term Memory).

### 3.2.6 Réseau de neurones convolutifs (CNN)

Également appelés **ConvNets**, les réseaux de neurones convolutifs, sont une famille de réseaux de neurones artificiels qui ont révolutionné l'apprentissage profond (ou **Deep Learning**). Alors que d'autres architectures de réseaux neuronaux ont montré leurs limites dans l'analyse de données multidimensionnelles telles que les images et les vidéos, les réseaux neuronaux convolutifs se sont avérés plus efficaces pour l'analyse et le traitement de ce type de données [10].

L'architecture d'un CNN se repartit en deux grands blocs. Le premier bloc se compose de couches convolutives suivies d'une couche de pooling, puis de couches convolutives, puis d'une autre couche de pooling, et ainsi de suite (voir figure 3.16). C'est d'ailleurs ce bloc qui fait la puissance des réseaux de neurones convolutifs (CNN). Il serait donc intéressant de comprendre comment fonctionnent chacune de ces deux couches.

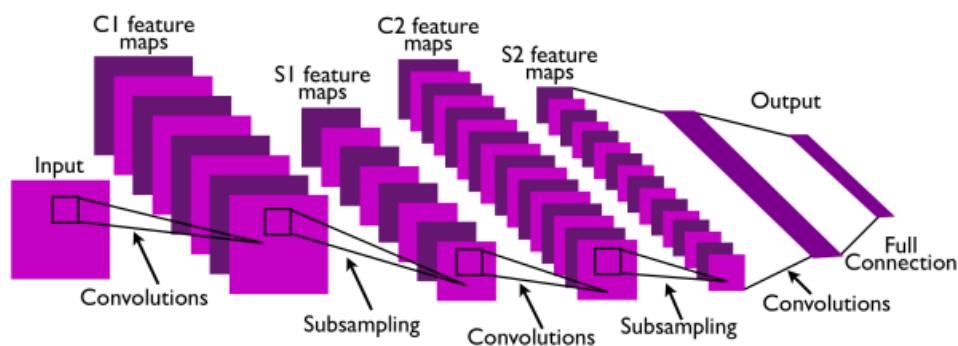


Figure 3.16: Une architecture ConvNet typique avec deux étapes de fonctionnalités [18]

#### 3.2.6.1 Couche de convolution ou (Convolutional Layer)

Les couches de convolution constituent toute la force des réseaux de neurones convolutifs [32]. Elles permettent de contourner l'extraction autrefois manuelle des caractéristiques (features) des données d'entrée.

Une couche convulsive fonctionne comme un extracteur automatique de caractéristiques. Les différentes caractéristiques des données d'entrée sont extraites par les neurones dans les couches de convolution grâce aux filtres à travers une opération appelée **opération de convolution**. Une illustration sur le fonctionnement cette opération est donnée dans la figure 3.17. Un filtre, ou noyau de convolution, est un ensemble de poids de connexion utilisés par les neurones d'une même couche de convolution. Les filtres sont initialisés avant l'apprentissage du réseau puis mirent à jour par rétropropagation du gradient. Ainsi, une carte de caractéristiques (ou feature map) est générée par filtre.

Dans le cadre de la reconnaissance d'image par exemple, l'image prise en entrée est découpée en petites zones ou champs réceptifs. Chaque champ réceptif sera traité individuellement par les neurones en utilisant les filtres. De chaque couche de convolution, on obtient une image intermédiaire qui est une compilation des fonctionnalités générées à partir de chacun de ces

filtres.

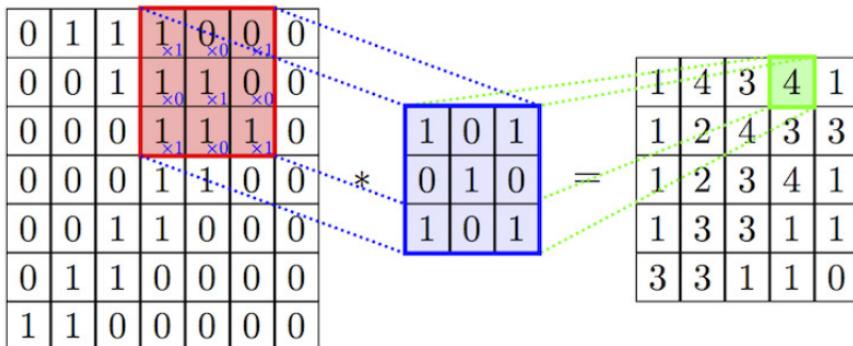


Figure 3.17: Opération de convolution

### 3.2.6.2 Couche de regroupement (ou Pooling layer)

Elle est souvent placée entre deux couches de convolution. La couche de pooling a pour but de réduire d'un certain rapport la taille de la carte d'entités prise en entrée [18] en utilisant une opération appelée **opération de mise en commun** ou **opération de pooling**.

Les fonctionnalités générées dans la couche de convolution sont combinées grâce à l'opération de pooling tout en préservant les caractéristiques les plus importantes. Cette opération permet au réseau d'être robuste et de résister aux petites variations des données [28].

Dans le cadre d'une image par exemple (voir figure 3.18), l'opération de pooling est effectuée en remplaçant simplement un groupe de pixels par leur valeur maximale (s'il s'agit du **max-pooling**) ou par leur valeur moyenne (s'il s'agit du **average-pooling**).

L'opération de mise en commun présente l'avantage de réduire l'exigence en mémoire et en puissance de calcul dans les couches suivantes. On améliore ainsi l'efficacité du réseau.

2x2 pooling, stride 2				Max pooling	Average pooling
9	3	5	3		
10	32	2	2	32	18
1	3	21	9	6	3
2	6	11	7	21	12

Figure 3.18: Différentes formes d'opération de pooling

Après l'introduction des réseaux de neurones convolutifs (CNN) en 1980 par Kunihiko Fukushima [12], la structure de ceux-ci a subi de nombreuses modifications et améliorations afin de faire face à la complexité des problèmes qui se posent constamment. De nombreuses variantes de

cette architecture fondamentale se sont développées, conduisant ainsi à des avancées étonnantes dans le domaine. Parmi ces variantes, nous avons : le **LeNet**: il s'agit d'une structure de 5 à 7 couches ; **AlexNet** une architecture de cinq couches convolutives, dont certaines sont suivies par des couches de max-pooling, et de trois couches entièrement connectées. Nous avons également **VGGNet** une architecture de 16 couches, **GoogLeNet** une architecture de 22 couche, etc.

Ainsi, il convient donc de choisir la bonne architecture (nombre d'unités, nombre de couches) qui convient à son problème. Bien que certaines architectures sont plus réputées que d'autres, il est possible de combiner différentes architectures pour atteindre son objectif.

### 3.2.7 Réseau de neurones profond

De nombreux défis liés à l'Intelligence Artificielle ont été surmontés grâce aux réseaux de neurones profonds (en anglais Deep Neural Network (DNN)) c'est-à-dire réseaux neuronaux ayant plusieurs couches entre les couches d'entrée et de sortie. Ces réseaux conviennent aux tâches complexes telles que le traitement d'images, la reconnaissance vocale, la traduction automatique, etc.

L'utilisation de tels réseaux donna naissance à une technique d'apprentissage automatique : le **Deep Learning**.

## 3.3 Deep Learning

L'apprentissage profond ou le **Deep Learning** est une sous-branche du Machine Learning capable d'apprendre à des niveaux d'abstraction très élevés. Il a révolutionné le domaine de l'intelligence artificielle avec la création d'applications intelligentes.

Le Deep Learning a connu un énorme succès ces dernières années surtout dans le domaine du traitement d'image, de la vision par ordinateur, de la reconnaissance vocale, de la traduction automatique et bien d'autres encore. Un excellent exemple de ce succès peut être illustré avec le défi ImageNet. Le principal avantage de l'apprentissage en profondeur par rapport à l'apprentissage automatique traditionnel est en fait l'extraction automatique des fonctionnalités, grâce à laquelle l'ingénierie de fonctionnalités artisanales coûteuse peut être contournée. Les algorithmes d'apprentissage profond extraient hiérarchiquement les connaissances des données brutes. Cette technique d'apprentissage est essentiellement basée sur des architectures particulières : les réseaux de neurones profonds.

Tout comme l'apprentissage automatique, l'apprentissage profond est généralement en deux types : l'apprentissage supervisé et l'apprentissage non supervisé.

1. L'**apprentissage supervisé** est une technique pour laquelle l'apprentissage se fait à l'aide de données étiquetées. Le but de l'algorithme supervisé est de produire un modèle capable de prédire la sortie avec exactitude d'une entrée donnée.
2. Contrairement à l'apprentissage supervisé, l'**apprentissage non-supervisé** est une approche d'apprentissage avec des données non étiquetées. Dans ce cas, le système apprend

la représentation interne des données. L'objectif est de découvrir des caractéristiques importantes ou des relations ou des structures inconnues dans les données d'entrée [6].

**NB :** Le système peut parfois apprendre avec des ensembles de données partiellement étiquetés. Cette approche n'est ni supervisée, ni non-supervisée. Elle s'appelle **apprentissage par renforcement**, parfois appelé **apprentissage semi-supervisé**. Le terme renforcement fait référence au fait que le système apprend des erreurs précédentes pour s'améliorer et se renforcer dans son apprentissage.

Le Deep Learning a connu une technique très révolutionnaire permettant de former des réseaux neuronaux profonds en utilisant d'autres modèles préformés au lieu d'entraîner le réseau à partir de zéro. Une telle technique est appelée apprentissage par transfert (en anglais, transfer learning).

### 3.3.1 Transfer learning

L'apprentissage par transfert est l'une des techniques les plus importantes de l'apprentissage profond. Un ensemble de données de 1 000 ou même de 25 000 images est parfois trop petit pour l'entraînement des réseaux de neurones profonds. Les réseaux de neurones profonds ont besoin d'être entraîner avec plusieurs centaines de milliers d'images afin de capter les différentes fonctionnalités de l'objet. La technique d'apprentissage par transfert vient alors alléger cette lourde tâche en permettant d'utiliser un modèle déjà entraîné sur un très grand ensemble de données tout en nous évitant d'entraîner le modèle à partir de zéro. Ainsi, on gagne non seulement en précision puisque le modèle profite des fonctionnalités déjà apprises et mais aussi en temps d'entraînement. Les modèles basés sur l'apprentissage par transfert sont souvent plus performants que les modèles entraînés de zéro. Une illustration en est donnée dans figure 3.19.

Quelques exemples de modèles pré-entraînés: SSD29, RCNN, Fast RCNN, YOLO, Mask RCNN, etc.

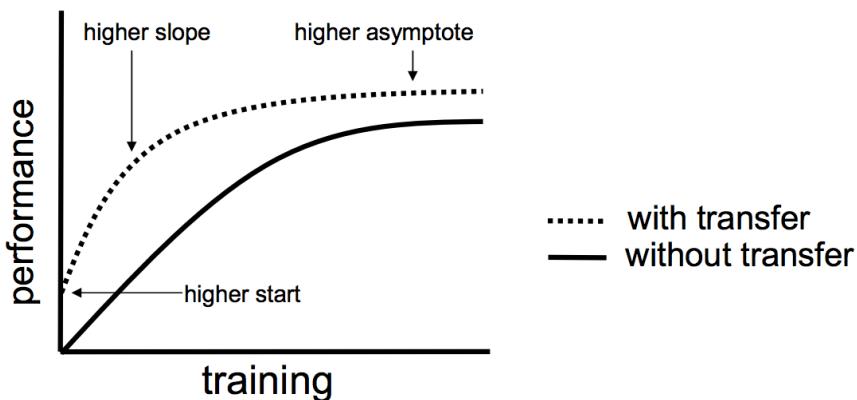


Figure 3.19: Performance modèle avec transfert vs modèle sans transfert

Différentes tâches nécessitent l'utilisation de la technique de l'apprentissage profond. Pour ce qui du traitement d'image, les tâches les plus fréquentes sont : la classification, la localisation, la segmentation et la détection d'objet.

### 3.3.2 Quelques tâches de Deep Learning sur le traitement d'images

#### 3.3.2.1 Classification et localisation d'images

La classification d'un objet dans une image consiste à identifier la présence de cet objet dans cette image et prédire la classe à laquelle il appartient.

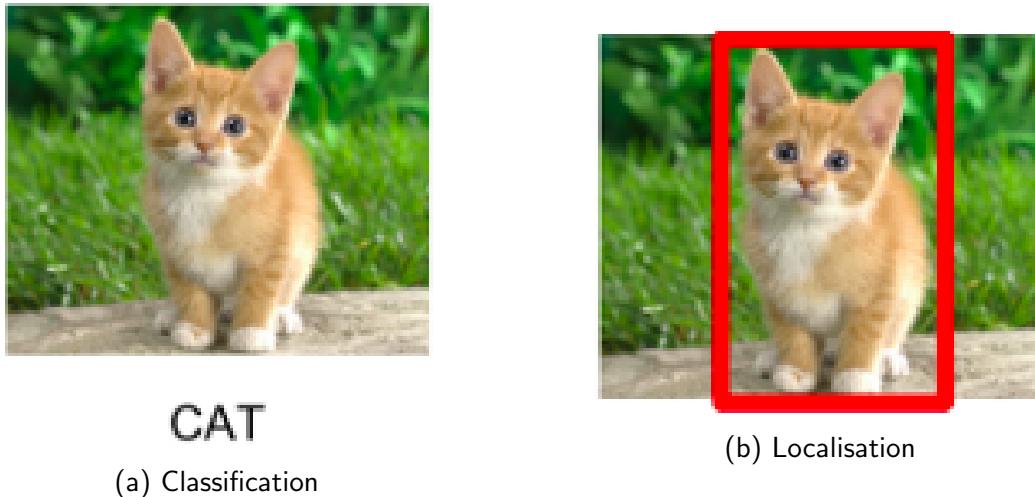


Figure 3.20: Classification vs localisation

Quant à la localisation, elle consiste à identifier un objet sur une image en traçant un cadre de délimitation autour de cet objet.

#### 3.3.2.2 La segmentation sémantique

C'est la tâche de classer chaque pixel d'une image en fonction de la classe de l'objet auquel il appartient (par exemple, route, voiture, piéton, bâtiment, etc.). Différents objets d'une même classe ne sont pas distingués. Par exemple, dans la figure 3.21, nous pouvons remarquer qu'après segmentation, tous les objets d'une même classe se retrouvent comme un gros morceau de pixels.

**La segmentation d'instance** (voir figure 3.21) est une tâche similaire à la segmentation sémantique. Mais ici, au lieu de fusionner tous les objets de la même classe en un seul gros morceau, chaque objet se distingue des autres.

#### 3.3.2.3 La détection d'objets

La détection d'objets constitue un grand défi dans le domaine de la vision par ordinateur. C'est la capacité pour une machine à prédire la classe de chaque objet d'une image et à tracer un cadre de délimitation autour de lui. C'est donc une tâche pour laquelle la machine classifie et localise à la fois un ou plusieurs objets dans une image donnée (voir figure 3.22).

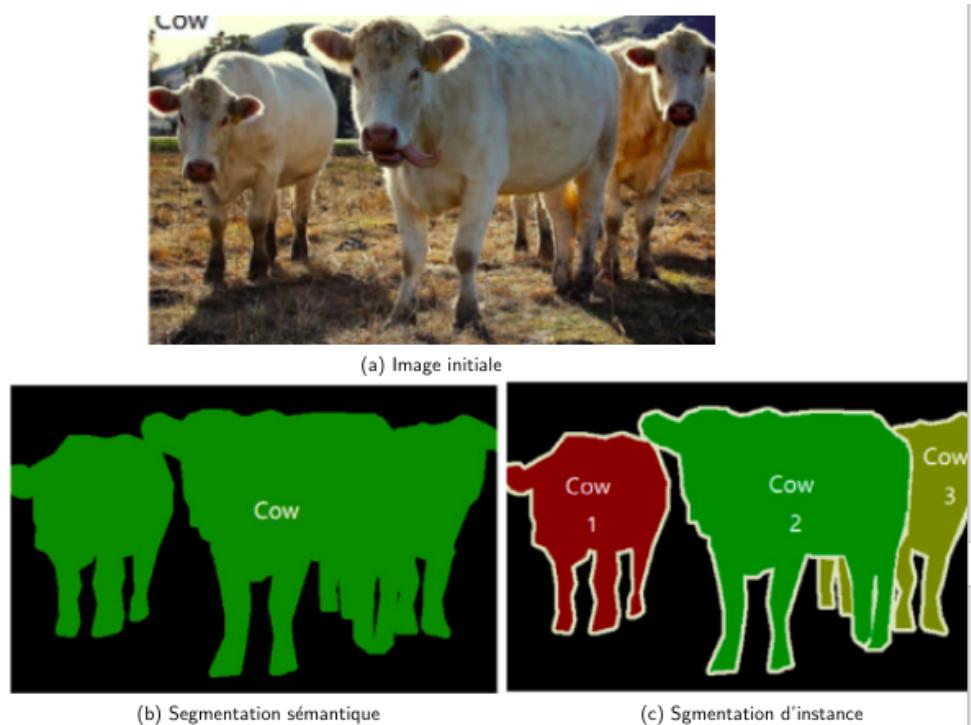


Figure 3.21: Segmentation sémantique vs segmentation d'instance

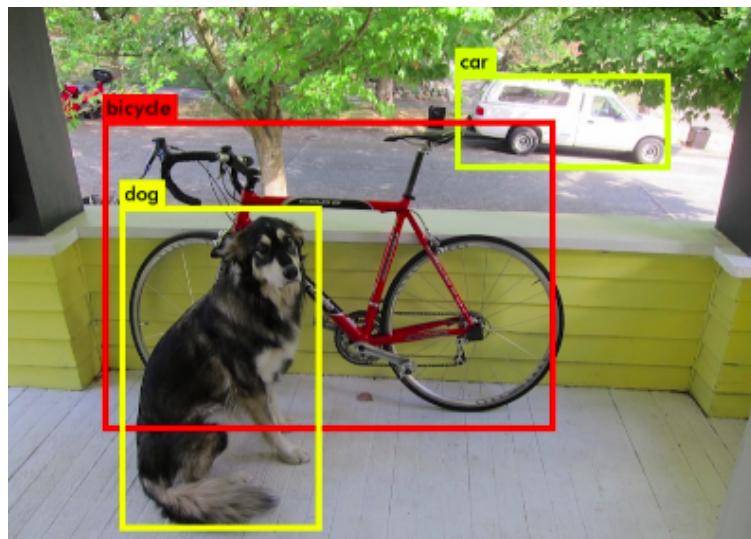


Figure 3.22: Exemple de détection d'objet [25]

### 3.3.2.4 Évaluation d'un modèle de détection d'objet

Dans une tâche de détection d'objet, le modèle est évalué sur sa capacité à prédire avec exactitude l'objet cible. La boîte englobante prédite peut ne pas recouvrir la véritable terrain, c'est-à-dire les limites réelles de l'objet comme nous pouvons le voir dans la figure 3.23. Plusieurs indicateurs sont donc utilisés afin mesurer l'exactitude ou l'acceptabilité de la prédiction. Du nombre de ces

indicateurs, nous avons : l'IoU(Intersection over Union), la Précision et le Rappel.

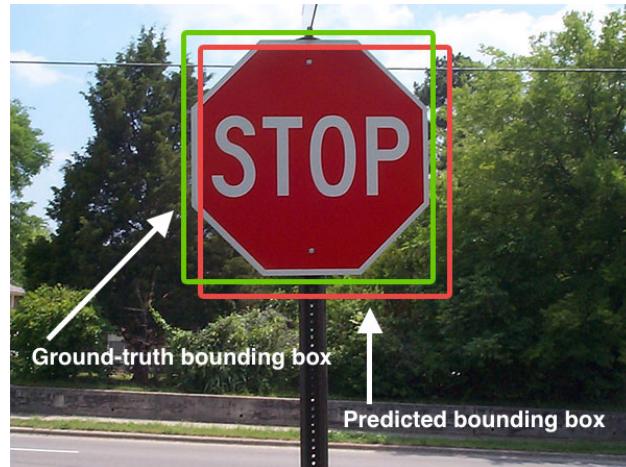


Figure 3.23: Boîte englobante prédictive (en rouge) - boîte englobante de vérité terrain (en vert)

1. **L'IoU** De l'anglais, **Intersection over Union**, l'**IoU** est l'indicateur utilisé pour mesurer à quel point la boîte englobante prédictive chevauche parfaitement avec la vérité terrain. L'IoU est donné par le rapport de la zone d'intersection et de la zone d'union de la boîte englobante prédictive et de la boîte englobante de la vérité terrain (voir figure 3.24).

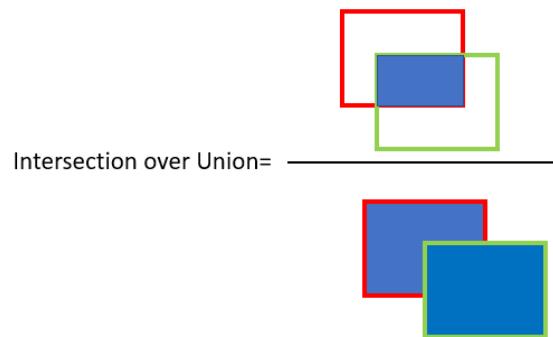


Figure 3.24: IoU

En terme mathématiques, l'IoU est donnée par :

$$IoU = \frac{\text{Vrai Positif}}{\text{Vrai Positif} + \text{Faux Positif} + \text{Vrai négatif}}$$

En se donnant une valeur de l'IoU, on définit ainsi un seuil d'acceptabilité du résultat produit par le modèle. Dans la figure 3.25, nous pouvons observer différents résultats selon de différents seuils.

Ainsi, selon la valeur de seuil fixé, une prédiction peut être classée comme étant un **vrai positif**, un **vrai négatif**, un **faux positif**, ou bien **faux négatif**. On parle de vrai positif lorsque l'objet est correctement détecté. Une prédiction est dite faux positif lorsque l'objet



Figure 3.25: En rouge, nous avons la prédiction et la vérité terrain En vert.

détecté n'est pas celui qui est attendu. On parle de faux négatif lorsque le modèle ne détecte pas l'objet alors qu'il est bien présent et enfin, une prédiction est dite vrai négatif lorsque le modèle ne fait aucune prédiction en absence de l'objet visé. Par exemple, si nous fixons l'IoU sur 0,5, alors pour l'IoU > 0,5, la détection d'objet est vue comme un **vrai positif (TP)**. Et si IoU < 0,5, alors il s'agit d'une mauvaise détection et c'est un faux positif (FP). Ainsi, à partir de ces classifications, deux indicateurs sont généralement calculés : la Précision et le Rappel.

## 2. La Précision

La Précision est le pourcentage de détections correctes. Il met en évidence l'exactitude des prédictions faites par le modèle [20]. Elle est donnée :

$$\text{Précision} = \frac{\text{Vrai Positif}}{\text{Vrai Positif} + \text{Faux Positif}}$$

3. Quant au **Rappel**, il mesure la capacité du modèle à prédire l'ensemble des résultats attendus [20]. Il est défini par :

$$\text{Rappel} = \frac{\text{Vrai Positif}}{\text{Vrai Positif} + \text{Faux négatif}}$$

## La métrique mAP

La métrique d'évaluation **mAP** (de l'anglais, mean Average Precision) est une métrique couramment utilisée afin d'évaluer les performances d'un modèle de détection d'objets. Le mAP est la moyenne des AP (Average Precision). Elle est donnée par :

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i, \text{ N étant le nombre de classes et } AP_i \text{ l'Average Precision de la classe i}$$

Ainsi, il n'y a pas de différence entre la mAP et l'AP dans le cas de la détection d'une seule classe. À partir de l'ensemble de ces indicateurs, il est maintenant possible de tracer la courbe de la Précision en fonction du Rappel (PR) comme on peut le voir dans la figure 3.26. La Précision moyenne ou Average Precision (AP) du modèle alors définie comme étant l'aire de la zone située sous cette courbe [17]. La moyenne de la précision moyenne est obtenue (mean Average Precision, mAP) en faisant la moyenne de toutes les AP sur l'ensemble des classes recherchées par le modèle. Si plusieurs valeurs sont choisies pour le seuil de l'IoU, pour chaque classe, l'AP moyenne est calculée sur l'ensemble de ces valeurs seuils et la moyenne des valeurs obtenues donne le mAP.

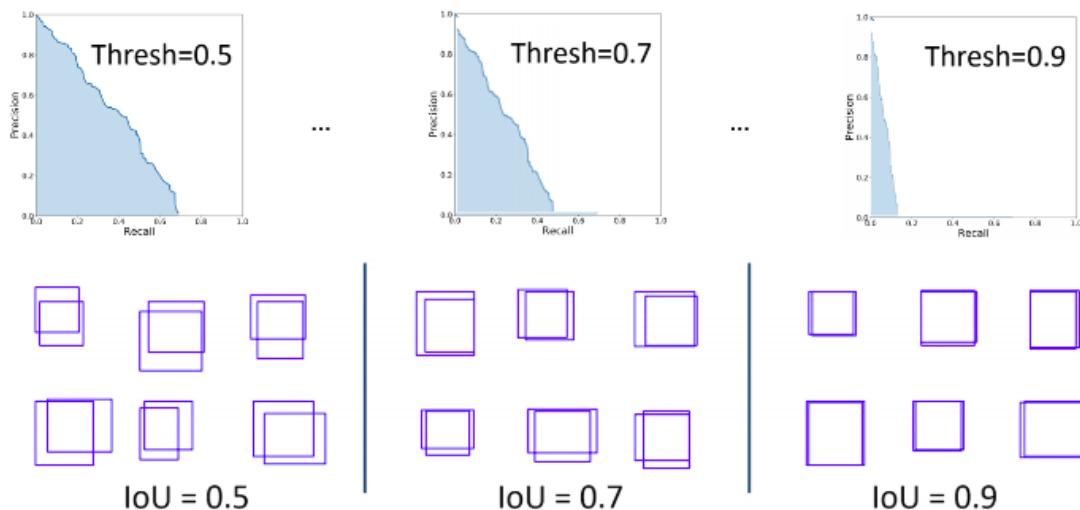


Figure 3.26: Courbe Rappel-Précision selon la valeur de l'IoU

Ainsi, comme nous pouvons le constater, plus le seuil de confiance est élevé, plus l'AP sera bas, mais nous serons plus confiants avec la précision.

### 3.3.3 Deep Learning framework

De nombreux frameworks ont été développés pour faciliter l'entraînement des réseaux de neurones profonds. Certains d'entre eux sont open-source. Les plus populaires sont :

#### 1. Tensorflow

Il s'agit d'un puissant framework d'apprentissage automatique et d'apprentissage profond publié en 2015 par Google. Il prend en charge Python et d'autres langages de programmation tels que C, C++, Java et Go.

#### 2. PyTorch

Développé par Facebook et NYU, PyTorch est un framework qui prend en charge les langages de programmation Python, C, C++ et Lua.

#### 3. Theano

Theano est un projet open source développé par MILA (Montreal Institute for Learning Algorithms) [2]. C'est une bibliothèque Python qui fournit à la fois les modes GPU et CPU.

Nous avons également des frameworks comme **Caffe**, **H2O**, **Néon**, **MXNET**, **CNTK**, etc.

En plus des frameworks mentionnés ci-dessus, il existe également des bibliothèques de haut niveau qui sont utilisées pour fournir une expérience plus universelle et un développement plus rapide. La bibliothèque **Keras** en est un bel exemple. Il est écrit en Python et prend en charge Tensorflow, CNTK et Theano.

Malgré la disponibilité de ces frameworks, sans les données, aucun apprentissage n'est réalisable. Les données constituent en réalité la matière première de toute tâche d'apprentissage automatique. Heureusement, il existe de nos jours de nombreux ensembles de données open source mise à la disposition des étudiants et chercheurs.

### 3.3.4 Quelques ensembles de données open-source pour le traitement d'images

**MNIST Digits** et **Fashion-MNIST**: sont des ensembles de données 70 000 images en noir et blanc de chiffres manuscrits, dont 60 000 images pour l'entraînement et 10 000 pour le test. Nous y avons 10 classes, une pour chaque chiffre de zéro à neuf.

**CIFAR-10** est composé de 10 classes mutuellement exclusives. Il y a 50 000 images de formation (5 000 par classe) et 10 000 images de test (1 000 par classe).

**PASCAL 07**: Il contient 9963 images de 20 catégories comprenant des personnes, des animaux et divers objets. Cet ensemble de données est considéré comme plus difficile que les ensembles de données MNIST, car les objets ne sont pas centrés et leurs apparences sont plus diverses.

**Pascal VOC** est un ensemble de données populaire de 20 classes. Il est utilisé pour la classification d'images, la détection d'objets et la segmentation.

Le **P-DESTRE**: c'est un ensemble de données multi-sessions de vidéos de piétons dans des environnements publics extérieurs, entièrement annotés. Il est utile pour des tâches de détection et de recherche de piétons.

**Open Images**: c'est un très grand ensemble de données constitué par Google. La version OpenImagesV6 compte environ 9 millions d'images annotées avec des étiquettes au niveau de l'image, des cadres de délimitation d'objets, des masques de segmentation d'objets, des relations visuelles et des récits localisés [1]. L'ensemble d'entraînement contient 41 620 images tandis que 125 436 images sont pour le test.

Le déploiement des modèles de réseaux neuronaux profonds pour une application dépend des performances et de la capacité de stockage du matériel hébergeant cette application. Ainsi, selon le type de déploiement, nous pouvons parler du Cloud Computing ou du Fog Computing.

### 3.3.5 Cloud Computing

Grâce à la technologie du "Cloud Computing", tout le traitement se fait dans le cloud, c'est-à-dire au niveau des serveurs distants ; après quoi, le résultat est renvoyé au dispositif IoT. Les applications bénéficient ainsi d'une bonne capacité de stockage et les ressources sont mises à disposition à tout moment et en tout lieu pour l'analyse à condition que la connectivité vers le cloud soit établie. Les plates-formes de cloud computing sont utiles pour créer des solutions IoT intelligentes, connectées et de bout en bout, car elles facilitent l'échange de données entre les appareils connectés.

### 3.3.6 Fog Computing

Le fog computing désigne un ensemble de techniques qui permettent de déployer des applications ou de stocker des données directement sur les dispositifs. Cela permet de réduire les frais généraux de communication, de décharger le trafic de données et de réduire de temps de réponse des applications. Certains cas d'utilisation de cette technologie se font au niveau des Smart phones et bien d'autres appareils portables. Comme les appareils ont des ressources informatiques et une puissance de batterie limitées, le fog computing nécessite du matériel et des logiciels spéciaux pour la mise en œuvre d'un modèle d'apprentissage approfondi.

# 4. Application : formation d'un modèle intelligent de détection de personnes

## 4.1 Énoncé du problème

De nos jours, on assiste à des cas d'enlèvement de personnes. Les maisons et sociétés sont souvent victimes d'attaques de la part de certains individus. La gestion des frontières devient de plus en plus une lourde tâche pour les forces de l'ordre et de sécurité. La recherche et le sauvetage des personnes après une catastrophe ou en zone difficile ne sont pas une tâche aisée pour les services de secours. Face à tous ces problèmes, un drone doté d'un système intelligent de détection de personnes peut être d'une grande utilité. Aussi, l'intégration de tels systèmes dans les caméras de surveillance pourrait renforcer la sécurité des locaux. Les modèles de détection sont également très utiles pour l'aide à la conduite et la technologie de véhicules autonomes. C'est ce qui nous motive à construire un modèle de détection de personne basé sur un réseau de neurones profond.

## 4.2 Collecte, préparation et nettoyage des données

C'est l'une des parties les plus difficiles et les plus coûteuses d'un projet d'apprentissage automatique et d'apprentissage profond. Heureusement, il existe actuellement des ensembles de données open source pour le bien des praticiens et apprenants en apprentissage automatique. Néanmoins, si l'on désire utiliser ses propres images, il peut utiliser un outil d'étiquetage d'image open source comme LabelImg, VGG Image Annotator, OpenLabeler ou ImgLab, etc. Cette tâche nécessite quand même plusieurs jours, voir plusieurs mois de travail. Tout dépendant de la taille des images à annoter. La figure 4.1 nous présente un exemple d'étiquetage avec l'outil LabelImg.

Pour ce qui est de notre travail, nous avons un ensemble de données constitué de 3 300 images dont 3 000 pour l'entraînement et 300 pour le test. Toutes ces images sont labellisées et ont été téléchargées de la plateforme Open Images V6. Le téléchargement a été facilité par l'outil OIDv4\_ToolKit développé par Vittorio [29]. Cet outil nous a permis de télécharger la taille d'images de notre choix de même que leurs labels en toute simplicité.

### 4.2.1 Choix du modèle

Pour la formation de nos modèles, nous avons utilisé l'apprentissage supervisé comme type d'apprentissage puisque nous n'utilisons que des données labellisées. Nous avons utilisé la méthode d'apprentissage par transfert. Celle-ci nécessite l'utilisation des poids de connexion de modèles pré-entraînés.

Il existe plusieurs modèles pré-entraînés pour la détection d'objets : SSD29, RCNN, Fast RCNN,

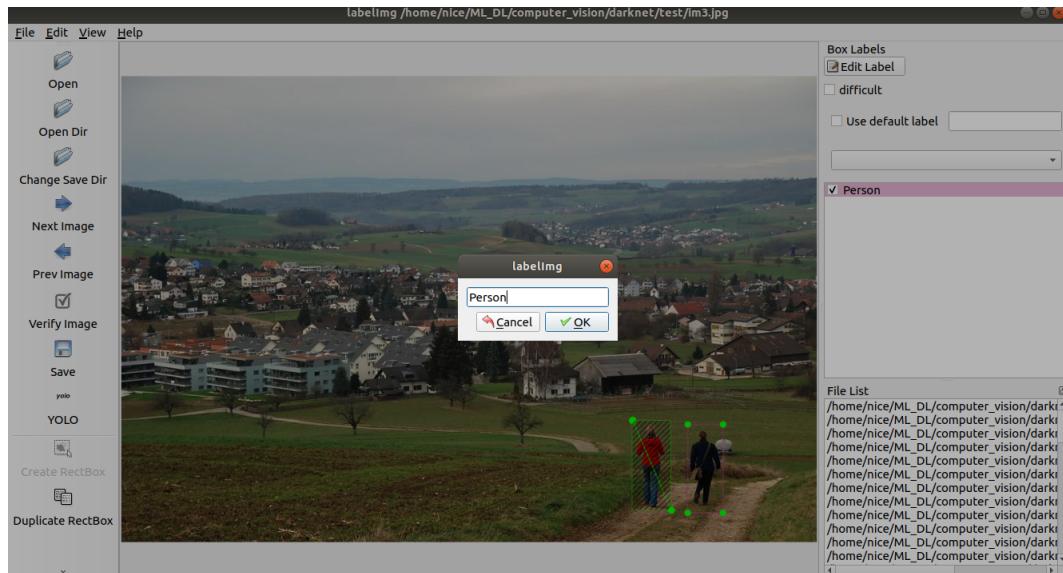


Figure 4.1: Etiquetage avec LabelImg

R-FCN, YOLO, Mask RCNN, etc. Des algorithmes comme Faster R-CNN et R-FCN sont des approches basées sur les régions. Dans un premier temps, ils cherchent toutes les régions qui contiennent les objets, puis transmettent ces régions à un classificateur, qui donne la classe des objets se trouvant dans ces régions. Tandis que le modèle YOLO (You Only Look Once) a une approche en une passe : il applique une seule fois le réseau de neurones à l'image toute entière puis prédit les boîtes englobantes et la probabilité qu'un objet détecté appartienne à une des classes [25]. Ce qui réduit considérablement la complexité du calcul au niveau des couches de convolution et accroît la vitesse de prédiction. Cette réduction de la complexité des calculs rend le modèle YOLO idéal pour les systèmes embarqués. De part sa précision, YOLO est classé dans le lot des top 5 des modèles de détection d'objet [9]. Ce sont d'ailleurs toutes ces deux caractéristiques (vitesse et précision) qui nous ont amenés à choisir ce modèle.

Les modèles pré-entraînés que nous avons utilisés sont YOLO v3 et YOLO v4 respectivement troisième et quatrième versions du modèle YOLO. Nous avons choisi ces deux dernières versions de YOLO parce qu'elles peuvent avoir de différents comportements sur notre ensemble de données. On pourra donc choisir la version nous donnant la meilleure performance.

## 4.2.2 Entrainement du modèle

Pour l'entraînement du modèle, il faut une machine ayant au moins 16 Go de GPU et 16 Go de RAM. Nous avons utilisé Google Colab, une plateforme fournissant des ressources de calcul GPU gratuites (12 heures de temps au plus) afin d'entraîner notre modèle. Malgré cette possibilité d'utiliser le GPU, la formation d'un très grand réseau de neurones profonds est parfois très coûteuse en temps.

Les hyperparamètres d'un modèle sont un aspect important à prendre en compte lors de l'entraînement de ce modèle. Ce sont des paramètres d'apprentissage généraux qui ne peuvent pas être automa-

tiquement appris à partir des données. Ils ont une grande influence sur l'apprentissage et la mise au point d'un modèle. Parmi les hyperparamètres utilisées pour l'entraînement des modèles, nous avons :

- **Batch** : c'est le nombre d'images qui passent par le réseau en une itération. Le batch peut également avoir un impact significatif sur les performances de votre modèle et le temps de formation. Pour notre modèle, il est fixé à 64.
- La **subdivision** : elle représente le nombre de mini-lots. Il est fixé à 32. Ainsi, à chaque itération, 64 images sont transmises au réseau en 32 mini-lots (chacun ayant 2 échantillons). Cette petite taille du mini-lot nous permet de diminuer les exigences de GPU et de la mémoire RAM.
- Le **max\_batches** est un paramètre très important qui influence la performance du modèle. Il détermine le nombre d'époques pour la formation du modèle.
- **Taux d'apprentissage** : le taux d'apprentissage ou **learning rate** est l'un des hyperparamètres qui affectent le plus les performances. C'est la vitesse à laquelle les paramètres des poids de connexion d'un modèle sont modifiés. Si nous ne pouvons ajuster qu'un seul hyperparamètre, alors le meilleur choix est le taux d'apprentissage.
- **burn\_in** : c'est le nombre d'itérations après lesquelles le taux d'apprentissage augmente lentement de 0 jusqu'à sa valeur finale (0,001 pour notre cas). Il permet de contrôler le taux d'apprentissage.
- **steps** : lots entre lesquels le taux d'apprentissage est ajusté pendant la formation.

Les valeurs des hyperparamètres que nous avons utilisées au cours de la formation de nos modèles sont récapitulés dans le tableau 4.1. Nous ne pouvons pas expérimenter toutes les valeurs possibles de tous ces hyperparamètres dans le cadre de ce travail compte tenu des contraintes liées à non-disponibilité du GPU de façon permanente et à la faible capacité de la RAM de notre machine (seulement 8 Go au lieu de 16 Go au moins).

## 1. Entraînement avec 2 200 images : 2 000 pour le training et 200 pour le test

Nous avons construit (02) modèles pour cet ensemble de données : un modèle basé sur version 3 de YOLO et un autre basé sur sa version 4. L'entraînement de chaque modèle a duré environ 72 heures.

## 2. Entraînement avec 3 300 images : 3 000 pour le training et 300 pour le test

Nous avons ensuite construit deux (02) autres modèles, cette fois-ci sur un dataset de 3 300 images, en utilisant toujours les versions 3 et 4 de YOLO. L'entraînement de chaque modèle a duré environ 96 heures.

Les valeurs des hyperparamètres que nous avons utilisées sont regroupées dans le tableau 4.1. Compte-tenu de la non-disponibilité d'un machine disposant des capacités sus-citées et de la

hyperparamètres	max_batches	batch	subdivision	learning rate	width x height	steps	burn_in
valeur	6000	64	16	0.001	416 x 416	4800, 5400	1000

Table 4.1: Hyperparamètres utilisés

limitation du temps d'utilisation du GPU de Google Colab, nous n'avons pas pu tester d'autres valeurs de ces hyperparamètres.

Passons à présent au test.

## 4.2.3 Evaluation des modèles et test

### 4.2.3.1 Evaluation

L'évaluation du modèle va permettre de lancer un ensemble de prédictions sur un lot d'images (non utilisées pour l'entraînement du modèle). Pour chaque image, la prédiction sera comparée à la vérité terrain contenue dans le fichier d'annotations. Si l'IoU est fixé à 0,5, alors l'AP correspondant est généralement noté AP@0.5 (ou AP@50%, ou parfois juste AP50).

On a construit des graphes qui nous renseignent sur la précision moyenne (AP) des différents modèles et leurs erreurs de prédiction. Ces graphes contiennent chacun deux courbes. Une courbe tracée au rouge : c'est la courbe AP-itérations. Cette courbe a pour avantage de nous permettre de voir à quelle itération avons-nous obtenu la meilleure précision. Ainsi, on pourra récupérer et utiliser les poids pour lesquels le meilleur mAP a été obtenu. La seconde courbe (en bleu) est la courbe average loss-itérations. Elle permet de voir l'évolution de la moyenne des erreurs au cours de l'entraînement. Elle a principalement pour but de contrôler le sur-apprentissage.

#### 1. Evaluation sur le dataset de 2 200 images

La figure 4.2a est celle basée sur YOLO v3. Nous pouvons constater que le meilleur résultat au cours de l'entraînement du modèle est obtenu à la 540<sup>e</sup> itération avec un AP50 = 22 %.

La figure 4.2b est celle basée sur YOLO v4. Le meilleur AP50 est 40 %. Il est obtenu à la 900<sup>e</sup> itération.

#### 2. Evaluation sur le dataset de 3 300 images

La figure 4.3a est celle basée sur YOLO v3. Nous pouvons constater que le meilleur résultat au cours de l'entraînement du modèle est obtenu à la 1800<sup>e</sup> itération avec un AP50 = 52 %.

La figure 4.3b est celle basée sur YOLO v4. Le meilleur AP50 est 88 %. Il est obtenu à la 1000<sup>e</sup> itération.

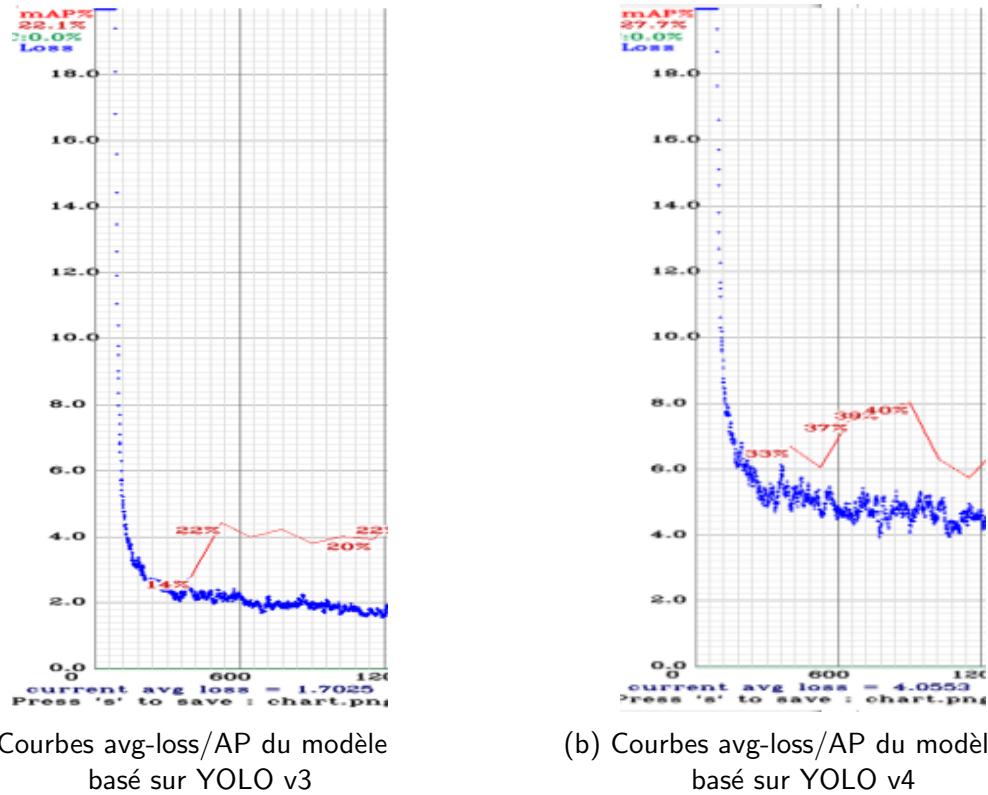


Figure 4.2

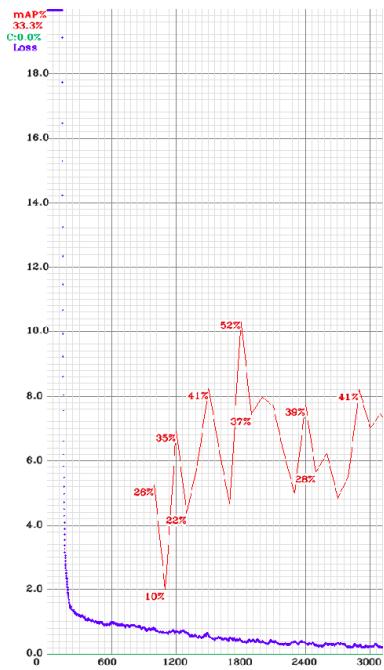
**NB :**

La limitation du temps d'utilisation du GPU de Google Colab nous contraint parfois à arrêter l'entraînement plus tôt que prévu. On a donc dû arrêter à 1 200 itérations au lieu de 6 000 itérations initialement prévues lors de l'entraînement sur le premier dataset (celui de 2 200 images). Les mêmes contraintes nous ont également amené à stopper l'entraînement sur le deuxième dataset après 3 000 itérations.

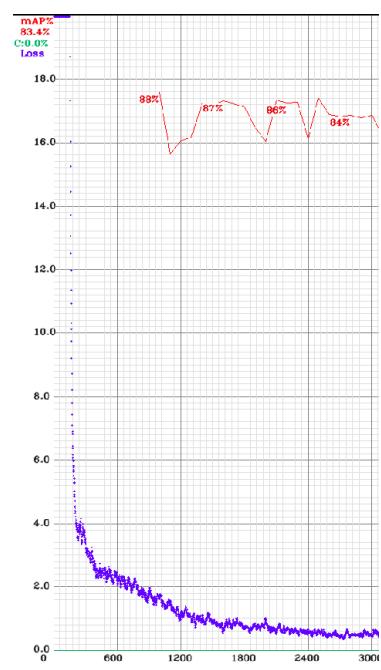
#### 4.2.3.2 Test du mode

Nous avons testé notre modèle avec les images 1.1a et 1.1b le modèle ayant donné la plus grande performance. Les résultats de ce test se trouvent dans la figure 4.4. Toutes les personnes se trouvant dans ces images ont été détectées et ceci avec une bonne précision.

Dans le cadre de notre étude, la meilleure performance en termes de précision moyenne (AP) est obtenue avec le modèle basé sur YOLO v4. Cette AP est égale à 88 % voir figure 4.3b. Elle est obtenue sur le dataset de 3 300 images.



(a) Courbes avg-loss/AP du modèle basé sur YOLO v3

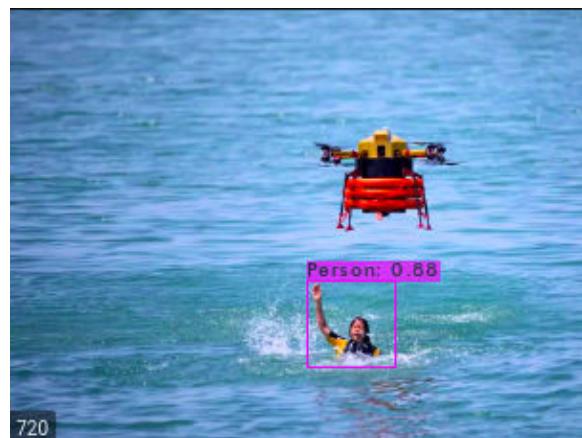


(b) Courbes avg-loss/AP du modèle basé sur YOLO v4

Figure 4.3



(a)



(b)

Figure 4.4: Le modèle détecté les personnes sinistrées

# 5. Conclusion et Perspectives

## 5.1 Conclusion

Ce travail nous a permis de passer en revue quelques architectures des réseaux de neurones profonds et d'utiliser la technique d'apprentissage supervisé et la méthode d'apprentissage par transfert.

Nous avons construit quatre modèles de détection de personnes. Nous avons eu de meilleurs résultats avec le modèle basé sur l'architecture YOLO v4. Ce résultat est obtenu sur un dataset de 3 300 images dont 3 000 pour l'entraînement et 300 pour le test.

Ce résultat n'est pas mal vu la petite taille des données d'entraînement utilisée. Les performances pourraient être augmentées avec l'augmentation des données d'entraînement puisque la précision d'un modèle de Deep Learning augmente sans doute avec la taille des données d'entraînement.

Le travail ne sait pas réaliser sans difficulté. Les difficultés que nous avons rencontrées sont essentiellement liées au matériel de travail : absence de GPU local, faible capacité de la RAM de notre machine et limitation du temps d'utilisation de la machine virtuelle Google Colab. Toutes ces difficultés ne nous ont pas permis d'explorer toutes possibilités pouvant nous permettre d'obtenir un modèle plus performant.

## 5.2 Contributions

Au cours de notre travail, nous avons fait les trois contributions. Nous avons pu énumérer de quelques ensembles de données open source utilisés pour les tâches de détection d'objet. Ces ensembles de données sont très utiles pour les apprenants dans le domaine, et même pour les chercheurs. De plus, ce travail nous a permis d'ouvrir une nouvelle brèche de recherche interdisciplinaire reliant la sécurité et l'apprentissage profond. Enfin, nous avons construit un modèle de détection de personne d'une performance de 88 % en termes d'Average Precision (AP).

## 5.3 Perspectives

Pour ce qui est des perspectives, notre but étant de construire un modèle spécialisé de détection de personnes, nos futurs travaux pourraient porter sur le meilleur choix des autres hyperparamètres et l'utilisation d'un ensemble de données plus volumineux.

# References

- [1] Open Images Dataset V6. <https://storage.googleapis.com/openimages/web/factsfigures.html>. Page consulted on Nov 28, 2020, Feb 2020.
- [2] Theano (software). [https://en.wikipedia.org/wiki/Theano\\_\(software\)](https://en.wikipedia.org/wiki/Theano_(software)). Page consulted on July 10, 2020, July 2020.
- [3] Ismaeel Abu Aballi. Internet of things. 10 2019. doi: 10.13140/RG.2.2.16678.47683.
- [4] actutem. Le marché de l'IoT promis à une fulgurante croissance jusqu'en 2026. <https://www.actutem.com/le-marche-de-liot-promis-a-une-fulgurante-croissance-jusquen-2026/>. Page consulted on Mar 17, 2020, Mar 2020.
- [5] Hong-Yuan Mark Liao Alexey Bochkovskiy, Chien-Yao Wang. Yolov4: Yolov4: Optimal speed and accuracy of object detection. [arXiv](#), 2020.
- [6] Md Zahangir Alom, Tarek M Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C Van Esen, Abdul A S Awwal, and Vijayan K Asari. The history began from alexnet: A comprehensive survey on deep learning approaches. [arXiv preprint arXiv:1803.01164](#), 2018.
- [7] Kevin Ashton et al. That ‘internet of things’ thing. [RFID journal](#), 22(7):97–114, 2009.
- [8] Chloé-Agathe Azencott. Empilez les perceptrons. <https://openclassrooms.com/fr/courses/4470406-utilisez-des-modeles-supervises-non-lineaires/4732186-empilez-les-perceptrons>. Page consulted on Nov 11, 2020, Jan 2020.
- [9] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. [arXiv preprint arXiv:2004.10934](#), 2020.
- [10] Ahmed Fawzy Gad, Ahmed Fawzy Gad, and Suresh John. [Practical computer vision applications using deep learning with CNNs](#). Springer, 2018.
- [11] Yves Grandmontagne. Qu'est-ce que l'IoT ? Définition de l'IoT, l'Internet des Objets. <https://itsocial.fr/enjeux-it/enjeux-tech/objets-connectes/quest-liot-definition-de-liot-linternet-objets/>. Page consulted on Fev 04, 2021, Jui 2017.
- [12] Aurélien Géron. [Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow](#). O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2e edition, 2019.
- [13] Jeff Heaton. [Artificial Intelligence for Humans, Volume 3: Neural Networks and Deep Learning](#). Heaton Research, Inc., 1.0 edition, 2015. ISBN 978-1505714340.
- [14] Wafa'a Kassab and Khalid A Darabkh. A-z survey of internet of things: Architectures, protocols, applications, recent advances, future directions and recommendations. [Journal of Network and Computer Applications](#), page 102663, 2020.

- [15] Nikhil Ketkar et al. Deep Learning with Python, volume 1. Springer, 2017.
- [16] Abhishek Khanna and Sanmeet Kaur. Evolution of internet of things (iot) and its significant impact in the field of precision agriculture. Computers and electronics in agriculture, 157: 218–231, 2019.
- [17] Harshit Kumar. Evaluation metrics for object detection and segmentation: mAP. <https://kharshit.github.io/blog/2019/09/20/evaluation-metrics-for-object-detection-and-segmentation>. Page consulted on Dec 03, 2020, Sep 2019.
- [18] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In Proceedings of 2010 IEEE international symposium on circuits and systems, pages 253–256. IEEE, 2010.
- [19] Jangwon Lee, Jingya Wang, David Crandall, Selma Šabanović, and Geoffrey Fox. Real-time, cloud-based object detection for unmanned aerial vehicles. In 2017 First IEEE International Conference on Robotic Computing (IRC), pages 36–43. IEEE, 2017.
- [20] Daphné Lercier Lucie Camanez. 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#:~:text=Union%20\(IoU\).-,Intersection%20Over%20Union%20\(IoU\),taille%20de%20l'objet%20recherch%C3%A9.](https://makina-corpus.com/blog/metier/2020/extraction-dobjets-pour-la-cartographie-par-deep-learning-evaluation-du-modele#:~:text=Union%20(IoU).-,Intersection%20Over%20Union%20(IoU),taille%20de%20l'objet%20recherch%C3%A9.) Page consulted on Dec 03, 2020, Jun 2020.
- [21] Racine Ly. Les fonctions d'activation - Part I. <https://www.racinely.com/post/les-fonctions-d-activation-part-i>. Page consulted on Nov 06, 2020, jan 2019.
- [22] Pierre Leroy Maxime Carrere. YOLOV4 Partie 1: à la pointe de la détection d'objet. <https://medium.com/scalian/comment-fonctionne-yolov4-la-vitesse-et-la-pr%C3%A9cision-maximale-en-d%C3%A9tection-dobjet-43724ab8786>. Page consulted on Dec 31, 2020, Oct 2020.
- [23] Ruihui Mu. A survey of recommender systems based on deep learning. IEEE Access, 6: 69009–69022, 2018.
- [24] Keyur K Patel, Sunil M Patel, et al. Internet of things-iot: definition, characteristics, architecture, enabling technologies, application & future challenges. International journal of engineering science and computing, 6(5), 2016.
- [25] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 779–788, 2016.
- [26] Oliver Struckmeier. Leagueai: Improving object detector performance and flexibility through automatically generated training data and domain randomization. arXiv preprint arXiv:1905.13546, 2019.

- [27] Priya Suresh, J Vijay Daniel, Velusamy Parthasarathy, and RH Aswathy. A state of the art review on the internet of things (iot) history, technology and fields of deployment. In 2014 International conference on science engineering and management research (ICSEMR), pages 1–8. IEEE, 2014.
- [28] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE, 105(12):2295–2329, 2017.
- [29] Angelo Vittorio. Toolkit to download and visualize single or multiple classes from the huge open images v4 dataset. [https://github.com/EscVM/OIDv4\\_ToolKit](https://github.com/EscVM/OIDv4_ToolKit), 2018.
- [30] Wikipédia. AlphaGo. <https://fr.wikipedia.org/wiki/AlphaGo>. Page consulted on Dec 29, 2020.
- [31] Qi Yue and Caiwen Ma. Deep learning for hyperspectral data classification through exponential momentum deep convolution neural networks. Journal of Sensors, 2016, 2016.
- [32] Qianru Zhang, Meng Zhang, Tinghuan Chen, Zhifei Sun, Yuzhe Ma, and Bei Yu. Recent advances in convolutional neural network acceleration. Neurocomputing, 323:37–51, 2019.